**Общие сведения**

Глобальная сеть состоит из множества связанных между собой сетей, которые функционируют как единое целое. Ключевым преимуществом глобальной сети Internet является то, что она обеспечивает универсальный способ взаимодействия систем, в каждой из которых используется различное аппаратное обеспечение, наилучшим образом удовлетворяющие потребности разработчиков.

Ниже мы опишем принципы, лежащие в основе межсетевого обмена данными, и подробности функционирования одного из семейств протоколов, использующихся для этих целей. Предметом нашего рассмотрения является технология TCP/IP (ее название происходит от названия двух основных протоколов), которая была разработана *Управлением перспективного планирования научно-исследовательских работ (ARPA)*. Она лежит в основе сети Internet — огромной глобальной сети, связавшей воедино учебные, научные, коммерческие и государственные организации во многих странах мира. В настоящее время происходит бурный рост Internet.

* [Необходимость интеграции сетей](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690453)
* [Глобальная сеть TCP/IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690454)
* [Службы Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690455)
* [Службы Internet уровня приложений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690456)
* [Службы Internet сетевого уровня](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690457)
* [История развития Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690458)
* [Архитектурный совет Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690459)
* [Реорганизация архитектурного совета Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045a)
* [Сообщество Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045b)
* [Запросы на комментарии в Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045c)
* [Протоколы Internet и стандартизация](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045d)
* [Перспективы роста и технология](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045e)
* [Проверка знаний: Общие сведения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69045f)

**Необходимость интеграции сетей**

Обмен информацией через Internet стал неотъемлемой частью нашей жизни. В World Wide Web можно найти самую разную информацию, например прогноз погоды, данные об урожае прошлого года, курсы акций и расписание полетов самолетов. Группы добровольцев создают электронные списки рассылки, позволяющие распространять информацию, которая относится к сфере их интересов. Партнеры по бизнесу и коллеги по работе обмениваются друг с другом электронными сообщениями, а родственники посылают друг другу электронные поздравления.

К несчастью, так сложилось, что большинство сетевых технологий было разработано под конкретные нужды. При создании сети каждое предприятие выбирает необходимое оборудование с учетом объема и необходимой скорости передачи данных, а также стоимости этого оборудования. Однако на основе одной сетевой технологии невозможно создать универсальную сеть, поскольку никакая отдельная сетевая технология, взятая сама по себе, не может подходить для всех случаев жизни. Одним требуется создать высокоскоростную сеть, связывающую компьютеры в пределах здания; однако дешевые сетевые технологии, удовлетворяющие конкретным нуждам, нельзя применять в случае огромных по протяженности территорий. Другие готовы мириться с низкой скоростью работы сети, лишь бы только она связывала компьютеры, находящиеся на расстоянии в несколько тысяч километров.

За прошедшие два десятилетия были разработаны новые технологии, с помощью которых стало возможным соединить между собой разнородные физические сети и заставить работать эту систему как единую скоординированную сеть. Технология *объединения сетей (internetworking)* заставляет работать множество разнородных аппаратных средств благодаря обеспечению способов подключения гетерогенных сетей и набору коммуникационных протоколов, которые делают возможным это взаимодействие. Технология Internet (или технология глобальных сетей) позволяет скрыть от конечного пользователя особенности применяемого аппаратного обеспечения и позволяет компьютерам обмениваться информацией независимо от того, к какой физической сети они подключены.

Технология глобальных сетей, описанная в этом курсе, является отличным примером *взаимодействия открытых систем (open system interconnection)*. Они названы *открытыми*, потому что спецификация на них полностью открыта и доступна для использования всеми желающими. Это в корне отличает их от *закрытых (proprietary)* коммуникационных систем, стандарты на которые являются правом собственности конкретного производителя.

Таким образом, любой разработчик может создать программу, которая будет обмениваться данными с другими программами через Internet. Однако самое важное то, что была разработана единая технология для обмена данными между компьютерами, функционирующими на различных аппаратных платформах, использующих практически любое оборудование для сетей с коммутацией пакетов, поддерживающая широкий спектр программного обеспечения и большое количество типов операционных систем.

Чтобы по достоинству оценить технологию глобальных сетей, задумайтесь над тем, насколько кардинально она изменила производственную деятельность человека, особенно сферу бизнеса. Уже никого не удивляет возможность высокоскоростного обмена данными по локальной сети предприятия. Технологии глобальных сетей сделали возможной оперативную обратную связь между производственной сферой, отделом продаж и маркетинга, а также конечными потребителями. В результате существенно выросла скорость планирования, производства, реализации и переоборудования производства. Причем изменения эти поистине разительные!

**Глобальная сеть TCP/IP**

Потенциальные возможности и важность сетевых технологий в США осознали много лет тому назад. Государственный департамент выделил средства на исследования, благодаря которым впоследствии была создана Internet.

В этом курсе описываются принципы и идеи, лежащие в основе технологии глобальных сетей, которые были получены в результате исследований, финансируемых *Управлением перспективного планирования научно-исследовательских работ (Advanced Research Projects Agency, или ARPA)1*. В основу технологии ARPA положен набор сетевых стандартов, в которых оговорены детали взаимодействия компьютеров, а также соглашения по объединению сетей и методы маршрутизации трафика. Официально все это было названо *семейством протоколов TCP/IP (TCP/IP Internet Protocol Suite)*, однако чаще всего используется название *протокол TCP/IP* (по имени двух основных стандартов семейства). Этот протокол может использоваться для обмена информацией между двумя компьютерами, проходящей через любое количество объединенных сетей. Например, в некоторых крупных компаниях протокол TCP/IP применяется во всех связанных между собой корпоративных сетях, даже если эти сети не имеют выхода во внешний мир. Другие же используют этот протокол для обмена информацией между представительствами, разбросанными по всему Земному шару.

Хотя технология применения TCP/IP заслуживает внимания сама по себе, особый интерес к ней проявляется в крупномасштабных проектах, поскольку она прошла проверку временем. Благодаря этой технологии стало возможным создать глобальную сеть Internet, которая сегодня объединяет свыше 170 млн пользователей в разных странах. Причем все эти люди могут находиться как дома, так и в школе, в офисе или научной лаборатории. В США в финансировании Internet принимали участие такие организации, как *Национальный научный фонд (National Science Foundation*, или *NSF*), *министерство энергетики (Department of Energy*, или *DOE*), *министерство обороны (Department of Defense*, или *DOD*), *министерство здравоохранения и социальной защиты (Health and Human Services Agency*, или *HHS*) и *национальное агентство по аэронавтике и космическим исследованиям (National Aeronautics and Space Agency*, или *NASA*). Во всех этих организациях протокол TCP/IP используется для связи их исследовательских центров. В результате проведенных исследований была создана глобальная коммуникационная система, которую назвали *ARPA/NSF Internet, TCP/IP Internet, глобальная сеть Internet*, или просто *Internet2*. Благодаря ей абоненты, находящиеся в разных уголках Земного шара, обмениваются информацией так же легко, словно они находятся в соседних комнатах. Громадный успех Internet в первую очередь обусловлен высокой устойчивостью семейства протоколов TCP/IP к изменениям и его способностью адаптироваться к основным сетевым технологиям.

Большую часть материала курса можно применить к любой сети на основе протокола TCP/IP, однако в некоторых разделах описаны особенности глобальной сети Internet. Поэтому читатели, интересующиеся какой-либо одной технологией, должны быть внимательны и понимать различия (там, где они есть) между существующей структурой глобальной Internet и локальными сетями TCP/IP, построенными по общим правилам. Однако будет глубокой ошибкой пропустить разделы курса, посвященные глобальной сети, поскольку структура внутренних сетей многих крупных корпораций зачастую во много раз сложнее Internet двадцатилетней давности. Поэтому многие проблемы, с которыми вам предстоит столкнуться, наверняка уже были решены раньше при разработке глобальной сети.

1. ARPA позже была переименована в DARPA (Defense Advanced Research Projects Agency), или Управление перспективного планирования научно-исследовательских работ при Министерстве обороны США.

2. В литературе часто можно встретить названия Internet и internet. Когда речь идет о глобальной сети, пишут Internet, а если имеется в виду локальная сеть предприятия, которая построена на основе технологии TCP/IP, пишут internet. Чтобы не путать два близких по написанию слова, часто вместо internet пишут intranet.

**Службы Internet**

Понять технические подробности работы протокола TCP/IP невозможно без знакомства со службами, реализованными в Internet. Далее мы рассмотрим эти службы очень кратко. Особое внимание будет уделено тем из них, к которым чаще всего обращаются пользователи. В следующих разделах будет описано, как компьютеры подключаются к сети TCP/IP и как реализуется нужная для служб функциональность.

При обсуждении служб речь в основном пойдет о стандартах, которые называются *протоколами (protocols)*. Протокол, такой как TCP или IP, представляет собой набор синтаксических и семантических правил, использующихся при обмене данными между двумя компьютерами. В нем оговаривается формат блоков сообщений, описывается реакция компьютера на получение определенного типа сообщения и указываются способы обработки ошибок и других необычных ситуаций. И что самое важное, благодаря протоколам, мы можем описать процесс обмена данными между компьютерами, не привязываясь к какой-то определенной компьютерной платформе или сетевому оборудованию конкретного производителя. В некотором смысле можно сказать, что при передаче данных протокол является тем же самым, чем алгоритм при выполнении вычислений. Как и протокол, алгоритм позволяет описать или понять процесс вычислений без привязки к системе команд конкретного процессора. Точно так же коммуникационный протокол позволяет описать или понять процесс обмена данными, не вдаваясь в детали функционирования сетевого оборудования конкретного производителя.

Скрытие низкоуровневых особенностей процесса передачи данных способствует повышению производительности труда разработчиков.

Во-первых, поскольку программистам приходится иметь дело с протоколами, относящимися к достаточно высокому уровню абстракции, им не нужно держать в голове (и даже изучать!) технические подробности используемого аппаратного обеспечения. Поэтому новую программу можно написать достаточно быстро.

Во-вторых, поскольку программы разрабатываются на основе модели, относящейся к высокому уровню абстракции, который не зависит от конкретной архитектуры компьютера или типа сетевого оборудования, в них не нужно вносить никаких изменений при переходе на другой тип оборудования или изменении конфигурации сети.

В-третьих, поскольку в прикладных программах используются протоколы высокого уровня, которые опять же не зависят от применяемого оборудования, они (программы) могут обеспечивать непосредственное соединение между двумя любыми компьютерами сети. При этом программистам не нужно создавать специальные версии программ для каждого типа используемого компьютера или сети. Программы разрабатываются с использованием универсальных протоколов. Поэтому один и тот же код можно скомпилировать и запустить на любом компьютере.

Ниже мы увидим, что подробности функционирования служб Internet оговорены в отдельных протоколах. В следующем разделе будут рассмотрены несколько протоколов, относящихся к сетевым службам Internet уровня приложений, а также к службам, формирующим сетевой уровень. В последующих разделах каждый из этих протоколов будет рассмотрен во всех подробностях.

**Службы Internet уровня приложений**

С точки зрения конечного пользователя, Internet состоит из набора прикладных программ, которые используют ресурсы существующей сети для выполнения полезной работы по обмену информацией. Для обозначения возможности кооперации разнородных систем при решении вычислительных проблем мы будем использовать специальный термин *степень взаимодействия (interoperability)*. Так вот, выражаясь научным языком, приложения для Internet имеют высокую степень взаимодействия. Большинство пользователей для доступа к Internet просто запускают одну из прикладных программ. При этом они даже не задумываются над тем, к компьютерам какого типа происходит обращение, как работает технология TCP/IP, какова структура сетей и по каким маршрутам проходят данные. Они во всем полагаются на прикладную программу и обслуживающее ее сетевое программное обеспечение, в которых все эти детали учитываются. Подробности работы сети на основе протокола TCP/IP и понимание происходящих при этом процессов необходимы только программистам, разрабатывающим сетевое программное обеспечение.

К наиболее популярным и широко распространенным службам Internet можно отнести те, что перечислены ниже.

* *World Wide Web*. Эта служба позволяет клиентам просматривать документы, содержащие текст и графику, и перемещаться по гиперссылкам от одного документа к другому. Бурный рост Web произошел в начале 1995 года. Как раз на это время приходится резкое увеличение потока данных этой службы по глобальной сети Internet. Служба Web не сдает своих позиций и в настоящее время. Некоторые из поставщиков услуг (провайдеров) Internet отмечают, что на Web приходится примерно 80% их общего трафика.
* *Электронная почта (e-mail)*. Программы электронной почты позволяют создавать текстовые документы и рассылать их копии одному или нескольким адресатам. Полученные сообщения можно прочитать с помощью тех же почтовых программ. Введение новых технологий позволяет включать в почтовые сообщения файлы любого типа. Развитие системы электронной почты было столь успешным, что теперь большинство пользователей Internet не представляют себе, как можно обходиться без электронных сообщений. Одна из причин огромной популярности службы электронной почты Internet состоит в хорошо продуманном проекте ее реализации: забота по доставке сообщений возложена на сам протокол. В рамках протокола оговорены не только способы установки прямой связи между почтовыми серверами отправителя и получателя, но и указывается, что сообщение не может быть удалено с компьютера отправителя до тех пор, пока получатель не поместит его копию в надежное для хранения место.
* *Пересылка файлов*. Как следует из названия, эта служба позволяет обмениваться данными путем пересылки файлов. Служба пересылки файлов — одна из старейших в Internet, однако она продолжает интенсивно использоваться до сих пор. И хотя файлы небольших размеров можно отправить в виде вложения по электронной почте, без службы пересылки файлов не обойтись при передаче больших файлов. В этой службе предусмотрены способы авторизации пользователей, а также возможность запретить доступ всем пользователям. Пересылка файлов через сеть TCP/IP так же надежна, как и передача электронной почты, поскольку два компьютера непосредственно устанавливают соединение между собой, без участия третьего компьютера, выполняющего промежуточное хранение данных.
* *Удаленный доступ к системе*. Эта служба позволяет подключиться к удаленной машине и установить с ней интерактивный сеанс связи. При этом на локальном компьютере пользователя появляется окно, в котором отображается информация, выводимая удаленным компьютером в ответ на ввод команд с клавиатуры локального компьютера. Как только сеанс связи с удаленной машиной завершается, управление возвращается операционной системе локального компьютера.

Работа этих и других служб Internet будет детально описана в следующих разделах. В них мы более конкретно расскажем, как в перечисленных службах используется семейство протоколов TCP/IP и почему применение протоколов уровня приложений гарантирует полную универсальность программ.

**Службы Internet сетевого уровня**

Программисты, разрабатывающие приложения с использованием протоколов TCP/IP, имеют совершенно другие представления о сети по сравнению с пользователями, которые только запускают прикладные программы типа почтового клиента. На сетевом уровне в семействе протоколов TCP/IP предусмотрено два обширных класса служб, которые используются во всех приложениях. И хотя на данном этапе мы не будем вдаваться в подробности их функционирования, ни одно описание протоколов TCP/IP без них не обходится.

* *Служба доставки пакетов, не требующая установки соединения (Connectionless Packet Delivery Service)*. Эта служба лежит в основе всех остальных сетевых служб (ее подробное описание будет приведено в следующих разделах). Она относится к абстрактным типам служб и реализована практически во всех сетях с коммутацией пакетов. В основе работы сети TCP/IP лежит принцип пересылки небольших сообщений (пакетов) от одного компьютера к другому, выполняющийся с учетом маршрутной информации, которая находится в самом сообщении. Поскольку служба, не требующая установки соединения, пересылает каждый пакет независимо друг от друга, она не может гарантировать их надежную доставку, а также доставку в заданном порядке. Однако, так как эта служба обычно тесно привязана к сетевому аппаратному обеспечению, эффективность ее работы чрезвычайно высока. И что самое важное, поскольку служба доставки пакетов, не требующая установки соединения, лежит в основе всех сетевых служб, это позволяет адаптировать семейство протоколов TCP/IP практически к любому типу сетевого оборудования.
* *Надежная потоковая транспортная служба (Reliable Stream Transport Service)*. Для работы большинства приложений недостаточно только одной службы, не гарантирующей доставку пакетов. Причина в том, что прикладные программы часто требуют от сетевого программного обеспечения автоматического восстановления данных при возникновении ошибок при передаче, потери пакетов или отказов одного из звеньев сети между отправителем и получателем. Надежная потоковая транспортная служба как раз занимается решением этих проблем. Она позволяет приложению, запущенному на одном компьютере, установить “соединение” с приложением, запущенным на другом компьютере, а затем переслать большой объем данных, как если бы эти два компьютера имели постоянный прямой канал связи между собой. Естественно, что при этом сетевые протоколы нижнего уровня разбивают поток данных на небольшие сообщения и передают их последовательно друг за другом, ожидая от получателя подтверждения успешного приема.

Службы, аналогичные описанным выше, реализованы практически во всех типах сетей, поэтому интересно рассмотреть, чем же отличаются службы сети TCP/IP. Ниже перечислены только основные отличия.

* *Независимость от сетевой технологии*. Несмотря на то что в семействе протоколов TCP/IP используется традиционная технология коммутации пакетов, она не зависит от применяемого сетевого оборудования. В глобальной сети Internet используется множество сетевых технологий, причем не все из них изначально были ориентированы для работы на больших расстояниях. В семействе протоколов TCP/IP определяется единица передачи информации, которая называется *дейтаграммой (datagram)*, и оговариваются методы передачи дейтаграмм по отдельным сетям.
* *Всеобщность подключения*. Сеть TCP/IP позволяет обмениваться информацией любым двум подключенным к ней компьютерам. Каждому компьютеру назначается адрес, который интерпретируется по одним и тем же правилам во всей сети. В любой дейтаграмме присутствует как адрес отправителя, так и адрес получателя пакета. Адрес получателя используется промежуточными звеньями сети для правильной маршрутизации этой дейтаграммы.
* *Подтверждение получения пакетов*. В сетях на основе протокола TCP/IP оговаривается процедура подтверждения получения пакетов. Конечный получатель посылает уведомление отправителю даже в том случае, если они не подключены к одной физической сети. При этом промежуточные звенья сети, через которые проходят пакеты, уведомления о получении отправителю не шлют.
* *Стандарты протоколов прикладных программ*. Кроме основных служб транспортного уровня, рассмотренных выше, в семейство протоколов TCP/IP включены стандарты для широко распространенных прикладных служб, таких как электронная почта, пересылка файлов и удаленный вход в систему. Поэтому, разрабатывая прикладные программы с использованием протоколов TCP/IP, программисты часто обнаруживают, что нужные им сетевые компоненты уже созданы.

В следующих разделах будут описаны службы, предназначенные для использования сетевыми программистами, а также стандарты протоколов уровня приложений.

**История развития Internet**

Несомненно, что основная причина столь высокой популярности и привлекательности технологии TCP/IP связана с ее универсальностью, а также с быстрым ростом глобальной сети Internet. В середине 1970-х годов по инициативе ARPA были начаты работы над новой перспективной технологией, которые к концу 1970-х принесли свои плоды — были сформированы структура и набор основных протоколов. В тот период организация ARPA выступала в роли основного источника финансирования в области исследования сетей с коммутацией пакетов. Оно внедрило множество новейших идей в известном проекте сети ARPANET. Хотя сеть ARPANET была построена на традиционных двухточечных проводных выделенных линиях, ARPA финансировала исследования технологии передачи пакетов данных по радиоканалу и каналам спутниковой связи. Наличие множества сетевых технологий передачи данных заставило ARPA заняться проблемой межсетевого взаимодействия, что в свою очередь подхлестнуло развитие глобальной сети.

Доступность результатов исследований, финансируемых ARPA, привлекла внимание нескольких групп разработчиков и вызвала их профессиональный интерес. Особенно это касается тех инженеров, кто уже имел определенный опыт в технологии коммутации пакетов, использовавшейся в проекте ARPANET. Поэтому ARPA стала периодически проводить неформальные конференции разработчиков, на которых они могли обменяться идеями и обсудить результаты экспериментов. Это “сборище” стали неформально называть *группой исследователей Internet (Internet Research Group*). Однако уже к 1979 году в исследование технологии TCP/IP было вовлечено такое количество людей, что ARPA было вынуждено создать неформальный комитет для координации их действий и руководства работами в области создания протоколов и структуры зарождающейся Internet. Эту организацию решили назвать *Советом по управлению и структуре Internet (Internet Control and Configuration Board, или ICCB)*. Заседания Совета регулярно проходили вплоть до 1983 года, после чего он был реорганизован.

История глобальной сети Internet начинается примерно с 1980 года, когда ARPA стало переводить компьютеры, подключенные к своим исследовательским сетям, на использование нового семейства протоколов TCP/IP. Модернизацию, естественно, начали с ARPANET, поэтому она быстро стала основой зарождающейся Internet, ее *магистральным каналом (backbone)*, который в дальнейшем стали использовать для экспериментов с создававшимся семейством протоколов TCP/IP. Повсеместный переход на новую технологию Internet был завершен в январе 1983 года, когда штаб-квартира Министерства обороны США объявила, что во всех компьютерах министерства, которые подключены к сетям, протянувшимся на большие расстояния, используется протокол TCP/IP. Тогда же управление связи Министерства обороны США (*Defense Communication Agency*, или *DCA*) разделило сеть ARPANET на две независимые сети. Одна была отведена для нужд науки, а вторая — для военных целей. За исследовательской сетью было сохранено название ARPANET, а военную сеть (которая была несколько больше) стали называть *MILNET (military network)*.

Чтобы заинтересовать университетскую общественность в использовании новых протоколов, ARPA стало продавать их программную реализацию по низкой цене. В то же время на большинстве компьютерных кафедр университетов использовалась версия операционной системы UNIX, разработанная в Калифорнийском университете. Она называлась *Berkeley Software Distribution*, но чаще всего ее называли *Berkeley UNIX*, или просто *BSD UNIX*. На средства ARPA корпорация *Bolt Beranek and Newman, Incorporated (BBN)* выполнила программную реализацию семейства протоколов TCP/IP для системы UNIX, а разработчики из Беркли включили их в дистрибутив своей операционной системы.

В результате ARPA удалось внедрить свое детище более чем в 90% университетских компьютерных лабораторий и факультетов. Новая программная реализация протоколов TCP/IP появилась в очень подходящее время, поскольку на большинстве факультетов имелось, как правило, несколько компьютеров и велись работы по их объединению в локальную сеть. Вот где пригодились новые коммуникационные протоколы!

Версия BSD UNIX оказалась чрезвычайно популярной еще и потому, что помимо основных протоколов TCP/IP в ней было реализовано много полезных вещей. Разработчики из Беркли предложили набор утилит для реализации сетевых служб, которые напоминали службы UNIX, использовавшиеся на локальной машине. Основным отличием этих утилит было то, что они удовлетворяли принятым стандартам UNIX. Поэтому, опытному пользователю UNIX не составляло большого труда понять, как работает, например, утилита удаленного копирования файлов rep, разработанная в Беркли, поскольку она функционировала аналогично команде копирования файлов системы UNIX.

Кроме набора сетевых утилит, в версии Berkeley UNIX было впервые введено новое абстрактное понятие *сокета (socket)*, которое позволяло прикладным программам обращаться к коммуникационным протоколам. По сути сокет являлся обобщением стандартного механизма ввода-вывода системы UNIX и, кроме протокола TCP/IP, позволял работать еще с несколькими сетевыми протоколами.

Структура системы сокетов вызвала полемику у специалистов после ее принятия, поэтому многие разработчики операционных систем придумали им замену. Но, независимо от достоинств и недостатков, введение в операционную систему абстрактного уровня сокетов было значительным достижением, поскольку они позволяли программистам легко пользоваться семейством протоколов TCP/IP. Все это подогрело интерес разработчиков к выполнению экспериментов с семейством протоколов TCP/IP.

Успешное использование технологии TCP/IP и Internet ведущими исследовательскими центрами привело к ее быстрому распространению. Понимая важность перспективы исследований в области сетевых коммуникаций, *Национальный научный фонд США (National Science Foundation,* или *NSF)* поставил задачу охватить технологией TCP/IP и Internet как можно больше научных учреждений. С этой целью в конце 1970-х годов NSF финансировал проект сети *CSNET (Computer Science NETwork)*, которая и связала все научные компьютерные лаборатории. В 1985 году NSF начал программу по обеспечению доступа к сети, объединившую его шесть суперкомпьютерных центров. В 1986 году NSF расширяет исследования в области сетевых технологий и выделяет средства на создание проекта новой глобальной магистральной сети, которую назвали *NSFNET1*. В конечном итоге эта сеть связала все суперкомпьютерные центры США и объединила их с ARPANET. В том же 1986 году NSF делает начальные инвестиции во множество проектов региональных сетей, каждая из которых связала основные научно-исследовательские учреждения в своем регионе. Во всех региональных сетях, финансируемых NSF, использовалось семейство протоколов TCP/IP, и все они стали частью глобальной сети Internet.

В течение семи лет с момента появления к Internet были подключены сотни частных сетей в США и Западной Европе. Общее количество компьютеров, объединенных в глобальную сеть, составило порядка 20 тыс. В основном их пользователи работали в учебных, государственных и корпоративных научных учреждениях. При этом масштабы роста и использования Internet значительно превысили все прогнозы. Исследования, проведенные в конце 1987 года, показали, что темпы роста Internet составляют 15% в месяц. К началу 2000 года глобальная сеть Internet охватывала более 50 млн компьютеров в 209 странах мира.

Выбор семейства протоколов TCP/IP на заре развития Internet и последовавший за этим грандиозный его рост не были обусловлены только проектами, финансируемыми государством. К Internet подключались как основные компьютерные гиганты, так и много других больших корпораций — нефтяные компании, автомобильные заводы, электронные фирмы, фармацевтические и телекоммуникационные компании. Средние и мелкие предприятия стали подключаться к Internet начиная с 1990 года. Кроме того, во внутренних сетях многих организаций использовалось семейство протоколов TCP/IP, даже если они не были подключены к Internet.

Темпы расширения Internet вызвали проблемы, непредвиденные на этапе проектирования. Это заставило разработчиков выработать технологию управления больших распределенных систем. Например, на этапе проектирования имена и адреса всех компьютеров, подключенных к Internet, хранились в одном файле, который редактировался вручную и после этого рассылался всем абонентам, подключенным к сети. Однако уже к середине 1980-х годов стало очевидно, что подобный подход централизованного хранения информации неэффективен.

Во-первых, поскольку к Internet подключалось все больше и больше компьютеров, запросы на обновление этого файла стали превышать физические возможности человека, который их обрабатывал.

Во-вторых, даже если предположить, что с файлом центральной базы данных не возникало бы никаких проблем, существующих в то время сетевых мощностей было явно недостаточно для частой рассылки информации об обновлении на каждый компьютер или предоставлению оперативного доступа к этой информации каждому абоненту сети.

Поэтому были разработаны новые протоколы, и по всей глобальной сети Internet запущена специальная служба системы имен, которая позволяла любому пользователю автоматически преобразовывать имя удаленного компьютера в его адрес. Эту службу назвали *системой доменных имен (Domain Name System*, или *DNS*). Ее основу составляли специальные компьютеры, называвшиеся *серверами имен (name servers)*, в задачи которых входила обработка запросов клиентов на преобразование имен. При этом ни на одном из серверов не хранилась база данных доменных имен целиком — данные были распределены между группами серверов, которые использовали семейство протоколов TCP/IP для связи между собой при выполнении запроса на преобразование имени.

1. Термин NSFNET иногда используется в более широком смысле и означает всю программу исследований в области сети, финансируемую фондом NSP. Однако в этой книге мы будем употреблять его в узком смысле, понимая при этом глобальный магистральный канал. В следующих разделах эта технология будет описана более подробно.

**Архитектурный совет Internet**

Поскольку семейство протоколов TCP/IP не было разработано конкретным производителем или признанным сообществом профессионалов, естественно, возникает вопрос: “Кто осуществляет техническое руководство и решает, когда протокол должен стать стандартом?” Ответ на этот вопрос таков: существует специальная группа лиц, которая называется *архитектурным советом Internet (Internet Architecture Board*, или *IAB1*. IAB направляет усилия разработчиков на решение конкретных проблем, координирует исследования и разработку семейства протоколов TCP/IP, а также руководит процессом роста Internet. IAB решает, какие протоколы нужно включить в семейство TCP/IP, и проводит официальную политику.

IAB был основан в 1983 году после того, как ARPA реорганизовала Совет по управлению и структуре Internet (ICCB). Он стал правопреемником всех ранних групп, разрабатывавших стандарты. Основными целями IAB были: всестороння поддержка обмена мнениями и идеями между разработчиками, вовлеченными в процесс исследования технологии TCP/IP и Internet; сосредоточение усилий разработчиков на решении основных проблем. За первые шесть лет IAB прошел путь от исследовательской группы при ARPA до самостоятельной организации. За это время каждый член IAB возглавил *специальную группу по решению конкретной задачи Internet (Internet Task Force*, или *ITF)*. Каждая такая группа занималась конкретной проблемой или рядом важных проблем. IAB выделил около десяти направлений. Круг его интересов был широк: от исследования такой глобальной проблемы, как влияние трафика различных приложений на функционирование Internet, до решения текущих технических задач. Архитектурный совет Internet проводил свои заседания несколько раз в год. На них заслушивались отчеты каждой из групп, анализировались и пересматривались технические инструкции, обсуждалась текущая политика и проводился обмен информацией с представителями таких организаций как ARPA и NSF, финансировавшими работы по Internet и исследования в области сетевых технологий.

Председателю архитектурного совета присваивалось звание *Главного архитектора Internet (Internet Architect)*. Он отвечал за принятие технических решений, а также координировал работу различных специальных групп IETF. Кроме того, на заседаниях совета IAB председатель ставил новые задачи специальным группам IETF, а также представлял IAB в других организациях.

Новички часто удивляются, когда узнают о том, что IAB не располагает большим бюджетом. Принимая основные решения, IAB не финансирует большинство исследований и инженерных разработок. Большую часть работы выполняют добровольцы. Каждый из членов IAB отвечает за привлечение добровольцев для решения поставленной задачи, созыв и проведение собрания специальной группы, а также за составление отчета для IAB. Как правило, добровольцы работают в научно-исследовательских или коммерческих организациях, занимающихся разработкой или использованием технологии TCP/IP. Активные исследователи принимают участие в заседаниях специальных групп IETF по двум причинам. С одной стороны, это позволяет им ознакомиться с новыми исследовательскими задачами и планами. С другой стороны, поскольку участники специальных групп занимаются разработкой новых идей, решением возникающих проблем и проведением экспериментов, которые впоследствии становятся частью технологии TCP/IP, применяемой в Internet, они осознают, что результаты их работы оказывают непосредственное и положительное влияние на развитие системы в целом.

1. IAB первоначально называлась как Internet Activities Board (Координационный совет Internet).

**Реорганизация архитектурного совета Internet**

К лету 1989 года масштабы использования технологии TCP/IP и Internet вышли далеко за рамки, которые были предусмотрены первоначальным исследовательским проектом. Они активно стали применяться в производственной сфере, в которую ежедневно были вовлечены десятки тысяч людей, причем деятельность этих людей непосредственно зависела от новой технологии. Поэтому стало невозможным отрабатывать новые идеи на реально работающем оборудовании, изменяя по ночам его конфигурацию. В большей степени от этого пострадали бы сотни коммерческих компаний, которые выпускали продукты, использовавшие технологию TCP/IP, поскольку разработчики определяли степень операционной совместимости своих продуктов путем внесения изменений в их программный код. Исследователи, создавая черновые варианты спецификаций стандартов и отрабатывая новые идеи, не могли больше рассчитывать на быстрое признание и внедрение результатов их работы. По иронии судьбы те, кто разрабатывал и внедрял технологию TCP/IP, оказались под влиянием коммерческого успеха созданного ими детища. Короче говоря, технология TCP/IP имела колоссальный коммерческий успех, и интересы рынка стали превалировать над интересами науки.

Реакцией на изменение политических и коммерческих реалий, связанных с семейством протоколов TCP/IP и самой Internet, стала реорганизация IAB летом 1989 года. Были изменены обязанности председателя совета. Исследователи были выведены из состава IAB, и из них была образована отдельная дочерняя группа. Состав нового совета IAB расширился, в него были введены представители от многочисленной сетевой общественности. На рис. 1.1 показана организационная структура нового совета IAB и взаимосвязи между его дочерними группами.

Рис. 1.1. Структура архитектурного совета Internet (IAB) после реорганизации, проведенной в 1989 году

Как видно из рис. 1.1, кроме правления, в структуру IAB входит еще две большие специальные группы: *исследовательская (Internet Research Task Force*, или *IRTF)* и *инженерная (Internet Engineering Task Force*, или *IETF)*.

Как следует из названия, инженерная группа IETF занимается решением краткосрочных или среднесрочных технических проблем. Эта группа существовала в первоначальной структуре IAB, и достигнутые ею успехи в определенной степени послужили причиной реорганизации Совета в целом. В отличие от большинства специальных групп IAB, состоящих из нескольких человек, которые решали одну конкретную задачу, до реорганизации инженерная группа IETF была очень многочисленна, в нее входили десятки активных сотрудников, работающих одновременно над несколькими проблемами. Поэтому ее разделили на 20 *рабочих групп (working groups)*, каждая из которых должна заниматься решением конкретной задачи. Для поиска методов решения задачи рабочие группы периодически проводят совещания. Кроме того, регулярно проводится общее собрание инженерной группы IETF, на котором заслушиваются отчеты рабочих групп и обсуждаются предложения или дополнения, которые нужно внести в стандарты протоколов семейства TCP/IP. Обычно проводится три собрания в год, в которых участвуют сотни людей — как участников инженерной группы IETF, так и приглашенных. Группа IETF стала настолько большой, что ею уже трудно управлять одному человеку.

Поскольку репутация группы IETF в Internet была чрезвычайно высока, а также потому, что собрания группы привлекали огромное внимание широких масс сетевой общественности, при реорганизации IAB было решено сохранить IETF в целом, но поделить ее примерно на десять подразделений, возглавляемых отдельным руководителем. Председатель инженерной группы IETF и руководители подразделений образовали так называемую *руководящую инженерную группу Internet (Internet Engineering Steering Group,* или *IESG)*, которая должна координировать действия рабочих групп IETF. В результате аббревиатура IETF стала относиться ко всей структуре инженерного совета в целом, включая председателя, руководителей подразделений и всех членов рабочих групп.

Во время реорганизации IAB была образована специальная *исследовательская группа*, которую назвали *Internet Research Task Force (IRTF)*. В ее задачи входит координация исследований, связанных с семейством протоколов TCP/IP и со структурой Internet в целом. Как и IETF, в IRTF создана небольшая *руководящая исследовательская группа (Internet Research Steering Group*, или *IRSG)*, в задачи которой входит расстановка приоритетов и координация направлений проводимых исследований. В отличие от IETF, группа IRTF малочисленной и ее члены проявляют гораздо меньшую активность. Фактически основная часть исследований проводится инженерной группой IETF.

**Сообщество Internet**

В 1992 году Internet, фактически, вышел из под контроля своего опекуна — правительства США. Поэтому, чтобы поддержать дух партнерства было создано специальное *сообщество Internet (Internet Society*, или *ISOC*). Эта международная организация создана по инициативе *Национального географического общества* США. Сообщество Internet выступает в роли головной организации для IAB и служит делу объединения людей во всем мире посредством использования Internet.

**Запросы на комментарии в Internet**

Выше мы уже говорили о том, что у технологии TCP/IP нет конкретного разработчика, за ней не стоит какое-либо признанное профессиональное сообщество, отдельный человек или группа лиц. В результате конечные пользователи не могут получить описание протоколов, стандартов и правила приемлемого использования сети непосредственно от производителя. Поэтому вся документация помещена в специальное хранилище, доступ к которому в оперативном режиме (online) открыт для всех пользователей Internet совершенно бесплатно.

Вся документация, относящаяся к работе Internet, планы введения новых и пересмотра старых протоколов, а также полное описание семейства протоколов TCP/IP хранится в виде набора технических статей, которые называются *запросами на комментарии в Internet (Requests For Comments*, или *RFC)*.

Статьи RFC могут иметь объем от нескольких до сотен страниц в зависимости от того, что в них описано, — общие концепции или детали работы конкретного протокола, существующий стандарт или только предложение по его введению. И хотя документы RFC не относятся к научным академическим трудам, информация в них тщательно выверяется и редактируется. Этой работой много лет занимался покойный ныне Джон Постел1. Сейчас задача редактирования документов RFC возложена на руководителей подразделений инженерной группы IETF. Утверждением новых документов RFC занимается руководящая инженерная группа Internet IESG в полном составе.

Кроме RFC, на заре развития Internet издавалась параллельная серия документов под названием *технические заметки об Internet (Internet Engineering Notes*, или *IEN*). И хотя IEN больше не выходят, тем не менее не вся содержащаяся в них информация была перенесена в документы RFC. Поэтому в тексте курса, помимо ссылок на RFC, вы столкнетесь со ссылками на документы IEN.

Документы RFC последовательно нумеруются в хронологическом порядке, соответствующем времени их написания. Каждому вновь созданному или пересмотренному документу назначается новый номер. Поэтому при поиске нужной информации будьте внимательны и используйте версию документа с наивысшим номером. В этом вам поможет полный список документов RFC, в котором перечислены их номера и названия.

Чтобы сделать документы RFC доступными широкой общественности, их копии размещены на многих серверах в Internet. Копии документов можно получить также по обычной почте, электронной почте или загрузить их из Internet с помощью FTP. Кроме того, существуют еще предварительные версии документов RFC, которые называются *рабочими документами Internet (Internet drafts)*. Чтобы получить доступ к документам RFC или черновикам Internet в вашем регионе, обратитесь к местному специалисту по сетям.

1. Джон умер осенью 1998 года. Он был одним из тех, кто внес значительный вклад в развитие Internet и технологии TCP/IP. Для тех, кто знал его лично, — это большая утрата.

**Протоколы Internet и стандартизация**

Искушенный в вопросах сетевой технологии читатель наверняка знает, что существует огромное количество стандартов сетевых протоколов. Причем многие из них были созданы еще до появления Internet. Поэтому возникает закономерный вопрос: “Зачем создатели Internet стали придумывать новые протоколы, если на тот момент существовало большое количество международных стандартов протоколов?” Однозначно ответить на этот вопрос сложно, но в приведенном ниже абзаце, определенно, есть здравый смысл.

Существующие стандарты протоколов следует использовать там, где это возможно. Новые протоколы создаются только в том случае, когда нельзя обойтись существующими стандартами. Однако, как только новые стандарты, обеспечивающие нужную функциональность будут приняты, их нужно сразу же использовать.

Итак, при создании семейства протоколов TCP/IP не преследовалась цель проигнорировать или отменить существующие стандарты, хотя внешне это может выглядеть именно так. Разработки велись, поскольку ни один из существующих стандартов не позволял взаимодействовать между собой сетевым коммуникационным системам.

**Перспективы роста и технология**

Развитие технологии TCP/IP и Internet продолжается. Время от времени предлагаются к принятию новые протоколы и пересматриваются старые стандарты. В свое время фонд NSF создал сложную коммуникационную систему, предложив архитектуру с магистральным каналом, к которому подключились региональные сети и сотни сетей учебных заведений. Сегодня к Internet продолжают подключаться тысячи различных организаций, разбросанных по всему миру. Однако наиболее существенные изменения в Internet происходят не от подключения дополнительных сетей, а от роста трафика. По мере подключения новых пользователей в Internet увеличивается количество запущенных приложений, что в свою очередь приводит к изменению общей картины потока данных. Например, после того как к Internet подключились физики, химики и биологи, они стали обмениваться друг с другом данными экспериментов. По сравнению с сообщениями электронной почты файлы с ними имели просто гигантские размеры. По мере роста популярности Internet пользователи стали использовать для получения информации новые службы, наподобие *World Wide Web*, что привело к еще большему увеличению трафика.

Чтобы справиться с возрастающим трафиком, пропускную способность магистрального канала NSFNET увеличивали уже три раза. Его последний вариант называют *ANSNET* (по имени модернизировавшей его компании). Пропускная способность этого канала в 840 раз больше первоначальной! Начиная с 1995 года многие коммерческие компании, называемые *поставщиками услуг Internet (Internet Service Provider*, или *ISP*), создают собственные магистральные каналы, причем пропускная способность многих из них существенно выше, чем у магистрального канала, финансируемого государством (даже после последней модернизации). В настоящее время трудно предсказать, когда прекратится наращивание мощности этих каналов.

Возросшая потребность в межсетевом обмене не является неожиданностью. Компьютерная индустрия постоянно нуждается в увеличении мощности вычислительных устройств и расширении хранилищ данных, способных вместить информацию, накопленную за много лет. Пользователи только сейчас начинают понимать, как следует пользоваться сетевой технологией. Поэтому нетрудно предсказать, что в будущем человечество будет испытывать постоянную потребность в увеличении мощности средств связи. Например, уже сейчас технология TCP/IP используется для передачи голоса и видеоизображения (а не только традиционных данных). Таким образом, современные высокоемкие коммуникационные технологии требуют расширения мощности каналов связи.

**Проверка знаний: Общие сведения**

**Задание 1**

Какую степень взаимодействия имеют приложения для Internet?

высокую

не взаимодействуют

низкую

**Задание 2**

Кто является разработчиком TCP/IP?

организация ARPA

архитектурный совет Internet

нет конкретного разработчика

**Задание 3**

Благодаря какому протоколу, в первую очередь, обусловлен громадный успех Internet?

TCP/IP

IPX/SPX

ARPA/NSF

**Задание 4**

Чем занимается архитектурный совет Интернет?

осуществляет техническое руководство и решает проблемы связанные с протоколами TCP/IP

контролирует работу интернета

построением архитектуры для протоколов TCP/IP

**Задание 5**

Завершите фразу корректно

Интернет зарождался в

годах 20 века.

**Задание 6**

Почему возникла необходимость интеграции сетей?

возникла необходимость создать высокоскоростную сеть

иерархия сетей не позволяла продолжать развитие сетевых технологий

большинство сетевых технологий было разработано под конкретные нужды

**Задание 7**

В чем одна из причин реорганизации архитектурного совета Internet?

относительно низкая популярность в использовании протоколов TCP/IP

гибель главы совета

интересы рынка стали превалировать над интересами науки

**Сеть Ethernet**

В начале 1970-х годов в исследовательском центре Пало Альто корпорации Хегох была разработана технология передачи данных по локальным сетям с коммутацией пакетов, которая впоследствии стала одной из самых популярных. Ее назвали Ethernet. В 1978 году фирмы Хеrох Corporation, Intel Corporation и Didital Equipment Corporation приняли эту сетевую технологию в качестве стандарта. Чуть позже Институт инженеров по электротехнике и электронике Institute of Electrical and Electronics Engineers (IEEE) выпустил совместимую версию стандарта и присвоил ей номер 802.3.

* [Основные понятия](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046a)
* [Тонкий Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046b)
* [Сеть Ethernet на основе витой пары](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046c)
* [Пропускная способность сети Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046d)
* [Технология быстрого Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046e)
* [Технология 10/100 Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69046f)
* [Гигабитовый Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690470)
* [Отличительные особенности технологии Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690471)
* [Адресация оборудования в сетях Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690472)
* [Формат фрейма Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690473)
* [Проверка знаний: Сеть Ethernet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690474)

**Основные понятия**

Ethernet стала одной из самых популярных технологий передачи данных по локальным сетям. В настоящее время она применяется практически во всех корпоративных сетях и в подавляющем большинстве небольших локальных сетей. Поскольку технология Ethernet пользуется огромной популярностью, было выпущено несколько ее вариантов. И хотя первоначальная схема подключения компьютеров к сети постепенно вытесняется новыми, более совершенными схемами, ее понимание поможет нам разобраться в принципах работы сети и уяснить некоторые важные конструкторские решения. Поэтому сначала мы рассмотрим первоначальную схему подключения компьютеров к сети, а затем обсудим ее варианты. Официально оригинальная технология Ethernet называется *10ВASE5*. Для подключения абонентов сети в ней применяется толстый *коаксиальный кабель*, поперечное сечение которого показано на рисунке 2.1.

Рис. 2.1. Поперечное сечение коаксиального кабеля

Кабель называется *средой передачи данных* или *эфиром (ether)* и является полностью пассивным элементом. Все активные электронные компоненты, необходимые для функционирования сети, сосредоточены в платах сетевого интерфейса, установленных в компьютерах. Диаметр кабеля Ethernet составляет 12,7 мм (0,5 “), а длина сегмента может достигать 500 м. Для того чтобы предотвратить отражение радиосигнала на концах кабеля, между центральной жилой и экраном подключают согласующие резисторы, сопротивление которых равно волновому сопротивлению кабеля.

Для подключения компьютера к оригинальному кабелю Ethernet требовалось специальное устройство, называемое *трансивером*, или приемопередатчиком. Физически подключение трансивера ко внутренней жиле коаксиального кабеля Ethernet осуществлялось через небольшое отверстие, проделанное в наружной оболочке. Специалисты для такого типа подключения часто используют термин отвод (*tap*). Обычно в трансивере устанавливали небольшие металлические штыри, с помощью которых осуществлялся электрический контакт с центральной жилой и металлической оплеткой экрана. Некоторые производители предлагали для подключения трансивера использовать специальные Т-образные ответвители. При этом кабель разрезался на небольшие сегменты, и на его концах устанавливались коаксиальные разъемы.

Для подключения компьютера к оригинальному кабелю Ethernet требовалось два основных электронных компонента: *трансивер* и *плата сетевого интерфейса*.

С помощью трансивера компьютер подключался к центральной жиле и экрану коаксиального кабеля. В его функции входили прием и отправка сигналов в передающую среду. Плата сетевого интерфейса, или сетевой адаптер, подключался непосредственно к системной шине компьютера (например, вставлялся в один из свободных слотов расширения материнской платы) и трансиверу.

Трансивер представлял собой небольшое электронное устройство, обычно располагавшееся в непосредственной близости от кабеля. Кроме аналоговой части, которая принимала и отправляла электрические сигналы в кабель, в состав трансивера входил цифровой блок, с помощью которого выполнялся обмен данными с компьютером. Трансивер улавливал аналоговые электрические сигналы, проходящие по кабелю, выделял из них цифровые данные и отправлял их в компьютер, а также выполнял обратное преобразование цифровых данных, полученных от компьютера в аналоговые и отправлял их в эфир.

Трансивер подключался к сетевому адаптеру компьютера с помощью специального интерфейсного кабеля подключаемых устройств, или *АUI-кабеля* (attachment unit interface). На техническом языке АUI-кабель назывался кабелем трансивера. Он состоял из ряда проводников, по которым проходили электрические сигналы управления трансивером, пакеты передаваемых и принимаемых данных, а также подавалось напряжение питания, необходимое для работы трансивера.

Каждый сетевой адаптер управлял одним трансивером согласно командам, получаемым от программного обеспечения компьютера. С точки зрения операционной системы, сетевой интерфейс представлял собой устройство ввода-вывода, на которое от компьютера поступали команды передачи данных и сами данные. Плата сетевого интерфейса передавала эти данные трансиверу и далее в эфир, генерировала сигнал прерывания процессору после выполнения операции и сообщала о состоянии сетевого устройства. По сравнению с трансивером, представлявшим собой довольно простое устройство, плата сетевого интерфейса была существенно сложнее. В некоторых платах интерфейса использовался встроенный микропроцессор, который управлял обменом информацией между памятью компьютера и эфиром.

На практике, в организациях, где использовалась оригинальная технология Ethernet для офисных применений, коаксиальный кабель прокладывался под потолком помещений. В местах, где располагались компьютеры, организовывались точки подключения к общему кабелю.

**Тонкий Ethernet**

Оригинальная технология Ethernet имела существенные недостатки. Например, поскольку трансивер являлся электронным устройством, он имел определенную отличную от нуля стоимость, которая умножалась на количество компьютеров, подключенных к сети. Более того, поскольку трансиверы располагались в непосредственной близости от кабеля, а не от компьютеров, задача поиска и замены таких устройств с случае выхода их из строя была не из легких. Кроме того, прокладка коаксиального кабеля также была затруднена. Дело в том, что для защиты кабеля от внешних электромагнитных помех, возникающих при работе электрических устройств, таких как, например, электродвигатели, экран коаксиального кабеля изготавливали достаточно толстым и плотным, что затрудняло его изгиб при прокладке. Ну и, наконец, АUI-кабель также был достаточно толстым и трудно поддавался укладке.

Поэтому, чтобы уменьшить стоимость оборудования, необходимого для создания локальных сетей, работающих при небольшом уровне электромагнитных помех (например, в офисе), был разработай альтернативный вариант подключения к сети Ethernet. Официально его назвали 10ВASE2, но чаще всего специалисты называли новую технологию подключения *тонким Ethernet* или *сетью Ethernet с тонким коаксиальным кабелем*. Альтернативный коаксиальный кабель был тоньше, дешевле и более гибкий, хотя и не лишен недостатков. Поскольку тонкий коаксиальный кабель не обеспечивал эффективной защиты от электромагнитных помех, его нельзя было прокладывать рядом с мощным электрическим оборудованием, которое используется в промышленных целях. Кроме того, по сравнению с толстым Ethernet тонкий кабель обеспечивал гораздо меньшую дальность связи и позволял подключать к локальной сети меньшее количество компьютеров.

Рис. 2.2. Тонкий коаксиальный кабель

При разработке новой технологии Ethernet достаточно дорогостоящий трансивер был заменен специальными высокоскоростными цифровыми схемами, которые располагались прямо на плате сетевого интерфейса. Это позволило подключать компьютеры напрямую к коаксиальному кабелю. Таким образом, при использовании технологии тонкого Ethernet в компьютере находились и плата сетевого интерфейса и специальное устройство, позволяющее непосредственно подключаться к коаксиальному кабелю. Производители небольших компьютеров и рабочих станций сочли технологию тонкого Ethernet очень привлекательной, поскольку она позволяла легко встраивать сетевое оборудование в компьютеры на базе микропроцессоров. При этом сетевой разъем располагался на задней стенке системного блока компьютера.

Поскольку компьютеры подключались непосредственно к тонкому коаксиальному кабелю, монтаж сети облегчался, когда в одной комнате находились много компьютеров. Тогда достаточно было протянуть кабель от одного компьютера к другому, а чтобы подключить к сети новый компьютер, достаточно было добавить к цепочке новое звено.

Технология тонкого Ethernet была разработана для того, чтобы облегчить подключение и отключение компьютеров от сети. При этом использовались коаксиальные разъемы, фиксирующиеся поворотом замка на 90 , так называемые *ВNС-разъемы*. Для подключения кабеля к компьютеру не требовались специальные устройства. Таким образом, с этой работой самостоятельно мог справиться любой пользователь, не прибегая к помощи специалиста.

Конечно, в этом есть и определенный недостаток, поскольку, если пользователь случайно разъединял кабель, вся сеть полностью выходила из строя. Однако во многих случаях преимущества технологии тонкого Ethernet перевешивали ее недостатки.

**Сеть Ethernet на основе витой пары**

Современные технологические достижения сделали возможным создание такой сети Ethernet, где не требуется экранировать сигнальные провода внешним электромагнитным экраном, как в коаксиальном кабеле. Новую технологию назвали *Ethernet на основе витой пары1, (twisted pair Ethernet)*. Она позволяла использовать в качестве эфира обычные неэкранированные медные провода, наподобие тех, что применяются в телефонии. Сеть на основе витой пары обладает двумя важными преимуществами, по сравнению с тонким Ethernet: она дешевле и защищена от случайных повреждений сети при отключении одной машины или повреждении сетевого кабеля, ведущего к компьютеру. В некоторых случаях технология витой пары позволяет создать сеть Ethernet на основе существующей в организации проводки. Кроме того, прокладка витой пары (ее называют *кабелем 5-й категории*) обходится дешевле и проще, чем коаксиального кабеля.

Официально технология на основе витой пары называется *10Base-T*. Первые сети с витой парой работали на скорости 10 Мбит/с, как и сети с толстым или тонким Ethernet. Подключение компьютеров к сети Ethernet на основе витой пары осуществляется через специальное устройство, которое называется *концентратором*, или *хабом (hub)*, как показано на рис. 2.3. При этом используется восьмижильный (четырехпарный) кабель.

Рис. 2.3. Схема сети Ethernet на основе витой пары. Каждый компьютер подключается к концентратору с помощью четырехпарного кабеля

Концентратор — это электронное устройство, которое имитирует сигналы в кабеле Ethernet. Он представляет собой небольшую коробку, которая обычно размещается в монтажном шкафу или распределительной коробке. Длина соединительного провода между компьютером и концентратором не должна превышать 100 м. Для работы концентратора требуется источник питания. Кроме того, некоторые концентраторы позволяют обслуживающему персоналу следить за их работой и состоянием прямо по сети. С точки зрения платы сетевого интерфейса, подключение к концентратору аналогично подключению к трансиверу.

Другими словами, концентратор обеспечивает такие же возможности в плане передачи данных, как и толстый или тонкий Ethernet. Просто с помощью концентратора реализуется альтернативная схема подключения компьютеров к сети.

1. Термин “витая пара” возник не случайно; он взят из телефонии, где сигнальные провода скручивали для уменьшения перекрестных помех между соседними каналами связи.

**Пропускная способность сети Ethernet**

За время своего существования большая часть элементов оригинальной технологии Ethernet не претерпела глубоких изменений, несмотря на то, что схема подключения компьютеров к сети изменялась при переходе от оригинального толстого кабеля Ethernet к тонкому, а затем — к витой паре. Об этом свидетельствует и то, что первоначально скорость обмена данными по витой паре была такой же, как и у толстого кабеля Ethernet (10 Мбит/с). И хотя современные компьютеры без особых проблем могут генерировать такой поток данных, истинной скоростью работы сети нельзя считать ту, с которой два компьютера могут обмениваться данными между собой.

Скорость работы сети должна оцениваться в зависимости от ее пропускной способности. Представьте себе, что сеть — это автомагистраль, соединяющая между собой несколько городов (компьютеров), а пакеты — движущиеся по ней автомобили. Так вот, высокая пропускная способность сети означает, что по шоссе может одновременно двигаться большой поток машин. Низкая пропускная способность говорит о том, что автомагистраль не может справиться с существующим потоком автотранспорта. Например, сеть Ethernet с пропускной способностью 10 Мбит/с может одновременно обслуживать либо несколько компьютеров, генерирующих большой поток данных, либо множество компьютеров, создающих небольшой трафик в сети.

В конце 1970-х годов, когда проводилась стандартизация технологии Ethernet, пропускная способность локальной сети, равная 10 Мбит/с, была более чем достаточна для обслуживания множества компьютеров, поскольку существующие на то время скорости работы центральных процессоров компьютеров и плат интерфейса не позволяли обмениваться данными с большими скоростями. Однако к середине 1990-х годов производительность микропроцессоров существенно выросла, что отразилось на интенсивности использования сетей. В результате сеть Ethernet с пропускной способностью 10 Мбит/с уже не могла использоваться в качестве магистральной не то что в крупных, а даже в средних корпорациях, поскольку она не справлялась с резко возросшим потоком данных. Таким образом, узким местом в технологии Ethernet стала ее пропускная способность.

**Технология быстрого Ethernet**

Чтобы увеличить пропускную способность сети Ethernet, инженеры разработали новую версию этого стандарта, в которой скорость обмена данными была увеличена на порядок. Официально новую технологию назвали *100BASE-T*, но чаще всего ее стали называть *быстрым Ethernet (Fast Ethernet)*. Как следует из официального названия, в технологии быстрого Ethernet использовался кабель витой пары 5-й категории, как и в технологии 10Base-T. Однако благодаря особому режиму использования проводников кабеля, удалось повысить скорость обмена данными до 100 Мбит/с.

Чтобы понять, как удалось увеличить пропускную способность сети на порядок, необходимо иметь в виду два момента. Во-первых, несмотря на то, что скорость работы компьютеров существенно возросла, передавать непрерывный поток данных со скоростью 100 Мбит/с могут только очень немногие компьютерные системы. И, во-вторых, стандарт 100BASE-T никак не повлиял на большую часть компонентов технологии Ethernet. В частности, величина максимального размера пакета осталась такой же, как и в технологии 10BASE-T. Все это свидетельствует о том, что технология быстрого Ethernet не оптимизирована для достижения максимально возможной скорости обмена между двумя абонентами сети. Она рассчитана на охват большего количества рабочих станций и большой общий трафик в сети.

**Технология 10/100 Ethernet**

Вскоре после появления технологии быстрого Ethernet производители стали выпускать комбинированные платы сетевого интерфейса, поддерживающие скорости передачи данных как 10, так и 100 Мбит/с. Новую технологию назвали *двухскоростным Ethernet (dual-speed Ethernet)*, или *10/100 Ethernet*. Кроме сетевых плат, ее стали использовать и в концентраторах. В этом нет ничего необычного, поскольку в сущности все сетевые платы стандарта 100BASE-T передавали дополнительные сигналы, благодаря которым оборудование, установленное на другом конце провода могло определить тип сетевой платы, от которой пришел пакет. Более того, если на концах кабеля в разъемах RJ-45 задействованы все восемь проводов, такой кабель можно использовать для подключения как устройств 10BASE-T, так и 100BASE-T.

И хотя оборудование, поддерживающее стандарт 10/100 Ethernet, было немного дороже, чем оборудование стандарта 10BASE-T, оно стало очень популярным. Двухскоростные сетевые платы оказались весьма кстати при модернизации сети и плавном переходе на 100-мегабитовую технологию. Например, если компьютер с двухскоростной сетевой платой стандарта 10/100 Ethernet подключить к концентратору 10BASE-T, специальная схема в плате автоматически определит возможную скорость передачи данных и перейдет на работу в 10-мегабитовом режиме. Если же впоследствии компьютер будет подключен к концентратору 100BASE-T, то оборудование опять же автоматически определит новую скорость передачи данных и перейдет в 100-мегабитовый режим. Следует заметить, что изменение скорости передачи данных происходит полностью автоматически; при этом не нужно перенастраивать какое бы то ни было программное или аппаратное обеспечение.

**Гигабитовый Ethernet**

К концу 1990-х годов, по мере распространения технологии 100BASE-T, стало очевидным, что в отдельных случаях пропускной способности канала связи в 100 Мбит/с явно недостаточно. Поэтому технология Ethernet была доработана, в результате чего скорость передачи данных возросла до 1 Гбит/с (миллиардов бит в секунду). Ее назвали *1000BASE-T*. Новая технология стала особенно привлекательной для создания корпоративных магистральных сетей, по которым проходит большой поток данных, поступающий от множества компьютеров. Однако при повышении скорости передачи данных снижается помехоустойчивость канала связи. Гигабитовый Ethernet оказался слишком восприимчивым к электромагнитным помехам. Поэтому кабель, без проблем работавший на скоростях 10 и даже 100 Мбит/с, не годился для технологии 1000BASE-T.

Подобно быстрому Ethernet, технология гигабитового Ethernet оптимизирована под высокую пропускную способность сети. Поэтому были сохранены первоначальные форматы пакетов и их максимальные размеры. Таким образом, пакеты данных, проходящие по сетям 10BASE-T, 100BASE-T и 1000BASE-T, совместимы друг с другом. Следовательно, можно сконцентрировать потоки данных, проходящие по десяти загруженным сетям 100BASE-T, в один поток и передать его по одной сети 1000BASE-T.

**Отличительные особенности технологии Ethernet**

При разработке технологии Ethernet ставилось несколько задач: она должна быть построена по принципу общей шины и поддерживать широковещательный режим передачи данных, обеспечивать негарантированную доставку пакетов (best-effort) и распределенное управление доступом. В сети Ethernet используется топология, которая называется *общей шиной*, поскольку все рабочие станции подключаются к одному совместно используемому каналу связи. Кроме того, сеть Ethernet построена по принципу *широковещательной передачи данных (broadcast technology)*, т.е. абоненты сети получают все посланные в эфир пакеты. Это позволяет передать пакет сразу всем абонентам сети.

О том, как переслать пакет от одной рабочей станции к другой или к их группе, речь пойдет чуть ниже. А пока достаточно понять, что на самом нижнем уровне в сети Ethernet находится оборудование, которое не различает адресатов в передаваемых пакетах. Другими словами, от концентратора пакеты поступают на все подключенные к нему сетевые платы, а плата уже “решает”, какие пакеты должен принять компьютер, в котором она установлена.

В технологии Ethernet не предпринимаются специальные усилия для доставки пакетов. Поскольку сетевому оборудованию отправителя не передается информация о доставке пакета получателю, нельзя гарантировать, что пакет будет доставлен получателю без проблем. Типична, например, ситуация, когда компьютер получателя выключен. При этом все посланные ему пакеты теряются, а отправитель об этом даже не подозревает. Ниже будет описано, как в семействе протоколов TCP/IP решается проблема негарантированной доставки пакетов сетевым оборудованием.

В технологии Ethernet применяется распределенное управление доступом, так как, в отличие от некоторых других сетевых технологий, в нем не предусмотрена централизованная система предоставления доступа. В Ethernet применяется система доступа, называемая *множественным доступом с контролем несущей и обнаружением коллизий (Carrier Sense Multiple Access with Collision Detect*, или *CSMA/CD)*. Название *множественный доступ с контролем несущей* говорит о том, что одновременный доступ к сети Ethernet имеет множество компьютеров. При этом каждая машина определяет, свободен ли эфир, по наличию несущей частоты в кабеле. Когда сетевая плата собирается передать пакет данных, она проверяет, не передается ли по сети в этот момент кем-либо другой пакет (т.е. выполняет контроль несущей). Если несущая в кабеле не обнаружена, сетевая плата начинает передачу данных. Процесс передачи пакета ограничен во времени, поскольку его длина конечна и не может превышать заранее оговоренного значения, называемого максимальным размером пакета. Кроме того, время, прошедшее после предыдущей отправки пакета сетевой платой, не должно быть меньше заранее установленного значения. Это сделано для того, чтобы предотвратить монопольное использование сети одним компьютером и предоставить доступ к сети другим абонентам.

**Адресация оборудования в сетях Ethernet**

В сетях Ethernet принят 48-битовый механизм адресации. Каждой плате сетевого интерфейса при изготовлении назначается уникальное 48-битовое число, которое называется *адресом Ethernet (Ethernet address)1*. Для этого производители оборудования приобретают блоки адресов Ethernet и назначают из них уникальные адреса каждой вновь произведенной сетевой плате. Таким образом, не может существовать двух плат сетевого интерфейса с одинаковыми адресами Ethernet.

Как правило, адрес Ethernet записывается в плату сетевого интерфейса в форме, удобной для обработки компьютером. Поскольку адреса Ethernet относятся к сетевому оборудованию, иногда их называют *аппаратными адресами (hardware addresses), физическими адресами (physical addresses), адресами доступа к среде передачи данных (media access (MAC) addresses)*, а также адресами второго уровня (*layer 2 addresses*). Обратите внимание на одну важную особенность физического адреса Ethernet.

Физический адрес назначается плате сетевого интерфейса Ethernet. Поэтому при замене в компьютере этой платы по причине ее отказа или установки новой платы соответственно, изменяется и физический адрес этого компьютера.

В дальнейшем при рассмотрении сетевых протоколов высокого уровня мы остановимся на том, как в их реализациях учитывается возможность изменения физического адреса Ethernet.

Плата сетевого интерфейса анализирует полученные пакеты и “решает”, какие из них предназначены для компьютера, в котором она установлена. Напомним, что в сетях Ethernet копии прошедших через концентратор пакетов рассылаются всем подключенным к нему сетевым платам, даже если пакет адресован только одному из компьютеров. Поэтому плата сетевого интерфейса для фильтрации пакетов использует поле адреса получателя. Она игнорирует пакеты, адресованные другим машинам, и пропускает только те, которые предназначены “ее” компьютеру. Понятно, что если бы сетевая плата передавала все полученные пакеты центральному процессору компьютера на фильтрацию и дальнейшую обработку, то при большом трафике, он вряд ли бы успел обработать все полученные пакеты в реальном масштабе времени. Поэтому чтобы избежать перегрузки компьютера входящим потоком данных, был введен механизм адресации, а фильтрация пакетов перенесена на уровень сетевого оборудования. Перенос фильтрации пакетов на аппаратный уровень преследовал и еще одну цель — освободить от этой работы центральный процессор. В результате производительность подключенных к сети компьютеров не будет снижаться при увеличении общего трафика в сети.

Выше мы уже говорили о том, что 48-битовый адрес Ethernet идентифицирует компьютер получателя. При этом адреса бывают трех типов.

* *Физический* адрес одной сетевой платы, или *одноадресатный адрес* (*unicast address*).
* *Широковещательный* сетевой адрес (*broadcast address*).
* *Многоадресатный* сетевой адрес (*multicast address*).

В соответствии с принятым соглашением, широковещательный адрес, состоящий из 48 единиц, зарезервирован для одновременной рассылки пакетов всем компьютерам.

С помощью многоадресатных адресов можно организовать усеченную форму широковещания в сети. При этом компьютеры, составляющие группу (она называется *многоадресатной*, или *multicast group*), необходимо настроить так, чтобы они реагировали на получение пакетов, посланных по указанному многоадресатному адресу. Для подключения к многоадресатной группе компьютер должен сообщить ее адрес сетевой плате и перевести плату в режим многоадресатной работы. Преимущество использования многоадресатной группы заключается в том, что имеется возможность сделать выборочную широковещательную передачу только на нужные компьютеры. При этом посылаемый по многоадресатному адресу пакет будет принят только компьютерами, входящими в группу, другие же компьютеры сети его не получат.

Для поддержки режимов широковещательной и многоадресатной адресации, плата сетевого интерфейса Ethernet на аппаратном уровне должна распознавать не только свой физический адрес. Обычно сетевые платы принимают два вида пакетов: посланные по ее физическому (одноадресатному) адресу и те, что посланы по сетевому широковещательному адресу. Некоторые сетевые платы можно запрограммировать на распознавание многоадресатных и, даже, альтернативных физических адресов. При начальной загрузке компьютера операционная система инициализирует сетевую плату Ethernet и назначает ей набор адресов, которые она должна распознавать. После этого сетевая плата на аппаратном уровне проверяет в каждом пакете поле адреса получателя и передает пакет для дальнейшей обработки компьютеру только в том случае, если адрес в пакете совпал с одним из запрограммированных адресов платы.

1. Распределением адресного пространства Ethernet занимается Институт инженеров по электротехнике и электронике (IEEE), который по мере необходимости выделяет производителям сетевого оборудования свободные блоки адресов.

**Формат фрейма Ethernet**

Стандарт Ethernet следует считать средством обеспечения соединения между абонентами сети, относящимся к канальному уровню (link-level layer). Поэтому имеет смысл рассматривать передаваемые по сети данные как состоящие из *фреймов1 (frame)*, или кадров. Фреймы Ethernet имеют переменную длину, однако их размер не может быть меньше 64 октетов2 и превышать 1518 октетов (с учетом преамбулы фрейма, передаваемых данных и кодов циклической проверки CRC). Как и в любой другой сети с коммутацией пакетов, в каждом фрейме Ethernet предусмотрено специальное поле для адреса получателя. Помимо адреса получателя, фрейм Ethernet содержит также физический адрес отправителя (рис. 2.4).

Рис. 2.4. Формат фрейма (пакета) Ethernet. Перед передачей по сети фрейм дополняется специальной преамбулой (рисунок сделан без соблюдения масштаба полей)

Таким образом, кроме полей адреса отправителя и получателя, в каждый передаваемый по сети Ethernet фрейм добавляется *преамбула, поле типа, поле данных* и *код циклического избыточного контроля (Cyclic Redundancy Check*, или *CRC*). Преамбула состоит из 64-битовой последовательности нулей и единиц и предназначена для облегчения синхронизации принимающей сетевой платы. Поле CRC (его размер 32 бита) служит для обнаружения ошибок при передаче данных на приемном конце. При этом отправитель вычисляет код циклического избыточного контроля как функцию от данных, находящихся во фрейме, и помещает их в передаваемый пакет. Получатель снова вычисляет код CRC и сравнивает его с находящимся во фрейме. Если коды совпадают, принимающая сторона считает, что пакет получен без искажений.

Поле типа состоит из 16-битового целого числа, которое предназначено для идентификации типа данных, передаваемых во фрейме. С точки зрения построения процесса межсетевого взаимодействия поле типа очень важно, поскольку оно позволяет *автоматически* распознавать фреймы Ethernet. Другими словами, операционная система на основе поля типа полученного фрейма определяет программный модуль протокола, который должен обработать поступивший пакет данных.

Одно из ключевых преимуществ автоматического распознавания фреймов заключается в том, что на одной и той же машине можно использовать несколько различных протоколов, а также передавать по физической сети пакеты данных разных протоколов без их взаимного влияния. Благодаря полю типа на одном и том же компьютере могут свободно сосуществовать приложения, использующие как семейство протоколов TCP/IP, так и любые другие протоколы, например те, что только разрабатываются и находятся на стадии тестирования. Таким образом, операционная система анализирует поле типа каждого полученного фрейма и определяет, как должно обрабатываться его содержимое. Ниже мы увидим, как автоматически опознаваемые фреймы Ethernet используются в семействе протоколов TCP/IP для выделения пакетов разных протоколов.

1. Термин фрейм (кадр) взят из терминологии, использовавшейся для описания процесса передачи данных по последовательному каналу связи. При этом отправитель разделял данные на фрагменты (фреймы) и перед передачей фрейма помещал до и после него специальные коды.

2. Формально для описания размера символа, зависящего от применяемого оборудования, используется термин байт. Однако в телекоммуникации используется другой термин — октет, который, в отличие от байта, не зависит от применяемого оборудования и всегда состоит из 8 бит.

**Проверка знаний: Сеть Ethernet**

**Задание 1**

До какой скорости была увеличена пропускная способность сети в быстром Ethernet (Fast Ethernet)?

96 Мбит/с

128 Мбит/с

100 Мбит/с

**Задание 2**

Официально технология на основе витой пары называется

**Задание 3**

Что стало слабым местом в сети Ethernet к середине 1990-х годов?

пропускная способность

безопастность

гибкость

**Задание 4**

Какой механизм адресации принят в сетях Ethernet?

48 битовый

32 битовый

54 битовый

**Задание 5**

Размер фреймов не может быть меньше, чем...?

256 октетов

64 октетов

128 октетов

**Задание 6**

В чем заключалась проблема гигабитного Ethernet?

восприимчивость к электромагнитным помехам

несовместимомть с 10BASE-T и 100BASE-T Ethernet

необходимость разработки нового протокола передачи данных

**Введение в маршрутизируемые протоколы**

Для того чтобы успешно перемещать данные из одного места в другое, всем маршрутизаторам необходимо одно: *протокол*. Поэтому на страницах этого раздела будет рассказано о работе протоколов и о маршрутизации пакетов данных.

Далее в этом курсе вы познакомитесь с конфигурированием работы маршрутизатора в отношении используемых протоколов. Чтобы приступить к конфигурированию, необходимо иметь четкое представление о том, как на самом деле работают протоколы.

* [Маршрутизация Cisco: технология и компания](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69047e)
* [История маршрутизаторов Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69047f)
* [Пришествие персонального компьютера](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690480)
* [Рождение Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690481)
* [Маршрутизация и Интернет](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690482)
* [Маршрутизация в повседневной жизни](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690483)
* [Категории протоколов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690484)
* [Модель OSI](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690485)
* [Уровень приложений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690486)
* [Уровень представлений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690487)
* [Сеансовый уровень](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690488)
* [Транспортный уровень](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690489)
* [Сетевой уровень](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048a)
* [Канальный уровень](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048b)
* [Физический уровень](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048c)
* [Типы и классы протоколов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048d)
* [Протоколы с установлением соединения и протоколы без установления соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048e)
* [Классовые и бесклассовые протоколы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69048f)
* [Инкапсуляция](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690490)
* [Проверка знаний: Введение в маршрутизируемые протоколы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690491)

**Маршрутизация Cisco: технология и компания**

Маршрутизация информации - это то, что заставляет «вращаться мир» в нашем изменчивом обществе, по мере того как оно становится все более и более зависимым от информации и от ее доставки из одного места в другое.

Вспомните о технологических новинках последнего десятилетия. Большинство из них связано с Интернетом (либо с другой похожей технологией повсеместного доступа). Теперь представьте себе эти достижения в условиях отсутствия возможности регулировать потоки информации. Многие услуги, ставшие привычными, такие как электронная почта, электронная торговля, интернет-телефония, станут практически невозможными. Большинство людей не сознают, как важна маршрутизация в технологическом обществе. Глобальная экономика находится в зависимости от способности доставлять информацию из одной системы в другую.

**История маршрутизаторов Cisco**

До 1984 года не существовало общепринятого разумного способа доставки цифровой информации из одного места в другое. Несмотря на существование метода коммутации пакетов, появившегося в середине 60-х, требования новых компьютерных сетей и коммуникационных протоколов (таких как TCP/IP) заставляли искать новые технологические решения. Компьютерные сети, которым к тому времени едва исполнилось десять лет, только начинали становиться общедоступными.

И в 60-х, и в 70-х годах отсутствие надежных средств маршрутизации не представляло проблемы (для тех случаев, когда требовалось объединение сетей, вполне подходила коммутация пакетов). Большинство компьютерных сетей в те времена состояло из пяти-шести больших машин с десятком терминалов у каждой. Эти мэйнфреймы обычно располагались в одном месте, благодаря чему не возникало потребности в длинных кабельных соединениях. Близость расположения значительно облегчала их объединение в сети. Сеть, обслуживающая 60 пользователей, требовала лишь шести сетевых соединений, расположенных в ограниченном пространстве; для соединения терминалов с мэйнфреймами сетевые соединения не требовались. На рис. 3.1 изображена сеть на основе мэйнфреймов.

Рис. 3.1. Сеть на основе мэйнфреймов

Несмотря на то что многие склонны рассматривать последовательные соединения терминалов с мэйнфреймом как «сети», с точки зрения нашего обсуждения они таковыми не являются. Мы имеем дело с сетями, используемыми и *разделяемыми* мэйнфреймами - другими словами, с разделяемыми сетевыми соединениями; последовательные соединения к ним не относятся.

Распространение больших коммутируемых сетей в 60-х и 70-х годах ограничивалось организациями, которые могли позволить себе содержать соответствующее оборудование. Это характерно для всех новых технологий: чем раньше вы вкладываете деньги в продукт, тем дороже он вам обходится. Например, правительственная сеть США ARPAnet (позднее превратившаяся в DARPAnet) была одной из крупнейших в мире и одной из первых. Оборудование этой знаменитой крупномасштабной сети, использовавшее прогрессивную технологию коммутации пакетов и сделавшее ARPAnet реальностью, из-за своей дороговизны так и осталось недоступным для большинства американских компаний.

В начале 1980-х началась революция персональных компьютеров (ПК). Компьютеры уменьшались в размерах и становились все более самодостаточными. Жесткие диски, вычислительные и сетевые мощности, ранее бывшие привилегией больших вычислительных машин, теперь могли уместиться на рабочем столе и обслуживать потребности единственного пользователя. Эти качества персональных компьютеров сделали их привлекательными для использования в бизнесе.

**Пришествие персонального компьютера**

По мере того как персональные компьютеры занимали места на рабочих столах (и в домах), администраторы начали сталкиваться с новыми проблемами. Та же сеть на 60 рабочих мест, которая раньше требовала шести сетевых соединений на сравнительно небольшой площади, теперь потребовала уже 66 соединений (60 ПК и 6 мэйнфреймов) значительно большей длины.

Для эффективного использования ПК недостаточно соединить его через последовательный интерфейс с мэйнфреймом, как терминал. Каждому ПК потребуется такое же сетевое соединение, как и у мэйнфрейма. Например, если мэйнфреймы были соединены сетью Token Ring, то и всем персональным компьютерам потребуется соединение Token Ring. В результате можно столкнуться не только с физическими ограничениями технологии, такими как MAU (Media Access Units, модули подключения к среде) и длина кабелей, но и с ограничениями пропускной способности. Сеть, обслуживавшая передачу данных между шестью устройствами, вдруг столкнулась с более чем десятикратным приростом трафика. Этот скачок был наиболее заметен в сетях Token Ring, имевших к тому времени пропускную способность 4 Мбит/c.

Персональные компьютеры требуют использования того же протокола и системы адресации, что и их старшие братья - мэйнфреймы. То есть каждый ПК, присутствующий в сети, требует собственного адресного пространства (в соответствии с используемым протоколом). Однако не это оказалось самой сложной проблемой для администраторов.

Следующая трудность, с которой столкнулись сетевые администраторы и которая потребовала применения маршрутизаторов, была вызвана низкой стоимостью ПК. Новые персональные компьютеры, готовые к работе в сети, оказались намного дешевле своих больших предшественников. В результате компании начали наращивать количество ПК в своих сетях. Администраторы столкнулись с опасностью неограниченного роста числа подключаемых к сети устройств.

Наиболее популярные в то время сетевые топологии, Token Ring и Ethernet, были разработаны еще до появления ПК. Дополнительная нагрузка на сеть, которую создали персональные компьютеры, помогла определить границы применимости существовавших сетевых решений. На рис. 3.2 изображена сеть, достигшая своего предела.

Рис. 3.2. Большая вычислительная сеть

Надо было что-то делать со все возрастающей сетевой нагрузкой. Существовавшие ранее проекты сетей уже не годились. Небольшие, ориентированные на подразделения сети пришли на смену большим и открытым сетям. Единая сетевая среда должна была разделиться (на логическом уровне) на меньшие и более простые в управлении подсети. Рисунок 3.3 иллюстрирует структуру сегментированной сети.

Рис. 3.3. Сегментированная сеть

Если вы знакомы с проектированием сетей, то уже знаете, что разделение большой сети на более мелкие подсети дает ряд преимуществ. Уменьшение размеров сети позволяет избавиться от перегрузок и упрощает управление ею. Однако такие маленькие, легко управляемые сети создают другую, более серьезную проблему. Как пользователи одного сегмента сети смогут получить доступ к ресурсам, расположенным в другом сегменте? Ответом стала маршрутизация.

Практически без привлечения стороннего финансирования Лернер и Босак на собственные средства провели исследования и разработку одного из возможных решений задачи маршрутизации. Процесс проектирования оказался сложнее, чем виделось вначале. Но все же через два года они разработали свой первый маршрутизатор и начали прокладывать сеть в кампусе Стенфорда.

Два сотрудника Стенфордского университета, Сэнди Лернер (Sandy Lerner), руководитель компьютерной службы факультета вычислительной техники, и Леонард Босак (Leonard Bosack), руководитель компьютерной службы Высшей школы бизнеса, начали работать над решением проблем растущей университетской сети. По ходу дела Лернер и Босак также начали работу над другой, намного более сложной задачей: проблемой маршрутизации межсистемных протоколов.

В конце концов, их труд и новые идеи принесли результат. Они успешно объединили оборудование Стенфордского университета при помощи своих маршрутизаторов, создав одну из крупнейших на то время университетских вычислительных сетей.

Осознавая технологический (и коммерческий) потенциал своих разработок, Лернер и Босак обратились к руководству университета с предложением коммерческого использования их продукта. (В то время в Стенфорде также искали способы коммерциализации своих технологий.) Казалось бы, каждая из сторон нашла идеальное решение своих проблем: Лернер и Босак могли получить поддержку, необходимую для вывода своих маршрутизаторов на рынок, а университет принял бы участие в коммерческом проекте.

Однако университет отказался от продвижения новой технологии маршрутизации, так что Лернеру и Босаку пришлось выходить на рынок самостоятельно. Они продолжили выпуск своих маршрутизаторов и основали компанию Cisco.

**Рождение Cisco**

Компания Cisco представила свой первый маршрутизатор широкой публике в октябре 1984 года. Лернер и Босак продолжали финансировать компанию из частных источников и доходов от продаж. Но поставки продукта не поспевали за растущим спросом. Начиналась эра сетевых вычислений. Компании, которые не интересовались технологией коммутации пакетов (либо не могли ее себе позволить), начали присматриваться к маршрутизаторам Cisco. Объединение сетей любых размеров выполнялось сравнительно легко. Закладывался фундамент Интернета (такого, каким мы его знаем сейчас).

Чтобы удовлетворить высокий спрос на маршрутизаторы, Лернер и Босак обратились к группе венчурных предпринимателей с предложением о финансировании, необходимом для роста компании в избранном направлении. Проблема в том, что многие (если не все) венчурные предприниматели стремятся войти в правление проинвестированной ими компании, чтобы сохранить контроль над расходованием денег. В случае с Cisco венчурные предприниматели в конце концов прибрали компанию к рукам, и Лернер и Босак ушли из созданной ими фирмы.

В 1991 году Джон Чамберс (John Chambers) из компании Wang занял в Cisco пост главного вице-президента, а в 1995 получил статус президента и CEO. Под управлением Чамберса годовой доход Cisco вырос до 20 миллиардов долларов. Несмотря на то, что компания изменилась с тех времен, когда Сэнди Лернер и Леонард Босак основали Cisco в Стенфордском университете, цель осталась прежней: выпуск лучшего в мире оборудования для маршрутизации.

Маршрутизаторы Cisco единодушно признаны лучшими в индустрии. На сегодняшний день им нет равных по сочетанию мощности, надежности и масштабируемости. Нигде эти характеристики не проявляются так хорошо, как в изменчивой среде Интернета.

**Маршрутизация и Интернет**

Одной из наиболее важных технологических новинок последнего времени стал Интернет. То, что начиналось как один из способов связи между военными учреждениями США, превратилось в индустрию с многомиллиардным годовым оборотом. Теперь Интернет используется везде, начиная с распространения новостей и другой информации, торговли и игр и заканчивая телевещанием и телефонией. Несомненно, Интернет стал неотъемлемой частью современного общества.

Первая инкарнация Интернета имела мало общего с современным киберпространством. В младенческие годы Интернет был совершенно другим. В начале 1980-х годов, когда в магазинах появились первые персональные компьютеры, по всей стране стали возникать небольшие сети серверов электронных досок объявлений (BBS). Эти серверы обычно располагались в крупных университетских городках, то есть там, где была доступна соответствующая технология и где пользователи могли обращаться к ним из дома при помощи модема.

Пользователи BBS могли позвонить на сервер и получить доступ к информации, новостям, чатам и службам сообщений (предшественникам электронной почты). Эти небольшие (по сегодняшним меркам) сообщества пользователей, соединявшихся по модему с большими BBS, стали той структурой, из которой впоследствии возник Интернет. Однако чудо электронной почты, ставшее началом массового компьютерного взаимодействия, было еще далеко от совершенства.

Главным недостатком систем BBS была их разобщенность. На протяжении более чем пяти лет (целая жизнь в мире технологий) пользователям всего мира были доступны только радости набора телефонного номера их локальной университетской BBS и «разговора» с незнакомцем из соседнего округа. Люди общались друг с другом, но те, кто не знал, какой номер следует набирать и как войти в систему, не могли вырваться из среды их каждодневного общения.

Со временем, по мере появления продуктов маршрутизации Cisco как в коммерческих организациях, так и в университетах, различные системы BBS начали общаться друг с другом. Теперь человек, звонящий на BBS Массачусетского технологического института (МТИ), мог не только узнать температуру газировки в научном корпусе, но и установить связь с университетом Вашингтона и получать оттуда последние геосейсмические данные.

80-е годы Cisco провела за стратегическим закладыванием основ Интернета. Именно маршрутизаторы Cisco сделали возможной и простой «бесшовную» стыковку сотен различных систем, начиная с самых первых BBS. Сотни невидимых пользователям BBS маршрутизаторов Cisco передавали сигналы МТИ в Университет Вашингтона. Рассвет Интернета был уже совсем близок.

В конце восьмидесятых годов Университет Миннесоты начал работу над системой Gopher. Являясь расширением таких технологий, как Lynx и Trumpet, она должна была дать пользователям BBS возможность просматривать графику на тех компьютерах, где когда-то доминировал текст. Фактически произошло рождение Интернета. Ради возможности побродить по Интернету люди тысячами подписывались на такие онлайн-службы, как Genie, Prodigy и CompuServe. Но никто из них не знал (или не интересовался), как информация из Интернета попадала на их экран. А все это стало возможным благодаря маршрутизации.

На заре появления Интернета функция маршрутизатора была ясно и четко определена. Маршрутизаторы были необходимы для эффективного перемещения данных из одной системы в другую. Ранние маршрутизаторы работали во многом так же, как и нынешние. С технической точки зрения они проверяют пакеты, рассчитывают пути и принимают «разумные» решения в отношении маршрутизации. Эти процессы позволяют маршрутизатору определить, откуда поступают данные, куда они направляются и как доставить их по назначению. Будет справедливо сказать, что существование Интернета в том виде, который доступен сегодня, было бы невозможно без маршрутизаторов и их способностей.

Отличие же маршрутизаторов Cisco, имеющихся на рынке в настоящее время, от их предшественников состоит в богатом наборе функций, накопленных ими за прошедшие годы. Изменялась технология, вслед за ней изменялись и маршрутизаторы. Многие нынешние усовершенствования и достижения в технологии маршрутизации исходят от Cisco. Другие производители быстро копируют многие элементы и функции, встроенные в оборудование маршрутизации Cisco, что подтверждает лидерство Cisco в данной отрасли информационных технологий.

Современные маршрутизаторы Cisco умеют транслировать внешние IP-адреса, выполнять функции межсетевых экранов, или брандмауэров, предоставлять удаленный доступ к сети, отфильтровывать лишний трафик и многое другое. Но основная обязанность маршрутизатора не изменилась за прошедшие годы — это рациональная передача данных из одной системы в другую.

Давайте посмотрим на то, как маршрутизатор Cisco работает в Интернете. На рис. 3.4 представлен пользователь, просматривающий веб-сайт в Интернете.

Рис. 3.4. Процесс маршрутизации в Интернете

Впервые пользователь может столкнуться с маршрутизатором Cisco в процессе регистрации, особенно если соединение с Интернетом осуществляется по широкополосному каналу.

Маршрутизаторы Cisco умеют работать с интерфейсами многих типов — от WAN-интерфейсов, таких как ISDN (Integrated Services Data Network, цифровая сеть с комплексными услугами), DSL (Digital Subscriber Lines, цифровые абонентские линии) и T-каналы, до LAN-соединений, таких как Ethernet; один маршрутизатор Cisco может управлять несколькими соединениями в разных средах. Поэтому вполне возможно, что первой точкой соприкосновения с сетью провайдера будет маршрутизатор Cisco, выполняющий функцию межсетевого экрана.

В зависимости от установок окружения интернет-провайдера межсетевой экран может быть расположен перед сетевым устройством аутентификации, после него или и там, и там.

Для большинства интернет-провайдеров маршрутизатор Cisco можно поместить на внешний край сети. Он будет первой линией обороны в системе обеспечения безопасности. Любой маршрутизатор, расположенный на границе сети, подобно рассматриваемому (пользователи могут подсоединяться через него), скорее всего, будет отправлять все данные с межсетевого экрана к NAT (Network Address Translation, трансляция сетевых адресов).

Обычно, говоря о «граничном маршрутизаторе», имеют в виду маршрутизатор, который использует BGP (Border Gateway Protocol — протокол граничного шлюза). Однако в этом сценарии пользователь услуг провайдера не имеет возможности подсоединиться к сети через граничный маршрутизатор BGP.

В данном примере на маршрутизаторе, находящемся на границе сети провайдера, работает межсетевой экран, защищающий устройство аутентификации сети (Network Authentication Device). После того как пользователь идентифицирован в сети провайдера, он может начать бродить по Интернету.

Приведенное ниже описание работы маршрутизаторов в Интернете является упрощенным. Дочитав курс до конца, вы получите полное представление о данном вопросе.

Предположим, что пользователь хочет посетить сайт *www.marzde-sign.com*. Броузер пользователя оповещает провайдера интернет-услуг об извлечении информации с IP-адреса 207.217.96.36, соответствующего доменному имени *www.marzdesign.com*. Запрос перемещается с одного маршрутизатора провайдера на другой. Каждый маршрутизатор, исследуя запрос, определяет, какой IP-адрес запрошен и кто является инициатором запроса, затем сопоставляет эти данные со своей собственной *таблицей маршрутов*. Таблица маршрутов указывает, где искать запрашиваемый IP-адрес, или же (если таблица маршрутов не знает, где найти такой адрес) сообщает адрес другого маршрутизатора, который может это знать (своего рода соглашение типа «знакомый моего знакомого»). Процесс продолжается до тех пор, пока запрос не достигнет *www.marzdesign.com*. Тогда таблицы маршрутов (которые следили за тем, кто сделал запрос) посылают необходимую информацию обратно пользователю. Этот сценарий повторяется миллионы раз в день, а пользователи в большинстве случаев даже не знают об этом.

**Маршрутизация в повседневной жизни**

К середине 1990-х годов состав оборудования, формирующего технологическую основу Интернета, в целом определился. Большая часть аппаратных средств маршрутизации, приводящих в действие Интернет, принадлежит семейству Cisco. Их маршрутизаторы, коммутаторы и другие продукты, обеспечивающие возможность соединения, делают Интернет более надежным, более масштабируемым и более быстрым, чем когда бы то ни было ранее.

Как же способность Cisco приводить в действие Интернет реализуется в повседневной жизни? Вскоре после того как Cisco начала искать новые пути по организации связи между компаниями, последние начали думать о расширении и разветвлении. Многие предприятия стали экспериментировать, разрешив сотрудникам выполнять свои служебные обязанности, находясь дома.

Чтобы соответствовать современным технологическим требованиям бизнеса, Cisco в середине девяностых выпустила линию оборудования SOHO (small office/home office — малый офис/домашний офис). Линия маршрутизаторов SOHO обеспечивала полную функциональность предприятия в гораздо более мелком масштабе. Такие возможности, как списки доступа, каналы WAN и межсетевые экраны, которые обычно реализуются в старших моделях оборудования, теперь могут быть приобретены домашним пользователем. Маршрутизатор Cisco может обеспечить необходимую маршрутизацию и в случае домашней сети с интернет-доступом для нескольких компьютеров, и если дом используется в качестве вспомогательного офиса.

Во многих компаниях есть сотрудники, которые некоторое время работают дома. Существует несколько способов обеспечить такой процесс, и в любом из них можно использовать маршрутизаторы Cisco. Можно соединить удаленный офис с главным с помощью канала WAN. Характерный домашний офис представлен на рис. 3.5.

Рис. 3.5. Типичный домашний офис

Домашний офис связан с главным офисом посредством ISDN. Больший маршрутизатор (например, серии 7500) расположен в главном офисе. Модем ISDN установлен как встроенное периферийное устройство на большом маршрутизаторе, а маленький маршрутизатор SOHO (скажем, Cisco серии 1600) находится дома у пользователя и снабжен другим периферийным устройством ISDN, принимающим второй конец ISDN-соединения. Теперь два удаленных друг от друга маршрутизатора соединены таким образом, что могут взаимодействовать так, как если бы находились в одном здании.

Cisco проделала очень большую работу для гарантии того, что ее «домашние» продукты во всех отношениях так же соответствуют существующим технологиям, как и продукты уровня предприятия. Большая часть не только работает с тем же программным обеспечением (Cisco IOS), но и может использовать одинаковые аппаратные модули, такие как энергозависимая и энергонезависимая память и платы расширения.

Несмотря на то, что WAN-технологии, такие как ISDN и Frame Relay, являются общепринятым решением для удаленного доступа, существует и более популярный вариант. Для предприятий с большим количеством сотрудников, работающих не выходя из дома, WAN-технологии могут быть слишком дорогостоящими. Поэтому многие компании предпочитают технологию VPN (Virtual Private Network, виртуальная частная сеть).

На рис. 3.6 изображена сеть «дом-работа», реализующая VPN-технологию. Если вы приглядитесь, то заметите, что отличие данной сети от сети, изображенной на рис. 3.5, невелико. В сети на рис. 3.6 нет домашнего устройства маршрутизации. Для установления соединения с Интернетом пользователю не требуется иметь дома устройство, подобное Cisco серии 1600.

Рис. 3.6. Виртуальная частная сеть «дом-работа»

Однако если посмотреть со стороны компании, можно заметить и более интересную особенность этой конкретной сети. Тот же самый маршрутизатор Cisco серии 7500, который использовался для удаленного доступа, может использоваться и для организации виртуальной частной сети. Фактически один маршрутизатор может одновременно обеспечивать работу обоих сценариев. Следовательно, один маршрутизатор Cisco может обслуживать распределенную сеть для практически любого вида бизнеса. Едва ли можно проектировать сеть, работать в сети или же каким-либо другим образом взаимодействовать с сетью, не столкнувшись с маршрутизатором Cisco. Компания занимает сильную позицию на рынке и имеет хорошую репутацию благодаря производству продукта маршрутизации высшего качества.

Теперь, после того как вы приняли решение освоить маршрутизацию Cisco и познакомились с историей продукта, давайте поговорим об оборудовании, обеспечивающем всю эту работу. В следующем разделе будет представлено аппаратное обеспечение, применяемое для маршрутизации.

**Категории протоколов**

Протоколы можно разделить на две категории: *маршрутизируемые протоколы* и *протоколы маршрутизации*. Маршрутизируемые протоколы — это то, что большинство людей понимает под словом «протокол». Маршрутизируемые протоколы могут маршрутизироваться другими устройствами. Такие общеизвестные протоколы, как IP и IPX, относятся к маршрутизируемым.

**Протоколы маршрутизации** — это протоколы, которые маршрутизаторы используют для взаимодействия друг с другом (когда маршрутизируют маршрутизируемые протоколы). К протоколам маршрутизации относятся RIP и OSPF. Эти протоколы помогают маршрутизаторам наиболее эффективно передавать данные.

Далее мы будем изучать работу протоколов, использование их маршрутизаторами, а также поговорим о том, в каком случае и какому протоколу отдать предпочтение. Наиболее подробно будут рассмотрены протоколы IP и IPX. Именно эти два протокола вы, скорее всего, встретите при работе с маршрутизаторами Cisco.

Помните (особенно это касается новичков в маршрутизации), что не все протоколы можно маршрутизировать. То есть даже если протокол может применяться для передачи информации в сети, это не означает, что маршрутизаторы могут взаимодействовать с ним. Поэтому при планировании маршрутизируемой среды можно принять решение избегать использования некоторых протоколов.

Чтобы иметь четкое представление о том, как работают протоколы, сначала необходимо понять, где они работают (имеется в виду не физическое местоположение, а способ взаимодействия протокола с компьютером или маршрутизатором). Разные протоколы работают на разных уровнях модели OSI; одно устройство может использовать набор протоколов для выполнения какой-либо задачи.

Многие протоколы сгруппированы в «наборы», или «стеки». Дело в том, что большая часть протоколов была создана для выполнения каких-то специальных функций, и один протокол умеет делать то, что другой не может. Объединившись, протоколы дополняют друг друга и могут коллективно выполнять ряд задач.

Примером двух протоколов, работающих вместе, служат TCP и IP. Эти протоколы работают на разных уровнях модели OSI, поддерживая связь между компьютерами. Далее будет описана роль модели OSI и ее влияние на работу маршрутизаторов Cisco.

Знание функций всех семи уровней модели OSI (даже тех, которые не имеют прямого отношения к маршрутизации) поможет вам лучше осознать перемещение данных. Вам также легче будет выявлять возможные проблемы. Например, зная, что World Wide Web работает на седьмом уровне, вы сможете определить источник связанной с Web ошибки. Так как седьмой уровень OSI управляет использованием приложений, то проблема с World Wide Web может быть связана с программой просмотра страниц.

Важно понимать, что протоколы служат определенной цели, и совсем не обязательно, что это маршрутизация. Одной из многих функций маршрутизируемых протоколов является сегментирование и инкапсулирование данных. Маршрутизаторы пользуются преимуществом этой функции, применяя ее для содействия процессу маршрутизации. Но маршрутизируемые протоколы могут существовать и в средах, где нет маршрутизаторов. (Поэтому бывают и немаршрутизируемые протоколы.)

Чтобы научиться безошибочно определять, какие протоколы будут работать с маршрутизаторами Cisco, а какие – нет, необходимо сначала понять различия между ними. Проиллюстрировать эти различия поможет модель OSI.

**Модель OSI**

В начале 1980-х годов Международная организация по стандартизации (International Organization for Standardization, ISO) разработала модель OSI (Open Systems Interconnect, взаимодействие открытых систем). В основе OSI лежит получившая дополнительное развитие эталонная модель DoD. ISO взяла четырехуровневую модель DoD и структурировала ее еще сильнее. В результате появились семь уровней OSI. Подразделение модели OSI на семь уровней помогает выделить некоторые области передачи данных для специализированных протоколов.

Уровни модели OSI пронумерованы снизу вверх. Чтобы перенести информацию с одного компьютера на другой, данные должны пройти через эти уровни как на передающем, так и на принимающем устройствах. Информация седьмого уровня одной машины должна:

1. переместиться с седьмого на первый уровень машины-отправителя;
2. быть передана протоколу (возможно, для маршрутизации) и доставлена на устройство назначения;
3. перейти с первого на седьмой уровень машины-получателя.

Каждый уровень модели OSI выполняет свою определенную функцию. Информация, поступающая с одного уровня, слегка изменяется, чтобы стать удобочитаемой для следующего уровня. Когда данные достигают принимающего устройства, они переходят от уровня к уровню в обратном порядке, чтобы аннулировать изменения, сделанные исходным компьютером. Поэтому даже если маршрутизатор имеет дело с данными только после того, как они попали на третий уровень, но при этом уровни с 4 по 7 работают некорректно, то эти уровни будут пагубно воздействовать на функционирование маршрутизатора.

В следующих разделах для обеспечения полного понимания роли протоколов в маршрутизации рассмотрены все уровни и выполняемые ими задачи. Изучив модель OSI, мы сможем перейти к обсуждению принципов IP и IPX.

**Уровень приложений**

Уровень приложений (уровень 7) занимается координированием связи между приложениями. Этот уровень модели OSI синхронизирует данные, перемещающиеся между клиентами и серверами, заведуя передачей файлов, сетевым управлением и обработкой услуг. В круг обязанностей уровня приложений входят:

* World Wide Web (WWW)
* Шлюзы электронной почты
* Электронный обмен данными (Electronic Data Interchange, EDI)
* Чаты
* Утилиты навигации по Интернету

Уровень приложений можно рассматривать как первый шаг, предпринимаемый данными, покидающими ПК, для их последующей маршрутизации. Уровень приложений — это прямой доступ маршрутизируемой информации к программе, в которую она направляется. То есть большая часть маршрутизируемой информации (если не вся) или исходит от приложения, работающего на устройстве, или же для него предназначена. Уровень приложений модели OSI задает правила, управляющие обработкой такой информации.

Например, когда вы просматриваете веб-страницу, вы просматриваете данные на уровне приложений. Информация, образующая веб-страницу, проходит через семь уровней модели OSI на удаленном сервере, «пересекает» пространство Интернета и поднимается по уровням OSI вашего компьютера. В конечном счете данные становятся удобочитаемыми на уровне приложений.

**Уровень представлений**

Функция уровня представлений состоит в преобразовании информации уровня приложений в формат, который понимают другие уровни. Любые шифрования, дешифрования и сжатия данных производятся на шестом уровне модели OSI. Уровень представлений также отвечает за все функции аудио- и видеопредставлений. Доступные на уровне представлений службы включают:

* MP3
* RealAudio
* RealVideo
* JPEG
* GIF

Заметьте, что все эти сервисы требуют сжатия. Для MP3 необходимо чрезвычайно эффективное сжатие звука, а GIF использует сжатие изображений. Если не применять к данным сжатие, их объем будет слишком велик, чтобы маршрутизатор смог обработать поступившие данные без ошибок. Чем больше информации маршрутизируется в рамках одного сеанса связи, тем меньше вероятность того, что она достигнет нужного получателя в целости и сохранности. Поэтому сжатие данных (вместе с шифрованием и дешифрованием) играет важную роль в обеспечении маршрутизации.

**Сеансовый уровень**

Сеансовый уровень (уровень 5) координирует связь между сетевыми устройствами. Сеансовый уровень (работая в паре с сеансовым уровнем другого устройства) устанавливает сеанс связи между двумя приложениями. Два сеансовых уровня контролируют «разговор» и в нужный момент заканчивают сеанс. Также сеансовый уровень отвечает за:

* SQL (Structured Query Language, структурированный язык запросов)
* X Windows
* NFS (Network File System, сетевая файловая система)

Информация, отправляемая (или получаемая) этими тремя верхними уровнями (уровни 5, 6 и 7), называется *пользовательскими данными*. Пользовательские данные преобразовываются в другие формы данных, подходящие для обработки остальными уровнями.

**Транспортный уровень**

Транспортный уровень (уровень 4) отвечает за получение пользовательских данных от верхних уровней и разбиение их (или же обратную сборку) на удобные для передачи порции. Порции данных, формируемые транспортным уровнем, называются *сегментами*. Такие сегменты передаются нижним уровням для дальнейшей обработки.

Работаете ли вы с сегментами, кадрами, дейтаграммами или ячейками — знание терминологии уровней модели OSI очень вам поможет. Для каждого уровня существует специальный формат данных, так что уровень (а обычно и протокол) можно идентифицировать по формату данных. Например, так как TCP работает на четвертом уровне, то все передаваемые по нему данные будут иметь форму «сегментов».

Транспортный уровень также предоставляет возможность управления потоками данных. Управление потоками помогает этому уровню обеспечить надежную передачу данных от одного устройства к другому (с установлением соединений). Транспортный уровень забирает пользовательские данные с верхних уровней и сегментирует их. Затем эти сегменты по одному передаются указанному получателю. Получив сегмент, получатель отправляет обратно уведомление. Если отправитель не получает уведомления, он заново передает сегмент. После нескольких попыток отправляющее устройство старается заново установить соединение с получателем. Если оказывается, что получатель не отвечает (принимающее устройство больше не работает в сети), то генерируется ошибка и остальные сегменты не отправляются. Поток данных, передаваемых с одного ПК на другой по TCP, изображен на рис. 3.7.

Рис. 3.7. Поток данных TCP

**Сетевой уровень**

Все уровни выполняют очень важные функции, но основная часть процесса маршрутизации проходит именно на уровне 3. Большая часть устройств межсетевого взаимодействия (маршрутизаторы, коммутаторы третьего уровня и мосты) работает на сетевом уровне модели OSI.

Для обеспечения правильности маршрутизации сетевой уровень создает логическую карту сети. Эта карта работает как проводник при направлении данных из одной части сети в другую. Хотя функция составления карты сети имеет очень важное значение для маршрутизации, используется она не только маршрутизаторами. Персональные компьютеры и другие устройства также используют службы сетевого уровня для определения местоположения маршрутизаторов в окружении. Это позволяет им посылать маршрутизаторам информацию, которая должна быть доставлена по сети.

Первым этапом создания карты сети является преобразование сегментов транспортного уровня в пакеты. Затем эти пакеты, к которым добавлена адресная информация, передаются канальному уровню.

Так как маршрутизаторы практически всегда работают только на третьем уровне модели OSI, они перемещают данные только в виде пакетов. Однако когда мы дойдем до WAN-технологий, вы увидите, что бывают маршрутизаторы, которые передают кадры и ячейки. Помните, что формат данных во многом зависит от протокола.

Когда устройство получает пакет, из него извлекается информация об отправителе и помещается в таблицу. По мере роста таблицы сетевой уровень получает более явную картину сетевого окружения. Другие протоколы и устройства могут использовать эту информацию для эффективной маршрутизации данных.

Данные, используемые сетевым уровнем, также сохраняются локально на маршрутизаторах Cisco в таблице маршрутов. Конкретная информация, содержащаяся в таблице маршрутов, зависит от используемого протокола маршрутизации. Более подробно таблицы маршрутов будут затронуты в следующих разделах, посвященных протоколам маршрутизации.

Например, если устройство сети A хочет отправить данные устройству сети B, оно посылает широковещательный пакет через свою локальную сеть (A). Этот широковещательный пакет проводит «разведку», разыскивая адрес получателя. Так как принимающее устройство не находится в сети A, то «разведчик» не получит ответ. Тогда устройство сети A предположит, что получатель находится в сети B, и отправит данные туда.

Прежде чем достичь сети B, пакет попадает на маршрутизатор 1. Маршрутизатор просматривает свою собственную таблицу маршрутов (составленную из данных сетевых уровней) и определяет, что получатель действительно находится в сети B. Тогда маршрутизатор 1 пересылает пакет в соответствующие сети. Маршрутизатор, соединяющий две сети, отметит эту ситуацию в своей таблице маршрутов и в дальнейшем будет направлять все пакеты, предназначающиеся данному устройству, в сеть B.

Ключевым элементом сценария является знание адреса устройства, с которым вы пытаетесь поделиться информацией. Сбором и отслеживанием этих адресов занимается канальный уровень.

**Канальный уровень**

Сетевой уровень поддерживает карту сети, а канальный уровень (уровень 2) обеспечивает корректность информации в этой карте за счет адресации. Канальный уровень принимает пакеты от сетевого уровня и преобразует их в кадры данных. Кадры содержат следующую информацию:

* Преамбулу (указывающую начало кадра)
* Адрес назначения (получателя)
* Адрес отправителя
* Поле длины (в стандартном кадре Ethernet) указывает размер данных, содержащихся в кадре
* Поле типа (в кадрах Ethernet\_II) указывает, какой протокол будет получать данные
* Данные
* Контрольная последовательность кадра (проверочный номер, соответствующий контрольной сумме кадра)

Канальный уровень также обеспечивает много других возможностей, которые доступны не всем устройствам. Поэтому уровень был разбит на два подуровня: MAC и LLC. В каждом подуровне существуют свои правила и атрибуты.

**Подуровень MAC**

Подуровень MAC (media access control, управление доступом к среде) отвечает за кадрирование пакетов сетевого уровня. Разбивая пакеты на кадры, подуровень MAC прикрепляет к пакету адресную информацию, в которую входит MAC-адрес.

Каждое устройство, пригодное для работы в сети, имеет присвоенный ему при изготовлении адрес, однозначно идентифицирующий этот компонент в сети. Такой адрес называется MAC-адресом.

Еще одной функцией подуровня MAC является обслуживание верхних уровней без установления соединения. Такое обслуживание имеет место, когда данные посылаются на устройство без предварительного установления с ним соединения. Другими словами, отправляющее устройство отсылает данные по сети, заранее не извещая об этом получателя.

Таблица 3.1. «За» и «против» обслуживания без установления соединения

| **За** | **Против** |
| --- | --- |
| Быстрее, чем обслуживание с установлением соединений. (Не тратится время на установление соединений и ожидание ответов) | Не гарантируется доставка кадров |
| Меньше нагрузка на сеть | Получатель не знает об отправке данных. Машина, которой вы отправляете информацию, может даже не работать в этот момент |

Далее вы узнаете, что подуровень MAC имеет очень важное значение для маршрутизации. Так как MAC-адреса уникальны и распознаются почти всеми протоколами, то их можно встретить во многих аспектах маршрутизации.

**Подуровень LLC**

Одной из задач подуровня LLC (logical link control, управление логическим соединением) является предоставление обслуживания с установлением соединений (в то время как MAC не устанавливает соединение). В этом случае перед отправкой кадров устанавливается соединение с получателем, благодаря чему доставка кадров гарантируется получением уведомлений.

**Физический уровень**

Первый уровень модели OSI определяет физическое соединение между устройствами. Физический уровень принимает кадры от верхних уровней и передает их в виде битов по среде сети передачи данных. При работе с интерфейсами маршрутизаторов Cisco это становится гораздо более понятным. Все физические порты и интерфейсы оборудования Cisco для маршрутизации работают на физическом уровне модели OSI. Весь вышеописанный процесс представлен на рис. 3.8.

Рис. 3.8. Схема прохождения данных через семиуровневую модель OSI

Осмысление модели OSI поможет вам понять, как работают протоколы, что, в свою очередь, поможет овладеть принципами маршрутизации, на постижение которых некоторые тратят долгие годы.

Закончив разговор о внутреннем устройстве модели OSI, обсудим, как маршрутизаторы используют эту информацию для перемещения данных из системы в систему.

**Типы и классы протоколов**

Cуществование протокола не означает, что он маршрутизируем. Примером немаршрутизируемого протокола служит NetBEUI. С помощью NetBEUI могут быть созданы целые сети, которые будут замечательно работать до тех пор, пока не будут использоваться маршрутизаторы.

Для того чтобы понять, как протоколы работают и как они перемещают данные, необходимо знать, что у них «внутри». Для каких целей протокол был создан, на каких уровнях модели OSI он работает и что от него можно ожидать? Рассмотрение различных типов протоколов поможет вам понять, как работают протоколы маршрутизатора и как маршрутизаторы Cisco с ними управляются.

**Протоколы с установлением соединения и протоколы без установления соединения**

Протоколы с установлением соединения разработаны так, чтобы способствовать установлению соединений между системами. То есть когда две системы используют протоколы с установлением соединений, то прежде чем передавать информацию, они устанавливают между собой соединение.

Протоколы с установлением соединения, такие как TCP (Transmission Control Protocol — протокол управления передачей), работают на транспортном уровне модели OSI. Так как многие маршрутизаторы работают на сетевом уровне (которому внутренне присуще отсутствие соединений), протоколы с установлением соединения не являются маршрутизируемыми. Но мы все же поговорим о них, так как у них имеется ряд полезных свойств, к тому же, они тесно связаны со многими дружественными маршрутизатору протоколами без установления соединений, с которыми мы их и сравним.

Когда одна система хочет отправить информацию в соседнюю, получателю отправляется запрос на установление соединения. Система-отправитель ждет уведомление о готовности к соединению. Получив его, система-отправитель устанавливает соединение, и две системы могут свободно обмениваться данными. По завершении соединения инициировавшая его система отправляет пакет «tear-down», который показывает, что соединение закончено и используемые для его поддержания ресурсы должны быть освобождены. Процесс установления соединения представлен на рис. 3.9.

Рис. 3.9. Процесс установления соединения

Сам процесс (инициирование соединений перед обменом информацией) не создает особых проблем; в действительности это один из наиболее привлекательных аспектов данной категории протоколов. Проблемы для некоторых сетей может вызвать тот способ, которым соединение поддерживается.

Когда инициатор запроса посылает пакет установления соединения, к пакету добавляется запись о пути, который он проходит. Таким образом пакет создает маршрут, по которому будет происходить обмен информацией между двумя устройствами. Получающая пакет система отправляет свое уведомление (acknowledgment — ACK) по тому же пути, который прошел пакет установления соединения. И после открытия сеанса связи между двумя системами все пересылаемые ими пакеты следуют по тому же самому маршруту, в результате чего между устройствами возникает гарантированный канал связи.

Но установление соединения и поддержание назначенного пути требуют значительных ресурсов. Накладные расходы протоколов с установлением соединения выше, чем у протоколов без установления соединений. К тому же, из-за того что в основе работы маршрутизатора лежит случайность, он не может поддерживать назначенные пути.

Маршрутизаторы не могут гарантировать, что два пакета будут отправлены по одному и тому же пути. Маршрутизатор всегда пытается выбрать «лучший путь» для отправки информации. А так как на этот выбор влияет множество связанных с состоянием сети обстоятельств, то от пакета к пакету маршрут может меняться.

Протоколы с установлением соединения имеют и ряд преимуществ для сетевого окружения. Так как прием любого пакета, отправленного той или иной системой, подтверждается, то доставка данных гарантирована. Когда две системы участвуют в сеансе обмена информацией, то пакет-подтверждение посылается, чтобы сообщить об успешном прибытии каждой порции данных. Поэтому первым очень приятным свойством протоколов с установлением соединения является гарантированная доставка информации.

Кроме того, протоколы с установлением соединения работают по заданному пути. То есть каждый пакет отправляется по тому же пути, что и предыдущий, что облегчает поиск неисправностей в случае проблем с соединением. Используя такие средства, как пакетные анализаторы или модули проверки текущего состояния LAN, специалист легко может определить маршрут, по которому взаимодействовали два устройства, и диагностировать возможные проблемы.

Протоколы без установления соединений, такие как IP, работают (за редким исключением) на сетевом уровне модели OSI. Маршрутизаторы Cisco (и все остальные маршрутизаторы, присутствующие на рынке) также работают на сетевом уровне. Из чего можно сделать логический вывод о том, что все маршрутизируемые протоколы относятся к протоколам без установления соединений и что маршрутизаторы поддерживают обмен информацией только без установления соединений.

Протоколы без установления соединений используют при доставке данных принцип «наибольших усилий». Когда устройство пытается отправить информацию другой системе, между ними не устанавливается специальное соединение. Использование протоколов без установления соединений несколько рискованно. С ними связаны такие возможные опасности, как:

* Несоблюдение порядка доставки пакетов
* Потеря пакетов
* Повышенная сложность при поиске и устранении неисправностей

Так как между устройствами нет выделенного соединения, каждый отправленный ими пакет может перемещаться по пути, не совпадающему с остальными. Поэтому вероятность того, что пакеты прибудут не в том порядке, в котором были отправлены (если вообще прибудут), гораздо больше, чем при использовании протоколов с установлением соединений.

Главным недостатком протоколов без установления соединений является неминуемая потеря данных, находящихся в недоставленных пакетах. И хотя в большинстве случаев потери минимальны и едва заметны, но риск все-таки есть.

Еще одним неудобством, вызванным отказом от использования выделенного пути, является то, что сложность поиска и устранения неисправностей возрастает экспоненциально при увеличении количества маршрутизаторов в сети. Так как пакеты могут использовать различные пути, чтобы попасть в один и тот же пункт назначения, то чрезвычайно сложно предсказать, где же может возникнуть проблема.

Один из способов контроля маршрутизатора Cisco за доставкой данных из одной системы в другую — это использование статических маршрутов (когда возможно). Хотя статические маршруты, конечно же, подходят не для каждого случая (на самом деле они в большей или меньшей степени лишают маршрутизацию смысла), они могут помочь в особенно ненадежных системах.

Есть и явные преимущества использования протоколов без установления соединений. Обычно они быстрее и требуют меньших ресурсов, чем протоколы с установлением соединений. Именно поэтому они идеально подходят для маршрутизации. В условиях полиморфной природы маршрутизации протоколы без установления соединения предоставляют возможность быстро перемещать данные из системы в систему.

Так как протоколы без установления соединения не требуют установления сеанса связи, они могут быть быстрыми. Время, необходимое для установления и завершения соединения, на отправку уведомлений и ответы на них, в больших сетях накапливается. Поэтому для современных маршрутизируемых сетей так важно быстродействие протоколов без установления соединения.

**Классовые и бесклассовые протоколы**

Объяснять отличие классовых протоколов от бесклассовых лучше всего на примере IP. В том, что касается протоколов, в списке ваших приоритетов на первом месте должен стоять IP.

Причина, по которой различия классовых и бесклассовых протоколов рассматриваются на примере IP, заключается в том, что в последние годы IP стал и тем и другим, то есть его можно сконфигурировать и как классовый, и как бесклассовый протокол.

**Классовый IP**

По своей природе IP относится к классовым протоколам. Когда IP разрабатывался (вместе с первыми сетевыми окружениями), он был задуман как классовый протокол. Это означает, что IP может быть разделен на классы, каждый из которых будет ориентироваться на нужды конкретной сети клиента.

Классы используются для определения предельного размера среды на основании соотношения количества машин и сетей. Каждый класс предлагает разное количество адресуемых машин для адресуемой сети. Один класс может предлагать 127 адресуемых сетей и более 16 миллионов адресуемых машин в сети, а другой — 2 миллиона сетей и 254 машины в сети.

В случае с IP протокол разделен на три (общепринятых) класса. Классы были разработаны так, чтобы соответствовать потребностям организаций разных размеров.

Хотя общепринятым считается разделение IP на три класса (A, B и C), на самом деле их пять. Классы A, B и C наиболее популярны при сетевой адресации. Но классы D и E могут использоваться в других целях, например для многоадресной передачи.

В классовом протоколе один адрес используется для задания как машины, так и сети на переменой основе. То есть та часть адреса, которая определяет сеть (а не машину), может иметь разный размер в зависимости от используемого класса. Например, в IP-сети класса A адрес 16.10.20.6 может быть разбит на адрес сети — 16.0.0.0 и адрес машины -0.10.20.6. В классе же C адрес 225.198.40.9 может быть разбит на сетевой — 225.198.40.0 и адрес машины — 0.0.0.9.

Маршрутизатор определяет, какая часть адреса использована для сети, применяя вторичный адрес или маску. Маски (уникальные для классовых протоколов) указывают маршрутизирующему устройству, какие биты протокольного адреса представляют сеть. Эти вычисления являются ключом к функциональности маршрутизатора.

**Бесклассовый IP**

Проблема классовых протоколов, таких как IP, в том, что они конечны в том смысле, что существует определенный предел количества адресов, которые могут быть присвоены в течение жизни протокола. Поэтому стал популярным бесклассовый способ использования IP.

Хотя бесклассовые протоколы тоже используют один адрес для представления как сети, так и машины, но их поведение неизменно. Часть адреса, относящаяся к машине, всегда имеет одинаковую длину. Поэтому маршрутизирующие устройства могут легко и быстро определять сетевые биты адреса, которые никогда не меняются.

**CIDR**

В последние несколько лет широкое распространение получил протокол CIDR (Classless Inter-Domain Routing — бесклассовая междоменная маршрутизация). Разработанный с целью помочь удовлетворить растущий спрос на IP-сети класса B, CIDR является бесклассовой формой IP.

Когда маршрутизаторы сконфигурированы для использования CIDR, сетевые IP-адреса сгруппированы в суперсети. Внутри структуры IP-адреса биты сетевой части передаются адресной части машины, что обеспечивает более гибкую схему адресации.

Эффект создания суперсетей заключается в том, что возникает возможность маршрутизации информации к любой машине группы сетей на основе адреса суперсети. Адрес суперсети используется для определения того, какие сети принадлежат к конкретной группе.

**Инкапсуляция**

Протоколы способствуют движению данных из системы в систему, используя процедуру под названием *инкапсуляция*. В процессе инкапсуляции данные, пересылаемые из одной системы в другую, упаковываются так, чтобы промежуточные устройства сетевого взаимодействия могли определить, куда направляются данные.

Инкапсуляция представляет собой процесс деления сплошного потока данных на небольшие сегменты одинакового размера. Эти маленькие одинаковые пакеты легче отправлять и получать, так как их размер легко спрогнозировать всем системам, использующим один и тот же протокол. Поэтому сеть может настраиваться на работу с сегментированными данными конкретного размера.

Во время инкапсуляции протокол присоединяет к каждому сегменту или пакету свой собственный заголовок.

Имя, которое дается каждому меньшему пакету данных, зависит от протокола и от уровня модели OSI, на котором он работает. Одни протоколы работают с пакетами, другие используют сегменты, кадры или ячейки. Но все это разные имена для одного и того же: подвергнувшегося инкапсулированию сегмента данных, готового к маршрутизации.

Содержимое заголовка зависит от протокола. (Поля заголовка будут описаны при рассмотрении конкретных протоколов в следующем разделе. Заголовки IP обсуждались в раздел [Перемещение данных маршрутизаторами](https://it.rfei.ru/course/%7EGZqN/-/%7Egq7X).) Но несколько полей присутствуют в заголовках почти всех протоколов.

Два из них — это адрес отправителя и адрес получателя. Эти адреса сообщают маршрутизатору и другим устройствам сетевого взаимодействия, откуда отправлен пакет данных и куда его необходимо доставить.

Следующее поле — это порядковый номер. Данное поле очень важно для инкапсуляции. Так как протоколы берут большие порции данных и делят их на более мелкие, легко управляемые части, то система должна уметь определять, как вновь собрать из них поток данных. (Помните, что маршрутизаторы работают на сетевом уровне без установления соединений, поэтому пакеты почти всегда прибывают на место назначения не в том порядке, в каком были отправлены. Обеспечением того, чтобы пакеты были снова собраны в правильном порядке, занимаются маршрутизируемые протоколы.)

Еще одно необходимое поле содержит размер передаваемого пакета данных.

Последнее поле, о котором мы сейчас поговорим, — это контрольная сумма. По существу, контрольная сумма представляет (в краткой и легко воспроизводимой форме) содержимое передаваемого пакета. После получения пакета (но до его обратной сборки) устройство назначения на основе полученных данных вычисляет контрольную сумму заново и сравнивает ее с полученной контрольной суммой, чтобы определить, не был ли пакет поврежден во время пересылки. Эта функция маршрутизируемых протоколов также жизненно важна для правильной работы маршрутизаторов.

Этот короткий, но достаточно подробный рассказ о маршрутизируемых протоколах служит введением в оставшуюся часть книги, которая познакомит вас непосредственно с протоколами и расскажет, как маршрутизаторы Cisco их используют. Теперь же мы углубимся в конфигурирование маршрутизатора Cisco с тем, чтобы он смог использовать свой первый протокол — IP.

**Проверка знаний: Введение в маршрутизируемые протоколы**

**Задание 1**

Чем занимается *уровень приложений* (уровень 7)?

координированием связи между приложениями

преобразованием информации в формат, который понимают другие уровни

координированием связи между сетевыми устройствами

**Задание 2**

За что отвечает *транспортный уровень* (уровень 4)?

за обеспечение корректности информации в карте за счет адресации

за получение пользовательских данных от верхних уровней и разбиение их на удобные для передачи порции

за создание логической карты сети

**Задание 3**

В чем заключается функция *канального уровня* (уровень 2)?

в приеме кадров от верхних уровней и передаче их в виде битов по среде сети передачи данных.

в координировании связи между приложениями

в обеспечении корректности информации в карте за счет адресации

**Задание 4**

В чем отличие бесклассового IP от классового IP?

не существует предела количества адресов

в скорости обработки данных

нет существенной разницы

**Задание 5**

В чем заключается функция *сетевого уровня* (уровень 3)?

в создании логической карты сети

в координировании связи между сетевыми устройствами

в обеспечении корректности информации в карте за счет адресации

**Задание 6**

Содержимое заголовка зависит от:

протокола

адреса отправителя

адреса получателя

**Задание 7**

В чем явные преимущества использования протоколов без установления соединений?

простота в поиске и устранении неисправностей

скорость и затрата меньших ресурсов

надежность

**Задание 8**

Зачем нужен *сеансовый уровень* (уровень 5)?

для координирования связи между приложениями

для координирования связи между сетевыми устройствами

для получения пользовательских данных от верхних уровней и разбиения их на удобные для передачи порции

**Задание 9**

В чем заключается *функция представлений* (уровень 6)?

в координировании связи между приложениями

в получении пользовательских данных от верхних уровней и разбиение их на удобные для передачи порции.

в преобразовании информации уровня приложений в формат, который понимают другие уровни.

**Задание 10**

Что делает *физический уровень*?

создает логическую карту сети

преобразует информацию уровня приложений в формат, который понимают другие уровни

принимает кадры от верхних уровней и передает их в виде битов по среде сети передачи данных

**Задание 11**

Верно ли утверждение: *«Не все протоколы можно маршрутизировать»*?

Верно

Не верно

**Изучение основ IP**

TCP/IP (Transmission Control Protocol/Internet Protocol — протокол управления передачей данных/протокол Интернета) — это самый популярный на сегодняшний день стек сетевых протоколов. Далее даются основы, которые помогут вам понять, как работает IP.

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). Он классифицируется как протокол третьего уровня по сетевой модели OSI. IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня — транспортного уровня сетевой модели OSI, — например, TCP, которые используют IP в качестве транспорта.

Именно IP стал тем протоколом, который объединил отдельные компьютерные сети во всемирную сеть Интернет. Неотъемлемой частью протокола является адресация сети.

В предыдущем разделе говорилось, что в терминах модели OSI TCP относится к протоколам транспортного уровня, в то время как IP — протокол сетевого уровня. Два протокола дополняют друг друга, и, чтобы осознать целостную картину, необходимо понять, как работают они оба. Но так как маршрутизатор работает только с одним протоколом стека TCP/IP — IP, то здесь мы не будем говорить о TCP.

* [IP (Internet Protocol)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a0)
* [Адреса класса A](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a1)
* [Адреса класса B](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a2)
* [Адреса класса C](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a3)
* [Маска подсети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a4)
* [Организация подсетей](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a5)
* [Надсеть IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a6)
* [Проверка знаний: Изучение основ IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904a7)

**IP (Internet Protocol)**

Сегодня практически невозможно работать в компьютерной индустрии и не знать, что такое IP. Но понимаете ли вы на самом деле, как он работает и почему он настолько популярен? В этом разделе мы поговорим о работе IP, об адресации и организации подсетей.

IP облегчает работу сетевого уровня по созданию карты сетевого окружения. Используя адресную IP-схему, сетевой уровень может составить подробную картину сети вокруг хоста. IP работает только на сетевом уровне.

Стандартный IP-адрес выглядит так 128.95.95.178 — четыре части, содержащие по байту каждая. IP-адрес состоит из 32 битов или 4 байтов. Так как биты двоичны, то максимальное значение, которое достижимо в пределах одного байта, — это 255. То есть возможный диапазон IP-адресов выглядит как 0.0.0.0-255.255.255.255. (Эти числа достижимы, но не все они могут быть использованы.) Будьте осторожны и не используйте для своих маршрутизаторов один из зарезервированных адресов.

Список IP-адресов, которые считаются зарезервированными, приведен в табл. 4.1.

Таблица 4.1. Зарезервированные IP-адреса

| **Адрес** | **Двоичный** | **Причина резервирования** |
| --- | --- | --- |
| 0.0.0.0 | 00000000.00000000. 00000000.00000000 | Адрес не может состоять из одних нулей. Используется протоколом RIP для маршрутизации |
| 255.255.255.255 | 11111111.11111111. 11111111.11111111 | Адрес не может состоять из одних единиц. Используется для широковещания |
| 127.0.0.1 | 01111111.00000000. 00000000.00000001 | Зарезервировано для внутреннего петлевого контроля |

В зависимости от класса адреса от одного до трех байтов заняты под идентификацию машины и от трех до одного байта используются для определения сети. Первая часть адреса задает сеть, а вторая — машину. Умение отличать хост от сети очень важно для маршрутизации.

Адрес хоста (машины) также иногда называют адресом узла.

Для того чтобы отслеживать количество адресов, выделенных организациям различных размеров, IP-адреса разделены на три класса: A, B и C. Существует также малоизвестный (и еще меньше используемый) класс D, о котором мы не будем рассказывать. (Класс D главным образом используется для многоадресной передачи.) Зная класс адреса, вы сможете правильно определить маску подсети при конфигурировании интерфейса.

**Адреса класса A**

У адресов класса A первый байт адреса представляет собой сеть, а остальные три байта — узлы.

Разбиение адреса класса A на адрес сети и адрес узла изображено на рис. 4.1.

Рис. 4.1. Адрес класса A

Сетевые адреса класса A имеют диапазон от 1 до 127. Следовательно, IP-адрес, начинающийся с числа, входящего в этот диапазон, относится к классу A. Поясним это на двоичном уровне. Первый бит первого октета адреса класса A – это всегда 0. Например, адрес 127.0.0.0 в двоичной записи выглядит так: 01111111.00000000.00000000.00000000. Заметьте, что первый бит равен 0.

**Адреса класса B**

В адресах класса B сеть представляют два первых байта адреса, а узел – два других. Первый бит адреса класса B имеет значение от 128 до 191. IP-адрес класса B, разделенный на адрес сети и адрес узла, изображен на рис. 4.2.

Рис. 4.2. Адрес класса B

Исследуя двоичную запись адреса, мы обнаружим, что все адреса класса B начинаются с 10, то есть два первых бита первого октета – это всегда 10. Например, адрес 140.75.0.0 соответствует двоичному 10001100.01001011.00000000.00000000. Первые два бита – это 10.

**Адреса класса C**

В адресах класса C три первых байта определяют сеть, а последний – узел. Разрешенные значения первого байта сетевого адреса в данном случае входят в диапазон от 192 до 223. Разбиение адреса класса C на адрес сети и адрес узла изображено на рис. 4.3.

Рис. 4.3. Адрес класса C

Давайте посмотрим на адрес класса C в двоичном формате: 198.40.50.0 соответствует двоичному адресу 11000110.00101000.00110010.00000000. Все адреса класса C начинаются с трех цифр: 110.

Вы могли обратить внимание на то, что верхняя граница разрешенных значений для адресов класса C равна 223, в то время как IP-адреса могут достигать значения 256. Такой зазор оставлен для класса D (223-239) и класса E (240-255). Не тратьте время на раздумья об этих адресах, они нечасто используются.

О существовании таких диапазонов полезно знать при поиске и устранении неисправностей. Например, если вы конфигурируете интерфейс маршрутизатора для сети класса C и присваиваете ему адрес 230.230.230.0 с маской подсети 255.255.255.0 (правильная маска подсети для сети класса C), то можете столкнуться с проблемами.

В данном конкретном случае устройство не сможет взаимодействовать ни с какими другими устройствами подсети 255.255.255.0. На первый взгляд адрес 230.230.230.0 выглядит вполне законно, хотя на самом деле это адрес класса D.

**Маска подсети**

Маска подсети используется IP для того, чтобы различать адрес сети и адрес узла. Чтобы понять, как работает маска подсети, преобразуем IP-адрес в двоичный формат. Возьмем, к примеру, адрес класса C – 198.68.85.114:

11000110.01000100.01010101.01110010

Маска подсети для IP-адреса класса C выглядит как 255.255.255.0, переведем и ее в двоичный формат:

11111111.11111111.11111111.00000000

Как она может помочь протоколу IP узнать, где сетевой адрес, а где адрес узла? Если вы посмотрите на двоичное представление адреса и маски, то увидите, что сеть обозначена единицами, в то время как часть, относящаяся к адресу узла, состоит из нулей. Теперь все кажется очевидным, но когда вы начинаете организовывать подсети в своей сети, оказывается, что есть и сложности.

Используемый класс IP-адресов определяет, какую маску вам следует применять. Приведем существующие по умолчанию маски подсети для трех классов адресов:  
Класс A = 255.0.0.0  
Класс B = 255.255.0.0  
Класс C = 255.255.255.0

Организация подсетей — это техническая операция, которой вам точно не удастся избежать при работе с маршрутизаторами Cisco. Чем больше вы этим занимаетесь, тем понятнее становится, как организовывать подсети, но такая задача может быть и сложной, а в некоторых случаях и мучительной.

Мы бы предложили вам сначала нарисовать схемы своих подсетей на бумаге, а потом уже реализовывать их. Но давайте все же немного поговорим о процессе разбиения сети на подсети, чтобы познакомить вас с данным вопросом.

**Организация подсетей**

При разделении сети на подсети биты из той части адреса, которая задает адрес узла, передаются в сетевую часть адреса. Благодаря этому одна лицензия на IP-сеть может использоваться для адресации более чем одной сети (подсети). На рис. 4.4 показана сеть, в которой организация подсетей принесла бы пользу.

Рис. 4.4. Сеть, не разбитая на подсети

Требуется разбить лицензию 10.0.0.0 так, чтобы получилось три подсети. С чего начать? Начнем с того, что посмотрим на маску подсети. (Даже если вы создаете три сети, у них должна быть общая маска подсети.) Если преобразовать маску подсети в двоичный формат, то легче увидеть, как организуются подсети.

Маска подсети для 10.0.0.0 — 255.0.0.0 или (в двоичном виде)

11111111.00000000.00000000.00000000.

Теперь следует определить количество бит, которые будут переданы сетевому адресу, чтобы предусмотреть еще три сети. Предположим, например, что мы передаем два бита из той части маски, которая соответствует адресу узла, в сетевую часть маски:

11111111.11000000.00000000.00000000

Получается новая маска подсети — 255.192.0.0.

При подсчете количества бит, которые должны быть переданы в сетевую часть адреса, чтобы обеспечить создание требуемого количества подсетей, можно использовать такое уравнение:

2*x* – 2 = количество адресов (где *x* — количество бит в адресе)

Во втором байте рассмотренного примера мы использовали два бита для сетевого адреса. Следовательно, количество сетей равно 22 – 2 = 2. Для адресов узлов осталось всего 22 бита, значит, их общее количество равно 222 – 2 = 4 194 302.

Почему следует вычесть 2 из общего количества адресов? Дело в том, что использование адресов, состоящих из одних единиц или нулей, не разрешено, поэтому необходимо вычесть два эти варианта.

Второе уравнение применяется для вычисления сетевых адресов. Размер интервала между сетевыми адресами равен разности 256 и новой маски подсети. Каждый сетевой адрес должен быть кратен этой разности, при этом быть положительным и не превышать значение маски подсети:

*i*-й сетевой адрес *yi = i* \* (256 – *x*), *i* = 1, 2, . . . , [меньше чем] *x*

где *x* — это новая маска подсети. В нашем примере маска подсети равна 192, значит

256 – 192 = 64

Следовательно, допустимыми являются сети 10.**64**.0.0 и 10.**128**.0.0 (128 = 64 + 64). Если еще раз прибавить 64, то получится 192, то есть наша маска подсети.

Чтобы успешно разбить на подсети пример, представленный на рис. 4.4, необходимы три сети. Два бита, добавленные к сетевому адресу, не привели к образованию достаточного количества сетей. Если же добавить три бита в маску подсети, чтобы получить 255.224.0.0 (224 = 128 + 64 + 32), то образуется шесть сетей (23 – 2), по 2 097150 узлов в каждой. Очевидно, что нам не потребуется так много узлов, но прежде всего нас интересует количество сетей.

С помощью организации подсетей можно лучше управлять трафиком между ними. Если не создавать подсети, то трафик (порождаемый широковещательными пакетами и CSMA/CD-пакетами) начинает отрицательно влиять на производительность сети. После же организации подсетей любой поток информации, у которого и отправитель и адресат находятся в одной подсети, остается в пределах этой подсети, «не лихорадя» сетевое окружение.

При создании подсетей биты из адреса узла передаются в сетевую часть адреса, при организации надсетей этот процесс происходит с точностью до наоборот. Образуется «надсеть» высшего уровня, которой принадлежат все ваши сети. Создание надсети применяется при бесклассовой междоменной маршрутизации (CIDR).

**Надсеть IP**

В связи с растущей популярностью CIDR появилась необходимость объединения группы сетей в надсеть. Так как административные процедуры выделения адресов мучительно медленно работают с большими классами IP-адресов, потребовалось найти способ продлить жизнь оставшихся адресов. Так и пришли к созданию надсетей.

Когда группа адресов объединяется в надсеть, создается маска, которая показывает, что отдельные сети принадлежат к одной большой супергруппе. Например, если окружению присвоены два сетевых адреса: 215.50.25.0 и 215.50.26.0, то можно создать надсеть, которая бы объединила и связала две сети для использования в рамках одной и той же физической среды.

Двоичная запись маски подсети 255.255.255.0 (маска класса C для обеих упомянутых сетей) выглядит как 11111111.11111111.11111111. 00000000. При организации подсетей биты передаются из адреса узла в сетевую часть адреса, а при создании надсети биты «покидают» сетевую часть адреса, чтобы уменьшить количество сетей.

Если передать два бита из сетевой части адреса в адрес хоста, то маска превратится в 11111111.11111111.11111100.00000000 (или 255.255.252.0). Будучи наложенной на сетевой адрес 215.50.25.0, эта маска сообщит маршрутизатору, что две сети включены в надсеть. Если адреса сетей начинаются с 215.50.25.0, то маска распространяется на две сети: 215.50.25.0 и 215.50.26.0.

Вспомните, если использовать те же вычисления, которые были представлены в начале раздела, то получится, что передача двух битов из адреса узла в сетевую часть адреса включит в состав вашей надсети две (то есть 22 – 2) сети.

Если вы беспокоитесь о том, как маршрутизатор узнает, что новая маска 255.255.252.0 относится не к подсети класса B, а к надсети класса C, то ответ очень прост. Первый октет сетевого адреса выглядит как 215.x.x.x. Только адрес сети класса C может начинаться с 215; поэтому новая маска должна относиться к надсети класса C.

Легче понять надсети, если рассматривать их во взаимосвязи с подсетями. Чтобы поднатореть в этом вопросе, попрактикуйтесь на бумаге в создании как подсетей, так и надсетей.

**Проверка знаний: Изучение основ IP**

**Задание 1**

Как выглядит двоичная запись маски подсети 255.255.255.0?

11111111.11111111.11111111.00000000

11111111.00000000.00000000.00000000

11111111.11111111.11111100.00000000

**Задание 2**

Что означает х в уравнении 2˟ – 2 для создания требуемого количества подсетей?

количество бит в адресе

количество маршрутизаторов

количество пользователей

**Задание 3**

Какое максимальное значение, достижимое в пределах одного байта, может быть использовано в IP-адресе?

255

144

512

**Основы и структура межсетевого взаимодействия**

В предыдущих разделах были рассмотрены механизмы передачи данных по сетям, которые лежат в основе любой компьютерной коммуникационной системы.

В этом разделе мы сосредоточимся на способах объединения разрозненных сетевых технологий в одну согласованную систему. Цель заключается в том, чтобы создать систему, обеспечивающую универсальные средства взаимодействия и не зависящую от использованного в ней сетевого оборудования. Поэтому сначала мы опишем абстрактную модель высокого уровня, которая лежит в основе всех проектных решений.

В последующих разделах будет показано, как на основе этой абстракции создаются требуемые уровни приложений, которые обеспечивают универсальный и независимый от физического транспортного уровня механизм передачи данных. Кроме того, в следующих разделах будет рассмотрено использование созданной нами коммуникационной системы в прикладных программах.

* [Взаимодействие на уровне приложений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ae)
* [Взаимодействие на сетевом уровне](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904af)
* [Особенности объединенной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b0)
* [Структура объединенной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b1)
* [Объединение сетей с помощью IP-маршрутизаторов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b2)
* [Преимущества взаимодействия на сетевом уровне](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b3)
* [Все сети равноправны](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b4)
* [Проверка знаний: Основы и структура межсетевого взаимодействия](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904b5)

**Взаимодействие на уровне приложений**

При проектировании независимых от сетевого аппаратного обеспечения систем передачи данных используется два различных подхода. Один из них заключается в том, что для скрытия неоднородностей в системе и особенностей низкоуровневых сетевых технологий используются специальные программы. В другом случае эти программы встраиваются прямо в операционную систему. Раньше разнородные сети объединялись в единообразную систему с помощью специальных прикладных программ, которые назывались *шлюзами уровня приложений (application gateways)*. В подобных системах на каждом подключенном к сети компьютере запускалась специальная прикладная программа, которая обрабатывала особенности сетевого оборудования и обеспечивала сетевое соединение для этого компьютера. Кроме того, эта программа взаимодействовала с аналогичными программами, запущенными на других компьютерах сети. Например, некоторые системы электронной почты состоят из набора прикладных программ, запущенных на компьютерах сети. Каждая из этих программ настраивается на передачу сообщений соседнему компьютеру. В результате путь, по которому проходит сообщение от отправителя до получателя, может быть очень длинным (включая узлы, принадлежащие другим сетям). Однако на доставку сообщений это никак не влияет, поскольку почтовые системы всех компьютеров могут взаимодействовать друг с другом.

Использование специальных прикладных программ для скрытия деталей низкоуровневого сетевого взаимодействия может на первый взгляд показаться вполне естественным. Однако подобный подход сдерживает дальнейшее развитие системы и делает процесс обмена данными чрезвычайно громоздким. В самом деле, чтобы расширить функциональные возможности системы, необходимо для каждого компьютера создать новую прикладную программу. Добавление нового сетевого оборудования требует модификации существующих программ или создания новых программ для всех возможных приложений. Таким образом, каждая прикладная программа компьютера должна самостоятельно обеспечивать для себя сетевое соединение, что неминуемо влечет за собой дублирование кодов этих программ.

Разбирающиеся в сетях пользователи понимают, что с ростом числа взаимодействующих сетей до нескольких сотен или тысяч уже никто не сможет создать необходимый на все случаи жизни набор прикладных программ. Более того, описанный выше принцип пошагового обмена данными на уровне приложений требует корректной работы программ на всех компьютерах, участвующих в этом процессе. Если что-то случится с программой на одном из промежуточных звеньев цепи, ни отправитель, ни получатель уведомлены не будут и никак не смогут повлиять на устранение возникшей проблемы. Следовательно, системы, использующие промежуточные программы для реализации механизма взаимодействия, не могут гарантировать надежный обмен данными.

**Взаимодействие на сетевом уровне**

Кроме систем, взаимодействующих на уровне приложений, существует и альтернативный вариант — системы, взаимодействующие между собой на сетевом уровне. Сетевой механизм взаимодействия обеспечивает доставку небольших пакетов данных от отправителя прямо к получателю без использования промежуточных прикладных программ. Пересылка небольших фрагментов данных вместо огромных и громоздких файлов обладает рядом преимуществ.

Во-первых, в нем непосредственно участвует низкоуровневое сетевое аппаратное обеспечение, что делает его чрезвычайно эффективным.

Во-вторых, взаимодействие на сетевом уровне позволяет отделить логику приложения от механизма пересылки данных. В результате компьютеры на промежуточных узлах могут оперировать трафиком, не вникая, для каких сетевых приложений он предназначен.

В-третьих, использование взаимодействия на сетевом уровне позволяет создать универсальную систему, предназначенную для выполнения практически любых видов компьютерной связи.

В-четвертых, структура системы позволяет легко управлять сетью. Например, при появлении новых сетевых технологий требуется только внести соответствующие изменения в программное обеспечение сетевого уровня, не затрагивая при этом самих прикладных программ.

Идея реализации универсальной коммуникационной системы, взаимодействующей на сетевом уровне, вытекает из абстрактной концептуальной модели *объединенной сети (internetworking)*.

Понятие объединенной сети *(internetwork*, или *internet*), — чрезвычайно емкое. Оно позволяет выполнять процесс взаимодействия без учета подробностей реализации конкретной сетевой технологии, и скрыть от пользователя все низкоуровневые моменты. Более того, это понятие определяет принципы разработки программного обеспечения и проясняет методы обработки физических адресов и маршрутизации. Поэтому после рассмотрения основных причин принятия модели межсетевого взаимодействия, будут подробно описаны ее особенности.

Прежде всего сформулируем два основных постулата, которые следует иметь в виду при разработке коммуникационных систем.

* Не существует одной универсальной сетевой технологии, которая удовлетворяла бы всем требованиям.
* Идея универсального межсетевого взаимодействия как нельзя лучше соответствует пожеланиям пользователей.

Первый постулат основан на экономических, а также чисто технических соображениях. Недорогие сетевые технологии, применяемые для создания высокоскоростных локальных сетей, могут работать только на небольших расстояниях. В то же время стоимость глобальных коммуникационных технологий, применяемых при создании протяженных сетей, очень высока. По этой причине они крайне редко используются в локальных сетях. Поскольку ни одна сетевая технология не может удовлетворить всем требованиям, мы рассмотрим несколько основных низкоуровневых сетевых технологий.

Второй постулат очевиден. Пользователям хотелось бы организовать взаимодействие между двумя любыми узлами сети. В частности, хотелось бы иметь такую коммуникационную систему, которая не была бы связана рамками физических сетей.

Итак, наша конечная цель — создать унифицированную и согласованную объединенную сетевую систему, которая бы поддерживала ряд универсальных сетевых служб. В сетях такой системы могут использоваться низкоуровневые технологии передачи данных. Поэтому очевидно, что нужно разработать программное обеспечение и поместить его в иерархической модели между физическим уровнем передачи данных, зависящим от применяемой аппаратной технологии, и уровнем приложений. Это программное обеспечение как раз и будет той ширмой, за которой спрячутся низкоуровневые особенности применяемого сетевого оборудования. Кроме того, с его помощью можно будет связать набор разнородных сетей в одну большую сеть. Описанный принцип межсетевого взаимодействия называется *объединенной сетью (internetwork*, или *internet*).

Идея создания объединенной сети соответствует стандартному подходу, применяемому при проектировании сложных систем. Разработчики придумывают высокоуровневую компьютерную систему, для реализации которой используются существующие компьютерные технологии. При этом создаются необходимые уровни программного обеспечения, которые эмулируют с требуемой эффективностью возможности высокоуровневой системы. В следующем разделе описывается первый этап процесса проектирования, на котором выполняется более конкретная постановка задачи.

**Особенности объединенной сети**

Безусловно, сама идея создания набора универсальных служб очень важна. Однако на ней не оканчивается список идей, которые было бы неплохо реализовать в объединенной сети, поскольку универсальным службам можно найти множество применений. При постановке задачи преследовалась цель скрыть от пользователя особенности низкоуровневой структуры объединенной сети. Другими словами, мы не хотим заставлять конечного пользователя или прикладную программу учитывать низкоуровневые особенности аппаратного обеспечения при использовании объединенной сети. Кроме того, нежелательно, чтобы взаимодействие зависело от топологии объединенной сети. В частности, добавление новой сети к системе не должно быть связано с ее подключением к центральному узлу коммутации или с прокладкой физических каналов связи ко всем существующим сетям. Идея заключается в том, чтобы можно было передавать данные между компьютерами отправителя и получателя по промежуточным сетям, даже если последние напрямую не соединены с сетями, в которых находятся взаимодействующие абоненты. Также желательно, чтобы всем компьютерам объединенной сети назначался универсальный набор идентификационных параметров, которые можно считать их *именами*, или *адресами*.

В концепцию унифицированной объединенной сети также следует включить идею независимости пользовательского интерфейса от сети. Другими словами, нужно сделать так, чтобы набор действий, выполняемых при установке соединения или передаче данных, всегда оставался однотипным и не зависел от типа используемой сетевой технологии и компьютера получателя. Само собой разумеется, что при создании или использовании прикладных программ, взаимодействующих друг с другом, пользователь не должен вникать в тонкости топологии связанной сети.

**Структура объединенной сети**

В предыдущих разделах были рассмотрены способы подключения компьютеров к самостоятельным сетям. Теперь возникает закономерный вопрос: “Как соединить между собой эти сети, чтобы они составляли единую сеть?” Ответ на него состоит из двух частей. Физически две сети можно соединить только при помощи компьютера, который подключен к обеим из них. Однако физическое соединение не обеспечивает межсетевое взаимодействие, идея которого была высказана выше. Причина в том, что подобное соединение не может гарантировать взаимодействие компьютеров, находящихся в разных сетях. Для того чтобы объединенная сеть заработала, необходимо выделить специальный компьютер, который будет передавать пакеты из одной сети в другую. Компьютеры, связывающие две сети и выполняющие пересылку пакетов между ними, называются *межсетевыми шлюзами (internet gateways)* или *маршрутизаторами1 (internet routers)*.

В качестве примера рассмотрим объединение двух физических сетей, изображенных на рис. 5.1.

Рис. 5.1. Пример объединения двух физических сетей с помощью маршрутизатора R (1Р-шлюза)

Как видно из рисунка, маршрутизатор R подключен как к сети 1, так и к сети 2. Поскольку R является маршрутизатором, он должен перехватывать все пакеты, посланные из сети 1 компьютеру, находящемуся в сети 2, и выполнять передачу этих пакетов между сетями. Аналогично, маршрутизатор R должен перехватывать пакеты, посланные из сети 2 компьютеру находящемуся в сети 1, и выполнять передачу этих пакетов между сетями.

На рис. 5.1. физические сети обозначены в виде облака; этим подчеркивается, что не имеет значения, какое сетевое оборудование в них используется. Эти сети могут быть как локальными, так и глобальными, причем количество компьютеров в них может быть любым.

1. Раньше в литературе часто можно было встретить термин *IP-шлюз (IP gateway)*. Поскольку этот термин прижился у производителей сетевого оборудования, мы иногда будем им пользоваться как синонимом.

**Объединение сетей с помощью IP-маршрутизаторов**

Хотя на рис. 5.1 продемонстрирован основной принцип объединения сетей, все же показанную конфигурацию сетей можно считать очень упрощенной. На самом деле сложные объединенные системы состоят из большого количества сетей и маршрутизаторов. При этом каждый маршрутизатор должен обладать информацией о топологии объединенной системы, находящейся за пределами той сети, к которой он подключен.

Давайте рассмотрим в качестве примера объединение трех сетей с помощью двух маршрутизаторов (рис. 5.2).

Рис. 5.2. Пример объединения трех сетей с помощью двух маршрутизаторов

В этом примере маршрутизатор R1 должен передавать из сети 1 в сеть 2 все пакеты, предназначенные для компьютеров, которые подключены как к сети 2, так и к сети 3. А теперь представьте себе большую объединенную сеть, состоящую из множества сетей. Очевидно, что в последнем случае принятие решения маршрутизатором о том, куда направлять пакеты, становится более сложной задачей.

На первый взгляд маршрутизаторы кажутся очень простыми устройствами, но от этого их значение не становится меньше, поскольку они обеспечивают средства для объединения сетей, а не просто отдельных компьютеров. Таким образом, только что мы установили важный принцип создания объединенной сети.

При создании объединенной сети на основе семейства протоколов TCP/IP, взаимодействие между сетями обеспечивают специальные компьютеры, называемые *IP-маршрутизаторами*, или *IР-шлюзами*.

Может показаться, что устройства, направляющие пакеты к получателям, должны быть мощными компьютерами с достаточным объемом внутренней и внешней памяти для хранения информации о каждой подключенной к объединенной сети машине. На самом деле маршрутизаторы, используемые в сетях TCP/IP, как правило, являются небольшими компьютерами. Часто они имеют жесткий диск небольшого размера и весьма скромный объем оперативной памяти. Это возможно благодаря тому, что маршрутизация пакетов выполняется в соответствии с приведенным ниже правилом.

При перенаправлении пакетов маршрутизаторы, используют не адрес получателя пакета, а адрес сети, в котором расположена машина получателя.

Таким образом, если при перенаправлении пакетов исходить из адреса сети, то количество информации, которое должен хранить маршрутизатор, будет пропорционально количеству сетей в объединении, а не количеству подключенных к ним машин.

Поскольку маршрутизаторы играют основную роль в межсетевом взаимодействии, мы еще вернемся к ним в следующих разделах и подробнее опишем принцип их работы и методы, с помощью которых они получают маршрутную информацию. А пока можно считать, что существует реальная возможность хранить в каждом маршрутизаторе объединенной сети информацию о маршрутах следования пакетов ко всем ее сетям. Кроме того, мы будем полагать, что средства взаимодействия между физическими сетями обеспечиваются только с помощью маршрутизаторов.

**Преимущества взаимодействия на сетевом уровне**

Напомним, что семейство протоколов TCP/IP было разработано для обеспечения универсального взаимодействия между компьютерами, не зависящего от типов конкретных сетей, к которым эти компьютеры подключены. Поэтому с точки зрения пользователя такая объединенная сеть выглядит как единая виртуальная сеть, к которой подключаются все компьютеры. При этом детали конкретного физического подключения не имеют значения. Поэтому, чтобы облегчить конечному пользователю понимание механизма взаимодействия, объединенная сеть должна представляться в виде единой сети, а не совокупности отдельных сетей, как показано на рис. 5.3 (а). Для реализации такого подхода маршрутизаторов, связывающих физические сети, недостаточно. На каждом компьютере объединенной сети должно быть установлено также специальное программное обеспечение, с помощью которого прикладные программы смогут использовать объединенную сеть так, как если бы это была одна физическая сеть.

Рис. 5.3. Объединенная сеть TCP/IP с точки зрения конечного пользователя — предполагается, что каждый компьютер подключен к одной большой сети (а); схема подключения физических сетей с помощью маршрутизаторов (б)

Таким образом, преимущества взаимодействия на сетевом уровне понятны. Поскольку для прикладных программ, взаимодействующих между собой по объединенной сети, не имеет значения какова внутренняя структура механизма организации соединений, они могут запускаться без каких бы то ни было изменений на любом компьютере. Кроме того, так как детали физического подключения каждого компьютера к сети обрабатывает специальное межсетевое программное обеспечение, то при изменении топологии сети (добавлении или удалении каналов связи), изменения вносятся только в сетевые программы. Фактически становится возможным оптимизировать внутреннюю структуру объединенной сети за счет изменения топологии физических каналов связи даже без остановки выполнения прикладных программ.

Второе преимущество взаимодействия на сетевом уровне заключается в его удобстве. Пользователи не должны знать структуру сетевых подключений и какие потоки данных по ним передаются. При написании прикладных программ, взаимодействующих по сети, физическую структуру сетевых подключений можно не учитывать. По сути, сетевые администраторы могут свободно изменять внутреннюю структуру объединенной сети, не изменяя прикладное программное обеспечение на большинстве компьютеров, подключенных к ней. Естественно, что при перемещении компьютера в другую физическую сеть придется изменить настройки его сетевого программного обеспечения.

Как видно из рис. 5.3 (б) между некоторыми парами сетей может не быть прямого соединения, обеспечиваемого маршрутизаторами. Поэтому для взаимодействия двух компьютеров, расположенных в таких сетях, нужно передавать пакеты через несколько маршрутизаторов и несколько промежуточных сетей.

Таким образом, взаимодействующие сети можно сравнить с большой автомагистралью между городами. Каждая сеть берет на себя обязательство передавать транзитные потоки данных по своим каналам связи в обмен на право посылать в объединенную сеть собственный трафик. При этом рядовые пользователи могут и не подозревать, что по их локальной сети проходит дополнительный трафик.

**Все сети равноправны**

Ранее был приведен краткий обзор сетевого оборудования, которое может использоваться для создания взаимодействующих между собой сетей TCP/IP, и продемонстрировано большое разнообразие существующих сетевых технологий. Выше отмечалось, что объединенная сеть состоит из набора связанных между собой равноправных сетей. На данный момент важно уяснить основной принцип: с точки зрения объединенной сети любая коммуникационная система, способная передавать пакеты, рассматривается как одна сеть, независимо от ее пропускной способность, задержек при передаче пакетов, максимального размера пакета и географической протяженности. В частности, на рис. 5.3 (б) физические сети обозначены небольшими облачками одинакового размера, поскольку, несмотря на их отличия, с точки зрения семейства протоколов TCP/IP они считаются равноправными. Следовательно можно сделать важный вывод.

С точки зрения семейства протоколов TCP/IP все сети являются равноправными. Одной сетью считается и локальная сеть Ethernet, и глобальный магистральный канал и двухточечное соединение между двумя компьютерами.

Пользователь, не разбирающийся в структуре межсетевых взаимодействий, может не понять такого упрощенного подхода к сетям. По существу в семействе протоколов TCP/IP определено абстрактное понятие “сети”, которое не зависит от физических особенностей конкретной сети. Ниже будет показано, что подобная абстракция делает протокол TCP/IP чрезвычайно мощным средством взаимодействия.

**Проверка знаний: Основы и структура межсетевого взаимодействия**

**Задание 1**

Все ли сети являются равноправными с точки зрения семейства протоколов TCP/IP?

нет

да

**Задание 2**

Продолжите верно фразу.

Системы, использующие промежуточные программы для реализации механизма взаимодействия, не могут гарантировать...

безопасность данных

надежный обмен данных

высокую пропускную способность

**Задание 3**

При перенаправлении пакетов маршрутизаторами, какой адрес они используют?

адрес отправителя пакета

адрес получателя пакета

адрес сети

**Задание 4**

Как называются компьютеры, связывающие две сети и выполняющие пересылку пакетов между ними?

администраторы

межсетевые шлюзы

концентраторы

**Задание 5**

Что делает чрезвычайно эффективным сетевой механизм?

низкоуровневое сетевое аппаратное обеспечение

наличие сильной системы безопасности

надежный обмен данных

**Перемещение данных маршрутизаторами**

Цель этого раздела — дать представление о процессах, происходящих при перемещении данных от системы к системе.

Мы обсудим роль протоколов в процессе маршрутизации и то, как они используются при передаче данных по сети.

В разделе рассматриваются следующие темы:

* Маршрутизаторы и сетевые уровни.
* Протоколы маршрутизации.
* Механизмы маршрутизации.
* Маршрутизируемые протоколы и протоколы маршрутизации.

Эти темы представляют широкий диапазон концепций, лежащих в основе процессов передачи данных в сетях. Понимание этих процессов поможет в усвоении материала последующих разделов, непосредственно касающихся маршрутизации и маршрутизируемых протоколов.

После прочтения данного раздела вы должны представлять себе полную картину процессов маршрутизации на многих уровнях. Мы будем рассматривать маршрутизацию с точки зрения модели OSI (Open Systems Interconnect — взаимодействие открытых систем) на уровне протоколов, пакетов и аппаратуры.

Этот раздел открывает путь к пониманию материала последующих разделов. Поняв, как маршрутизаторы перемещают данные, вы легко справитесь с настройкой Cisco IOS для работы с различными протоколами и обеспечения маршрутизации между разными средами.

* [Маршрутизаторы и сетевые уровни](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904be)
* [Маршрутизация протокола](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904bf)
* [Заголовки протокола](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c0)
* [Пакетирование данных](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c1)
* [Механизм маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c2)
* [Маршрутизация в простой сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c3)
* [Маршрутизация в сложной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c4)
* [Таблицы маршрутов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c5)
* [Достижение конвергенции](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c6)
* [Маршрутизируемые протоколы и протоколы маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c7)
* [Проверка знаний: Перемещение данных маршрутизаторами](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904c8)

**Маршрутизаторы и сетевые уровни**

Все применяемые ныне сетевые протоколы передачи данных соответствуют одной общей спецификации. Эти протоколы были разработаны в соответствии с набором требований модели OSI.

Модель OSI — это универсальный каркас, на основе которого различные разработчики создают протоколы, способные взаимодействовать друг с другом. Благодаря такому общему набору правил компьютеры и другие устройства способны общаться между собой независимо от их производителей, разработчиков и платформ.

В обсуждении технологий маршрутизации основополагающей концепцией является способность к взаимодействию. При том количестве различных систем, протоколов и типов данных, которые встречаются в Интернете, производитель не в состоянии предугадать, где и как будут использоваться его маршрутизаторы.

Есть две фундаментальные причины, заставляющие строить все протоколы по одной общей схеме. Во-первых, разработчик, следуя заранее определенным указаниям по реализации протокола, не упустит из вида важных функций. Если, например, команда разработчиков начинает работу над новым транспортным протоколом, она должна следовать правилам, определенным для транспортного уровня модели OSI. Согласно правилам, этот протокол должен быть ориентированным на соединение и уметь (среди прочего) управлять потоком. Выполнив все эти требования, разработчики могут быть уверены в том, что их новый транспортный протокол сможет работать с любым устройством, совместимым с транспортным уровнем OSI. В скоординированной подобным образом среде облегчается создание сетей.

Маршрутизаторы тоже используют преимущества общего подхода. Процесс перемещения данных из одного места в другое полностью основан на принципах модели OSI и инкапсуляции протоколов. Маршрутизаторы Cisco работают на сетевом уровне модели OSI. Это позволяет им взаимодействовать с любым протоколом, соответствующим этой спецификации.

*Способность* маршрутизировать протоколы, относящиеся к определенному сетевому уровню, и фактическая *маршрутизация каждого* протокола — далеко не одно и то же. Другими словами, даже если маршрутизаторы Cisco теоретически и могут работать с любыми протоколами сетевого уровня OSI, это не означает, что IOS каждого конкретного маршрутизатора может быть настроена на выполнение таких операций. Маршрутизатор Cisco может оказаться неспособным обрабатывать некоторый протокол просто потому, что в IOS не предусмотрены необходимые настройки.

То, что все маршрутизаторы работают на сетевом уровне, имеет одно простое объяснение. Именно сетевой уровень отвечает за адресацию протокола. Следовательно, каждый протокол сетевого уровня должен иметь возможность обратиться (адресоваться) к любой доступной системе. Такие адреса и лежат в основе маршрутизации.

Чтобы лучше разобраться в концепциях маршрутизации, адресации и протоколов, приведем пример из повседневной жизни. Например, адрес в протоколе можно сравнить с адресом дома. Этот адрес определяет номер дома, улицу, город и штат:

123 Maple Street

Anytown, Massachusetts

Адрес дома однозначно определяет его местонахождение. Точно так же и компьютеры адресуются протоколами.

Адрес компьютера в некотором протоколе очень похож на адрес дома. Адрес определяет сеть, в которой находится компьютер и его номер. Для доставки любой информации, предназначенной этому компьютеру, достаточно знать его адрес.

Давайте рассмотрим сценарий, определяющий передачу информации от одного компьютера к другому. Если некто хочет послать письмо другу, живущему в соседнем штате, он должен выполнить определенные действия. Отправитель письма должен положить его в конверт. На лицевой стороне конверта следует написать адрес. Затем письмо необходимо отнести в ближайшее почтовое отделение.

В почтовом отделении, получив письмо, прочитают адрес, чтобы определить, куда его следует отправить. Местное почтовое отделение отправляет письмо в почтовое отделение получателя. Затем почтальон приносит письмо адресату на дом.

Не путайте протокольные адреса с физическими адресами. За физическую адресацию отвечает канальный уровень модели OSI. Физические адреса, в частности MAC-адрес, служат уникальными идентификаторами устройств и обычно назначаются аппаратно. Протокольные адреса, в свою очередь, назначаются протоколами и могут совпадать у различных устройств (желательно, чтобы эти устройства находились в разных сетях).

В то время как физический адрес устройства остается неизменным, протокольные адреса зависят от протоколов и могут меняться. Некоторые протоколы, например IPX, используют физический адрес как часть протокольного адреса. И все же не путайте их.

В нашем сценарии почтовое отделение соответствует маршрутизатору. Маршрутизатор читает адрес пункта назначения, присутствующий в каждом пакете, чтобы определить, куда должны отправиться данные. Но, как и почтовое отделение, маршрутизатор сможет прочитать адрес пункта назначения, только если он записан в определенном формате. Формат адреса определяется протоколом.

То есть в то время как маршрутизатор занимается доставкой информации из одного места в другое, протокол отвечает за то, чтобы эта информация была представлена в правильном формате.

Это поверхностное описание процесса маршрутизации выглядит не очень сложным. В действительности процесс не отличается от описанного. Но если бы все было так просто, не возникло бы необходимости в этом курсе.

Сложность процессов, связанных с маршрутизацией, начинает проявляться, когда мы решаем добавить к базовому алгоритму метрики и другие правила. Дополнительные правила — это то, за что маршрутизацию считают трудной для понимания. Рассмотрим базовые правила, оставив пока в стороне все сложности; вернемся к ним позже.

Далее поясняет назначение протоколов с точки зрения маршрутизации. Одна из функций, выполняемых протоколами, заключается в инкапсуляции передаваемых данных. Это значительно облегчает процедуру маршрутизации.

**Маршрутизация протокола**

Прежде чем с головой окунуться в глубины маршрутизации (которая будет описана со стороны маршрутизатора), необходимо определить роль протоколов в этом процессе. Или точнее, что имеют в виду, когда говорят об инкапсуляции данных протоколом, и как инкапсуляция может помочь процессу маршрутизации? Чтобы ответить на эти вопросы, мы должны выяснить, что происходит с данными до того, как они попадают в маршрутизатор.

Когда такое устройство, как ПК, хочет посылать данные другому устройству, оно передает их драйверу протокола, находящемуся на ПК. Протокол выполняет инкапсуляцию данных и подготавливает их к отправке.

Инкапсуляция — один из ключевых факторов, делающих маршрутизацию возможной. Маршрутизаторы не могут перемещать данные произвольного формата. Информация должна быть структурирована таким образом, чтобы маршрутизатор мог легко определить, какого типа данные он получил и куда их следует отправить. Смысл инкапсуляции как раз и заключается в том, чтобы предоставить эти данные.

Инкапсулируя данные, протокол добавляет к ним адрес пункта назначения, чтобы маршрутизатор мог их туда послать.

До того как данные инкапсулированы (и к ним добавлен адрес получателя), маршрутизатор не может узнать, куда их следует отправить.

Хотя протокол и предусматривает указание адреса доставки, маршрутизатору необходимо знать, что представляют собой данные. Когда человек получает письмо, он знает, что конверт, в котором оно пришло, не является частью посланной ему информации. Он служит лишь контейнером для самого письма. Инкапсуляция играет роль конверта, в котором маршрутизаторы пересылают данные.

Протокол форматирует данные таким образом, чтобы они помещались в определенный размер (как и в случае с конвертом). Благодаря этому маршрутизатор знает, что информация, отправляемая по указанному адресу, помещается в *x* бит. Все эти сведения хранятся в блоках данных, называемых заголовками протокола.

**Заголовки протокола**

Одна из обязанностей протокола, когда он получает поток данных для передачи, состоит в инкапсуляции. Инкапсулируя поток данных, протокол преобразует их в формат, который может быть легко понят любой системой, получившей эти данные. Но протокол не предусматривает того, что каждое устройство, встретившееся на пути данных, должно читать весь поток. Поэтому данные отформатированы так, что промежуточным устройствам достаточно прочитать небольшой фрагмент, чтобы понять, кому предназначен конкретный пакет.

Именно с этой целью протокол добавляет в поток данных заголовки, содержащие информацию, на основе которой любое устройство, поддерживающее этот протокол, может сделать заключение об инкапсулированных данных.

В процессе инкапсуляции данных, полученных от устройства, протокол не изменяет их содержания. Заголовки вставляются в начале блоков данных, не изменяя их.

Не путайте инкапсуляцию с шифрованием. В результате шифрования данные преобразуются к такому виду, что ни одно устройство (за исключением адресата) не может прочитать содержимое пакета.

Проиллюстрируем назначение заголовков на примере полей IP-протокола. Одна из функций IP, когда он получает поток данных для передачи по сети, состоит в добавлении к данным заголовков. Из этих заголовков любое устройство, пропускающее поток (например, маршрутизатор), может получить некоторые важные сведения об инкапсулированных данных. На рис. 6.1 показаны поля IP-заголовка.

Рис. 6.1. Поля IP-заголовка

Первое поле, «Версия», сообщает маршрутизатору, какая версия IP была использована для формирования пакета. На данный момент основной является версия IP-4, хотя IP-6 быстро становится реальностью (а некоторые компании уже приступили к тестированию IP-8). Следовательно, маршрутизаторы нуждаются в стандартном способе определения версии протокола, с которым они имеют дело.

Следующее поле называется «Длина заголовка». Согласно названию, оно содержит значение длины самого IP-заголовка. Получив длину заголовка, маршрутизатор может определить, где этот заголовок заканчивается и начинаются собственно данные.

Поле «Тип сервиса» (TOS, type of service) определяет приоритет инкапсулированного пакета. Хотя это и относится к дополнительным возможностям, многие маршрутизаторы Cisco способны выполнять так называемую «приоритетную маршрутизацию» (TOS routing) с учетом значения данного поля. Таким образом (хотя сейчас это поле нас не интересует) оно может оказывать влияние на процесс перемещения данных от устройства к устройству.

Поле «Общая длина» содержит значение длины всего инкапсулированного пакета. Пользуясь этим значением, маршрутизатор может определить длину блока данных, следующего за заголовком (то есть длина блока данных равна общей длине за вычетом длины заголовка).

Поле «Идентификатор» содержит уникальный номер, присваиваемый каждому пакету. Эти номера помогают принимающему устройству собрать вместе полученные пакеты.

Следующее поле, «Флаг», используется при определении того, может ли пакет быть фрагментирован. Если может, то флаг определяет также, является ли данный пакет последним в последовательности фрагментов. Маршрутизаторы могут фрагментировать пакеты перед дальнейшей отправкой.

Маршрутизаторы Cisco могут быть настроены так, чтобы не маршрутизировать пакеты, имеющие размер, превышающий заданный. Если, например, какой-либо из каналов имеет низкую пропускную способность, маршрутизатор может отправлять в него только те пакеты, размер которых находится внутри определенного диапазона. Кроме того, маршрутизатор может быть сконфигурирован так, что будет приводить слишком длинные пакеты к требуемому размеру, деля их на части. В этом процессе используется поле флага. Однако при этом возникает необходимость в еще одном поле — поле смещения фрагмента.

Поле «Смещение фрагмента» тесно связано с полем «Флаг». Оно определяет, на каком байте заканчивается фрагментированный пакет. Это позволяет устройству-получателю правильно собрать фрагментиро-ванные пакеты (для последующей сборки потока данных).

Обычно маршрутизаторы передают пакет в течение ограниченного времени. То есть если маршрутизатор не может найти получателя пакета за указанный промежуток времени, он прекращает поиски. Поле «Время жизни» (TTL, Time to Live) используется для определения срока существования пакета. В нем указывается максимальная продолжительность попыток доставки пакета, по достижении которой пакет считается недоставленным. Когда маршрутизатор пытается отправить пакет, а адресат недоступен, он отмечает время, хранящееся в поле TTL, и начинает обратный отсчет. Если пакет не будет доставлен раньше, чем истечет его время жизни, он будет отброшен.

Малое значение TTL может вызвать проблемы в больших сетях. Например, если пакет должен пройти через несколько маршрутизаторов (объединяющих несколько сетей), а значение TTL недостаточно велико, оно может исчерпаться прежде, чем пакет достигнет точки назначения. Поэтому даже нормально функционирующие системы могут не получить данные из-за низкого значения TTL.

Поле «Протокол» используется приемным устройством, а не маршрутизатором. Это поле сообщает получателю, какому протоколу передать данные по окончании их доставки и сборки.

Поле «Контрольная сумма заголовка» вычисляется по содержанию заголовка и служит для проверки его целостности. Проверяя контрольную сумму, маршрутизатор может гарантировать, что испорченные пакеты не будут отправлены далее.

Следующее поле «Адрес отправителя» — одно из наиболее важных для процесса маршрутизации. Оно сообщает маршрутизатору, кто отправил пакет. Этот адрес может быть использован для фильтрации и определения непрерывности маршрута. Например, маршрутизатор может быть настроен так, чтобы отправлять пакеты, пришедшие от одного источника в определенную сеть. Соответствие пакета этому критерию будет определяться по значению данного поля. Поле «Адрес получателя» расположено непосредственно после адреса отправителя и, безусловно, является самым важным для маршрутизации. Это поле определяет адрес принимающего устройства. Очевидно, что без адреса получателя маршрутизация была бы невозможна.

Поле «Параметры» содержит специфичные для IP параметры, и, наконец, в поле «Данные» помещается та информация, которая должна быть передана из одной системы в другую.

Большая часть информации, используемой для маршрутизации, хранится в заголовках пакетов. В заголовках передаются все данные, необходимые для доставки пакетов в пункт назначения. Маршрутизаторы получают из этих заголовков сведения, необходимые им для успешной пересылки данных.

Другое, не менее важное назначение инкапсуляции состоит в том, чтобы разделить поток данных на небольшие, легко управляемые порции. Такие пакеты облегчают чтение и обработку данных маршрутизаторами.

**Пакетирование данных**

Одной из функций протоколов является разделение потока данных в процессе инкапсуляции на пакеты такого размера, который облегчает манипулирование ими. Благодаря такому разбиению упрощается работа маршрутизаторов по пересылке больших объемов данных. Даже при сегодняшней кажущейся бесконечной пропускной способности отсутствие сегментирования больших объемов данных привело бы к многочисленным заторам.

Однако проблема трафика — не единственная проблема, решаемая с помощью инкапсуляции. Отправка данных небольшими (и, как правило, унифицированными) пакетами, увеличивает вероятность того, что приемное устройство останется доступным до окончания передачи и данные будут благополучно доставлены.

В процессе инкапсуляции данные, предназначенные для пересылки, «нарезаются» протоколом на фрагменты переменной длины. Затем каждый фрагмент, или пакет, маркируется и получает заголовок. Благодаря тому что каждый пакет снабжается заголовком, не обязательно отправлять все пакеты одновременно.

Каждый протокол имеет свои, определенные спецификацией, размеры пакетов. При возникновении проблем в сети, просматривая пакеты, можно по их размерам определить тип протокола.

Когда пакеты достигают пункта назначения, система может прочитать все заголовки и определить последовательность пакетов (это поможет ей собрать их в правильном порядке). Затем из пакетов удаляются заголовки, и восстанавливается исходный поток данных.

Описанное выше должено было дать вам информацию, достаточную для понимания того, как протоколы способствуют процессу маршрутизации. Однако функции, предоставляемые маршрутизируемыми протоколами, составляют лишь одну треть от полной картины. Аппаратура маршрутизатора должна выполнить еще много работы, прежде чем данные смогут попасть к адресату.

**Механизм маршрутизации**

Вопреки распространенному мнению, маршрутизация сама по себе достаточно проста для понимания. Любая маршрутизация основывается на логических правилах и стратегиях. Сложность возникает тогда, когда используются уровни безопасности, вторичные и третичные протоколы, сложная топология сети.

В оставшейся части этого раздела рассказывается о том, как происходит маршрутизация и как маршрутизаторы Cisco перемещают данные от системы к системе.

Маршрутизация в простой сети заключается в перемещении данных из одной сети в другую, в то время как в сложных сетях может потребоваться передача данных между несколькими маршрутизаторами, охватывающими несколько сетей. Существует значительная вероятность столкнуться с такими трудностями, как, например, множество метрик в сложной маршрутизируемой среде.

Поэтому давайте рассмотрим отдельно физический процесс маршрутизации, поскольку он применим ко всем случаям — и простым, и сложным.

**Маршрутизация в простой сети**

Простейший случай маршрутизации имеет место, когда данные должны перемещаться только между двумя сетями (рис. 6.2).

Обратите внимание на то, что рассматриваемая среда состоит из двух сетей (обозначенных A и B), каждая со своими собственными IP-адресами. В силу того что сети разделены физически и имеют разные схемы адресации, они не могут свободно обмениваться данными. Решение заключается в установке простого маршрутизирующего оборудования.

Приведенный сценарий и решение для него являются очень упрощенными. Однако все маршрутизируемые сети (как простые, так и сложные) построены на тех же принципах.

Рис. 6.2. Простая сетевая среда

Давайте рассмотрим работу маршрутизатора, помещенного между этими двумя сетями.

Персональным компьютерам и другим адресуемым устройствам (серверам и прочим) необходимо указать адрес нового маршрутизатора. Этот параметр обычно называется «шлюзом по умолчанию» (default gateway), но в некоторых системах название может быть другим. Однако независимо от названия этот параметр в любой системе должен содержать адрес маршрутизатора. На рис. 6.3 показаны те же сети с установленным между ними маршрутизатором.

Рис. 6.3. Две сети, объединенные маршрутизатором

Обратите внимание: маршрутизатор соединен с концентратором в каждой из сетей. Маршрутизация имеет место не между отдельными системами, а между сетями. Следовательно, возникает необходимость в еще одном устройстве, соединяющем сеть с маршрутизатором.

Хотя в этом примере и использованы концентраторы (hubs), предпочтение следует отдавать коммутаторам (switchs) , особенно в сильно загруженных сетях. Коммутаторы имеют возможность пересылки данных между заданными устройствами в отличие от концентраторов, выполняющих широковещательную рассылку пакетов всем устройствам сети.

Правильная установка маршрутизатора между двумя сетями может оказаться самой сложной частью работы. Дело в том, что каждому из интерфейсов маршрутизатора, соединяемого с сетью, необходимо назначить правильный адрес. Это значит, что интерфейс, соединяемый с сетью 10.198.50.0, должен получить адрес из диапазона 10.198.50.x, а интерфейс, соединяемый с сетью 10.198.60.0, — из диапазона 10.198.60.x.

В большинстве сетей первые пять или десять адресов зарезервированы для маршрутизаторов. Хотя этого не требует ни один стандарт, большинство администраторов придерживаются такого соглашения.

Теперь, когда маршрутизатор установлен и на всех ПК указан его адрес, посмотрим, как данные будут перемещаться из сети A через маршрутизатор в сеть B.

Пусть компьютеры сети A (10.198.50.0) посылают данные друг другу через концентратор. Допустим, ПК с адресом 10.198.50.5 отправляет пакет по адресу 10.198.50.8. У него есть только один путь — к концентратору. Концентратор, будучи достаточно несложным устройством, просто ретранслирует полученный им пакет всем присоединенным к нему устройствам. В заголовке пакета указаны адреса отправителя и получателя. Каждое устройство, соединенное с концентратором, читает заголовок и проверяет, не ему ли послан данный пакет. Одним из этих устройств и будет 10.198.50.8. Когда оно получит сообщение, то обработает его должным образом и отправит ответ.

Остальные устройства прочитали заголовок и определили, что пакет предназначен не им, поэтому они просто проигнорировали его и продолжили ожидание сообщений.

При наличии маршрутизатора устройства сети A могут адресовать пакеты компьютерам сети B. Когда такой пакет, покинув отправившее его устройство, достигнет концентратора, то будет проигнорирован (по причине несовпадения адреса) всеми устройствами сети A. Но пакет попадет и в маршрутизатор, который (в отличие от остальных компьютеров сети А) не отбросит его сразу, а проанализирует, чтобы проверить, не известно ли ему местонахождение адресата.

Первым делом маршрутизатор откроет и прочитает заголовок пакета, откуда узнает адрес его получателя. С целью определения сети назначения (10.198.60.0) он сравнит полученный адрес (10.198.60.17) с маской подсети интерфейса, соединенного с сетью A.

Затем маршрутизатор просмотрит свои конфигурационные записи, чтобы проверить, не соответствует ли одна из них указанному сетевому адресу. Маршрутизатор обнаружит, что интерфейс 10.198.60.1 принадлежит сети 10.198.60.0. Теперь интерфейс 10.198.50.1 может передать пакет интерфейсу 10.198.60.1, который, в свою очередь, отправит его концентратору сети B.

Это весьма упрощенное описание того, как маршрутизатор перемещает данные между сетями и как в этом участвует протокол, в действительности довольно точно. Однако вряд ли вы встретите где-нибудь подобную сеть. Большинство имеющихся сегодня сетей используют сложные схемы маршрутизации.

**Маршрутизация в сложной сети**

Совокупность большого числа сетей и маршрутизаторов образует сложную систему, в которой между каждой парой сетей имеется более одного маршрута. Пример сложной маршрутизируемой сети приведен на рис. 6.4.

Рис. 6.4. Сеть со сложной маршрутизацией

Предположим, что устройство, находящееся в сети A, отправляет данные устройству сети D. Маршрутизирующее оборудование выполняет следующую последовательность действий:

1. Инкапсулированный протоколом пакет достигает маршрутизатора A.
2. Маршрутизатор A просматривает таблицу маршрутов в поисках записи, определяющей местоположение сети D, и находит в ней два пути, ведущих к этой сети. Первый путь проходит через маршрутизаторы F и G, а второй — через маршрутизаторы H и G.
3. Маршрутизатор A, применяя алгоритм, реализованный в протоколе маршрутизации — в данном случае это протокол OSPF (Open Shortest Path First — протокол первоочередного открытия кратчайших маршрутов), — сравнивает метрики обоих путей и определяет, что путь, проходящий через маршрутизаторы F и G, является наилучшим (в смысле наикратчайшим).
4. Маршрутизатор A инкапсулирует пакет (уже инкапсулированный протоколом IP) в соответствии с протоколом OSPF. Затем он записывает в поле адреса назначения OSPF-заголовка адрес сети D.
5. Маршрутизатор A посылает OSPF-инкапсулированный пакет маршрутизатору F.
6. Маршрутизатор F читает заголовки OSPF и видит, что пакет адресован маршрутизатору G.
7. Маршрутизатор F просматривает свою таблицу маршрутов, выясняет, что у него имеется физическое соединение с маршрутизатором G, и пересылает пакет ему.
8. Маршрутизатор G получает пакет, удаляет OSPF-заголовок и пересылает IP-пакет в сеть D.

Этот сценарий мог бы быть и более сложным. Во многих сетях применяются списки доступа и множественные метрики (оба понятия будут рассмотрены в этой книге позже).

Как бы то ни было, этот пример является хорошей иллюстрацией того, как происходит маршрутизация. Понимание этого процесса поможет вам правильно выбрать параметры для настройки маршрутизатора Cisco в различных ситуациях.

В обоих сценариях, как в простом, так и в сложном, упоминалась таблица маршрутов. Эта таблица — своего рода база данных, размещенная в памяти маршрутизатора и содержащая всю информацию, необходимую для поиска местонахождения адресуемой сети.

**Таблицы маршрутов**

Таблицу маршрутов можно рассматривать как маленькую базу данных. Но описать ее непросто, так как формат записей в ней изменяется в зависимости от используемого протокола.

Поэтому здесь мы обсудим только общие элементы, содержащиеся в таблицах маршрутов. Частные случаи, связанные с использованием конкретных протоколов (например, RIP).

Основная информация, хранящаяся в таблице маршрутов, заключается в сведениях о соответствии сетей и маршрутизаторов, записанных в форме «один к одному». Например, фрагмент таблицы маршрутов маршрутизатора A (из сценария, изображенного на рис. 6.4) может выглядеть так:

Network A — ME

Network B — Router B

Network C — Router C

Network D — Router F

Network D — Router H

Network E — Router F

Network E — Router H

Network F — Router E

При внимательном рассмотрении вы можете обнаружить одно несоответствие между таблицей и схемой, приведенной на рис. 6.4. Маршрутизатор G служит шлюзом сети D, но с точки зрения маршрутизатора A сеть D обслуживают маршрутизаторы F и H. Причина кроется в том, что большинство протоколов маршрутизации не позволяют маршрутизаторам ссылаться на устройства, непосредственно с ними не связанные. Другими словами, отсутствие физического соединения между маршрутизаторами G и A заставляет последний считать, что путь к сети D лежит только через маршрутизаторы F и H.

Маршрутизатор A знает о существовании сети D, но его не интересует, как в нее попасть. Все, что ему надо знать — это то, кому надо отправить пакеты, адресованные в сеть D. В данном случае A может передать пакеты любому из маршрутизаторов F или H, в таблицах маршрутов которых есть записи вида:

Network D — Router G

При такой организации таблиц отдельный маршрутизатор не несет ответственности за всю сеть. Каждый из них отвечает только за свой небольшой участок (в пределах своих физических соединений). Использование полной таблицы маршрутов для всей сети на каждом маршрутизаторе повлекло бы за собой ряд проблем.

Во-первых, это размер самой таблицы. По мере разрастания таблицы маршрутов поиск в ней будет требовать все больше и больше времени, что замедлит обработку данных. Кроме того, хранение такой таблицы потребует дополнительной памяти, необходимой программам маршрутизации и конфигурационным файлам.

Вторая проблема, связанная с хранением полной таблицы, носит название *конвергенции (convergence)*. Конвергенцией называют такое состояние сети, в котором таблицы маршрутов всех устройств не противоречат друг другу. Целью протокола маршрутизации является наискорейшее достижение конвергенции при изменении данных в таблице. Если каждый маршрутизатор будет хранить полную таблицу, то изменения, сделанные на одном конце сети, могут не скоро попасть в таблицы маршрутов на другом ее конце. Замедление конвергенции приведет к возникновению петель и разрывов в маршрутах, в результате чего данные могут быть утеряны.

Таблицы маршрутов выполняют еще одну важную функцию: хранение и обновление *метрик маршрутов (routing metrics)*. На основании метрик делается выбор в пользу того или иного маршрута. То есть метрики, сопоставленные каждому из маршрутов в таблице, используются для расчетов, на основании которых принимается решение, по какому из маршрутов, ведущих к одному и тому же адресу, следует отправить пакет.

В сценарии, показанном на рис. 6.4, маршрутизатор A имеет выбор из двух маршрутов, по которым он может отправить пакеты в сеть D. Можно послать данные маршрутизатору F, а можно и маршрутизатору H — оба пути, в конце концов, приведут к сети D. Для принятия решения маршрутизатор A использует метрики, присвоенные каждому из путей.

Значения метрик зависят от используемого протокола. Но для всех протоколов одна характеристика метрик остается неизменной: 99% из них — это произвольные значения, присвоенные администратором сети или специалистом по маршрутизаторам.

Метрики типа «стоимости», назначаемые администраторами, определяют относительную стоимость использования данного маршрута по отношению к другим. Существует множество факторов, способных повлиять на решение администратора присвоить меньшее значение одному пути и большее — другому. Выбор может зависеть от объема трафика, надежности оборудования, стоимости (в денежном выражении) канала. Однако конкретные значения определяются только администратором.

В нашем примере путь через маршрутизатор H может иметь метрику 200, а через маршрутизатор F — метрику 100. Когда маршрутизатор A проверит все метрики (а в некоторых протоколах их может быть десяток), он рассчитает окончательную стоимость маршрута. Путь с наименьшим значением будет выбран в качестве наилучшего.

В этом разделе уже упоминалась конвергенция. Однако эта тема заслуживает более подробного рассмотрения, поэтому давайте поближе познакомимся с этим важным понятием.

**Достижение конвергенции**

Конвергенция маршрутов в сети — такая же неотъемлемая часть процесса маршрутизации, как и все рассмотренные выше факторы. Проще говоря, если каждый маршрутизатор получит собственную, отличную от остальных таблицу маршрутов, то и пересылку данных он будет выполнять своим уникальным способом. Это вызовет множество проблем, и сеть, скорее всего, перестанет работать.

Следовательно, система должна быть организована так, чтобы все маршрутизаторы некоторой сети работали с одной и той же таблицей. В результате достигается конвергенция маршрутизаторов.

Ответственность за достижение конвергенции маршрутизаторов возложена на протокол маршрутизации. Поэтому способов достижения конвергенции столько же, сколько протоколов. Однако базовая концепция этого процесса остается неизменной.

Чтобы добиться конвергенции, каждый маршрутизатор сети посылает свою часть таблицы маршрутов всем окружающим маршрутизаторам. Получив порцию изменений, каждый маршрутизатор обновляет свою таблицу и посылает изменения дальше. Этот процесс продолжается до тех пор, пока все маршрутизаторы не получат одни и те же данные о сети.

По мере рассмотрения различных протоколов мы будем обсуждать соответствующие особенности достижения конвергенции.

**Маршрутизируемые протоколы и протоколы маршрутизации**

При изложении материала этого раздела мы использовали термины «маршрутизируемый протокол» и «протокол маршрутизации». Когда в сети находится несколько маршрутизаторов, оба протокола необходимы и оба одинаково важны.

Скорее всего, вы уже хорошо знакомы с маршрутизируемыми протоколами. К ним относятся такие распространенные в локальных и глобальных сетях протоколы, как TCP/IP и Frame Relay. Пакеты этих протоколов перемещаются между маршрутизаторами в пакетах, сформированных протоколами маршрутизации. Помимо этого пакеты маршрутизирующих протоколов используются для передачи изменений в маршрутах и значений метрик. Без использования протоколов маршрутизации информация, передаваемая маршрутизируемыми протоколами, не смогла бы перемещаться от маршрутизатора к маршрутизатору.

При наличии в сети единственного маршрутизатора не возникает необходимости в применении протокола маршрутизации. То есть эти протоколы переносят данные только между маршрутизаторами. Следовательно, если в вашей сети только один маршрутизатор, то протокол маршрутизации вам не потребуется.

Теперь, познакомившись с концепциями маршрутизации, как с точки зрения протоколов, так и с точки зрения оборудования, мы сможем в следующем разделе «испачкать руки». Это значит, что в ней мы будем заниматься конфигурированием маршрутизатора Cisco «с нуля».

**Проверка знаний: Перемещение данных маршрутизаторами**

**Задание 1**

Продолжите верно фразу.

*Конвергенцией* называют состояние сети, в котором таблицы маршрутов всех устройств...

не способны преобразоваться к единому виду.

не противоречат друг другу.

противоречат друг другу.

**Задание 2**

При каких условиях протокол маршрутизации вам не потребуется?

Если в вашей сети от 1 до 10 маршрутизаторов.

Если в вашей сети только один маршрутизатор.

Если в вашей сети от 1 до 4 маршрутизаторов.

**Задание 3**

За что отвечает протокол?

доставка информации из одного места в другое

предоставление информации в правильном формате

определение местоположения

**Задание 4**

Как добиться конвергенции?

Маршрутизатор сети посылает часть своей таблицы маршрутов всем окружающим маршрутизаторам — получая изменения, маршрутизаторы обновляют свою таблицу и посылают изменения дальше — процесс длится, пока все маршрутизаторы не получает одинаковые данные о сети.

Маршрутизатор сети посылает целиком свою таблицу маршрутов всем окружающим маршрутизаторам — маршрутизаторы получают таблицы от других маршрутизаторов и обновляют свою — все маршрутизаторы получают одни и те же данные о сети.

**Задание 5**

Зачем нужна *инкапсуляция*?

для проверки и корректировки данных

для определения типа данных и куда их следует отправить

для определение типа протокола

**Задание 6**

Просматривая пакеты, по какой характеристике можно определить их тип протокола?

по размерам

по адресу отправления

по адресу получения

**Задание 7**

Чем отличается маршрутизация в сложной сети от простой?

ничем не отличается

наличием коммутатора

наличием между каждой парой сетей более одного маршрута

**Задание 8**

Какое поле является самым важным для маршрутизации?

«Протокол»

«Адрес получателя»

«Идентификатор»

**Введение в сегментированные сети**

Данный раздел введет вас в мир проектирования сетей. Работа с маршрутизаторами подразумевает не только их конфигурирование и грамотное сопровождение, но и выбор наиболее эффективного расположения маршрутизаторов в сети. Можно считать, что правильное размещение маршрутизатора в значительной степени определяет эффективность работы всего оборудования.

К этому моменту мы уже рассмотрели все основные принципы, лежащие в основе успешного выбора, конфигурирования и сопровождения простого маршрутизатора IP. Однако мир не ограничивается приведенными в предыдущих разделах примерами. Реальные маршрутизируемые сети намного сложнее и запутаннее, чем те, с которыми мы успели познакомиться.

В оставшейся части этого курса мы рассмотрим технические решения, с которыми чаще всего приходится иметь дело специалистам по Cisco. Сегментация сетей — одно из таких решений. Сегментированные сети, в силу своей запутанности и сложности, могут стать камнем преткновения для многих профессионалов в области сетей и маршрутизации.

Имейте в виду, что хотя этот раздел и посвящена маршрутизации IP между сегментированными сетями, в ней не затрагиваются вопросы, связанные с протоколами маршрутизации, так как их мы еще не изучали. Сейчас мы рассмотрим маршрутизацию только с точки зрения протокола IP. Понимание маршрутизации в целом требует знания как маршрутизируемых протоколов, так и протоколов маршрутизации, и мы, двигаясь последовательно, займемся сначала основами маршрутизации IP. В последующих разделах мы рассмотрим различные протоколы маршрутизации.

Глубокое понимание предмета сегментированных сетей и маршрутизации в них потребует обсуждения таких тем, как:

* Определение потребности в сегментировании
* Конфигурирование статических маршрутов между подсетями

Освоение сегментированных сетей потребует от вас углубленного понимания протокола IP и его маршрутизации. До сих пор мы рассматривали процессы, происходящие в «плоской» среде, то есть в статичной и имеющей лишь один путь, соединяющий две сети. Если вам уже приходилось работать с компьютерными сетями, вы знаете, что большинство компаний, использующих маршрутизаторы, не подходят под это описание.

Сегментированная IP-среда на сегодняшний день является самым распространенным вариантом конфигурации локальных и глобальных сетей. Сегментирование IP-сети предполагает ее разделение на подсети, в результате чего образуются десятки небольших IP-сетей. Каждая из этих подсетей связана с главной сетью и с другими подсетями. Зачастую такое многообразие маршрутов и схем адресации сбивает с толку даже бывалых профессионалов.

В предыдущих разделах мы вкратце рассмотрели процесс создания IP-подсети и получения ее маски в двоичном виде. В этом разделе данная тема рассмотрена более подробно, показана физическая сторона процесса сегментирования сети и представлены некоторые принципы, которые могут пригодиться при сегментировании вашей собственной сети.

* [Определение потребности в сегментировании](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904d5)
* [Деление IP-сети на подсети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904d6)
* [Размещение маршрутизаторов в сегментированных сетях](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904d7)
* [Конфигурирование статических маршрутов между подсетями](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904d8)
* [Проверка знаний: Введение в сегментированные сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904d9)

**Определение потребности в сегментировании**

До сих пор наши рассуждения о сегментировании сводились к тому, что биты, позаимствованные у адреса устройства, передаются адресу сети. Хотя этот способ и увеличивает количество адресов сетей, он одновременно уменьшает количество устройств в каждой из них. В действительности существует много веских причин для разделения IP-сети на подсети. Одна из самых серьезных причин, способная побудить администратора к сегментированию сети, — это постоянная нехватка IP-адресов. Применяемая ныне схема адресации IP-4 ограничена и скоро будет исчерпана.

Хотя IP-4 является наиболее широко используемой версией IP, на горизонте уже появилась IP-6. Версия IP-6 предлагает значительно больший диапазон адресов и способна обеспечить нужды компьютерных сетей в обозримом будущем.

С наступлением эры сетевых технологий статический пул IP-адресов быстро исчерпался. Сетевые администраторы, специалисты по маршрутизации и интернет-провайдеры вынуждены дорожить доставшимися им адресами.

**Деление IP-сети на подсети**

Когда сеть спроектирована и построена, на администратора ложится обязанность разработать действующую схему IP-адресации. Эта схема должна обеспечивать количество IP-адресов, адекватное потребностям вычислительной среды. Каждое устройство, которое потребует сетевого взаимодействия — персональные компьютеры, серверы, маршрутизаторы, — нуждается, по крайней мере, в одном адресе. Разработка схемы адресации требует способностей к планированию и предвидению.

IP-схема представляет собой множество IP-адресов (как для сетей, так и для устройств), которые присваиваются и используются внутри единой рабочей среды. IP-схема может состоять из диапазона адресов одного класса, а может включать в себя несколько диапазонов адресов разных классов.

Например, на рис. 7.1 изображена планируемая сетевая среда. Для этой среды определены географическое расположение объектов и предполагаемые требования к штату сотрудников. Имея такую информацию, администратор может вычислить приблизительное количество IP-адресов, необходимых для удовлетворения имеющихся требований.

Рис. 7.1. Планируемая сеть

Анализируя рисунок, администратор видит, что необходимо 695 IP-адресов. К этому количеству компания хотела бы добавить еще адреса, которые могут быть зарезервированы для последующего расширения: приема новых сотрудников или покупки оборудования. В данном примере администратор и компания договорились о том, что IP-схема сети должна содержать как минимум 750 адресов узлов и один адрес сети.

Такому требованию удовлетворяет IP-лицензия класса B. IP-лицензии класса B позволяют разметить 65 534 узла. Может показаться, что это оружие избыточной мощности для сети, которой требуется всего 750 адресов, но следующий, более мелкий класс C поддерживает только 254 узла. Поэтому организация должна приобрести лицензию класса B и начать готовиться к созданию своей новой сети.

За последние несколько лет очень возрос спрос на лицензии IP. В связи с этим сегодня вряд ли можно представить себе, что организации, состоящей из 750 человек, будет выдана лицензия на целый класс. По всей вероятности, компания из нашего примера получит подсеть интернет-провайдера, имеющего лицензию на класс B. Однако для наглядности процесса сегментирования будем считать, что компании выдана полная лицензия.

Пусть нашей фирме выдана лицензия класса B — 135.55.0.0, с маской подсети 255.255.0.0. Помните, что такая лицензия обеспечивает наличие одной сети (135.55) и 65 534 узлов (от 0.1 до 254.254). Администратор назначает новые адреса в сети. Получившаяся сетевая среда изображена на рис. 7.2.

Рис. 7.2. Сетевая среда с IP-схемой

На рис. 7.2 мы видим, что администратор рассматривал всю сеть как единый объект (каковым она технически и является) и присваивал адреса узлов, взяв за основу физическое расположение ПК. Компьютеры отдела HR получили адреса узлов с 1.1 по 1.51; департаменту IT присвоены адреса с 1.52 по 1.152, и т. д. Но это не самый эффективный способ распределения адресов.

Например, если у компании появится новый офис или она приобретет другую маленькую компанию, то им уже не останется сетевых адресов (используется всего один сетевой адрес — 135.55). Любым новым сетям, добавляемым в среду, должны присваиваться адреса, чтобы они могли участвовать в работе среды. Сетевому администратору необходимо каким-то способом создать новые адреса сетей, в то же время, сохранив достаточное количество адресов узлов для назначения устройствам по всей сетевой среде. Другими словами, необходимо сегментировать лицензию класса B.

Сегментирование лицензии класса B разделяет ее на несколько сетей. Разделив один сетевой адрес (135.55) на несколько сетевых адресов, администратор получает возможность использовать один из них для текущей сети и сохранить остальные на будущее. Тогда сетевая среда будет наиболее масштабируемой.

Однако сегментирование сетевого IP-адреса имеет и свои недостатки. Получая дополнительные сети, вы теряете узлы. То есть, создавая больше адресов сетей, вы уменьшаете количество адресов узлов, которые могут быть назначены в каждой из сетей. Для многих организаций потребность в сетях перевешивает потребность в узлах каждой сети. Давайте посмотрим, как сегментировать данную сеть так, чтобы использовать IP-адреса с максимальной пользой. Затем мы обсудим маршрутизацию в новой среде.

Так как сегментирование увеличивает количество сетей в IP-схеме, администратор должен спрогнозировать, сколько сетей может понадобиться компании в будущем, и при этом еще сохранить необходимое количество узлов. Администратор уже знает, что компании нужна как минимум одна сеть и 695 узлов.

Теперь следует решить, сколько бит IP-адреса должно быть передано из части, соответствующей адресу узла, сетевому адресу для обеспечения необходимого количества сетей. Формулы расчета количества сетей и узлов были приведены на странице [Организация подсетей](https://it.rfei.ru/course/%7EGZqN/-/%7EBT9K/%7E8dy7zmL3). Количество сетей и узлов, возникающее в результате передачи каждого бита, представлено в табл. 7.1.

Таблица 7.1. Количество сетей, появляющихся в результате сегментирования

| **Количество бит** | **Двоичный адрес** | **Количество сетей** | **Узлов в сети** |
| --- | --- | --- | --- |
| 2 | 11000000.00000000 | 2 | 16382 |
| 3 | 11100000.00000000 | 6 | 8190 |
| 4 | 11110000.00000000 | 14 | 4094 |
| 5 | 11111000.00000000 | 30 | 2046 |
| 6 | 11111100.00000000 | 62 | 1022 |
| 7 | 11111110.00000000 | 126 | 510 |

Если администратор примет решение передать 6 бит адреса узла сетевому адресу, то он получит 62 возможные сети с 1022 узлами в каждой. Этого будет достаточно для того, чтобы всей среде были присвоены адреса, при этом она будет работать должным образом.

После того как администратор определил, сколько бит необходимо использовать для получения правильного количества сетей и узлов, эти адреса должны быть назначены сети. Первым шагом в присвоении новых адресов является определение сетевых адресов, которые будут использоваться, и соответствующей им маски подсети.

Для вычисления нового сетевого адреса требуется немного математики, мы уже говорили об этом на странице [Организация подсетей](https://it.rfei.ru/course/%7EGZqN/-/%7EBT9K/%7E8dy7zmL3). Формула для вычисления сетевых адресов нашей сегментированной сети выглядит так:

**256 — маска подсети = интервал между сетевыми адресами**

Адрес первой сети в подсети — это просто интервал между сетевыми адресами. Все последующие адреса определяются последовательным добавлением интервала до тех пор, пока сумма не достигнет значения маски подсети, которое уже не годится для адреса. Звучит достаточно сложно, но после того как вы выполните приведенный выше пример, все станет понятно.

Первой переменной в уравнении для определения сетевых адресов является маска подсети. Соответственно, прежде чем вычислить сетевой адрес, вы должны определить, какой будет маска подсети для новой IP-схемы.

Маска подсети остается неизменной в рамках IP-схемы. То есть несмотря на то, что мы разделили IP-адрес между несколькими сетями, все эти сети будут совместно использовать общую маску подсети. Это будет та нить, которая свяжет сети вместе.

Чтобы вычислить маску подсети, просто запишите единицы (в двоичном формате) во все разряды сетевой части адреса. Так вы получите маску подсети, которая будет применяться во всей вашей сетевой среде. Различные маски подсети, доступные для адресов класса B, представлены в табл. 7.2.

Таблица 7.2. Маски подсети класса B

| **Количество бит** | **Двоичная маска** | **Маска подсети** |
| --- | --- | --- |
| 1 | 11111111.11111111.10000000.00000000 | 255.255.128.0 |
| 2 | 11111111.11111111.11000000.00000000 | 255.255.192.0 |
| 3 | 11111111.11111111.11100000.00000000 | 255.255.224.0 |
| 4 | 11111111.11111111.11110000.00000000 | 255.255.240.0 |
| 5 | 11111111.11111111.11111000.00000000 | 255.255.248.0 |
| 6 | 11111111.11111111.11111100.00000000 | 255.255.252.0 |
| 7 | 11111111.11111111.11111110.00000000 | 255.255.254.0 |

Согласно таблице, для нашей сети подойдет маска 255.255.252.0 — маска для адреса класса B, в котором 6 бит адреса узла были переданы адресу сети. Если бы было решено передать 5 бит, то использовалась бы маска 255.255.248.0.

После того как маска сети определена, вставим ее значение в уравнение и вычислим значения сетевых адресов. Напоминаем, что уравнение для вычисления сетевых адресов нашей сегментированной среды выглядит следующим образом:

**256 — маска подсети = интервал между сетевыми адресами = первый сетевой адрес**

Определив маску подсети (255.255.252.0), подставляем ее значение в уравнение:

256 – **252** = 4

Применив формулу, мы узнали, что первая сеть будет иметь адрес 135.55.**4**.0 с маской подсети 255.255.252.0. Используя этот сетевой адрес, можно приступить к перераспределению адресов в сети, рассматриваемой нами в качестве примера. Первые 1022 адреса сети 135.55.**4**.0 начинаются с 135.55.4.1 и заканчиваются 135.55.7.254. На рис. 7.3 изображена сеть, используемая нами в качестве образца, которой назначены адреса из нашей подсети.

Рис. 7.3. Пример сети, поделенной на подсети

Сравните сети на рис. 7.2 и 7.3. План, представленный на рис. 7.2, использовал только один сетевой адрес и терял до 64 тысяч адресов узлов, в то время как новый проект использует одну из 63 сетей и имеет приблизительно 300 резервных узлов. Такое решение гораздо более эффективно и «дружественно» маршрутизатору.

Теперь предположим, что после того как сеть построена, у компании появляется второй офис. Адресация в новой сети должна быть построена по той же IP-схеме, что и в первой. Кроме того, между двумя сетями должен быть помещен маршрутизатор Cisco, который свяжет их. Новая сеть представлена на рис. 7.4.

Рис. 7.4. Второй сегмент сети

Вспомните, как был вычислен первый сетевой адрес нашей подсети:

256 – 252 = 4 (135.55.**4**.0)

Чтобы получить следующий сетевой адрес, добавьте к первому сетевому адресу приращение. Если взять приращение (4) и сложить его с первым сетевым адресом, получится второй сетевой адрес (135.55.**8**.0).

Вторая сеть имеет точно такие же характеристики, как и первая. То есть в ней может быть 1022 узла, и она использует маску подсети 255.255.252.0.

Процедура добавления первого сетевого адреса для получения следующего сетевого адреса может повторяться до тех пор, пока сумма не станет равна адресу самой маски подсети, с которым сетевой адрес не может совпадать. Таким способом можно определить все доступные сети. Оставшиеся доступные сети нашей подсети класса B представлены в табл. 7.3.

Таблица 7.3. Оставшиеся сетевые адреса подсети

| **Маска подсети** | **Сеть подсети** | **Начало диапазона адресов узлов** | **Конец диапазона адресов узлов** |
| --- | --- | --- | --- |
| 255.255.252.0 | 135.55.4.0 | 135.55.4.1 | 135.55.7.254 |
|  | 135.55.8.0 | 135.55.8.1 | 135.55.11.254 |
|  | 135.55.12.0 | 135.55.12.1 | 135.55.15.254 |
|  | 135.55.16.0 | 135.55.16.1 | 135.55.19.254 |
|  | 135.55.20.0 | 135.55.20.1 | 135.55.23.254 |
|  | 135.55.24.0 | 135.55.24.1 | 135.55.27.254 |
|  | 135.55.28.0 | 135.55.28.1 | 135.55.31.254 |
|  | 135.55.32.0 | 135.55.32.1 | 135.55.35.254 |
|  | 135.55.36.0 | 135.55.36.1 | 135.55.39.254 |
|  | 135.55.40.0 | 135.55.40.1 | 135.55.43.254 |
|  | 135.55.44.0 | 135.55.44.1 | 135.55.47.254 |
|  | 135.55.48.0 | 135.55.48.1 | 135.55.51.254 |
|  | 135.55.52.0 | 135.55.52.1 | 135.55.55.254 |
|  | 135.55.56.0 | 135.55.56.1 | 135.55.59.254 |
|  | 135.55.60.0 | 135.55.60.1 | 135.55.63.254 |
|  | 135.55.64.0 | 135.55.64.1 | 135.55.67.254 |
|  | 135.55.68.0 | 135.55.68.1 | 135.55.71.254 |
|  | 135.55.72.0 | 135.55.72.1 | 135.55.75.254 |
|  | 135.55.76.0 | 135.55.76.1 | 135.55.79.254 |
|  | 135.55.80.0 | 135.55.80.1 | 135.55.83.254 |
|  | 135.55.84.0 | 135.55.84.1 | 135.55.87.254 |
|  | 135.55.88.0 | 135.55.88.1 | 135.55.91.254 |
|  | 135.55.92.0 | 135.55.92.1 | 135.55.95.254 |
|  | 135.55.96.0 | 135.55.96.1 | 135.55.99.254 |
|  | 135.55.100.0 | 135.55.100.1 | 135.55.103.254 |
|  | 135.55.104.0 | 135.55.104.1 | 135.55.107.254 |
|  | 135.55.108.0 | 135.55.108.1 | 135.55.111.254 |
|  | 135.55.112.0 | 135.55.112.1 | 135.55.115.254 |
|  | 135.55.116.0 | 135.55.116.1 | 135.55.119.254 |
|  | 135.55.120.0 | 135.55.120.1 | 135.55.123.254 |
|  | 135.55.124.0 | 135.55.124.1 | 135.55.127.254 |
|  | 135.55.128.0 | 135.55.128.1 | 135.55.131.254 |
|  | 135.55.132.0 | 135.55.132.1 | 135.55.135.254 |
|  | 135.55.136.0 | 135.55.136.1 | 135.55.139.254 |
|  | 135.55.140.0 | 135.55.140.1 | 135.55.143.254 |
|  | 135.55.144.0 | 135.55.144.1 | 135.55.147.254 |
|  | 135.55.148.0 | 135.55.148.1 | 135.55.151.254 |
|  | 135.55.152.0 | 135.55.152.1 | 135.55.155.254 |
|  | 135.55.156.0 | 135.55.156.1 | 135.55.159.254 |
|  | 135.55.160.0 | 135.55.160.1 | 135.55.163.254 |
|  | 135.55.164.0 | 135.55.164.1 | 135.55.167.254 |
|  | 135.55.168.0 | 135.55.168.1 | 135.55.171.254 |
|  | 135.55.172.0 | 135.55.172.1 | 135.55.175.254 |
|  | 135.55.176.0 | 135.55.176.1 | 135.55.179.254 |
|  | 135.55.180.0 | 135.55.180.1 | 135.55.183.254 |
|  | 135.55.184.0 | 135.55.184.1 | 135.55.187.254 |
|  | 135.55.188.0 | 135.55.188.1 | 135.55.191.254 |
|  | 135.55.192.0 | 135.55.192.1 | 135.55.195.254 |
|  | 135.55.196.0 | 135.55.196.1 | 135.55.199.254 |
|  | 135.55.200.0 | 135.55.200.1 | 135.55.203.254 |
|  | 135.55.204.0 | 135.55.204.1 | 135.55.207.254 |
|  | 135.55.208.0 | 135.55.208.1 | 135.55.211.254 |
|  | 135.55.212.0 | 135.55.212.1 | 135.55.215.254 |
|  | 135.55.216.0 | 135.55.216.1 | 135.55.219.254 |
|  | 135.55.220.0 | 135.55.220.1 | 135.55.223.254 |
|  | 135.55.224.0 | 135.55.224.1 | 135.55.227.254 |
|  | 135.55.228.0 | 135.55.228.1 | 135.55.231.254 |
|  | 135.55.232.0 | 135.55.232.1 | 135.55.235.254 |
|  | 135.55.236.0 | 135.55.236.1 | 135.55.239.254 |
|  | 135.55.240.0 | 135.55.240.1 | 135.55.243.254 |
|  | 135.55.244.0 | 135.55.244.1 | 135.55.247.254 |
|  | 135.55.248.0 | 135.55.248.1 | 135.55.251.254 |

Хотя новой сети нашей среды может быть присвоен любой сетевой адрес из этого списка, мы остановимся на 135.55.8.0. Теперь можно распределить новые адреса сети 8.0. Повторяя процедуру, выполненную для первой сети, будем назначать новые адреса в соответствии с функциями подразделений. То есть отдел продаж (Sales) получит адреса с 135.55.8.1 по 135.55.8.101 и т.д. Полностью снабженная адресами сеть изображена на рис. 7.5.

Рис. 7.5. Полностью адресованная сегментированная среда, состоящая из двух сетей

Две сети нашей среды не имеют возможности общаться друг с другом. Так как адреса им были присвоены как двум разным сетям, они не могут обмениваться данными. Чтобы обеспечить перемещение данных, необходимо поместить маршрутизатор так, чтобы он мог обслуживать обе части единой среды.

**Размещение маршрутизаторов в сегментированных сетях**

Многих сетевых проблем можно избежать за счет грамотного проектирования сети. Изучение маршрутизаторов Cisco побуждает к принятию на себя некоторой ответственности за всеобъемлющий план сети (включающий в себя и физическое оборудование для маршрутизации и программное обеспечение протоколов). Из данного раздела вы узнаете, как правильно размещать маршрутизаторы Cisco в сегментированных сетях.

Маршрутизатор, помещенный в сеть без учета окружающей его архитектуры, не будет работать эффективно. Необходимо уделить особое внимание таким элементам, как сетевой трафик и количество доступных интерфейсов. Помещая маршрутизатор Cisco между сегментированными сетями, необходимо учитывать несколько факторов:

* Сосчитайте доступные интерфейсы вашего маршрутизатора. Большая часть маршрутизаторов Cisco имеет два интерфейса LAN. Один маршрутизатор может соединить две сети, но для конфигурирования необходимых путей между тремя сетями может потребоваться до трех маршрутизаторов.
* Размещайте маршрутизатор так, чтобы наилучшим образом обслуживались самые загруженные части сети. Например, можно решить использовать отдельный маршрутизатор для наиболее загруженной части сети, даже если маршрутизация осуществляется только между двумя сетями.
* Если вы соединяете критически важные сетевые сегменты, разместите маршрутизаторы так, чтобы это привело к созданию избыточных соединений.
* Физически размещайте маршрутизаторы вместе с другим оборудованием, вблизи других устройств связи. Это упростит процедуру расширения, особенно если будет задействован Интернет.
* При размещении маршрутизаторов необходимо принимать во внимание то, какой именно протокол маршрутизации вы собираетесь использовать. Некоторые протоколы маршрутизации требуют, чтобы вы размещали маршрутизаторы в определенных местах или не выходили за рамки определенных ограничений. (Например, если вы планируете работать с RIP, то вы не сможете «дотянуться» до сетей, находящихся от вас на расстоянии, превышающем 16 переходов между маршрутизаторами.)

Хотя необходимо всегда принимать во внимание используемый в сети протокол маршрутизации, но в данном разделе мы откажем себе в этом удовольствии, так как ни один протокол маршрутизации еще не был изучен. Поэтому в этом разделе при принятии решения о размещении маршрутизаторов ограничимся другими факторами. (О протоколах маршрутизации поговорим в оставшихся разделах.)

Помня обо всех обстоятельствах, влияющих на выбор места, давайте выберем наилучшее расположение нашего маршрутизатора. Очевидно, что нужно поместить его между двумя сетями так, чтобы физически он был близок к ним обеим. Логичное размещение маршрутизатора изображено на рис. 7.6.

Рис. 7.6. Маршрутизатор Cisco, помещенный между двумя сетями

Работая с двумя сетями, можете при выборе места для вашего маршрутизатора положиться на логику. Помните, что временами работа с маршрутизаторами может быть достаточно сложной. Если возникает логичное решение, воспользуйтесь им. Так как в рассматриваемом сценарии присутствуют всего две сети, можно без сомнений расположить маршрутизатор Cisco между ними.

После того как место для маршрутизатора выбрано, остается сконфигурировать его. Давайте рассмотрим этапы настройки маршрутизатора Cisco, который должен соединить две сети.

1. Установить для каждого интерфейса адрес, соответствующий сети, к которой он подключен.
2. Включить интерфейсы.
3. Определить маску подсети.
4. Разрешить маршрутизацию IP.
5. Создать статические маршруты для соединения двух сетей.

Первый шаг заключается в конфигурировании каждого интерфейса для работы с IP. Это означает, что каждому интерфейсу маршрутизатора нужно назначить адрес, соответствующий сети, с которой он связан. В той же самой последовательности команд можно определить маску подсети для каждого интерфейса и включить интерфейсы. Таким образом, три первых этапа из списка реализует одна серия команд, которая и представлена ниже (помните, что по умолчанию все интерфейсы имеют статус «shutdown», поэтому их необходимо включить вручную):

Router>\*enable\*

Router#\*configure terminal\*

Router(configure)#\*interface ethernet 0\*

Router(configure-interface)#\*ip address 135.55.4.1 255.255.252.0\*

Router(configure-interface)#\*no shutdown\*

Router#\*configure terminal\*

Router(configure)#\*interface ethernet 1\*

Router(configure-terminal)#\*ip address 135.55.8.1 255.255.252.0\*

Router(configure-interface)#\*no shutdown\*

Router(configure-terminal)#\*^Z\*

Давайте посмотрим, к какому из вышеупомянутых этапов конфигурирования относится каждая команда.

Первые три строки последовательности команд переводят маршрутизатор в режим конфигурирования интерфейсов. Чтобы попасть в этот режим, нужно сначала войти в привилегированный режим, а затем в режим глобального конфигурирования:

Router>\*enable\*

Router#\*configure terminal\*

Router(configure)#\*interface ethernet 0\*

Цифра в конце команды interface ethernet 0 указывает, какой именно интерфейс мы хотим конфигурировать.

Перейдя в режим конфигурирования интерфейсов, можно задать IP-адрес и маску подсети (четвертая строка) для нужного интерфейса. В данном случае интерфейс Ethernet 0 подключен к сети 135.55.4.0; поэтому устанавливаем для него адрес 135.55.4.1 (первый шаг в списке) и подсеть 255.255.252.0 (шаг 3):

Router(configure-interface)#\*ip address 135.55.4.1 255.255.252.0\*

Адрес и маска подсети определены, и пришло время включить интерфейсы (шаг 2 в списке). Следующая команда включает интерфейс и сохраняет конфигурацию в файле running-config (+ во второй строке обеспечивает выход из режима конфигурирования интерфейсов):

Router(configure-interface)#\*no shutdown\*

Router(configure-interface)#\*^Z\*

Обратите внимание на структуру команды. Вместо того чтобы сказать маршрутизатору, чтобы он включил интерфейс, мы указываем, что маршрутизатор должен не выключать его. Затем нажатием + возвращаем маршрутизатор в привилегированный режим. Теперь можно повторить процесс для второго интерфейса:

Router#\*configure terminal\*

Router(configure)#\*interface ethernet 1\*

Router(configure-terminal)#\*ip address 135.55.8.1 255.255.252.0\*

Router(configure-interface)#\*no shutdown\*

Router(configure-terminal)#\*^Z\*

Необязательно выходить из режима конфигурирования интерфейсов после выполнения каждой «порции» настроек. В данном примере + было использовано для того, чтобы проиллюстрировать весь процесс целиком. Обычно оба интерфейса можно сконфигурировать в рамках одной сессии.

Шаг 4 нашего списка — это разрешение IP-маршрутизации. Фактически разрешается передача пакетов от одного интерфейса к другому. Прежде чем появится возможность использовать маршрутизатор между двумя подсетями, необходимо разрешить IP-маршрутизацию. Этот шаг выполняют две команды:

Router#\*configure terminal\*

Router(configure)# \*ip routing\*

Так как маршрутизация IP относится ко всему маршрутизатору, то команда выполняется в режиме глобального конфигурирования. По логике вы не можете разрешать IP-маршрутизацию для одного интерфейса и не разрешать для другого, ведь тогда данные все равно не будут никуда перемещаться, поэтому все параметры маршрутизации относятся к глобальному конфигурированию.

Последний этап — это конфигурирование статических маршрутов для направления потоков данных между подсетями. Так как наш проект не предусматривает использования протоколов маршрутизации, то нет возможности воспользоваться преимуществами динамической маршрутизации. Поэтому, несмотря на то, что маршрутизация происходит в пределах одного маршрутизатора (от одного интерфейса к другому), необходимо задать статические маршруты:

Router#\*configure terminal\*

Router (configure)#\*ip route 135.55.4.0 255.255.252.0 Ethernet 0 perm\*

Router (configure)#\*ip route 135.55.8.0 255.255.252.0 Ethernet 1 perm\*

Эти два выражения сообщают маршрутизатору, что сеть 135.55.4.0 подключена к интерфейсу Ethernet 0, а сеть 135.55.8.0 — к Ethernet 1. Но команда ip route обладает гораздо более мощными возможностями, чем использованные в данном примере. По мере изложения курса вы будете все более полно использовать ip route.

**Конфигурирование статических маршрутов между подсетями**

Маршрут — это карта или правило, используемые маршрутизатором для перемещения данных из одной сети в другую. Маршрутизаторы определяют движение информации в вашей сети. Маршрутизаторы Cisco могут использовать два разных типа маршрутов: статические и динамические. О динамических путях мы поговорим в следующих разделах, а в этом разделе сосредоточимся на изучении статических путей. Говоря буквально, статические маршруты — это предопределенные пути для передачи данных из одной сети в другую, которые были жестко запрограммированы в памяти маршрутизатора.

Читая этот раздел, помните, что статические маршруты хороши только в некоторых случаях. Существует три основные сетевые ситуации, в которых оправдано использование статических IP-маршрутов:

* Маршрутизаторы сети не используют протокол маршрутизации.
* Правила безопасности требуют, чтобы определенные маршрутизаторы пропускали только определенный трафик.
* Среда маршрутизации не изменяется.

О динамических маршрутах будет рассказано в последующих разделах, так как их поддерживают только протоколы маршрутизации.

Команда конфигурирования статических маршрутов между подсетями на самом деле достаточно проста, хотя у нее есть ряд необязательных параметров (табл. 11.4), которые обеспечивают выполнение различных задач. Формат командной строки для команды ip route таков:

#ip route <Destination Network> <Destination Subnet> <Next Hop | Interface | Null> <Next Hop | perm>

Таблица 7.4. Параметры команды ip route

| **Параметр** | **Описание** |
| --- | --- |
| <Destination Network> | IP-адрес сети адресата (то есть путь, по которому должны отправляться данные) |
| <Destination Subnet> | Маска подсети сети адресата |
| `<Next Hop | Interface |
| Next Hop | IP-адрес маршрутизатора, которому должны пересылаться пакеты для сети адресата |
| Interface | Внутренний интерфейс, которому должны пересылаться пакеты для сети адресата |
| Null | Указание на то, что статический путь будет использоваться в другой команде (этот параметр чаще используется протоколами маршрутизации, такими как BGP) |
| `<Next Hop | perm>` |
| Next Hop | IP-адрес маршрутизатора, которому должны отправляться пакеты, относящиеся к определенному интерфейсу (необязателен, если в предыдущей позиции задан интерфейс) |
| perm | Указание маршрутизатору записать маршрут в файл startup-config, тем самым сделать маршрут постоянным |

Давайте определим статические пути для более сложного случая сегментации сети, используя табл. 7.4 в качестве инструкции.

На рис. 7.7 изображена часть большой сегментированной сети.

Рис. 7.7. Сегмент сети

В этом сценарии у нас есть четыре взаимосвязанных маршрутизатора, каждый из которых обслуживает небольшой сегмент большой сегментированной сети. Чтобы обеспечить перемещение данных из сети 198.10.0.0 в сеть 198.13.0.0, необходимо задать следующие маршруты:

* Router 198.10.1.1:

RouterA(configure)#\*ip route 198.13.0.0 255.255.0.0 ethernet 1 198.11.1.1 perm\*

* Router 198.11.1.1:

RouterB(configure)#\*ip route 198.13.0.0 255.255.0.0 ethernet 1 198.12.1.1 perm\*

* Router 198.12.1.1:

RouterC(configure)#\*ip route 198.13.0.0. 255.255.0.0 ethernet 1 198.13.1.1 perm\*

* Router 198.13.1.1:

RouterD(configure)#\*ip route 198.13.0.0 255.255.0.0 ethernet 1 perm\*

Если эти четыре пути сконфигурированы, то информация будет пересылаться с одного маршрутизатора на другой, пока не достигнет адресата. Основу команды составляет информация о том, что данные для сети 198.13.0.0 должны покинуть маршрутизатор через определенный интерфейс и (в трех случаях) быть переданы другому маршрутизатору.

Материал, представленный в этом разделе, пригодится, когда вы дойдете до изучения таких протоколов маршрутизации, как RIP и OSPF. Для создания полнофункциональной среды маршрутизации мы будем комбинировать маршрутизируемые протоколы и протоколы маршрутизации. На одном маршрутизаторе можно использовать как динамические маршруты (посредством протоколов маршрутизации), так и статические IP-маршруты. Данный раздел будет особенно полезна при работе с крупными действующими сетями.

Пройдя материал этого раздела, вы должны были получить достаточно полное представление о маршрутизации и сегментировании IP. И маршрутизация и сегментирование будут использоваться далее в курсе при изучении более сложных вопросов.

**Проверка знаний: Введение в сегментированные сети**

**Задание 1**

Какая одна из самых серьезных причин, способна побудить администратора к *сегментированию сети*?

неравномерное распределение нагрузки на сеть

постоянная нехватка IP-адресов

отсутствие возможности создавать подсети

**Задание 2**

Что такое *статические маршруты*?

Это предопределенные пути для передачи данных из одной сети в другую, которые были жестко запрограммированы в памяти маршрутизатора.

Это разнообразные (свободные) пути для передачи данных из одной сети в другую, которые были жестко запрограммированы в памяти маршрутизатора.

**Задание 3**

Продолжите верно фразу.

Маршрутизатор не будет работать эффективно, если его поместить в сеть без учета...

окружающей его архитектуры

протокола

класса IP-лицензии

**Задание 4**

Сколько узлов позволяет разметить IP-лицензия класса B?

750

65 534

14 543

**Классовая адресация**

Ранее было дано определение объединенной сети TCP/IP как единой виртуальной сети, состоящей из множества физических сетей, связанных между собой с помощью маршрутизаторов. В этом разделе речь пойдет об адресации — важной составляющей любой системы, с помощью которой программам поддержки протокола TCP/IP удается скрыть особенности физической реализации конкретной сети и представить объединенную сеть как единую унифицированную систему.

* [Универсальная система идентификации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e2)
* [Оригинальная классовая система адресации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e3)
* [Адреса определяют сетевые соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e4)
* [Сетевые и направленные широковещательные адреса](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e5)
* [Ограниченная широковещательная передача](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e6)
* [Интерпретация нуля как значения "это"](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e7)
* [Механизмы адресации подсетей и суперсетей](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e8)
* [Многоадресатная передача](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904e9)
* [Недостатки IP-адресации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ea)
* [Точечная десятичная форма записи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904eb)
* [Адрес петли обратной связи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ec)
* [Соглашение по использованию специальных адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ed)
* [Управление адресацией в Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ee)
* [Зарезервированные префиксы IР-адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904ef)
* [Пример](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904f0)
* [Порядок следования байтов в сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904f1)
* [Проверка знаний: Классовая адресация](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6904f2)

**Универсальная система идентификации**

Говорят, что коммуникационная система предоставляет *универсальные услуги связи*, если она позволяет любому своему узлу взаимодействовать с любым другим узлом сети. Поэтому, чтобы сделать коммуникационную систему универсальной, необходимо определить приемлемый для всех метод идентификации каждого подключенного к ней компьютера.

Чаще всего узел сети характеризуют с точки зрения его *имени, адреса* и *маршрута*. В работе Шоч (Shoch) SHOCH, J. F., Internetwork Naming, Addressing, and Routing, Proceedings of COMPCON, 1978] предложил считать, что имя узла должно характеризовать сам объект, его адрес должен определять *положение1* объекта в сети, а маршрут — способ достижения объекта.

Хотя приведенные выше определения даны на интуитивном уровне, они могут ввести вас в заблуждение. На самом деле имена, адреса и маршруты являются, в порядке понижения уровня иерархии, основными характеристиками системы идентификации узла. Вообще говоря, для идентификации компьютеров люди предпочитают пользоваться удобопроизносимыми именами, тогда как программное обеспечение эффективнее работает с компактной формой идентификации, которую называют *адресом*.

Для универсальной идентификации машин в протоколе TCP/IP можно пользоваться либо именами, либо адресами. Однако было принято решение стандартизовать компактную двоичную форму записи адреса, которая позволяет повысить эффективность выполнения ряда компьютерных операций, таких как выбор маршрута следования пакетов. Поэтому далее будут рассматриваться только двоичные адреса, а вопросы их соответствия удобопроизносимым именам и методы использования адресов для маршрутизации пакетов мы оставим на потом.

1. Данные, указывающие на то, где можно найти объект, называют *указателем ресурса*.

**Оригинальная классовая система адресации**

Итак, объединенную сеть следует рассматривать как одну большую сеть, подобно любой другой физической сети. Конечно, между ними есть разница, которая заключается в том, что объединенная сеть является виртуальной структурой, придуманной инженерами и реализованной целиком и полностью в виде программного обеспечения. В результате разработчики были вольны в выборе форматов пакетов, их размеров и методов доставки, системы адресации и т.п., поскольку их не ограничивали рамки применяемого оборудования. Для определения адресов разработчики протокола TCP/IP выбрали систему, принятую для обозначения сетевых физических адресов. Каждому узлу объединенной сети назначается 32-разрядный двоичный адрес, называемый *межсетевым адресом (internet address)*, или *IP-адресом*. Особенность межсетевых адресов заключается в том, что они выбираются с учетом методов выполнения эффективной маршрутизации. В частности, в IP-адресе кодируется идентификационная информация о сети, к которой подключен узел, и уникальный номер узла в этой сети. Таким образом, можно сделать некоторые выводы.

Каждому узлу объединенной сети TCP/IP назначается уникальный 32-разрядный адрес, который используется во всех сеансах связи с этим узлом.

Для того чтобы вы уловили основную идею системы адресации протокола TCP/IP, мы для начала изложим материал в упрощенном виде, а затем постепенно будем его углублять. Итак, в простейшем случае каждому узлу объединенной сети назначается 32-разрядный универсальный идентификатор, который является его адресом. Часть IP-адреса (префикс) отводится под идентификатор сети. Таким образом, у всех машин некоторой сети префиксы IP-адресов будут одинаковыми.

Теоретически каждый адрес состоит из пары значений (netid, hostid), где часть netid идентифицирует сеть, а часть hostid — узел в этой сети. На практике же в рамках объединенной сети нет единого соглашения о том, какая часть IP-адреса выделяется для идентификации сети, а какая — узла, поскольку разработчики не указали в стандарте единых требований по этому поводу. В оригинальной системе адресации, которая теперь называется *классовой (classful)*, IP-адрес должен быть представлен в одном из первых трех форматов1, показанных на рис. 8.1.

Рис. 8.1. Пять форматов IP-адресов, используемых в оригинальной классовой системе адресации. Первые три класса — А, В и С — можно идентифицировать по первым трем битам IP-aдpeca

В классовой системе адресации IP-адрес является *автоматически опознаваемым*, поскольку его тип (или границу между префиксом и суффиксом) можно легко определить по значению адреса без обращения ко внешним источникам информации. В частности, класс адреса можно определить по значению трех старших битов. Обратите внимание, что для определения трех основных классов адреса достаточно проанализировать только два старших бита. Система адресации класса А используется в тех сетях, количество узлов в которых превышает 216 (или 65536). При этом 7 битов отводится под поле netid, а 24 бита — под поле hostid. Система адресации класса В используется для сетей средних размеров, количество узлов в которых превышает 28 (или 256), но меньше 216. При этом под поле netid отводится 14 битов, а под поле hostid — 16 битов. Система адресации класса С применяется для сетей, имеющих менее 28 узлов. При этом под поле netid отводится 21 бит, а под поле hostid — 8 битов. Обратите внимание, что изначально система IP-адресов была задумана так, чтобы можно было быстро извлечь из двоичного адреса поля netid и hostid. Подобная эффективность особенно важна для тех маршрутизаторов, которые используют поле адреса netid для принятия решения о том, куда следует направлять пакеты.

К способам поиска эффективных методов маршрутизации мы еще вернемся после рассмотрения изменений и дополнений, которые были внесены в оригинальную систему адресации.

1. Четвертый формат зарезервирован для многоадресатной передачи в объединенной сети и будет описан чуть позже. А пока что мы ограничимся описанием первых трех форматов IP-адреса, определяющего объект в сети.

**Адреса определяют сетевые соединения**

Для того чтобы облегчить понимание материала, мы говорили, что один IP-адрес соответствует одному узлу сети. Однако, строго говоря, такое утверждение не совсем верно. Давайте рассмотрим маршрутизатор, подключенный к двум физическим сетям.

Поскольку IP-адрес определяет как сеть, так и узел в ней, может возникнуть вопрос: можем ли мы назначить маршрутизатору один IP-адрес, и если да, то какой?. Ответ на этот вопрос отрицательный. В случае если обычный компьютер имеет несколько физических соединений с сетью, он называется *многоадресным узлом (multi-homed hosts)*. Естественно, что многоадресному узлу, коим является и маршрутизатор, нужно назначить несколько IP-адресов. Причем каждый из адресов должен соответствовать одному из сетевых подключений компьютера. Итак, рассмотрение многоадресных узлов навело нас на важную мысль.

Поскольку IP-адрес идентифицирует и сеть, и узел в этой сети, то можно сказать, что он соответствует не определенному компьютеру в сети, а его сетевому подключению.

Таким образом, если маршрутизатор подключен к *п* сетям, то ему должно быть назначено *п* IP-адресов, по одному для каждого сетевого подключения.

**Сетевые и направленные широковещательные адреса**

Мы уже говорили о преимуществах кодирования информации о сети в IP-адресе. Речь шла о том, что подобная структура адреса позволяет задействовать наиболее эффективные механизмы маршрутизации. Еще одно преимущество заключается в том, что IP-адрес может соответствовать как отдельному компьютеру сети, так и сетевому подключению этого компьютера. В соответствии с принятым соглашением, значение поля hostid равное нулю никогда не назначается отдельному узлу сети. Другими словами, IP-адрес с нулевым полем hostid используется для обозначения самой сети. Поэтому из всего сказанного выше можно сделать такой вывод.

IP-адреса могут идентифицировать как отдельные узлы, так и сами сети. По принятому соглашению, IP-адрес, у которого все биты поля *hostid* равны нулю зарезервирован и используется для обозначения самой сети.

Еще одним существенным преимуществом системы IP-адресов является то, что в ней предусмотрена возможность *направленной широковещательной адресации (directed broadcast address)* всех узлов сети. Согласно принятым стандартам, адрес, у которого значения всех битов поля hostid равны единице, зарезервирован и используется для направленной широковещательной передачи1. При отправке пакета по широковещательному адресу его копия передается по объединенной сети к сети получателя. Промежуточные маршрутизаторы на пути следования пакета обрабатывают только часть netid IP-адреса. При этом значение поля hostid никак не влияет на доставку пакета к сети получателя. Как только пакет достигнет маршрутизатора, подключенного к нужной сети, маршрутизатор выделяет из IP-адреса значение поля hostid и определяет метод доставки пакета. Если все биты этого поля равны единице, маршрутизатор выполняет широковещательную передачу пакета всем узлам указанной сети.

В большинстве сетевых технологий (таких как, например, Ethernet) широковещательная передача пакетов выполняется с той же эффективностью, что и передача обычных пакетов. В некоторых сетях широковещательная передача пакетов выполняется программным путем, в результате чего немного возрастают задержки в передаче, по сравнению с обычной передачей. Однако оборудование некоторых сетей вовсе не поддерживает режим широковещательной передачи. Поэтому использование направленного широковещательного IP-адреса отнюдь не гарантирует, что в сети получателя поддерживается широковещательный режим передачи или что пакеты будут доставлены с должной эффективностью всем узлам сети. Подводя итоги, можно сказать.

*IP-адрес может использоваться для указания режима направленной широковещательной передачи, в котором пакет посылается сразу всем компьютерам конкретной сети. В случае, если это возможно, широковещательный IP-адрес преобразовывается в соответствующий аппаратный широковещательный адрес. Согласно принятому соглашению, направленный широковещательный адрес должен содержать в поле* netid *корректный идентификатор сети, а в поле* hostid — *все единицы*.

1. К сожалению, в первых версиях программ, реализующих протокол TCP/IP для операционной системы Berkeley UNIX, по ошибке для широковещательной передачи использовались IP-адреса, у которых поле hostid равнялось нулю. Поскольку эта ошибка прижилась, во многих реализациях протокола TCP/IP предусматривалась возможность использования нулевого значения поля hostid для направленной широковещательной передачи.

**Ограниченная широковещательная передача**

В предыдущем разделе были рассмотрены так называемые *направленные (directed)* широковещательные адреса. Название “направленные” говорит о том, что в каждом адресе присутствует как корректный идентификатор сети, так и идентификатор узла, определяющий режим широковещательной передачи в этой сети. Направленные широковещательные адреса могут однозначно интерпретироваться в любой точке объединенной сети, поскольку из них можно выделить уникальный идентификатор сети получателя, а также определить, что заказан режим широковещательной передачи. Таким образом, направленные широковещательные адреса могут использоваться для реализации мощного (и поэтому иногда весьма опасного) механизма, позволяющего удаленному компьютеру послать единственный пакет, который затем будет автоматически разослан всем компьютерам указанной сети.

С точки зрения системы адресации основным недостатком направленной широковещательной передачи является то, что в адресе необходимо указывать идентификатор сети. Поэтому существует еще одна форма широковещательного адреса, который называют *ограниченным широковещательным адресом (limited broadcast address)*, или *широковещательным адресом локальной сети (local network broadcast address)*. Она предназначена для организации режима широковещательной передачи в локальной сети независимо от назначенного ей префикса IP-адреса.

Локальный широковещательный адрес состоит из 32 единиц (поэтому иногда его называют широковещательным адресом, состоящим из всех единиц). Локальный широковещательный адрес используется в процедурах инициализации узла сети для определения его IP-адреса, или префикса IP-адреса локальной сети. После определения корректного IP-адреса узла или префикса локальной сети, узел должен переходить на использование направленного режима широковещательной передачи.

Как правило, в семействе протоколов TCP/IP режим широковещательной передачи данных ограничивается минимально возможным набором машин.

**Интерпретация нуля как значения "это"**

Выше уже шла речь о том, что поле, состоящее из всех единиц, может быть интерпретировано как значение “для всех”, например “для всех узлов” сети. Вообще говоря, в межсетевом программном обеспечении поля, состоящие из одних нулей, интерпретируются как значение “это”. Такая интерпретация повсеместно встречается в литературе. Таким образом, IP-адрес, поле hostid которого состоит из одних нулей, интерпретируется как “этот” узел, а если поле netid равно нулю, то его значение интерпретируется как “эта” сеть. Естественно, подобная интерпретация адреса уместна только в том случае, если она может быть сделана однозначно. Например, если компьютер получает пакет, у которого в IP-адресе получателя поле netid состоит из одних нулей, а значение поля hostid соответствует адресу получателя, то он интерпретирует значение поля netid как “эта сеть” (т.е. сеть, по которой был получен пакет).

Нулевое значение поля netid используется в том случае, когда компьютер должен обменяться данными по локальной сети, но префикс IP-адреса этой сети еще не известен. Как уже было сказано, другие компьютеры локальной сети будут интерпретировать нулевое значение поля netid IP-адреса получателя пакета как “эта сеть”. В большинстве случаев в ответных пакетах будет содержаться реальный префикс сети, поэтому отправитель исходного пакета сможет сохранить его для дальнейшего использования.

В разделе [Протокол IP: обработка ошибок и управляющие сообщения (ICMP)](https://it.rfei.ru/course/%7EGZqN/-/%7EJ26Z) будут более подробно описаны механизмы, с помощью которых компьютер локальной сети может определить идентификатор своей сети.

**Механизмы адресации подсетей и суперсетей**

В описанных механизмах адресации требовалось, чтобы каждой физической сети был назначен уникальный префикс IP-адреса. И хотя это требование было частью первоначального плана, оно не прижилось. В 1980-х годах, когда локальные сети получили широкое распределение, стало очевидным, что при соблюдении требований уникальности префиксов IP-адресов для каждой физической сети все доступное адресное пространство быстро исчерпается. Поэтому для экономии сетевых префиксов механизм адресации был несколько изменен. В результате стало возможным назначать нескольким физическим сетям один и тот же префикс IP-адреса. Подобную систему адресации назвали *адресацией подсетей (subnet addressing)*.

В 1990-х года был придуман еще один механизм расширения адресации, построенный без учета классовой иерархии адресов. Он позволял произвольным образом разбивать IP-адрес на префикс и суффикс. Такую систему адресации назвали *бесклассовой (classless)*, а сами адресуемые сети — *суперсетями (supernet)*. Новая система адресации позволяла более эффективно использовать выделенное адресное пространство.

**Многоадресатная передача**

Кроме *одноадресатной передачи (unicast delivery)*, когда пакеты доставляются на конкретный компьютер, и *широковещательной передачи (broadcast delivery)*, когда пакеты доставляются на все компьютеры конкретной сети, в системе IP-адресации предусмотрен специальный вид многоточечной доставки, который называется *многоадресатной передачей (multicasting delivery)*. В последнем случае пакеты доставляются на заранее определенный набор компьютеров.

Многоадресатная IP-адресация с большим успехом применяется в тех сетях, где оборудование позволяет реализовать режим многоадресатной доставки. Cледует уяснить, что для многоадресатной передачи используются адреса класса D.

**Недостатки IP-адресации**

Выше уже говорилось о том, что в IP-адресе “зашита” информация о самой сети. Однако помещение информации о сети в IP-адрес имеет ряд недостатков. Один из самых очевидных недостатков заключается в том, что IP-адрес относится к сетевому соединению, а не к конкретному компьютеру.

*Поэтому, при перемещении компьютера из одной сети в другую, необходимо изменить его IP-aдpec*.

Чтобы понять, насколько это неудобно, представьте себе абстрактного мобильного пользователя с портативным компьютером, который должен подключаться к Internet в разных географических точках в зависимости от того, куда его забросила судьба. Понятно, что в такой ситуации портативному компьютеру нельзя назначить фиксированный IP-адрес, поскольку сам адрес должен идентифицировать сеть, к которой этот компьютер подключен.

Еще одним недостатком классовой системы адресации является ограниченное количество компьютеров в сети класса С. Как только количество машин в такой сети превосходит 255, необходимо переходить на адреса класса В. И хотя эта проблема на первый взгляд кажется несущественной, изменение адресов в сети может привести к значительным потерям времени и сложности в локализации возможных проблем. Поскольку в большинстве сетевого программного обеспечения не предусмотрена возможность назначения одной и той же физической сети нескольких IP-префиксов, сетевые администраторы не могут спланировать и осуществить постепенный переход к новой системе адресации. Вместо этого они должны менять адреса сразу на всех машинах (со всеми вытекающими последствиями).

Один из существенных недостатков системы межсетевой адресации станет понятен только после ознакомления с принципами маршрутизации. Однако учитывая важность, мы вкратце его здесь опишем. Выше уже шла речь о том, что маршрутизация в объединенной сети выполняется на основе IP-адресов. Для принятия решения о том, куда направлять конкретный пакет, маршрутизатор использует поле netid сетевого адреса. Давайте рассмотрим случай, когда компьютер имеет два физических подключения к объединенной сети. Выше мы уже говорили, что такому компьютеру нужно назначить два IP-адреса. Таким образом будет справедливым приведенное ниже утверждение.

Поскольку при маршрутизации используется поле идентификатора сети IP-адреса, маршрут, по которому будут доставляться пакеты, к компьютеру с несколькими IP-адресами, зависит от того, какой адрес получателя указан.

Из этого утверждения следуют довольно неожиданные выводы. С точки зрения пользователя каждый компьютер в сети логично рассматривать как единое целое и использовать для его идентификации одно имя. Поэтому пользователей часто удивляет тот факт, что некоторым компьютерам назначается несколько имен и что маршрут следования пакетов к этому компьютеру может меняться в зависимости от того, по какому из его имен (точнее, адресов) будут посланы пакеты.

Рис. 8.2. Пример простой объединенной сети с многоадресным узлом В, демонстрирующий недостатки системы IP-адресации. Если интерфейс I3 перестанет функционировать, для обмена пакетами с узлом В через маршрутизатор R следует использовать IP-адрес I5

Еще один неожиданный вывод заключается в том, что иногда недостаточно знать только один из IP-адресов узла сети для обмена с ним данными. Бывают случаи, когда невозможно доставить пакеты по одному из IP-адресов. Давайте в качестве примера рассмотрим простую объединенную сеть, показанную на рис. 8.2.

Как видно из рисунка, к сети 1 подключены два компьютера — A и В, — которые могут напрямую взаимодействовать между собой. Поэтому для обращения к компьютеру В, пользователь компьютера А должен использовать IP-адрес I3. Однако существует и альтернативный маршрут от компьютера А к компьютеру В — через маршрутизатор R. Он используется том случае, когда пользователь компьютера А посылает пакеты по IP-адресу I5 (IP-адрес компьютера В в сети 2). А теперь предположим, что подключение компьютера В к сети 1 перестало функционировать, но сам компьютер В находится в рабочем состоянии. Такое может случиться при повреждении кабеля, с помощью которого компьютер В подключен к сети 1, или выходе из строя соответствующей сетевой платы. Таким образом, пакеты, посланные с компьютера А по адресу I3 не смогут достичь компьютера В, хотя, если их отправить по адресу I5 они достигнут получателя.

Описанная только что проблема соответствия имен и адресов будет рассмотрена в следующих раздеах при обсуждении маршрутизации и привязки имен.

**Точечная десятичная форма записи**

Для удобства обращения, в технической документации или прикладных программах, IP-адрес указывают в виде четырех десятичных чисел, разделенных точками. При этом каждое число представляет один октет IP-адреса1. Таким образом, 32-разрядный IP-адрес 10000000 00001010 00000010 00011110 можно записать как 128.10.2.30.

В этом курсе при выполнении операций с IP-адресами мы будем пользоваться точечной десятичной формой записи. Эта же форма записи используется практически во всех программах семейства протоколов TCP/IP, отображающих значение IP-адреса или требующих от пользователя его ввода. Например, в команде netstat системы UNIX при отображении или вводе IP-адреса пользователем чаще всего применяется точечная десятичная форма записи. (Для тех, кто не знает: эта команда используется для отображения маршрутной информации, сведений о сетевых соединениях и прикладных программах, таких как telnet или ftp.) Поэтому, при использовании классовой системы адресации важно понять, как выражаются IP-адреса разных классов в точечной десятичной форме записи. В табл. 8.1 приведены диапазоны изменения адресов для каждого из классов.

Таблица 8.1. Диапазоны изменения IP-адресов разных классов, выраженных в точечной десятичной форме записи

| **Класс** | **Минимальный адрес** | **Максимальный адрес** |
| --- | --- | --- |
| А | 1.0.0.0 | 126.0.0.0 |
| В | 128.1.0.0 | 191.255.0.0 |
| С | 192.0.1.0 | 223.255.255.0 |
| D | 224.0.0.0 | 239.255.255.255 |
| Е | 240.0.0.0 | 255.255.255.254 |

Примечание. Некоторые значения адресов зарезервированы для специального применения.

1. В точечной десятичной форме записи октеты иногда называют *тетрадами*.

**Адрес петли обратной связи**

Проанализировав содержимое табл. 8.1, можно заметить, что классы сетей охватывают не все возможные диапазоны адресов. В частности префикс сети 127.0.0.0, который по принятой выше классификации должен относиться к сети класса А, зарезервирован для создания так называемой *петли обратной связи (loopback)*. Она предназначена для тестирования работоспособности протокола TCP/IP и реализации механизма межпроцессного взаимодействия на локальном компьютере. Если прикладная программа помещает в поле адреса получателя пакета адрес петли обратной связи, сетевое программное обеспечение обрабатывает пакет так, будто он только что получен по сети. При этом никакие данные в сеть не посылаются. В литературе четко сказано, что пакеты, предназначенные для сети с префиксом 127, не должны передаваться по какой бы то ни было физической сети. Более того, никакие узлы сети или маршрутизаторы не должны распространять информацию о маршрутизации или достижимости сети с адресом 127, поскольку он не является префиксом сетевого адреса.

**Соглашение по использованию специальных адресов**

На практике в системе IP-адресации используется несколько специальных адресов, содержащих в полях netid и hostid различные комбинации нулей (символ “это”) и единиц (символ “для всех”), как показано на рис. 8.3.

Рис. 8.3. Специальные формы IP-адресов, включающие допустимые комбинации нулей (символ “это”) и единиц (символ “для всех”). Длина сетевого префикса направленных широковещательных адресов зависит от класса используемой сети

**Примечания**

1. Разрешено использовать только в процессе инициализации сетевого программного обеспечения. Не является допустимым адресом получателя
2. Не является допустимым адресом отправителя
3. Пакеты с подобным адресом не должны появляться в сети

Как отмечено в примечаниях к рис. 8.3, использование нулевого префикса сети разрешено только при выполнении инициализации сетевого программного обеспечения, которое обычно происходит при начальной загрузке компьютера. Он рассматривается как временная мера, позволяющая компьютерам в локальной сети обменяться начальной информацией. Как только компьютер определит свой IP-адрес и корректный идентификатор сети, он больше не должен использовать нулевой префикс сетевого адреса.

**Управление адресацией в Internet**

В пределах объединенной сети на основе семейства протоколов TCP/IP должно соблюдаться требование уникальности префиксов сетевых адресов. В тех организациях, где технология TCP/IP используется для создания закрытой внутренней сети (т.е. сети, которая не имеет выхода в глобальную сеть Internet), префиксы адресов могут назначаться администрацией сети самостоятельно без согласования с какими бы то ни было организациями. Однако если сеть предприятия планируется подключить к Internet, то в ней нужно использовать уникальные префиксы сетевых адресов, которые не должны совпадать с идентификаторами сетей других организаций. Поэтому, чтобы гарантировать уникальность префиксов IP-адресов по всей глобальной сети Internet, распределение сетевого адресного пространства должно выполняться централизованно. Первоначально контроль над назначением адресов сетей и установка единых правил их использования осуществлялся *Агентство по выделению уникальных параметров протоколов Internet (Internet Assigned Number Authority*, или *IAN А*). С момента основания Internet и до осени 1998 г. назначением адресов ведал один человек — Джон Постел (Jon Postel). Он же руководил работой агентства IANA. После безвременной смерти Джона в конце 1998 г. для управления процессом назначения сетевых адресов была создана некоммерческая *организация по назначению имен и адресов в Internet (Internet Corporation For Assigned Names and Numbers*, или *ICANN*). Эта организация устанавливает единые правила использования и назначает имена, адреса и другие параметры протоколов.

В оригинальной классовой системе адресации, центральные органы Internet назначали адреса в зависимости от размеров сети. Адреса класса С выделялись сетям, к которым было подключено небольшое количество компьютеров (меньше 255), а адреса класса В резервировались для сетей с большим количеством компьютеров. И, наконец, чтобы получить адреса класса А к сети должно было быть подключено больше чем 65535 компьютеров. Адресное пространство выделялось неравномерно, поскольку большинство регистрируемых сетей было небольшого размера. Сетей среднего размера было совсем немного, а огромные сетевые системы вообще можно было пересчитать по пальцам.

Большинство организаций никогда не взаимодействовали с центральными органами Internet напрямую. Для подключения своих сетей к глобальной сети Internet они обычно заключали договора с локальными *поставщиками услуг*, или *провайдерами Internet (Internet Service Provider*, или *ISP*). Кроме обеспечения поддержки соединения между организацией заказчика и остальной частью Internet, провайдеры также назначали корректные префиксы адресов для всех сетей заказчика. На самом деле многие из мелких провайдеров являлись клиентами более крупных провайдеров. Поэтому все запросы на получение префиксов IP-адресов, поступившие от своих клиентов, они пересылали крупным провайдерам. Таким образом, на прямой контакт с ICANN выходили только очень крупные ISP.

Обратите внимание, что центральные органы Internet назначали только сетевую часть IP-адреса. После получения префикса сетевого адреса организация сама должна выбирать способы назначения уникального суффикса IP-адреса каждому компьютеру сети без вступления в контакт с центральными органами. Напомним, что в обязанности центральных органов входит назначение IP-адресов только тем сетям, которые должны быть подключены к глобальной сети Internet.

**Зарезервированные префиксы IР-адресов**

Как уже было сказано, если сеть предприятия не имеет выхода во внешний мир, назначение уникальных адресов для его внутренних сетей TCP/IP является прерогативой администрации самого предприятия. Более того, в подобной ситуации каждая из групп может выбирать диапазоны адресов по своему усмотрению. Например, во внутренней сети фирмы IBM Corporation используются адреса, начинающиеся с 9.0.0.0, а в сети фирмы AT&T— с 12.0.0.0. Таким образом, если организация располагает двумя локальными сетями, в которых планируется использовать протокол TCP/IP без выхода в Internet, системный администратор вполне может назначить для них префиксы адресов 9.0.0.0 и 12.0.0.0.

Однако опыт подсказывает, что неразумно создавать закрытые объединенные сети, в которых используются такие же адреса, как и в глобальной сети Internet. Причина проста: большинство сетей рано или поздно все-таки подключаются к глобальной сети Internet, в результате чего возникают конфликты по адресам с другими сетями. Поэтому, чтобы избежать такой ситуации, инженерная группа IETF зарезервировала несколько префиксов IP-адресов и рекомендовала их использовать исключительно в закрытых сетях. Поскольку в набор зарезервированных префиксов включены как классовые, так и бесклассовые адреса.

**Пример**

Чтобы прояснить систему IP-адресации, давайте рассмотрим в качестве примера топологию двух сетей, существовавших в середине 1980-х годов на факультете вычислительной техники университета Пердью, и покажем, как они были подключены к Internet. На рис. 8.4 показаны адреса сетей и схема их взаимодействия с помощью маршрутизаторов.

Рис. 8.4. Схема подключения двух сетей к магистральному каналу Internet. Каждой сети был назначен уникальный префикс IP-aдpeca

В приведенном выше примере задействованы три сети, которым назначены следующие адреса:

* ARPANET — 10.0.0.0;
* Ethernet — 128.10.0.0;
* Token ring — 192.5.48.0.

Согласно табл. 8.1, эти сети относятся к классам А, В и С, соответственно. На рис. 8.5 показана структурная схема двух факультетских сетей с именами подключенных к ним компьютеров и IP-адресами, назначенными для каждого сетевого соединения.

Рис. 8.5. Пример назначения IP-адресов для маршрутизаторов и компьютеров, подключенных к сетям, изображенным на рис. 8.4.

Как видно из рисунка, к объединенной сети подключено четыре компьютера: Arthur, Merlin, Guenevere и Lancelot. Два компьютера объединенной сети выполняют функции маршрутизатора пакетов. Это машина Taliesyn, которая связывает сети ARPANET и Token Ring, и машина Glatisant, связывающая сети Token Ring и Ethernet. Машина Merlin подключена как к сети Ethernet, так и к сети Token Ring, поэтому она может напрямую взаимодействовать с абонентами этих сетей. И хотя любой многоадресный узел, типа Merlin, можно настроить так, чтобы он выполнял маршрутизацию пакетов между двумя сетями, чаще всего для этих целей используются специальные выделенные компьютеры, или маршрутизаторы. Причина заключается в том, что выполнение функций маршрутизации требует привлечения дополнительных ресурсов как центрального процессора, так и оперативной памяти. Особенно это относится к сетям с большим трафиком. Поэтому, как показано на рис. 8.5, функции маршрутизации трафика между сетями Ethernet и Token Ring выполняет отдельный компьютер Glatisant. (Реальный поток данных между этими двумя сетями существенно выше, чем предполагается в рассматриваемом нами примере. Причина в том, что для упрощения на рисунке показано только несколько машин, подключенных к объединенной сети, а на самом деле их гораздо больше.)

Как показано на рис. 8.5, каждому сетевому подключению назначен свой IP-адрес. Так, машине Lancelot назначен адрес 128.10.2.26, поскольку она имеет единственное подключение к сети Ethernet. Машине Merlin назначен IP-адрес 128.10.2.3 для интерфейса сети Ethernet и адрес 192.5.48.3 — для интерфейса сети Token Ring. Обратите внимание, что в рассматриваемом примере при назначении адресов для многоадресного узла сети для удобства сохранено значение его младшего байта адреса. Однако для маршрутизаторов Glatisant и Taliesyn адреса назначены несколько иначе. Например, для интерфейсов машины Taliesyn выбраны адреса 10.0.0.37 и 192.5.48.6, т.е. две строки цифр, абсолютно не связанных между собой. С точки зрения протокола IP не имеет значения, связаны как-то между собой цифры адреса компьютера, выраженные в точечной десятичной форме записи, или нет. Однако назначение компьютерам IP-адресов, цифры в которых как-то связаны между собой или имеют какое-либо мнемоническое значение, облегчает жизнь обслуживающим сеть специалистам, системным администраторам и менеджерам.

Таким образом, если для компьютера с несколькими интерфейсами назначить IP-адреса так, чтобы цифры в последнем октете совпадали, то обслуживающему персоналу легче их запомнить или идентифицировать интерфейс в конкретной сети.

**Порядок следования байтов в сети**

Чтобы работоспособность объединенной сети не зависела от используемых в ней компьютеров и сетевого оборудования, нужно определить единый вид представления данных с точки зрения программного обеспечения. Представьте себе, что произойдет, если, к примеру, программа, работающая на одном компьютере, отправит 32-разрядное целое число на другой компьютер. Сетевое оборудование, работающее на физическом уровне, передает последовательность бит с одной машины на другую без изменения порядка их следования.

Однако не во всех типах компьютерных систем 32-х разрядные целые числа хранятся в одном и том же формате. В некоторых — младший по значимости байт целого числа хранится по меньшему адресу. Такой порядок следования байтов называется *прямым (Little Endian)*.

В других системах по меньшему адресу хранится старший по значимости байт целого числа. Подобный порядок следования байтов называется *обратным (Big Endian)*.

Кроме этих двух, существуют компьютерные системы, в которых целые числа составляются из групп 16-битовых слов. При этом по меньшему адресу хранится младшее по значимости слово, байты которого переставлены местами. Таким образом, прямое копирование байтов с одного компьютера на другой без изменения порядка их следования может привести к искажению значения целых чисел.

Стандартизация порядка следования байтов в целых числах особенно важна в объединенной сети, поскольку такая информация, как адрес получателя или длина пакета, передается в виде бинарных данных. Естественно, что эти данные должны быть правильно интерпретированы как отправителем, так и получателем.

В семействе протоколов TCP/IP описанная выше проблема решается за счет введения *стандартного* порядка следования байтов в сети, который должны использовать все компьютеры при помещении в пакет полей, содержащих двоичные данные. Перед отправкой пакета по сети, каждый компьютер или маршрутизатор должен преобразовать двоичные данные из машинного формата в стандартный. Аналогично, перед обработкой полученного пакета компьютер должен выполнить обратное преобразование. Естественно, что преобразование форматов содержащихся в пакете пользовательских данных, не выполняется, поскольку для протокола TCP/IP они представляют собой “черный ящик”. Прикладные программы могут по своему усмотрению форматировать и преобразовывать данные.

В большинстве случаев для представления в пакете двоичных данных в сетевых прикладных программах выбирается стандартный порядок следования байтов, как и в протоколе TCP/IP. То есть эта данная проблема решается программистами уже на этапе написания приложений. Поэтому запускающий программы пользователь никогда не сталкивается с несовместимостью порядка следования байтов в полученных данных.

Для объединенной сети принят обратный (*Big Endian*) порядок следования байтов, т.е. самый старший байт передается первым. Таким образом, если попытаться представить себе пакет данных, который побайтно передается от одной машины к другой, то старший значащий байт целого двоичного числа, содержащегося в этом пакете, будет располагаться ближе к началу пакета, а самый младший значащий байт — ближе к его концу. По поводу формата представления данных в свое время было много дискуссий и до сих пор принятый стандарт периодически подвергается нападкам со стороны специалистов. В частности, ведутся споры по поводу того, что обратный порядок следования байтов уже давно устарел, поскольку практически во всех современных компьютерах для представления двоичных чисел используется прямой порядок следования байтов. Проблема состоит в том, что много лет тому назад, когда принимался этот стандарт, в большинстве компьютеров использовался как раз обратный порядок следования байтов. Тем не менее каждый согласится с тем, что ключевым в этом споре является сам факт принятия стандарта, а каков этот стандарт — уже второстепенно. Поэтому пока стоит смириться с таким положением дел.

**Проверка знаний: Классовая адресация**

**Задание 1**

Продолжите верно фразу.

Префикс сети 127.0.0.0 зарезервирован для создания...

адреса обратной связи (петли).

адреса двусторонней связи.

временного адреса связи.

**Задание 2**

Почему, при перемещении компьютера из одной сети в другую, необходимо изменить его IP-aдpec?

IP-адрес относится к сетевому соединению, а не к конкретному компьютеру

нет необходимости изменять IP-адрес

IP-адрес относится к конкретному компьютеру, а не к сетевому соединению

**Задание 3**

Как звали того, единственного человека, который занимался назначением адресов сетей до 1998 г.?

Томас Харрис

Джек Пэланс

Джон Постел

**Задание 4**

Что понимается под *суперсетями*?

Такая система, в которой разбивался IP-адрес на суффикс.

Такая система, в которой разбивался IP-адрес на префикс и суффикс.

Такая система, в которой разбивался IP-адрес на префикс.

**Задание 5**

IP-адрес, поле *hostid* которого состоит из одних нулей, интерпретируется как?

«этот» узел

«эта сеть»

«для всех узлов»

**Задание 6**

Для чего используется IP-адрес с нулевым полем *hostid*?

для обозначения самой сети

для направленной широковещательной передачи

для назначения отдельному узлу сети

**Задание 7**

Закончите следующую фразу.

Проблему порядка следования байтов в сети решили введением

порядка.

**Задание 8**

Почему направленные широковещательные адреса могут однозначно интерпретироваться в любой точке объединенной сети?

поскольку можно выделить уникальный IP-адрес получателя

поскольку можно выделить уникальный идентификатор сети получателя

поскольку можно выделить уникальный идентификатор узла

**Задание 9**

Можем ли мы назначить маршрутизатору один IP-адрес?

да

это возможно, но нужно соблюсти некоторые условия

нет

**Задание 10**

Какова разрядность адреса, назначаемая каждому узлу объединенной сети TCP/IP?

32

64

16

**Преобразование IP-адресов в физические адреса (ARP)**

В предыдущем разделе была описана система адресации протокола TCP/IP, согласно которой каждому узлу сети назначается уникальный 32-х разрядный адрес. Было сказано, что функционирование объединенной сети напоминает одну большую виртуальную сеть, поскольку при отправке и получении пакетов используются заранее назначенные адреса узлов сети.

Были также рассмотрены аппаратные сетевые технологии и отмечено, что два компьютера, подключенные к одной физической сети, могут взаимодействовать между собой *только при условии, что им известны, физические адреса друг друга*. При этом за кадром остался вопрос о том, как узел сети или маршрутизатор при отправке пакета по физической сети преобразует IP-адрес в корректный физический адрес. Этот вопрос и будет рассмотрен в данном разделе на примере двух самых распространенных физических систем адресации.

* [Необходимость преобразования адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690500)
* [Два типа физических адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690501)
* [Преобразование адресов методом прямого отображения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690502)
* [Преобразование адресов методом динамической привязки](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690503)
* [Кэширование запросов ARP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690504)
* [Тайм-аут кэширования запросов ARP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690505)
* [Усовершенствования протокола ARP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690506)
* [Проверка знаний: Преобразование IP-адресов в физические адреса (ARP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690507)

**Необходимость преобразования адресов**

Давайте рассмотрим две машины *А* и *В*, подключенные к одной физической сети. Каждой из них назначен уникальный IP-адрес — IA и IB — и, кроме того, платы сетевого интерфейса этих машин имеют также свои физические адреса РA и РB.

Наша цель — создать низкоуровневое программное обеспечение, которое позволит программам высокого уровня, игнорируя физические адреса, взаимодействовать между собой с помощью IP-адресов. Однако не стоит забывать о том, что в конечном счете взаимодействие между двумя машинами осуществляется с помощью физической сети. При этом может использоваться только та система физической адресации узлов, которая поддерживается конкретным сетевым оборудованием.

Предположим, что машина *А* собирается отправить пакет машине *В* по физической сети, к которой они обе подключены. Однако машине *А* известен только IP-адрес (IB) машины *В*. При этом возникает вопрос: как машина *А* сможет определить физический адрес (РB) машины *В*?

Следует заметить, что преобразование адресов должно выполняться на каждом участке по пути следования пакета от отправителя до конечного получателя. В частности, возможно возникновение одной из двух ситуаций.

Во-первых, на последнем этапе доставки пакета, его нужно отправить конечному получателю по той же физической сети. В этом случае компьютер, отправляющий пакет, должен преобразовать IP-адрес конечного получателя в физический адрес.

Во-вторых, на промежуточных участках по пути следования пакет должен отправляться на очередной узел маршрутизации. Поэтому промежуточный отправитель должен преобразовать IP-адрес следующего узла маршрутизации в физический адрес.

Описанная выше задача конвертирования адресов высокого уровня в физические адреса называется *проблемой преобразования адресов (address resolution problem)* и решается несколькими способами. В некоторых семействах протоколов ведется специальная таблица соответствия адресов высокого уровня физическим адресам. Эта таблица хранится на каждом узле сети. В других семействах протоколов проблема решается путем кодирования аппаратного адреса и создания на его основе сетевого адреса высокого уровня. Следует заметить, что при использовании только одного из приведенных выше методов редко удается создать эффективную систему адресации высокого уровня. Поэтому ниже будут рассмотрены два способа преобразования адресов, используемые в семействе протоколов TCP/IP, а также указано, когда наиболее уместно использовать каждый из них.

**Два типа физических адресов**

Существует два типа физических адресов. Один из них представляет собой двоичное целое число высокой разрядности, которое назначается в качестве физического адреса при изготовлении устройства и в дальнейшем не может быть изменено. Второй тип физического адреса — небольшое целое число, которое назначается в качестве физического адреса сетевым администратором при установке сетевой платы в компьютер и может быть при необходимости изменено. Таким образом, проблема преобразования адресов для сетей наподобие proNET решается очень просто, чего нельзя сказать для сетей типа Ethernet. Начнем с рассмотрения легкой задачи.

**Преобразование адресов методом прямого отображения**

Рассмотрим сеть с передачей маркера типа proNET. В качестве физических адресов в ней используются небольшие целые числа и пользователь должен выбрать значение физического адреса непосредственно перед установкой платы сетевого интерфейса в компьютер.

Ключевой момент, облегчающий процесс преобразования адресов в сети данного типа заключается в том, что в процессе инсталляции сетевому адаптеру нужно назначить как IP-адрес, так и физический адрес. Следовательно, ничто не мешает выбрать эти адреса так, чтобы их части совпадали.

Обычно IP-адреса назначаются так, что значение их поля hostid увеличивается на единицу при переходе от одной машины к другой, например 1, 2, 3. Поэтому при установке платы сетевого интерфейса можно выбрать ее физический адрес так, чтобы он соответствовал значению поля hostid IP-адреса. Например, системный администратор вполне может назначить физический адрес 3 компьютеру, IP-адрес которого равен 192.5.48.3, поскольку последний относится к классу С и его поле hostid равно 3.

Таким образом, для сетей типа proNET задача определения физического адреса по IP-адресу является очень простой. Достаточно из IP-адреса извлечь значение поля hostid. Эта операция на всех компьютерных платформах выполняется очень быстро, поскольку для ее реализации требуется всего несколько машинных команд. Описанный выше процесс преобразования адреса эффективен еще и потому, что для его выполнения не требуется обращаться ко внешним источникам данных. И наконец, при подключении новых компьютеров к сети не нужно изменять старые установки или перекомпилировать код.

Выражаясь математическим языком, при назначении системы нумерации, которая бы позволяла эффективно преобразовывать адреса, нужно выбрать функцию *f*, с помощью которой по IP-адресу будет вычисляться физический адрес. В зависимости от используемого оборудования разработчики могут также выбирать ту или иную систему физических адресов. Таким образом, преобразование адреса IA сводится к вычислению функции

РA= f(IA)

Наша задача сделать так, чтобы функция *f* вычислялась с максимальной эффективностью. Если набор физических адресов ограничен, то эффективное преобразование адресов можно организовать иначе, чем в приведенном примере. Например, при использовании протокола IP в сети, требующей установки соединения, наподобие ATM, отсутствует возможность выбора физических адресов. В таких сетях на одном или нескольких компьютерах, выполняющих роль серверов, хранятся пары адресов. Каждая пара представляет собой IP-адрес и соответствующий ему физический адрес. Обычно для ускорения поиска серверы хранят пары адресов в таблице, находящейся в оперативной памяти. В подобных случаях для повышения эффективности поиска соответствия в таблице программы могут использовать заранее оговоренную хеш-функцию.

**Преобразование адресов методом динамической привязки**

Чтобы понять, почему преобразование адреса в некоторых сетях трудно осуществить, давайте в качестве примера рассмотрим технологию Ethernet. Каждой плате сетевого интерфейса Ethernet назначается уникальный 48-битовый адрес на заводе-изготовителе. Поэтому физический адрес компьютера может измениться, если плата сетевого интерфейса выйдет из строя и будет заменена. Более того, поскольку адрес Ethernet состоит из 48 битов, нет никаких шансов его закодировать в 32-х разрядный IP-адрес1.

Разработчики семейства протоколов TCP/IP подошли творчески к решению проблемы преобразования адресов в сетях, поддерживающих возможность широковещательной передачи данных, наподобие Ethernet. Придуманный ими метод позволяет добавлять к сети новые узлы или маршрутизаторы без необходимости перекомпиляции исходного кода сетевого программного обеспечения и без поддержки централизованной базы данных. Чтобы избежать необходимости поддерживать таблицу соответствия адресов, разработчики предпочли использовать низкоуровневый протокол динамической привязки адресов, который назвали *протоколом преобразования адресов (Address Resolution Protocol*, или *ARP*). С его помощью удалось реализовать довольно эффективный и несложный механизм конвертирования адресов.

Рис. 9.1. Схема работы протокола ARP. Чтобы определить физический адрес (РB) узла В, узел А отправляет широковещательный запрос, содержащий IP-адрес (IB) узла В, ко всем машинам сети (а). В ответ на запрос узел В отправляет узлу А пару значений (IB, РB) (б)

Как показано на рис. 9.1, идея, положенная в основу механизма динамической привязки протокола ARP, очень проста. Как только узлу сети *А* требуется преобразовать IP-адрес *IB*, он посылает в режиме широковещания специальный пакет, который предписывает узлу с IP-адресом *IB* сообщить свой физический адрес (*PB*). Данный пакет будет получен всеми узлами сети, включая *В*. Однако ответ, содержащий физический адрес, посылает только узел *В*, поскольку только ему назначен IP-адрес *IB*. Получив ответный пакет, узел *А* использует присланный ему физический адрес для непосредственной отправки пакета узлу *В*. Таким образом, можно подвести некоторые итоги.

Протокол преобразования адресов ARP позволяет узлу сети узнать физический адрес узла-получателя, подключенного к той же физической сети, используя при этом только IP-адрес получателя.

1. Поскольку метод прямого отображения, рассмотренный в предыдущем разделе является более удобным и эффективным, чем метод динамической привязки, новое поколение протоколов IPv6 создано с учетом возможности кодирования 48-битового аппаратного адреса в IP-адресе.

**Кэширование запросов ARP**

На первый взгляд может показаться весьма странным, что, прежде чем машина *А* сможет отправлять пакеты машине *В*, она должна послать широковещательный запрос всем машинам сети, который достигает в том числе и машины *В*. Еще более странным кажется то, что машина *А* сначала посылает широковещательный запрос, в котором, выражаясь человеческим языком, содержится вопрос: “Как мне связаться с машиной *В*?”, вместо того, чтобы сразу послать пакет машине *В* в широковещательном режиме. Однако для этого существует весомая причина. Все дело в том, что широковещательный режим — слишком дорогой и ресурсоемкий, чтобы использовать его для простого обмена пакетами между двумя машинами. В самом деле, пакет, посланный в этом режиме, получают все машины сети, и каждая из них (кроме одной) должна впустую потратить время на его обработку.

Чтобы уменьшить накладные расходы, программы, использующие ARP, должны кэшировать ответы на несколько последних сделанных запросов, содержащих информацию об IP-адресах и соответствующих им физических адресах. Другими словами, как только программа получает ответ на посланный ARP-запрос, она сохраняет IP-адрес и соответствующий ему физический адрес в локальной памяти компьютера для дальнейшего использования. Перед отправкой нового ARP-запроса программа всегда сначала просматривает кэш-память. Если нужное соответствие адресов найдено, широковещательный ARP-запрос в сеть не посылается.

Таким образом, если двум компьютерам в сети нужно обменяться информацией, они вначале шлют друг другу широковещательные ARP-запросы. После получения ответа все последующие передачи пакетов выполняются уже напрямую, без предварительных ARP-запросов.

Опыт эксплуатации сетей показывает, что, поскольку в большинстве сеансов обмена информацией пересылается более одного пакета данных, использование кэш-памяти даже небольшого размера уже имеет смысл.

**Тайм-аут кэширования запросов ARP**

Рассмотренный выше механизм кэширования запросов ARP представляет собой один из примеров применения технологии систем с *неустойчивым состоянием (soft state)*. Эта технология широко используется в сетевом программном обеспечении. Ее название говорит о том, что возможны ситуации, когда информация, содержащаяся у клиента, устаревает, а он об этом даже ничего не подозревает, поскольку никакие уведомления он не получает.

Возвращаясь к ARP, давайте рассмотрим две машины, *А* и *В*, подключенные к сети Ethernet. Предположим, что машина *А* посылает ARP-запрос, на который машина *В* отвечает и после этого выходит из строя. При этом машина *А* не получит уведомления о том, что с машиной *В* что-то случилось. Более того, поскольку в кэш-памяти ARP-запросов машины *А* находится адресная привязка для машины *В*, машина *А* будет продолжать напрямую посылать пакеты машине *В*. В аппаратном обеспечении Ethernet не предусмотрены средства для контроля за состоянием машины *В*, поскольку в технологии Ethernet отсутствует механизм гарантированной доставки пакетов. Поэтому машина *А* не располагает средствами, позволяющими установить момент, когда информация, хранящаяся в ее кэш-памяти ARP-запросов, не соответствует действительности.

Таким образом, в системах с неустойчивым состоянием ответственность за достоверность информации возложена на владельцев этой информации. На практике это означает, что в сетевых программах, реализующих подобные системы, используются специальные таймеры, определяющие интервал времени, в течение которого информация считается достоверной. По истечении установленного времени какая бы то ни было информация о состоянии системы удаляется из кэш-памяти. Например, после того, как информация об адресной привязке помещена в кэш-память ARP-запросов, протокол ARP требует, чтобы программа установила значение таймера, равное, как правило, 20 минутам. По истечении установленного времени информация из кэш-памяти должна быть удалена.

После этого события могут развиваться по одному из двух сценариев.

Во-первых, если никакие пакеты больше не будут посылаться получателю, проблема решается сама собой.

Во-вторых, если получателю нужно отправить пакет, а в кэш-памяти заданная адресная привязка отсутствует, программа выполняет обычную процедуру посылки широковещательного ARP-запроса и обработки его ответа. Если к этому моменту машина получателя возобновила работу, информация об адресной привязке снова помещается в кэш-память ARP-запросов отправителя. Если же нет, отправитель отмечает тот факт, что машина получателя недоступна.

Использование в протоколе ARP алгоритмов с неустойчивым состоянием имеет как преимущества, так и недостатки. Основное преимущество заключается в независимости подобных систем.

Во-первых, каждый компьютер может независимо от других компьютеров сети определять момент, когда нужно обновить информацию в кэш-памяти ARP-запросов.

Во-вторых, отправителю для определения факта неправильной адресной привязки не нужно связываться с машиной получателя или какой бы то ни было другой машиной. Достаточно отправить ARP-запрос, и если предполагаемый получатель на него не ответит, значит, его компьютер по какой-либо причине отключен от сети.

В-третьих, предложенная методика обеспечивает надежный механизм передачи сообщений независимо от типа используемого сетевого оборудования.

Основной недостаток применения алгоритмов с неустойчивым состоянием заключается в возникающих при этом задержках. В самом деле, если величина тайм-аута равняется *N* секунд, то отправитель узнает о том, что с получателем что-то случилось только по прошествии *N* секунд.

**Усовершенствования протокола ARP**

В протокол ARP было внесено несколько усовершенствований.

Во-первых, очевидно, что, если машина *А* посылает в сеть ARP-запрос для определения физического адреса машины *В*, чтобы отправить ей пакет данных, высока вероятность того, что впоследствии машине *В* потребуется отправить пакет машине *А*. Поэтому, чтобы упредить ARP-запрос от машины *В* в отношении физического адреса машины *А* и таким образом минимизировать лишний трафик в сети, посылая ARP-запрос, машина *А* должна поместить в него свою адресную привязку (IP-адрес машины *А* и соответствующий ему физический адрес). При получении запроса машина *В* извлекает из него адресную привязку машины *А* и помещает ее в свою кэш-память, после чего посылает ответ машине *А*.

Во-вторых, поскольку машина *А* посылает в сеть исходный ARP-запрос в широковещательном режиме, его принимают все машины сети. А это означает, что они могут извлечь из него адресную привязку машины *А* и поместить ее в свою кэш-память.

В-третьих, при замене платы сетевого интерфейса (например, в случае выхода ее из строя) меняется физический адрес компьютера. А это означает, что необходимо предусмотреть механизм оповещения компьютеров локальной сети, в кэш-памяти ARP-запросов которых могла остаться старая адресная привязка. Компьютер может уведомить других абонентов сети о своем новом адресе, послав широковещательный ARP-запрос в момент начальной загрузки.

Приведенные выше соображения можно подытожить следующим образом.

В каждый ARP-запрос отправитель включает свою адресную привязку (IP-адрес и соответствующий ежу физический адрес). Получатели помещают эту привязку в свою кэш-память перед обработкой полученного ARP-пакета.

**Проверка знаний: Преобразование IP-адресов в физические адреса (ARP)**

**Задание 1**

Основное преимущество ARP алгоритмов с неустойчивым состоянием — это...

независимость

скорость

безопасность

**Задание 2**

Как часто должно выполняться преобразование адресов по пути следования пакета?

на каждом участке

на участке получателя

на участке отправителя

**Задание 3**

Протокол преобразования адресов ARP позволяет узлу сети узнать физический адрес узла-получателя, подключенного к той же физической сети, используя при этом только...

IP-адрес отправителя

IP-адрес получателя

IP-адрес получателя и отправителя

**Задание 4**

Обычно IP-адреса назначаются так, что значение их поля hostid увеличивается на определенное количество единиц при переходе от одной машины к другой, на сколько?

на 2

на 4

на 1

**Протокол IP: доставка дейтаграмм без установки соединения**

В предыдущих разделах были рассмотрены сетевое оборудование и программное обеспечение, обеспечивающие межсетевое взаимодействие. Кроме того, были описаны низкоуровневые сетевые технологии и преобразование адресов.

В этом разделе будет рассмотрен основополагающий принцип доставки пакетов данных без установки соединения, используемый в *межсетевом протоколе (Internet Protocol*, или *IP)*. Этот протокол является одним из двух основных протоколов, применяемых в процессе межсетевого взаимодействия (вторым протоколом является TCP).

* [Виртуальная сеть](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69050f)
* [Структура и принцип действия объединенной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690510)
* [Принцип организации служб](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690511)
* [Служба доставки, не требующая установки соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690512)
* [Назначение протокола IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690513)
* [Межсетевая дейтаграмма](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690514)
* [Формат дейтаграммы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690515)
* [Способы обработки дейтаграммы и схемы дифференцированного обслуживания](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690516)
* [Инкапсуляция дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690517)
* [Размеры дейтаграммы, MTU и процесс фрагментации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690518)
* [Сборка фрагментов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690519)
* [Управление фрагментацией](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051a)
* [Время жизни дейтаграммы (TTL)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051b)
* [Остальные поля заголовка дейтаграммы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051c)
* [Параметры IP-дейтаграммы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051d)
* [Параметры регистрации маршрута следования](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051e)
* [Параметры маршрутизации от источника](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69051f)
* [Параметр регистрации временных меток](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690520)
* [Обработка параметров при фрагментации дейтаграммы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690521)
* [Проверка знаний: Доставка дейтаграмм без установки соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690522)

**Виртуальная сеть**

В разделе [Основы и структура межсетевого взаимодействия](https://it.rfei.ru/course/%7EGZqN/-/%7EQNE3) была рассмотрена структура объединенной сети, состоящей из нескольких физических сетей, соединенных между собой посредством маршрутизаторов. Рассмотрение структуры сети не должно вводить вас в заблуждение, поскольку основное внимание уделяется не технологии межсетевого взаимодействия, а интерфейсу, который предоставляет объединенная сеть для пользователей.

С точки зрения пользователей объединенная сеть представляет собой единую виртуальную сеть, соединяющую между собой все компьютеры. Благодаря этому можно легко осуществить взаимодействие между любыми двумя машинами объединенной сети. При этом внутренняя структура физических сетей не имеет значения — она скрыта от пользователя.

В каком-то смысле, объединенная сеть является обобщенной физической сетью, поскольку на самом нижнем уровне она обеспечивает те же функциональные возможности — прием пакетов и доставку их по назначению. На ее основе межсетевым программным обеспечением более высокого уровня создаются развитые службы, используемые пользователями.

**Структура и принцип действия объединенной сети**

Объединенную сеть на основе протокола TCP/IP можно разделить на три логических уровня, как показано на рис. 10.1. Положение уровней на рисунке соответствует существующим между ними взаимосвязям. На самом нижнем уровне находится служба доставки пакетов, не требующая установки соединения. Она является своего рода фундаментом, на котором построены все остальные сетевые службы. На следующем уровне располагается надежная транспортная служба, которая обеспечивает для прикладных программ высокоуровневый механизм передачи данных. Каждая из этих служб будет описана чуть ниже; мы рассмотрим выполняемые ими функции и используемые при этом протоколы.

Рис. 10.1. Три логических уровня служб объединенной сети

**Принцип организации служб**

Каждой службе, показанной на рис. 10.1, можно сопоставить соответствующее протокольное программное обеспечение. Однако мы разбили весь спектр служб, предоставляемых объединенной сетью, на три логические составляющие части, поскольку такой подход позволяет прояснить основополагающие принципы, лежащие в основе проектирования объединенной сети.

Программное обеспечение для объединенной сети разрабатывалось с учетом его принадлежности к одной из трех логических сетевых служб, организованных в иерархическом порядке. Его успешное применение в большей степени обусловлено тем, что такая логическая структура является необычайно надежной и гибкой.

Одним из существенных преимуществ описанной логической структуры является возможность замены одной службы без влияния на все остальные службы. Таким образом, выполнение исследований и разработка новых сетевых служб может выполняться параллельно, независимо от логического уровня, который занимает служба в иерархической структуре.

**Служба доставки, не требующая установки соединения**

Служба доставки пакетов является одной из основных в объединенной сети. Формально, ее можно охарактеризовать как ненадежную, не гарантирующую доставку пакетов, и не требующую предварительной установки соединения с получателем. Рассматриваемую службу можно сравнить с сетевой аппаратной технологией, такой как Ethernet, предпринимающей максимум усилий для доставки пакетов, но не гарантирующей их доставку получателю.

Служба названа *ненадежной*, поскольку она не гарантирует доставку пакетов получателю. Это означает, что по пути следования пакет может быть утерян, продублирован, задержан или доставлен с нарушением порядка следования. При этом служба доставки не сможет обнаружить перечисленные выше проблемы и сообщить о них отправителю и получателю.

Служба доставки *не требует установки соединения*, поскольку каждый пакет считается независимым от остальных. Последовательности пакетов, передаваемые между двумя компьютерами, могут проходить по разным маршрутам. При этом некоторые пакеты могут быть утеряны и не достигнуть компьютера получателя. И наконец, можно сказать, что рассматриваемая нами служба не обеспечивает каких-либо усилий для доставки пакетов, поскольку межсетевое программное обеспечение предпринимает для этого эффективные меры. Другими словами, в объединенной сети пакеты просто так не пропадают. Надежность системы нарушается только в случае перегрузки какого-либо сетевого ресурса или повреждении участка сети.

**Назначение протокола IP**

Протокол, в котором используется ненадежный, не требующий установки соединения с получателем механизм доставки, назвали *межсетевым протоколом*, или *Internet Protocol*. Однако чаще всего для обозначения этого протокола используют аббревиатуру IP1.

В протоколе IP сделано три важных определения.

Во-первых, в нем определен базовый элемент передачи данных, используемый во всей объединенной сети на основе протокола TCP/IP. Это означает, что в протоколе IP четко определен формат передаваемых по объединенной сети данных.

Во-вторых, программы поддержки протокола IP выполняют функцию *маршрутизации*, которая заключается в выборе пути, по которому будут посылаться данные.

В-третьих, помимо точной, формальной спецификации форматов данных и функций маршрутизации, в протокол IP включен набор правил, воплощающих в жизнь идею ненадежной доставки пакетов данных.

В этих правилах оговариваются способы обработки пакетов узлами сети и маршрутизаторами, а также условия, при возникновении которых должны генерироваться сообщения об ошибке и удаляться пакеты. Протокол IP стал неотъемлемой частью структуры всей объединенной сети, причем настолько, что сети, в которых используется протокол TCP/IP, иногда называют сетями на основе *IP-технологии*.

1. От аббревиатуры IP происходит термин *IP-aдpec*.

**Межсетевая дейтаграмма**

Между физической сетью и объединенной сетью на основе протокола TCP/IP имеется большое сходство.

В физической сети единицей передачи информации является фрейм, который состоит из заголовка и блока данных. В заголовке фрейма содержится служебная информация, а также физические адреса отправителя и получателя.

В объединенной сети единицей передачи информации является *межсетевая дейтаграмма (Internet datagram)*, которую иногда называют *IP-дейтаграммой*, или просто *дейтаграммой*.

Как и обычный сетевой фрейм, дейтаграмма состоит из заголовка и блока данных. По аналогии с фреймом, в заголовке дейтаграммы указываются адреса отправителя и получателя пакета, а также поле типа, которое позволяет определить ее содержимое. Различие состоит в том, что в заголовке дейтаграммы находятся IP-адреса, а в заголовке фрейма — физические адреса.

Общий вид дейтаграммы представлен на рис. 10.2.

Рис. 10.2. Общий вид IP-дейтаграммы, которая является аналогом сетевого фрейма. В протоколе IP оговаривается формат ее заголовка, содержащий адреса отправителя и получателя. Формат области данных не оговаривается, поэтому в дейтаграмме можно передать информацию произвольного объема

**Формат дейтаграммы**

После описания общего вида IP-дейтаграммы пришло время поговорить о ее содержимом более подробно. Расположение полей дейтаграммы показано на рис. 10.3.

Рис. 10.3. Формат IP-дейтаграммы — основной единицы передачи данных в объединенной сети на основе протокола TCP/IP

Поскольку обработкой дейтаграмм занимается специальное протокольное программное обеспечение, их содержимое и формат не должны привязываться к какому бы то ни было сетевому оборудованию. В первом поле заголовка дейтаграммы (его длина составляет 4 бита) указывается номер версии протокола IP, которому соответствует дейтаграмма. Этот номер устанавливается отправителем и используется для проверки согласованности форматов дейтаграмм получателем, а также промежуточными маршрутизаторами, через которые проходят пакеты. Перед обработкой дейтаграммы программное обеспечение протокола IP, установленное на компьютере, должно сверить версию протокола. Эта операция гарантирует соответствие формата дейтаграммы обрабатывающему ее программному обеспечению. В случае изменения формата дейтаграммы изменится и версия протокола IP. При несоответствии версий программного обеспечения и дейтаграммы, обработка последней не производится. Это гарантирует, что данные дейтаграммы не будут некорректно интерпретированы устаревшим программным обеспечением. Текущей версией протокола IP является 4. Поэтому для ее идентификации часто используется аббревиатура *IPv4*.

Поле, в котором содержится длина заголовка дейтаграммы, выраженная в 32-разрядных словах, также имеет длину 4 бита. Как вы увидите в дальнейшем, все поля заголовка имеют фиксированную длину, кроме поля, в котором указываются параметры протокола IP, и соответствующего ему поля выравнивания. Длина типового заголовка IP-дейтаграммы, не содержащей дополнительных параметров и поэтому не имеющей поля выравнивания, составляет 20 октетов. При этом в поле длины заголовка дейтаграммы заносится цифра 5.

Как следует из названия, в поле *общей длины дейтаграммы* заносится общий размер дейтаграммы (включая заголовок и область данных), выраженный в октетах. Размер области данных можно легко вычислить, вычтя из общей длины дейтаграммы размер заголовка, предварительно выразив его в октетах. Поскольку длина поля, в которое помещается длина дейтаграммы, составляет 16 бит, максимально возможный размер IP-дейтаграммы составляет 216–1 или 65 535 октетов. Для большинства приложений это ограничение не является слишком жестким. Однако, учитывая наметившуюся в последнее время тенденцию к повышению пропускной способности каналов связи, в будущем вполне могут появиться высокоскоростные сетевые технологии, в которых размер пакета данных превысит 65 535 октетов.

**Способы обработки дейтаграммы и схемы дифференцированного обслуживания**

В поле типа обслуживания (чаще всего его называют *типом сервиса*, или *TOS*), размер которого составляет 8 бит, указывается способ обработки дейтаграммы. Это поле первоначально было разделено на пять подполей, как показано на рис. 10.4.

Рис. 10.4. Оригинальная разбивка на пять подполей поля типа обслуживания

В поле приоритета, длина которого составляет три бита, указывается важность дейтаграммы. Значение 0 соответствует обычной дейтаграмме, а 7 — сетевому управляющему пакету. Отправитель может также установить в этом поле любые промежуточные значения, соответствующие важности каждой посылаемой дейтаграммы. И хотя некоторые маршрутизаторы игнорируют значение этого поля, оно имеет важное значение, поскольку позволяет установить приоритет управляющих сетевых пакетов, над обычными пакетами данных. Например, многие маршрутизаторы при обмене служебной информацией помещают в поле приоритета значение 6 или 7. Это позволяет им обменяться пакетами даже в том случае, когда сеть сильно перегружена.

Биты D, Т, и R определяют желательный тип транспортировки дейтаграммы. При установке бита D дейтаграмма должна быть передана с минимальной задержкой. При установленном бите Т дейтаграмму нужно передавать по каналу с высокой пропускной способностью, а бит R — говорит о том, что доставка дейтаграммы должна быть максимально надежной.

Естественно, что в объединенной сети не всегда возможно обеспечить запрашиваемый дейтаграммой тип транспортировки (например, может оказаться, что нельзя проложить к получателю маршрут, удовлетворяющий некоторым требованиям). Поэтому указанные биты следует рассматривать только как рекомендации при выполнении алгоритма маршрутизации, а не как обязательное требование.

Если существует несколько маршрутов следования пакетов к получателю, то при их выборе маршрутизатор может руководствоваться описанными битами, чтобы обеспечить требуемый тип транспортировки пакетов. Например, предположим, что маршрутизатор может отправлять пакеты получателю с малым временем задержки, но по каналу с низкой пропускной способностью (по наземной выделенной линии связи) и с большой задержкой — по высокоскоростному спутниковому каналу связи. Тогда в дейтаграммах, транспортирующих данные, поступающие на удаленный компьютер от клавиатуры пользователя, следует установить бит D. В результате они будут доставлены получателю с минимальной задержкой. А для дейтаграмм, транспортирующих большой объем данных из файла, следует установить бит T. В результате они будут передаваться по высокоскоростному спутниковому каналу связи. При этом время задержки передачи пакетов не имеет существенного значения.

В конце 1990-х годов инженерной группой IETF была изменена первоначальная раскладка 8-битового поля типа обслуживания, как показано на рис. 10.5. Это позволило реализовать различные схемы *дифференцированного обслуживания (differentiated services, DS)*.

Рис. 10.5. Интерпретация поля типа обслуживания IP-дейтаграммы для реализации различных схем дифференцированного обслуживания (DS)

В новой интерпретации в первых шести битах хранится *код указателя (codepoint)* службы, который иногда обозначается аббревиатурой DSCP (*differentiated services codepoint*), а последние два бита не используются. С помощью кода указателя программа протокола может осуществить поиск нужной службы, как правило через массив указателей. И хотя в поле размером 6 бит можно закодировать информацию о 64 отдельных службах, разработчики протокола, сочли нужным ограничить это число, поскольку на реально работающих маршрутизаторах задействовано всего несколько типов служб. В результате разные коды указателя могут соответствовать одной и той же службе. Более того, для совместимости с первоначальной раскладкой рассматриваемого нами поля в стандарте предусмотрены специальные различия для трех младших (раньше в них помещалось значение приоритета) и трех старших битов кода указателя. Если в трех старших битах кода содержатся нули, то три младших бита (приоритета) определяют восемь больших классов служб, соответствующих тем же основополагающим принципам, которым отвечала первоначальная раскладка 8-битового поля типа обслуживания. Речь идет о приоритете дейтаграмм, у которых в трех младших битах поля указателя содержится большее значение. Таким образом, для определения восьми упорядоченных классов используются коды указателя в формате ххх000, где х может быть как нулем, так и единицей.

В новой раскладке поля типа обслуживания учтен также другой часто используемый случай, когда для маршрутизации трафика используется значение приоритета 6 или 7. В стандарте отдельно оговорен метод обработки этих значений приоритета. Таким образом, в маршрутизаторе должно быть предусмотрено по меньшей мере два алгоритма приоритетного обслуживания: один для обычного трафика, а другой — для трафика с высоким приоритетом. Если три старших бита поля кода указателя равны нулю, маршрутизатор должен так настроить свои таблицы указателей, чтобы кодам 6 и 7 соответствовали классы служб с наивысшим приоритетом, а все остальным кодам — классы служб с более низким приоритетом. Таким образом, если будет получена дейтаграмма, созданная с помощью старой версии сетевого программного обеспечения (с первоначальной раскладкой поля типа обслуживания), маршрутизатор, используя систему дифференцированных классов служб, обработает значения приоритета 6 и 7 так, как и было задумано отправителем.

Все возможные значения кодов указателя (их может быть 64) разбиты на 3 административных группы, как показано в табл. 10 .1.

Таблица 10.1. Три административных группы значений кодов указателя

| **Группа** | **Код указателя\*** | **Описание** |
| --- | --- | --- |
| 1 | ххххх0 | Стандартизовано IETF |
| 2 | xxxx11 | Для локального или экспериментального использования |
| 3 | xxxx01 | В настоящее время доступны для локального или экспериментального использования |

**Примечание.** Здесь используется принятый в сети порядок битов “слева направо”, т.е. младший бит находится слева.

Как следует из табл. 10.1, процесс интерпретации половины значений кодов указателя (т.е. 32, относящихся к первой группе) стандартизован инженерной группой IETF. В настоящее время все значения, относящиеся к группам 2 и 3, пока доступны для локального или экспериментального использования. Однако следует отметить, что как только значения кодов из первой группы, назначаемые центральным органом, будут исчерпаны, выделение кодов может быть продолжено уже из третьей группы.

Таким образом, для определения номера группы необходимо проанализировать два старших бита кода указателя. Описанный принцип разбиения на группы был выбран для того, чтобы величины кодов указателя, соответствующие шаблону ххх000, находились в одной группе.

Независимо от того, какая раскладка поля типа обслуживания используется (оригинальная TOS или новая — с классами дифференцированного обслуживания) важно понимать, что алгоритмы маршрутизации могут выбирать способ доставки дейтаграмм только в том случае, если к получателю существует несколько маршрутов с разными характеристиками, которые в свою очередь зависят от применяемого сетевого оборудования и установленных локальных правилах. Таким образом, запрошенный в дейтаграмме уровень обслуживания еще не означает, что все маршрутизаторы, расположенные по пути следования пакета смогут его удовлетворить. Подводя итоги, можно сказать следующее.

Спецификацию типа обслуживания дейтаграммы, следует рассматривать только как рекомендацию при выполнении алгоритма маршрутизации, а не как обязательное требование. В случае, если к конечному получателю существует несколько маршрутов с разными характеристиками, спецификация позволяет выбрать тот из них, который наиболее полно удовлетворяет выдвинутым требованиям. Следует иметь в виду, что эти требования могут вступать в противоречие с применяемым сетевым оборудованием и установленными локальными правилами. В объединенной сети невозможно гарантировать выполнение какого бы то ни было обслуживания заданного типа.

**Инкапсуляция дейтаграмм**

Прежде чем перейти к рассмотрению следующих полей дейтаграммы, необходимо уяснить, как связаны между собой дейтаграммы и сетевые фреймы. Сразу же возникает вопрос: каков максимальный размер дейтаграммы? В отличие от фреймов физической сети, формат которых зависит от используемого сетевого оборудования, обработка дейтаграмм выполняется исключительно программными средствами. Поэтому, теоретически, их длина может быть любой, точнее такой, какой ее выбрали разработчики протокола. Выше мы уже говорили, что в дейтаграмме стандарта IPv4 под поле, содержащее размер дейтаграммы, было выделено 16 бит. Таким образом, максимальная длина дейтаграммы не может превышать 65 535 октетов.

Однако на практике на размер дейтаграмм налагаются более существенные ограничения. Вы уже знаете, что передача дейтаграмм от одного компьютера к другому всегда выполняется средствами физической сети. Поэтому, для повышения эффективности работы механизма транспортировки объединенной сети необходимо иметь гарантию, что каждая дейтаграмма поместится в один физический фрейм. Другими словами, нужно по возможности сделать так, чтобы существовала прямая связь между принятым абстрактным сетевым пакетом (дейтаграммой) и фреймом физической сети.

Метод транспортировки одной дейтаграммы в одной сетевом фрейме называется *инкапсуляцией (encapsulation)*. С точки зрения физической сети, дейтаграмма ничем не отличается от любого другого сообщения, посылаемого от одной машины до другой. Для сетевого оборудования формат дейтаграммы не имеет никакого значения, точно так же, как и указанный в ней IP-адрес получателя. Таким образом, как показано на рис. 10.6, при передаче IP-дейтаграммы от одного компьютера до другого, она целиком помещается в область данных сетевого фрейма1.

Рис. 10.6. Инкапсуляция IP-дейтаграммы в сетевом фрейме. С точки зрения физической сети, дейтаграмма вместе со своим заголовком, рассматривается как обычные данные

1. Как правило, в заголовке сетевого фрейма предусмотрено специальное поле, предназначенное для идентификации находящихся в нем данных. Например, в сети Ethernet для идентификации инкапсулированной IP-дейтаграммы, помещенной в область данных сетевого фрейма, в поле типа фрейма записывается значение 0x0800.

**Размеры дейтаграммы, MTU и процесс фрагментации**

В идеальном случае IP-дейтаграмма должна целиком помещаться в одном сетевом фрейме, что позволяет передать ее по сети с наибольшей эффективностью. Поэтому для достижения подобной эффективности разработчики протокола IP должны были выбрать максимальный размер дейтаграммы, который бы позволял всегда помещать ее в один фрейм. Однако здесь возникает вопрос: каким должен быть этот размер? Кроме того, не следует забывать, что в процессе передачи от отправителя к конечному получателю, дейтаграмма может проходить по физическим сетям разного типа.

Чтобы до конца понять суть проблемы, вспомним, как работает сетевое оборудование. В любой сетевой технологии с коммутацией пакетов существует такое понятие, как максимальный размер данных, которые могут быть переданы в одном физическом фрейме. Например, в технологии Ethernet это значение равняется 1500 октетам, а в технологии FDDI в одном фрейме может находиться примерно 4470 октетов данных. Для обозначения подобных ограничений используют специальный термин — *максимальная единица передачи данных в сети*, или *MTU (maximum transfer unit)*. Величина MTU может быть достаточно маленькой. В некоторых сетях значение MTU составляет порядка 128 октетов и даже меньше.

Таким образом, ограничение размеров дейтаграммы размерами минимального MTU, существующего в объединенной сети, приводит к падению эффективности передачи данных на тех участках, где сетевое оборудование позволяет передавать фреймы большего размера. В то же время, если размер дейтаграммы будет больше, чем размер минимального MTU в объединенной сети, то это означает, что на некоторых участках сети она может не поместиться в один сетевой фрейм.

Решение проблемы должно быть уже очевидным. Вспомним, что ключевым моментом при создании объединенной сети было удобство работы с ней. Поэтому разработчики сделали так, чтобы особенности используемых сетевых технологий были скрыты от пользователя. Они не стали создавать какой-то особый формат дейтаграммы, который бы удовлетворял всем существующим ограничениям для физических сетей. Вместо этого был выбран исходный размер дейтаграммы, удобный с точки зрения программного обеспечения протокола TCP/IP, а также оговорен способ разделения больших дейтаграмм на части в том случае, если дейтаграмма должна передаваться по физической сети с малым значением MTU. Части, на которые разделяется дейтаграмма, называются *фрагментами*, а сам процесс разделения дейтаграммы — *фрагментацией*.

Как показано на рис. 10.7, фрагментация обычно выполняется на одном из промежуточных маршрутизаторов, расположенных по пути следования дейтаграммы от ее отправителя до конечного получателя. В данном случае маршрутизатор получил дейтаграмму из сети с большим значением MTU и должен ее передать по сети, где значение MTU меньше, чем размер дейтаграммы.

Рис. 10.7. Пример, иллюстрирующий процесс выполнения фрагментации в сети. Маршрутизатор R1 выполняет фрагментацию больших дейтаграмм, посланных от узла А до узла В; маршрутизатор R2 выполняет фрагментацию больших дейтаграмм, посланных от узла В до узла А

На рис. 10.7 оба узла подключены к сети Ethernet, в которой размер MTU составляет 1500 октетов1. Это значит, что оба узла могут отправлять дейтаграммы, максимальный размер которых составляет 1500 октетов. Однако по пути от узла А до узла В пакеты попадают в промежуточную сеть, в которой размер MTU составляет 620 октетов. Если узел А посылает узлу В дейтаграмму, размер которой больше 620 октетов, маршрутизатор R1, должен выполнить фрагментацию. Точно так же, если узел В посылает большую дейтаграмму узлу А, фрагментацию дейтаграмм выполняет маршрутизатор R2.

Размер фрагмента выбирается так, чтобы он целиком помещался в одном сетевом фрейме. Кроме того, так как в протоколе IP величина смещения фрагмента от начала заголовка кратна 8 октетам, размер фрагмента также выбирается кратным 8 октетам. Само собой разумеется, что выбор фрагмента, размер которого приблизительно равен MTU участка сети и при этом кратен 8 октетам, приводит к делению дейтаграмм на неравные части. Обычно размер последней части намного меньше остальных. Перед тем, как получатель сможет обработать фрагментированную дейтаграмму, он должен получить все ее части и произвести *сборку* (т.е. восстановить ее первоначальный вид).

В протоколе IP нет каких-либо рекомендаций по поводу минимального размера дейтаграммы. Кроме того, нет никакой гарантии, что большая дейтаграмма будет доставлена получателю без фрагментации. Таким образом, получатель может выбирать размер дейтаграммы по своему усмотрению; ее фрагментация и последующая сборка выполняются автоматически на уровне протокола IP, без какого-либо вмешательства отправляющей стороны. В спецификации протокола IP указывается, что маршрутизаторы должны принимать дейтаграммы, размер которых не превосходит MTU сетей, к которым они подключены. Кроме того, маршрутизаторы должны всегда обрабатывать дейтаграммы размером до 576 октетов. От узлов сети также требуется, чтобы они принимали и при необходимости восстанавливали дейтаграммы, размером как минимум 576 октетов.

В процессе фрагментации дейтаграмма делится на части. Вас может удивить, что каждый фрагмент имеет такой же формат, как и исходная дейтаграмма. Результат фрагментации показан на рис. 10.8.

Рис. 10.8. Схематическое изображение исходной дейтаграммы, содержащей 1400 октетов данных (а); та же дейтаграмма, разделенная на три фрагмента данных для сети с MTU, равным 620. В заголовках фрагментов 1 и 2 устанавливается специальный бит (more fragments), означающий, что существует следующий по порядку фрагмент. Смещения фрагментов выражены в октетах в десятичной системе счисления. Для определения значения, помещаемого в заголовок фрагмента, значение смещения нужно разделить на 8

Как видно из рис. 10.8, у каждого фрагмента существует свой заголовок, информация в котором дублирует информацию из заголовка исходной дейтаграммы (кроме одного бита в поле флагов, обозначающего фрагмент). За заголовком фрагмента следует поле данных. Его размер вместе с заголовком не должен превышать размера MTU той сети, по которой пересылается фрагмент.

Ограничение в 1500 октетов взято из стандарта Ethernet. При использовании стандарта IEEE 802.3 и заголовка SNAP размер поля данных ограничивается 1492 октетами. При использовании сетевого оборудования Ethernet, изготовленного некоторыми независимыми разработчиками, размер области данных может быть несколько большим.

**Сборка фрагментов**

Далее будет рассмотрен интересный вопрос, касающийся сборки дейтаграмм. Нужно ли собирать фрагменты дейтаграмм сразу после прохождения критичного участка сети или передать их по назначению и поручить сборку получателю?

В объединенной сети TCP/IP после фрагментации дейтаграмм каждый фрагмент передается конечному получателю как самостоятельная дейтаграмма. Сборкой фрагментов занимается конечный получатель. Однако такой подход имеет два недостатка.

Во-первых, поскольку дейтаграммы не собираются сразу после прохождения участка сети с малым размером MTU, конечному получателю могут отравляться фрагменты небольшого размера. Таким образом, сборка дейтаграмм конечным получателем приводит к снижению эффективности функционирования объединенной сети в случае, если после точки фрагментации пакеты будут проходить по сетям с большим значением MTU.

Во-вторых, если хотя бы один из фрагментов потеряется, конечному получателю не удастся собрать дейтаграмму. После доставки получателю первого фрагмента дейтаграммы запускается специальный *таймер сборки (reassembly timer)*. Если значение таймера истекает до того, как получены все фрагменты дейтаграммы, получатель не обрабатывает дейтаграмму и удаляет полученные фрагменты. То есть фрагментация существенно повышает вероятность потери целой дейтаграммы, поскольку потеря любого фрагмента автоматически означает потерю всей дейтаграммы.

Однако, несмотря на описанные недостатки, процесс сборки дейтаграмм получателем работает очень хорошо. Кроме того, такой подход позволяет выполнять маршрутизацию каждого фрагмента независимо от остальных фрагментов дейтаграммы. При этом промежуточные маршрутизаторы не должны сохранять фрагменты дейтаграммы с целью их последующей сборки.

**Управление фрагментацией**

В заголовке дейтаграммы предусмотрены три поля, управляющие процессом ее фрагментации и последующей сборки: идентификационные данные, флажки и смещение фрагмента. В поле идентификационных данных указывается уникальное целое число, предназначенное для идентификации текущей дейтаграммы. Напомним, что при фрагментации маршрутизатор копирует информацию из большинства полей заголовка дейтаграммы в аналогичные поля заголовков фрагментов. Поэтому поле идентификационных данных также будет скопировано. Оно служит для того, чтобы получатель смог установить, какой из дейтаграмм принадлежат поступившие фрагменты. При получении очередного фрагмента получатель пытается идентифицировать дейтаграмму, используя предназначенное для этой цели поле и адрес отправителя. При отправке IP-дейтаграммы компьютер должен занести в поле идентификационных данных ее заголовка уникальное число1. Один из методов получения уникального идентификатора, который используется программами протокола IP, заключается в организации в оперативной памяти глобального счетчика отправляемых пакетов.

При создании каждой новой дейтаграммы счетчик увеличивается на единицу, и полученное значение помещается в поле идентификационных данных.

Напомним, что формат каждого из фрагментов практически совпадает с форматом целой дейтаграммы. В заголовке каждого фрагмента в соответствующем поле указывается смещение, по которому расположены переносимые им данные, относительно исходной дейтаграммы. Для первого фрагмента смещение будет равно нулю. С целью экономии места в заголовке фрагмента, смещение выражается не в октетах, а в относительных единицах. Каждая единица равна 8 октетам. Перед сборкой дейтаграммы, компьютер должен получить по сети все ее фрагменты и упорядочить их по значению смещения, начиная с нулевого и заканчивая самым максимальным. При этом фрагменты не обязательно должны приходить в том порядке, как это необходимо для сборки. Не забывайте, что маршрутизатор, выполнивший фрагментацию дейтаграммы, и ее получатель, собирающий фрагменты, не обмениваются между собой никакой информацией по этому поводу.

Два младших бита поля флажков (его размер составляет 3 бита) предназначены для управления процессом фрагментации. Обычно, процесс фрагментации не является прерогативой прикладного программного обеспечения, использующего протокол TCP/IP. Все дело в том, что фрагментация и последующая сборка дейтаграмм выполняются автоматически протоколами нижнего уровня, включенными в операционную систему, и поэтому скрыты от конечного пользователя. Тем не менее при тестировании сетевого программного обеспечения либо при устранении возникших проблем, может понадобиться определить размер фрагментируемой дейтаграммы. Первый (по номеру, а не по порядку) управляющий бит как раз и предназначен для целей отладки и указывает, можно ли фрагментировать дейтаграмму. Установка этого бита в единицу говорит маршрутизатору о том, что фрагментацию данной дейтаграммы выполнять не нужно.

Прикладные программы могут отменить режим фрагментации только в том случае, если для нормального функционирования какой-либо системы требуется дейтаграмма целиком. В качестве примера рассмотрим процедуру начальной загрузки небольшой встроенной системы, программа управления которой зашита в ПЗУ. Система отправляет по объединенной сети запрос серверу, который в ответ присылает образ памяти, содержащий программу начальной загрузки. Обычно из-за ограниченного объема ПЗУ программы управления встроенными системами поддерживают только элементарные сетевые операции. Поэтому они не умеют собирать воедино фрагментированные дейтаграммы. Следовательно, отправляя ответ, сервер должен установить бит *запрета фрагментации*. Если, получив такую дейтаграмму, маршрутизатор не сможет передать ее дальше без фрагментации, он посылает отправителю сообщение об ошибке и удаляет дейтаграмму из очереди на отправку.

Младший (нулевой) бит поля флажков предназначен для идентификации фрагментов дейтаграммы, т.е. указывает, является ли этот фрагмент последним или за ним следуют еще фрагменты. Этот бит называется More fragments (дополнительные фрагменты).

Чтобы продемонстрировать, зачем нужен этот бит, давайте рассмотрим работу программного обеспечения протокола IP на компьютере конечного получателя, которое пытается собрать фрагментированную дейтаграмму. Фрагменты дейтаграммы доставляются конечному получателю по очереди, причем необязательно по порядку. Поэтому получателю нужно определить момент, когда ему будут доставлены все фрагменты. Вспомним, что в заголовке дейтаграммы предусмотрено поле, в котором указывается ее общая длина. Однако, при получении фрагмента, в данном поле будет находиться не общая длина дейтаграммы, а только длина текущего фрагмента. Поэтому, используя только описываемое поле, конечный получатель не сможет определить момент, когда ему будут доставлены все фрагменты дейтаграммы. Для решения проблемы и предназначен бит, указывающий на наличие последующих фрагментов. Получив фрагмент, у которого этот бит сброшен, получатель знает, что ему доставлен последний фрагмент дейтаграммы. Просуммировав значения, находящиеся в его поле смещения и поле общей длины, получатель легко сможет вычислить длину исходной дейтаграммы. Проанализировав аналогичные поля всех доставленных фрагментов, получатель сможет определить, доставлены ли все необходимые для сборки дейтаграммы фрагменты.

1. Теоретически, при повторной передаче пакета в поле его идентификационных данных может быть помещено такое же значение, что и у оригинала. Однако на практике, выполнением повторной передачи пакета занимаются протоколы более высокого уровня. В результате каждый повторный пакет будет размещаться в новой дейтаграмме, содержащей в заголовке уникальное число.

**Время жизни дейтаграммы (TTL)**

В принципе, в поле *времени жизни* заголовка дейтаграммы должен указываться интервал времени в секундах, в течение которого дейтаграмме разрешено находиться в объединенной сети. Эта идея проста и в то же время очень важна. Суть ее состоит в том, что, запуская дейтаграмму в объединенную сеть, компьютер устанавливает для нее максимально допустимый интервал времени, в течение которого дейтаграмма должна достичь получателя. Проходя по объединенной сети от одного узла к другому, дейтаграмма последовательно подвергается обработке разными маршрутизаторами или обычным компьютерами. После выполнения обработки каждый компьютер уменьшает значение времени жизни дейтаграммы и передает ее на обработку другому узлу. Если при этом окажется, что время жизни исчерпано, обрабатывающий дейтаграмму узел должен удалить ее из объединенной сети.

Проблема заключается в том, что маршрутизатор не может точно узнать время, прошедшее с момента запуска дейтаграммы в сеть, поскольку нельзя определить время передачи дейтаграммы по разным физическим сетям без синхронизации часов на всех маршрутизаторах объединенной сети. Последнее требование выполнить довольно сложно, учитывая масштабы объединенной сети. Поэтому был принят ряд правил, упрощающих обработку дейтаграмм и не требующих синхронизации часов.

Во-первых, от каждого маршрутизатора, находящегося по пути следования дейтаграммы от отправителя до конечного получателя, требуется, чтобы после ее обработки он уменьшил на единицу значение поля времени жизни.

Во-вторых, в случае перегрузки маршрутизатора дейтаграмма может обрабатываться довольно продолжительное время. Поэтому после получения дейтаграммы каждый маршрутизатор должен зафиксировать локальное время ее прибытия, а после обработки дейтаграммы уменьшить значение поля времени жизни на число секунд, которые дейтаграмма провела в маршрутизаторе в ожидании обслуживания1.

Как только значение поля времени жизни дейтаграммы станет равно нулю, маршрутизатор должен удалить дейтаграмму из сети, а ее отправителю послать соответствующее сообщение об ошибке. Идея установки времени жизни дейтаграммы интересна еще и тем, что она не позволяет дейтаграмме бесконечно блуждать по сети от одного узла до другого, например, в случае повреждения таблиц маршрутизации или неправильной их настройки, когда маршрутизаторы начинают пересылать дейтаграммы друг другу или по кругу.

Хотя последнее замечание относительно большого времени обработки дейтаграммы маршрутизатором крайне важно, следует отметить, что оно уже не актуально. Дело в том, что разработчики современных сетевых технологий добились того, чтобы каждая дейтаграмма проходила узлы маршрутизации за приемлемое время. Если же по какой-либо причине задержка при обработке дейтаграммы становится чрезмерной, маршрутизатор попросту удаляет такую дейтаграмму.

Поэтому на практике, значение поля времени жизни можно рассматривать скорее как максимальное число узлов маршрутизации, через которые должна пройти дейтаграмма, а не как время в секундах, за которое дейтаграмма должна достичь получателя. Не забывайте, что каждый маршрутизатор уменьшает его значение на единицу.

1. На практике современные маршрутизаторы никогда не обрабатывают дейтаграммы по несколько секунд.

**Остальные поля заголовка дейтаграммы**

Поле *типа протокола* является аналогом поля типа сетевого фрейма. По указанному в нем значению можно узнать, к какому типу протокола высокого уровня относится сообщение, помещенное в область данных дейтаграммы. По существу значение поля типа протокола определяет формат области данных дейтаграммы. Соответствие между протоколом высокого уровня и значением, помещенным в поле типа протокола, должно устанавливаться на уровне центрального органа объединенной сети. Только в этом случае можно гарантировать адекватность обслуживания дейтаграммы по всей сети.

Поле *контрольной суммы*, заголовка предназначено для проверки целостности полей заголовка дейтаграммы. При подсчете контрольной суммы в протоколе IP заголовок дейтаграммы рассматривается как состоящий из последовательности 16-разрядных целых чисел (с соблюдением сетевого порядка следования байтов). При сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Напомним, что для представления отрицательного числа в инверсном коде следует инвертировать все значения его битов. Полученный результат инвертируется (т.е. его знак меняется на противоположный, чтобы значение контрольной суммы было положительным). При вычислении контрольной суммы всех полей заголовка дейтаграммы, значение поля контрольной суммы полагается равным нулю.

Важно отметить, что контрольная сумма вычисляется только для заголовка IP-дейтаграммы. При этом ее область данных не учитывается. Такой подход имеет как преимущества, так и недостатки. К преимуществам можно отнести скорость вычисления контрольной суммы маршрутизаторами, поскольку длина заголовка составляет всего несколько десятков октетов. Кроме того, это позволяет протоколам более высокого уровня использовать собственные методы вычисления контрольных сумм для инкапсулированных данных. Основной недостаток такого подхода заключается в том, что он заставляет другие протоколы более высокого уровня вычислять собственные значения контрольных сумм, иначе снижается вероятность обнаружения поврежденной дейтаграммы.

Как следует из названия, в полях IP-адреса отправителя и получателя указываются 32-разрядные адреса отправителя дейтаграммы и ее *конечного* получателя. Хотя при выполнении маршрутизации дейтаграмма может проходить через множество промежуточных узлов, значение полей адресов отправителя и получателя никогда не изменяются. В них всегда содержится исходный адрес отправителя и адрес конечного получателя1.

В поле области данных (см. рис. 10.9), помещается переносимое дейтаграммой сообщение. Его длина, конечно же, зависит от типа пересылаемых в дейтаграмме данных. Поле параметров протокола IP имеет переменную длину. Длина поля выравнивания зависит от того, какие параметры протокола IP установлены. Оно содержит нулевые биты, количество которых выбирается так, чтобы длина заголовка дейтаграммы была кратна 32 битам. (Напомним, что в поле длины заголовка дейтаграммы помещается значение, выраженное в 32-х разрядных словах.)

Рис. 10.9. Формат IP-дейтаграммы — основной единицы передачи данных в объединенной сети на основе протокола TCP/IP

1. За исключением случаев, когда в дейтаграмме содержится параметр, определяющий маршрутизацию от источника, о котором речь пойдет ниже.

**Параметры IP-дейтаграммы**

За полем IP-адреса получателя следует поле параметров протокола IP, однако оно присутствует далеко не в каждой дейтаграмме. В основном параметры протокола IP помещаются в дейтаграммы только в целях тестирования и отладки. Тем не менее обработка параметров является составляющей частью протокола IP, поэтому она должна быть включена во все его стандартные реализации.

Длина рассматриваемого поля зависит от выбранных параметров. Для некоторых параметров длина этого поля составляет один октет. В него помещается так называемый *код параметра (option code)*. Другие параметры имеют переменную длину. При включении в дейтаграмму параметров, они следуют один за другим без специальных разделителей. Каждый параметр состоит из одного октета кода, за которым может следовать один октет, в котором указывается длина расположенных следом данных. Октет кода параметра состоит из трех полей (рис. 10.9).

Рис. 10.9. Октет кода параметра поделен на три поля, длина которых составляет 1, 2 и 5 битов

В первом поле, размер которого составляет 1 бит, находится флаг копирования. Во втором поле (его размер составляет 2 бита) указывается класс параметра, а в третьем поле размером 5 битов — номер параметра. Флаг копирования предназначен для управления фрагментацией дейтаграммы, выполняемой маршрутизаторами. Когда его значение равно 1, маршрутизаторы переносят поле параметров во все фрагменты дейтаграммы. Если же значение флага равно нулю, поле параметров переносится не во все, а только в первый фрагмент дейтаграммы.

В следующих двух полях (*класс параметра* и *номер параметра*) указываются общий класс параметров и конкретный параметр в заданном классе. Распределение классов параметров протокола IP приведено в табл. 10.2.

Таблица 10.2. Распределение классов параметров протокола IP

| **Класс параметра** | **Описание** |
| --- | --- |
| 0 | Управление дейтаграммой или сетью |
| 1 | Зарезервировано для дальнейшего использования |
| 2 | Для отладочных целей и измерения параметров сети |
| 3 | Зарезервировано для дальнейшего использования |

В табл. 10.3 приведены значения классов и номера параметров, которые могут встречаться в IP-дейтаграмме. Как видно, большинство параметров используются для управляющих и служебных целей.

Таблица 10.3. Значения классов и номера параметров IP-дейтаграммы

| **Класс параметра** | **Номер параметра** | **Длина** | **Описание** |
| --- | --- | --- | --- |
| 0 | 0 | – | Конец списка параметров. Используется в случае, если конец списка параметров не совпадает с концом заголовка дейтаграммы (см. описание поля выравнивания IP-дейтаграммы) |
| 0 | 1 | – | Холостая операция. Используется для выравнивания октетов в списке параметров |
| 0 | 2 | 11 | Управление системами безопасности и ограничений (используется в приложениях, применяемых в военных целях) |
| 0 | 3 | Переменная | Нестрогая маршрутизация от источника. Используется для запроса на выполнение маршрутизации через указанные узлы |
| 0 | 7 | Переменная | Регистрация маршрута следования. Используется для трассировки маршрута |
| 0 | 8 | 4 | Идентификатор потока. Используется для передачи идентификатора потока SATNET (устарело) |
| 0 | 9 | Переменная | Строгая маршрутизация от источника. Используется для указания точного маршрута дейтаграммы |
| 0 | 11 | 4 | Запрос на MTU. Используется для определения значения MTU в сетях, по которым проходит маршрут пакета |
| 0 | 12 | 4 | Ответ MTU. Используется для определения значения MTU в сетях, по которым проходит маршрут пакета |
| 0 | 20 | 4 | Сообщение, привлекающее внимание маршрутизатора. Маршрутизатор должен проанализировать данную дейтаграмму даже в том случае, если он не является ее получателем |
| 2 | 4 | Переменная | Межсетевые метки времени. Используется для регистрации временных меток на пути следования дейтаграммы |
| 2 | 18 | Переменная | Трассировка маршрута. Используется программой traceroute для поиска маршрутизаторов на пути следования дейтаграммы |

**Параметры регистрации маршрута следования**

Одними из самых интересных являются параметры маршрутизации и регистрации временных меток, поскольку они позволяют проконтролировать правильность передачи дейтаграмм в объединенной сети. Параметр *регистрации маршрута* позволяет отправителю создать пустой список IP-адресов и “попросить” каждый маршрутизатор, обрабатывающий дейтаграмму, поместить свой IP-адрес в этот список. Формат параметра регистрации маршрута приведен на рис. 10.10.

Рис. 10.10. Формат параметра регистрации маршрута IP-дейтаграммы. В начале поля параметров располагается три служебных октета, сразу за которыми помещается список адресов. Несмотря на то, что на рисунке формат параметра изображен в виде 32-разрядных блоков, на самом деле в заголовке дейтаграммы они не выравниваются на границу 32-х бит

В октет кода параметра, который был описан выше, помещаются значения класса и номера параметра (для регистрации маршрута следования они, соответственно, равны 0 и 7). В поле длины, следующее за октетом кода, помещается значение общей длины поля параметра, которое он занимает в заголовке дейтаграммы (с учетом первых трех октетов). Сразу за тремя служебными октетами располагается область, зарезервированная для записи IP-адресов. Первый IP-адрес помещается в начало этой области (рис. 10.10), следом за ним — второй и т.д. В поле указателя помещается смещение следующего свободного элемента для записи IP-адреса, выраженное в октетах относительно начала текущего параметра.

Получив дейтаграмму, в заголовке которой задан параметр регистрации маршрута следования, компьютер должен добавить свой адрес в конец списка IP-адресов узлов маршрутизации. При этом отправитель подобной дейтаграммы должен предусмотреть в поле параметра регистрации достаточно места для размещения всех элементов списка. Перед добавлением нового элемента в список компьютер должен сравнить значения полей указателя и длины. Если величина указателя превосходит значение поля длины, значит, в области, выделенной под размещение списка IP-адресов, больше нет свободного места. В таком случае компьютер пересылает дейтаграмму дальше без помещения в ее заголовок своего IP-адреса. Если же в заголовке дейтаграммы есть место, компьютер помещает в него свой 4-октетный IP-адрес. Место расположения нового элемента списка определяется значением поля указателя. После добавления нового элемента в список значение указателя увеличивается на 4.

После того как дейтаграмма будет доставлена конечному получателю, содержимое ее списка адресов узлов маршрутизации может быть легко извлечено из заголовка и обработано. Обычно после получения дейтаграммы этот список игнорируется. Поэтому использование параметра регистрации маршрута должно приводить к взаимодействию двух компьютеров — отправителя и конечного получателя. Причина состоит в том, что список узлов маршрутизации не посылается автоматически отправителю дейтаграммы, в которой установлен параметр регистрации маршрута следования. Поэтому, прежде чем посылать такую дейтаграмму, отправитель должен быть уверен, что ее получатель обработает результирующий список узлов маршрутизации и вернет его обратно.

**Параметры маршрутизации от источника**

Еще одной интересной идеей, реализованной разработчиками в протоколе IP, является возможность задания параметров *исходного маршрута (source route)* дейтаграммы, или маршрутизации от источника. Идея заключается в том, что отправитель дейтаграммы может указать в ее заголовке путь, по которому она должна пройти в объединенной сети. Например, эта возможность часто используется сетевыми администраторами для определения пропускной способности некоторой физической сети в случае, если при нормальных условиях маршрутизаторы не пересылают через нее дейтаграммы. Возможность проведения подобных тестов на реально действующем оборудовании особенно важна в производственных условиях. При этом системный администратор может быть уверен, что при тестировании одного участка сети дейтаграммы пользователей будут пересылаться по другому, уже протестированному участку. Естественно, что установка параметров маршрута дейтаграммы может быть интересна только тем сетевым администраторам, которые хорошо представляют топологию сети. Для обычного пользователя описанная в этом разделе возможность не представляет никакой ценности.

В протоколе IP предусмотрена поддержка двух форм маршрутизации от источника. Одна из них называется строгой маршрутизацией (strict source routing). Суть ее заключается в том, что в заголовке дейтаграммы перечисляются IP-адреса узлов сети, через которые дейтаграмма должна обязательно пройти. Формат описываемого параметра показан на рис. 10.11.

Рис. 10.11. В параметре строгой маршрутизации указывается список IP-адресов узлов, через которые дейтаграмма должна обязательно пройти

Строгая маршрутизация от источника означает, что в заголовке дейтаграммы указан точный путь, по которому она должна пройти до конечного получателя. Следует учитывать, что маршрут между двумя соседними IP-адресами, указанными в списке, должен проходить по одной физической сети. Если маршрутизатор не сможет отправить дейтаграмму по указанному маршруту, отправителю посылается сообщение об ошибке.

Существует и другая форма маршрутизации от источника, называемая *нестрогой (loose source routing)*. Суть ее состоит в том, что в заголовке дейтаграммы также указывается список адресов узлов, через которые должна пройти дейтаграмма. Только, в отличие от строгой маршрутизации, путь между соседними адресами, указанными в списке, не обязательно должен проходить по одной физической сети. То есть между двумя адресами узлов маршрутизации, указанными в списке, может находиться любое количество других маршрутизаторов.

Согласно протоколу, от устройств, выполняющих как строгую, так и нестрогую маршрутизацию, требуется, чтобы они заменяли IP-адреса, указанные в списке узлов маршрутизации дейтаграммы своими локальными сетевыми адресами. Таким образом, после доставки дейтаграммы получателю в ее заголовке будет содержаться список адресов узлов объединенной сети (так же, как и при использовании параметра регистрации маршрута), через которую она прошла.

Формат параметров маршрутизации от источника практически повторяет формат параметра, использовавшегося для регистрации маршрута пакетов, о котором речь шла выше. Перед обработкой очередного элемента списка IP-адресов каждый маршрутизатор должен проанализировать значения полей указателя и длины и убедиться, что список элементов еще не исчерпан. Если окажется, что значение указателя больше значения поля длины, маршрутизатор направляет дейтаграмму к получателю обычным образом. Если же список не исчерпан, маршрутизатор, опираясь на значение указателя, выбирает очередной элемент с IP-адресом, заменяет его своим адресом и отправляет дейтаграмму по адресу узла1, взятому из списка.

1. Не забывайте, что каждому сетевому интерфейсу маршрутизатора назначается отдельный адрес. Поэтому в список заносится адрес, соответствующий сети, по которой передается дейтаграмма.

**Параметр регистрации временных меток**

Параметр *регистрации временных меток* во многом напоминает параметр регистрации маршрута следования дейтаграммы, о котором говорилось выше. Отправитель создает пустой список определенного размера, элементы в который добавляются промежуточными маршрутизаторами, расположенными по пути следования дейтаграммы. Каждый элемент в списке состоит из двух 32-разрядных слов: IP-адреса маршрутизатора, создавшего элемент, и целого числа, содержащего временную метку (рис. 10.12).

Рис. 10.12. Формат параметра регистрации временных меток. Точный формат этого параметра и правила, по которым он будет обрабатываться маршрутизаторами, зависят от состояния битов флажков

Как показано на рис. 10.12, в поле длины указывается размер области параметра, зарезервированной для размещения элементов списка, а в поле указателя хранится смещение следующего свободного элемента (по аналогии с параметром регистрации маршрута, рассмотренным выше). В 4-битовом поле переполнения хранится целочисленный счетчик, указывающий на количество маршрутизаторов, которые не смогли занести в список свою временную метку из-за того, что размер выделенной области параметра оказался слишком мал. Значение 4-битового поля флажков определяет точный формат описываемого параметра и тип временной метки, которую должны поместить в список маршрутизаторы (табл. 10.4).

Таблица 10.4. Значения поля флажков параметра регистрации временных меток

| **Значение** | **Описание** |
| --- | --- |
| 0 | Регистрировать только временные метки без записи IP-адресов |
| 1 | Помещать перед каждой временной меткой IP-адрес маршрутизатора. Формат этого параметра приведен на рис. 7.12 |
| 3 | IP-адреса указаны отправителем. Маршрутизатор должен зарегистрировать временную метку только в том случае, если следующий IP-адрес в списке соответствует IP-адресу маршрутизатора |

Временные метки представляют собой дату и время обработки дейтаграммы маршрутизатором, выраженные количеством миллисекунд, прошедших после полуночи по всемирному времени (Universal Time)1. Если по каким-либо причинам невозможно использовать представление стандартного времени, маршрутизатор может использовать любую форму представления местного времени. При этом он должен установить в единицу старший бит поля временной метки.

Естественно, что временные метки, занесенные в список различными компьютерами, будут не всегда согласованными, даже если для их представления используется всемирное время. Причина в том, что каждый компьютер вычисляет значение временной метки, основываясь на показаниях своих локальных часов, которые могут отличаться от показаний часов другого компьютера. Поэтому временные метки следует рассматривать как приблизительные оценки, независимо от формата их представления.

Может показаться странным, что в параметре регистрации временных меток предусмотрена возможность записи маршрутизаторами своих IP-адресов. Выше уже шла речь о том, что аналогичную возможность обеспечивает параметр регистрации маршрута дейтаграммы. Однако запись IP-адресов вместе с регистрацией временных меток позволяет избежать неоднозначности. Кроме того, IP-адреса позволяют получателю определить не только временные параметры прохождения дейтаграммы, но и точный путь, по которому она была доставлена.

1. Всемирное время соответствует времени нулевого (Гринвичского) меридиана.

**Обработка параметров при фрагментации дейтаграммы**

Идея применения бита копирования, расположенного в октете кода параметра, теперь уже должна быть понятна (см. рис. 10.13).

Рис. 10.13. Октет кода параметра поделен на три поля, длина которых составляет 1, 2 и 5 битов

При выполнении фрагментации дейтаграммы маршрутизатор может продублировать некоторые ее параметры во всех фрагментах, а часть параметров оставить только в первом фрагменте. В качестве примера рассмотрим параметр регистрации маршрута следования дейтаграммы. Выше уже было сказано, что каждый фрагмент дейтаграммы обрабатывается независимо от других. Следовательно, нет никакой гарантии, что все фрагменты достигнут конечного получателя по одному и тому же маршруту. Если параметр регистрации маршрута будет находиться во всех фрагментах дейтаграммы, то для одной дейтаграммы конечный получатель сможет выделить несколько разных списков IP-адресов узлов маршрутизации. В результате будет трудно понять, по какому маршруту прошла собранная дейтаграмма. Поэтому в стандарте IP указано, что параметр регистрации маршрута дейтаграммы должен быть скопирован только в один из ее фрагментов.

Однако не все параметры IP-дейтаграммы должны находиться только в одном фрагменте. В качестве примера давайте рассмотрим параметр маршрутизации от источника, который позволяет указать маршрут дейтаграммы в объединенной сети. Нетрудно догадаться, что этот параметр должен быть продублирован во все фрагменты дейтаграммы. В противном случае часть фрагментов может быть доставлена по другому маршруту, отличному от указанного. Таким образом, в октете кода параметра маршрутизации от источника всегда должен быть установлен бит копирования.

**Проверка знаний: Доставка дейтаграмм без установки соединения**

**Задание 1**

Продолжите верно фразу.

Строгая маршрутизация от источника означает, что в заголовке дейтаграммы...

указан точный путь

указан точный конечный адрес

указано точное имя

**Задание 2**

Почему служба доставки, не требующая установки соединения, названа ненадежной?

она не гарантирует доставку получателю

крайне легко проникнуть к данным посторонним лицам

отсутствует надежный канал передачи данных

**Задание 3**

В каких целях параметры протокола IP помещаются в дейтаграммы?

возможности для восстановления

резервного копирования

тестирования и отладки

**Задание 4**

В какое количество фрагментов *по стандарту IP* должен быть скопирован параметр регистрации маршрута дейтаграммы?

в один

в два

в любое количество

**Задание 5**

Как называется процесс разделения дейтаграммы?

фрагментация

разбивка

дробление

**Задание 6**

Каков максимальный размер дейтаграммы стандарта IPv4?

65 535 октетов

750 октетов

17 593 октетов

**Задание 7**

Какая служба находится на нижнем («фундаментальном») уровне в объединённой сети на основе протокола TCP/IP?

служба поддержки прикладных программ

надежная транспортная служба

служба доставки пакетов, не требующая установки соединения

**Задание 8**

Что определяют Биты D, Т, и R в обслуживании дейтаграмм?

желательный тип транспортировки

размер данных

адрес назначения

**Задание 9**

Что находится в заголовке дейтаграммы?

IP-адреса

физические адреса

сетевые адреса

**Задание 10**

Что должен занести при отправке IP-дейтаграммы компьютер в поле идентификационных данных ее заголовка?

уникальное имя

уникальное число

уникальный IP-адрес

**Задание 11**

Для чего предназначен младший (нулевой) бит поля флажков?

сборки дейтаграммы

идентификации фрагментов дейтаграммы

проверки на целостность дейтаграммы

**Протокол IP: маршрутизация дейтаграмм**

В предыдущих разделах уже шла речь о том, что в основе работы всех служб объединенной сети лежит система доставки пакетов, не требующая установки соединения, а также о том, что основной единицей передачи данных в семействе протоколов TCP/IP является IP-дейтаграмма. В этом разделе мы продолжим рассмотрение базовой службы доставки пакетов, не требующей установки соединения, и опишем, как маршрутизаторы осуществляют перенаправление дейтаграмм и их доставку до конечного получателя.

В разделе “Протокол IP: доставка дейтаграмм без установки соединения” был рассмотрен формат дейтаграммы, который можно считать статичным свойством протокола IP. В этом разделе описываются механизмы маршрутизации, относящиеся к рабочим характеристикам данного протокола.

В [следующем разделе](https://it.rfei.ru/course/%7EGZqN/-/%7EJ26Z) мы закончим рассмотрение основных черт протокола IP, описав способы обработки ошибочных ситуаций. В последующих разделах будет показано, как на основе протокола IP создаются протоколы более высокого уровня, обеспечивающие нужный уровень услуг.

* [Маршрутизация в объединенной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69053b)
* [Прямая и непрямая доставка дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69053c)
* [Доставка дейтаграмм по одной физической сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69053d)
* [Непрямая доставка](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69053e)
* [Алгоритм табличной IР-маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69053f)
* [Маршрутизация "на шаг вперед"](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690540)
* [Стандартный маршрут следования дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690541)
* [Маршрутизация отдельных узлов сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690542)
* [Алгоритм IР-маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690543)
* [Маршрутизация с использованием IР-адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690544)
* [Обработка входящих дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690545)
* [Создание таблиц маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690546)
* [Проверка знаний: маршрутизация дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690547)

**Маршрутизация в объединенной сети**

В системах с коммутацией пакетов под термином *маршрутизация (routing)* понимается процесс выбора пути, по которому должны передаваться пакеты к конечному получателю. Термином *маршрутизатор (router)* мы будем называть компьютер, который выбирает маршрут следования дейтаграмм.

Маршрутизация выполняется на нескольких уровнях. Например, в рамках глобальной сети, состоящей из нескольких коммутаторов пакетов, соединенных между собой физическими каналами связи, маршрутизация проходящих по ней пакетов выполняется самой сетью. Процесс маршрутизации начинается с момента попадания пакета в глобальную сеть и продолжается до момента его выхода за пределы сети. Причем внутри глобальной сети процесс маршрутизации происходит абсолютно автономно. Другими словами, машины, находящиеся за пределами глобальной сети, не могут повлиять на маршрутизацию проходящих по ней пакетов. Для них эта сеть является черным ящиком, выполняющим доставку пакетов из пункта А в пункт Б.

Напомним, что основная задача, которая ставилась при разработке протокола IP, — создать единую виртуальную сеть, объединяющую в себе множество физических сетей, в которой реализованы средства доставки дейтаграмм, не требующие установки соединения с получателем. Поэтому основное внимание при изложении материала будет сосредоточено на процессе *перенаправления* IP-дейтаграмм (*IP forwarding*), который также называется *межсетевой маршрутизацией*, или *IP-маршрутизацией*. Данные, которые используются для выбора пути следования пакетов до конечного получателя, называются *маршрутной информацией*. Процесс IP-маршрутизации в объединенной сети напоминает работу обычной физической сети, поскольку он связан с поиском маршрута, по которому будут передаваться дейтаграммы. Однако в отличие от одной физической сети, в алгоритмах IP-маршрутизации следует учитывать, что доставка дейтаграмм будет проходить по разным физическим сетям.

Процесс маршрутизации в объединенной сети может быть затруднен, особенно в случае использования компьютеров с несколькими сетевыми подключениями. В идеальном случае программы маршрутизации должны анализировать загрузку сети, длину дейтаграммы, а также запрошенный в ее заголовке тип обслуживания и на основе полученных данных выбирать оптимальный маршрут. Однако большинство программ маршрутизации не обладают столь развитой логикой и выбирают маршрут следования дейтаграммы, исходя из ряда сделанных допущений о его минимальной длине.

Чтобы до конца разобраться с процессом IP-маршрутизации, мы должны вспомнить структуру объединенной сети, в которой используется протокол TCP/IP. В предыдущих разделах уже говорилось о том, что объединенная сеть состоит из нескольких физических сетей, связанных посредством специальных компьютеров, называемых *маршрутизаторами (routers)*. Каждый маршрутизатор напрямую подключается как минимум к двум сетями. Для сравнения следует сказать, что обычно компьютеры пользователей подключаются к одной физической сети. В предыдущих разделах отмечалось, что иногда компьютеры пользователей подключаются сразу к нескольким физическим сетям. Такие компьютеры, или узлы, называются *многоадресными (multi-homed host)*.

В процессе доставки IP-дейтаграмм до конечного получателя участвуют не только специализированные маршрутизаторы, но и обычные компьютеры, за которыми работают пользователи. Когда одной из прикладных программ, запущенных на компьютере пользователя, необходимо обменяться информацией с удаленным узлом, она обращается к программам протокола TCP/IP. В конечном итоге это приводит к отправке с компьютера пользователя одной или нескольких IP-дейтаграмм. Перед отправкой дейтаграммы компьютер должен просчитать ее начальный маршрут и решить, какому из ближайших узлов маршрутизации она должна быть перенаправлена. Как показано на рис. 11.1, расчет начального маршрута дейтаграммы выполняется даже в том случае, если компьютер имеет только одно физическое подключение к сети.

Рис. 11.1. Выполнение маршрутизации одноадресным узлом. При отправке дейтаграммы узел должен решить, какому из двух маршрутизаторов — R1 или R2 — ее следует перенаправить. Проблема состоит в том, что каждый из маршрутизаторов может доставить дейтаграмму по оптимальному маршруту только определенному количеству получателей

Основным назначением маршрутизаторов является определение пути следования дейтаграмм и принятие решения об их перенаправлении на один из ближайших узлов маршрутизации. Если с маршрутизаторами все более-менее понятно, то каково же назначение многоадресных узлов? Суть в том, что любой компьютер, имеющий несколько физических сетевых подключений, может выступать в роли маршрутизатора. Как будет показано ниже, любой многоадресный узел, на котором поддерживается протокол TCP/IP, имеет все необходимые для этого средства. Более того, в небольших сетевых центрах часто экономят на выделенных маршрутизаторах и совмещают на одном компьютере функции маршрутизатора и рабочей станции. Тем не менее в описании стандартов протокола TCP/IP четко разграничены функции узла сети и маршрутизатора. В случае совмещения этих функций на одном компьютере пользователь должен быть готов к тому, что его многоадресный узел будет выполнять, помимо основной, дополнительную работу, причем в самые неожиданные моменты времени. Поэтому мы будем различать узлы сети и маршрутизаторы, предполагая, что узлы сети не могут выполнять функции маршрутизатора по пересылке пакетов из одной сети в другую.

**Прямая и непрямая доставка дейтаграмм**

Для того чтобы упростить изложение, разделим процесс маршрутизации на две фазы, которые назовем *прямой (direct delivery)* и *непрямой (indirect delivery)* доставкой.

*Прямая доставка* является основой всех межсетевых взаимодействий. Она происходит при передаче дейтаграммы между двумя машинами по одной физической сети. Таким образом, прямая доставка дейтаграмм между двумя компьютерами может осуществляться только в том случае, если они подключены к одной физической системе передачи данных (например, сети Ethernet).

*Непрямая доставка* происходит в том случае, когда конечный получатель дейтаграммы находится в другой физической сети. При этом отправитель пересылает дейтаграмму ближайшему узлу маршрутизации, который выполняет ее дальнейшую доставку до конечного получателя.

**Доставка дейтаграмм по одной физической сети**

Ранее уже говорилось о том, что два компьютера могут обмениваться между собой физическими фреймами, если они подключены к одной физической сети. Для передачи IP-дейтаграммы отправитель помещает ее в физический фрейм (инкапсулирует дейтаграмму в физический фрейм), конвертирует IP-адрес получателя в физический адрес и отправляет по нему физический фрейм, используя установленное в компьютере сетевое оборудование. В разделе [Преобразование IP-адресов в физические адреса (ARP)](https://it.rfei.ru/course/%7EGZqN/-/%7EJ4Ux) было описано два возможных механизма преобразования адресов, а также протокол ARP, который используется для динамической привязки адресов в сетях, наподобие Ethernet. Процесс инкапсуляции дейтаграмм рассмотрен в разделе [Протокол IP: доставка дейтаграмм без установки соединения](https://it.rfei.ru/course/%7EGZqN/-/%7EyGAH). Таким образом, у вас уже есть необходимые знания, чтобы понять механизм прямой доставки. Подытожив все сказанное выше, можно сделать такой вывод.

Передача IP-дейтаграмм между двумя компьютерами, подключенными к одной физической сети происходит напрямую, без участия маршрутизаторов. Отправитель помещает дейтаграмму в физический фрейм, преобразует IP-адрес получателя в физический адрес и посылает по нему фрейм, используя сетевое оборудование.

Теперь осталось выяснить, как отправитель узнает о том, что он находится с конечным получателем в одной физической сети. Ответить на этот вопрос очень просто. В предыдущих разделах говорилось о том, что IP-адрес состоит из двух частей: префикса, идентифицирующего сеть, и суффикса, определяющую узел в сети. Поэтому, чтобы определить, находится ли получатель в одной физической сети с отправителем, последний должен выделить номер сети из IP-адреса получателя и сравнить его с номером сети, выделенным из собственного IP-адреса. Если номера сетей совпадают, значит, дейтаграмма может быть послана получателю напрямую. Только что было продемонстрировано одно из основных преимуществ системы адресации в объединенной сети, использующей протокол TCP/IP.

Для всех компьютеров, подключенных к одной физической сети, префиксы IP-адресов, идентифицирующие сеть, совпадают. Поскольку для извлечения префикса из IP-адреса требуется всего несколько машинных команд, процесс выяснения возможности напрямую отправить дейтаграмму конечному получателю является чрезвычайно эффективным.

В объединенной сети прямая доставка выполняется на заключительном этапе пересылки любой дейтаграммы, независимо от того, через какое количество сетей и промежуточных маршрутизаторов она прошла. Последний из маршрутизаторов, находящийся на пути следования дейтаграммы и подключенный к одной физической сети с конечным получателем, выполняет ее прямую доставку до получателя. Таким образом, прямую доставку дейтаграмм следует рассматривать как частный случай общего процесса маршрутизации, выполняемого на всем этапе следования дейтаграммы от отправителя до конечного получателя. При прямой доставке дейтаграмма не проходит через промежуточные узлы маршрутизации.

**Непрямая доставка**

Процесс непрямой доставки является намного сложнее по сравнению с прямой доставкой дейтаграмм, поскольку отправителю нужно определить адрес ближайшего маршрутизатора, находящегося в одной с ним сети, которому должна быть послана дейтаграмма.

Чтобы понять, как осуществляется маршрутизация, представьте себе большую объединенную сеть, к которой посредством маршрутизаторов подключено множество физических сетей. Представьте также два удаленных друг от друга компьютера, подключенных к объединенной сети, которым необходимо обменяться информацией. Как только одному из компьютеров нужно отправить дейтаграмму другому компьютеру, он инкапсулирует ее в сетевой фрейм и пересылает его по физической сети ближайшему маршрутизатору. По определению, данную операцию может выполнить любой компьютер, поскольку все физические сети соединены между собой посредством маршрутизаторов. А это означает, что к каждой физической сети подключен как минимум один маршрутизатор. Таким образом, отправитель всегда может связаться с маршрутизатором по одной физической сети. Приняв дейтаграмму, маршрутизатор с помощью своего программного обеспечения извлекает ее из сетевого фрейма и передает на обработку программам протокола IP. После этого программы маршрутизации выполняют поиск адреса следующего маршрутизатора, находящегося на пути следования пакета до конечного получателя. Как только будет определен IP-адрес следующего узла маршрутизации, дейтаграмма снова помещается в сетевой фрейм и пересылается этому узлу по соответствующему участку физической сети. Этот процесс повторяется многократно до тех пор, пока дейтаграмма не будет отправлена конечному получателю методом прямой доставки. Подводя итог всему сказанному, можно отметить следующее.

Маршрутизаторы, в объединенной сети на основе протокола TCP/IP составляют взаимосвязанную и скооперированную структуру. Во время доставки дейтаграмма передается от одного маршрутизатора до другого до тех пор, пока она не достигнет маршрутизатора, находящегося с конечным получателем в одной физической сети. На заключительном этапе доставки дейтаграмма посылается конечному получателю напрямую.

В описанном выше процессе маршрутизации непонятными остались два момента. Как маршрутизатор узнает, куда именно нужно передавать каждую из полученных дейтаграмм? Как компьютер отправителя определяет адрес ближайшего маршрутизатора, которому должна быть передана дейтаграмма, следующая в заданном направлении? Эти два вопроса тесно связаны, поскольку оба они затрагивают тему IP-маршрутизации. Чтобы ответить на них, мы должны сначала рассмотреть две важные темы. В этом разделе будет описан простой алгоритм маршрутизации, использующий специальные таблицы. А в одном из следующих разделов мы поговорим о том, как маршрутизаторы определяют новые пути следования дейтаграмм.

**Алгоритм табличной IР-маршрутизации**

Обычно маршрутизация дейтаграмм в объединенной сети выполняется с помощью специальных *таблиц межсетевой маршрутизации (Internet routing table)*, которые иногда называют *таблицами IP-маршрутизации (IP routing table)*. В них хранится информация о возможных путях следования дейтаграмм и способах их достижения. Поскольку в процесс маршрутизации вовлечены как компьютеры пользователя, так и сетевые маршрутизаторы, такие таблицы должны храниться на *каждой* машине объединенной сети, независимо от выполняемых ею функций. В момент отправки дейтаграммы с компьютера пользователя запущенная на нем программа маршрутизации (она является частью программ поддержки протокола IP) с помощью таблицы маршрутизации определяет узел сети, которому следует послать эту дейтаграмму.

Какая же информация должна храниться в таблицах маршрутизации? Если на каждом компьютере в таблице маршрутизации будет содержаться информация о всех возможных адресах получателя, то очевидно, что состояние такой таблицы очень трудно поддерживать актуальным постоянно. Более того, поскольку количество потенциальных адресатов очень велико, в компьютере пользователя может не хватить ни оперативной, ни дисковой памяти для хранения всего массива информации.

Интуитивно хотелось бы использовать принцип намеренного скрытия информации и сделать так, чтобы компьютеры выполняли маршрутизацию на основании минимально необходимых данных. Например, логично, чтобы подробная информация о конкретных узлах была сосредоточена в месте их подключения. Также крайне желательно, чтобы удаленный компьютер выполнял маршрутизацию пакетов к этим узлам, не вдаваясь в излишние подробности.

Следует отметить, что система IP-адресации как раз и создавалась с учетом этих требований. Напомним, что у всех машин, подключенных к одной физической сети, должны совпадать префиксы IP-адресов (т.е. как раз та часть IP-адреса, которая идентифицирует конкретную сеть). Выше уже шла речь о том, что подобный принцип назначения адресов позволяет с высокой эффективностью проверять, возможна ли прямая доставка дейтаграмм между двумя машинами. Таким образом, в таблицах маршрутизации достаточно указать только один префикс сети, а не IP-адреса всех ее машин.

**Маршрутизация "на шаг вперед"**

Использование выделенного из адреса получателя префикса, который идентифицирует сеть, вместо полного адреса узла позволяет резко повысить эффективность маршрутизации и сократить размер соответствующих таблиц. Кроме того, подобный подход позволяет скрыть информацию, сосредоточить данные о конкретных узлах в рамках локальной среды, к которой они подключены. Обычно в таблице маршрутизации содержатся пары значений (N, R), где N представляет IP-адрес *сети* получателя, а R является IP-адресом “следующего” по порядку маршрутизатора, расположенного на пути следования пакетов до сети N. Маршрутизатор R называется *ближайшей точкой перехода (next hop)*, а сама идея хранения в таблице маршрутизации адреса ближайшей точки перехода для каждого получателя называется *маршрутизацией на шаг вперед (next-hop routing)*. Таким образом, в таблице маршрутизации, хранящейся на узле R, содержатся данные о пути следования дейтаграмм от узла R до ближайшей точки перехода в направлении сети получателя. Заметьте, что маршрутизатор R не располагает данными о полном маршруте дейтаграммы к конечному получателю.

Важно отметить, что в каждом элементе таблицы маршрутизации содержится адрес маршрутизатора, которому могут быть посланы дейтаграммы по одной физической сети. Это означает, что все маршрутизаторы, адреса которых указаны в таблице маршрутизации узла *М*, должны находиться в тех же физических сетях, к которым напрямую подключена машина *М*. В момент отправки дейтаграммы с машины *М*, программное обеспечение протокола IP анализирует IP-адрес получателя и выделяет из него префикс сети. На основании значения этого префикса машина *М* принимает решение о том, какому из маршрутизаторов, находящихся “на расстоянии прямой видимости”, следует направить дейтаграмму.

На практике принцип скрытия информации применяется не только для маршрутизаторов, но и для узлов сети. Поэтому, хотя таблицы маршрутизации есть на всех узлах сети, в них должна храниться только необходимая для отправки дейтаграммы информация. Идея состоит в том, чтобы максимально освободить узлы сети от выполнения маршрутизации дейтаграмм и передать эти функции специализированным маршрутизаторам.

На рис. 11.2 приведен пример, позволяющий объяснить принцип применения таблиц маршрутизации. Объединенная сеть состоит из четырех физических сетей, соединенных посредством трех маршрутизаторов.

Рис. 11.2. Пример объединенной сети, состоящей из 4 физических сетей и трех маршрутизаторов (а); таблица маршрутизации устройства R (б)

На рис. 11.2 показана таблица маршрутизации, находящаяся на устройстве R. Поскольку маршрутизатор R напрямую подключен к двум сетям (20.0.0.0 и 30.0.0.0), для отправки пакетов любым машинам этих сетей он может пользоваться методом прямой доставки (преобразование логических адресов в физические, вероятнее всего, будет выполняться с помощью протокола ARP). При получении дейтаграммы, адресованной одному из узлов, расположенных в сети 40.0.0.0, маршрутизатор R должен перенаправить ее для дальнейшей доставки маршрутизатору S, адрес которого — 30.0.0.7. Поскольку маршрутизатор S имеет прямое подключение к сети 40.0.0.0, он сможет переслать данную дейтаграмму конечному получателю методом прямой доставки. В свою очередь маршрутизатор R может связаться напрямую с маршрутизатором S, поскольку последний имеет прямое подключение к сети 30.0.0.0.

Как видно из рис. 11.2, размер таблицы маршрутизации устройства R зависит от количества физических сетей в объединенной сети. Увеличение количества элементов в этой таблице происходит только в случае подключения к объединенной сети новых физических сетей. Однако следует отметить, что на размер таблицы маршрутизации и на ее содержимое не влияет количество индивидуальных компьютеров, подключенных ко всем сетям. Таким образом, можно сформулировать следующий основополагающий принцип.

Программное обеспечение протокола IP сохраняет в таблицах маршрутизации только информацию об адресах сетей (а не об адресах отдельных узлов этих сетей), в которых могут находиться потенциальные получатели дейтаграмм. Это позволяет избавиться от ненужной информации, уменьшить размер таблиц и повысить эффективность процесса маршрутизации.

Выбор маршрута следования дейтаграмм на основе адреса сети получателя имеет несколько недостатков.

Во-первых, в большинстве реализаций сетевого программного обеспечения такой подход означает, что весь поток данных, направленных в конкретную физическую сеть, будет проходить по одному и тому же маршруту. Как следствие, в случае если к сети получателя существует несколько возможных маршрутов следования, они не могут быть использованы одновременно. Кроме того, по одному и тому же маршруту будут направляться все дейтаграммы, независимо от запрошенного в их заголовках типа обслуживания (минимальное время доставки или высокая пропускная способность сети).

Во-вторых, поскольку непосредственное соединение с конечным получателем может установить только последний из маршрутизаторов, находящихся на пути следования дейтаграммы, отправитель не в состоянии определить, подключен ли к физической сети узел получателя и нормально ли он функционирует. Таким образом, нужно найти способ, с помощью которого маршрутизатор сможет сообщить отправителю о проблемах, возникших при доставке дейтаграммы.

В-третьих, поскольку все маршрутизаторы направляют потоки данных независимо друг от друга, может возникнуть ситуация, когда дейтаграмма, посланная от узла *А* к узлу *В*, будет проходить по другому маршруту, чем дейтаграмма, посланная от узла *В* к узлу *А*. Следовательно, нужно предусмотреть средства взаимодействия между маршрутизаторами, которые бы всегда гарантировали возможность двухстороннего обмена данными между этими узлами.

**Стандартный маршрут следования дейтаграмм**

Существует еще один метод, позволяющий избавиться от лишней информации и уменьшить размеры таблиц маршрутизации. Речь идет об объединении нескольких элементов таблицы маршрутизации в один стандартный блок. Суть его заключается в том, что программное обеспечение протокола IP при выполнении процедуры маршрутизации вначале должно просмотреть таблицу маршрутизации в поисках информации о сети получателя. Если в таблице маршрутизации информация не найдена, дейтаграмма посылается *стандартному маршрутизатору (default router)*.

Понятие стандартного маршрута следования дейтаграммы широко применяется в небольших сетевых центрах, которым выделено несколько IP-адресов и которые имеют только одно сетевое соединение с остальной частью объединенной сети. В качестве примера можно привести небольшую локальную сеть, к которой подключено несколько компьютеров пользователей и только один маршрутизатор, связывающий их с внешним миром. Маршрутизация в такой сети будет выполняться в два этапа: анализ адресов локальной сети и отправка дейтаграмм, предназначенных для внешних адресатов, локальному маршрутизатору. Даже если локальная сеть предприятия состоит из нескольких физических сетей, применение стандартных маршрутов позволяет упростить процедуру маршрутизации — после проверки узлом сети адресов всех локальных сетей дейтаграммы, предназначенные для получателей за пределами локальной сети, передаются стандартному маршрутизатору.

**Маршрутизация отдельных узлов сети**

Выше мы уже говорили о том, что в сетях TCP/IP маршрутизация выполняется на основе адреса сети, а не адресов отдельных ее узлов. Тем не менее в большинстве реализаций протокола IP как частный случай предусмотрена возможность выполнения маршрутизации для адресов отдельных узлов. Это позволяет сетевым администраторам более точно распределять потоки данных в сети, тестировать участки сети, а также управлять правами доступа к сетевым ресурсам. Возможность задать индивидуальный маршрут дейтаграмм до конкретной машины особенно полезна при отладке сетевых подключений или проверке таблиц маршрутизации.

**Алгоритм IР-маршрутизации**

Принимая во внимание все сказанное выше, можно сформулировать алгоритм, используемый для маршрутизации IP-дейтаграмм (листинг 11.1). По заданному IP-адресу конечного получателя и существующей таблице маршрутизации, данный алгоритм определяет адрес ближайшей точки перехода, куда следует направить дейтаграмму. Не забывайте, что адрес ближайшей точки перехода должен находиться в той же сети, к которой имеет прямое подключение отправитель дейтаграммы.

**Листинг 11.1. Алгоритм маршрутизации IP-дейтаграмм**

Маршрутизация*дейтаграммы (Дейтаграмма, Таблица*Маршрутизации)

1. Извлечь из *Дейтаграммы* IP-адрес конечного получателя *D* и определить префикс сети *N*.
2. Если *N* совпадает с префиксом одной из сетей, к которой непосредственно подключена машина, выполнить прямую доставку дейтаграммы получателю *D* по соответствующей сети. (При этом нужно определить физический адрес машины *D*, инкапсулировать дейтаграмму в сетевой фрейм и отправить фрейм получателю).
3. Иначе, если в таблице маршрутизации указан специфический маршрут к машине *D*, переслать дейтаграмму в ближайшую точку перехода, адрес которой берется из таблицы.
4. Иначе, если в таблице маршрутизации указан маршрут для сети *N*, переслать дейтаграмму в ближайшую точку перехода, адрес которой берется из таблицы.
5. Иначе, если в таблице маршрутизации указан стандартный маршрут следования, отправить дейтаграмму стандартному маршрутизатору, адрес которого берется из таблицы.
6. Иначе, сгенерировать ошибку маршрутизации.

**Маршрутизация с использованием IР-адресов**

Прежде всего следует отметить, что, за исключением уменьшения значения поля времени жизни дейтаграммы и пересчета контрольной суммы ее заголовка, при выполнении маршрутизации в тело дейтаграммы не вносятся никакие изменения. В частности, не изменяются IP-адреса, находящиеся в полях заголовка дейтаграммы. Они всегда будут указывать на начального отправителя дейтаграммы и ее конечного получателя1. В результате выполнения алгоритма маршрутизации программа должна определить новый IP-адрес промежуточного получателя или IP-адрес машины, которой следует переслать дейтаграмму. В качестве промежуточного получателя чаще всего выступает следующий маршрутизатор, расположенный на пути следования дейтаграммы к конечному получателю. Если дейтаграмма может быть доставлена непосредственно получателю, то новый IP-адрес будет совпадать с адресом конечного получателя. Выше уже шла речь о том, что IP-адрес, полученный в результате выполнения алгоритма маршрутизации, называется адресом *ближайшей точки перехода*, поскольку он указывает на узел в сети, которому должна быть послана дейтаграмма. Возникает вопрос, где в протоколе IP хранится адрес ближайшей точки перехода, поскольку в заголовке дейтаграммы поле для него не предусмотрено?

Ответ прост: нигде. Выполнив маршрутизацию, программа протокола IP передает дейтаграмму и адрес ближайшей точки перехода сетевому программному обеспечению более низкого уровня. Оно поддерживает работу физической сети, по которой дейтаграмма должна быть послана. Низкоуровневое сетевое программное обеспечение преобразует IP-адрес ближайшей точки перехода в соответствующий физический адрес, помещает его в заголовок фрейма, а дейтаграмму — в область данных сетевого фрейма. Сформированный сетевой фрейм посылается в сеть. После определения физического адреса ближайшей точки перехода ее IP-адрес теряется.

Может показаться странным, что в таблицах маршрутизации хранятся IP-адреса ближайших точек перехода (а не их физические адреса) для каждой из сетей, в которых находятся машины получателей. Ведь выше шла речь о том, что перед посылкой дейтаграммы в ближайшую точку перехода ее IP-адрес преобразуется в физический адрес. Поэтому, если представить себе абстрактную машину, отправляющую большое количество дейтаграмм по одним и тем же адресам, использование IP-адресов в таблице маршрутизации приведет к резкому снижению эффективности работы сети в целом. Ведь каждый раз после получения очередной дейтаграммы программное обеспечение протокола IP должно извлечь из ее заголовка IP-адрес получателя и с помощью таблицы маршрутизации определить адрес ближайшей точки перехода. После этого дейтаграмма вместе с адресом ближайшей точки перехода передается сетевому программному обеспечению более низкого уровня, отвечающему за работу сетевого интерфейса. Низкоуровневое программное обеспечение должно по IP-адресу определить физический адрес ближайшей точки перехода. Если бы в таблице маршрутизации хранились физические адреса ближайших точек перехода, то каждый из IP-адресов нужно было бы преобразовывать в физический адрес только один раз. В результате можно было бы сэкономить время центрального процессора сетевой машины, не выполняя ненужных вычислений.

Почему же все-таки в реализациях протокола IP не используются физические адреса при определении маршрутов следования дейтаграмм и создании таблиц маршрутизации? На то есть две причины, которые вы поймете, рассмотрев рис. 11.3.

Рис. 11.3. Программа протокола IP и используемая ею таблица маршрутизации находятся выше воображаемой границы раздела адресов. Использование при маршрутизации только IP-адресов облегчает поиск или изменение маршрута следования дейтаграмм и позволяет скрыть особенности структуры физических адресов сетевого оборудования

Во-первых, таблица маршрутизации является связующим звеном между программами протокола IP, выполняющими маршрутизацию дейтаграмм, и высокоуровневым программным обеспечением, которое управляет маршрутами следования дейтаграмм. При возникновении проблем в сети администраторы часто прибегают к анализу таблиц маршрутизации. Поэтому использование в них IP-адресов упрощает задачу сетевым администраторам и позволяет легко проконтролировать правильность обновления таблиц маршрутизации соответствующими программами.

Во-вторых, основная цель при разработке протокола IP состояла в том, чтобы скрыть детали используемых низкоуровневых сетевых технологий.

На рис. 11.3 изображена важная концептуальная деталь — воображаемая *граница раздела адресов (address boundary)*, которая позволяет показать, на каком уровне происходит разделение между низкоуровневым сетевым программным обеспечением, использующим физические адреса, и высокоуровневыми прикладными программами, работающими исключительно с логическими IP-адресами. При разработке программного обеспечения, логика работы которого относится к уровню, расположенному выше этой линии, должны использоваться исключительно IP-адреса. Круг применения физических адресов должен ограничиваться только несколькими небольшими низкоуровневыми подпрограммами. Ниже будет показано, что введение этой границы облегчает понимание принципа реализации программ остальных протоколов семейства TCP/IP, а также их отладку и дальнейшее сопровождение.

1. За исключением случая, когда в дейтаграмме указаны параметры маршрутизации от источника.

**Обработка входящих дейтаграмм**

Выше был описан механизм IP-маршрутизации, а также методы обработки исходящих дейтаграмм. Однако, должно быть уже понятно, что программы протокола IP должны также обрабатывать и входящие дейтаграммы.

Получив сетевой фрейм, низкоуровневое сетевое программное обеспечение компьютера выделяет из него IP-дейтаграмму, которая затем передается на обработку модулю протокола IP. Если IP-адрес получателя, указанный в заголовке дейтаграммы, совпадает с одним из IP-адресов интерфейсов компьютера, программа поддержки протокола IP принимает такую дейтаграмму и передает ее для дальнейшей обработки соответствующему модулю протокола более высокого уровня1. Если же IP-адреса не совпадают, программное обеспечение протокола IP должно аннулировать дейтаграмму (не забывайте, что рабочим станциям запрещено пересылать дальше дейтаграммы, которые случайно попали к ним из-за ошибок в настройке таблиц маршрутизации).

В отличие от узлов сети, маршрутизаторы выполняют пересылку пакетов. Поэтому, когда IP-дейтаграмма поступает на маршрутизатор, она передается на обработку модулю протокола IP. В нем анализируется адрес получателя и определяется, может ли маршрутизатор доставить дейтаграмму непосредственно конечному получателю или он должен передать ее для дальнейшей обработки другому маршрутизатору. Как и в случае рабочей станции, при совпадении IP-адреса конечного получателя, находящегося в заголовке дейтаграммы с одним из IP-адресов интерфейсов маршрутизатора программы протокола IP передают поступившую дейтаграмму на обработку программам протокола более высокого уровня. Если маршрутизатор не может непосредственно передать дейтаграмму конечному получателю, программы протокола IP выполняют для нее маршрутизацию, используя в этих целях стандартный алгоритм и информацию из локальной таблицы маршрутизации.

Определение возможности доставки дейтаграммы получателю — не такая простая задача, как может показаться на первый взгляд. Напомним, что даже обычная рабочая станция может иметь несколько сетевых интерфейсов, подключенных к разным физическим сетям, каждому из которых назначается отдельный IP-адрес. Получив дейтаграмму, компьютер должен сравнить IP-адрес получателя с IP-адресами всех своих сетевых интерфейсов. Если один из адресов совпадет, дейтаграмма принимается для дальнейшей обработки. Компьютер также принимает для обработки дейтаграммы, разосланные по локальной сети в широковещательном режиме. При этом IP-адрес получателя в заголовке дейтаграммы, должен совпадать с ограниченным широковещательным адресом или направленным широковещательным адресом этой сети.

Использование бесклассовых методов адресации, подсетей и многоадресатных адресов еще больше усложняет дело. Однако в любом случае, если IP-адрес получателя дейтаграммы не совпадет ни с одним из локальных IP-адресов данного компьютера, программы протокола IP уменьшают на единицу значение поля времени жизни в заголовке дейтаграммы и на основе этого значения решают, как поступить с дейтаграммой. Если значение времени жизни стало равным нулю, дейтаграмма аннулируется. В противном случае вычисляется новое значение контрольной суммы заголовка и выполняется маршрутизация дейтаграммы.

Должен ли любой подключенный к сети компьютер выполнять дальнейшую пересылку всех полученных IP-детайграмм? Очевидно, что эти функции по определению должен выполнять маршрутизатор. Выше уже шла речь о том, что функции маршрутизатора может выполнять практически любой многоадресный узел сети, даже если обычно он используется только в качестве рабочей станции пользователя. И хотя такое решение нельзя назвать удачным, тем не менее, установив соответствующие параметры настройки протокола TCP/IP можно добиться того, чтобы многоадресный узел пересылал дейтаграммы так же, как обычный маршрутизатор.

А как быть с остальными узлами сети, которые не предназначены для работы в качестве маршрутизаторов? Ответ прост: узлы сети, которые не являются маршрутизаторами, *не должны* выполнять пересылку случайно попавших к ним дейтаграмм; такие дейтаграммы должны быть аннулированы.

Существует по крайней мере четыре причины, по которым узлы сети, не являющиеся маршрутизаторами, не должны пересылать дейтаграммы.

Во-первых, получение компьютером пользователя дейтаграмм, предназначенных для других машин сети, свидетельствует о возникновении проблем в системе межсетевой адресации, маршрутизации или доставки. Если бы узел сети, выполняя функции маршрутизатора, корректно пересылал дейтаграммы, то о возникших проблемах в сети никто бы не узнал.

Во-вторых, процесс маршрутизации связан с дополнительными потоками пересылаемых по сети данных, а также c нагрузкой на центральный процессор компьютера, предсказать которые заранее невозможно. Все это может замедлить работу прикладных программ, запущенных на компьютере пользователя.

В-третьих, возникновение простых ошибочных ситуаций может привести к хаосу в сети. Предположим, что любой узел в сети может выполнять маршрутизацию дейтаграмм. Представьте себе, что произойдет, если одна из машин случайно отправит в широковещательном режиме дейтаграмму, которая предназначается для одной из машин сети, например *Н*. Поскольку для отправки дейтаграммы был выбран широковещательный режим, ее копию получит каждый из узлов сети и перешлет ее узлу *Н*. В результате компьютер *Н* будет завален множеством копий дейтаграммы.

В-четвертых, как будет описано в следующих разделах, маршрутизаторы, кроме перенаправления потоков данных в сети, выполняют ряд других полезных функций. Одна из них — уведомление отправителя о возникших в сети проблемах, связанных с доставкой его дейтаграммы. Для этой цели используется специальный протокол, о котором пойдет речь в следующем разделе.

Следует отметить, что обычные узлы сети не рассылают подобных уведомлений (причина все та же — чтобы не перегружать отправителя одними и теми же сообщениями об ошибках). Кроме того, маршрутизаторы периодически рассылают информацию о маршрутах следования дейтаграмм. Делается это для согласования информации в таблицах маршрутизации. Таким образом, если узел сети будет выполнять маршрутизацию дейтаграмм, не поддерживая при этом всех функций маршрутизатора, то результат подобных действий — непредсказуем.

1. Обычно маршрутизатор обрабатывает только служебные дейтаграммы, предназначенные для проверки работоспособности канала связи или содержащие команды управления процессом маршрутизации. Однако в некоторых случаях маршрутизатор отслеживает и сохраняет у себя все дейтаграммы, посланные по локальной сети в широковещательном режиме.

**Создание таблиц маршрутизации**

Выше в этом разделе было описано, как в протоколе IP выполняется маршрутизация дейтаграмм на основе информации, хранящейся в таблицах маршрутизации. Однако мы не упомянули о том, как в системе инициализируются эти таблицы или обновляется их содержимое при изменении топологии сети. Затронутый нами вопрос более подробно описан в последующих разделах. Там же речь пойдет о специальном протоколе, который используется маршрутизаторами для согласования содержимого своих таблиц маршрутизации. А пока вам достаточно понять один важный момент. Программы протокола IP обращаются к таблице маршрутизации всякий раз, когда нужно определить адрес ближайшей точки перехода, по которому следует передать дейтаграмму. Поэтому изменение содержимого таблиц маршрутизации автоматически влечет за собой изменение маршрута следования дейтаграмм.

**Проверка знаний: маршрутизация дейтаграмм**

**Задание 1**

Почему в реализациях протокола IP не используются физические адреса при определении маршрутов следования дейтаграмм и создании таблиц маршрутизации?

Основная цель при разработке протокола IP состояла в том, чтобы скрыть детали используемых низкоуровневых сетевых технологий.

Использование IP-адреса вместо физического значительно ускоряет передачу информации в сети из-за уменьшения нагрузки на вычислительные ресурсы узлов сети.

**Задание 2**

Что понимается под *маршрутной информацией*?

Данные, используемые для настройки маршрутизатора.

Данные, которые используются для выбора пути следования пакетов до конечного получателя.

Данные, которые передаются по пути следования пакетов до конечного пользователя.

**Задание 3**

Какая информация хранится в *таблице межсетевой маршрутизации* (Internet routing table)?

информация о всех возможных адресах получателя

информация только о конкретных узлах маршрутизации

информация о возможных путях следования дейтаграмм и способах их достижения

**Задание 4**

Что называют *ближайшей точкой перехода*?

IP-адрес начального отправителя дейтограммы

Ближайший узел сети

IP-адрес, полученный в результате выполнения алгоритма маршрутизации

**Задание 5**

Расставьте в правильном порядке алгоритм маршрутизации IP-дейтаграмм.

Извлечь из Дейтаграммы IP-адрес конечного получателя D и определить префикс сети N.

Иначе, если в таблице маршрутизации указан специфический маршрут к машине D, переслать дейтаграмму в ближайшую точку перехода, адрес которой берется из таблицы.

Если N совпадает с префиксом одной из сетей, к которой непосредственно подключена машина, выполнить прямую доставку дейтаграммы получателю D по соответствующей сети. (При этом нужно определить физический адрес машины D, инкапсулировать дейтаграмму в сетевой фрейм и отправить фрейм получателю).

**Задание 6**

Чем отличается доставка дейтаграммы между компьютерами в одной физической сети от механизма доставки, когда конечный получатель дейтаграммы находится в другой физической сети?

Передача IP-дейтаграмм между двумя компьютерами, подключенными к одной физической сети, происходит c участием маршрутизаторов. Во время доставки дейтаграмма передается от одного маршрутизатора до другого до тех пор, пока она не достигнет маршрутизатора, находящегося с конечным получателем в одной физической сети. На заключительном этапе доставки дейтаграмма посылается конечному получателю напрямую.

Передача IP-дейтаграмм между двумя компьютерами, подключенными к одной физической сети, происходит напрямую, без участия маршрутизаторов. Отправитель помещает дейтаграмму в физический фрейм, преобразует IP-адрес получателя в физический адрес и посылает по нему фрейм, используя сетевое оборудование.

**Задание 7**

В каком случае перед отправкой дейтаграммы компьютер должен выполнять расчет начального маршрута дейтаграммы?

Только в том случае, если компьютер имеет несколько физических подключений к сети.

Компьютер всегда должен просчитать начальный маршрут дейтаграммы и решить, какому из ближайших узлов маршрутизации она должна быть перенаправлена.

Только в том случае, если компьютер имеет одно физическое подключений к сети.

**Задание 8**

Какая служба занимается преобразованием ближайшей точки перехода в соответствующий физический адресс?

протокол TCP/IP

низкоуровневое сетевое программное обеспечение

нет необходимости преобразования IP-адреса в физический адрес

**Задание 9**

Маршрутизация на шаг вперед – это…

сохранение в таблицах маршрутизации информации об адресах отдельных узлов в сетях (а не только информацию об адресах сетей)

сохранение в таблицах маршрутизации только информации об адресах сетей (а не об адресах отдельных узлов этих сетей), в которых могут находиться потенциальные получатели дейтаграмм

сохранение в таблицах маршрутизации информации о уже осуществленных передачах дейтаграмм для быстрой отправки дейтаграмм потенциальным получателям

**Задание 10**

Должен ли любой подключенный к сети компьютер выполнять дальнейшую пересылку всех полученных IP-детайграмм?

должен отсылать уведомление отправителю о возникших в сети проблемах, связанных с доставкой его дейтаграммы

должен выполнять дальнейшую пересылку всех полученных IP-детайграмм

эти функции по определению должен выполнять маршрутизатор

**Протокол IP: обработка ошибок и управляющие сообщения (ICMP)**

В предыдущих разделах был описан механизм ненадежной доставки дейтаграмм протокола IP, не требующий установки соединения с получателем. Напомним, что доставка дейтаграммы конечному получателю происходит за счет ее пересылки от одного маршрутизатора до другого. Этот процесс продолжается до тех пор, пока дейтаграмма не попадет в ту сеть, к которой подключен конечный получатель. В результате последний из маршрутизаторов сможет переслать ее напрямую получателю. Однако в некоторых случаях маршрутизатор по какой-либо причине не может переслать дейтаграмму другому маршрутизатору или доставить ее получателю. Кроме того, иногда в сети возникают непредвиденные ситуации, которые могут повлиять на возможность пересылки дейтаграмм самим маршрутизатором (например, в случае перегрузки сети). Поэтому для маршрутизатора должны быть предусмотрены средства извещения отправителя о проблемах с доставкой дейтаграммы, чтобы он смог принять необходимые меры или попытался устранить причину возникшей проблемы.

В этом разделе описывается механизм взаимодействия маршрутизаторов и узлов объединенной сети, позволяющий им обмениваться служебной информацией и сообщениями об ошибках. Ниже будет показано, как этот механизм используется в маршрутизаторах для извещения узлов сети о возникших проблемах, а также возможность его использования узлами сети для проверки связи с конечным получателем.

* [Протокол межсетевых управляющих сообщений (ICMP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690555)
* [Уведомление об ошибках или коррекция ошибок?](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690556)
* [Доставка ICMP-сообщений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690557)
* [Формат ICMP-сообщений](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690558)
* [Проверка связи с получателем и его состояния (ping)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690559)
* [Форматы запроса на эхо и ответного сообщения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055a)
* [Извещение об отсутствии связи с получателем](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055b)
* [Перегрузка сети и управление потоком дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055c)
* [Формат сообщения о подавлении источника данных](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055d)
* [Запрос на изменение маршрута следования дейтаграмм от маршрутизатора](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055e)
* [Обнаружение замкнутых и слишком длинных маршрутов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69055f)
* [Уведомление об остальных проблемах](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690560)
* [Синхронизация часов и оценка времени передачи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690561)
* [Информационные запросы и ответы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690562)
* [Определение маски подсети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690563)
* [Поиск маршрутизатора](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690564)
* [Запрос адреса маршрутизатора](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690565)
* [Проверка знаний: Протокол IP. Обработка ошибок и управляющие сообщения (ICMP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690566)

**Протокол межсетевых управляющих сообщений (ICMP)**

В системах не требующих установки соединения с конечным получателем, которые мы рассматривали до сих пор, каждый маршрутизатор является автономным устройством. А это означает, что пересылка и доставка поступающих на маршрутизатор дейтаграмм выполняется без какого-либо участия со стороны их отправителей. Такая система будет работать очень хорошо только в том случае, если все компьютеры, вовлеченные в процесс доставки дейтаграмм, нормально функционируют, а все маршруты следования дейтаграмм согласованы.

К сожалению, составляющие любой большой коммуникационной системы рано или поздно выходят из строя. Кроме обрывов линий связи и сбоев в работе компьютеров, доставка дейтаграмм также невозможна в случае отключения (временного или постоянного) машины конечного получателя от сети, при обнулении счетчика времени жизни или когда один или несколько промежуточных маршрутизаторов оказываются настолько перегруженными, что не могут справиться со входящим потоком данных.

Основное отличие сети, реализованной на аппаратном уровне, от объединенной сети, реализованной на программном уровне, заключается в том, что в первом случае разработчик может добавить специальное оборудование, которое будет информировать подключенные к сети узлы о возникших проблемах. В объединенной сети, которая не имеет подобного аппаратного механизма оповещения, отправитель не сможет определить причину, по которой дейтаграмма не была доставлена получателю, — обусловлено ли это сбоем в работе локального или удаленного компьютера. В подобных случаях выявление источника проблемы становится крайне затруднительным. В протоколе IP также не предусмотрено никаких средств, позволяющих отправителю проверить связь с получателем или получить информацию о возникшей проблеме.

Поэтому, чтобы маршрутизаторы объединенной сети могли оповещать узлы сети о возникающих ошибках или нештатных ситуациях, разработчики добавили в семейство протоколов TCP/IP механизм рассылки специальных сообщений. Этот механизм, который назвали *протоколом межсетевых управляющих сообщений (Internet Control Message Protocol*, или *ICMP*), считается неотъемлемой частью протокола IP и должен быть обязательно включен в любую его реализацию.

Управляющие ICMP-сообщения (по аналогии с любыми другими сообщениями) перед передачей по объединенной сети помещаются в область данных IP-дейтаграммы. Однако конечным получателем таких сообщений является не конкретная прикладная программа или пользователь удаленного компьютера, а программное обеспечение протокола IP, запущенное на этом компьютере. Другими словами, полученное ICMP-сообщение, обрабатывает специальный модуль ICMP протокола IP. Обнаружив, что проблема была вызвана протоколом высокого уровня или прикладной программой, модуль ICMP информирует об этом соответствующий программный модуль. Таким образом, из всего сказанного выше напрашивается следующий вывод.

Протокол ICMP позволяет маршрутизаторам отправлять другим маршрутизаторам или узлам сети сообщения об ошибках или управляющие сообщения. Этот протокол обеспечивает средство связи между программами протокола IP двух компьютеров.

Первоначально протокол ICMP был предназначен для уведомления маршрутизаторами узлов сети о возникших проблемах с доставкой посланных ими пакетов. Однако его область применения не исчерпывается исключительно маршрутизаторами. Узел сети может отправить ICMP-сообщение любому узлу сети. Хотя стоит отметить, что в стандарте ограничивается использование узлами сети некоторых типов ICMP-сообщений. Таким образом, узел сети может использовать протокол ICMP как средство связи с маршрутизатором или другим узлом. Основным преимуществом использования протокола ICMP узлами сети является то, что при этом обеспечивается единый универсальный механизм обмена управляющими и информационными сообщениями любого типа.

**Уведомление об ошибках или коррекция ошибок?**

Говоря формально, протокол ICMP является *механизмом уведомления об ошибках*. В нем предусмотрены средства, позволяющие маршрутизатору сообщить отправителю о возникшей при доставке дейтаграммы проблеме. Хотя в спецификации протокола ICMP описаны допустимые способы его использования и предполагаемые действия, которые должны выполняться в ответ на получение уведомлений о ряде ошибок, тем не менее там не указан полный перечень действий, которые должны быть выполнены при возникновении каждой из возможных ошибок. Подводя итог, можно отметить.

Eсли при доставке дейтаграммы возникает нештатная ситуация, то с помощью средств протокола ICMP отправителю дейтаграммы может быть послано только соответствующее уведомление. Отправитель должен переслать это уведомление нужной прикладной программе или предпринять другие действия для устранения проблемы.

Во многих случаях ошибочные ситуации возникают вследствие некорректных действий со стороны отправителя дейтаграммы, однако это не всегда так. Поскольку протокол ICMP предназначен для уведомления отправителя дейтаграммы, его нельзя использовать для рассылки сообщений о возникшей проблеме промежуточным маршрутизаторам. Например, предположим, что дейтаграмма передается по некоторому маршруту и последовательно проходит через маршрутизаторы R1, R2, Rk. Предположим также, что в таблице маршрутизации устройства содержится некорректная информация, поэтому дейтаграмма будет переслана устройству RE. При этом маршрутизатор Rk с помощью протокола ICMP не сможет уведомить о возникшей проблеме устройство Rk поскольку сообщение об ошибке будет послано отправителю дейтаграммы. К сожалению, в данном случае отправитель дейтаграммы никак не сможет повлиять на источник проблемы и перенастроить таблицу маршрутизации устройства Rk. Более того, отправитель может и не определить, какой из маршрутизаторов работает некорректно.

Почему же тогда протокол ICMP предназначен для уведомления только отправителя дейтаграммы? Ответ на этот вопрос станет понятен после анализа формата дейтаграммы и принципов маршрутизации, которые были описаны в предыдущих разделах. В заголовке дейтаграммы предусмотрены поля только для адресов отправителя и конечного получателя. В нем не предусмотрено места для адресов всех маршрутизаторов, через которые прошла дейтаграмма по объединенной сети (за исключением особых случаев применения параметра регистрации маршрута дейтаграммы). Причина проста — поскольку каждый маршрутизатор ведет собственную таблицу маршрутизации и может без предупреждения изменить ее содержимое, заранее предсказать маршрут конкретной дейтаграммы нельзя. Поэтому нет никакой возможности определить путь, по которому прошла дейтаграмма до заданного маршрутизатора. Обнаружив проблему, маршрутизатор не сможет сообщить об этом всем промежуточным устройствам, через которые прошла дейтаграмма, поскольку их адреса неизвестны. Однако вместо того, чтобы просто “тихо” аннулировать дейтаграмму, маршрутизатор с помощью протокола ICMP информирует отправителя о возникшей проблеме. При этом подразумевается, что, получив уведомление, пользователь компьютера свяжется с сетевым администратором, и проблема будет установлена и устранена.

**Доставка ICMP-сообщений**

Для доставки ICMP-сообщений требуется выполнить процесс инкапсуляции два раза, как показано на рис. 12.1.

Рис. 12.1. Двухуровневая инкапсуляция ICMP-сообщений. Вначале ICMP-сообщение помещается в поле данных IP-дейтаграммы, а затем сама дейтаграмма помещается в поле данных фрейма для передачи по физической сети. Для идентификации ICMP-сообщения в поле типа протокола заголовка дейтаграммы помещается значение 1

ICMP-сообщение помещается в область данных IP-дейтаграммы, которая, в свою очередь, помещается в область данных сетевого фрейма. Процесс маршрутизации дейтаграмм с ICMP-сообщениями ничем не отличается от маршрутизации любых других дейтаграмм, содержащих пользовательские данные. Им не уделяется особое внимание, они не имеют также никакого приоритета перед обычными дейтаграммами. Сообщения об ошибках могут быть утеряны по дороге к отправителю дейтаграммы, их может также аннулировать один из промежуточных маршрутизаторов. Более того, в перегруженной сети сообщения об ошибках приводят к еще большему увеличению трафика. Однако следует отметить, что если нештатная ситуация возникает при доставке IP-дейтаграммы, содержащей ICMP-сообщение, ее обработка выполняется по особому алгоритму. Суть его состоит в том, что в сети не должны появляться сообщения об ошибках на сообщения об ошибках. Другими словами, ICMP-сообщения не генерируются в ответ на ошибки, вызванные доставкой дейтаграмм, содержащих ICMP-сообщения.

Однако следует иметь в виду, что, хотя ICMP-сообщения инкапсулируются в IP-дейтаграммы и посылаются с помощью протокола IP, протокол ICMP не относится к высокоуровневым сетевым протоколам, так как он является неотъемлемой частью протокола IP. Поскольку по пути до своего получателя ICMP-сообщения могут проходить по сетям разных типов, для их доставки нельзя использовать протоколы физического уровня. Поэтому и был выбран протокол IP.

**Формат ICMP-сообщений**

Каждый тип ICMP-сообщений имеет собственный формат, однако все они начинаются с трех одинаковых полей: 8-разрядного целого поля *типа* сообщения, идентифицирующего сообщение; 8-разрядного поля *кода* сообщения, в котором хранится дополнительная информация о сообщении; и 16-разрядного поля *контрольной* суммы. В протоколе ICMP используется такой же аддитивный алгоритм вычисления контрольной суммы, как и в протоколе IP, однако в данном случае вычисляется только контрольная сумма самого ICMP-сообщения.

При возникновении ошибочной ситуации в соответствующее ICMP-сообщение помещается заголовок дейтаграммы, вызвавшей проблему, и ее первые 64 бита данных. Это позволяет получателю более точно определить, какой из протоколов и какая из прикладных программ несет ответственность за возникновение проблемы. Как будет показано ниже, высокоуровневые протоколы, входящие в семейство протоколов TCP/IP, спроектированы так, чтобы вся ключевая информация содержалась в первых 64 битах их сообщений.

Поле типа ICMP-сообщения определяет его формат и содержимое. Типы ICMP-сообщений приведены в табл. 9.1. Форматы сообщений и их назначение будут описаны в следующих разделах.

Таблица 12.1. Типы ICMP-сообщений

| **Значение в поле типа** | **Описание ICMP-сообщения** |
| --- | --- |
| 0 | Ответ на запрос эха |
| 3 | Получатель недостижим |
| 4 | Подавление источника данных |
| 5 | Переадресация (изменение маршрута) |
| 8 | Запрос эха |
| 9 | Извещение о маршрутизаторе |
| 10 | Запрос на адрес маршрутизатора |
| 11 | Истекло время ожидания получения фрагментов или время жизни дейтаграммы |
| 12 | Ошибка в параметрах дейтаграммы |
| 13 | Запрос временной метки |
| 14 | Ответ, содержащий временную метку |
| 15 | Информационный запрос (устаревший) |
| 16 | Информационный ответ (устаревший) |
| 17 | Запрос маски адреса |
| 18 | Ответ, содержащий маску адреса |

**Проверка связи с получателем и его состояния (ping)**

В семействе протоколов TCP/IP предусмотрены специальные средства, облегчающие сетевым администраторам и пользователям поиск неисправностей в сети. Чаще всего для целей отладки используют два типа ICMP-сообщений: *запрос эха* и ответ на него. Любой узел сети или маршрутизатор может отправить заданному получателю ICMP-сообщение с запросом эха. Получив такой запрос, компьютер должен сформировать ответ и отослать его отправителю. Обычно в запрос включается необязательная область данных, содержимое которой переписывается в ответное сообщение.

Запрос эха и ответное сообщение на него могут использоваться для проверки возможности установки с получателем двухсторонней связи. Поскольку ICMP-сообщения пересылаются по сети в виде IP-дейтаграмм, получение ответа на запрос эха свидетельствует о том, что основные элементы системы транспортировки дейтаграмм работают нормально. Напомним их вкратце.

Во-первых, программное обеспечение, работающее на компьютере отправителя, должно корректно выполнять маршрутизацию дейтаграмм.

Во-вторых, все промежуточные маршрутизаторы, находящиеся по пути следования дейтаграммы от отправителя до конечного получателя, должны быть в рабочем состоянии и правильно пересылать дейтаграммы.

В-третьих, машина конечного получателя должна находиться в рабочем состоянии (по крайней мере отвечать на поступившие запросы), а ее программное обеспечение протокола IP и ICMP должно функционировать в штатном режиме.

И, наконец, в таблицах маршрутизации всех промежуточных устройств должна содержаться корректная информация.

В большинстве операционных систем для отправки запросов эха предусмотрена специальная команда, которая называется ping1. Усложненные версии этой команды посылают несколько последовательных ICMP-запросов, обрабатывают поступающие на них ответы и выдают статистику по прохождению и потере дейтаграмм. С помощью специальных параметров команды ping, пользователь может определить размер посылаемых данных и интервал времени между запросами. Более простые версии этой программы посылают только один ICMP-запрос и ожидают на него ответ.

Как заметил однажды Дэйв Миллс (Dave Mills), название *PING* является сокращением от *Packet InterNet Groper*, или *межсетевой пакетный тестер*.

**Форматы запроса на эхо и ответного сообщения**

На рис. 12.2 показан формат запроса на эхо и ответа на него.

Рис. 12.2. Формат запроса на эхо и ответного ICMP-сообщения

Область необязательных данных имеет переменную длину, в нее помещаются данные, которые должны быть возвращены отправителю. В ответном ICMP-сообщений всегда содержатся те же данные, которые были получены в первоначальном запросе эха. Поля идентификатора и порядкового номера предназначены для того, чтобы отправитель мог сопоставить запросы эха и получаемые на них ответы. В запросе эха значение поля “тип” устанавливается равным 8, а в ответе на него — 0.

**Извещение об отсутствии связи с получателем**

Если маршрутизатор не может переслать или доставить IP-дейтаграмму, он посылает ее отправителю уведомление о *недостижимости получателя*. Формат сообщения показан на рис. 12.3.

Рис. 12.3. Формат ICMP-сообщения о недостижимости получателя

В поле кода ICMP-сообщения рассматриваемого типа помещается целое число, позволяющее уточнить суть проблемы (табл. 12.2).

Таблица 12.2. Коды ICMP-сообщений о недостижимости получателя

| **Код** | **Описание** |
| --- | --- |
| 0 | Сеть недостижима |
| 1 | Узел сети недостижим |
| 2 | Протокол недоступен |
| 3 | Порт недоступен |
| 4 | Необходима фрагментация, однако установлен бит ее запрета |
| 5 | Ошибка маршрутизации от источника |
| 6 | Сеть получателя неизвестна |
| 7 | Узел получателя неизвестен |
| 8 | Узел отправителя изолирован |
| 9 | Связь с сетью получателя запрещена на административном уровне |
| 10 | Связь с узлом получателя запрещена на административном уровне |
| 11 | Сеть недостижима для данного типа обслуживания |
| 12 | Узел недостижим для данного типа обслуживания |

Хотя в протоколе IP применен механизм, не гарантирующий доставку пакетов, дейтаграммы в нем не теряются бесследно. Как только маршрутизатор обнаруживает, что не сможет переслать или доставить дейтаграмму получателю, он посылает отправителю уведомление о недостижимости получателя и только тогда *аннулирует* (т.е. уничтожает) дейтаграмму. Ошибки, при которых сеть получателя оказывается недостижимой, обычно свидетельствуют об отказе одного из промежуточных маршрутизаторов. Если же недостижим узел сети, это свидетельствует о проблемах с доставкой дейтаграммы. Поскольку в ICMP-сообщений об ошибке содержится фрагмент начала дейтаграммы, которая вызвала проблему, отправитель всегда сможет точно определить, какой из адресов получателя недостижим.

Связь с узлом получателя может отсутствовать по нескольким причинам. Одна из них — отказ оборудования. Другая — отправитель указал несуществующий адрес получателя. Третья (в редких случаях) — маршрутизатор не располагает данными о маршруте к сети получателя. Обратите внимание, что маршрутизаторы сообщают только о тех нештатных ситуациях, которые они обнаружили, поскольку другие проблемы доставки им могут быть не известны. Например, если машина получателя подключена к сети Ethernet, то ее сетевое оборудование никак не сообщает о факте получения дейтаграммы. Более того, маршрутизатор может продолжать некоторое время посылать пакеты этой машине даже после ее внезапного отключения от сети (например, в случае пропадания напряжения питания или обрыва кабеля). При этом маршрутизатор так и не узнает, доставлены ли пакеты. Можно сделать следующий вывод.

Несмотря на то, что при невозможности переслать или доставить дейтаграмму получателю, маршрутизатор посылает ее отправителю соответствующее уведомление, сам маршрутизатор не может выявить все возможные проблемы с ее доставкой.

Смысл сообщений об ошибке, связанной с недоступностью протокола и порта, станет понятен чуть позже, когда мы рассмотрим, как в протоколах высокого уровня используются абстрактные точки назначения, называемые *портами (ports)*. Смысл большинства остальных сообщений об ошибках очевиден из их названия. Если в дейтаграмме указан параметр маршрутизации от источника, и в нем задан некорректный маршрут следования, то маршрутизатор может прислать извещение об ошибке *маршрутизации от источника*. Если для дальнейшей передачи дейтаграммы маршрутизатор должен ее фрагментировать, а в ее заголовке установлен бит запрета фрагментации, то отправителю посылается соответствующее уведомление.

**Перегрузка сети и управление потоком дейтаграмм**

Поскольку в протоколе IP для обмена данными не требуется предварительная установка соединения с получателем, маршрутизатор не может зарезервировать память или другие сетевые ресурсы до получения дейтаграммы. В результате входящий поток данных может привести к тому, что все ресурсы маршрутизатора будут исчерпаны, т.е. произойдет его *перегрузка*. Перегрузка может произойти по двум совершенно разным причинам.

Во-первых, современные высокоскоростные компьютеры способны генерировать поток данных, во много раз превышающий пропускную способность канала связи. Например, представьте себе, что по объединенной сети должен передаваться трафик от суперкомпьютера. Даже если предположить, что сам суперкомпьютер подключен к высокоскоростной локальной сети, в конечном итоге данные попадут в низкоскоростную глобальную сеть. Очевидно, что в этом случае возникнет перегрузка маршрутизатора, соединяющего локальную сеть с глобальной, поскольку дейтаграммы будут прибывать быстрее, чем они могут быть отправлены.

Во-вторых, перегрузка маршрутизатора может возникнуть в том случае, если через него будут проходить дейтаграммы, посланные за короткий промежуток времени с нескольких компьютеров. При этом поток данных от одного компьютера чаще всего не вызывает проблем с маршрутизатором.

Если дейтаграммы прибывают быстрее, чем может обработать узел сети или маршрутизатор, они временно помещаются в очередь в памяти компьютера. Если дейтаграммы являются частью кратковременного пикового трафика, подобная буферизация позволяет решить проблему перегрузки маршрутизатора. Однако если поток данных продолжает поступать, вся доступная на узле сети или маршрутизаторе память рано или поздно исчерпается. При этом все “лишние” дейтаграммы будут попросту утеряны. При возникновении подобной ситуации маршрутизатор шлет отправителю ICMP-сообщение о *подавлении источника данных (source quench)*. Этим сообщением маршрутизатор “просит” отправителя уменьшить генерируемый им поток данных. Обычно при перегрузке маршрутизатора отправителю посылается сообщение о подавлении источника данных для каждой утерянной дейтаграммы.

В современных маршрутизаторах применяются более сложные алгоритмы обработки ситуаций перегрузки. В частности, в некоторых маршрутизаторах измеряется скорость входящего потока данных, и если объем трафика превышает заданный уровень, узлам сети, генерирующим максимальный трафик, посылается сообщение о подавлении источника данных. В других системах предпринимается попытка избежать перегрузки маршрутизатора в целом. При этом сообщения о подавлении источника данных рассылаются, как только размеры очереди становятся угрожающими, но до возникновения перегрузки устройства.

Следует заметить, что не существует ICMP-сообщений, отменяющих подавление источника данных. Узел сети должен снижать уровень трафика, отсылаемый в заданном направлении, до тех пор, пока ему не перестанут приходить сообщения о подавлении источника данных, работающего в этом направлении. После этого он может постепенно увеличивать трафик до тех пор, пока снова не начнут приходить сообщения о подавлении источника данных.

**Формат сообщения о подавлении источника данных**

Формат ICMP-сообщения о подавлении источника данных ничем не отличается от рассмотренного выше сообщения о недостижимости получателя. Как и в большинстве других ICMP-сообщений, уведомляющих отправителя об ошибке, в таком сообщении также содержится начальный фрагмент дейтаграммы, вызвавшей перегрузку маршрутизатора (рис. 12.4).

Рис. 12.4. Формат ICS1P сообщения о подавлении источника данных. В случае перегрузки маршрутизатор шлет такое сообщение отправителю каждой утерянной дейтаграммы, помещая в него ее начальный фрагмент

**Запрос на изменение маршрута следования дейтаграмм от маршрутизатора**

Обычно содержимое таблиц маршрутизации в объединенной сети не изменяется достаточно продолжительное время. Их инициализация происходит при начальной загрузке компьютера путем считывания содержимого специальных файлов конфигурации. В штатном режиме системные администраторы сравнительно редко вносят изменения в систему маршрутизации дейтаграмм. Однако при изменении топологии сети содержимое таблиц маршрутизации как на самом маршрутизаторе, так и на отдельных узлах сети может не соответствовать действительности. Подобные изменения могут носить как временный (например, в случае выхода из строя одного из маршрутизаторов), так и постоянный характер (скажем, при подключении к объединенной сети новой локальной сети).

Как будет показано в следующих разделах, маршрутизаторы периодически обмениваются между собой маршрутными данными. Это позволяет им учитывать все произошедшие в объединенной сети изменения и согласовывать содержимое таблиц маршрутизации.

Таким образом, вывод такой: при выполнении маршрутизации подразумевается, что устройства маршрутизации владеют корректной информацией. Для узлов сети нужны только минимально необходимые данные. Информация о новых маршрутах приходит от маршрутизаторов.

Соблюдение этих правил позволяет избежать дублирования маршрутных данных в файлах конфигурации узлов сети. При инициализации таблицы маршрутизации в ней указывается только минимальная информация, необходимая для начала процесса обмена данными по сети (т.е. адрес ближайшего маршрутизатора). Таким образом, перед началом работы в таблице маршрутизации узла сети указана минимально необходимая информация, а ее изменение выполняется маршрутизатором. В некоторых случаях маршрутизатор может обнаружить, что узел сети использует неоптимальный маршрут. Тогда узлу сети посылается ICMP-сообщение о *переадресации (redirect)*, предписывающее изменить маршрут. Кроме того, маршрутизатор пересылает полученную дейтагрaмму конечному получателю.

Преимущество системы переадресации протокола ICMP — в ее простоте. При начальной загрузке узлу сети достаточно сообщить только адрес одного маршрутизатора, находящегося в локальной сети. Как только узел сети отправит маршрутизатору дейтаграмму, для которой существует другой, оптимальный маршрут, узлу будет послано соответствующее ICMP-сообщение. Таким образом, удается сохранить минимальный размер таблицы маршрутизации и при этом поместить в нее информацию об оптимальных маршрутах ко всем возможным получателям.

Однако сообщения о переадресации не могут в общем случае решить проблему распространения информации о маршрутах дейтаграмм. Причина состоит в том, что маршрутизатор может отправлять такие сообщения только тем узлам, которые находятся в одной с ним физической сети. Эта проблема проиллюстрирована на рис. 12.5.

Рис. 12.5. ICMP-сообщения не могут решить проблему обмена информацией между маршрутизаторами о путях следования дейтаграмм. В представленном примере устройство R5 не сможет отправить ICMP-сообщение о переадресации устройству R1, чтобы сократить путь следования дейтаграмм от узла S к узлу D

Предположим, что машина S посылает дейтаграммы машине D. Предположим также, что маршрутизатор R1, некорректно выполняет маршрутизацию и вместо устройства R4, пересылает дейтаграммы устройству R2 (т.е. R1 выбирает более длинный маршрут). После того как устройство R5 получит дейтаграмму, оно не сможет отправить ICMP-сообщение о переадресации устройству R1, поскольку адрес последнего ему неизвестен. Проблема распространения информации о маршрутах дейтаграмм по нескольким сетям будет описана в следующих разделах.

Кроме трех стандартных полей (типа, кода и контрольной суммы), в каждом сообщении о переадресации указывается 32-разрядное поле IP-адреса маршрутизатора, а также начальный фрагмент дейтаграммы (рис. 12.6).

Рис. 12.6. Формат ICMP-сообщения о переадресации

В поле IP-адреса маршрутизатора указывается адрес устройства, которому узел сети должен переслать дейтаграмму при отправке ее конечному получателю, адрес которого указан в заголовке. В ICMP-сообщение о переадресации включается также заголовок дейтаграммы, посланной по неоптимальному маршруту, и следующие за ним 64 бита данных. Таким образом, получив ICMP-сообщение о переадресации, узел сети анализирует начальный фрагмент содержащейся в нем дейтаграммы и определяет адрес ее конечного получателя. В поле кода ICMP-сообщения указывается число, предназначенное для определения способа интерпретации адреса получателя (табл. 12.3).

Таблица 12.3. Значения кодов интерпретации адреса получателя

| **Код** | **Описание** |
| --- | --- |
| 0 | Переадресовывать дейтаграммы для указанной сети (устарело) |
| 1 | Переадресовывать дейтаграммы для указанного узла сети |
| 2 | Переадресовывать дейтаграммы с указанным типом обслуживания\* и для указанной сети |
| 3 | Переадресовывать дейтаграммы с указанным типом обслуживания и для указанного узла сети |

Напомним, что в каждом заголовке IP-дейтаграммы существует поле, в котором указывается тип обслуживания дейтаграммы. Значение этого поля интерпретируется в процессе маршрутизации.

Итак, можно сказать, что маршрутизаторы отправляют ICMP-сообщения о переадресации только узлам сети, а не другим маршрутизаторам.

В следующих разделах будут описаны специальные протоколы, с помощью которых маршрутизаторы обмениваются информацией о маршрутах дейтаграмм.

**Обнаружение замкнутых и слишком длинных маршрутов**

Поскольку маршрутизаторы объединенной сети определяют адрес ближайшей точки перехода на основании данных, находящихся в локальных таблицах маршрутизации, ошибки в них могут привести к *зацикливанию маршрутов (routing cycle)* следования дейтаграмм в направлении некоторых получателей D. Зацикливание маршрутов может произойти на участке сети между двумя и более маршрутизаторами. При этом маршрутизаторы будут передавать дейтаграммы, следующие в направлении получателей D, друг другу (или по кругу, если в цикл вовлечено больше двух маршрутизаторов). Если дейтаграмма попадает в замкнутый цикл маршрутизации, она будет передаваться по нему бесконечно.

Как было отмечено в предыдущих разделах, для предотвращения явления зацикливания в объединенной сети на основе протокола TCP/IP в заголовке дейтаграмм предусмотрено специальное поле времени жизни, которое иногда называют *счетчиком nepexoдов (hop count)*. Обработав дейтаграмму, маршрутизатор уменьшает на единицу значение счетчика ее времени жизни. Когда значение счетчика становится равным нулю, маршрутизатор удаляет дейтаграмму из сети.

Маршрутизатор может удалить дейтаграмму из сети по двум причинам: при достижении счетчиком времени жизни нулевого значения и при истечении интервала времени, установленного для ожидания получения всех фрагментов дейтаграммы. В обоих случаях маршрутизатор посылает отправителю дейтаграммы специальное ICMP-сообщение об *истечении времени ожидания*, формат которого показан на рис. 12.7.

Рис. 12.7. Формат ICMP-сообщения об истечении времени ожидания. Маршрутизатор отправляет это сообщение в том случае, когда он удалил дейтаграмму из сети, поскольку счетчик ее времени жизни достиг нулевого значения, или истекло время ожидания при получении фрагментов дейтаграммы

В поле кода ICMP-сообщения об истечении времени ожидания помещается значение 0 или 1, которое помогает отправителю установить причину ошибки (табл. 9.4).

Таблица 12.4. Значения кодов ошибки сообщения об истечении времени ожидания

| **Код** | **Описание** |
| --- | --- |
| 0 | Исчерпано значение счетчика времени жизни |
| 1 | Истекло время ожидания при получении фрагментов дейтаграммы |

Напомним, что процесс сборки фрагментов выполняется конечным получателем дейтаграммы. При поступлении первого фрагмента дейтаграммы получатель запускает специальный таймер. Если значение таймера истечет до того, как будут получены все фрагменты дейтаграммы, считается, что возникла ошибочная ситуация. В этом случае отправителю дейтаграммы посылается соответствующее ICMP-сообщение, в поле кода которого помещается значение 1. Такое сообщение отправляется каждый раз при возникновении подобной ошибки.

**Уведомление об остальных проблемах**

Если маршрутизатор или узел сети обнаруживают проблему, связанную с доставкой дейтаграммы, не описанную в предыдущих разделах (например, если у дейтаграммы оказался некорректный заголовок), то ее отправителю посылается сообщение об *ошибке в параметрах*. Одна из возможных причин возникновения подобной ошибки — указание некорректных параметров дейтаграммы. Сообщение, формат которого показан на рис. 12.8, посылается отправителю дейтаграммы только в том случае, если ошибка настолько серьезна, что дейтаграмма должна быть удалена из сети.

Рис. 12.8. Формат ICMP-сообщения об ошибке в параметрах дейтаграммы. Подобное сообщение посылается отправителю только в случае удаления дейтаграммы из сети

Чтобы помочь отправителю дейтаграммы конкретизировать проблему, в заголовок ICMP-сообщения об ошибке было введено специальное поле указателя. В него помещается смещение октета исходной дейтаграммы, значение которого вызвало ошибку. Значение поля кода, равное 1, указывает на то, что отсутствует нужный параметр дейтаграммы (например, параметр безопасности при работе в защищенной среде). Следует заметить, что значение поля указателя не используется в том случае, если значение поля кода равно 1.

**Синхронизация часов и оценка времени передачи**

Хотя все компьютеры в объединенной сети могут взаимодействовать между собой, обычно они функционируют независимо друг от друга. При этом текущее время на каждом компьютере может быть разным. Подобная разница в представлении времени может сбить с толку пользователей распределенных программных систем. Поэтому для синхронизации часов в семейство протоколов TCP/IP включено несколько протоколов. Один из простейших способов синхронизации времени — запросить текущее время у другого компьютера сети с помощью ICMP-сообщения. При этом машина отправителя посылает другому компьютеру специальное ICMP-сообщение, которое запрашивает у него *временную метку (timestamp request)*. В ответе на запрос другой компьютер должен прислать отправителю значение своего текущего времени, помещенное в специальное ICMP-сообщение. Форматы запроса временной метки и ответа на него приведены на рис. 12.9.

Рис. 12.9. Формат ICMP-сообщения, запрашивающего временную метку, и ответ на него

В поле типа рассматриваемого ICMP-сообщения помещается значение 13 (для запроса временной метки) или значение 14 (для ответного сообщения). Поля идентификатора и порядкового номера используются отправителем сообщения для сопоставления запросов и полученных на них ответов. В оставшихся трех полях указываются значения текущего времени, выраженные в миллисекундах, прошедших после полуночи по всемирному времени (Universal Time)1. Прежде чем послать пакет получателю, отправитель помещает в поле *времени отправления* запроса свое текущее время. При получении пакета компьютер помещает в поле *времени получения* запроса свое текущее время. И непосредственно при посылке ответа отправителю в поле *времени отправления* запроса снова помещается текущее время.

Описанные выше три временных метки используются узлами сети для вычисления задержки распространения пакета по сети, что позволяет точно синхронизировать их часы. Поскольку в ответном сообщении содержится время посылки запроса, отправитель может легко вычислить общее время, затрачиваемое на передачу запроса получателю, формирования ответа на него и обратную пересылку. Так как в ответном сообщении содержится как время получения запроса удаленным компьютером, так и время отправки ответа на него, отправитель может легко определить время передачи пакета по сети и исходя из него оценить разницу во времени между локальным и удаленным компьютерами.

На практике часто бывает сложно оценить время передачи запроса по сети и возвращения на него ответа, что существенно снижает эффективность использования ICMP-сообщений, запрашивающих временные метки. Для точного определения времени прохождения пакетов можно выполнить несколько измерений и усреднить полученный результат. Однако задержка прохождения пакетов между двумя машинами, подключенными к большой объединенной сети, часто колеблется в очень широких пределах, причем в течение короткого промежутка времени. Более того, поскольку протокол IP не гарантирует доставку пакетов, при прохождении по сети дейтаграммы могут быть аннулированы, задержаны или доставлены в порядке, отличном от исходного. Таким образом, проведение нескольких измерений еще не гарантирует точности полученного результата. Для правильной оценки результатов измерений необходимо прибегнуть к сложным методам статистического анализа.

1. Всемирное время соответствует времени нулевого (Гринвичского) меридиана.

**Информационные запросы и ответы**

*Информационные запросы и ответы* протокола ICMP (типы сообщений 15 и 16) считаются устаревшими и не должны использоваться в современном программном обеспечении. Они были разработаны для того, чтобы узел сети мог определить свой IP-адрес при начальной загрузке. В дальнейшем для этих целей были разработаны протоколы RARP.

**Определение маски подсети**

Необходимо просто понять, что, если для адресации узла используется концепция подсети, часть битов поля hostid IP-адреса идентифицирует физическую сеть. Таким образом, чтобы можно было использовать адресацию подсетей, узел сети должен “знать”, какая часть битов его 32-разрядного IP-адреса идентифицирует физическую сеть, а какая — узел в этой сети. Эту информацию кодируют в виде 32-разрядного числа, называемого *маской подсети (subnet mask)*.

Чтобы определить маску подсети, которая используется в некоторой локальной сети, компьютер должен послать запрос (ICMP-сообщение) соответствующему маршрутизатору и получить ответное сообщение, содержащее маску. При отправке запроса компьютер может послать сообщение либо напрямую маршрутизатору, если он знает его IP-адрес, либо послать это же сообщение в широковещательном режиме, если адрес маршрутизатора неизвестен. Формат запроса маски подсети показан на рис. 12.10.

Рис. 12.10. Форматы ICMP-запроса маски подсети и ответного сообщения. Обычно запрос на маску подсети посылается в широковещательном режиме, поскольку адрес маршрутизатора, который должен прислать ответ, еще не известен

В поле типа ICMP-сообщения указывается код 17, если это запрос на маску подсети, либо код 18, если это ответное сообщение. Маска адреса подсети помещается в соответствующее поле ответного сообщения. Назначение полей идентификатора и порядкового номера традиционное. Они позволяют отправителю запроса сопоставить присланный на него ответ.

**Поиск маршрутизатора**

Для того чтобы после выполнения начальной загрузки компьютер смог посылать дейтаграммы получателям из других сетей, он должен знать адрес хотя бы одного маршрутизатора, расположенного в локальной сети, к которой он подключен. В протоколе ICMP предусмотрен специальный механизм *поиска маршрутизатора*, который позволяет узлу сети обнаружить адрес ближайшего к нему маршрутизатора.

Следует заметить, что этот механизм поиска адреса маршрутизатора — не единственный. Существуют два других протокола — ВООТР и DHCP, каждый из которых позволяет узлу сети получить во время начальной загрузки адрес стандартного маршрутизатора, а также другую необходимую информацию. Однако следует отметить, что протоколы ВООТР и DHCP имеют серьезный недостаток: они возвращают из базы данных только ту информацию, которая занесена туда вручную сетевым администратором. А это значит, что информация в базе данных не может меняться достаточно оперативно.

Естественно, что в некоторых случаях адрес маршрутизатора можно прописать статично в один из файлов конфигурации, и это будет работать очень хорошо. В качестве примера рассмотрим сеть, которая имеет выход во внешний мир через единственный маршрутизатор. В этом случае узлу сети нет необходимости динамически определять адреса маршрутизаторов или на ходу изменять маршруты дейтаграмм. Хотя, если в сети для выхода во внешний мир предусмотрено несколько маршрутизаторов, то полученный узлом во время начальной загрузки стандартный маршрут может оказаться недействующим в случае, если маршрутизатор, принятый по умолчанию, выйдет из строя. Более того, об отказе маршрутизатора узлу сети ничего не будет известно.

Механизм поиска маршрутизаторов протокола ICMP имеет два преимущества. Во-первых, он позволяет узлу сети получать информацию непосредственно от самого маршрутизатора, а не из статичных файлов конфигурации при начальной загрузке. Во-вторых, в нем используется технология систем с *неустойчивым состоянием (soft state)* на основе таймеров. Это позволяет предотвратить получение узлом сети неправильной информации при выходе маршрутизатора из строя. Маршрутизаторы периодически рассылают узлам сети информацию о себе. Если эта информация не будет получена в течение установленного интервала времени, узел сети считает, что данный маршрутизатор более недоступен. Формат ICMP-сообщения, которое присылают маршрутизаторы, показан на рис. 12.11.

Рис. 12.11. Формат ICMP-сообщения, используемого в протоколе IPv4 для извещения узлов сети об адресах маршрутизаторов. Такие сообщения периодически рассылаются самими маршрутизаторами

Кроме уже знакомых нам полей типа, кода и контрольной суммы, в рассматриваемом ICMP-сообщений находятся еще три новых поля. Количество адресов маршрутизаторов (обычно 1), находящихся в сообщении, указывается в одноименном поле. В поле размера адреса указывается количество блоков по 32 бита, которое занимает поле адреса. Для протокола IPv4 в этом поле указывается значение *1*. В поле времени жизни указывается время в секундах, в течение которого узел сети может использовать полученную информацию о маршрутизаторах. Стандартное значение времени жизни равно 30 мин., а рассылка информации о маршрутизаторах по умолчанию выполняется каждые 10 мин. Таким образом, даже если одно из ICMP-сообщений об адресе маршрутизатора не будет получено узлом сети, информация об адресе маршрутизатора останется действительной еще по меньшей мере 20 мин.

В остальной части сообщения находятся пары чисел (их количество указывается в поле *Количество адресов*), первое из которых соответствует IP-адресу маршрутизатора, а второе определяет приоритет данного маршрутизатора (или степень предпочтения маршрута, проходящего через него). Приоритет маршрутизатора выражается двоичным целым числом, представленным в дополнительном коде. Узел сети должен выбирать маршруты с максимальным значением приоритета.

Если в сети, к которой подключен маршрутизатор, поддерживается режим многоадресатной передачи данных, ICMP-сообщения, содержащие адреса маршрутизаторов, должны рассылаться по всем многоадресатным адресам системы (т.е. 224.0.0.1). Если же этот режим не поддерживается, маршрутизатор должен посылать сообщения по ограниченному широковещательному адресу (т.е. адресу, состоящему из всех единиц). Естественно, что узел сети не должен посылать ICMP-сообщения, содержащие адреса маршрутизаторов.

**Запрос адреса маршрутизатора**

Несмотря на то, что разработчики предусмотрели возможность изменения интервала времени между рассылками информации о маршрутизаторе, по умолчанию был выбран интервал, равный 10 мин. Эта величина является своего рода компромиссом между возможностью быстрого обнаружения отказов в сети и снижением накладных расходов, связанных с увеличением дополнительного трафика. Уменьшение значения интервала времени между рассылками ICMP-сообщений позволит быстрее обнаружить вышедший из строя маршрутизатор, однако это приведет к увеличению трафика в сети. В то же время увеличение интервала уменьшит накладные расходы, однако скорость обнаружения отказов в сети при этом упадет. Одной из задач, которую ставили перед собой создатели протокола ICMP, было согласование работы нескольких маршрутизаторов, находящихся в одной локальной сети.

С точки зрения узла сети, выбранное по умолчанию значение интервала времени между рассылками информации о маршрутизаторе имеет существенный недостаток. Дело и том, что во время начальной загрузки неразумно заставлять узел сети ожидать прихода извещения об адресе маршрутизатора, поскольку интервал ожидания может составлять несколько минут. Поэтому, чтобы не допустить подобных задержек, в протоколе ICMP предусмотрено специальное сообщение, позволяющее узлу сети *немедленно зanpocumь* информацию о маршрутизаторе. Формат этого сообщения показан на рис. 12.12.

Рис. 12.12. Формат ICMP-запроса данных о маршрутизаторе. Узел сети посылает этот зaпpoc во время начальной загрузки для того, чтобы немедленно узнать адрес ближайшего маршрутизатора

**Проверка знаний: Протокол IP. Обработка ошибок и управляющие сообщения (ICMP)**

**Задание 1**

Какое поле не существует в ICMP-сообщении, запрашивающем временную метку, и ответ на него?

время получения запроса

время отправления запроса

время отправления ответа

время получения ответа

**Задание 2**

Какие преимущества имеет механизм поиска маршрутизаторов протокола ICMP? Выберите 2 правильных ответа.

поскольку он функционирует на самом низком уровне, использующая его прикладная программа должна напрямую взаимодействовать с сетевым оборудованием

он позволяет узлу сети получать информацию непосредственно от самого маршрутизатора

в нем используется технология систем с неустойчивым состоянием (soft state) на основе таймеров

он позволяет узлу сети получать информацию из статичных файлов конфигурации при начальной загрузке

**Задание 3**

Для чего используется протокол ICMP?

Протокол ICMP анализирует нагрузку на маршрутизаторы и посылает им управляющие сообщения для устранения неполадок.

Протокол ICMP защищает маршрутизаторы от ошибок и перегрузок.

Протокол ICMP позволяет маршрутизаторам отправлять другим маршрутизаторам или узлам сети сообщения об ошибках или управляющие сообщения.

**Задание 4**

Какое ICMP-сообщение посылает маршрутизатор, если он удаляет дейтаграмму из сети?

ICMP-сообщение о подавлении источника данных (source quench)

ICMP-сообщение переадресация (изменение маршрута)

ICMP-сообщение об истечении времени ожидания

**Задание 5**

Какой интервал времени между рассылками информации о маршрутизаторе по умолчанию?

5 минут

15 минут

10 минут

**Задание 6**

Может ли маршрутизатор, при невозможности доставить или переслать дейтаграмму получателю, сам выявить все возможные проблемы с ее доставкой?

не может

может

**Задание 7**

Какое ICMP-сообщение посылает маршрутизатор, если дейтаграммы прибывают быстрее, чем он может обработать?

ICMP-сообщение об истечении времени ожидания

ICMP-сообщение переадресация (изменение маршрута)

ICMP-сообщение о подавлении источника данных (source quench)

**Задание 8**

С каких полей начинаются все ICMP-сообщения? Выберите 3 правильных варианта

8-разрядного поля контрольной суммы

16-разрядного поля контрольной суммы

8-разрядного поля кода сообщения

16-разрядного поля кода сообщения

8-разрядного целого поля типа сообщения

32-разрядного целого поля типа сообщения

**Задание 9**

Сколько раз необходимо выполнить процесс инкапсуляции для доставки ICMP-сообщений?

2 раза

1 раз

3 раза

**Задание 10**

Какое ICMP-сообщение посылает маршрутизатор, если он обнаруживает, что узел сети использует не оптимальный маршрут?

ICMP-сообщение об истечении времени ожидания

ICMP-сообщение переадресация (изменение маршрута)

ICMP-сообщение о подавлении источника данных (source quench)

**Задание 11**

Какие ICMP-сообщения чаще всего используются для целей отладки?

запрос эха и ответ на него

запрос адресной маски

запрос хост

**Передача пользовательских дейтаграмм (UDP)**

В предыдущем разделе был рассмотрен способ передачи IP-дейтаграмм через объединенную сеть TCP/IP. Напомним, что маршрутизация каждой дейтаграммы выполняется на основе IP-адреса конечного получателя. С точки зрения протокола IP, этот адрес определяет только компьютер пользователя в сети. При этом не уточняется, какому именно пользователю или прикладной программе следует доставить дейтаграмму.

В этом разделе продолжается изучение семейства протоколов TCP/IP. Будет рассмотрен механизм, позволяющий получателю различать входящие дейтаграммы и определять, какой из прикладных программ они адресованы. Этот же механизм позволяет нескольким программам отправлять и получать дейтаграммы независимо друг от друга.

* [Идентификация конечного получателя](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690575)
* [Протокол передачи пользовательских дейтаграмм (UDP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690576)
* [Формат UDP-сообщения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690577)
* [Псевдозаголовок пользовательской дейтаграммы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690578)
* [Инкапсуляция и разделение на уровни](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690579)
* [Вычисление контрольной суммы UDP-дейтаграммы при разделении на уровни](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69057a)
* [Мультиплексирование и демультиплексирование UDP-дейтаграмм с помощью портов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69057b)
* [Зарезервированные и свободные номера портов UDP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69057c)
* [Проверка знаний: Передача пользовательских дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69057d)

**Идентификация конечного получателя**

Операционные системы большинства компьютеров поддерживают так называемый *мультипрограммный (multiprogramming)* режим работы, при котором обеспечивается одновременное и независимое выполнение нескольких пользовательских программ. В различных операционных системах выполняющиеся программы могут называться по-разному: *процесс (process), задача (task), приложение (application program)* или *пользовательский процесс (user level process)*. При этом сама операционная система называется *многозадачной (multitasking)*. Исходя из этого вполне правомерно сказать, что конечным получателем дейтаграмм является процесс, запущенный на узле сети с указанным IP-адресом. Однако подобное утверждение может ввести в заблуждение.

Во-первых, поскольку процессы создаются и уничтожаются динамически, отправитель редко имеет достаточное количество информации о процессе, запущенном на другой машине.

Во-вторых, все процессы, принимающие дейтаграммы на конкретном компьютере, могут быть заменены без уведомления отправителей. Например, перезагрузка компьютера вызывает замену всех запущенных на нем процессов, однако стоит ли об этом уведомлять всех потенциальных отправителей дейтаграмм?

В-третьих, получатель дейтаграмм должен идентифицироваться по выполняемым им функциям, а не по процессу, в котором эти функции реализованы. Другими словами, отправитель должен иметь возможность подключиться к файловому серверу, не выясняя, в каком именно процессе на машине получателя реализованы функции файлового сервера.

Кроме того, в системах, позволяющих в одном процессе обрабатывать две и более функций, необходимо обеспечить для процесса возможность определять, какая именно функция запрошена отправителем.

Из всего сказанного уже, должно быть, понятно, что процессы нельзя рассматривать в качестве конечного получателя дейтаграмм. Необходимо сделать так, чтобы на каждой машине имелся набор абстрактных получателей, называемых *портами протокола (protocol ports)*. Каждому порту протокола поставлено в соответствие целое положительное число. При этом в локальной операционной системе должны быть предусмотрены средства идентификации портов и доступа к ним.

В большинстве операционных систем используется синхронный доступ к портам. С точки зрения процесса это означает, что на время доступа к порту, операционная система приостанавливает его выполнение. Например, если процесс попытается считать данные из порта до того, как они будут получены, операционная система приостановит (заблокирует) его до поступления данных. Когда данные будут получены, операционная система передаст их процессу и возобновит его выполнение. Обычно в портах используется *буферизация данных*, поэтому данные не будут потеряны, даже если поступят раньше, чем процесс сможет их принять. При буферизации протокольное программное обеспечение, находящееся внутри операционной системы помещает полученные для определенного порта пакеты в очередь конечного размера, откуда процесс сможет их извлечь.

Для связи с внешним портом отправителю необходимо знать IP-адрес машины получатели и номер порта на этой машине. В каждом сообщении должен быть указан *номер порта получателя*, находящегося на машине, которой послано сообщение, а также *номер порта отправителя*, которому должно быть адресовано ответное сообщение. Это позволяет любому процессу получать сообщение и отвечать отправителю.

**Протокол передачи пользовательских дейтаграмм (UDP)**

В семействе протоколов TCP/IP *протокол UDP* (*User Datagram Protocol*) обеспечивает основной механизм передачи дейтаграмм между двумя прикладными программами. Выбор нужной программы на компьютере получателя в протоколе UDP выполняется по номеру порта. Поэтому, кроме передаваемых данных, в каждом UDP-сообщении содержатся номера портов получателя и отправителя. То есть для программ протокола UDP, запушенных на компьютере конечного получателя, обеспечивается механизм доставки сообщений до нужной прикладной программы, а для самой прикладной программы — возможность ответить отправителю.

Для транспортировки сообщений от одной машины к другой в протоколе UDP используется протокол нижнего уровня — IP. Поэтому для протокола UDP (так же, как и для IP) характерна ненадежная доставка дейтаграмм без установки соединения с получателем. В нем *не* используется механизм подтверждения доставки сообщений. В протоколе UDP не отслеживается порядок получения пакетов, а также не обеспечивается обратная связь для управления скоростью информационного потока между машинами. В результате сообщения UDP могут быть утеряны, переданы повторно или получены в неправильном порядке. Более того, в течение некоторого промежутка времени может быть получено больше пакетов, чем получатель сможет их обработать. Таким образом, можно подвести итог.

Протокол передачи пользовательских дейтаграмм (User Datagram Protocol, или UDP) обеспечивает ненадежную доставку сообщений без установки соединения с получателем. При этом для транспортировки сообщений между машинами используется протокол IP. В отличие от IP, в протоколе UDP предусмотрена возможность выбора заданного получателя на конкретной машине.

При использовании протокола UDP за надежность доставки отвечает прикладная программа. Это подразумевает обработку ситуаций потери сообщений, их дублирования, возникновения задержек, нарушения порядка, а также потери связи с получателем. К сожалению, программисты, разрабатывающие прикладные программы, часто не уделяют должного внимания перечисленным проблемам. Более того, поскольку для тестирования сетевых программ, как правило, используется локальная сеть, отличающаяся высокой надежностью и малым временем задержки при передаче пакетов, на этапе тестирования все потенциальные проблемы могут быть не выявлены. Поэтому, даже если программа, использующая протокол UDP, хорошо работает в локальной сети, она может некорректно работать в большой объединенной сети на основе протокола TCP/IP.

**Формат UDP-сообщения**

Любое сообщение протокола UDP называется *пользовательской дейтаграммой (user datagram)*. Пользовательская дейтаграмма состоит из двух частей: UDP-заголовка и области данных. Как показано на рис. 13.1 заголовок разделен на четыре 16-битовых поля, в которых указываются номера портов отправителя и получателя, длина сообщения и контрольная сумма UDP-сообщения.

Рис. 13.1. Формат пользовательской дейтаграммы

Как видно из рис. 13.1, в первых двух 16-битовых полях заголовка пользовательской дейтаграммы указываются *номера портов отправителя и получателя*. Последний используется программой протокола UDP для распределения поступивших дейтаграмм между ожидающими их процессами. Номер *порта отправителя* указывать необязательно. Если порт отправителя не указан, в этом поле должно содержаться нулевое значение. В противном случае в этом поле задается номер порта, по которому получатель может послать ответ отправителю.

В поле *длины UDP-сообщения* указывается размер дейтаграммы в октетах, включая заголовок и пользовательские данные. Поэтому, минимальный размер UDP-дейтаграммы равняется восьми октетам (в том случае, если в сообщении отсутствуют пользовательские данные).

Поле *контрольной суммы* является необязательным и может вообще не использоваться. Если его значение равно нулю, значит, контрольная сумма пользовательской дейтаграммы не вычислялась. По замыслу разработчиков такой подход позволяет передавать UDP-дейтаграммы через высоконадежные локальные сети, с минимальными накладными расходами, связанными с вычислением и последующей проверкой контрольных сумм. Напомним, что в протоколе IP контрольная сумма вычисляется только для заголовка дейтаграммы и не вычисляется для поля данных. Таким образом, по значению контрольной суммы UDP-дейтаграммы можно судить о том, что в процессе доставки данные не были изменены.

Новички часто удивляются, узнав о том, что вычисленная контрольная сумма UDP-сообщения может равняться нулю. В этом нет ничего странного, поскольку в протоколе UDP используется тот же алгоритм вычисления контрольной суммы, что и в протоколе IP. То есть при подсчете контрольной суммы данные пользовательской дейтаграммы разбиваются на последовательности 16-разрядных целых чисел, которые затем суммируются. При сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Затем полученный результат инвертируется, чтобы получилось положительное значение контрольной суммы. Самое удивительное то, что при использовании подобной арифметики проблема нуля, полученного в результате вычисления значения контрольной суммы, решается очень просто. Все дело в том, что при использовании инверсного кода для представления отрицательных чисел существует два нулевых значения: положительное (когда все биты числа равны нулю) и отрицательное (когда все биты числа равны единице). Поэтому, если при вычислении контрольной суммы пользовательской дейтаграммы окажется, что она равно нулю, используется отрицательное представление нуля.

**Псевдозаголовок пользовательской дейтаграммы**

При вычислении контрольной суммы UDP-дейтаграммы используются не только данные, хранящиеся в дейтаграмме. Перед вычислением к UDP-дейтаграмме добавляется псевдозаголовок и общая длина полученной конструкции выравнивается на 16-битовую границу (в конец дейтаграммы добавляется необходимое количество нулевых октетов). После этого вычисляется контрольная сумма полученного объекта. Следует учесть, что нулевые октеты, добавленные к дейтаграмме для выравнивания ее длины, и сам псевдозаголовок *не* передаются конечному получателю. Они также не учитываются при вычислении длины дейтаграммы. При вычислении контрольной суммы значение поля контрольной суммы в заголовке дейтаграммы полагается равным нулю. После суммирования 16-битовых значений псевдозаголовка, заголовка UDP-дейтаграммы и области данных, полученное значение инвертируется, чтобы результат был неотрицательным.

Псевдозаголовок предназначен для проверки корректности доставки UDP-дейтаграммы заданному получателю. Для понимания происходящих процессов необходимо вспомнить, что конечный получатель дейтаграммы определяется IP-адресом компьютера и номером порта на этом компьютере. В заголовке UDP-дейтаграммы указывается только номер порта конечного получателя. Поэтому, чтобы гарантировать корректность доставки дейтаграммы, программы протокола UDP, работающие на машине отправителя, вычисляют контрольную сумму с учетом IP-адреса машины и данных UDP-дейтаграммы. На приемном конце программы протокола UDP проверяют значение контрольной суммы, используя значение IP-адреса, полученное из заголовка IP-дейтаграммы, в которой было доставлено сообщение UDP. Если контрольные суммы совпадают, значит, с высокой долей вероятности можно сказать, что дейтаграмма была корректно доставлена не только указанному компьютеру сети, но и заданному порту этого компьютера.

При вычислении контрольной суммы UDP-дейтаграммы используется псевдозаголовок размером 12 октетов, формат которого показан на рис. 13.2.

Рис. 13.2. Формат псевдозаголовка размером 12 октетов, который используется для вычисления контрольной суммы UDP-дейтаграммы

В первых двух полях псевдозаголовка указываются IP-адреса отправителя и конечного получателя, которые помещаются в заголовок IP-дейтаграммы, содержащей UDP-сообщение. В поле типа протокола указывается код семейства протоколов TCP/IP (для протокола UDP он равен 17). В последнем поле псевдозаголовка указывается длина UDP-дейтаграммы без учета длины самого псевдозаголовка.

При проверке контрольной суммы получатель должен извлечь значения этих полей из заголовка IP-дейтаграммы, поместить их в соответствующие поля псевдозаголовка и заново вычислить контрольную сумму.

**Инкапсуляция и разделение на уровни**

В многоуровневой модели семейства протоколов TCP/IP, транспортный протокол UDP находится над протоколом IP. Теоретически к протоколу UDP должны обращаться пользовательские программы, которые применяют протокол IP для передачи и приема дейтаграмм (рис. 13.3).

Рис. 13.3. В абстрактной многоуровневой модели протокол UDP занимает место между пользовательскими программами и протоколом IP

Помещение протокола UDP над протоколом IP означает, что готовое к отправке UDP-сообщение, состоящее из UDP-заголовка и передаваемых данных, инкапсулируется в IP-дейтаграмму для последующей передачи по объединенной сети (рис. 13.4).

Рис. 13.4. Для передачи по объединенной сети UDP-дейтаграмма инкапсулируется в IP-дейтаграмму. В дальнейшем каждый раз перед передачей по заданному участку физической сети IP-дейтаграмма инкапсулируется в физический фрейм

Для рассмотренных протоколов процесс инкапсуляции означает следующее. Пользовательская программа передает блок данных программе протокола UDP. Последняя добавляет к ним заголовок и формирует UDP-дейтаграмму, которая затем передается программам протокола IP. На уровне протокола IP полученная UDP-дейтаграмма рассматривается как обычные пользовательские данные. Поэтому к ней добавляется заголовок и формируется IР-дейтаграмма. На заключительном этапе IP-дейтаграмма передается программам сетевого интерфейса, которые перед передачей по участку физической сети от одной машины до другой помещают ее в физический фрейм. Формат физического фрейма зависит от используемой для передачи данных сетевой технологии. Как правило, физический фрейм содержит дополнительный заголовок.

После доставки пакета конечному получателю происходит процесс, обратный инкапсуляции. Т.е. вначале пакет поступает на обработку программам нижнего сетевого уровня, а затем “поднимается” вверх по иерархии протоколов. На каждом уровне из пакета удаляется один заголовок, после чего сообщение передается для дальнейшей обработки программам протоколов более высокого уровня. Процесс продолжается до тех пор, пока из сообщения не будут удалены все заголовки. В результате прикладной программе — получателю сообщения данные поступают без служебных заголовков. Таким образом, самый внешний заголовок передаваемого сообщения относится к протоколу самого нижнего уровня, соответственно, самый внутренний заголовок сообщения соответствует протоколу самого верхнего уровня.

При рассмотрении процесса добавления и удаления заголовков важно не забывать о принципе разделения протоколов на абстрактные уровни. В частности, соблюдение многоуровневого принципа в применении к протоколу UDP гарантирует, что пользовательская дейтаграмма, переданная из программы протокола IP на машине получателя, будет точно такой же, как дейтаграмма, которая была передана программам протокола IP на машине отправителя. Поэтому данные, которые были доставлены программой протокола UDP прикладной программе на машине получателя, будут такими же, как были переданы прикладной программой на уровень протокола UDP на машине отправителя. Таким образом, разделение обязанностей между различными протоколами строгое и четкое.

Программы, протокола IP отвечают только за передачу данных между двумя узлами объединенной сети. Программы, протокола UDP выполняют разделение потоков данных между различными источниками или получателями в пределах одного компьютера.

Таким образом, информация об узлах отправителя и получателя находится только в IP-заголовке. Порты отправителя и получателя, с помощью которых разделяются потоки данных в пределах одной машины, указываются только в UDP-заголовке.

**Вычисление контрольной суммы UDP-дейтаграммы при разделении на уровни**

Если вы внимательны, то наверняка уже заметили явное противоречие между принципом разделения на уровни и вычислением контрольной суммы UDP-дейтаграммы. Напомним, что при вычислении контрольной суммы используется псевдозаголовок, в который помещаются IP-адреса отправителя и конечного получателя. Поэтому при передаче сообщения программам протокола UDP прикладной программе должен быть известен IP-адрес конечного получателя. Более того, прикладная программа *должна* его передать вместе с сообщением на уровень протокола UDP. Следовательно, программы протокола UDP могут определить IP-адрес получателя, не обращаясь к программам протокола IP. Однако IP-адрес отправителя зависит от принятого метода маршрутизации IP-дейтаграмм. Напомним, что IP-адрес идентифицирует сетевой интерфейс, через который передается IР-дейтаграмма. Поэтому для определения IP-адреса отправителя программы протокола UDP должны обратиться к программам протокола IP.

Исходя из сказанного, можно предположить, что программы протокола UDP обращаются к уровню протокола IP для определения IP-адреса отправителя и, возможно, получателя. Затем эти адреса помещаются в псевдозаголовок, после чего вычисляется контрольная сумма UDP-дейтаграммы. Далее псевдозаголовок отбрасывается, и пользовательская дейтаграмма передается программам протокола IP для отправки ее по сети.

Для повышения эффективности работы программ протокола UDP был придуман альтернативный подход. Суть его заключается в том, что пользовательская дейтаграмма инкапсулируется в IP-дейтаграмму на уровне протокола UDP. После определения IP-адреса отправителя с помощью программ протокола IP оба адреса (отправителя и получателя) заносятся в соответствующие поля заголовка IР-дейтаграммы. После вычисления контрольной суммы UDP-дейтаграммы, IP-дейтаграмма передается на уровень протокола IP. Программам протокола IP нужно заполнить только оставшиеся поля заголовка IР-дейтаграммы.

Как вы могли убедиться, между уровнями протоколов UDP и IP существует очень тесная связь. При этом возникает вопрос, не нарушает ли она нашу исходную предпосылку насчет жесткого разделения функциональных обязанностей между уровнями? Конечно, нарушает! Протокол UDP тесно связан с протоколом IP. Разработчики пошли на явный компромисс из практических соображений. Они сознательно нарушили принцип разделения протоколов на уровни, поскольку невозможно указать координаты прикладной программы, которой предназначены пакеты, без указания адреса машины конечного получателя, на которой она запущена. Кроме того, желательно иметь эффективный механизм преобразования адресов, используемых как на уровне протокола UDP, так и на уровне протокола IP.

В качестве упражнения вам предлагается рассмотреть описанный подход с другой точки зрения: когда протокол UDP полностью изолирован от протокола IP.

**Мультиплексирование и демультиплексирование UDP-дейтаграмм с помощью портов**

Протокольное программное обеспечение каждого из уровней выполняет мультиплексирование и демультиплексирование объектов, предназначенных для обработки на соседнем уровне. В этом плане протокол UDP не являются исключением. В его задачу входит прием пользовательских дейтаграмм, поступающих от различных прикладных программ, и пересылка их расположенному ниже уровню протокола IP для передачи по сети. Кроме того, протокол UDP получает пользовательские дейтаграммы, поступившие от протокола IP, и перенаправляет их соответствующей прикладной программе.

Теоретически мультиплексирование и демультиплексирование дейтаграмм между программами протокола UDP и прикладными программами должно выполняться исключительно на основе номеров портов. На практике же, прежде чем прикладная программа сможет посылать UDP-дейтаграммы, она должна обратиться к операционной системе и получить стандартный номер порта для данного протокола и ассоциированный с ним номер порта. После назначения номера порта прикладная программа может отправлять через этот порт пользовательские дейтаграммы. При этом номер выделенного порта будет автоматически занесен в поле номера порта отправителя заголовка UDP-дейтаграммы.

При обработке поступающих от протокола IP дейтаграмм, протокол UDP демультиплексирует их на основе номера порта получателя, как показано на рис. 13.5.

Рис. 13.5. Пример демультиплексирования на следующем после протокола IP уровне. При выборе соответствующего номера порта для поступивших дейтаграмм в программе протокола UDP используется поле номера порта получателя заголовка UDP-дейтаграммы

Проще всего представить UDP-порт в виде очереди. В большинстве реализаций протокола при обращении прикладной программы к операционной системе за выделением порта последняя создает внутреннюю очередь, в которую помещаются прибывающие в порт сообщения. Часто прикладной программе разрешается указывать операционной системе размер очереди сообщений или изменять его в процессе работы. При поступлении дейтаграммы в модуль протокола UDP проверяется номер порта получателя. Программа проверяет, был ли выделен указанный в заголовке UDP-дейтаграммы номер порта одной из запущенных прикладных программ. Если указанный номер порта не найден, программа протокола UDP посылает отправителю дейтаграммы ICMP-сообщение о недоступности указанного порта (*port unreachable*), после чего аннулирует дейтаграмму. Если номер порта найден, прибывшая дейтаграмма помещается в ассоциированную с ним очередь, откуда ее сможет прочитать прикладная программа. В случае переполнения очереди сообщений отправляется соответствующее сообщение об ошибке, и дейтаграмма аннулируется.

**Зарезервированные и свободные номера портов UDP**

Теперь пришло время поговорить о назначении номеров портов. Эта проблема крайне важна, поскольку, прежде чем два компьютера смогут взаимодействовать между собой, они должны согласовать номера используемых портов. Например, когда компьютер А хочет переписать файл с компьютера Б, ему нужно знать, какой номер порта назначен программе передачи файлов, запущенной на компьютере Б. При распределении номеров портов возможны два подхода. Первый состоит в использовании центрального органа управления. Все полномочия по назначению номеров портов передаются этому органу, который по запросу выделяет новый номер и публикует список всех назначенных номеров портов. Программное обеспечение разрабатывается с учетом этого списка портов. Такой подход иногда называют методом *универсального назначения* номеров портов. При этом сами порты, номера которым назначены центральным органом управления, называют *стандартными, или широко известными (well-known)*.

Второй подход состоит в динамической привязке номеров портов. При этом не существует никакого списка стандартных назначений. Когда прикладной программе понадобится порт для передачи данных, она обращается с соответствующим запросом к операционной системе, которая выделяет его из списка свободных номеров. Для того чтобы узнать, какой номер порта назначен некоторой программе, запущенной на другом компьютере, необходимо отправить соответствующий запрос на этот компьютер. То есть на компьютер отправляется запрос о номере порта, используемого программой передачи файлов. В ответ компьютер присылает отправителю текущий номер порта, который используется указанной программой.

Создатели протокола TCP/IP применили смешанный подход, при котором части портам заранее назначаются стандартные номера. Все остальные номера портов остаются свободными и могут использоваться прикладными программами в рамках локальных систем. Назначение стандартных номеров портов происходило по принципу “снизу вверх”, т.е. от меньших номеров к большим. При этом большие номера зарезервированы для динамического выделения. Список некоторых номеров портов, используемых в настоящее время программами протокола UDP, приведен в табл. 13.1. Во втором столбце указаны идентификаторы портов, используемые в принятых для Internet стандартах. В третьем столбце приведены идентификаторы, используемые в большинстве систем UNIX.

Таблица 13.1. Номера портов, используемые в протоколе UDP

| **Номер** | **Идентификатор порта в стандарте** | **Идентификатор в UNIX** | **Описание** |
| --- | --- | --- | --- |
| 0 | - | - | Зарезервировано |
| 7 | ECHO | echo | Эхо |
| 9 | DISCARD | discard | Аннулирование |
| 11 | USERS | systat | Список активных пользователей |
| 13 | DAYTIME | daytime | Текущая дата |
| 15 | - | netstat | Программа выдачи состояния сети |
| 17 | QUOTE | qotd | Цитата дня |
| 19 | CHARGEN | chargen | Генератор символов |
| 37 | TIME | time | Текущее время |
| 42 | NAMESERVER | name | Сервер имен хостов |
| 43 | NICNAME | whois | Информация о пользователе |
| 53 | DOMAIN | nameserver | Сервер доменных имен (DNS) |
| 67 | BOOTPS | bootps | Сервер ВООТР или DHCP |
| 68 | BOOTPC | bootpc | Клиент ВООТР или DHCP |
| 69 | TFTP | tftp | Простейший протокол передачи файлов (TFTP) |
| 88 | KERBEROS | kerberos | Служба аутентификации Kerberos |
| 111 | SUNRPC | sunrpc | Дистанционный вызов процедур в системе Sun |
| 123 | NTP | ntp | Протокол синхронизации сетевого времени (NTP) |
| 161 | - | snmp | Простой протокол сетевого управления (SNMP) |
| 162 | - | snmp-trap | Прерывания протокола SNMP |
| 512 | - | biff | Сервер UNIX comsat |
| 513 | - | who | Демон UNIX rwho |
| 514 | - | syslog | Системный журнал |
| 525 | - | timed | Демон службы времени timed |

Учтите, что данный список далеко не полон и приводится лишь в познавательных целях. Если в других транспортных протоколах реализованы функции, аналогичные протоколу UDP, то по возможности в них используются такие же номера портов.

**Проверка знаний: Передача пользовательских дейтаграмм**

**Задание 1**

В чем отличие протокола UPD от протокола IP?

для протокола UDP характерна надежная доставка дейтаграмм c установкой соединения с получателем

в протоколе UPD используется механизм подтверждения доставки сообщений

в протоколе UDP предусмотрена возможность выбора заданного получателя на конкретной машине

**Задание 2**

Что указывается в первых двух 16-битовых полях заголовка пользовательской дейтаграммы?

контрольная сумма UDP-сообщения

размер дейтаграммы в октетах, включая заголовок и пользовательские данные

номера портов отправителя и получателя

**Задание 3**

Расположите процесс инкапсуляции в правильном порядке.

пользовательская программа передает блок данных программе протокола UDP

программа протокола UDP добавляет к блоку данных заголовок и формирует UDP-дейтаграмму, которая затем передается программам протокола IP

на уровне протокола IP к полученной UDP-дейтаграмме добавляется заголовок и формируется IР-дейтаграмма

IP-дейтаграмма передается программам сетевого интерфейса, которые помещают ее в физический фрейм

**Задание 4**

На основе чего протокол UPD демультиплексирует поступающие от протокола IP дейтаграммы?

на основе номера порта получателя

на основе стандартного номера порта для данного протокола

на основе номера порта отправителя

**Задание 5**

Для чего предназначен псевдозаголовок UDP-дейтаграммы?

для выравнивания ее длины

для проверки корректности доставки UDP-дейтаграммы заданному получателю

сам псевдозаголовок передается конечному получателю для проверки контрольной суммы полученного объекта

**Задание 6**

Можно ли рассматривать процессы, запущенные на компьютере пользователя, в качестве конечного получателя дейтаграмм?

Да, можно.  Все процессы, принимающие дейтаграммы на конкретном компьютере, могут быть заменены с уведомлением всех потенциальных отправителей дейтаграмм

Нельзя.  Необходимо сделать так, чтобы на каждой машине имелся набор абстрактных получателей, называемых портами протокола (protocol ports)

Да, можно. Отправитель имеет достаточное количество информации о процессе, запущенном на другой машине

**Надежная потоковая транспортная служба (TCP)**

Ранее была описана ненадежная служба доставки пакетов, не требующая установки соединения с получателем, которая лежит в основе процесса взаимодействия по объединенной сети. Кроме того, вы уже познакомились с протоколом IP, посредством которого IP-дейтаграммы доставляются на машину конечного получателя.

В этом разделе речь пойдет о еще одной важной и популярной сетевой службе — надежной потоковой транспортной службе — и лежащем в ее основе протоколе управления передачей (*Transmission Control Protocol*, или *TCP*). Ниже мы покажем, что по сравнению с описанными ранее протоколами протокол TCP обладает обширными функциональными возможностями. Однако следует заметить, что и реализовать его значительно сложнее.

Хотя выше мы уже говорили о том, что протокол TCP является частью семейства протоколов TCP/IP, используемого в Internet, он по сути является независимым протоколом общего назначения, который можно адаптировать для использования практически с любыми средствами доставки. В частности, поскольку протокол TCP не зависит от применяемого сетевого оборудования, его можно использовать как в пределах одной небольшой локальной сети, наподобие Ethernet, так и в сложной объединенной сети. Более того, протокол TCP стал настолько популярным, что Международная организация по стандартизации (ISO) создала на его основе свой протокол ТР-4.

* [Необходимость в потоковой доставке дейтаграмм](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690587)
* [Особенности надежной службы доставки пакетов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690588)
* [Обеспечение надежности](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690589)
* [Использование движущихся окон](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058a)
* [Протокол управления передачей (TCP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058b)
* [Порты, соединения и конечные точки](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058c)
* [Пассивное и активное открытие соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058d)
* [Сегменты, потоки и порядковые номера](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058e)
* [Окна переменного размера и управление потоком данных](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69058f)
* [Формат ТСР-сегмента](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690590)
* [Передача данных вне очереди](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690591)
* [Параметр максимального размера сегмента](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690592)
* [Вычисление контрольной суммы ТСР-сегмента](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690593)
* [Подтверждение приема и повторная передача](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690594)
* [Время ожидания и повторная передача сегментов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690595)
* [Точный замер полного времени доставки пакета](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690596)
* [Алгоритм Карна и коррекция тайм-аута](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690597)
* [Обработка большого разброса значений задержки](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690598)
* [Реакция на перегрузку сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690599)
* [Перегрузка сети и усечение хвоста очереди](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059a)
* [Произвольное раннее обнаружение (RED)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059b)
* [Установка TCP-соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059c)
* [Начальный порядковый номер](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059d)
* [Закрытие ТСР-соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059e)
* [Сброс ТСР-соединения](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69059f)
* [Протокол TCP и теория конечных автоматов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a0)
* [Принудительная доставка данных](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a1)
* [Зарезервированные номера портов протокола TCP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a2)
* [Производительность протокола TCP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a3)
* [Синдром полного окна и короткие пакеты](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a4)
* [Как избежать синдрома полного окна](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a5)
* [Борьба с синдромом полного окна на стороне получателя](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a6)
* [Задержка сигналов подтверждения приема](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a7)
* [Борьба с синдромом полного окна на стороне отправителя](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a8)
* [Проверка знаний: Надежная потоковая транспортная служба (TCP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905a9)

**Необходимость в потоковой доставке дейтаграмм**

На самом нижнем уровне компьютерные сетевые коммуникационные системы обеспечивают лишь ненадежную доставку пакетов. Это означает, что при передаче по сети пакеты могут быть утеряны, а их содержимое искажено в результате возникновения перекрестных помех, отказов сетевого оборудования или его перегрузки. В сетях, где выполняется динамическая маршрутизация пакетов, возможно нарушение порядка их доставки, возникновение значительных задержек или дубликатов. Более того, от используемого сетевого оборудования может зависеть оптимальный размер пакета, а также другие ограничения, налагаемые для достижения максимальной пропускной способности канала передачи данных.

В тоже время прикладным программам, находящимся на верхнем уровне сетевой иерархии, часто требуется передавать большие массивы данных от одного компьютера к другому. С точки зрения программирования использование для этих целей низкоуровневых ненадежных сетевых протоколов, не требующих установки соединения с получателем, является довольно скучным, однообразным и раздражающим занятием. Причина заключается в том, что в каждой прикладной программе должен быть предусмотрен код для обнаружения и коррекции ошибок, возникающих в процессе передачи данных. Создать такой код, понять как он функционирует, а тем более внести изменения в корректно работающую программу не так-то просто, поэтому для выполнения подобной работы требуются высококвалифицированные программисты, обладающие необходимым багажом знаний. Вследствие этого одной из целей, ставившихся при разработке семейства сетевых протоколов, был поиск общего метода решения проблемы надежности доставки больших потоков данных. Экспертам поручили создать единую универсальную программу потокового сетевого протокола, которая могла бы использоваться всеми прикладными программами. При таком подходе удалось добиться независимости прикладных программ от используемых сетевых технологий и стандартизовать единый интерфейс для служб потоковой передачи данных.

**Особенности надежной службы доставки пакетов**

Интерфейс между прикладными программами и надежной службой доставки пакетов протокола TCP/IP характеризуется пятью основными особенностями.

* *Обработка потоков данных*. При пересылке между двумя прикладными программами (пользовательскими процессами) большого массива данных он рассматривается как *поток* битов, разделенных на октеты, состоящие из 8 битов. Октеты часто по традиции называют *байтами*. Служба потоковой доставки данных, запущенная на машине получателя, должна передать прикладной программе такую же последовательность октетов, которая была передана этой же службе на машине отправителя.
* *Подключение по виртуальному каналу*. Передачу потока данных можно сравнить с обычным телефонным звонком. Перед началом обмена данными отправляющая и принимающая программы должны, вступив во взаимодействие со своими локальными операционными системами, проинформировать их о том, что необходимо запустить процесс потоковой передачи данных. Далее, как и в телефонии, одна из программ “звонит” другой программе, которая должна этот звонок принять. Эти действия выполняют специальные модули сетевого протокола двух операционных систем. Для того чтобы проверить возможность передачи данных и готовность сторон, эти модули обмениваются по объединенной сети специальными сообщениями. Выполнив все формальности, модули протокола информируют свои прикладные программы о том, что *соединение* с получателем (отправителем) установлено и можно начинать передачу данных. Во время передачи данных модули сетевого протокола, работающие на компьютерах, продолжают взаимодействовать, проверяя правильность переданных данных. Если передача данных по какой-либо причине завершается неудачей (например, вышло из строя сетевое оборудование одного из компьютеров, находящегося на пути следования пакетов от отправителя до конечного получателя), это сразу же становится известно вовлеченным в процесс сторонам (точнее, прикладным программам, выполняющим передачу данных). Далее для описания подобного типа соединения мы будем использовать термин *виртуальный канал (virtual circuit)*, поскольку с точки зрения прикладных программ оно функционирует наподобие физического выделенного канала связи. Таким образом, при использовании потоковой службы доставки пакетов для прикладных программ создается иллюзия абсолютно надежной передачи данных.
* *Буферизованная передача*. При передаче потоков данных по виртуальным каналам прикладные программы периодически посылают нижележащему модулю протокола октеты данных. При этом количество октетов данных может быть произвольным — в зависимости от типа используемых прикладных программ (вплоть до одного байта!). На приемном конце модуль протокола доставляет прикладной программе октеты данных из полученного потока в том же порядке, как они были посланы. При этом данные становятся доступными прикладным программам сразу после их получения и проверки на правильность. Отметим, что модулю протокола ничто не мешает разделить поток данных на пакеты, не привязываясь к фрагментам, полученным от прикладных программ. Обычно во время реализации стараются повысить эффективность передачи данных и минимизировать ненужный сетевой трафик. Поэтому сначала модуль протокола собирает необходимое количество данных для формирования дейтаграммы достаточно большого размера и только после этого передает ее по объединенной сети. Таким образом, даже если прикладная программа генерирует поток данных, состоящий из последовательности одиночных октетов, это никак не скажется на эффективности передачи по объединенной сети. Точно так же, если прикладная программа генерирует блоки данных слишком большого размера, перед передачей модуль протокола разобьет их на меньшие части.

Если генерируемые приложением данные должны отправляться сразу (без задержки, связанной с заполнением буфера), в службе потоковой доставки предусмотрен специальный механизм выталкивания, немедленно запускающий процесс передачи данных. На передающей стороне это вызовет немедленную передачу накопленных в буфере данных, даже если он еще не полон. На машине получателя механизм выталкивания также заставляет модуль протокола TCP доставить поступившие данные прикладной программе без задержки. Однако следует заметить, что использование механизма выталкивания гарантирует только немедленную отправку всех накопленных данных. При этом первоначальная структура пакетов не сохраняется. Другими словами, после того как механизм выталкивания приведен в действие, модуль протокола может разделить поток данных самым неожиданным образом.

* *Неструктурированные потоки*. Следует отметить, что потоковая служба доставки протокола TCP/IP не поддерживает обработку структурированных потоков данных. Например, не существует средств, с помощью которых приложение баз данных смогло бы заставить потоковые службы доставки каким-либо образом отслеживать границы записей. Нельзя также идентифицировать содержимое потока на принадлежность к какой-либо прикладной программе. Поэтому при использовании потоковой службы доставки прикладные программы должны сами определять содержимое потока и заранее согласовывать форматы передаваемых данных.
* *Дуплексное соединение*. При установке соединения между потоковыми службами протокола TCP/IP отправителя и получателя, возможна параллельная передача данных в обеих направлениях. Подобное соединение называют *дуплексным (full duplex)*. Прикладная программа воспринимает дуплексное соединение как два независимых потока данных, текущих в противоположных направлениях. При этом между потоками отсутствует какое-либо взаимодействие. В потоковой службе доставки предусмотрен также *полудуплексный режим (half duplex)*, когда данные прикладных программ могут передаваться только в одном направлении в некоторый момент времени. Преимуществом дуплексного режима является то, что модуль протокола может передавать управляющую информацию о пересылаемом потоке данных отправителю (т.е. в обратном направлении). Подобное совмещение передаваемых потоков позволяет уменьшить сетевой трафик.

**Обеспечение надежности**

Выше уже отмечалось, что надежная потоковая служба гарантирует доставку потока данных от одной машины к другой без потери или дублирования данных. При этом у вас может возникнуть вопрос: как модулю протокола удается достичь надежной передачи данных, если все задействованные при этом протоколы нижнего уровня обеспечивают только ненадежную доставку пакетов? Ответить на этот вопрос очень сложно, однако в двух словах можно сказать, что в большинстве протоколов для обеспечения надежности используется технология *подтверждения приема с повторной передачей (positive acknowledgement with retransmission)*. Суть ее заключается в том, что после приема данных, получатель должен связаться с отправителем и отправить ему специальное сообщение о *подтверждении приема (acknowledgement*, или *АСК*). Отправитель сохраняет информацию о каждом посланном пакете. При этом следующий пакет посылается только после получения подтверждения о приеме предыдущего пакета. Кроме того, посылая пакет, отправитель устанавливает значение таймера, по истечении которого выполняется *повторная передача (retransmit)* пакета, если не будет получено подтверждение о его успешном приеме. Упрощенная схема передачи данных с помощью механизма подтверждения приема с повторной передачей приведена на рис. 14.1.

Рис. 14.1. Иллюстрация механизма подтверждения приема с повторной передачей, при котором отправитель ждет уведомления об успешном получении каждого пакета. Величина временной задержки отложена на этой схеме по вертикали, а диагональные линии в центре рисунка представляют процесс передачи пакета по сети

События, происходящие на стороне отправителя и получателя, изображены, соответственно, слева и справа. Диагональные линии в центре рисунка соответствуют передаче одного сообщения по сети.

На рис. 14.2 показано, что произойдет, если в процессе передачи пакет будет утерян или поврежден. Передав пакет, отправитель устанавливает значение таймера, по истечении которого (если не подтвержден прием) считается, что пакет не достиг получателя, и выполняется его повторная передача.

Рис. 14.2. При потере пакета через определенный интервал времени выполняется его повторная передача. Пунктирной линией показан процесс нормальной передачи пакета и получения подтверждения

Еще одна проблема обеспечения надежности возникает при получении системой доставки нижнего уровня дублей пакетов. Одна из причин появления дублей — перегрузка сети, в результате которой увеличивается время доставки пакетов и, как следствие, преждевременно выполняется повторная передача. Решение этой проблемы требует тщательно продуманного подхода, поскольку продублированы могут быть как сами пакеты, так и сигналы об их получении. Обычно для обнаружения дубликатов в надежных протоколах доставки используется нумерация пакетов. Каждому посылаемому пакету присваивается порядковый номер. При этом получатель должен запоминать порядковые номера всех полученных пакетов. Чтобы избежать путаницы, вызванной дубликатами сигналов о получении пакетов, в механизме подтверждения приема с повторной передачей порядковые номера посылаются обратно отправителю вместе с сигналом об успешном получении пакета. В результате отправитель сможет корректно сопоставить сигнал о получении пакета с самим пакетом.

**Использование движущихся окон**

Прежде чем приступить к рассмотрению потоковой службы доставки протокола TCP, необходимо познакомиться еще с одним методом, используемым в процессе потоковой передачи данных. Речь идет о *движущихся (скользящих) окнах (sliding window)*, позволяющих повысить эффективность потоковой передачи данных. Чтобы понять принцип их работы, необходимо проанализировать последовательность событий, происходящих при надежной передаче пакета (рис. 14.1). Передав пакет, отправитель ждет подтверждения его получения, а затем передает следующий пакет.

Как показано на рис. 14.1 в определенный момент поток данных между машинами течет только в одном направлении, и это при том, что в сети возможна одновременная передача потоков данных в обоих направлениях! Очевидно, что при возникновении задержек между отправкой пакетов (например, когда компьютеры вычисляют маршруты следования пакетов или контрольные суммы), сеть будет загружена не полностью. А если представить себе сеть, в которой возникают большие задержки при передаче пакетов, проблема становится понятной.

В простейшем случае использование технологии подтверждения приема с повторной передачей приводит к существенному снижению пропускной способности сети, поскольку отправка нового пакета откладывается до момента получения подтверждения о приеме предыдущего пакета.

Метод движущихся окон является более сложным вариантом технологии подтверждения приема с повторной передачей, по сравнению с описанным выше упрощенным вариантом. Она позволяет эффективно использовать полосу пропускания сети, поскольку отправитель может послать несколько пакетов сразу, не дожидаясь подтверждения приема каждого пакета. Проще всего описать работу метода движущихся окон на примере последовательности передаваемых пакетов, изображенных на рис. 14.3. Суть метода состоит в том, что небольшое *окно* фиксированного размера накладывается на последовательность предназначенных для передачи пакетов. При этом передаются все пакеты, попавшие внутрь окна.

Рис. 14.3. Движущееся окно, внутрь которого помещено 8 пакетов (а); при получении подтверждения о приеме пакета 1 окно сдвигается на один пакет вправо, и в сеть отправляется 9-й пакет (б). Если для какого-либо пакета, попавшего в окно, не получен сигнал подтверждения, то выполняется повторная передача этого пакета

Пакет считается *непринятым (unacknowledged)*, если он был послан в сеть, но не было получено подтверждение о его приеме. Формально количество непринятых пакетов в любой момент времени не может превышать *размеров окна* и всегда является небольшим целым числом. Например, если размер движущегося окна равен 8, отправитель может отправить в сеть до 8 пакетов и только после этого перейти в состояние ожидания подтверждений об их приеме.

Как показано на рис. 14.3, получив подтверждение о приеме первого из пакетов, находящихся внутри окна, отправитель сдвигает окно на один пакет вправо и посылает в сеть следующий пакет. По мере получения новых подтверждений окно все время передвигается вправо.

Производительность метода движущихся окон зависит как от размера самого окна, так и от скорости пересылки пакетов по сети и задержек, возникающих при их обработке. На рис. 14.4 этот метод продемонстрирован на примере окна, в которое помещается три пакета. Обратите внимание, что отправитель успевает послать все три пакета в сеть до того, как будет получено хотя бы одно сообщение о подтверждении их приема.

Рис. 14.4. Пересылка трех пакетов с использованием метода движущихся окон. Идея заключается в том, что отправитель может послать в сеть сразу все три пакета, не дожидаясь сообщений о подтверждении их приема

Если принять размер окна равным единице, то метод движущихся окон ничем не будет отличаться от рассмотренного в предыдущем разделе метода подтверждения приема с повторной передачей. Увеличивая размер окна, можно полностью исключить простои в сети. Это произойдет при достижении равновесия между количеством посланных в сеть пакетов, скоростью их передачи и задержками, возникающими при их обработке на узлах сети.

При использовании метода движущихся окон правильный выбор размера окна позволяет полностью исключить простои в сети, достичь большей производительности по сравнению с методом подтверждения приема с повторной передачей

При использовании метода движущихся окон в модуле протокола необходимо отслеживать сигналы подтверждения приема для каждого находящегося в окне пакета. А это означает, что для каждого непринятого пакета должен быть установлен отдельный таймер. Тогда по исчерпании значения соответствующего таймера легко определить, какой из пакетов утерян, и передать его повторно. Получив подтверждение приема, отправитель должен переместить окно и вытеснить из него все принятые пакеты, идущие подряд. В модуле протокола на машине получателя создается аналогичное окно. Это позволяет получателю отслеживать принятые пакеты и отправлять подтверждения об их приеме. Таким образом, окно разделяет последовательность пакетов на три части. Те, что находятся слева от окна, уже переданы в сеть и успешно достигли получателя. Пакеты, внутри окна передаются, а пакеты, находящиеся справа от окна, ожидают отправки. Самый левый из находящихся в окне пакетов имеет наименьший порядковый номер и является первым в списке непринятых пакетов.

**Протокол управления передачей (TCP)**

Рассмотрев принцип работы движущихся окон, можно переходить к изучению надежной потоковой службы, являющейся составной частью семейства протоколов TCP/IP. Эта служба называется *протоколом управления передачей (Transmission Control Protocol*, или *TCP*). Она настолько важна, что все семейство протоколов назвали в ее честь — TCP/IP. Однако следует помнить, что

TCP является коммуникационным протоколом, а не просто частью программы.

Разница между протоколом и программой примерно такая же, как между описанием языка программирования и его реализацией (т.е. компилятором). Как и при разработке компиляторов, иногда трудно провести грань между спецификацией протокола и его реализацией. Пользователям чаще всего приходится сталкиваться именно с реализацией протокола TCP, а не с его спецификацией. Поэтому неудивительно, что конкретную реализацию протокола TCP часто путают со стандартом. Тем не менее необходимо четко различать эти две вещи.

Итак, какие же функции выполняет протокол TCP? Их так много и протокол так сложен, что ответить на этот, казалось бы, простой вопрос не так-то просто. В спецификации протокола определен точный формат данных и подтверждающих сообщений, которыми обмениваются два компьютера при организации надежного канала передачи данных. Кроме того, в спецификации протокола описаны процедуры, которые должны выполнять машины в сети для обеспечения корректной доставки данных. В стандарте протокола TCP оговорено, как различать получателей пакетов на конкретной машине, а также указаны способы восстановления данных взаимодействующими сторонами при возникновении различного рода ошибок, таких как потеря или дублирование пакетов. Кроме того в стандарте протокола TCP указаны способы установки и завершения сеансов потоковой передачи данных.

Несколько слов о том, что не включено в стандарт протокола. Хотя в спецификации протокола описаны методы использования протокола TCP прикладными программами, в ней не оговорены детали реализации конкретного интерфейса между прикладной программой и модулем протокола TCP. Другими словами, в стандарте протокола зафиксированы только действия, выполняемые конкретной реализацией модуля TCP, но не описаны процедуры, которые должна вызывать прикладная программа для выполнения этих действий.

Описание прикладного интерфейса не включено в стандарт протокола TCP по одной причине — стандарт должен быть универсальным. В частности, поскольку разработчики обычно встраивают модуль реализации протокола TCP в операционную систему, им нужно поддержать все виды интерфейсов, предоставляемые системой. Кроме того, у разработчиков появляется возможность создавать программное обеспечение, выполняющее одинаковые функции согласно спецификации протокола TCP, для различных компьютерных платформ.

Поскольку протокол TCP не привязан к системе передачи данных, он может использоваться совместно с различными системами доставки пакетов (протоколами транспортного уровня), включая широко используемую службу доставки дейтаграмм протокола IP. Например, реализации протокола TCP могут быть созданы для связи по коммутируемым телефонным каналам, локальным сетям, высокоскоростным оптоволоконным каналам или низкоскоростным сетям большой протяженности. Фактически, одним из самых больших преимуществ протокола TCP является возможность его совместной работы практически с любым средством доставки пакетов.

**Порты, соединения и конечные точки**

Как и протокол передачи пользовательских дейтаграмм (UDP), описанный в разделе [Передача пользовательских дейтаграмм (UDP)](https://it.rfei.ru/course/%7EGZqN/-/%7E3nCx) протокол TCP располагается в системе иерархии выше уровня протокола IP. Абстрактная многоуровневая модель протоколов семейства TCP/IP приведена на рис. 14.5.

Рис. 14.5. Абстрактная многоуровневая модель протоколов семейства TCP/IP. Из рисунка видно, что оба протокола (как UDP, так и TCP) расположены выше уровня протокола IP. При этом протокол TCP обеспечивает надежную потоковую службу доставки пакетов, тогда как UDP выполняет только ненадежную доставку пакетов получателю. Несмотря на это оба протокола успешно используются в прикладных программах

Протокол TCP позволяет нескольким, выполняющимся на компьютере прикладным программам параллельно и независимо обмениваться данными с прикладными программами, запущенными на других машинах. Так же, как и UDP, протокол TCP демультиплексирует свой входной трафик между несколькими прикладными программами. Поэтому по аналогии с протоколом UDP, в протоколе TCP также используется понятие *номеров портов*, позволяющих идентифицировать получателя на конкретной машине. Каждому из портов для идентификации присваивается небольшое целое число1.

При описании портов протокола UDP мы говорили о том, что каждый порт можно представить в виде очереди, в которую сетевое программное обеспечение помещает все полученные дейтаграммы. Порты протокола TCP — намного более сложное понятие, поскольку номеру порта нельзя поставить в соответствие какой-либо один объект. Поэтому в протокол TCP введено абстрактное понятие *соединения (connection abstraction)*, в котором идентифицируемыми объектами являются соединения, осуществляемые по виртуальным каналам, а не отдельные порты. Таким образом, ключевой в протоколе TCP является идея использования соединений. Понимание этой идеи поможет вам понять назначение и принцип использования номеров портов протокола TCP.

В качестве основной абстракции в протоколе TCP используется понятие соединения, а не номера порта. Соединения идентифицируются с помощью пары конечных точек.

Осталось ответить на вопрос, что же такое конечные точки соединения? Как уже было сказано, соединением называется виртуальный канал, проложенный между двумя прикладными программами. Поэтому вполне естественно предположить, что конечными точками соединения являются сами прикладные программы. Однако это не так. В протоколе TCP *конечная точка* определяется с помощью 2-мерного кортежа, или пары целых чисел (узел, порт), где параметр узел определяет IP-адрес узла сети, а параметр порт является номером TCP-порта на этом узле. Например, запись (128.10.2.3, 25) определяет в качестве конечной точки TCP-порт с номером 25 на машине, IP-адрес которой 128.10.2.3.

Теперь, после определения понятия конечной точки, несложно понять, что такое само соединение. Напомним, что соединение определяется парой своих конечных точек. Поэтому если установлено соединение между машиной с IP-адресом 18.26.0.36, расположенной в Массачуссетском технологическом институте, и машиной с IP-адресом 128.10.2.3, расположенной в университете Пердью, оно идентифицируется парой конечных точек:

(18.26.0.36,1069) и (128.10.2.3, 25).

Если в это же время к тому же самому компьютеру, находящемуся в университете Пердью, поступит запрос на установку соединения от другой машины с IP-адресом 128.9.0.32, расположенной в институте информационных исследований (Information Sciences Institute), то соединение будет идентифицироваться двумя конечными точками следующим образом:

(128.9.0.32,1184) и (128.10.2.3, 53).

Приведенные примеры соединений были довольно простыми, поскольку номера портов, используемые во всех конечных точках, имели уникальные значения. Однако в концепции абстрактного соединения допускается совместное использование одной конечной точки несколькими соединениями. Например, можно добавить к двум рассмотренным выше соединениям с машиной университета Пердью еще одно, установленное с машиной 128.2.254.139, находящейся в университете Карнеги-Меллона:

(128.2.254.139, 1184) и (128.10.2.3, 53).

На первый взгляд может показаться странным, что два соединения могут одновременно использовать ТСР-порт 53 на машине 128.10.2.3, однако никакой неопределенности при этом не возникает. Все дело в том, что в протоколе TCP все входящие сообщения привязываются не к заданному номеру порта протокола, а к открытому соединению, которое в свою очередь идентифицируется двумя конечными точками. Итак, необходимо запомнить следующее.

Поскольку в протоколе TCP соединение идентифицируется по двум конечным точкам, один номер TCP-порта может использоваться несколькими соединениями, открытыми на одной и той же машине.

1. Хотя и в протоколе TCP и в протоколе UDP для идентификации портов используются целые числа, начиная с 1, никакого конфликта по номерам портов двух протоколов не происходит. Причина состоит в том, что во входящей дейтаграмме указывается как используемый тип протокола, так и номер порта, относящийся к этому протоколу.

**Пассивное и активное открытие соединения**

В отличие от UDP, протокол TCP требует установки соединения с получателем. А это означает, что обе конечные точки должны предварительно дать согласие на участие в соединении. Другими словами, прежде чем поток данных протокола TCP сможет проследовать по объединенной сети, прикладные программы, расположенные на концах соединения, должны разрешить установку соединения. Для этого одна из прикладных программ вызывает функцию *пассивного открытия соединения* операционной системы. Вызов такой функции является сигналом о том, что данная сторона готова к приему входящих соединений.

После вызова описываемой функции операционная система назначает номер ТСР-порта для своей конечной точки данного соединения. Для установки соединения, прикладная программа, находящаяся на другом конце виртуального канала должна обратиться к операционной системе с запросом на *активное открытие* соединения. При этом два модуля протокола TCP, запущенные на подключающихся компьютерах, взаимодействуют между собой и проверяют возможность установки соединения. После того как соединение установлено, прикладные программы могут обмениваться данными. В процессе работы модули протокола TCP, находящиеся по обе стороны соединения, незаметно для прикладных программ обмениваются служебными сообщениями, гарантирующими надежность доставки данных. Подробнее процесс установки соединения будет рассмотрен чуть ниже после изучения формата сообщения протокола TCP.

**Сегменты, потоки и порядковые номера**

В протоколе TCP перед передачей по сети поток данных разбивается на *сегменты*, состоящие из последовательности октетов, или байтов. Как правило, каждый сегмент данных передается по объединенной сети в виде одной IP-дейтаграммы.

Для решения двух важных задач — повышения эффективности передачи данных и управления потоком данных — в протоколе TCP применяется специализированный механизм движущихся окон, наподобие того, что был описан выше. Он позволяет модулю протокола TCP отправлять сразу несколько сегментов данных до получения сообщений об их доставке. Подобный подход позволяет увеличить суммарную пропускную способность сети за счет сокращения времени ее простоя. Вариант движущихся окон, реализованный в протоколе TCP, позволяет также решить еще одну проблему — *сквозного управления потоком данных*. В распоряжение получателя предоставлены средства, позволяющие ограничить поток передаваемых данных в случае исчерпания буферного пространства для приема данных.

Механизм движущихся окон протокола TCP оперирует октетами данных, а не сегментами или пакетами. При этом октеты потока данных нумеруются последовательно, а отправитель запоминает три указателя для каждого открытого соединения. Эти указатели определяют движущееся окно, как показано на рис. 14.6.

Рис. 14.6. Пример движущегося окна протокола TCP. Октеты 1 и 2 были успешно доставлены получателю; октеты 3-6 посланы в сеть, но подтверждение об их доставке еще не получено; октеты 7-9 еще не отправлены, но могут быть отправлены без всяких задержек; октеты с номерами 10 и выше не могут быть посланы в сеть до тех пор, пока не попадут внутрь окна

Первый указатель обозначает левую границу движущегося окна. Он разделяет октеты на те, что были успешно доставлены получателю, и те, которые отправлены в сеть, но подтверждение их доставки еще не получено. Второй указатель обозначает правую границу движущегося окна. Он определяет номер старшего октета последовательности, который может быть послан в сеть до поступления сигналов о подтверждении доставки других октетов, находящихся в окне. Третий указатель обозначает границу внутри окна, разделяющую последовательность октетов на те, что уже были посланы в сеть, и те, которые еще только предстоит отправить. Модуль протокола отправляет в сеть все октеты, находящиеся в окне, без задержки. Поэтому граница внутри окна, определяемая третьим указателем, обычно перемещается слева направо очень быстро.

Выше было описано, как модуль протокола TCP, работающий на машине отправителя перемещает окно вдоль последовательности октетов. Кроме того, было отмечено, что модуль протокола TCP, запущенный на машине получателя, должен создать точно такое же окно для того, чтобы собрать поток получаемых данных. Однако не стоит забывать, что устанавливаемые протоколом TCP соединения являются дуплексными. А это означает, что одновременно по каждому из виртуальных каналов можно передавать два потока данных в противоположных направлениях. Таким образом, передачу этих двух потоков можно считать полностью независимыми процессами, поскольку в любой момент времени данные могут течь как в одном направлении, так и в обоих. Поэтому программы реализации протокола TCP, запущенные на обеих концах соединения, должны поддерживать для каждого соединения по два окна (всего окон четыре). Одно окно используется для передачи потока данных, а другое — для приема.

**Окна переменного размера и управление потоком данных**

Еще одно различие между механизмом движущихся окон протокола TCP и упрощенным механизмом, рассмотренным выше, состоит в том, что размер окна в протоколе TCP может с течением времени изменяться. В каждом сообщении, подтверждающем получение данных, которое присылает получатель, указывается количество принятых октетов и так называемый *анонс окна (window advertisement)*. В анонсе указывается количество дополнительных октетов данных, которые сможет принять получатель. Анонс окна можно считать сообщением, присылаемым получателем потока данных, о текущем размере своего буфера. В ответ на получение анонса окна с увеличенными размерами отправитель соответственно увеличивает размер своего движущегося окна и отправляет получателю дополнительные октеты, не дожидаясь получения сигналов подтверждения приема. Получив анонс окна с уменьшенными размерами, отправитель должен уменьшить размеры своего движущегося окна и не отправлять октеты, расположенные за границами окна. Модуль протокола TCP не должен сразу реагировать на поступление анонса окна с уменьшенными размерами и сдвигать влево границу окна в потоке данных. Он просто должен дождаться подтверждения приема, не посылая октеты получателю. Таким образом, через некоторое время размер окна уменьшится автоматически по мере сдвига вправо его левой границы.

Преимущество использования окна переменного размера заключается в том, что, помимо повышения надежности передачи данных, оно обеспечивает возможность управления потоком данных. Чтобы не допустить переполнения буфера, получатель должен по мере его заполнения посылать анонсы окон меньшего размера. В исключительных случаях для прекращения передачи данных получатель анонсирует окно нулевого размера. После того как освободится место в буфере, получатель присылает анонс ненулевого окна для возобновления передачи данных1.

Описанный механизм управления потоком данных протокола TCP очень важен для работы реальной объединенной сети, состоящей из большого количества сетей разной протяженности и пропускной способности, соединенных маршрутизаторами различной мощности и скорости. Однако следует знать о двух не связанных между собой проблемах управления потоком. Первая — заключается в том, что в семействе протоколов TCP/IP должен быть предусмотрен механизм сквозного управления потоком данных между отправителем и получателем. Предположим, например, что небольшой персональный компьютер подключается по сети с большой пропускной способностью к мощной суперЭВМ. При этом у персонального компьютера должны быть средства управления поступающим от суперЭВМ потоком данных, иначе очень быстро переполнится входной буфер модуля протокола. Поэтому для гарантирования надежной доставки данных в протоколе TCP должны быть реализованы средства сквозного управления потоком данных. Вторая проблема состоит в том, что в семействе протоколов TCP/IP должен быть реализован механизм управления потоком данных для промежуточных устройств (т.е. маршрутизаторов). Он задействуется, когда промежуточная машина не справляется с потоком данных, поступающим от отправителя.

Если промежуточное устройство не справляется с поступающим потоком данных, такая ситуация называется *перегрузкой (congestion)*. При этом в действие приводятся механизмы *устранения перегрузки (congestion control)*. Для сквозного управления потоком данных в протоколе TCP используется механизм движущихся окон. Однако в нем не предусмотрен отдельный механизм устранения перегрузки. Тем не менее ниже будет показано, что при хорошо продуманной и тщательной реализации программ протокола TCP можно обнаружить перегрузку на промежуточном устройстве и устранить ее. В то же время некачественная реализация программ протокола может даже усугубить ситуацию. В частности, хорошо продуманная схема повторной передачи данных может устранить перегрузку, возникшую на одном из промежуточных устройств, тогда как плохо выбранная схема может привести к еще большей перегрузке устройства.

1. Существует два исключительных случая при передаче данных, когда получен анонс окна нулевого размера. Во-первых, отправитель имеет право передать сегмент данных, в заголовке которого установлен бит срочности. Это позволяет проинформировать получателя о том, что должны быть переданы срочные данные. Во-вторых, чтобы избежать потенциальной тупиковой ситуации, когда после получения анонса окна нулевого размера будет утерян анонс окна ненулевого размера, отправитель периодически посылает запросы на возобновление передачи данных.

**Формат ТСР-сегмента**

Единицей передачи данных между двумя машинами в протоколе TCP является *сегмент*. Сегменты используются для установки соединения, передачи данных, отправки сигналов подтверждения приема, анонсирования размера окон и закрытия соединения. Поскольку протокол TCP является дуплексным (т.е. потоки данных могут течь одновременно в двух направлениях) сигналы подтверждения, посланные от машины А к машине Б, могут передаваться в тех же сегментах, что и потоки данных от машины А к машине Б. И это несмотря на то, что сигналы подтверждения приема относятся к потокам данных, текущих от машины Б к машине А1. Формат сегмента протокола TCP приведен на рис. 14.7.

Рис. 14.7. Формат TCP-сегмента, состоящего из заголовка и блока данных. Сегменты используются не только для передачи данных, а также для отправки сигналов подтверждения приема, установки и разрыва соединения

Каждый сегмент состоит из двух частей — заголовка и блока данных. Как и следовало ожидать, в заголовке, называемом также *TCP-заголовком*, находятся идентификационные данные и управляющая информация. В первых двух полях заголовка располагаются *номера TCP-портов отправителя* и *получателя*, которые идентифицируют прикладные программы по обе стороны соединения. Поле *порядкового номера* идентифицирует положение текущего сегмента в потоке данных отправителя. В поле *номера сигнала подтверждения* указывается номер октета, который отправитель ожидает в дальнейшем получить обратно. Обратите внимание: порядковый номер сегмента относится к потоку данных, текущему в том же направлении, что и сегмент, номер же сигнала подтверждения — потоку данных, текущему в противоположном направлении.

В поле *длины заголовка2* указывается целое число, обозначающее длину заголовка сегмента, выраженное в блоках длиной 32 бита. Данное поле включено в заголовок, поскольку *поле параметров протокола* имеет переменную длину, которая зависит от того, какие параметры включены в сегмент. Таким образом, размер TCP-заголовка зависит от того, какие параметры в него включены. Поле, обозначенное на рис. 14.7 как *Резерв* имеет размер 6 битов и зарезервировано для использования в будущих стандартах протокола.

Поскольку в протоколе TCP сегменты используются и для передачи данных, и для передачи сигналов подтверждения их приема, а также для передачи запросов на установку и закрытие соединения, в TCP-заголовок включено специальное 6-битовое поле *кода сегмента*, которое определяет его формат и содержимое. От значения битов этого поля зависит, как будут интерпретироваться другие поля заголовка (табл. 13.1).

Таблица 14.1. Значение битов кода сегмента ТСР-заголовка

| **Название бита (слева направо)** | **Значение, если бит установлен в 1** |
| --- | --- |
| URG | В заголовке присутствует указатель срочных данных |
| АСК | В заголовке указано поле подтверждения приема |
| PSH | В данном сегменте указан запрос на немедленную отправку данных (push) |
| RST | Сброс соединения |
| SYN | Синхронизация порядковых номеров |
| FIN | Отправитель достиг конца потока данных |

В поле *размера окна* отправляемого сегмента программы протокола TCP помещают количество октетов, которые они могут принять (т.е. указывают размер своего приемного буфера). В это поле помещается 16-битовое беззнаковое целое число, имеющее стандартный сетевой порядок следования байтов. Анонсирование окна — еще один пример дуплексной передачи данных, поскольку оно выполняется для всех сегментов, включая сегменты, в которых передаются данные, и сегменты с сигналами подтверждения приема.

1. На практике дуплексный режим обычно редко используется, поскольку большинство приложений, как правило, не посылают данные одновременно в двух направлениях.

2. В спецификации протокола TCP отмечается, что в поле длины заголовка указывается смещение области данных относительно начала сегмента.

**Передача данных вне очереди**

Поскольку протокол TCP является потоковоориентированным, иногда прикладной программе, запущенной на одном конце соединения, необходимо отправить данные другой программе *вне очереди (out of band)*. Речь идет о том, чтобы программа на другом конце соединения получила их сразу, не ожидания приема октетов, которые были отправлены раньше. Например, если протокол TCP используется для создания сеанса связи с удаленным терминалом, пользователю иногда нужно послать удаленной программе специальный сигнал с клавиатуры, который *прервет (interrupt)* ее выполнение или вовсе завершит (abort) работу программы. Обычно подобные сигналы чаще всего посылаются, если программа, запущенная на удаленном компьютере, зависла или перестала корректно работать. Понятно, что управляющие сигналы должны быть переданы программе вне очереди, без ожидания, пока она считает из входного потока все посланные ранее октеты. В противном случае, если программа по какой-либо причине прекратит считывание потока данных, то управляющие сигналы никогда не смогут ее достичь.

Для реализации режима передачи внеочередных сигналов в протоколе TCP для отправителя предусмотрена возможность помечать находящиеся в сегменте данные как *срочные (urgent)*. Это означает, что программа-получатель должна быть извещена о поступлении таких данных так быстро, насколько это возможно, вне зависимости от состояния входящего потока данных. В спецификации протокола указано, что после поступления срочных данных, модуль протокола TCP должен уведомить прикладную программу, открывшую соединение, о переходе в “срочный режим” работы. После того как поток срочных данных будет исчерпан, модуль протокола TCP вновь должен уведомить прикладную программу о переходе в обычный режим работы.

Естественно, что все детали механизма уведомления прикладной программы о прибытии срочных данных зависят от используемой на компьютере операционной системы. Однако при отправке срочных данных этот механизм унифицирован. В поле *кода сегмента* должен быть установлен бит срочных данных URG, и в поле *указателя срочных данных* занесено соответствующее значение. Если бит URG установлен, то поле *указателя срочных данных* определяет позицию в сегменте, где заканчиваются срочные данные.

**Параметр максимального размера сегмента**

Следует отметить, что через открытое соединение посылаются сегменты разного размера. Естественно, обе стороны должны заранее договориться о том, какова максимально возможная длина передаваемого сегмента. Для обмена информацией с модулем протокола TCP, запущенным на противоположной стороне соединения, используется поле *параметров протокола*. Один из этих параметров позволяет программе протокола TCP указать *максимальный размер сегмента (maximum segment size*, или *MSS*), который она будет принимать. Например, если небольшой встроенный микрокомпьютер, оснащенный малым объемом оперативной памяти подключается по сети к мощному суперкомпьютеру, он сначала должен сообщить суперкомпьютеру максимальный размер сегмента, чтобы получаемые от суперкомпьютера сегменты могли поместиться в выделенных для этой цели буферах.

Описываемый параметр играет важную роль, когда компьютеры подключены к высокоскоростной локальной сети. В этом случае для повышения пропускной способности сети максимальный размер сегмента должен быть таким, чтобы сегмент целиком помещался в пакете. Поэтому, если отправитель и получатель находятся в пределах одной физической сети, модуль протокола TCP обычно выбирает максимальный размер сегмента так, чтобы получившаяся в результате IP-дейтаграмма соответствовала максимальной единице передачи данных (MTU) физической сети. Если же отправитель и получатель находятся в разных физических сетях, то можно попытаться определить минимальный размер MTU сети, находящейся по пути передачи пакетов, или же установить максимальный размер сегмента равным *536* октетов, т.е. стандартный размер IP-дейтаграммы (*576* октетов) минус стандартный размер IP- и ТСР-заголовков.

В реально работающей объединенной сети выбрать оптимальный размер сегмента не так-то просто, поскольку ее пропускная способность будет снижаться при передаче как очень больших, так и малых сегментов. С одной стороны, при небольшом размере сегмента эффективность использования сети остается крайне низкой. Чтобы понять, почему так происходит, необходимо вспомнить, что ТСР-сегменты при передаче инкапсулируются в IP-дейтаграммы, а те, в свою очередь, — в физические сетевые фреймы. Поэтому, кроме данных, к каждому сегменту добавляется по меньшей мере еще 40 октетов — заголовки ТСР-сегмента и IP-дейтаграммы. В результате при передаче дейтаграммы, содержащей только один октет данных, для передачи пользовательских данных используется всего 1/41 часть выделенной полосы пропускания сети. На практике, если учесть минимальный межпакетный промежуток и биты синхронизации фреймов, автоматически добавляемые сетевым оборудованием, то это соотношение будет еще меньше.

С другой стороны, очень большие сегменты также снижают производительность сети. Для больших сегментов создаются большие IP-дейтаграммы. Когда такая дейтаграмма передается по сети с малым значением MTU, на уровне протокола IP она фрагментируется. В отличие от TCP-сегментов, подтверждение получения каждого фрагмента на уровне протокола не выполняется, кроме того отдельные фрагменты дейтаграммы при необходимости не могут быть повторно переданы независимо от всей дейтаграммы. Таким образом, в случае утери или повреждения одного из фрагментов повторно должна быть передана вся дейтаграмма. Поскольку вероятность потери отдельного фрагмента отлична от нуля, увеличение размеров сегмента больше определенного значения (порога фрагментации) уменьшает вероятность успешного приема дейтаграммы и, следовательно, ведет к уменьшению пропускной способности сети.

Теоретически оптимальный размер сегмента S определяется максимально возможным размером генерируемых IP-дейтаграмм, которые на протяжении всего маршрута от отправителя до конечного получателя не фрагментируются. На практике нахождение значения S затруднено по нескольким причинам.

Во-первых, в большинство реализаций протокола TCP не включен механизм для определения оптимального размера сегмента1.

Во-вторых, поскольку маршрутизаторы в объединенной сети могут динамически изменять маршруты дейтаграмм, возможно изменение пути их следования между двумя взаимодействующими компьютерами. Следовательно, величина MTU в любой момент может без предупреждения измениться, после чего неминуемо последует фрагментация дейтаграмм.

В-третьих, оптимальный размер сегмента зависит от длины заголовков протоколов более низкого уровня. Кроме того, при использовании параметров протокола IP размер сегмента должен быть еще меньше. Поэтому с грустью приходится констатировать, что проблема нахождения оптимального размера сегментов протокола TCP не решена до сих пор.

1. Для нахождения минимального значения MTU на всем пути следования дейтаграмм отправитель должен прозондировать маршрут их следования с помощью специальных IP-дейтаграмм, в заголовке которых установлен бит запрета фрагментации. При получении ICMP-сообщения об ошибке фрагментации отправитель должен уменьшить размер дейтаграммы и повторить зондирование. Описанный алгоритм выполняется до тех пор, пока дейтаграмма не пройдет весь маршрут без ошибок.

**Вычисление контрольной суммы ТСР-сегмента**

В поле *контрольной суммы*, TCP-заголовка помещается 16-битовое целое число, которое используется для проверки целостности полученных данных и заголовка TCP-сегмента. При вычислении контрольной суммы модуль протокола TCP, работающий на машине отправителя, использует алгоритмы, аналогичные описанным в разделе [Передача пользовательских дейтаграмм (UDP)](https://it.rfei.ru/course/%7EGZqN/-/%7E3nCx) для протокола UDP. Перед вычислением контрольной суммы к началу сегмента добавляется *псевдозаголовок*, а в конец сегмента — необходимое количество нулевых битов для выравнивания общей длины полученной конструкции на границу 16-битов. После выполнения описанных подготовительных действий вычисляется значение 16-битовой контрольной суммы полученной конструкции. Длина псевдозаголовка и битов, добавленных для выравнивания, не учитывается в общей длине TCP-сегмента, поскольку они не передаются получателю. Кроме того, при вычислении значение поля *контрольной суммы*, TCP-заголовка полагается равным нулю. Как и при вычислении других контрольных сумм, при сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Затем полученный результат инвертируется, чтобы получилось положительное значение контрольной суммы. После получения сегмента, модуль протокола TCP выполняет над ним аналогичные вычисления и сравнивает контрольные суммы. Если они совпадают, значит сегмент получен без ошибок.

Псевдозаголовок используется в тех же целях, что и в протоколе UDP, — он позволяет принимающей стороне удостовериться в том, что сегмент доставлен по назначению, путем проверки IP-адресов отправителя и конечного получателя, а также номера порта протокола. В протоколе TCP IP-адреса отправителя и конечного получателя играют очень важную роль, поскольку они используются для идентификации соединения, к которому относится полученный сегмент. Поэтому после прибытия дейтаграммы, содержащей TCP-сегмент, модуль протокола IP должен передать модулю протокола TCP, кроме самого сегмента, еще и IP-адреса отправителя и конечного получателя.

При отправке сегмента модуль протокола TCP помещает в поле *типа протокола* значение, которое затем будет использоваться в соответствующем поле типа протокола низкоуровневой системы доставки. Для IP-дейтаграмм, содержащих TCP-сегменты, это значение, согласно спецификации, равно 6. В поле *длины ТСР-сегмента* указывается общая длина сегмента вместе с заголовком. На принимающей стороне информация, помещаемая в псевдозаголовок, извлекается из IP-дейтаграммы, содержащей TCP-сегмент, и используется затем в процессе вычисления контрольной суммы. По значению контрольной суммы можно судить о том, доставлен ли сегмент указанному получателю без ошибок.

**Подтверждение приема и повторная передача**

Поскольку в протоколе TCP данные посылаются в виде сегментов переменного размера и в повторно передаваемые сегменты, кроме оригинала, могут быть включены дополнительные данные, получаемые сигналы подтверждения приема не так-то просто соотнести с отправленными дейтаграммами или сегментами. По этой причине сигналы подтверждения приема соотносятся с положением данных в передаваемом потоке, которым присваиваются порядковые номера. Получатель собирает поступившие в сегментах данные и воспроизводит точную копию передаваемого потока данных. Так как сегменты поступают получателю в виде IP-дейтаграмм, которые в процессе передачи могут быть утеряны или доставлены в другом порядке, для упорядочения сегментов получатель использует их порядковые номера. В какой-то момент времени у получателя может быть восстановлено произвольное количество октетов (включая ноль) от начала потока. Кроме того, получателю может быть доставлена еще какая-то часть потока в виде прибывших вне очереди дейтаграмм. Несмотря на это, получатель всегда посылает сигналы подтверждения приема только на самую большую и непрерывную часть потока (считая от его начала), которая была корректно получена. В каждом подтверждении приема указывается порядковый номер, который соответствует положению самого старшего октета в непрерывной части доставленного потока (точнее, больше его на единицу). Таким образом, по мере передачи потока данных отправитель постоянно получает сигналы обратной связи, отражающие состояние принятого потока данных. Исходя из сказанного выше можно сделать такой вывод.

В подтверждениях приема протокола TCP указывается порядковый номер в потоке следующего октета, который должен быть принят получателем.

Принятую в протоколе TCP систему подтверждения приема называют *кумулятивной (cumulative*), поскольку она отражает количество октетов потока данных, накопленных получателем. Следует отметить, что кумулятивная система подтверждения приема имеет как преимущества, так и недостатки. Одно из преимуществ состоит в том, что сигналы подтверждения приема легко генерируются и являются однозначными. Еще одно преимущество заключается в том, что в случае потери сигналов подтверждения приема не нужно повторно передавать ни их, ни соответствующие им сегменты данных. Недостатком описываемого кумулятивного метода является отсутствие у отправителя информации об успешных передачах. Ему только известно, какая часть потока была успешно доставлена получателю.

Чтобы понять, почему отсутствие информации об успешно выполненных передачах делает кумулятивный метод подтверждения приема менее эффективным, давайте рассмотрим такой пример. Предположим, существует окно, расположенное в потоке данных с позиции *101*, в которое включено *5000* октетов. Предположим также, что отправитель передал все находящиеся в окне данные в виде пяти сегментов. А теперь представим, что первый сегмент в процессе передачи был утерян, тогда как все остальные успешно достигли получателя. По мере прибытия сегментов, получатель будет отправлять сигналы подтверждения приема. В каждом из них будет указан порядковый номер октета *101*, т.е. номер следующего по порядку старшего октета, принадлежащего непрерывному потоку, который ожидает получить принимающая сторона. В данном случае у получателя нет никакой возможности сообщить отправителю, что большая часть данных, принадлежащих к текущему окну, уже получена.

Когда истечет время ожидания сигнала подтверждения приема, отправитель попытается выйти из сложившейся ситуации одним из двух потенциально неэффективных способов — повторно передать один сегмент или сразу все пять. Очевидно, что повторная передача пяти сегментов неэффективна. После того как первый сегмент будет успешно доставлен, у получателя будет полный комплект данных, составляющих окно. Поэтому он подтвердит прием октета с порядковым номером *5101*. Таким образом, если отправитель будет действовать согласно принятым стандартам и выполнит повторную передачу только первого не принятого сегмента, то прежде, чем предпринимать дальнейшие шаги, он должен дождаться подтверждения приема этого сегмента. В результате отправитель невольно возвратится к простейшему методу подтверждения приема, описанному в начале этого раздела. При этом все преимущества использования большого окна теряются.

**Время ожидания и повторная передача сегментов**

Одно из самых важных и сложных понятий протокола TCP — метод обработки истекшего времени ожидания и выполнения повторной передачи. Как и в других надежных протоколах, в протоколе TCP предполагается, что получатель пришлет сигнал подтверждения приема, успешно получив новые октеты из потока данных. Каждый раз при отправке сегмента в модуле протокола TCP устанавливается значение таймера ожидания получения сигнала подтверждения приема. Если время ожидания истечет, прежде чем будет подтвержден прием отправленного сегмента, модуль протокола TCP считает, что сегмент не достиг получателя или его данные были искажены в процессе передачи, и передает его повторно.

Чтобы понять, почему алгоритм повторной передачи протокола TCP отличается от аналогичных алгоритмов, используемых в других сетевых протоколах, необходимо вспомнить, что протокол TCP изначально предназначался для работы в межсетевом окружении. Передаваемый по объединенной сети между парой машин сегмент может проходить по одной сети с низким временем задержки (т.е. по высокоскоростной локальной сети) либо передаваться через несколько взаимодействующих между собой сетей, объединенных маршрутизаторами. Поэтому *заранее* невозможно предсказать, насколько быстро отправитель получит сигнал подтверждения приема. Ситуация усугубляется еще и тем, что задержка прохождения каждого маршрутизатора зависит от величины трафика в конкретной сети. Поэтому суммарное время задержки на передачу сегмента и получение отправителем подтверждения его приема может колебаться в очень широких пределах. Это проиллюстрировано на рис. 14.9, где изображены данные измерений полного (т.е. туда и обратно) времени доставки 100 последовательных пакетов по глобальной сети Internet. Из графика видно, что в программах протокола TCP должны учитываться два фактора: существенные отличия во времени передачи пакетов разным получателям и колебания времени задержки при доставке пакетов одному получателю, вызванные изменением трафика.

Рис. 14.9. Данные измерений полного времени доставки 100 последовательных IР-дейтграмм. Хотя в современной сети Internet задержки передачи пакетов намного ниже, разброс остается таким же большим

Для учета изменяющегося в широких пределах времени задержки в объединенной сети в протоколе TCP используется *адаптивный алгоритм повторной передачи {adaptive retransmission algorithm)*. В общих чертах суть его состоит в том, что в протоколе TCP отслеживается производительность каждого соединения и на основе полученных значений выбирается приемлемое время задержки. По мере изменения производительности соединения соответственно изменяется и величина задержки (т.е. величина задержки адаптируется к изменениям в производительности соединения).

При сборе данных, необходимых для работы адаптивного алгоритма, модуль протокола TCP регистрирует время отправки каждого сегмента и время получения сигналов подтверждения приема данных, находящихся в этих сегментах. По разности определяется время, затраченное на доставку каждого пакета. Описанная выше процедура называются *замером полного времени доставки пакета (sample round trip time)*. После очередного замера модуль протокола TCP пересчитывает среднее время доставки для данного соединения. Обычно модуль протокола TCP вычисляет предполагаемую средневзвешенную величину полного времени доставки пакетов *RTT (round trip time)*. При таком подходе вновь полученные результаты замеров мало влияют на среднее время доставки. Например, при вычислении новой средневзвешенной величины по одной из ранних методик использовалось постоянное значение весового коэффициента а, где 0<α<1. Это позволяло регулировать степень влияния новых результатов измерений на среднее значение полного времени доставки:

**RTT = (α × Старый*RTT) + ((1 - α) × Новый*RTT)**

Выбирая величину α близкой к 1, можно добиться того, чтобы новые результаты измерений, полученные за короткий промежуток времени, не влияли на средневзвешенную величину доставки пакетов. Это позволяет отсеять случайные отклонения значений от среднего, например, если один из сегментов доставлялся к получателю слишком долго. Если величину α выбрать близкой к 0, то средневзвешенная величина доставки пакетов будет достаточно быстро корректироваться в соответствии с текущими изменениями времени доставки пакетов.

При отправке пакета, модуль протокола TCP вычисляет величину тайм-аута как функцию от текущего значения предполагаемой величины полного времени доставки. В ранних реализациях протокола TCP использовалась линейная функция с постоянным весовым коэффициентом β (β>1). Это позволяло сделать величину тайм-аута большей, чем текущее значение предполагаемой величины полного времени доставки:

**Тайм-аут = β×RTT**

Выбрать величину β не так-то просто. С одной стороны, величина тайм-аута не должна сильно отличаться от текущего времени полной доставки пакетов (т.е. значение коэффициента β должно быть близким к 1). Это позволит быстро обнаруживать потери пакетов и благодаря этому увеличить пропускную способность сети, поскольку модулю протокола TCP не нужно будет понапрасну долго ожидать, пока истечет тайм-аут перед повторной передачей пакета. С другой стороны, при β=1 модуль протокола TCP может генерировать ненужный сетевой трафик, поскольку любая небольшая задержка в доставке пакета будет сразу же вызывать его повторную передачу. В результате пропуская способность сети снизится. В первоначальной спецификации стандарта TCP рекомендуется выбирать значение β=2. Однако проведенные недавно исследования (о них пойдет речь ниже) позволили выработать более точную методику выбора времени тайм-аута. Таким образом, можно подвести некоторые итоги.

Чтобы, учесть большой разброс задержек во времени доставки пакетов в реально действующей объединенной сети, в протоколе TCP используется адаптивный алгоритм повторной передачи. Суть его состоит в том, что величина тайм-аута выбирается в соответствии с текущим временем задержки доставки пакета, которое постоянно отслеживается во всех открытых соединениях.

**Точный замер полного времени доставки пакета**

Теоретически измерить полное время доставки пакета несложно: нужно из времени получения сигнала подтверждения доставки сегмента вычесть время отправки сегмента. Следует заметить, что при этом могут возникнуть определенные затруднения, поскольку в протоколе TCP используется кумулятивная система подтверждения приема. Суть ее заключается в том, что сигнал подтверждения приема отражает факт успешного получения данных, но не конкретной дейтаграммы, в которой эти данные были переданы. Рассмотрим подробнее процесс повторной передачи. Сначала модуль протокола TCP формирует сегмент, помещает его в дейтаграмму и отправляет ее получателю. После исчерпания значения таймера выполняется повторная передача сегмента, но уже в другой дейтаграмме. Поскольку в обеих дейтаграммах содержатся одни и те же данные, отправитель не может узнать, к какой из дейтаграмм (оригинальной или повторной) относится полученный сигнал подтверждения приема. Налицо явление, которое было названо *неоднозначностью сигналов подтверждения приема (acknowledgement ambiguity)*. Таким образом, сигналы подтверждения приема данных в протоколе TCP являются *неоднозначными*.

Проблема заключается в том, к какому из двух моментов передачи модулю протокола TCP следует привязать полученный сигнал подтверждения приема — к более раннему (т.е. оригинальному) или к более позднему (т.е. к тому, который был выполнен совсем недавно). Самое удивительное заключается в том, что нельзя принять ни одно из этих предположений. Если сигнал подтверждения приема соотнести с моментом передачи оригинальной дейтаграммы, то если эта дейтаграмма будет утеряна, предполагаемое полное время доставки пакета будет большим1.

Рано или поздно сигнал подтверждения приема будет получен, даже если для этого потребуется выполнить одну или несколько повторных передач сегмента. В результате модуль протокола TCP измерит полное время доставки сегмента относительно времени его первоначальной посылки и на основе этого чрезвычайно большого значения вычислит новое значение RTT. Таким образом, по сравнению со старым, новое значение RTT вырастет незначительно. В следующий раз, когда модуль протокола TCP будет отправлять сегмент получателю, увеличенное значение RTT приведет к заметному увеличению времени ожидания сигнала, подтверждения приема (тайм-аута). Поэтому, если сигнал подтверждения приема будет получен также после одной или нескольких повторных передач, полное время доставки пакета будет еще больше, и т.д.

Нельзя также соотносить сигнал подтверждения приема со временем самой последней повторной передачи сегмента. Давайте рассмотрим, что произойдет, если внезапно возрастет полное время доставки сегмента. При отправке сегмента модулем протокола TCP для вычисления времени тайм-аута будет использоваться старое (небольшое) значение предполагаемого полного времени доставки пакета. Предположим, что, после успешной доставки сегмента получателю, отправителю был послан сигнал подтверждения приема. Однако из-за перегрузок в сети сигнал подтверждения приема не будет получен до истечения значения таймера. В этом случае модуль протокола TCP выполнит повторную передачу сегмента. Вскоре после этого отправитель получит первый сигнал подтверждения приема и соотнесет его с моментом последней повторной передачи. Измеренное полное время доставки пакета будет маленьким, что приведет к небольшому снижению значения предполагаемого полного времени доставки пакета, или RTT. К сожалению, уменьшение значения RTT приведет к тому, что для передачи следующего сегмента модуль протокола TCP выберет малое значение тайм-аута. В конечном счете значение предполагаемого полного времени доставки пакета стабилизируется на величине *Т*, такой, что корректное значение полного времени доставки будет немного превышать значение *Т*, помноженное на некоторый коэффициент. Проведенные исследования показали, что в тех реализациях протокола TCP, где сигналы подтверждения приема соотносятся с моментом последней повторной передачи сегмента, стабильное значение RTT немного меньше половины корректного значения полного времени доставки. Следовательно, при отсутствии потерь сегментов в сети, модуль протокола TCP будет два раза посылать один и тот же сегмент получателю.

1. В худшем случае, когда утеряны все посланные сегменты, значение предполагаемого времени доставки увеличивается до очень больших значений.

**Алгоритм Карна и коррекция тайм-аута**

В предыдущем разделе шла речь о том, что независимо от того, к какому из моментов времени будет соотнесен полученный сигнал подтверждения приема (к передаче оригинального сегмента или к его повторной передаче), измерение полного времени доставки пакета будет неточным. Как же быть? Традиционный ответ на этот вопрос очень простой: модуль протокола TCP не должен пересчитывать значение предполагаемого полного времени доставки пакета (RTT) на основе данных, полученных при повторной передаче сегмента. Эта идея, называемая также *алгоритмом Карна (Karn's Algorithm)*, позволяет полностью избежать проблем, связанных с неоднозначностью сигналов подтверждения приема. Значение предполагаемого полного времени доставки пакета должно вычисляться только на основании данных, полученных для однозначных сигналов подтверждения приема (т.е. тех сигналов, которые были получены в ответ на однократную передачу пакета).

Естественно, что упрощенный вариант реализации алгоритма Карна (тот, в котором попросту игнорируются замеры, сделанные для повторно переданных сегментов) также не может использоваться для определения точного времени доставки сегмента.

Давайте рассмотрим, что произойдет, если в момент отправки сегмента модулем протокола TCP резко возрастет задержка в сети. Напомним, что значение тайм-аута вычисляется на основании текущего значения предполагаемого полного времени доставки пакета. Очевидно, что для больших задержек в сети вычисленное значение тайм-аута будет слишком малым. Это вызовет повторную передачу сегмента. Таким образом, если игнорировать сигналы подтверждения приема для повторно переданных сегментов, новое значение предполагаемого полного времени доставки пакета никогда не изменится, и описанный выше процесс будет продолжаться до тех пор, пока не уменьшится время задержки.

Чтобы устранить подобные недостатки, в алгоритме Карна используется метод *коррекции значения тайм-аута (timer backoff)*. Суть его состоит в том, что начальное значение тайм-аута вычисляется на основании текущих данных (по формуле, подобной приведенной выше). Однако, если в результате истечения тайм-аута произойдет повторная передача сегмента модуль протокола TCP увеличит значение тайм-аута. На практике, каждый раз перед повторной передачей сегмента, модуль протокола TCP увеличивает значение тайм-аута. Чтобы не допустить бесконтрольного увеличения тайм-аута, в большинстве реализаций оно ограничивается сверху заранее заданным значением, которое всегда больше максимально возможной задержки передачи пакета по любому из маршрутов объединенной сети.

Алгоритм вычисления нового значения тайм-аута зависит от конкретной реализации протокола TCP. В большинстве реализаций оно вычисляется путем умножения старого значения тайм-аута на специальный множитель γ:

**Новый тайм-аут = γ × Тайм-аут**

Обычно γ выбирается равным 2. (Выше уже шла речь о том, что при γ<2 система будет работать нестабильно.) В некоторых реализациях протокола TCP значение множителя выбирается из таблицы. Таким образом, значение нового тайм-аута будет непосредственно зависеть от номера шага1.

Для того чтобы решить проблему постоянства предполагаемого полного времени доставки пакета, в алгоритме Карна используется методика принудительного изменения тайм-аута.

Суть алгоритма Карна состоит в следующем. При вычислении предполагаемого полного времени доставки пакета игнорируются замеры, сделанные для повторно посланных сегментов. Для определения точного времени доставки пакетов в алгоритме Карна используется методика увеличения значения тайм-аута при повторной передаче сегмента.

Обобщая все сказанное выше можно отметить, что при возникновении больших задержек в объединенной сети, алгоритм Карна позволяет разделить вычисление текущего значения тайм-аута и определение предполагаемого полного времени доставки пакета. Предполагаемое значение полного времени доставки используется только для вычисления начального значения тайм-аута. При каждой повторной передаче сегмента значение тайм-аута увеличивается на некоторую величину до тех пор, пока сегмент не будет успешно доставлен получателю. Таким образом, при отправке последовательности сегментов используется значение тайм-аута, полученное в результате принудительной коррекции таймера. Так происходит до тех пор, пока не будет получен сигнал подтверждения приема, соответствующий однократно посланному сегменту. После этого модуль протокола TCP на основании сделанного замера пересчитывает предполагаемое значение полного времени доставки и соответствующим образом изменяет значение тайм-аута. Эксперименты показывают, что алгоритм Карна2 хорошо себя зарекомендовал даже в тех сетях, где потери пакетов очень велики.

1. Самой известной операционной системой, в которой используется таблица множителей, является Berkeley UNIX. Однако в текущих версиях системы в таблице хранятся одинаковые значения, эквивалентные γ=2.

2. Филипп Карн — радиолюбитель и энтузиаст своего дела. Он придумал этот алгоритм для того, чтобы можно было использовать протокол TCP для передачи данных по радиоканалу в условиях помех, когда происходят большие потери пакетов.

**Обработка большого разброса значений задержки**

Исследования, связанные с вычислением предполагаемого значения полного времени доставки пакетов показали, что описанные выше методы неприменимы в том случае, если значение времени задержки в сети колеблется в очень широких пределах. В теории массового обслуживания доказывается, что отклонение полного времени доставки пакетов а, изменяется пропорционально 1 / (1-*L*), где *L* — коэффициент текущей загрузки сети (0<*L*<1). Таким образом, при 50-процентной загрузке объединенной сети предполагаемое полное время доставки пакетов будет изменяться в пределах ± 2 ς, или пропорционально коэффициенту *4*. При 80-процентной загрузке сети коэффициент изменения будет равняться *10*.

Выше была описана методика определения предполагаемого значения полного времени доставки на основе информации, приведенной в первоначальном варианте стандарта протокола TCP. Таким образом, полагая значение β=2 (оно рекомендовано в первоначальном стандарте), получим, что описанная выше методика применима при загрузках сети, колеблющихся в районе 30%.

В спецификации протокола TCP 1989 года требуется, чтобы в реализациях протокола оценивалось как среднее время полной доставки пакетов, так и величины его отклонения. При этом предполагаемая величина отклонения должна использоваться в формуле вместо коэффициента β. В результате новые реализации протокола TCP могут работать в очень широком диапазоне задержек и позволяют существенно повысить пропускную способность сети.

Большим преимуществом описанного выше метода является то, что он не требует проведения сложных вычислений. Высокоэффективная программа может быть создана на основе нескольких простых формул, приведенных ниже.

**РАЗНОСТЬ = ЗАМЕР – Старый\_RТТ**

**Сглаженный*RTT = Старый*RTT + λ × РАЗНОСТЬ**

**ОТКЛ = Старое*ОТКЛ + ρ (| РАЗНОСТЬ | – Старое*ОТКЛ)**

**ТАЙМ-АУТ = Сглаженный\_RTT + ν × ОТКЛ**

Здесь:

* ОТКЛ — оценка среднего значения отклонения;
* λ — дробный коэффициент, значение которого лежит между *0* и *1*, определяющий насколько сильно значения новых замеров будут влиять на средневзвешенную величину *RTT*;
* ρ — дробный коэффициент, значение которого лежит между *0* и *1*, определяющий насколько сильно значения новых замеров будут влиять на оценку среднего значения отклонения;
* ν — коэффициент, определяющий, насколько сильно оценка среднего значения отклонения будет влиять на величину тайм-аута.

Для повышения скорости вычислений в протоколе TCP значения коэффициентов λ и ρ выбираются так, чтобы они были кратны 1/2n. Тогда для вычислений по приведенным выше формулам можно использовать целочисленную арифметику1. Опытным путем установлено, что приемлемый результат достигается при λ=1/23 и ρ=1/22. Первоначальное значение коэффициента ν в 4.3BSD UNIX равнялось *2*. Однако в версии 4.4 BSD UNIX оно было изменено на *4*.

На рис. 14.10 с помощью набора случайно выбранных величин продемонстрировано, как изменяется вычисленное значение тайм-аута в зависимости от разброса полного времени доставки пакета. Хотя данные для графика получены искусственным путем, они соответствуют реальной ситуации: при передаче последовательности пакетов наблюдается малый разброс задержек, а общее среднее время то увеличивается, то уменьшается.

Рис. 14.10. Изменение вычисленного значения тайм-аута в зависимости от разброса полного времени доставки пакета. Точками показан набор из 200 случайно выбранных значений полного времени доставки, а вычисленное значение тайм-аута изображено сплошной линией. Обратите внимание: увеличение времени задержки вызывает соответствующее увеличение значения тайм-аута

Частое изменение полного времени доставки (включая циклы увеличения и уменьшения) может вызвать увеличение среднего значения тайм-аута. Более того, хотя при увеличении задержек значение тайм-аута растет очень быстро, при их уменьшении это значение падает не так быстро.

Чтобы показать, как описанный алгоритм вычисления тайм-аута будет реагировать на чрезвычайно большие изменения задержки, мы воспользовались данными, приведенными на рис. 14.9 и построили график изменения значения тайм-аута (рис. 14.11).

Рис. 14.11. Значение тайм-аута, вычисленное с помощью алгоритма протокола TCP для данных, приведенных на рис. 14.9. Стрелками обозначены две последовательные дейтаграммы, при передаче которых величина задержки возросла в два раза

Напомним, что наша цель — вычислить текущее значение тайм-аута так, чтобы оно как можно ближе соответствовало реальному полному времени доставки пакета, и чтобы при этом не было недооценки.

Из рис. 14.11 видно, что, хотя значение тайм-аута довольно близко соответствует реальным данным, случаи недооценки полностью исключить нельзя. Например, во время передачи двух последовательных дейтаграмм, обозначенных стрелками, величина задержки возросла более чем в два раза (была менее 4-х секунд, а стала больше 8). Важно отметить, что внезапное большое изменение задержки произошло после периода относительной стабильности, когда разброс задержек был невелик. Поэтому нельзя применить какой-либо алгоритм, позволяющий предсказать это изменение.

В рассматриваемом нами случае применение алгоритма протокола TCP для вычисления тайм-аута (его значение равно 5 с) привело к существенной недооценке времени задержки. В результате будет выполнена ненужная повторная передача одного из сегментов. Тем не менее оценочное значение тайм-аута очень быстро “догоняет” возросшее значение задержки, а это означает, что все последующие пакеты будут доставлены получателю без повторной передачи.

1. Напомним, что для деления целого числа на 2n нужно сдвинуть его двоичное представление на *n* разрядов вправо, а для умножения — на *n* разрядов влево.

**Реакция на перегрузку сети**

На первый взгляд может показаться, что при разработке модуля протокола TCP нужно учитывать только механизм взаимодействия между двумя конечными точками соединения и задержки, возникающие при передаче пакетов между этими точками. Однако на практике модуль протокола TCP должен также учитывать *перегрузку*, возникающую в объединенной сети. О возникновении перегрузки свидетельствует резкое увеличение задержки при доставке пакетов, вызванное скоплением большого количества дейтаграмм на одном или нескольких устройствах коммутации (или маршрутизаторах). При перегрузке устройства маршрутизации задержка увеличивается, поскольку поступающие дейтаграммы ставятся в очередь и находятся там до тех пор, пока маршрутизатор не сможет их обработать.

Не стоит забывать, что емкость запоминающего устройства каждого маршрутизатора имеет ограниченный объем и что каждая дейтаграмма занимает часть этой памяти. Другими словами, при передаче дейтаграмм по объединенной сети никакие ресурсы памяти заранее не выделяются для отдельных TCP-соединений. В самом худшем случае, когда на перегруженный маршрутизатор поступает такое количество дейтаграмм, что они не могут разместиться в его памяти, все “лишние” дейтаграммы попросту теряются.

Как правило, конечным точкам соединения не известно, в каком месте объединенной сети и по какой причине возникла перегрузка. Для них перегрузка просто означает увеличение задержки. К сожалению, в большинстве транспортных протоколов используется стратегия повторной передачи пакетов в случае истечения времени ожидания сигнала подтверждения приема. Поэтому увеличение задержки вызывает повторную передачу дейтаграмм, что, в свою очередь, еще больше усугубляет ситуацию. Если ничего не предпринять, то увеличение трафика вызовет увеличение задержки, а увеличение задержки снова приведет к увеличению трафика в сети, и т.д. до тех пор, пока работа сети не будет полностью парализована. Подобная ситуация называется *полным затором в сети (congestion collapse)*.

Чтобы избежать полного затора в сети, при возникновении перегрузки модуль протокола TCP должен уменьшить интенсивность передачи пакетов. Устройства маршрутизации постоянно следят за длиной внутренней очереди дейтаграмм1 и в случае ее переполнения сообщают о возникшей перегрузке всем своим отправителям с помощью механизма, напоминающего рассылку ICMP-сообщений о подавлении источника данных. Однако протоколы транспортного уровня, наподобие TCP, также могут регулировать степень перегрузки в сети, автоматически снижая интенсивность передачи данных при возникновении задержек.

Естественно, что применяемые при этом алгоритмы должны быть тщательно продуманы, поскольку даже при нормальных условиях полное время доставки пакета в объединенной сети может колебаться в очень широких пределах.

Для предотвращения перегрузки в сети современная версия стандарта TCP рекомендует использовать две методики: *медленный запуск (slow-start)* и *мультипликативное уменьшение (multiplicative decrease)*. Обе методики взаимосвязаны и могут быть легко реализованы. Выше уже говорилось о том, что для каждого открытого соединения модуль протокола TCP хранит размер окна получателя (т.е. размер буфера, анонсированного в сигнале подтверждения приема).

Для недопущения перегрузки в протоколе TCP установлено еще одно ограничение, которое называется *ограничением размера окна перегрузки* или просто *окном перегрузки*. При возникновении перегрузки оно ограничивает поток данных настолько, чтобы он был меньше размера приемного буфера. Таким образом, в любой момент времени модуль протокола TCP функционирует так, как если бы размер его окна был равен:

**Размер*окна = Мин*размер (Анонс*получателя. Окно*перегрузки)**

В стационарном режиме при неперегруженном соединении размер окна перегрузки совпадает с размером окна получателя. При уменьшении размера окна перегрузки уменьшается поток данных, передаваемый модулем протокола TCP через конкретное соединение. Чтобы определить ориентировочный размер окна перегрузки, модуль протокола TCP считает, что большая часть дейтаграмм теряется из-за перегрузки. При этом используется описанная ниже стратегия.

Стратегия мультипликативного уменьшения заключается в том, что каждый раз после возникновения перегрузки размер окна перегрузки уменьшается в два раза (вплоть до минимума, составляющего одну дейтаграмму). Для тех сегментов, которые попали в окно нового размера, применяется стратегия экспоненциального увеличения значения таймера повторной передачи (тайм-аута).

Поскольку в протоколе TCP при *каждой* потере пакета размер окна перегрузки уменьшается в два раза, размер основного окна передачи уменьшается по экспоненциальному закону в случае продолжения потери пакетов. Другими словами, при возникновении перегрузки модуль протокола TCP уменьшает величину потока данных в сети по экспоненциальному закону. При этом интенсивность повторной передачи сегментов также снижается экспоненциально. При продолжительной перегрузке модуль протокола TCP в конечном счете ограничивает интенсивность передачи данных до одной дейтаграммы и удваивает значение тайм-аута перед повторной передачей сегмента. Идея состоит в том, чтобы в критических ситуациях система могла резко уменьшить величину потока данных в сети.

При этом маршрутизаторам будет предоставлено достаточно времени на проверку и аннулирование “лишних” дейтаграмм, находящихся во внутренней очереди.

Теперь рассмотрим, как происходит восстановление работоспособности системы на основе протокола TCP после устранения перегрузки. Логично предположить, что все должно выполняться в обратном, по сравнению с методом мультипликативного уменьшения порядке. Т.е., как только в сети нормализуется скорость передачи потока данных, размер окна перегрузки должен удваиваться. Однако следствием подобных действий может стать нестабильность работы системы: ее состояние будет постоянно изменяться в широких пределах — от перегруженности до полного отсутствия трафика. Поэтому в протоколе TCP для постепенного увеличения интенсивности передачи данных используется так называемый метод *медленного запуска2* (*slow-start*), суть которого состоит в следующем.

Методика медленного запуска предназначена для восстановления работоспособности сети после перегрузки, а также для начала передачи данных по новому соединению. При этом первоначальный размер окна перегрузки выбирается равным одному сегменту, и каждый раз после получения сигнала подтверждения приема он увеличивается на один сегмент.

Таким образом, метод медленного запуска позволяет избежать затора в объединенной сети сразу после прекращения перегрузки или при начале передаче данных по новому соединению.

Описанный выше термин *”медленный запуск”* может ввести в заблуждение, поскольку в идеальных условиях запуск механизма передачи данных происходит не так уж медленно. Первоначально размер окна перегрузки устанавливается равным *1* сегменту, после чего выполняется передача этого сегмента, и система переходит в состояние ожидания. После получения сигнала подтверждения приема этого сегмента, размер окна перегрузки устанавливается равным *2* сегментам, после чего посылается уже *2* сегмента и система снова переходит в состояние ожидания. После получения каждого из двух сигналов подтверждения приема этих сегментов, размер окна перегрузки снова увеличивается на *1* сегмент. Поэтому модуль протокола TCP может отправить уже *4* сегмента. После получения *4* сигналов подтверждения приема размер окна перегрузки будет равен уже *8* сегментам. Таким образом, после завершения четырех циклов передачи-приема, модуль протокола TCP может отправить уже *16* сегментов, что часто превышает размеры приемного окна получателя пакетов. Таким образом, перед тем, как модуль протокола TCP сможет отправить N сегментов, он должен выполнить *log2N* циклов передачи-приема. Очевидно, что даже в случае очень больших размеров окон выход системы на проектную мощность будет довольно быстрым.

Чтобы не допустить слишком быстрого увеличения размеров окна и возникновения дополнительной перегрузки, в протокол TCP введено дополнительное ограничение. Как только размер окна перегрузки достигнет половины своего первоначального значения (того, что было до возникновения перегрузки), модуль протокола TCP переходит к фазе *аннулирования перегрузки* и снижает скорость нарастания размеров этого окна. Во время этой фазы размер окна перегрузки увеличивается на *1* только в том случае, если для всех сегментов, находящихся в окне, будут получены сигналы подтверждения приема.

Подводя итоги всему сказанному, необходимо отметить, что методы медленного запуска, мультипликативного уменьшения, фаза аннулирования перегрузки, измерение отклонения полного времени доставки и экспоненциальное изменение значения тайм-аута позволяют существенно поднять производительность системы на основе протокола TCP без выполнения сложных математических расчетов внутри программы, реализующей функции этого протокола. В тех версиях протокола TCP, где применены все описанные методы, удалось поднять производительность системы от *2* до *10* раз по сравнению с реализациями, где одна или несколько методик не использовались.

1. В перегруженной сети длина очереди дейтаграмм с течением времени возрастает по экспоненциальному закону.

2. Этот термин приписывают Джону Нейглу (John Nagle). Первоначально этот метод восстановления работоспособности системы назывался плавным запуском (soft-start).

**Перегрузка сети и усечение хвоста очереди**

Выше уже отмечалось, что коммуникационные протоколы разделены на логические уровни. Это облегчает их разработку, поскольку инженеры могут сосредоточиться на решении какой-то одной проблемы. Кроме того, разделение выполняемых функций по уровням является очень полезным и нужным делом. Это означает, что в программу, реализующую функции одного уровня, можно легко вносить изменения. При этом программы других уровней остаются без изменений.

Однако в методе разделения есть и негативная сторона — работа программ всех уровней осуществляется изолированно друг от друга. Например, протокол TCP ориентирован на обмен данными между двумя конечными точками соединения. Поэтому его работоспособность сохраняется при изменении маршрута следования дейтаграмм по объединенной сети между этими точками (т.е. при изменении или добавлении дополнительных сетевых маршрутов).

Несмотря на это изоляция уровней непосредственно сказывается на возможности взаимодействия между ними. В частности, хотя модуль протокола TCP машины отправителя может взаимодействовать с модулем протокола машины получателя, он не может взаимодействовать с модулями протоколов более низкого уровня, находящимися на пути следования пакетов.

Таким образом, модули протокола TCP отправителя и получателя никогда не смогут получить отчет о текущем состоянии сети, а также проинформировать модули протоколов низкого уровня, находящиеся на пути следования пакетов о начале передачи данных.

Разработчики заметили одну интересную особенность: отсутствие средств взаимодействия между уровнями часто приводит к тому, что изменение алгоритма работы или программы реализации на одном из уровней кардинальным образом влияет на производительность более высоких уровней.

В случае протокола TCP от алгоритмов, которые используют маршрутизаторы для обработки дейтаграмм, в значительной степени зависит как производительность одного TCP-соединения, так и суммарная пропускная способность всех соединений. Например, если при обработке одних дейтаграмм в маршрутизаторе будет возникать большая задержка1, чем при обработке других, то это приведет к увеличению тайм-аута повторной передачи протокола TCP. Если эта задержка превысит величину тайм-аута, модуль протокола TCP “решит”, что в сети возникла перегрузка. Поэтому, несмотря на то, что стандарты протоколов каждого уровня определены независимо от стандартов остальных уровней, разработчики попытались продумать и реализовать механизм взаимодействия между протоколами разных уровней.

Одно из основных взаимодействий между модулями протоколов IP и TCP происходит в случае перегрузки маршрутизатора, следствием которой является потеря дейтаграмм. Поскольку маршрутизатор помещает каждую вновь прибывшую дейтаграмму в очередь на обработку, находящуюся в оперативной памяти, основное внимание инженеров было сосредоточено на алгоритмах манипуляции элементами этой очереди. Если скорость поступления дейтаграмм превышает скорость их обработки в маршрутизаторе, размер очереди будет постоянно увеличиваться. Если же маршрутизатор перенаправляет дейтаграммы быстрее, чем они поступают, размер очереди сокращается. Поскольку объем оперативной памяти в маршрутизаторе ограничен, размер очереди не может увеличиваться до бесконечности. Поэтому в ранних версиях программного обеспечения для маршрутизаторов при переполнении очереди использовалась стратегия *усечения ее хвоста (tail-drop)*.

Стратегия усечения хвоста очереди заключается в том, что в случае переполнения входной очереди маршрутизатора, все вновь прибывшие дейтаграммы аннулируются.

Название *усечение хвоста* происходит от эффекта, возникающего при обработке последовательности прибывающих дейтаграмм в маршрутизаторе. При переполнении очереди маршрутизатор начинает аннулировать все вновь поступающие дейтаграммы. Другими словами, маршрутизатор усекает “хвост” последовательности прибывающих дейтаграмм.

Усечение хвоста очереди оказывает необычный эффект на работу протокола TCP. В простейшем случае, когда в проходящих через маршрутизатор дейтаграммах содержатся сегменты, относящиеся к одному TCP-соединению, потеря дейтаграмм заставляет модуль протокола TCP применять методику медленного запуска, о которой шла речь выше. В результате производительность ТСР-соединения падает до тех пор, пока не начнут приходить сигналы подтверждения приема и увеличиваться размер окна перегрузки.

Если через маршрутизатор проходят дейтаграммы, относящиеся к разным TCP-соединениям, применение методики усечения хвоста очереди приводит к более серьезным проблемам. Так происходит потому, что аннулирование последних прибывающих дейтаграмм может привести к эффекту глобальной синхронизации. Чтобы понять суть проблемы, следует отметить, что обычно дейтаграммы поступают на маршрутизатор вперемешку (т.е. друг за другом могут быть получены дейтаграммы от разных отправителей). Поэтому усечение хвоста очереди в этом случае вызывает потерю одного сегмента в каждом из *N* соединений, а не потерю *N* сегментов, относящихся к одному соединению. Потеря сегментов во всех *N* соединениях заставляет модули протоколов TCP этих соединений одновременно переходить к медленному запуску.

1. Формально разброс значений задержек называют *дрожанием (jitter)*.

**Произвольное раннее обнаружение (RED)**

Как же избежать эффекта глобальной синхронизации в маршрутизаторе? Для этого необходим алгоритм, который позволит избежать усечения хвоста очереди там, где это возможно. И такой алгоритм был найден. Его называют по разному: *произвольное раннее аннулирование (Random Early Discard), произвольное раннее удаление (Random Early Drop)*, или *произвольное раннее обнаружение (Random Early Detection)*. Но чаще всего используют сокращение *RED*.

В маршрутизаторе, где используется методика RED, задаются два пороговых значения, относящихся к положению дейтаграммы в очереди: Тmin и Тmax.

В общих чертах работу алгоритма RED можно описать тремя правилами, с помощью которых определяется положение в очереди каждой из вновь прибывающих дейтаграмм.

* Если в настоящий момент в очереди находится меньше, чем Тmin дейтаграмм, новая дейтаграмм просто добавляется в конец очереди.
* Если в очереди находится больше, чем Тmax дейтаграмм, то все новые дейтаграммы аннулируются.
* Если число дейтаграмм в очереди колеблется между Тmin и Тmax, то аннулируется одна из поступивших дейтаграмм, выбранная случайным образом в соответствии с заданной долей вероятности *р*.

Стратегия случайного выбора, заложенная в RED, позволяет маршрутизатору по мере увеличения перегрузки переходить к постепенному и случайному удалению дейтаграмм. В этом ее коренное отличие от алгоритма усечения хвоста очереди, при использовании которого в случае переполнения очереди большое количество TCP-соединений переводится в состояние медленного запуска. Таким образом, можно повести некоторый итог.

Суть метода RED, используемого в маршрутизаторах состоит в следующем. Если входная очередь дейтаграмм переполнена, то все вновь поступающие дейтаграммы аннулируются. Если же входная очередь не заполнена до конца, но ее размер превышает заранее установленный минимальный порог, то во избежание эффекта глобальной синхронизации, аннулируется одна из поступивших дейтаграмм, выбранная случайным образом в соответствии с заданной долей вероятности *р*.

Ключом к эффективной работе метода RED является правильный выбор пороговых значений Тmin и Тmax, а также величины вероятности потери *р*. Значение Тmin должно быть достаточно большим, чтобы обеспечить высокую пропускную способность выходного канала связи. Кроме того, поскольку в случае, когда размер очереди превосходит Тmax, алгоритм RED работает также как и алгоритм усечения хвоста очереди, значение Тmax должно быть больше Тmin по крайней мере на среднюю величину увеличения размера очереди за время одной полной доставки ТСР-сегмента (т.е. значение Тmax должно быть по меньшей мере в два раза больше, чем Тmin). В противном случае использование методики RED может привести к тем же глобальным колебаниям трафика в сети, что и при использовании метода усечения хвоста очереди.

Самой сложной задачей в алгоритме RED является нахождение величины вероятности потери дейтаграммы *р*. Очевидно, что величина *р* не может быть постоянной и должна вычисляться заново для каждой из вновь прибывших дейтаграмм. Ее значение зависит от текущего размера очереди и установленных пороговых значений Тmin и Тmax.

Идея будет понятнее, если описать алгоритм, используемый в RED, с вероятностной точки зрения. Когда размер очереди меньше Тmin, потери дейтаграмм не происходит. Поэтому можно считать, что в этом случае значение *р* равно нулю. Когда же размер очереди превышает Тmax, все дейтаграммы теряются. Поэтому в последнем случае значение *р* равно единице. Если размер очереди колеблется между Тmin и Тmax, значение вероятности потери дейтаграммы р можно изменять по линейному закону от *0* до *1*.

Описанный выше линейный алгоритм вполне можно положить в основу используемого в RED метода вычисления вероятности потери дейтаграммы *р*. Однако в него следует внести изменения для того, чтобы система не так остро реагировала на изменение загрузки сети. Это вызвано тем, что в реально действующей сети значение сетевого трафика все время колеблется в очень широких пределах. В результате размер внутренней очереди маршрутизатора также колеблется с очень высокой скоростью.

Если в RED используется упрощенный линейный метод, то вероятность потери последних поступивших в одном блоке, вызвавшем перегрузку, дейтаграмм будет очень высокой. Так происходит потому, что дейтаграммы поступают на маршрутизатор в тот момент, когда размер его внутренней очереди становится очень большим.

Как бы то ни было, маршрутизатор не должен без крайней необходимости аннулировать полученные дейтаграммы, поскольку это отрицательно сказывается на производительности ТСР-соединения. Поэтому при возникновении кратковременной перегрузки в сети незачем удалять дейтаграммы, поскольку внутренняя очередь маршрутизатора никогда не переполнится. С другой стороны, нельзя бесконечно долго оттягивать процесс удаления дейтаграмм, поскольку в случае возникновения затяжной перегрузки очередь переполнится очень быстро. Тогда пропадет весь эффект от использования алгоритма RED, и наверняка возникнут проблемы с глобальной синхронизацией, описанные выше.

Как же назначить в алгоритме RED более высокую вероятность потери р по мере увеличении размера внутренней очереди и при этом сделать так, чтобы без крайней необходимости не аннулировались дейтаграммы в блоках, вызвавших перегрузку? Для ответа на этот вопрос необходимо вспомнить метод, использовавшийся в протоколе TCP. Суть его состоит в том, что при вычислениях в каждый момент времени не стоит использовать реальный размер очереди. Вместо этого, в алгоритме RED определяется средневзвешенное значение размера внутренней очереди *avg*, которое используется для вычисления вероятности потери дейтаграмм.

Величина средневзвешенного значения *avg* изменяется по экспоненциальному закону и корректируется каждый раз при получении очередной дейтаграммы. При этом используется следующая формула:

**avg = (1 - γ) × Старое*avg + γ × Текущий*размер\_очереди**

В этой формуле значение коэффициента γ находится в интервале между *0* и *1*. Если значение γ достаточно мало, средневзвешенное значение *avg* будет отслеживать тенденции, происходящие на большом интервале времени. При этом оно будет мало восприимчиво1 к кратковременным перегрузкам сети.

Кроме формулы, по которой определяется значение γ, в алгоритме RED есть и другие тонкости, в которые мы не будем вдаваться. Например, процесс вычислений можно сделать чрезвычайно эффективным, если выбрать значения чисел кратными 2n и использовать целочисленную арифметику.

Еще одна тонкость состоит в способе измерения размера очереди, который влияет как на процесс вычисления по алгоритму RED, так и на общую производительность протокола TCP. В частности, поскольку время, необходимое для перенаправления дейтаграммы, прямо пропорционально ее длине, то имеет смысл измерять размер очереди не в количестве находящихся в ней дейтаграмм, а в октетах. Тогда в формулы, по которым определяется значение коэффициентов *р* и γ, нужно внести небольшие изменения.

Измерение размера очереди в октетах влияет также на тип трафика, который будет аннулироваться при перегрузке сети. Все дело в том, что в этом случае вероятность потери будет пропорциональна количеству данных, а не количеству сегментов, помещенных отправителем в поток. Поэтому, чем меньше размер дейтаграммы, тем ниже будет вероятность ее потери.

Следовательно, небольшие дейтаграммы (например, те, в которых находятся данные сеанса регистрации на удаленном компьютере или запросы к серверу) будут проходить к получателю практически без потерь, тогда как дейтаграммы большого размера (например, те, в которых находятся данные передаваемого файла) с высокой долей вероятности будут аннулироваться. Из этого следует положительный вывод: при прохождении сигналов подтверждения приема по перегруженному маршруту вероятность их потери будет низкой. Если будет получен сегмент данных (большого размера), то отправитель практически всегда получит сигнал подтверждения его приема, что позволит избежать ненужной повторной передачи этого сегмента.

В результате моделирования работы алгоритма RED и исследования его поведения в реальных условиях было выявлено много преимуществ этого метода. При перегрузке сети он позволяет избежать эффекта глобальной синхронизации, который возникает при применении метода усечения хвоста очереди. Кроме того, при кратковременной перегрузке метод RED без крайней необходимости не аннулирует дейтаграммы.

В настоящее время группа IETF рекомендует использовать в программном обеспечении маршрутизаторов метод RED.

1. Для примера можно положить γ=0,002.

**Установка TCP-соединения**

Для установки соединения в протоколе TCP используется *трехэтапный метод квитирования (three-way handshake)*. В простейшем случае процесс квитирования происходит так, как показано на рис. 14.12.

Рис. 14.12. Последовательность сообщений, посылаемых во время трехэтапного метода квитирования. Ось времени отложена по вертикали и направлена вниз. Диагональные линии отображают процесс пересылки сегментов между узлами сети. В сегменте типа SYN пересылается начальный порядковый номер

Первый сегмент, посылаемый в процессе квитирования, легко распознать, поскольку в поле кода сегмента его заголовка установлен бит SYN1. В заголовке второго сегмента устанавливаются сразу два бита: SYN и АСК. Это означает, что в нем находится сигнал подтверждения приема первого сегмента SYN и данные, предназначенные для продолжения процесса квитирования. В заголовке последнего сегмента устанавливается только бит АСК. Этот сегмент используется только для информирования получателя о том, что обе стороны уведомлены об установке соединения.

Обычно инициатором установки соединения выступает модуль протокола, запущенный на одной из машин. При этом модуль, запущенный на другой машине, просто ожидает момента начала процесса квитирования. Тем не менее при квитировании предусмотрена ситуация, когда обе машины одновременно выступят инициаторами начала соединения.

Таким образом, TCP-соединение может быть установлено как по инициативе одной из сторон, так и двух сторон сразу. После установки соединения данные могут течь в обоих направлениях совершенно одинаково. Другими словами, обе стороны ТСР-соединения являются равноправными, при этом нет ни основного, ни подчиненного устройства.

Трехэтапный метод квитирования является необходимым и достаточным условием выполнения успешной синхронизации между двумя сторонами соединения. Чтобы понять это, необходимо вспомнить, что в протоколе TCP используется ненадежная служба доставки пакетов. А это означает, что сообщения могут быть потеряны, задержаны, продублированы или доставлены с нарушением порядка их следования.

Для решения перечисленных проблем в протоколе TCP используется механизм повторной передачи утерянных пакетов, который задействуетcя после истечения времени ожидания сигналов подтверждения приема.

Серьезная проблема возникает, если в процессе установки соединения получателю приходят два запроса (первоначальный и повторный) либо когда повторный запрос приходит с большой задержкой (т.е. после того, как соединение было установлено, использовано и разорвано).

Перечисленные выше проблемы и позволяет решить метод трехэтапного квитирования. Кроме того, в протоколе TCP принято правило, согласно которому после установки соединения все дополнительные запросы на его установку попросту игнорируются.

1. Название бита SYN происходит от английского слова “synchronization”, т.е. синхронизация и произносится “син”.

**Начальный порядковый номер**

Трехэтапный метод квитирования выполняет две важные функции. Во-первых, он гарантирует, что обе стороны соединения будут готовы к приему данных и что о готовности одной стороны будет знать другая сторона. Во-вторых, с его помощью выполняется процесс согласования начального порядкового номера.

Во время квитирования обе стороны соединения обмениваются начальными порядковыми номерами и ожидают подтверждения их приема. Порядковые номера используются для идентификации потоков данных, посылаемых каждой из сторон открытого соединения. Они выбираются сторонами самостоятельно (обычно случайным образом) во время открытия соединения.

Порядковые номера не могут всегда начинаться с одного и того же значения. В частности, модуль протокола TCP не может каждый раз при создании нового соединения назначать порядковый номер, равный *1* (в конце раздела вам предлагается самостоятельно ответить на вопрос, почему именно не может). Естественно, что при открытии соединения обе стороны должны согласовать свои порядковые номера. Это делается для того, чтобы номера октетов, указываемые в сигналах подтверждения приема, соответствовали номерам, используемым в заголовках при передаче сегментов данных.

Чтобы понять, как стороны, обменявшись всего тремя сообщениями, могут согласовать порядковые номера для двух потоков, необходимо вспомнить, что в заголовке любого сегмента содержится как поле порядкового номера, так и поле номера сигнала подтверждения приема.

Предположим, что машина А является инициатором соединения. Тогда во время трехэтапного процесса квитирования в первом SYN-сегменте она передает второй стороне (т.е. машине Б) свой начальный порядковый номер *х*. Получив SYN-сегмент, машина Б сохраняет в памяти начальный порядковый номер машины А, после чего посылает ей ответный сегмент.

В одноименном поле его заголовка она указывает уже свой начальный порядковый номер *у*, а в поле номера сигнала подтверждения приема — значение *х+1*. Тем самым машина Б уведомляет машину А, что ее начальный порядковый номер успешно получен, и что машина Б ожидает получить от машины А поток октетов, начинающийся с номера *х+1*.

В третьем (последнем) сегменте процесса квитирования машина А подтверждает получение порядкового номера машины Б и сообщает ей, что она ожидает получить от машины Б поток октетов, начинающийся с номера *у – 1*. В обоих случаях номера в сигналах подтверждения приема соответствуют принятому в протоколе TCP соглашению о *следующем* ожидаемом сегменте.

Выше было описано, как в протоколе TCP выполняется процесс трехэтапного квитирования с помощью обмена сегментами, содержащими минимальное количество информации.

Протокол TCP спроектирован так, что в нем возможна передача данных вместе с начальным порядковым номером непосредственно в сегментах квитирования. В подобных случаях модуль протокола TCP должен заблокировать данные до завершения процесса квитирования. После того как соединение установлено, модуль протокола TCP может разблокировать данные и доставить их очень быстро ожидающему приложению. Подробнее об этом можно прочитать в спецификации протокола TCP.

**Закрытие ТСР-соединения**

Две взаимодействующие по протоколу TCP программы могут корректно завершить сеанс соединения с помощью так называемой операции *закрытия (close)*. В протоколе для этой цели используется модифицированный метод трехэтапного квитирования.

Напомним, что TCP-соединение является дуплексным. Поэтому его можно использовать для передачи двух независимых потоков данных в противоположных направлениях. Получив от прикладной программы сообщение о том, что все данные переданы, модуль протокола TCP закрывает соединение *в одном направлении*. Перед тем как закрыть половину соединения, модуль протокола должен отправить получателю все находящиеся в буфере данные и подождать подтверждения их приема. После этого получателю отправляется сегмент, в заголовке которого устанавливается бит FIN. После того, как FIN-сегмент достигнет машины получателя, ее модуль протокола посылает отправителю сигнал подтверждения приема и уведомляет свою прикладную программу о том, что отправителем переданы все данные. Для этой цели могут использоваться стандартные средства операционной системы, например устанавливаться признак конца файла при очередной попытке считать данные из открытого соединения.

Закрыв соединение в заданном направлении, модуль протокола TCP отвергает все попытки прикладной программы передать данные в этом направлении. Тем не менее в противоположном направлении данные могут передаваться до тех пор, пока отправитель не закроет вторую половину соединения. Естественно, что даже после закрытия первой половины соединения, по ней все равно будут передаваться отправителю сигналы подтверждения приема. После закрытия соединения в обоих направлениях модули протокола TCP машин отправителя и получателя удаляют из своих системных таблиц записи об этом соединении.

Следует заметить, что на самом деле закрытие соединения происходит несколько сложнее, чем было описано выше. Все дело в том, что для этой цели в протоколе TCP используется модифицированный метод трехэтапного квитирования, работа которого показана на рис. 14.13.

Рис. 14.13. Иллюстрация принципа работы модифицированного метода трсхэтапного квитирования, используемого для закрытия TCP-соединения. После получения первого FIN-сегмента сразу же посылается сигнал, подтверждающий его прием. Второй FIN-сегмент посылается отправителю с некоторой задержкой, связанной с закрытием прикладной программой соединения

Отличие модифицированного трехэтапного квитирования, используемого для разрыва соединения, от аналогичного метода, применяемого для установки соединения, состоит в следующем. После получения первого FIN-сегмента второй FIN-сегмент посылается отправителю не сразу (в отличие от SYN-сегмента). Отправителю посылается сигнал подтверждения приема первого FIN-сегмента, после чего запущенная на машине получателя прикладная программа уведомляется о получении запроса на закрытие соединения. От момента уведомления прикладной программы до получения от нее отклика может пройти довольно много времени (например, если для закрытия соединения потребуется вмешательство пользователя программы). Отправка сигнала подтверждения приема первого FIN-сегмента позволяет исключить повторную передачу этого сегмента отправителем по истечении тайм-аута. Получив от прикладной программы, запущенной на машине получателя, команду на закрытие соединения, модуль протокола TCP отошлет отправителю второй FIN-сегмент. При этом отправитель должен прислать уведомление о получении этого сегмента (третий АСК-сегмент).

**Сброс ТСР-соединения**

Обычно после окончания передачи данных прикладная программа закрывает соединение с помощью специальной команды. Поэтому данную операцию следует рассматривать как неотъемлемую часть нормального функционирования системы (по аналогии с операцией закрытия обычного файла). Однако иногда обстоятельства могут сложиться так, что прикладная программа или сетевое программное обеспечение вынуждены разорвать соединение. Поэтому в протоколе TCP предусмотрены средства обработки подобных нештатных ситуаций (сброс соединения).

Для сброса соединения одна из сторон должна инициировать процесс досрочного прекращения передачи данных путем отправки сегмента, в заголовке которого в поле кода установлен бит RST. При получении подобного запроса другая сторона должна немедленно разорвать соединение. При этом о поступившем запросе на сброс соединения модуль протокола TCP уведомляет также прикладную программу. Таким образом, операция сброса соединения приводит к немедленному прекращению передачи данных в обоих направлениях и освобождению всех занятых ресурсов, например буферов памяти.

**Протокол TCP и теория конечных автоматов**

Подобно большинству других протоколов, функционирование протокола TCP может быть описано с помощью специальной модели, принятой в теории конечных автоматов. На рис. 14.14 показан конечный автомат протокола TCP. На нем состояние автомата отображено в виде окружностей, а переходы из состояния в состояние показаны стрелками. Надписи на каждой стрелке позволяют показать, какая информация должна быть получена протоколом TCP для перехода системы в другое состояние и что модуль протокола должен отослать в ответ. Например, рассмотрение функционирования программ протокола TCP, находящихся на каждой из сторон соединения, начинается с состояния, которое называется *ЗАКРЫТЫМ*.

Рис. 14.14. Модель конечного автомата протокола TCP. В исходном состоянии каждая из сторон соединения находится в закрытом состоянии. Надписи на стрелках перехода соответствуют входным данным, которые вызывают изменение состояние системы, а также выходным данным, которые генерирует система (при наличии)

Для вывода системы из этого состояния одна из прикладных программ должна выполнить команду *пассивного открытия* соединения (если она собирается ожидать, пока другая машина установит с ней соединение) или команду *активного открытия* соединения (если нужно инициировать соединение). Команда активного открытия соединения вызывает переход системы из *ЗАКРЫТОГО* состояния в состояние сигнал *SYN ПОСЛАН*. Перейдя в это состояние, модуль протокола TCP отправляет получателю SYN-сегмент. Как только от противоположной стороны будет получен сегмент, в заголовке которого установлены биты SYN и АСК, модуль протокола TCP переходит в состояние *УСТАНОВЛЕНО* и начинает передачу данных.

Состояние *ОЖИДАНИЯ ТАЙМ-АУТА* показывает, как в протоколе TCP обрабатываются ситуации, связанные с ненадежной доставкой. Для этой цели в нем предусмотрена специальная переменная, называемая *максимальным временем жизни сегмента (maximum segment lifetime*, или *MSL*). ЭТО время определяет, как долго сегмент может находиться в объединенной сети.

Протокол TCP переходит в состояние *ОЖИДАНИЯ ТАЙМ-АУТА* сразу после закрытия соединения. Это сделано для того, чтобы сегменты, относящиеся к предыдущему соединению, не влияли на текущее соединение. В таком состоянии протокол TCP должен находиться определенное время, которое равно удвоенному значению максимального времени жизни сегмента, после чего из системных таблиц удаляется запись, относящаяся к текущему соединению.

Если во время ожидания тайм-аута к получателю поступит один из сегментов-дубликатов, относящихся к текущему соединению, он будет проигнорирован. Однако, чтобы обработать ситуацию возможной потери последнего сигнала подтверждения, модуль протокола TCP отправляет сигналы подтверждения на все успешно полученные сегменты и перезапускает таймер. Поскольку таймер предназначен для различения сегментов, относящихся к старым и новым соединениям, он не позволяет модулю протокола TCP отвечать RST-сегментами на посланные повторно FIN-запросы.

**Принудительная доставка данных**

Выше уже шла речь о том, что в протоколе TCP деление потока данных на сегменты, предназначенные для передачи по сети, происходит без учета количества данных, переданных модулю протокола прикладной программой. Разработчики поступили так из соображений эффективности. Это позволяет модулю протокола накапливать во внутреннем буфере необходимое для эффективной передачи количество данных. Благодаря этому существенно снижаются накладные расходы при передаче по сети сегментов небольшого размера.

Несмотря на то, что буферизация повышает пропускную способность сети, она может повлиять на работу некоторых прикладных программ.

Давайте рассмотрим процесс передачи символов с клавиатуры терминала на удаленный компьютер по установленному TCP-соединению. Очевидно, что пользователь хотел бы увидеть “мгновенную” реакцию удаленного компьютера на нажатие любой клавиши терминала. Поэтому, если модуль протокола TCP перед отправкой символов помещает их в буфер, удаленный компьютер будет реагировать на каждое нажатие клавиши с большой задержкой (вполне возможно, что удаленный компьютер отреагирует только после нажатия нескольких сотен клавиш, причем на все сразу).

Точно так же модуль протокола TCP, запущенный на машине получателя, выполняет буферизацию полученных по сети данных, прежде чем доставить их прикладной программе. Очевидно, что для своевременной доставки данных прикладной программе получателя, недостаточно заставить отправителя отослать данные по сети.

Чтобы обеспечить работу интерактивных программ, в протоколе TCP предусмотрена специальная команда *принудительной* отправки данных (push). Ею могут воспользоваться прикладные программы для немедленной (без задержки, вызванной заполнением выходного буфера) передачи октетов данных, помещенных в выходной поток.

Однако команда принудительной отправки вызывает не только немедленную передачу сегмента данных модулем протокола TCP. Она предписывает модулю протокола TCP также установить в поле кода сегмента бит PSH. Эта операция гарантирует, что данные будут доставлены без задержек прикладной программе, запущенной на машине получателя. Таким образом, работая с интерактивным терминалом, прикладная программа должна при получении символа нажатой клавиши вызывать команду принудительной отправки данных.

По аналогии, чтобы пользователь без задержки смог увидеть на экране терминала результат нажатия на клавишу, прикладная программа, запущенная на машине получателя, также должна вызвать команду принудительной отправки данных после помещения в выходной поток отдельного символа или целой строки текста.

**Зарезервированные номера портов протокола TCP**

Как и в протоколе UDP, в TCP используется статическое и динамическое назначение номеров портов. При этом для часто используемых программ, таких как, например, программы работы с электронной почтой, назначаются фиксированные номера портов, называемые *широко известными (well-known)*, или *стандартными*. Оставшаяся большая часть номеров портов поступает в распоряжение операционной системы, которая распределяет их по мере надобности приложениям. Хотя в первоначальном варианте стандарта TCP для стандартных портов резервировались номера меньше 256, в настоящее время можно встретить номера широко известных портов, большие чем 1024. Список наиболее распространенных номеров стандартных портов протокола TCP приведен в табл. 14.2.

Таблица 14.2. Часто используемые стандартные номера портов протокола TCP

| **Номер порта** | **`Идентификатор в Internet** | **`Идентификатор в UNIX** | **`Описание** |
| --- | --- | --- | --- |
| 0 |  |  | Зарезервирован |
| 1 | ТСРМОХ | — | Мультиплексор протокола TCP |
| 7 | ECHO | echo | Эхо |
| 9 | DISCARD | discard | Аннулирование |
| 11 | USERS | systat | Список активных пользователей |
| 13 | DAYTIME | daytime | Текущая дата |
| 15 | — | netstat | Программа выдачи состояния сети |
| 17 | QUOTE | qotd | Цитата дня |
| 19 | CHARGEN | chargen | Генератор символов |
| 20 | FTP-DATA | ftp-data | Данные протокола передачи файлов (FTP) |
| 21 | FTP | ftp | Протокол передачи файлов (FTP) |
| 22 | SSH | ssh | Защищенная оболочка |
| 23 | TELNET | telnet | Подключение к терминалу |
| 25 | SMTP | smtp | Простой протокол передачи электронной почты (SMTP) |
| 37 | TIME | time | Текущее время |
| 43 | NICNAME | whois | Информация о пользователе |
| 53 | DOMAIN | nameserver | Сервер доменных имен (DNS) |
| 67 | BOOTPS | bootps | Сервер ВООТР или DHCP |
| 77 | — | rje | Любая приватная служба RJE |
| 79 | FINGER | finger | Программа finger |
| 80 | WWW | www | Сервер World Wide Web |
| 88 | KERBEROS | kerberos | Служба аутентификации Kerberos |
| 95 | SOPDOP | supdup | Протокол SUPDUP |
| 101 | HOSTNAME | hostnames | Сервер NIC имен хостов |
| 102 | ISO-TSAP | iso-tsap | Служба ISO-TSAP |
| 103 | X400 | x400 | Почтовая служба Х.400 |
| 104 | X400-SND | x400-snd | Отправитель почты Х.400 |
| 110 | POP3 | рор3 | Почтовый протокол версии 3 (РОРЗ) |
| 111 | SONRPC | sunrpc | Дистанционный вызов процедур в системе Sun |
| 113 | AOTH | auth | Служба аутентификации |
| 117 | OOCP-PATH | uucp-path | Служба путей протокола UUCP |
| 119 | NNTP | nntp | Протокол передачи сетевых новостей Usenet (NNTP) |
| 123 | NTP | ntp | Протокол синхронизации сетевого времени (NTP) |
| 139 | NETBIOS-SSN | — | Сеансовая служба протокола NETBIOS |
| 161 | SNMP | snmp | Простой протокол сетевого управления (SNMP) |

*Примечание*. В других протоколах, например UDP, для доступа к одноименным службам по возможности назначаются такие же номера портов.

Следует отметить, что хотя порты протоколов UDP и TCP являются независимыми, разработчики назначили одинаковые номера портов для тех служб, к которым можно обратиться как через протокол UDP, так и TCP. Например, к серверу доменных имен можно обратиться с помощью любого из этих двух протоколов. Поэтому в протоколах UDP и TCP для него назначен один и тот же порт с номером *53*.

**Производительность протокола TCP**

Как вы уже, наверное, заметили, TCP является довольно сложным протоколом. Независимость от сетевого оборудования позволяет использовать его для обмена данными в сетях практически любого типа. Поэтому многие считают, что, поскольку TCP выполняет куда более сложные задачи, чем любой другой транспортный протокол, его программный код должен быть весьма запутан и потому неэффективен.

Однако универсализм протокола TCP, о котором мы говорили, никак не сказался на его производительности. Эксперименты, проведенные в Беркли, показали, что протокол TCP может эффективно работать не только в глобальной сети Internet. При связи двух компьютеров по локальной сети Ethernet с пропускной способностью 10 Мбит/с средняя скорость передачи непрерывного потока данных за достаточно большой отрезок времени достигала 8 Мбит/с1. Инженеры компании Cray Research, Inc. показали, что при наличии соответствующего оборудования, пропускная способность протокола TCP может достигать 1 Гбит/с.

1. Оставшиеся 2 Мбит/с пропускной способности локальной сети использовались для передачи заголовков фрейма Ethernet, IP- и TCP-пакетов. Кроме того, при передаче последовательности пакетов часть полосы попросту теряется на межпакетные промежутки.

**Синдром полного окна и короткие пакеты**

При разработке протокола TCP было замечено значительное снижение производительности соединения в том случае, если отправляющая и принимающая прикладные программы работают с разными скоростями.

Чтобы понять суть проблемы, давайте вспомним, что полученные модулем протокола TCP данные помещаются в буфер. А теперь представим, что произойдет, если прикладная программа будет считывать данные из этого буфера по одному октету. Сразу после установки соединения модуль протокола TCP, запущенный на машине получателя, выделяет буфер размером *К* байт. При этом отправителю посылается анонс, в котором указывается исходный размер приемного буфера. (Напомним, что анонс размера приемного буфера посылается отправителю в поле размера окна заголовка сегмента, содержащего сигнал подтверждения приема данных.) Если приложение-отправитель генерирует поток данных с очень высокой скоростью, то модуль протокола TCP начнет передачу сегментов с учетом полного размера окна приемного буфера. В конечном итоге отправитель получит анонс, сообщающий, что приемное окно полностью заполнено, означающий, что во входном буфере не осталось свободного места.

Как только приложение-получатель считает из заполненного буфера октет данных, освободится место для получения еще одного октета данных. Выше уже было сказано, что модуль протокола TCP машины получателя должен проинформировать об этом отправителя. Он посылает отправителю сигнал подтверждения приема, в заголовке которого в поле размера окна указано количество свободного места в приемном буфере. В нашем примере получатель пришлет отправителю анонс на один октет. В ответ на это модуль протокола TCP машины отправителя отошлет получателю сегмент, содержащий один октет данных.

И хотя анонсы окна размером в один октет будут отрабатываться вполне корректно, для полного заполнения приемного буфера отправителю понадобится переслать несколько коротких сегментов данных. При этом модуль протокола TCP машины отправителя должен скомпоновать сегмент, содержащий один октет данных, поместить его в IP-дейтаграмму и передать ее по сети. Когда приложение-получатель считает из входного потока еще один октет данных, будет сгенерирован соответствующий сигнал подтверждения приема, который вызовет передачу отправителем следующего сегмента с одним октетом данных. В результате такого взаимодействия отправителя с получателем система может перейти в стабильное состояние, т.е. модуль протокола TCP будет посылать каждый октет данных в отдельном сегменте.

Передача коротких сегментов по сети резко снижает ее пропускную способность и приводит к излишней нагрузке на центральные процессоры машин отправителя и получателя. Снижение пропускной способности сети объясняется тем, что при передаче коротких сегментов увеличивается соотношение (длина заголовка)/(длина данных). Т.е., по сути, по сети вместо данных передаются одни заголовки. Нагрузка на процессоры повышается из-за того, что как приемной, так и передающей стороне приходится обрабатывать большое количество сегментов.

Модуль протокола TCP машины отправителя должен выделить в памяти место под буфер, сформировать заголовок и вычислить для сегмента контрольную сумму. Программы протокола IP машины отправителя должны инкапсулировать сегмент в дейтаграмму, вычислить контрольную сумму ее заголовка, определить маршрут следования дейтаграммы и передать ее соответствующему модулю сетевого интерфейса. Модуль протокола IP, запущенный на машине получателя, должен проверить контрольную сумму IP-заголовка и передать сегмент на обработку модулю протокола TCP. В свою очередь, модуль протокола TCP должен проверить контрольную сумму сегмента и его порядковый номер, извлечь из сегмента данные и поместить их во входной буфер.

Мы рассмотрели процесс образования коротких сегментов в сети в результате получения анонса на окно небольшого размера. Однако отправитель также может генерировать сегменты, содержащие небольшое количество данных. Например, представьте, что в некоторой реализации протокола TCP данные, как только попадают в выходной буфер, сразу же отправляются получателю.

Давайте рассмотрим, что произойдет, если прикладная программа при обращении к протоколу TCP будет передавать за раз один октет данных. После того как приложение сгенерирует октет данных, модуль протокола TCP поместит его в сегмент и передаст по сети получателю. Модуль протокола TCP может также посылать в сеть короткие сегменты, если прикладная программа генерирует данные блоками фиксированного размера, равными *В* октетов. Если предположить, что размер выходного буфера модуля протокола TCP равен *М* октетов, причем М ≠ В, то размер последнего передаваемого сегмента вероятнее всего будет небольшим.

Описанная в этом разделе проблема называется *синдромом полного окна (silly window syndrome*, или *SWS*). Она проявлялась в ранних реализациях протокола TCP. Подводя итоги, можно сказать следующее.

Ранним реализациям протокола TCP была присуща проблема, которая называлась синдромом полного окна. Суть ее состоит в том, что при практически полностью заполненном приемном буфере отправителю посылаются анонсы, на окна небольшого размера. В результате отправитель посылает в сеть сегменты небольшого размера.

**Как избежать синдрома полного окна**

В современные реализации протокола TCP включен специальный эвристический алгоритм, позволяющий избежать синдрома полного окна. Он не позволяет отправителю посылать в сеть несколько сегментов подряд, содержащих небольшое количество данных.

Еще один эвристический алгоритм запрещает получателю анонсировать окна небольшого размера в случае, если происходит небольшой прирост их размера. Напомним, что именно анонсирование окон небольшого размера является причиной появления коротких сегментов в сети.

Следует отметить, что два упомянутых выше эвристических алгоритма очень хорошо работают не только вместе, но и по отдельности. Это позволяет добиться приемлемой производительности сети в случае, если по какой-либо причине одна из сторон не сможет самостоятельно подавить синдром полного окна.

При создании реально действующего модуля протокола TCP разработчики должны включить в него специальный код, подавляющий синдром полного окна как на стороне отправителя, так и на стороне получателя. Чтобы понять, зачем это нужно, достаточно вспомнить, что протокол TCP является дуплексным, т.е. данные в нем могут течь в обоих направлениях. Поэтому при любой реализации модуля протокола TCP в него должен быть включен как код для отправки данных, так и код для их получения.

**Борьба с синдромом полного окна на стороне получателя**

Эвристический алгоритм, используемый для подавления синдрома полного окна на стороне получателя, довольно простой, поэтому его легко понять. В двух словах, суть его заключается в том, что получатель отслеживает с помощью специальной переменной текущий размер свободного буфера. При его увеличении анонс окна не посылается отправителю сразу, а задерживается до того момента, когда освободиться значительная часть приемного буфера. Осталось только выяснить, что вкладывается в понятие “значительная часть”. Все зависит от текущего размера приемного буфера и максимального размера сегмента. В стандарте протокола TCP сказано, что анонс окна посылается в случае освобождения как минимум половины приемного буфера либо пространства в буфере, достаточного для приема сегмента максимальной длины.

Таким образом, эвристический алгоритм, используемый для подавления синдрома полного окна на стороне получателя, препятствует анонсированию окон небольшого размера в том случае, если прикладная программа считывает медленно принятые октеты данных. Например, при заполнении приемного буфера отправителю будет послан сигнал подтверждения, содержащий анонс окна нулевого размера. По мере того как приложение-получатель считывает октеты из приемного буфера, модуль протокола TCP машины-получателя пересчитывает количество свободного места в этом буфере. Однако при этом анонс окна не посылается отправителю сразу, а задерживается до момента, пока не освободится половина приемного буфера или его часть, равная максимальной длине сегмента. Следовательно, отправитель будет всегда получать анонсы на окна большого размера, что позволит ему передавать только длинные сегменты. Из всего сказанного выше можно сделать такой вывод.

Суть эвристического алгоритма, используемого для подавления синдрома полного окна на стороне получателя, состоит в следующем. После отправки анонса на окно нулевого размера, следующий анонс посылается отправителю только после освобождения не менее половины, приемного буфера либо при освобождении в буфере места, достаточного для приема сегмента максимального размера.

**Задержка сигналов подтверждения приема**

Для подавления синдрома полного окна на стороне получателя используются два подхода.

Первый подход был рассмотрен в предыдущем разделе. Суть его заключается в том, что после получения каждого сегмента, модуль протокола TCP сразу же посылает отправителю сигнал подтверждения его приема. Однако при этом отправка анонсов на увеличение приемного окна задерживается до момента, пока в приемном буфере не освободится достаточно места, как было сказано выше при описании эвристического алгоритма.

При использовании второго подхода модуль протокола TCP не посылает отправителю сигнал подтверждения приема сегмента, если в приемном буфере недостаточно свободного места.

В стандарте протокола TCP рекомендуется использовать второй подход.

Метод задержки сигналов подтверждения приема имеет как преимущества, так и недостатки.

Основное преимущество заключается в том, что при задержке подтверждения приема уменьшается трафик в сети, т.е. увеличивается ее общая пропускная способность. Например, если за время задержки будут получены дополнительные сегменты данных, то подтвердить их успешное получение можно с помощью всего одного сегмента. Если же прикладная программа, получив данные, сразу должна сгенерировать на них ответ (как, например, при выполнении сеанса регистрации на удаленном компьютере, когда на терминал должны посылаться символы, соответствующие нажатым клавишам), введение небольшой задержки позволит поместить эти данные в тот же сегмент, в котором отправителю будет передан сигнал подтверждения приема. Более того, модуль протокола TCP не сможет переместить свое окно в потоке данных до тех пор, пока приложение-получатель не считает эти данные из буфера. Если приложение-получатель считывает данные из буфера сразу после их получения, введение небольшой задержки позволяет модулю протокола TCP в одном сегменте подтвердить прием данных и анонсировать изменение размеров приемного окна. Без введения задержки модуль протокола TCP пошлет отправителю подтверждение приема сразу после получения сегмента данных. По прошествии некоторого времени отправителю посылается второй сигнал подтверждения приема, в котором указывается анонс на изменение размеров приемного окна.

Недостатки метода задержки сигналов подтверждения приема, должно быть, уже очевидны. Самый существенный из них состоит в том, что если получатель вовремя не отошлет сигнал подтверждения приема, отправитель пришлет сегмент повторно. Очевидно, что подобные ненужные повторы существенно снижают пропускную способность сети. Кроме того, повторная отправка сегментов требует дополнительных затрат процессорного времени на машине как отправителя, так и получателя. Следует также напомнить, что в протоколе TCP по времени прибытия сигналов подтверждения приема оценивается полное время доставки сегмента. Поэтому любые задержки этих сигналов приводят к неточности результатов измерений, что влечет за собой увеличение времени тайм-аута повторной передачи.

Чтобы избежать потенциальных проблем, в стандарте протокола TCP установлены предельные значения задержек отправки сигналов подтверждения приема. При реализации модуля протокола TCP максимальная величина задержки не должна превышать 500 миллисекунд. Кроме того, для гарантии достоверности измерений полного времени доставки пакета в стандарте TCP рекомендуется, чтобы получатель сразу же посылал сигналы подтверждения приема как минимум на каждый второй сегмент данных.

**Борьба с синдромом полного окна на стороне отправителя**

Для подавления синдрома полного окна на стороне отправителя в протоколе TCP используется довольно элегантный эвристический алгоритм. Напомним, что конечная цель — избежать отправки в сеть коротких сегментов. Кроме того, следует иметь в виду, что приложение-отправитель может генерировать данные блоками сколь угодно малого размера (вплоть до одного октета). Поэтому, чтобы достичь конечной цели, модуль протокола TCP машины отправителя должен предоставить прикладной программе возможность многократного вызова функции *записи* в выходной буфер. При этом все переданные программой данные должны собираться в один достаточно большой сегмент и только затем отправляться получателю. Таким образом, модуль протокола TCP должен задерживать отправку сегмента до тех пор, пока его размер не достигнет приемлемых значений. Эта методика называется *накоплением (clumping)*.

Возникает вопрос: на сколько модуль протокола TCP должен задержать отправку пакета?

С одной стороны, при слишком больших задержках возрастает время доставки сегментов прикладной программе-получателю. Хуже всего то, что модуль протокола TCP не может точно определить время задержки, поскольку заранее никогда не известно, будут ли в ближайшем будущем получены какие-либо данные от прикладной программы.

С другой стороны, при слишком короткой задержке в сеть посылаются сегменты небольшого размера, в результате чего падает общая производительность сети.

С подобной проблемой разработчики протоколов столкнулись уже давно (еще до создания протокола TCP). Решалась она путем объединения данных в большие пакеты. Например, для повышения эффективности передачи данных по сети в ранних протоколах работы с терминалом передача каждого введенного с клавиатуры символа задерживалась на несколько сотен миллисекунд. За это время программа могла определить, будет ли пользователь продолжать ввод данных с клавиатуры. Поскольку протокол TCP создавался для универсального применения, его можно приспособить для решения специфических задач. Данные, передаваемые по TCP-соединению, могут быть введены с клавиатуры, прочитаны из файла либо получены в результате работы программы. Очевидно, что фиксированное значение задержки не может быть оптимальным для программ всех типов. Поэтому в протоколе TCP для ее определения используется адаптивный алгоритм, наподобие того, что использовался для определения времени тайм-аута повторной передачи и алгоритма медленного запуска, применяющегося для избегания кратковременных перегрузок сети. Величина задержки зависит от текущей скорости работы объединенной сети.

Как и алгоритм медленного запуска, эвристический алгоритм подавления синдрома полного окна на стороне отправителя называется *самосинхронизирующимся (self clocking)*, поскольку в нем не вычисляются величины задержек. Сигналом к началу передачи дополнительных пакетов служит момент получения сигнала подтверждения приема. Работу эвристического алгоритма можно вкратце описать следующим образом.

Суть эвристического алгоритма подавления синдрома полного окна на стороне отправителя состоит в следующем.

Если по установленному ТСР-соединению были посланы данные, но прием их не подтвержден, то новые данные, сгенерированные прикладной программой, помещаются в выходной буфер, но не отправляются получателю. Эти данные будут отправлены только после того, как от получателя поступит анонс на окно, размер которого позволяет принять сегмент максимальной длины.

Если после получения сигнала подтверждения приема в выходном буфере все еще находятся данные, ожидающие отправки, они немедленно посылаются получателю.

Это правило применяется даже в том случае, если от прикладной программы получен запрос на принудительную отправку данных.

Если при каждом обращении к модулю протокола TCP прикладная программа генерирует по одному октету данных, то первый октет посылается получателю немедленно. Остальные же октеты накапливаются в выходном буфере до тех пор, пока отправитель не получит сигнал подтверждения приема. Поэтому, если скорость потока данных, генерируемых прикладной программой, сравнима с пропускной способностью сети (как, например, при передаче файлов), во всех последующих сегментах будет передаваться много октетов данных.

Если же скорость потока данных, генерируемых прикладной программой, крайне мала по сравнению с пропускной способностью сети (как в случае программы ввода данных с клавиатуры пользователя), короткие сегменты будут посылаться без существенной задержки.

Описанная в этом разделе методика борьбы с синдромом полного окна на стороне отправителя называется *алгоритмом Нейгла (Nagle algorithm)* по имени его создателя.

Ценность этой методики заключается в том, что для выполнения алгоритма не требуются большие вычисления. Узлу сети не нужно поддерживать работу отдельных таймеров для каждого соединения. При получении данных от прикладной программы модуль протокола не должен опрашивать значение таймера. Однако важнее всего то, что эту методику можно использовать при любых значениях задержек в сети, максимальных размерах сегмента и скорости работы приложения. При нормальных условиях она не снижает пропускную способность сети.

Следует объяснить, почему при нормальных условиях алгоритм Нейгла не снижает пропускную способность сети. Если прикладная программа рассчитана на обработку большого потока данных, она не будет генерировать по одному октету данных за раз, поскольку это приведет к излишней нагрузке на операционную систему. При каждом обращении к модулю протокола подобные программы, передают большой блок данных. Поэтому в выходном буфере всегда будет содержаться достаточное количество данных (по крайней мере не меньше размера одного максимального сегмента). Более того, поскольку прикладные программы обычно генерируют данные гораздо быстрее, чем их может передать модуль протокола TCP, выходной буфер практически всегда полон, и модулю протокола TCP незачем откладывать передачу данных.

Если прикладные программы постоянно заполняют выходной буфер данными, модуль протокола TCP отправляет сегменты с максимально возможной для объединенной сети скоростью. Подведем итоги.

В современные реализации протокола TCP включен эвристический алгоритм подавления синдрома полного окна как на стороне отправителя, так и на стороне получателя. Суть его состоит в том, что получатель старается избежать анонсирования окон небольшого размера, а отправитель использует адаптивный алгоритм для задержки передачи сегментов. Это дает возможность отправителю собрать данные в сегмент большого размера.

**Проверка знаний: Надежная потоковая транспортная служба (TCP)**

**Задание 1**

Как выглядит бит срочных данных?

SYN

FIN

URG

АСК

**Задание 2**

В подтверждениях приема протокола TCP указывается порядковый номер в потоке определенного октета, который должен быть принят получателем. Какой это октет?

предыдущий

следующий

может быть как предыдущий, так и следующий

**Задание 3**

Что требует протокол TCP в отличие от протокола UDP?

установки соединения с получателем и отправителем

установки соединения с получателем

установки соединения с отправителем

**Задание 4**

Чем оперирует механизм движущихся окон протокола TCP?

октетами

сегментами

пакетами

**Задание 5**

Что дополнительно сможет принять получатель, указанный в анонсе окна?

октеты

байты

IP-адреса

**Задание 6**

Какой выбирается первоначальный размер окна перегрузки при *методике медленного запуска*?

равному четырём сегментам

равному одному сегменту

равному двум сегментам

**Задание 7**

Какой алгоритм используется в протоколе TCP, чтобы учесть большой разброс задержек во времени доставки пакетов в реально действующей объединенной сети?

ускоренный алгоритм повторной передачи

адаптивный алгоритм повторной передачи

дублированный алгоритм повторной передачи

**Задание 8**

Какой бит в заголовке поля кода должен быть отправлен для начала процесса досрочного прекращения передачи данных?

URG

RST

SYN

**Задание 9**

Если отправитель и получатель находятся в разных физических сетях и требуется установить максимальный размер сегмента, то каков будет его размер?

576 октетов

536 октетов

616 октетов

**Задание 10**

В чем недостаток механизма выталкивания?

наличие большого количества потенциальных ошибок

первоначальная структура пакетов не сохраняется

медленная отправка накопленных данных

**Задание 11**

Как называется ситуация, когда увеличение трафика вызывает увеличение задержки, а увеличение задержки снова приводит к увеличению трафика в сети?

медленный запуск

сетевая задержка

полный затор в сети

**Задание 12**

В современные реализации протокола TCP включен эвристический алгоритм подавления синдрома полного окна как на стороне отправителя, так и на стороне получателя. Каковы действия отправителя и получателя?

получатель старается избежать анонсирования окон небольшого размера, а отправитель использует адаптивный алгоритм для задержки передачи данных

получатель использует адаптивный алгоритм для задержки передачи данных, а отправитель старается избежать анонсирования окон небольшого размера

оба варианта не верны

**Задание 13**

О чем можно судить по значению контрольной суммы?

доставлен ли сегмент указанному получателю без ошибок

по значению контрольной суммы невозможно судить о доставке

доставлен ли пакет указанному получателю без ошибок

**Задание 14**

Одна из проблем обеспечения надежности возникает при получении дублей пакетов системой доставки нижнего уровня. Назовите одну из причин появления дублей?

перегрузка сети

сбой в работе микропрограммы маршрутизатора

малая пропускная способность сети

**Задание 15**

Что позволяет полностью исключить простои в сети при использовании метода движущихся окон?

повторная передача

правильный выбор размера окна

правильная длина заголовка дейтаграммы

**Задание 16**

Как в системе иерархии располагается протокол TCP относительно протокола IP?

выше

на одном уровне

ниже

**Задание 17**

В качестве основной абстракции в протоколе TCP используется понятие

номер порта

соединение

конечные точки

**Задание 18**

Как называют принятую в протоколе TCP систему подтверждения приема?

избирательной

сплит

кумулятивной

**Задание 19**

Что является единицей передачи данных между двумя машинами в протоколе TCP?

сегмент

пакет

октет

**Задание 20**

Из чего состоит каждый сегмент?

из заголовка

из блока данных

из заголовка и блока данных

**Основы маршрутизации**

В предыдущих разделах внимание уделялось службам сетевого уровня семейства протоколов TCP/IP, а также особенностям протоколов, которые используются в узлах сети и маршрутизаторах, предоставляющих такие услуги. При обсуждении этих вопросов мы предположили, что маршрутизаторы всегда содержат только правильную информацию о маршрутах, а также отметили, что маршрутизаторы могут заставить узлы, непосредственно подключенные к одной с ними сети, изменить маршруты посредством механизма перенаправления протокола ICMP.

В этом разделе рассматриваются два обширных вопроса: какие значения должна содержать таблица маршрутизации и каким образом можно получить эти значения?

Чтобы ответить на первый вопрос, рассмотрим взаимосвязь между структурой объединенной сети и механизмом маршрутизации. В частности, мы рассмотрим принципы объединения сети на основе одной широкополосной магистрали и нескольких равноправных сетевых магистралей. Затем мы обсудим необходимые условия для осуществления маршрутизации. Поскольку большинство примеров из этого раздела взяты из глобальной сети Internet, рассматриваемые здесь вопросы касаются также небольших корпоративных объединенных сетей.

Чтобы получить ответ на второй вопрос, мы рассмотрим два основных вида алгоритмов распространения маршрутной информации, а также увидим, каким образом каждый из них автоматически передает маршрутную информацию.

В начале мы рассмотрим, что представляет собой процесс маршрутизации в общем смысле этого слова. В последующих разделах мы уделим внимание структуре объединенной сети и алгоритмам, которые используются маршрутизаторами для обмена маршрутной информацией.

* [Понятие о таблицах маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d0)
* [Маршрутизация при неполной информации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d1)
* [Первоначальная структура сети Internet и ее ядро](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d2)
* [Базовая система маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d3)
* [От сетевого ядра до системы равноправных магистралей](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d4)
* [Автоматическое распространение маршрутной информации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d5)
* [Дистанционно-векторная маршрутизация (метод Беллмана-Форда)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d6)
* [Межшлюзовый протокол (GGP)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d7)
* [Упорядочение расстояния](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d8)
* [Надежность и протоколы маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905d9)
* [Маршрутизация, отслеживающая состояние соединения (SPF)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905da)
* [Проверка знаний: Основы маршрутизации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905db)

**Понятие о таблицах маршрутизации**

Ранее мы уже говорили, что IP-маршрутизаторы обеспечивают активные взаимосвязи между сетями. Каждый маршрутизатор подключается к двум или более физическим сетям и выполняет пересылку IP-дейтаграмм между ними. Он получает дейтаграммы, поступающие через один сетевой интерфейс, и переправляет их через другой. За исключением получателей, которые непосредственно подключены к тем же сетям, что и маршрутизатор, узлы передают весь свой информационный поток маршрутизаторам, используя для этого протокол IP. Последние, в свою очередь, переправляют дейтаграммы далее по маршруту к конечным пунктам назначения. Дейтаграмма переходит от одного маршрутизатора к другому до тех пор, пока не достигнет маршрутизатора, непосредственно подключенного к той же сети, что и ее конечный получатель. Таким образом, система маршрутизаторов создает структурную основу объединенной сети и управляет всем трафиком, за исключением непосредственной доставки пакетов от одного узла до другого.

В разделе [Протокол IP: маршрутизация дейтаграмм](https://it.rfei.ru/course/%7EGZqN/-/%7EXVZ0) описан алгоритм маршрутизации протокола IP, которым руководствуются узлы сети и маршрутизаторы, перенаправляющие дейтаграммы. В ней также демонстрируется, как в этом алгоритме используются таблицы маршрутизации для выбора маршрута следования дейтаграмм. В каждом элементе таблицы маршрутизации указывается сетевая часть IP-адреса получателя и адрес следующей машины, расположенной по маршруту следования пакетов до сети получателя. Подобно узлам сети, маршрутизаторы доставляют дейтаграммы получателям, которые непосредственно подключены к тем же сетям, что и маршрутизатор.

Рассматривая основы передачи дейтаграмм, мы не выяснили, каким образом узлы сети и маршрутизаторы получают информацию для своих таблиц маршрутизации. Этот вопрос имеет два аспекта: *какие именно* значения должны быть занесены в таблицу и *как* маршрутизаторы получают эти значения? Оба аспекта зависят от структуры и размера объединенной сети, а также принципов управления ею.

Создание системы маршрутизации происходит в два этапа:

1. инициализация
2. обновление

При включении питания каждый маршрутизатор должен установить исходный набор маршрутов. По мере изменения маршрутов маршрутизатор должен обновлять свою таблицу (например, если происходит отказ одного из сетевых интерфейсов).

Этап инициализации зависит от типа используемой операционной системы. В некоторых системах маршрутизатор считывает при запуске исходную таблицу маршрутизации из внешнего запоминающего устройства и сохраняет ее в своей оперативной памяти. В других системах работа маршрутизатора начинается с пустой таблицы, которую необходимо заполнить, выполняя определенные команды (например, команды, находящиеся в сценарии начальной загрузки). И, наконец, некоторые операционные системы начинают свою работу с того, что логически определяют исходный набор маршрутов из набора адресов локальных сетей, к которым подключена машина, и затем связываются с соседней машиной с целью запроса дополнительных маршрутов.

Когда исходная таблица маршрутизации создана, маршрутизатор должен внести изменения в маршруты. В небольших, редко изменяющихся сетях администраторы могут устанавливать и модифицировать маршруты вручную. В крупных, быстро изменяющихся средах ручное обновление происходит слишком медленно и чревато ошибками. В этом случае необходимо применять автоматические методы обновления.

Для того чтобы понять сущность протоколов для автоматического обновления таблиц маршрутизации в IP-маршрутизаторах, необходимо рассмотреть несколько основополагающих понятий. В последующих разделах изложены концептуальные основы процесса маршрутизации. В конце раздела обсуждается структура объединенной сети и протоколы, которые используются маршрутизаторами для обмена маршрутной информацией.

**Маршрутизация при неполной информации**

Существенное отличие между маршрутизатором и типичным узлом сети заключается в том, что узлы, как правило, владеют недостаточной информацией о структуре объединенной сети, к которой они подключены. Узлы не располагают полной информацией обо *всех возможных* адресах получателей или *всех возможных* сетях-адресатах. На самом деле в таблицах маршрутизации многих узлов есть информация только для двух маршрутов: один из них — для локальной сети, а другой — маршрут по умолчанию, или стандартный маршрут, определяющий путь прохождения пакетов к соседнему маршрутизатору.

Все дейтаграммы, адресованные во внешний мир, узел сети отсылает локальному маршрутизатору для дальнейшей доставки.

Обратите внимание, что узел сети может успешно выполнять маршрутизацию дейтаграмм, даже если он располагает частичной информацией о маршрутизации. Причина в том, что узел сети может свободно положиться на маршрутизатор.

Могут ли маршрутизаторы также успешно выполнять маршрутизацию дейтаграмм, располагая лишь частичной информацией? Да, могут, но только при определенных условиях. Чтобы понять необходимые для этого условия, представим, что объединенная сеть — это чужая страна, которую вдоль и поперек пересекают грунтовые дороги, а на их перекрестках установлены указатели. Представьте, что у вас нет карты, вы не можете спросить, в каком направлении вам необходимо передвигаться, так как не говорите на языке местных жителей, не имеете представления о местности, в которой находитесь. Но вам просто необходимо добраться до селения под названием *Суссекс*. Вы отправляетесь в путешествие, передвигаясь по единственной дороге, ведущей из вашего города, и начинаете искать глазами указатели. На первом указателе написано1:

**Нортфок — налево; Хаммонд — направо; другие города — прямо**.

Поскольку необходимый вам пункт назначения явно не указан, вы продолжаете двигаться прямо. На языке маршрутизации мы назовем маршрут, по которому вы следуете, *стандартным маршрутом*, или *маршрутом, принятым по умолчанию*. Через несколько знаков вы, наконец, встречаете указатель, на котором написано:

**Эссекс — налево; Суссекс — направо; другие города — прямо**.

Вы поворачиваете направо, передвигаетесь по нескольким последующим указателям и, наконец, попадаете на дорогу, ведущую в Суссекс.

Наше воображаемое путешествие является полным аналогом прохождения дейтаграммы по объединенной сети, а дорожные знаки — это таблицы маршрутизации, которые находятся в устройствах маршрутизации, расположенных по пути следования дейтаграммы. Без карты или других навигационных приборов путешествие полностью зависит от дорожных знаков, точно так же, как маршрутизация дейтаграммы в объединенной сети полностью зависит от содержимого таблиц маршрутизации. Понятно, что передвижение возможно несмотря на то, что каждый дорожный знак содержит только частичную информацию.

Главный вопрос заключается в достоверности информации. Как путешественник вы можете спросить: “Как я могу быть уверен что, следуя по знакам, достигну конечного пункта назначения?” или “Могу ли я быть уверен, что, следуя по знакам, достигну пункта назначения кратчайшим путем?” Ответить на эти вопросы особенно трудно, если на пути вам встречается много знаков, в которых явно не указан ваш пункт назначения.

Конечно, ответы на поставленные вопросы зависят от топологической схемы дорог, а также содержащейся на знаках информации. Но основной смысл заключается в том, что в целом информация, содержащаяся на знаках, должна быть полной и непротиворечивой.

С другой стороны, очевидно, что на каждом перекрестке необязательно должен присутствовать знак, указывающий на любой пункт назначения. На указателях могут перечисляться стандартные маршруты при условии, что на других знаках будет явно указан кратчайший путь к нужному пункту назначения. Кроме того, должны быть обозначены повороты, ведущие ко всем пунктам назначения.

Ниже мы на нескольких примерах продемонстрируем, как достичь согласованности информации.

Как крайний случай, рассмотрим простую схему дорог, которая имеет форму звезды. В этой схеме к каждому селению ведет только одна дорога, а все пути к другим населенным пунктам сходятся в одной центральной точке. Чтобы обеспечить непротиворечивость информации, указатель на главном перекрестке должен содержать информацию обо всех возможных пунктах назначения.

Есть и другая крайность. Представьте себе произвольный набор дорог, на всех перекрестках которых размещены указатели, содержащие информацию о всех возможных пунктах назначения. Тогда, чтобы обеспечить непротиворечивость информации, на любом перекрестке должны соблюдаться следующие правила: если для пункта назначения *Г* указатель показывает на дорогу *Д*, то только дорога *Д* ведет к пункту *Г* по кратчайшему пути.

Ни одна из описанных схем не работает хорошо в системе маршрутизации объединенной сети.

С одной стороны, первый, приведенный выше подход к решению проблемы, неэффективен, потому что ни одна машина не может выполнять роль центрального коммутатора, через который проходит весь сетевой трафик.

С другой стороны, держать на всех маршрутизаторах информацию о маршрутах ко всем возможным пунктам назначения непрактично, поскольку это требует распространения большого объема информации в случае каких-либо изменений в сети, или каждый раз, когда администраторам необходимо проверить непротиворечивость информации.

Таким образом, необходимо найти решение, которое позволило бы группе лиц управлять локальными маршрутизаторами в автономном режиме так, чтобы добавление новых сетевых взаимосвязей и маршрутов не влияло на работу удаленных маршрутизаторов.

Для того чтобы в дальнейшем было легче объяснить некоторые из особенностей структуры сетевой маршрутизации, рассмотрим третью топологическую схему, в которой половина населенных пунктов расположена в восточной части страны, а другая половина — в западной. Предположим, что через реку, которая отделяет Восток от Запада, перекинут только один мост. Допустим, что люди, живущие в восточной части, не любят уроженцев Запада. По этой причине они используют только те дорожные знаки, на которых указаны пункты восточного региона. Предположим также, что люди, проживающие на Западе, делают обратное. Маршрутизация будет непротиворечивой только в том случае, если на любом дорожном знаке в восточной части страны будут явно перечислены все пункты этого региона, а также указан стандартный маршрут, ведущий к мосту. Что касается западной части, то на каждом знаке, соответственно, должны быть явно перечислены все пункты западного региона, а также указан стандартный маршрут, ведущий к мосту.

1. Предполагается, что указатели написаны на понятном вам языке.

**Первоначальная структура сети Internet и ее ядро**

Большая часть наших знаний о маршрутизации и протоколах распространения маршрутов была почерпнута из опыта, приобретенного в результате работы в глобальной сети Internet. Когда впервые был создан протокол TCP/IP, сетевые центры, принимающие участие в исследованиях, были подключены к сети APRANET, которая выполняла функции магистрального канала Internet. Когда проводились первые эксперименты, каждый из сетевых центров сам управлял таблицами маршрутизации и вручную устанавливал маршруты к другим получателям. Когда сеть Internet начала расти и развиваться, стало ясно, что вручную устанавливать и контролировать маршруты непрактично. Возникла потребность в выработке автоматических механизмов.

Создатели сети Internet выбрали такую структуру системы маршрутизации, которая состояла из небольшого набора базовых маршрутизаторов, содержащих полную информацию обо всех возможных получателях, и более обширного набора внешних маршрутизаторов, содержащих неполную информацию о маршрутах. Если провести аналогию, этот подход напоминает создание небольшого количества перекрестков в центре города, на которых есть указатели. На них перечислены все возможные пункты назначения. Причем на внешних перекрестках перечислены только местные пункты. При условии, что стандартный маршрут на каждом внешнем перекрестке указывает на один из центральных перекрестков, путешественники в конечном счете достигнут нужных пунктов назначения.

Преимущество использования неполной информации во внешних маршрутизаторах заключается в том, что это позволяет местным администраторам управлять изменениями в структуре локальной сети, не затрагивая другие части объединенной сети.

Недостаток этого подхода заключается в том, что создается вероятность возникновения противоречий. В самом худшем случае, ошибка во внешнем маршрутизаторе может помешать доступу к удаленным маршрутизаторам. Подведем итог всему сказанному выше.

Таблица маршрутизации произвольного маршрутизатора содержит неполную информацию о возможных получателях. Маршрутизация, в которой используется неполная информация, позволяет сетевым центрам самостоятельно вносить изменения в локальные маршруты, но вместе с тем создает вероятность возникновения противоречий. Все это может помешать доступу к определенным получателям со стороны некоторых отправителей.

Несоответствие информации в таблицах маршрутизации, как правило, возникает в результате ошибок в алгоритмах, по которым вычисляются значения в таблице маршрутизации, а также из-за неправильных данных, предоставленных этим алгоритмам, или в результате ошибок, возникающих при передаче результатов другим маршрутизаторам.

Разработчики протоколов ищут способы, с помощью которых можно было бы ограничить последствия ошибок. Их главная задача заключается в том, чтобы обеспечить отсутствие противоречий во всех маршрутизаторах в любой момент их работы.

Если по какой-либо причине в маршрутизаторах возникают несоответствия, протоколы маршрутизации должны быстро обнаружить и исправить ошибки. И самое важное: протоколы должны быть разработаны таким образом, чтобы ограничить последствия этих ошибок.

**Базовая система маршрутизации**

Первые появившиеся в сети Internet маршрутизаторы можно условно разделить на две группы: небольшой набор *базовых маршрутизаторов (core routers)*, обслуживаемых Информационным центром сети Internet (INOC, или Internet Network Operations Center), и более обширный набор *второстепенных маршрутизаторов1* (*noncore routers*), обслуживаемых отдельными административными группами. Система базовых маршрутизаторов была разработана для того, чтобы обеспечить создание надежной и согласованной системы маршрутизации для всех возможных получателей. Она была связующим звеном в сети Internet и создавала возможность глобальной связи. Согласно нормативным документам, каждый сетевой центр должен был сообщить выделенный ему IP-адрес сети системе базовых маршрутизаторов. Поскольку базовые маршрутизаторы обменивались информацией, существовала гарантия, что их общая информация не содержит противоречий. Благодаря тому что работу базовых маршрутизаторов контролировал центральный административный орган Internet, надежность их работы не вызывала сомнений.

Для того чтобы понять особенность системы базовых маршрутизаторов, необходимо вспомнить, что Internet возникла на основе уже существующей глобальной сети APRANET. Когда начали проводить первые эксперименты с Internet, разработчики выбрали APRANET в качестве основной сетевой магистрали, на которой в будущем должна была строиться сеть Internet. Таким образом, толчком к созданию системы базовых маршрутизаторов стало прежде всего желание подключить локальные сети к магистрали APRANET. На рис. 15.1 показана структурная схема этой системы.

Рис. 15.1. Первая система базовых маршрутизаторов в Internet состояла из набора маршрутизаторов, с помощью которых локальные сети подключались к магистрали APRANET. Узлы в локальных сетях направляли весь внешний трафик ближайшему из базовых маршрутизаторов

Чтобы понять, почему такую структуру нельзя применить для маршрутизации при неполной информации, предположим, что большая объединенная сеть полностью состоит из локальных сетей, которые подключены к магистральной сети через маршрутизаторы. Представим также, что некоторые из маршрутизаторов выполняют маршрутизацию на основе стандартных маршрутов. Теперь рассмотрим маршрут, по которому следует дейтаграмма. В сетевом центре отправителя локальный маршрутизатор проверяет, есть ли у него явно заданный маршрут к получателю. При отсутствии такого маршрута маршрутизатор отсылает дейтаграмму по стандартному маршруту. Все дейтаграммы, для которых у маршрутизатора отсутствует явно заданный маршрут, следуют по одному и тому же стандартному маршруту, независимо от получателя, которому они предназначены. Следующий маршрутизатор, расположенный по маршруту следования, отводит в сторону дейтаграммы, для которых он имеет явно заданный маршрут, а остальные отсылает по своему стандартному маршруту.

Чтобы обеспечить согласованную работу глобальной сети, цепочка стандартных маршрутов должна объединять каждый из маршрутизаторов в одно гигантское кольцо, как показано на рис. 15.2. Таким образом, структура сети требует, чтобы все локальные сетевые центы согласовывали свои стандартные маршруты. Кроме того, зависимость от маршрутов по умолчанию может быть неэффективной, даже если в самой системе нет противоречий.

Рис. 15.2. Набор маршрутизаторов, подключенных к магистрали. В данном случае процесс маршрутизации неэффективен, даже при условии, что в системе нет противоречий, поскольку каждое из устройств выполняет маршрутизацию на основе стандартного маршрута

Как показано на рис. 15.2, в самом худшем случае при передаче дейтаграммы от источника к получателю, она пройдет через все *п* маршрутизаторов, вместо того, чтобы прямо пойти через магистраль.

Чтобы избежать недоразумений при использовании стандартных маршрутов, создатели Internet организовали обмен информацией между всеми базовыми маршрутизаторами. Таким образом, каждый из них располагал полной информацией об оптимальных маршрутах ко всем возможным получателям. Поскольку любому из базовых маршрутизаторов были известны маршруты ко всем возможным получателям, им не нужно задавать стандартные маршруты. Если адрес конечного получателя, указанный в дейтаграмме, отсутствовал в таблице базового маршрутизатора, он посылал отправителю ICMP-сообщение о недоступности получателя и аннулировал дейтаграмму. По существу, введение системы базовых маршрутизаторов позволило решить проблему неэффективной маршрутизации и исключить применение стандартных маршрутов.

На рис. 15.3 схематично показана структура системы базовой маршрутизации. Она состоит из ядра, в которое входит один или несколько базовых маршрутизаторов, и набора внешних маршрутизаторов, расположенных в локальных сетевых центрах. Внешние маршрутизаторы содержат информацию о получателях, расположенных в их локальных сетях, и стандартный маршрут. Благодаря ему дейтаграммы, предназначенные для других сетевых центров, пересылаются в ядро системы.

Рис. 15.3. Упрощенная схема базовой системы маршрутизации, на которой показаны стандартные маршруты. В базовых маршрутизаторах стандартные маршруты не используются. Каждый из внешних маршрутизаторов, обозначенных *Li* имеет свой стандартный маршрут, ведущий в ядро системы

Хотя упрощенная структурна схема базовой системы маршрутизации (см. рис. 15.3) достаточно прозрачна, но в то же время непрактична по трем причинам.

Во-первых, Internet выросла из одной протяженной сетевой магистрали, управляемой из единого центра. Топологическая схема усложнилась, нетривиальными стали и протоколы, необходимые для сохранения непротиворечивости информации в базовых маршрутизаторах.

Во-вторых, не каждый сетевой центр мог иметь в своем составе базовый маршрутизатор, подключенный к магистрали. Таким образом, возникла необходимость создания дополнительной структуры маршрутизации и протоколов.

В-третьих, поскольку для поддержания непротиворечивости информации все базовые маршрутизаторы должны были взаимодействовать между собой, структуру сетевого ядра нельзя было увеличивать до произвольных размеров.

1. Кроме термина *второстепенный (noncore)*, для обозначения этих маршрутизаторов использовались также термины *тупиковый (stub)*, или *немаршрутизируемый (nonrouting)*.

**От сетевого ядра до системы равноправных магистралей**

Подключение магистрали NSFNET к сети Internet создало дополнительную сложность в структуре системы маршрутизации. С точки зрения базовых маршрутизаторов подключение сети NSFNET вначале ничем не отличалось от подключения любого другого сетевого центра к магистральной сети Internet. Магистраль NSFNET была подключена к магистрали APRANET через единственный маршрутизатор, расположенный в Питсбурге. В ядре системы явно указывались маршруты ко всем получателям, подключенным к магистрали NSFNET. Маршрутизаторы внутри магистрали NSFNET располагали информацией о локальных получателях и использовали стандартные маршруты для отправки сетевому ядру всего трафика, не принадлежащего магистрали, посредством маршрутизатора в Питсбурге.

Когда магистраль NSFNET разрослась и стала основной составляющей сети Internet, стало ясно, что структура базовой маршрутизации не сможет удовлетворить всем требованиям. Важное изменение в структуре системы маршрутизации произошло тогда, когда между магистралями ARPANET и NSFNET возникло несколько дополнительных соединений. Таким образом, ARPANET и NSFNET превратились в *равноправные сетевые магистрали (peer backbone networks)*. На рис. 15.4 показана получившаяся в результате топологическая схема равноправных сетевых магистралей.

Рис. 15.4. Система равноправных сетевых магистралей, соединенных между собой посредством нескольких маршрутизаторов. На схеме показана структура сети Internet по состоянию на 1989 год. При дальнейших расширениях Internet каждая из параллельных сетевых магистралей принадлежала одному из провайдеров

Чтобы понять трудности IP-маршрутизации между равноправными магистралями, рассмотрим маршруты, ведущие от узла 3 до узла 2, которые изображены на рис. 15.4. Представим себе, что на рисунке изображена географическая карта. Поэтому узел 3 находиться на Западном побережье, которое принадлежит магистрали NSFNET, в то время как узел 2 — на Восточном побережье, которое принадлежит магистрали ARPANET. При установке маршрутов между узлами 3 и 2 администраторам необходимо решить следующие вопросы:

а) стоит ли направлять трафик от узла 3 через маршрутизатор R1 расположенный на Западном побережье, а затем по магистрали ARPANET, или

б) направлять трафик от узла 3 по магистрали NSFNET, через центральный маршрутизатор R2, а затем по магистрали ARPANET — к узлу 2, или все-таки

в) направлять трафик по магистрали NSFNET, через маршрутизатор на Восточном побережье, R3, а затем по магистрали ARPANET — к узлу 2.

Можно также проложить маршрут и обходными путями. Например, трафик может проходить от узла 3 через маршрутизатор на Западном побережье, по магистрали ARPANET к центральному маршрутизатору, снова по магистрали NSFNET к маршрутизатору на Восточном побережье и, наконец, по магистрали ARPANET к узлу 2. Такой маршрут может быть целесообразным или — нет, в зависимости от целей использования сети и пропускной способности различных маршрутизаторов и магистралей.

Для большинства конфигураций равноправных сетевых магистралей должно соблюдаться следующее правило: трафик между двумя географически близко расположенными узлами сети должен проходить по кратчайшему пути, независимо от настроек маршрутизаторов, перенаправляющих межгосударственный трафик. Например, трафик от узла 3 до узла 1 должен проходить через маршрутизатор на Западном побережье, потому что он сокращает до минимума путь прохождения пакетов по обеим магистралям.

Все эти положения звучат довольно просто, но на самом деле их тяжело реализовать по двум причинам.

Во-первых, хотя стандартный алгоритм маршрутизации протокола IP использует сетевую часть IP-адреса для выбора маршрута, при реализации методов оптимальной маршрутизации в структуре равноправных сетевых магистралей для отдельных узлов требуется проложить отдельные маршруты. Что касается примера, приведенного выше, в таблице маршрутизации узла 3 должны присутствовать разные маршруты для достижения узлов 1 и 2, несмотря на то, что оба этих узла подключены к магистрали APRANET.

Во-вторых, администраторы обоих магистралей должны договориться о том, чтобы в таблицах маршрутизации всех устройств содержалась непротиворечивая информация. В противном случае могут возникать так называемые *маршрутные петли (routing loops)*. (Маршрутная петля возникает тогда, когда несколько маршрутизаторов “перебрасывают” пакеты по кругу друг другу.)

Важно уметь отличать топологическую схему сети от структуры системы маршрутизации. Например, может существовать единое ядро сети, состоящее из нескольких магистральных сетей. Машины сетевого ядра можно запрограммировать таким образом, чтобы они скрывали низкоуровневые особенности структуры системы и вычисляли кратчайшие маршруты друг к другу. Однако невозможно разбить систему сетевого ядра на подмножества, каждое из которых содержало бы частичную информацию, не теряя при этом своих функциональных возможностей. Эта проблема продемонстрирована на рис. 15.5.

Рис. 15.5. Попытка разбить систему базовой маршрутизации на два набора маршрутизаторов, содержащих неполную информацию и поэтому использующих стандартные маршруты. Такая структура приводит к образованию маршрутной петли для дейтаграмм, ошибочно посланных несуществующему получателю

Как показано на рис. 15.5, стандартные маршруты внешних маршрутизаторов направлены в одну из частей разделенного сетевого ядра. Маршрутизаторы каждой из частей ядра обладают информацией о получателях, находящихся в своем регионе, а также используют стандартный маршрут для передачи пакетов, предназначенных для другого региона. В такой структуре любая дейтаграмма, отосланная по недопустимому адресу, будет циклически передаваться по кругу между двумя частями ядра до тех пор, пока не истечет время ее жизни. Таким образом, можно сделать следующий вывод.

Структура базовой системы маршрутизации предполагает, что централизованный набор маршрутизаторов выполняет функции хранилища информации обо всех возможных получателях в объединенной сети. Базовые системы маршрутизации лучше всего функционируют в тех объединенных сетях, которые имеют одну, управляемую из центра магистраль. Расширение топологических схем и подключение нескольких равноправных магистралей усложняет систему маршрутизации, а попытки разделить ядро сети на несколько частей таким образом, чтобы все маршрутизаторы могли использовать стандартные маршруты, создает предпосылку для образования маршрутных петель.

**Автоматическое распространение маршрутной информации**

Выше уже отмечалось, что в первоначальной системе базовой маршрутизации сети Internet не использовались стандартные маршруты. Причина в том, что маршрутизаторы, находящиеся в ядре сети, располагали полной информацией обо всех возможных получателях и обменивались ею между собой. В настоящее время похожая система используется во многих корпоративных объединенных сетях. Корпоративные маршрутизаторы работают под управлением специальной программы, которая передает маршрутную информацию.

В последующих разделах обсуждаются два основных вида алгоритмов, которые используются для вычисления и распространения маршрутной информации. Для иллюстрации одного из алгоритмов в них используется оригинальный протокол базовой маршрутизации. Затем, в следующем разделе описывается протокол, в котором применяется другой тип алгоритма.

На первый взгляд может показаться, что в механизмах для автоматического распространения маршрутной информации нет необходимости, особенно в случае небольших объединенных сетей. Однако объединенные сети не являются статичными. Межсетевые соединения периодически выходят из строя и со временем заменяются на другие. В определенный момент в сети может возникнуть перегрузка либо сеть может использоваться не на полную мощность.

Целью механизма автоматического распространения маршрутов является не только поиск набора маршрутной информации, но и постоянное ее обновление. Человек просто не может достаточно быстро реагировать на изменения, происходящие в объединенной сети, поэтому возникает необходимость в использовании компьютерных программ.

Таким образом, говоря о распространении маршрутов, важно учитывать работу протоколов и алгоритмов в динамике.

**Дистанционно-векторная маршрутизация (метод Беллмана-Форда)**

Термин *дистанционный вектор1* (*distance-vector*) относится к классу алгоритмов, используемых маршрутизаторами для распространения маршрутной информации.

Идея, положенная в основу дистанционно-векторного алгоритма, довольно проста. Маршрутизатор хранит в таблице список всех известных маршрутов. При загрузке он инициализирует таблицу маршрутизации и помещает в нее данные для всех непосредственно подключенных сетей. В каждом элементе таблицы определяется сеть получателя и указывается расстояние до этой сети, которое, как правило, измеряется количеством переходов (более точное определение этому понятию будет дано чуть позже). Например, в табл. 15.1 показано начальное содержимое таблицы маршрутизации устройства, подключенного к двум сетям. В ней хранится IP-адрес сети, а также целое число, определяющее расстояние до этой сети.

Таблица 15.1. Начальное содержимое таблицы дистанционно-векторной маршрутизации, содержащей данные о каждой непосредственно подключенной сети

| **Получатель** | **Расстояние** | **Маршрут** |
| --- | --- | --- |
| Сеть 1 | 0 | прямой |
| Сеть 2 | 0 | прямой |

Периодически каждый маршрутизатор отсылает копию своей таблицы маршрутизации другим маршрутизаторам, к которым он может получить непосредственный доступ. Когда маршрутизатору *К* приходит сообщение от маршрутизатора *J*, он анализирует полученный в нем набор адресатов и расстояние к каждому из них. Маршрутизатор *К* заменяет данные в своей таблице в следующих случаях:

а) если маршрутизатору *J* известен более короткий маршрут к получателю или в его списке есть получатель, сведения о котором отсутствуют в таблице маршрутизатора *К*;

б) если маршрутизатор *К* выполняет маршрутизацию своих пакетов к конечному получателю через маршрутизатор *J*, но расстояние от маршрутизатора *J* до конечного получателя изменилось. Например, в табл. 14.2(a) показана существующая таблица маршрутизатора *К* и сообщение об обновлении, полученное от маршрутизатора, *J* (б). Отмеченные данные будут использованы для обновления существующей информации или для введения новых данных в таблицу маршрутизатора *К*.

Таблица 15.2. Существующая таблица маршрутизации устройства К (a) и входящее сообщение об обновлении маршрутной информации от маршрутизатора J (б)

| **Получатель** | **Расстояние** | **Маршрут** | **Получатель** | **Расстояние** |
| --- | --- | --- | --- | --- |
| Сеть 1 | 0 | прямой | Сеть 1 | 2 |
| Сеть 2 | 0 | прямой | → **Сеть 4** | **3** |
| Сеть 4 | 8 | к устройству L | Сеть 17 | 6 |
| Сеть 17 | 5 | к устройству M | → **Сеть 21** | **4** |
| Сеть 24 | 6 | к устройству J | Сеть 24 | 5 |
| Сеть 30 | 2 | к устройству Q | Сеть 30 | 10 |
| Сеть 42 | 2 | к устройству J | → **Сеть 42** | **3** |
| (a) |  |  | (б) |  |

Обратите внимание, если маршрутизатор *J* сообщает о расстоянии N до получателя, то в обновленных данных маршрутизатора *К* будет указано расстояние до этого же получателя, равное *N + 1* (т.е. расстояние, которое необходимо преодолеть для достижения получателя от маршрутизатора *J* плюс расстояние *К* самому маршрутизатору *J* от маршрутизатора *К*.)

Разумеется, что в элементах таблицы маршрутизации есть и третий столбец, в котором указывается адрес ближайшей точки перехода. Для всех непосредственно подключенных сетей в поле адреса ближайшей точки перехода отмечается, что они доступны *напрямую*. Если маршрутизатор *К* добавляет или обновляет элемент таблицы маршрутизации в ответ на сообщение, полученное от маршрутизатора *J*, он назначает маршрутизатор *J* в качестве адреса ближайшей точки перехода для этого элемента.

Термин *дистанционный вектор* происходит от типа информации, которая отсылается маршрутизаторами в периодических сообщениях. В сообщении содержится список пар (*V, D*), в которых *V* определяет получателя (называемого вектором), a *D* — расстояние к этому получателю. Обратите внимание, что дистанционно-векторные алгоритмы сообщают маршруты от первого лица (т.е. подразумевается, что маршрутизатор сообщает: “Я могу достичь получателя *V* за *D* переходов”.)

В описанной выше схеме все маршрутизаторы должны принимать участие в дистанционно-векторном обмене для того, чтобы обеспечить эффективную и согласованную работу системы маршрутизации.

Хотя дистанционно-векторные алгоритмы легко реализовать, они имеют ряд недостатков. В полностью статичной среде дистанционно-векторные алгоритмы распространяют информацию о маршрутах, ведущих ко всем получателям. Однако, если маршруты быстро изменяются, результат работы дистанционно-векторного алгоритма может быть нестабильным. При изменении маршрутов (т.е. если появляется новое соединение или выходит из строя старое) информация будет медленно распространяться от одного маршрутизатора до другого. При этом некоторые маршрутизаторы могут содержать неправильную маршрутную информацию.

А сейчас рассмотрим простой протокол, который используется в дистанционно-векторном алгоритме, не обращая внимание на все его недостатки.

1. У данного метода есть несколько синонимов: векторно-дистанционная маршрутизация, а также методы Форда-Фалкерсона (Ford-Fulkerson), Беллмана-Форда и Беллмана. Последние два синонима названы по имени исследователей, которые первыми опубликовали идею алгоритма.

**Межшлюзовый протокол (GGP)**

В первоначальной системе базовой маршрутизации для обмена маршрутной информацией между устройствами использовали дистанционно-векторный протокол, который назывался *межшлюзовый протокол1* (*GGP*, или *Gateway-To-Gateway Protocol*).

Хотя протокол GGP только управлял информацией о классовых маршрутах и сейчас не входит в стандарты протокола TCP/IP2, он все же представляет собой конкретный пример дистанционно-векторной маршрутизации.

Протокол GGP был разработан для передачи маршрутной информации в IP-дейтаграммах, по аналогии с тем, как передаются обычные данные в UDP-дейтаграммах или TCP-сегментах. Каждое сообщение протокола GGP имеет заголовок установленного формата, в котором указывается тип сообщения, а также формат остальных полей.

Поскольку протокол GGP использовался только в базовых маршрутизаторах, а они, в свою очередь, управлялись центральным административным органом Internet, другие маршрутизаторы не могли повлиять на процесс обмена информацией.

Первоначальная система ядра сети была организована так, чтобы к ней можно было подключить новые базовые маршрутизаторы, при этом не изменяя информацию в существующих маршрутизаторах. Когда к ядру системы подключался новый маршрутизатор, для него назначались один или несколько базовых *соседних маршрутизаторов*, с которыми он должен был обмениваться информацией. Соседние маршрутизаторы, входящие в ядро сети, также распространяли маршрутную информацию между другими базовыми маршрутизаторами. Поэтому новому маршрутизатору нужно было лишь сообщить своим соседям о сетях, к которым он имел непосредственный доступ. Соседние маршрутизаторы, в свою очередь, обновляли свои таблицы маршрутизации и распространяли уже обновленную информацию другим маршрутизаторам.

GGP — это дистанционно-векторный протокол. Информация, которой маршрутизаторы обмениваются по протоколу GGP, состоит из парного набора (*N, D*), в котором *N* — это IP-адрес сети, a *D* — расстояние, до этой сети, измеряемое *количеством переходов (hops)*. Таким образом, маршрутизатор с помощью протокола GGP объявляет о сетях, доступ к которым он может получить, а также о затратах, необходимых для получения этого доступа.

В протоколе GGP расстояние измеряется *количеством переходов через маршрутизаторы*. Причем маршрутизатор считается размещенным на нулевом расстоянии от непосредственно соединенных сетей, на расстоянии одного перехода от сетей, доступ к которым можно получить через другой маршрутизатор, и т.д. Таким образом, *количество*, или *счетчик, переходов* по маршруту от отправителя до конечного получателя позволяет судить о количестве маршрутизаторов, через которые проходит дейтаграмма. Очевидно, что использование метода подсчета количества переходов для определения кратчайшего пути не всегда приводит к желаемым результатам. Например, передать дейтаграмму по маршруту с количеством переходов, равным *3*, проходящему по локальным сетям, может оказаться намного быстрее, чем по маршруту с количеством переходов, равным *2*, но проходящему через два низкоскоростных канала последовательной передачи данных. Поэтому во многих маршрутизаторах для маршрутов, проходящих по низкоскоростным сетям, используется искусственно завышенное количество переходов.

1. Напомним, что хотя у производителей сетевого оборудования прижился термин *IP-маршрутизатор*, исследователи первоначально использовали термин IР-шлюз.

2. Группа IETP признала протокол GGP устаревшим, а это означает, что его больше не рекомендуется использовать совместно с семейством протоколов TCP/IP.

**Упорядочение расстояния**

Подобно большинству протоколов маршрутизации, в протоколе GGP используется несколько *типов* сообщений, каждое из которых имеет определенный формат и назначение.

Поле в заголовке сообщения содержит код, который определяет тип сообщения. Благодаря этому коду получатель знает, как обработать сообщение. Например, прежде чем два маршрутизатора обменяются маршрутной информацией, они должны наладить между собой связь, для чего используются сообщения определенного типа.

Основным типом сообщения в протоколе GGP является сообщение об обновлении маршрутной информации, с помощью которого маршрутизаторы обмениваются информацией. Этот тип сообщения также является основным для любого другого дистанционно-векторного протокола.

Теоретически в сообщении об обновлении маршрутной информации должен содержаться список пар, каждая из которых состоит из IP-адреса сети и расстояния до этой сети. Однако на практике во многих протоколах маршрутизации порядок подачи информации изменен, чтобы уменьшить объем передаваемых сообщений. В частности, существует незначительное количество структур, в которых маршрутизаторы и сами объединяемые ими сети размещены в линейном порядке.

Чаще всего можно встретить иерархическую структуру, в которой к каждой сети подключено несколько маршрутизаторов. Следовательно, значения, указывающие на расстояние в сообщениях об обновлении, представляют собой небольшие числа. Кроме того, одни и те же значения могут часто повторяться.

Чтобы уменьшить размер сообщения, в протоколах маршрутизации часто применяется метод, который был впервые использован в протоколе GGP. При использовании этого метода, который называется *упорядочением расстояния (distance factoring)*, копии сообщений с одним и тем же показателем расстояния не отсылаются. Вместо этого список пар сортируется по расстоянию. Причем каждое значение расстояния указывается только один раз, а после него перечисляются сети, доступные на этом расстоянии. Далее в курсе показано, каким образом другие протоколы маршрутизации упорядочивают информацию.

**Надежность и протоколы маршрутизации**

Большинство протоколов маршрутизации используют транспортный протокол, не требующий предварительной установки соединения. Например, протокол GGP инкапсулирует сообщения непосредственно в IP-дейтаграммы. Современные протоколы маршрутизации, как правило, инкапсулируют сообщения в [UDP-дейтаграммы](https://it.rfei.ru/course/%7EGZqN/-/%7E3nCx/%7Egkc8VRIS).

Оба протокола, IP и UDP, не гарантируют доставку дейтаграмм. Это означает, что в процессе доставки дейтаграммы могут затеряться, задержаться в пути, а также появиться их дубли. Кроме того, информация, содержащаяся в дейтаграммах, может быть искажена или доставлена в неправильном порядке. Поэтому протокол маршрутизации, который использует ненадежные транспортные службы доставки пакетов, должен самостоятельно компенсировать сбои в сети.

В протоколах маршрутизации используется несколько методов для решения проблем, связанных с ненадежной доставкой сообщений. Чтобы определить факт искажения информации используется механизм подсчета контрольных сумм. Проблема потери данных решается либо с помощью технологии систем с *неустойчивым состоянием* (*soft state*) на основе таймеров, либо через механизм подтверждений и повторной передачи сообщений. Например, в протоколе GGP применяется расширенная схема подтверждения приема, в которой получатель может после получения сообщения прислать положительный или отрицательный сигнал подтверждения приема.

Проблема нарушения порядка доставки сообщений или поступления соответствующего ответа после получения старого сообщения в протоколах маршрутизации часто решается с помощью *порядковых номеров*. Например, в протоколе GGP каждая из сторон при установке соединения выбирает свой начальный номер, который используется для управления обменом информацией. Другая сторона должна сначала подтвердить этот номер. После начального обмена информацией в каждом посылаемом сообщении указывается следующий по порядку номер, который позволяет получателю определить, в правильном ли порядке получены сообщения.

В последующем разделе будет рассмотрен пример протокола маршрутизации, в котором используется технология систем с неустойчивым состоянием.

Однако существуют и исключения. В следующем разделе будет описан протокол маршрутизации, использующий в качестве транспортного протокол TCP.

Напомним, что в системах с неустойчивым состоянием информация считается устаревшей после истечения тайм-аута, а не после получения специального сообщения от отправителя.

**Маршрутизация, отслеживающая состояние соединения (SPF)**

Самый большой недостаток дистанционно-векторного алгоритма заключается в том, что он недостаточно хорошо приспособлен к изменениям, происходящим в сети. Кроме замедленной реакции на изменения, о которой уже упоминалось, для работы алгоритма требуется, чтобы маршрутизаторы обменивались большими объемами информации. Поскольку в каждом сообщении об обновлении маршрутной информации содержатся данные о каждой из существующих сетей, размер сообщения пропорционален общему количеству сетей, подключенных к объединенной сети. Кроме того, поскольку дистанционно-векторный протокол требует, чтобы в процессе обмена информацией были задействованы все маршрутизаторы, объем передаваемой при этом информации может достигать огромных размеров.

Основной альтернативой дистанционно-векторных алгоритмов является класс алгоритмов, отслеживающих *состояние соединения (link state, link status)*, или выполняющих поиск *первого кратчайшего пути* (*SPF*, или *Shortest Path First*).

Для работы алгоритма поиска кратчайшего пути (SPF) требуется, чтобы каждый задействованный маршрутизатор имел полную информацию о топологической схеме сети. Проще всего понять, что такое топологическая информация, представив себе, что у каждого маршрутизатора есть карта, на которой изображены все остальные маршрутизаторы, а также сети, к которым они подключены. В абстрактном понимании, маршрутизаторы соответствуют вершинам графа, а сети, связывающие маршрутизаторы, соответствуют ребрам графа. Между двумя вершинами графа существует ребро (соединение), если соответствующие им маршрутизаторы могут напрямую обмениваться информацией.

Маршрутизатор, который задействован в алгоритме SPF, не отсылает сообщения, содержащие списки получателей, а выполняет две задачи.

Во-первых, он активно проверяет состояние соседних маршрутизаторов. В терминах теории графов, два маршрутизатора являются соседними, если они пользуются одним и тем же соединением. С точки зрения сетей два соседних маршрутизатора должны быть подключены к общей сети.

Во-вторых, алгоритм SPF периодически распространяет информацию о состоянии соединения всем остальным маршрутизаторам.

Чтобы проверить состояние своего “соседа”, маршрутизатор периодически обменивается с ним короткими сообщениями и таким образом выясняет, в рабочем ли он состоянии и возможен ли к нему доступ.

Если соседний маршрутизатор отвечает на запросы, считается, что соединение между ними *установлено (up)*. В противном случае считается, что соединение *разорвано* *(down)*. Чтобы проинформировать все остальные маршрутизаторы, каждый маршрутизатор периодически рассылает в широковещательном режиме сообщение, в котором указывает состояние каждого своего соединения.

Сообщение о состоянии соединения не определяет маршруты, а просто сообщает, возможен ли обмен информацией между парами маршрутизаторов. Программы протокола маршрутизатора обеспечивают доставку экземпляра каждого сообщения о состоянии соединения всем задействованным маршрутизаторам. Если используемая низкоуровневая сетевая технология не позволяет осуществить режим широковещательной передачи, доставка сообщений осуществляется путем рассылки его отдельных экземпляров каждому из получателей.

Когда приходит сообщение о состоянии соединения, маршрутизатор использует эту информацию, чтобы обновить свою карту объединенной сети, обозначая на ней установленные и разорванные соединения. Каждый раз, когда изменяется состояние соединения, маршрутизатор повторно вычисляет маршруты, применяя хорошо известный *алгоритм поиска кратчайшего пути Дейкстра (Dijkstra)* к результирующему графу. Алгоритм Дейкстры вычисляет кратчайшие пути от одного отправителя ко всем получателям.

Одно из основных преимуществ алгоритмов SPF заключается в том, что каждый маршрутизатор вычисляет кратчайшие пути независимо, используя одни и те же данные о состоянии соединения, которые не зависят от вычислений промежуточных машин. Поскольку сообщения о состоянии соединения распространяются без изменений, то возникающие проблемы легко локализовать. Поскольку маршрутизаторы вычисляют маршруты локально, их правильность гарантирована. И наконец, поскольку в сообщениях о состоянии соединения указывается информация только о тех соединениях, которые непосредственно подключены к данному маршрутизатору, их размер не зависит от количества сетей в объединенной сети. Таким образом, алгоритмы SPF лучше приспособлены к изменениям, происходящим в сети, чем дистанционно-векторные алгоритмы.

Название “первый кратчайший путь” немного сбивает с толку, поскольку алгоритмы во всех маршрутизаторах просто выполняют поиск кратчайшего пути.

На практике для предотвращения колебаний системы из одного состояния в другое в большинстве протоколов используется метод статистической оценки. Соединение считается установленным до тех пор, пока соседний маршрутизатор отвечает на большую часть посланных ему запросов. И наоборот, соединение считается разорванным, если большая часть запросов остается без ответа.

**Проверка знаний: Основы маршрутизации**

**Задание 1**

Узел сети может успешно выполнять маршрутизацию дейтаграмм, если...

он располагает частичной информацией о маршрутизации.

только он располагает полной информацией о маршрутизации

в обоих случаях

**Задание 2**

В чем заключается недостаток использования неполной информации во *внешних маршрутизаторах*?

вероятность возникновения противоречий

появляется вероятность, что маршрутизаторы будут перезагружены

вероятность появления непонятных для маршрутизаторов команд

**Задание 3**

По какому пути должен проходить трафик между двумя географически близко расположенными узлами сети (для большинства конфигураций равноправных сетевых магистралей)?

по кратчайшему

по любому, главное, чтобы он не попал в маршрутную петлю

по самому длинному

**Задание 4**

GGP, состоит из парного набора (N, D), что они означают?

D — это IP-адрес сети, a N — расстояние, до этой сети

D — получатель, a N — расстояние, до этой сети

N — это IP-адрес сети, a D — расстояние, до этой сети

N — получатель, a D — расстояние, до этой сети

**Задание 5**

Создание системы маршрутизации происходит в этап

инициализации

обновления

в два этапа, инициализации и обновления

**Задание 6**

Какова цель механизма *автоматического распространения маршрутов*?

только поиск набора маршрутной информации

поиск и обновление набора маршрутной информации

только обновление набора маршрутной информации

**Взаимодействие частных сетей (NAT, VPN)**

В предыдущих разделах объединенная сеть описывалась как одноуровневая абстракция, которая состоит из сетей, связанных между собой маршрутизаторами. В этом разделе в качестве альтернативы рассмотрена двухуровневая структура объединенной сети, в которой организации имеют частные (или закрытые) объединенные сети, соединенные между собой посредством открытой объединенной сети (или открытых каналов связи).

В этом разделе рассмотрены сетевые технологии, использующиеся в подобной двухуровневой структуре. Одна из технологий предназначена для решения проблемы ограниченного адресного пространства, а другая — позволяет расширить функциональные возможности сетевой технологии, чтобы не допустить посторонних к просмотру закрытых данных. Речь идет об обеспечении *секретности (privacy)* в объединенной сети.

* [Частные и гибридные сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905eb)
* [Виртуальная частная сеть](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905ec)
* [Адресация и маршрутизация в виртуальной частной сети](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905ed)
* [Виртуальная частная сеть с локальными адресами](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905ee)
* [Преобразование сетевых адресов (NAT)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905ef)
* [Создание таблицы преобразования сетевых адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f0)
* [Многоадресная NAT](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f1)
* [Преобразование сетевого адреса с распределением по портам](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f2)
* [Взаимодействие между NAT и ICMP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f3)
* [Взаимодействие между NAT и прикладными программами](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f4)
* [Проверка знаний: Взаимодействие частных сетей (NAT, VPN)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a6905f5)

**Частные и гибридные сети**

Один из основных недостатков одноуровневой структуры сети Internet — отсутствие секретности. Если у организации есть несколько сетевых центров, то при пересылке информации между ними через открытую объединенную сеть дейтаграммы проходят через сети, принадлежащие сторонним организациям. При этом их содержимое может быть просмотрено посторонними лицами. В двухуровневой сетевой структуре проводится различие между *внутренними* и *внешними* дейтаграммами (т.е. дейтаграммами, пересылаемыми между двумя компьютерами в пределах одной организации, и дейтаграммами, пересылаемыми между компьютерами, находящимися в разных организациях). Это сделано для того, чтобы гарантировать *секретность* внутренних дейтаграмм при их пересылке по открытым каналам связи.

Простейшим способом обеспечения секретности при передаче информации между компьютерами организации является создание полностью изолированной *частной* объединенной сети, которую обычно называют *частной сетью*. То есть, организация создает собственную объединенную сеть на основе протокола TCP/IP и изолирует ее от глобальной сети. В каждом сетевом центре частной сети для объединения внутренних сетей используются маршрутизаторы, а для соединения сетевых центров между собой используется закрытый цифровой выделенный канал связи. Поскольку никто из посторонних не имеет доступа ни к какой части закрытой сети, все данные при этом остаются секретными. Более того, поскольку частная сеть изолирована от глобальной объединенной сети, в ней могут использоваться произвольные IP-адреса.

Естественно, полная изоляция не всегда желательна. Поэтому многие организации выбирают *гибридную сетевую* структуру, сочетающую в себе преимущества закрытой сети с возможностью соединения с глобальной сетью Internet. Это значит, что организация подключает каждый сетевой центр к глобальной сети Internet, используя при этом реальные IP-адреса (действительные во всей глобальной сети). Преимуществом подобной структуры является то, что при необходимости компьютеры внутренней сети организации могут получить доступ к глобальной сети Internet, и при этом гарантируется полная секретность информации, пересылаемой по внутренним сетям.

Для примера рассмотрим гибридную сетевую структуру, представленную на рис. 16.1. В подобной системе два сетевых центра организации соединены посредством закрытого канала связи, и каждый из центров имеет соединение с сетью Internet.

Рис. 16.1. Пример гибридной сети. В дополнение к закрытому выделенному каналу связи, соединяющему два сетевых центра, каждый из этих центров имеет соединение с глобальной сетью Internet

На рис. 16.1 закрытый выделенный канал связи между маршрутизаторами R2 и R4, обеспечивает секретность потока информации, передаваемой между сетевыми центрами. Поэтому процесс маршрутизации между центрами организован так, чтобы поток информации преимущественно направлялся через выделенный канал связи, а не через глобальную сеть Internet.

**Виртуальная частная сеть**

Основным недостатком и полностью закрытой сети, и гибридной системы является их высокая стоимость, поскольку аренда выделенного канала связи (например, типа Т1) — довольно дорогое удовольствие. Поэтому многие организации ищут более дешевые альтернативные варианты. Один из путей снижения стоимости заключается в использовании альтернативных сетевых технологий. Например, организация постоянного виртуального канала (permanent virtual circuit, или PVC) на основе технологий ретрансляции кадров (Frame Relay) или ATM через одного из национальных операторов связи может стоить гораздо дешевле, чем выделенный Т-канал с такой же пропускной способностью. Другой путь снижения стоимости состоит в использовании меньшего количества выделенных каналов. Очевидно, что минимальной стоимости системы удается достичь, отказавшись от использования выделенных каналов связи и пустив весь трафик через глобальную сеть Internet.

На первый взгляд может показаться, что использование глобальной сети Internet для взаимосвязи между сетевыми центрами не позволяет достичь такой же секретности, как в случае полностью закрытой сети. Поэтому возникает следующий вопрос.

**Как может организация, которая использует глобальную сеть Internet для соединения своих центров, сохранить секретность передаваемых данных?**

Этого можно достичь с помощью технологии, позволяющей конфигурировать *виртуальную частную сеть (Virtual Private Network*, или *VPN1*. Технология VPN является полным аналогом реальной частной (или закрытой) сети, поскольку гарантирует, что соединение между любой парой компьютеров в VPN остается секретным для посторонних. Слово “виртуальная” в названии технологии VPN говорит о том, что для соединения сетевых центров не используется реальный выделенный канал связи. Вместо этого для передачи информации от одного узла VPN к другому используется глобальная сеть Internet.

В основе работы VPN лежат два фундаментальных понятия: *туннелирование* и *шифрование*. Для построения виртуальных частных сетей используется та же идея. Между маршрутизаторами, находящимися в разных сетевых центрах, прокладывается туннель через глобальную сеть Internet. При этом для пересылки дейтаграмм по туннелю используется инкапсуляция типа *IP-в-IP*.

Несмотря на использование аналогичной идеи, VPN-туннель в значительной степени отличается от туннелей, описанных в предыдущих разделах. В частности, чтобы гарантировать секретность передаваемых данных, в VPN каждая выходная дейтаграмма перед инкапсуляцией в другую дейтаграмму для передачи шифруется2 (рис. 16.2).

Рис. 16.2. Инкапсуляция типа IP-в-IP, используемая в VPN. Для обеспечения секретности внутренняя дейтаграмма зашифровывается перед помещением ее во внешнюю дейтаграмму для отправки

Как показано на рис. 16.2, вся внутренняя дейтаграмма, включая заголовок, зашифровывается перед инкапсуляцией. После прохождения дейтаграммы через туннель, ее область данных расшифровывается принимающим маршрутизатором, в результате чего восстанавливается внутренняя дейтаграмма, которая затем пересылается получателю. Хотя при движении через туннель внешняя дейтаграмма может проходить через любое количество сетей, принадлежащих третьим лицам, посторонние компьютеры не могут расшифровать ее содержимое, поскольку они не знают ключа шифрования. Более того, они даже не смогут определить адреса отправителя и конечного получателя внутренней дейтаграммы, поскольку ее заголовок также зашифрован. Из заголовка внешней дейтаграммы можно определить только IP-адреса маршрутизаторов, расположенных в начале и конце туннеля. Эти адреса помещаются в поля адресов отправителя и получателя внешней дейтаграммы. Из всего сказанного выше можно сделать такой вывод.

В технологии VPN данные пересылаются через открытую сеть Internet, а для обеспечения секретности содержимое всех передаваемых между сетевыми центрами дейтаграмм шифруется.

1. Это название не совсем правильное, поскольку в действительности эта технология обеспечивает создание виртуальной частной объединенной сети.

2. Вопросы безопасности протокола IP и процесс инкапсуляции, использующийся в протоколе IPsec, рассматривается в разделе [Безопасность в объединенной сети и брандмауэры (IPsec)](https://it.rfei.ru/course/%7EGZqN/-/%7E4jjR).

**Адресация и маршрутизация в виртуальной частной сети**

Легче всего понять, как происходит процесс адресации и маршрутизации в виртуальной частной сети, представив себе, что каждый VPN-туннель является аналогом выделенного канала связи в реальной частной сети. Как и в случае частной сети, в маршрутизаторе явно указываются маршруты ко всем внутренним сетям организации. Однако вместо передачи данных по выделенному каналу связи, в VPN данные передаются через туннель.

В качестве примера на рис. 16.3 показана виртуальная частная сеть, которая эквивалентна реальной частной сети, представленной на рис. 16.1, а также содержимое таблицы маршрутизации устройства R1, управляющего туннелированием.

Рис 16.3. Виртуальная частная сеть, которая соединяет два сетевых центра и таблица маршрутизации устройства R1. Туннель между маршрутизаторами R1 и R3 сконфигурирован по типу двухточечного выделенного канала связи

В качестве примера пересылки данных по виртуальной частной сети рассмотрим дейтаграмму, отправленную с компьютера, находящегося в сети 128.10.2.0 на компьютер, находящийся в сети 128.210.0.0. Отправляющий сетевой узел пересылает дейтаграмму маршрутизатору R2, который пересылает ее маршрутизатору R1. Согласно таблице маршрутизации устройства R1 дейтаграмма должна быть отравлена по туннелю маршрутизатору R3. Поэтому маршрутизатор R1 зашифровывает дейтаграмму, помещает ее в область данных внешней дейтаграммы, которую адресует получателю R3. Затем маршрутизатор R1 пересылает внешнюю дейтаграмму маршрутизатору местного провайдера Internet, который отправляет ее по сети Internet маршрутизатору R3. После того как дейтаграмма поступит на маршрутизатор R3, последний анализирует ее содержимое и определяет, что дейтаграмма пришла по туннелю от маршрутизатора R1. Затем устройство R3 расшифровывает область данных и восстанавливает исходную дейтаграмму. После этого просматривается локальная таблица маршрутизации и определяется, что дейтаграмма должна быть переслана устройству R4, для дальнейшей доставки конечному получателю.

**Виртуальная частная сеть с локальными адресами**

Применение технологии VPN предоставляет организациям такие же возможности адресации, как и технологии реальной частной сети. Если сетевые узлы в VPN не подключаются к глобальной сети Internet, то в VPN можно использовать произвольные IP-адреса. Если же сетевые узлы в VPN должны обмениваться данными с глобальной сетью Internet, можно использовать гибридную схему адресации. При такой схеме внутри виртуальной частной сети используются локальные адреса, а для организации через Internet туннеля между сетевыми центрами, их маршрутизаторам назначается по одному реальному IP-адресу (рис. 16.4)

Рис. 16.4. Гибридная адресация в виртуальной частной сети, два полностью закрытых сетевых центра которой соединены через глобальную сеть Internet. Компьютерам каждого сетевого центра назначены локальные адреса

Как показано на рис. 16.4, в сетевом центре 1 используется подсеть 10.1.0.0/16, а в сетевом центре 2 — подсеть 10.2.0.0/16. В такой системе необходимы только два реальных IP-адреса, которые используются для доступа маршрутизаторов R1 и R2 к глобальной сети Internet. В таблицах маршрутизации компьютеров, принадлежащих сетевым центрам, используются локальные адреса. Таким образом, два реальных IP-адреса необходимы только для работы программы, управляющей туннелированием. Две копии этой программы запускаются на маршрутизаторах R1 и R2.

В виртуальных частных сетях используется такая же структура адресации, как и в реальной частной сети. Сетевым узлам в полностью изолированной VPN могут назначаться произвольные адреса. Однако для доступа к глобальной сети Internet должна применяться гибридная система адресации с использованием реальных IP-адресов, действительных во всей сети Internet. Остается открытым вопрос: как может сетевой центр обеспечить доступ к глобальной сети Internet всем своим компьютерам без назначения каждому сетевому узлу реального IP-адреса? Существует два решения этой проблемы.

Первый подход состоит в использовании так называемого *шлюза уровня приложений (application gateway)*. Он позволяет сетевым узлам осуществлять доступ к службам Internet без установки непосредственного соединения с ними на уровне протокола IP. В каждом сетевом центре имеется многоадресный узел, подключенный как к глобальной сети Internet (при этом одному из его интерфейсов назначается реальный IP-адрес), так и к внутренней сети (второму интерфейсу назначается локальный IP-адрес). На этом узле запускается ряд прикладных программ, которые называются *шлюзами уровня приложений*, каждая из которых управляет работой одной из служб. Узлы в сетевом центре не отправляют дейтаграммы непосредственно в глобальную сеть Internet. Они отправляют каждый запрос многоадресному узлу, точнее, соответствующему шлюзу уровня приложений, который в свою очередь обращается к службам глобальной сети Internet, а затем пересылает полученную от них информацию по внутренней сети отправителю запроса.

Основное преимущество использования шлюза уровня приложений заключается в том, что такая система может работать без изменения низкоуровневой сетевой инфраструктуры или системы адресации. Основной недостаток этого подхода состоит в том, что его нельзя считать универсальным и общеприменимым. Напрашивается следующий вывод.

Каждый шлюз уровня приложений управляет работой только одной из служб, поэтому для поддержки нескольких служб приходится использовать несколько разных шлюзов.

Хотя при соблюдении определенных условий шлюзы уровня приложений могут быть полезны, они не позволяют решить проблему доступа в Internet в общем. Поэтому и было придумано второе решение.

**Преобразование сетевых адресов (NAT)**

Для решения общей проблемы доступа между узлами сетевого центра и остальной частью глобальной сети Internet на уровне протокола IP была разработана технология, которая в сетевой терминологии называется *преобразованием сетевого адреса (Network Address Translation*, или *NAT)*. При использовании этой технологии не требуется, чтобы каждый узел в сетевом центре имел реальный IP-адрес. Для ее реализации необходимо, чтобы сетевой центр имел по крайней мере одно соединение с глобальной сетью Internet и один реальный IP-адрес, *G*, действительный во всей глобальной сети Internet. Адрес *G* предназначен для компьютера (многоадресного узла или маршрутизатора), через который сетевой центр подключается к глобальной сети Internet и на котором запущена программа поддержки NAT. Компьютер, на котором запущены средства NAT, называют *NAT-блоком (NAT box)*. Через NAT-блок проходят все дейтаграммы, следующие из сетевого центра в Internet и, наоборот, из Internet в сетевой центр.

Программа поддержки NAT выполняет преобразование адресов как в исходящих, так и во входящих дейтаграммах. При этом адрес отправителя каждой исходящей дейтаграммы заменяется на *G*, а в каждой входящей дейтаграмме выполняется обратное преобразование (т.е. адрес получателя *G* заменяется на локальный адрес соответствующего узла сетевого центра). Таким образом, с точки зрения внешнего получателя все дейтаграммы, отправленные из сетевого центра, приходят от адреса NAT-блока, а все отклики возвращаются на NAT-блок. Для внутренних сетевых узлов NAT-блок является маршрутизатором, который имеет доступ к глобальной сети Internet.

Преимущество технологии NAT заключается в том, что она является универсальной и прозрачной для узлов сети. Кроме того, использование NAT предпочтительнее, чем использование шлюза уровня приложений, поскольку она позволяет обеспечить доступ любого внутреннего узла сети к любой службе глобальной сети Internet. Технология NAT является прозрачной, поскольку она позволяет любому внутреннему узлу сети отправлять и получать дейтаграммы, используя локальные (т.е. немаршрутизируемые) адреса. Подводя итоги можно сказать:

технология преобразования сетевого адреса (Network Address Translation, или NAT) обеспечивает для любого узла сети прозрачный доступ к глобальной сети Internet на уровне протокола IP помощью локальных (т.е. немаршрутизируемых) адресов.

**Создание таблицы преобразования сетевых адресов**

При рассмотрении технологии NAT был опущен важный аспект, а именно: как NAT определяет, какому из внутренних узлов сети предназначена дейтаграмма, пришедшая из глобальной сети Internet. Для решения этой задачи в NAT поддерживается специальная таблица, которая используется для преобразования адресов. В каждом элементе таблицы указывается два значения: IP-адрес узла глобальной сети Internet и внутренний IP-адрес узла сетевого центра. При получении дейтаграммы из глобальной сети Internet программа NAT ищет адрес ее отправителя в таблице преобразования адресов. Когда нужный элемент таблицы найден, из него извлекается соответствующий адрес внутреннего узла сетевого центра, который помещается в поле адреса получателя дейтаграммы1, после чего дейтаграмма пересылается получателю по локальной сети.

Понятно, что таблица преобразования адресов должна составляться заблаговременно. В противном случае при получении из Internet дейтаграммы программа NAT никак не сможет определить адрес внутреннего узла сети, на который нужно ее переправить. Поэтому возникает вопрос: как и когда надо инициализировать таблицу? Существует несколько методов инициализации.

* *Ручная инициализация*. Таблица преобразования адресов создается вручную администратором сети до начала работы программы NAT.
* *Исходящие дейтаграммы*. Таблица создается одновременно с отправкой дейтаграмм. При поступлении дейтаграммы от внутреннего узла сети программа NAT создает элемент в таблице преобразования, в котором фиксируется локальный адрес отправителя и глобальный адрес получателя.
* *Входящие запросы на преобразование имен*. Таблица создается одновременно с обработкой входящих запросов на преобразование доменных имен. Когда узел в глобальной сети Internet посылает запрос на преобразование доменного имени2, соответствующего внутреннему узлу сети, в IP-адрес, сервер имен создает элемент в таблице NAT, а затем отвечает на запрос, отсылая адрес *G*. Таким образом, для внешнего получателя все выглядит так, как будто именам внутренних узлов сети соответствует один глобальный адрес *G*.

Каждый из методов инициализации имеет преимущества и недостатки. Ручная инициализация обеспечивает постоянную привязку адресов и позволяет отправлять IP-дейтаграммы в любое время в любом направлении. Использование исходящих дейтаграмм позволяет автоматизировать процесс инициализации таблицы преобразования адресов, однако, с другой стороны, при использовании этого метода невозможно инициировать соединение извне. Применение третьего метода инициализации требует модификации программного обеспечения, обрабатывающего запросы на преобразование доменных имен. Этот метод позволяет инициировать соединение из-за пределов сетевого центра, однако он работает только в том случае, если отправитель посылает запрос на преобразование доменного имени до отправки дейтаграмм.

В большинстве реализаций NAT для инициализации таблицы преобразования адресов используется метод исходящих дейтаграмм. Он наиболее популярен среди поставщиков услуг Internet. Чтобы понять, почему, рассмотрим мелкого провайдера Internet, обеспечивающего своим клиентам доступ к сети по коммутируемому соединению с помощью обычного модема. Структурная схема такой системы показана на рис. 16.5.

Рис. 16.5. Структурная схема сети мелкого провайдера Internet, в которой используется технология NAT, и коммутируемый доступ к сети со стороны клиентов. Применение NAT позволяет провайдеру назначать клиентам локальные IP-адреса

Каждый раз при подключении клиента к сети провайдера ему назначается IP-адрес. Технология NAT позволяет провайдеру Internet присваивать всем своим клиентам локальные IP-адреса. Например, первому клиенту присваивается адрес 10.0.0.1, второму— 10.0.0.2 и т. д. Когда клиент отправляет дейтаграмму получателю, расположенному в сети Internet, программа NAT использует содержимое ее заголовка для инициализации таблицы преобразования адресов.

1. Естественно, всякий раз при замене IP-адреса в заголовке дейтаграммы программа NAT должна пересчитать значение контрольной суммы заголовка.

2. О том, как работает система доменных имен (Domain Name System, или DNS), речь пойдет в разделе [Система доменных имен (DNS)](https://it.rfei.ru/course/%7EGZqN/-/%7EhOtY).

**Многоадресная NAT**

Выше была описана работа технологии NAT в самом простом случае, когда между внешними и внутренними адресами существует однозначное соответствие. Другими словами, однозначное соответствие позволяет только одному компьютеру в сетевом центре получить доступ в произвольный момент времени к некоторой машине в глобальной сети Internet. Однако на практике используются более сложные формы NAT, которые позволяют различным узлам сетевого центра получать доступ к одному внешнему получателю.

В одном из вариантов NAT одновременный доступ клиентов осуществляется за счет сохранения принципа однозначного соответствия адресов в NAT-блоке, которому дополнительно назначено несколько реальных IP-адресов. Этот метод называется *многоадресной NAT*. Суть его состоит в том, что NAT-блоку назначается *k* реальных IP-адресов, действительных во всей глобальной сети Internet, G1, G2,... Gk. Когда первый внутренний узел сети отправляет дейтаграмму получателю в Internet, NAT-блок выбирает адрес G1, от которого она будет послана в Internet, добавляет элемент в таблицу преобразования адресов и отправляет дейтаграмму получателю. Если другой узел сети попытается отправить дейтаграмму тому же получателю, NAT-блок выберет адрес G2, и т.д. Таким образом, *многоадресная NAT- позволяет* k\_ внутренним узлам сети одновременно получать доступ к одному получателю.

**Преобразование сетевого адреса с распределением по портам**

Другой широко распространенный вариант NAT обеспечивает одновременный доступ клиентов путем преобразования не только IP-адресов, но и номеров портов протоколов TCP или UDP. Иногда этот метод называют *преобразованием сетевого адреса и порта (Network Address Port Translation*, или *NAPT*). В нем таблица преобразования сетевых адресов расширяется за счет присоединения дополнительных полей. Кроме пары IP-адресов (отправителя и получателя), в таблице указывается пара номеров портов (локального и удаленного), а также номер порта протокола, используемый NAT-блоком. Пример таблицы преобразования адресов приведен в табл. 20.1.

Таблица 16.1. Пример таблицы преобразования адресов, используемой в NAPT

| **Локальный адрес** | **Локальный порт** | **Внешний адрес** | **Внешний порт** | **Порт NAT** | **Используемый протокол** |
| --- | --- | --- | --- | --- | --- |
| 10.0.0.5 | 21023 | 128.10.19.20 | 80 | 14003 | tcp |
| 10.0.0.1 | 386 | 128.10.19.20 | 80 | 14010 | tcp |
| 10.0.2.6 | 26600 | 207.200.75.200 | 21 | 14012 | tcp |
| 10.0.0.3 | 1274 | 128.210.1.5 | 80 | 14007 | tcp |

В табл. 16.1 показаны элементы для четырех внутренних компьютеров, которые в настоящий момент имеют доступ к Internet. Для всех соединений используется протокол TCP. Интересно, что в таблице показаны два внутренних узла сети— 10.0.0.5 и 10.0.0.1, которые оба осуществляют подключение через порт *80* протокола (Web-сервер) к компьютеру 128.10.19.20. В данном случае оказалось, что номера двух локальных портов, использующихся для двух соединений, отличаются. Однако уникальность номера локального порта нельзя гарантировать, поскольку два внутренних узла сети могут совершенно случайно выбрать один и тот же номер локального порта. Поэтому, чтобы избежать возможного конфликта, в NAT каждому соединению с сетью Internet присваивается уникальный номер порта. Напомним, что в протоколе TCP каждое соединение идентифицируется набором из четырех взаимосвязанных величин (4-мерный кортеж), которые представляют собой IP-адреса и номера портов отправителя и конечного получателя. Таким образом, первые два элемента в таблице преобразования адресов соответствуют TCP-соединениям, которые внутренние узлы сети идентифицируют при помощи следующих 4-мерных кортежей:

(10.0.0.5, 21023,128.10.19.20, 80)

(10.0.0.1, 386,128.10.19.20, 80)

В свою очередь, компьютер в сети Internet, получивший дейтаграммы после преобразования адреса с помощью механизма NAPT, идентифицирует те же два соединения с помощью двух других 4-мерных кортежей:

(G,14003, 128.10.19.20, 80)

(G, 14010, 128.10.19.20, 80) ,

где *G* — реальный IP-адрес NAT-блока.

Основное преимущество технологии NAPT — ее универсальность, которая достигается за счет применения лишь одного реального IP-адреса. Ее основной недостаток состоит в том, что в процессе взаимодействия стороны должны использовать только протоколы TCP или UDP. Только в этом случае технология NAPT позволяет без взаимных помех внутреннему компьютеру получить доступ к нескольким внешним компьютерам, а также нескольким внутренним компьютерам получить доступ к одному и тому же внешнему компьютеру. Под номер порта отводится 16 бит, поэтому одновременно до 216 пар приложений могут одновременно взаимодействовать между собой. Можно сделать следующий вывод.

Существует несколько вариантов реализации технологии NAT, включая ее популярную форму NAPT, в которой преобразование IP-адреса выполняется на основе номеров портов протокола TCP или UDP.

**Взаимодействие между NAT и ICMP**

Даже самые простые изменения, внесенные в структуру IP-адресации, могут привести к самым неожиданным последствиям в протоколах более высокого уровня. В частности, чтобы сохранить иллюзию “прозрачности” технологии NAT, в ней должна поддерживаться обработка ICMP-сообщений. Предположим, например, что внутренний сетевой узел использует программу ping для проверки доступности получателя в сети Internet. При этом программа посылает получателю ICMP-запрос на эхо и переходит в режим ожидания отклика.

Таким образом, в технологии NAT входящие ICMP-сообщения, являющиеся откликами на посланные ICMP-запросы, должны пересылаться на соответствующие узлы сети.

Однако на самом деле далеко не все ICMP-сообщения, полученные из Internet, пересылаются локальным получателям. Например, если NAT-блок содержит некорректные маршруты, то ICMP-сообщение о *перенаправлении* должно быть обработано локально. Поэтому когда ICMP-сообщение приходит из сети Internet, NAT-блок должен первым делом определить, нужно ли обработать его локально или отправить на внутренний узел сети для дальнейшей обработки. Прежде чем направить ICMP-сообщение на внутренний узел сети, NAT-блок должен преобразовать в нем адреса.

Чтобы понять необходимость преобразования адресов, рассмотрим ICMP-сообщение о недоступности получателя. В нем содержится заголовок дейтаграммы, *D*, которая вызвала ошибку. К сожалению, NAT-блок преобразует адреса в заголовке дейтаграммы *D* до ее отправления. Поэтому в поле адреса ее отправителя не будет указан IP-адрес внутреннего узла сети. Следовательно, перед пересылкой ICMP-сообщения внутреннему узлу сети, NAT-блок должен открыть его содержимое и преобразовать адреса в заголовке дейтаграммы *D* так, чтобы они соответствовали адресам, используемым внутренним узлом сети. После внесения изменений NAT-блок должен пересчитать контрольную сумму заголовка дейтаграммы *D*, контрольную сумму заголовка ICMP-сообщения и контрольную сумму заголовка внешней дейтаграммы.

**Взаимодействие между NAT и прикладными программами**

В предыдущем разделе было показано, что поддержка протокола ICMP в NAT-блоке существенно усложняет его реализацию. Однако еще хуже дело обстоит с поддержкой протоколов уровня приложений. Через NAT не будут работать те приложения, которое пересылают IP-адреса или номера портов в качестве данных. Например, когда две программы используют *протокол передачи файлов (File Transfer Protocol*, или *FTP*) для соединения между собой они используют протокол TCP. Согласно протоколу (FTP), одна из программ должна получить на локальной машине номер TCP-порта, преобразовать его в ASCII-код и отправить результат через TCP-соединение другой программе. Если пакеты между этими программами по пути от внутреннего узла сети до узла в сети Internet проходят через NAPT-блок, то номер порта, помещенный в поток данных внутренним узлом сети, должен быть изменен на номер порта, выбранный NAPT-блоком. По сути, если NAPT-блок не сможет открыть поток данных и изменить в нем номер порта, то программы не смогут обменяться данными по протоколу FTP. На сегодняшний день существуют реализации NAT, которые автоматически распознают дейтаграммы популярных протоколов, наподобие FTP, и делают необходимые изменения в потоке данных. Однако существуют приложения, которые не могут обмениваться данными через NAT. Резюме такое.

NAT-блок изменяет содержимое ICMP-сообщений, а также дейтаграмм, относящихся к протоколам более высокого уровня. Поэтому, за исключением нескольких стандартных протоколов наподобие FTP, протоколы уровня приложений, которые передают IP-адреса или номера портов в потоке данных, не будут правильно работать при использовании NAT.

Передаваемые в потоке данных элементы, которые должны быть изменены NAPT-блоком, увеличивают сложность его реализации по двум причинам.

Во-первых, это означает, что NAPT-блок должен располагать подробной информацией о формате передаваемых каждым приложением данных (для того, чтобы “на ходу” внести в них изменения).

Во-вторых, если приложения передают номера портов в ASCII-коде, как в случае протокола FTP, то их изменение может повлечь за собой изменение общего количества передаваемых через TCP-соединение данных.

Вставить даже один дополнительный октет в поток данных крайне трудно, поскольку каждый октет в потоке имеет порядковый номер. Поскольку отправитель не знает, что в процессе передачи потока данных в него были вставлены дополнительные данные, он продолжает присваивать порядковые номера октетам без учета этих данных. Более того, получатель будет воспринимать дополнительные данные как часть потока, передаваемого отправителем и присылать на них сигналы подтверждения приема. Таким образом, после вставки NAT-блоком дополнительных данных должны быть изменены порядковые номера октетов в каждом выходящем сегменте и каждом входящем сигнале подтверждения приема.

**Проверка знаний: Взаимодействие частных сетей (NAT, VPN)**

**Задание 1**

В технологии VPN данные пересылаются через открытую сеть Internet. Что делается для обеспечения секретности содержимого, передаваемых между сетевыми центрами дейтаграмм?

шифруются

выставляется уникальный пароль на каждую дейтаграмму

кодируются

**Задание 2**

Какие протоколы, передающие IP-адреса или номера портов в потоке дынных, не будут правильно работать при использовании NAT?

протоколы уровня приложений

все стандартные протоколы FTP

протоколы TCP

**Задание 3**

Как передаются данные в VPN?

через туннель

по выделенному каналу связи

по протоколу ICMP

**Задание 4**

В чем преимущество технологии NAT? Выберите два верных ответа.

скорость обработки данных

универсальность

надежность

является прозрачной для узлов сети

**Задание 5**

Как называются сети, в которых сочетаются преимущества частных сетей и возможность соединения с глобальной сетью Internet?

гибридные

двоичные

перекрестные

**Задание 6**

Работой какого количества служб может управлять каждый шлюз уровня приложений?

десятью

одной

пятью

тремя

**Задание 7**

Как называют компьютер, на котором запущены средства NAT?

NАТ-сервером

NAT-администратором

NAT-блоком

**Задание 8**

Как называется *метод инициализации*, когда таблица преобразования адресов создается администратором сети до начала работы программы NAT?

исходящая дейтаграмма

ручная

входящие запросы на преобразование имен

**Задание 9**

На основе чего выполняется преобразование IP-адреса в методе NAPT?

номеров порта протокола UDP

номеров портов протокола TCP и UDP

номеров порта протокола TCP

**Система доменных имен (DNS)**

В предыдущих разделах были описаны протоколы, в которых для идентификации машин используются 32-битовые целые числа, называемые адресами протокола IP, или IP-адресами. Они обеспечивают удобную и компактную форму идентификации отправителя и получателя пакетов, пересылаемых по объединенной сети. Тем не менее пользователи предпочитают присваивать машинам удобопроизносимые и легко запоминающиеся имена.

В этом разделе рассматривается система назначения значащих высокоуровневых имен для большой совокупности машин и обсуждается механизм преобразования между высокоуровневыми именами и IP-адресами. Мы рассмотрим — преобразование высокоуровневых имен в IP-адреса и IP-адресов — в высокоуровневые имена машин. Назначения имен представляет интерес по двум причинам.

Во-первых, она используется для идентификации машин по имени во всей глобальной сети Internet.

Во-вторых, для преобразования имен в адреса используется набор распределенных серверов, которые находятся на значительном расстоянии друг от друга.

* [Идея присвоения имен машинам](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690602)
* [Одноуровневая система имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690603)
* [Иерархическая система имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690604)
* [Делегирование полномочий в системе имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690605)
* [Дальнейшее разделение полномочий](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690606)
* [Имена доменов сети Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690607)
* [Официальные и неофициальные имена доменов в глобальной сети Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690608)
* [Именованные элементы и синтаксис имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690609)
* [Преобразование доменных имен в адреса](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060a)
* [Распознавание доменных имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060b)
* [Эффективное преобразование имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060c)
* [Кэширование — залог эффективности](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060d)
* [Формат сообщений системы доменных имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060e)
* [Сжатый формат представления имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69060f)
* [Сокращение доменных имен](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690610)
* [Инверсные преобразования](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690611)
* [Запросы указателя](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690612)
* [Типы объектов и содержимое записи ресурса](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690613)
* [Получение полномочий для управления поддоменом](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690614)
* [Проверка знаний: Система доменных имен (DNS)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690615)

**Идея присвоения имен машинам**

Пользователи, работавшие с первыми компьютерными системами, были вынуждены мириться с тем, что для таких объектов, как системные таблицы и периферийные устройства, назначались цифровые адреса. Прогресс в деле обработки данных был достигнут благодаря системам, работающим в режиме разделения времени. Пользователи получили возможность назначать значащие символические имена как для физических объектов, (например, периферийных устройств), так и для абстрактных объектов (например, файлов). Подобная система идентификации сохранилась и при создании объединенной сети вычислительных машин. В первых системах взаимодействие между машинами обеспечивалось благодаря двухточечным соединениям, проложенным между компьютерами, а для идентификации машин использовался низкоуровневый физический адрес. Для обеспечения межсетевого обмена была придумана универсальная система адресации. В ней с помощью специальных программ выполнялось преобразование универсальных адресов в низкоуровневые физические адреса сетевых адаптеров. Поскольку в большинстве вычислительных сред содержится большое количество машин, для их идентификации пользователям требуются значащие символические имена.

Первые системы именования машин были ориентированы на работу в небольшом сетевом окружении. Имена машин в сетевых центрах с небольшим количеством машин обычно выбирались исходя из их назначения. Например, машины часто имели имена типа research (исследовательская лаборатория), production (производство), accounting (бухгалтерия) и development (разработчики). Именно такие имена предпочитают употреблять пользователи вместо громоздких и трудно запоминающихся физических адресов.

Хотя на интуитивном уровне различие между *адресом* и *именем* понятно, оно несколько искусственно, поскольку любое *имя* — это всего лишь идентификатор, который состоит из последовательности буквенно-цифровых символов, выбранных из алфавита ограниченного размера.

Использование имен оправданно только в том случае, если система может эффективно соотнести их с объектом, который они обозначают. Поэтому IP-адрес является не чем иным, как *низкоуровневым именем*, а имя, присваиваемое пользователем своему компьютеру — *высокоуровневым*.

Выбор формы высокоуровневого имени очень важен, поскольку именно она определяет способ преобразования этого имени в низкоуровневое (или способ привязки его к объекту), а также степень ответственности за выполнение этой операции.

Выбор системы именования для небольшой сети, в которой взаимодействует несколько машин, не представляет особого труда, поскольку здесь подойдет любая форма. Однако в глобальной сети Internet, связывающей приблизительно сто миллионов машин, выбор символических имен становится непростой задачей. Например, когда в 1980 году главный ведомственный компьютер факультета информатики университета Пердью (Purdue University) был подсоединен к глобальной сети Internet, для его идентификации было выбрано имя purdue. При этом список зарезервированных имен едва превышал десяток. Однако уже к середине 1986 года список официально зарегистрированных имен узлов сети Internet содержал 3100 официально зарегистрированных имен и 6500 официальных псевдонимов1. Несмотря на то, что этот список быстро пополнялся, в середине 80-х годов XX века в большинстве сетевых центров имелись машины (например, персональные компьютеры пользователей), которым не было назначено официальное имя.

1. К 1990 году более 137 000 сетевых узлов Internet имели имена, а к 2000 году их число превысило 60 миллионов.

**Одноуровневая система имен**

Одной из первых систем именования машин, использовавшейся долгое время в Internet, была *одноуровневая система имен* (flat namespace), т.е. каждое имя состояло из последовательности символов без какой-либо структуризации. В первоначальной системе пространством имен управлял *сетевой информационный центр (Network Information Center*, или *NIC*). Он определял допустимость использования нового имени, то есть запрещалось использовать непристойные имена, а также новые имена, которые находились в противоречии с уже существующими именами.

Основным преимуществом одноуровневой системы именования были короткие имена, которыми очень удобно пользоваться. Главный же недостаток такой системы заключался в том, что ее нельзя было применить для больших наборов машин как в силу технических, так и административных причин.

Во-первых, поскольку все имена были одноуровневыми и состояли из одного набора символов, вероятность конфликта возрастала с увеличением количества зарегистрированных имен.

Во-вторых, поскольку право добавления новых имен должно принадлежать одному сетевому административному органу, с увеличением количества зарегистрированных узлов сети возрастает административная нагрузка на этот орган. Чтобы понять серьезность проблемы, представьте себе быстро растущую объединенную сеть, к которой подключено несколько тысяч сетей предприятий. В каждой из них могут находиться сотни или тысячи персональных компьютеров и рабочих станций. Каждый раз, когда кто-либо приобретает и подключает к сети новый персональный компьютер, имя этого компьютера должно быть утверждено центральным административным органом.

В-третьих, поскольку привязка имени к адресу часто меняется, стоимость поддержки достоверных копий полного списка имен на каждом узле сети высока и возрастает с увеличением количества узлов. С другой стороны, если база данных имен будет находиться на одном сервере, поток данных к этому серверу будет расти с увеличением количества узлов объединенной сети.

**Иерархическая система имен**

Как изменить систему назначения имен так, чтобы она легко справлялась с большим, быстро расширяющимся пространством имен и при этом для ее управления не требовалось бы вмешательства центрального административного органа Internet? Решение этой проблемы состоит в децентрализации механизма назначения имен, когда полномочия по управлению частью пространства имен делегируются другим административным органам. При этом процесс преобразования имен в адреса и наоборот, должен выполняться распределенным наборов серверов. Подобная система используется в объединенной сети на основе протокола TCP/IP. Перед изучением деталей этой системы, рассмотрим основные идеи, лежащие в основе ее работы.

Пространство имен должно быть разделено так, чтобы можно было выполнять эффективное преобразование имен в адреса и гарантировать автономность процесса назначения имен. Если оптимизацию проводить только с точки зрения эффективности процесса преобразования имен, то в конечном счете мы придем к выводу, что нужно сохранить старую одноуровневую систему именования. При этом для уменьшения количества запросов на преобразование, поступающих к одному серверу, все пространство имен следует разделить между несколькими физическими машинами. Если же проводить оптимизацию только с точки зрения облегчения административного управления, то это может привести к принятию решений, которые облегчают процесс делегирования полномочий, но делают процесс преобразования имен дорогим или сложным.

Чтобы понять, как следует разделить пространство имен, рассмотрим внутреннюю структуру больших организаций. Стоящий на вершине власти президент или генеральный директор имеет полную власть. Поскольку он не может следить за всем, что происходит внутри организации, ее разделяют на подразделения, которые возглавляют отдельные руководители. Обычно президент предоставляет каждому руководителю подразделения ограниченную автономию. Другими словами, руководитель каждого подразделения может нанимать или увольнять служащих, изменять штатное расписание и делегировать полномочия другим руководителям, не получая прямой санкции от президента.

Иерархическая структура большой организации, помимо облегчения делегирования полномочий, позволяет ввести также автономное управление. Например, когда работнику офиса нужна информация о номере телефона нового служащего, то он сначала спрашивает об этом у ближайших сотрудников, которые могут входить в контакт со служащими из других подразделений. Дело в том, что, хотя полномочие всегда передается через корпоративную иерархию сверху вниз, информация может передаваться через эту иерархию между подразделениями.

**Делегирование полномочий в системе имен**

Принципы управления иерархической системой назначения имен во многом напоминают принципы управления большой организацией. Пространство имен *разделяется* на зоны на высшем уровне, а полномочия для назначения имен в отдельных зонах предоставляют специальным агентам. Например, можно разделить пространство имен, взяв за основу *имя сетевого центра*, и делегировать каждому сетевому центру полномочия по поддержанию имен в пределах своей зоны. Деление на зоны и делегирование полномочий административным органам каждой зоны происходит на самом верхнем уровне иерархии. Дальнейшие изменения пространства имен в пределах зоны могут выполняться без участия административных органов верхнего уровня иерархии.

Синтаксис имен в иерархической системе зачастую отражает принцип делегирования полномочий того административного органа, который зарегистрировал эти имена. В качестве примера рассмотрим пространство имен, в котором имена имеют вид:

local.site

Здесь site — имя сетевого центра, зарегистрированное центральным административным органом, local — часть имени, назначаемая локальным сетевым центром, и точка (”.”) — разделитель, отделяющий две части имени. Когда центральный административный орган регистрирует новый сетевой центр *X*, он добавляет его имя в список зарегистрированных сетевых центров и делегирует сетевому центру *X* полномочия по управлению всеми именами, которые заканчиваются на *”. X”*.

**Дальнейшее разделение полномочий**

В иерархической системе пространство имен может подразделяться на более мелкие зоны на каждом уровне иерархии. Выше был описан пример разделения пространства имен между сетевыми центрами верхнего уровня. Однако в сам сетевой центр верхнего уровня может входить несколько административных групп (*group*). Администрация этого сетевого центра имеет полномочия разделить его пространство имен между мелкими группами, чтобы обеспечить легкость и удобство управления.

Синтаксически такое разделение пространства имен приводит к добавлению еще одного имени, отделенного точкой, в первоначальную иерархическую структуру. Например, разделение пространства имен, выделенного сетевому центру, между мелкими административными группами приведет к тому, что имена в пределах этих групп будут иметь следующий вид:

local.group.site

Поскольку центральный административный орган делегировал полномочия по управлению частью пространства имен сетевому центру верхнего уровня, центр не должен согласовывать имена мелких административных групп с другими сетевыми центрами верхнего уровня. Например, университетский сетевой центр может выбрать следующие имена для групп: engineering (инженерная группа), science (научная группа) и arts (группа художников). Корпоративный сетевой центр скорее всего назначит группам следующие имена: production (производственный отдел), accounting (бухгалтерия) и personnel (отдел кадров).

Еще одним примером иерархической системы именования является обычная телефонная сеть. Десять цифр телефонного номера разбиты на трехзначный *междугородный телефонный код*, трехзначный *номер АТС (автоматической телефонной станции)* и четырехзначный *номер абонента* в пределах АТС. Каждая АТС имеет полномочия для назначения номеров абонентов в пределах своей части пространства имен (точнее, номеров). Хотя вполне возможно, сгруппировав абонентов разных АТС, назначить им один номер АТС, а АТС разных городов можно сгруппировать и назначить им один и тот же междугородный телефонный код. При назначении телефонных номеров обычно соблюдают определенный порядок. Это облегчает маршрутизацию звонков в телефонной сети.

Пример с телефонной сетью очень важен, поскольку с его помощью было показано основное различие между иерархической системой именования, используемой в объединенной сети TCP/IP, и других иерархиях. Суть состоит в том, что разделение адресного пространства по принципу принадлежности к одному административному органу верхнего уровня отнюдь не означает, что это разделение происходит по физическому положению узлов сети. Например, может случиться так, что в одном здании некоторого университета находятся и факультет математики, и факультет информатики. Может даже оказаться, что, хотя машины этих двух групп находятся в полностью отдельных административных доменах, они подключены к общей физической сети. Кроме того, машины, принадлежащие к одной административной группе, могут находиться в разных физических сетях. Поэтому принятая в протоколе TCP/IP система именования не привязана к физическим сетевым подключениям и позволяет делегировать полномочия по управлению частью пространства имен произвольным административным группа. Эта концепция может быть сформулирована так.

Иерархические имена машин в объединенной сети TCP/IP присваиваются в соответствии со структурой организации, которой переданы полномочия по управлению частью пространства имен. Сам процесс назначения имени никак не связан со структурой физических сетевых подключений.

Конечно, во многих сетевых центрах иерархическая структура организации соответствует структуре физических подключений к сети. Например, у большинства отделов в большом университете есть собственная локальная сеть. Если отдел уполномочен управлять частью иерархии пространства имен, то все машины, имена которых принадлежат к его части иерархии, будут также соединены в единую физическую сеть.

**Имена доменов сети Internet**

Механизм, с помощью которого реализована иерархическая система именования машин в объединенной сети TCP/IP, называется *системой доменных имен (Domain Name System*, или *DNS*).

DNS имеет два концептуально независимых аспекта. Первый из них является абстрактным: он определяет синтаксис имен и правила делегирования полномочия по управлению частью пространства имен. Второй — конкретный: он определяет работу распределенной вычислительной системы, которая эффективно преобразует имена в адреса.

В этом разделе рассмотрен синтаксис имен, а в следующих разделах —практическая реализация механизма преобразования имен.

В принятой системе именования используется иерархический метод присвоения имен, которые называются *доменными именами*. Как и в примерах, приведенных в начале разделах, доменное имя состоит из набора алфавитно-цифровых символов (строк), разделенных точкой. В предыдущих примерах говорилось, что отдельные части имени могут соответствовать сетевым центрам или административным группам, но в системе доменных имен каждая часть называется *меткой (label)*. Таким образом, имя домена

cs.purdue.edu

состоит из трех меток: cs, purdue, и edu. Любой суффикс метки в имени домена также называется *доменом*.

В вышеупомянутом примере домен самого низкого уровня — cs.purdue.edu (имя домена факультета информатики университета Пердью), домен второго уровня— purdue.edu (имя домена университета Пердью), и домен верхнего уровня — edu (имя домена образовательных учреждений). Как видно из примеров, в начале имени домена указывается локальная метка, за которой следует имя домена высшего уровня. Как будет показано ниже, подобная форма записи позволяет сжимать сообщения, содержащие повторяющиеся имена доменов.

**Официальные и неофициальные имена доменов в глобальной сети Internet**

Теоретически стандарт доменных имен определяет абстрактное иерархическое пространство имен с произвольными значениями меток. Поскольку доменная система определяет только форму имен, а не их фактические значения, то любая административная группа может создать собственную независимую систему доменных имен и по своему усмотрению выбрать названия для меток всех частей иерархии. Например, частная компания может создать собственную иерархию доменов, в которой метки самого верхнего уровня будут определять филиалы корпорации, метки следующего уровня — подразделения в филиале, а метки нижнего уровня — отделы.

Однако большинство пользователей придерживается иерархии меток, официально принятых в системе доменных имен сети Internet. Это происходит по двум причинам.

Во-первых, как станет ясно позже, структура сети Internet — универсальная и гибкая. Она может применяться в большом числе организаций и позволяет каждой административной группе выбирать свой метод разделения пространства имен: по территориальному и по организационному признаку.

Во-вторых, структура большинства сетевых центров соответствует структуре, принятой в глобальной сети Internet. Это позволяет им подключать свои сети к Internet, не изменяя при этом имен отдельных машин и доменов.

Поскольку принятая в Internet система назначения имен подходит практически для всех случаев, во всех примерах, приведенных далее, используются метки, взятые из иерархии глобальной сети Internet. Однако не стоит забывать, что, хотя эти метки встречаются чаще всего в Internet, система доменных имен позволяет при желании создать любую другую иерархию меток.

Центральным административным органом глобальной сети Internet утверждены несколько официальных доменов верхнего уровня, перечисленные в табл. 17.1. Кроме того, предложено для использования еще несколько дополнительных доменов верхнего уровня, но они не были формально приняты: .firm, .store, .web, .arts, .reс, .info и .nom. Следует отметить, что обработка меток имени домена выполняется без учета регистра их символов. Поэтому домен .EDU эквивалентен .edu.

Таблица 17.1. Список официальных доменов верхнего уровня глобальной сети Internet и их описание

| **Имя домена** | **Описание** |
| --- | --- |
| . com | Коммерческие организации |
| . edu | Образовательные учреждения (4-летние) |
| . gov | Государственные учреждения |
| .mil | Военные группы |
| .net | Основные центры поддержки сети Internet |
| . org | Организации, не упомянутые выше |
| . arpa | Временный домен ARPANET (устаревший) |
| . int | Международные организации |
| Географический код страны | Организации, относящиеся к стране с указанным географическим кодом (территориальная схема разделения пространства имен) |

Концептуально принятая система имен доменов верхнего уровня позволяет использовать две различные иерархии присвоения имен: территориальную и организационную. В территориальной системе машины разделяются по географическому положению. Например, компьютеры, подключенные к сети и территориально расположенные в Соединенных Штатах Америки, относятся к домену верхнего уровня .us. Если какая-либо страна захочет зарегистрировать собственный географический домен, то центральный административный орган Internet создаст для нее новый домен верхнего уровня и присвоит ему метку в соответствии со стандартным международным двухбуквенным классификатором страны. Например, администрация домена .us поделила его на домены второго уровня, соответствующие каждому штату США. Скажем, для штата Виргиния домен второго уровня выглядит так:

va.us

Кроме территориальной системы иерархии, принятая система именования доменов верхнего уровня позволяет сгруппировать организации по типу. Если организация хочет зарегистрировать собственный домен второго уровня, она должна сообщить центральному административному органу его имя, а также имя домена верхнего уровня, к которому он будет относиться. Рассмотрев заявку, центральный административный орган выделяет организации *поддомен*1 в одном из существующих доменов верхнего уровня. Например, администрация университетской сети может зарегистрировать на себя домен второго уровня в домене .edu (обычная практика) либо в домене штата и страны, в которой расположен университет. Пока что немногие организации выбрали территориальную иерархию имен. Большинство предпочитает регистрироваться в доменах .com, .edu, .mil или .gov. Это происходит по двум причинам.

Во-первых, географические имена длиннее и, соответственно, более трудны для запоминания и набора.

Во-вторых, поскольку географические имена менее очевидны, их труднее соотнести с названием организации. Например, университет Пердью расположен в городе Вест Лафейетт (West Lafayette), штат Индиана.

С точки зрения пользователя очень легко запомнить или угадать доменное имя организации, типа purdue.edu. Что касается территориальных имен, то конструкция типа laf.in.us не выдерживает никакой критики, хотя она и состоит из официально зарегистрированных региональных кодов.

С помощью следующего примера мы проиллюстрируем принцип передачи полномочий при регистрации иерархических имен. Одна из машин факультета информатики университета Пердью названа xinu; ее официальное доменное имя

xinu.cs.purdue.edu

Это имя было авторизовано и зарегистрировано администратором локальной сети факультета информатики, который получил полномочия на управление поддоменом cs.purdue.edu от администрации университетской сети, которая, в свою очередь, получила полномочия на управление поддоменом purdue.edu от центрального административного органа Internet. Последний сохраняет контроль над доменом .edu, поэтому новые домены второго уровня для учебных заведений могут быть зарегистрированы только с его разрешения. Точно так же администрация сети университета Пердью сохраняет свои полномочия по управлению доменом purdue.edu. Поэтому новые домены третьего уровня могут быть добавлены только с ее ведома.

На рис. 17.1 показана небольшая часть иерархии доменных имен глобальной сети Internet.

Рис. 17.1. Небольшая часть дерева иерархии имен доменов глобальной сети Internet. В действительности дерево гораздо разветвленнее. Обычно имена узлов сети располагаются на четвертом или пятом уровнях

Как видно из рисунка, фирма IBM зарегистрировала свой домен второго уровня ibm.com в домене .com, относящемся к коммерческим организациям. Университет Пердью зарегистрировал домен purdue.edu, а Национальный научный фонд США (National Science Foundation) — домен nsf.gov, поскольку он является государственной организацией. Корпорация национальных исследовательских инициатив (Corporation for National Research Initiatives, или CNRI), напротив, выбрала регистрацию в территориальном домене cnri.reston.va.us2.

1. В стандарте нет определения термину *поддомен (subdomain)*. Однако здесь мы использовали его по аналогии с термином *подмножество (subset)*, чтобы помочь читателю прояснить существующие взаимосвязи между доменами.

2. Интересно, что CNRI также зарегистрировала доменное имя nri.reston.va.us.

**Именованные элементы и синтаксис имен**

Система доменных имен является универсальной, поскольку допускает вложенность нескольких иерархий присваивания имен в одной системе. Чтобы клиенты могли легко ориентироваться во множестве элементов, каждому поименованному элементу, хранящемуся в системе, присваивается *тип*. Он определяет, чем является данный элемент: адресом машины, именем почтового ящика, пользователя и т.д. Посылая серверу доменных имен запрос на преобразование имени в адрес, клиент должен определить тип предполагаемого ответа. Например, когда программа работы с электронной почтой использует систему доменных имен для преобразования строковых адресов электронной почты в IP-адреса машин, она должна указать в запросе, что в ответе ожидается адрес *почтового сервера*, выполняющего пересылку сообщений адресату (*mail exchanger*). Программа для регистрации на удаленной машине указывает в запросе, что ее интересует IP-адрес этой машины. Важно понять следующее.

Имя может соответствовать нескольким элементам системы, доменных имен. Отправляя запрос, клиент должен указать предполагаемый тип объекта, имя которого нужно преобразовать, а сервер возвращает объекты, указанного типа.

Кроме определения типа возвращаемого объекта, доменная система имен позволяет клиенту определить семейство протоколов, которое будет использоваться. В доменной системе полное пространство имен разделено на *классы*. Это позволяет хранить в одной базе данных информацию о привязках имен к сетевым адресам, относящимся к разным семействам протоколов.

Проанализировав синтаксис имени, нельзя определить, к какому типу объекта или классу семейства протоколов оно относится. В частности, количество меток в имени не определяет, относится ли оно к индивидуальному объекту (машине) или домену. Например, имя gwen.purdue.edu определяет физический компьютер в Internet. И это при том, что имя cs.purdue.edu относится к поддомену. Можно подытожить этот важный момент следующим образом.

Невозможно отличить имена поддоменов от имен индивидуальных объектов или их типов, проанализировав только синтаксис доменных имен.

На практике такая возможность используется крайне редко.

**Преобразование доменных имен в адреса**

Кроме синтаксиса имен и правил делегирования полномочий, в систему доменных имен включен эффективный, надежный, распределенный и универсальный механизм, выполняющий преобразование имен в адреса. Его реализация является распределенной в техническом смысле этого слова. Это означает, что для решения проблемы преобразования имен в адреса во всех сетевых центрах запускают несколько серверов, которые тесно взаимодействуют друг с другом. Система эффективна в том смысле, что большинство имен может быть преобразовано локально. Поэтому общий трафик службы доменных имен в Internet небольшой. Она может считаться универсальной системой, поскольку ее элементы не обязательно должны соответствовать именам машин (хотя ниже будет использоваться именно этот пример). И наконец, система доменных имен надежна в том смысле, что она будет корректно функционировать, даже если произойдет сбой в работе одного или нескольких удаленных серверов.

Механизм преобразования доменных имен состоит из ряда независимых, но взаимодействующих между собой компьютерных систем, называемых *серверами имен (name servers)*. Сервер имен — это программа, запущенная на компьютере, выполняющем роль сервера, которая преобразует имена доменов в IP-адреса. Очень часто эта программа запускается на специализированном компьютере. В этом случае сама машина также называется сервером имен. Клиентская часть программы называется *распознавателем имен (name resolver)*. В процессе преобразования имени она обращается к одному или нескольким серверам имен.

Для того чтобы понять, как работают серверы доменных имен, нужно расставить их в виде древовидной структуры, которая соответствует иерархии присвоения имен, показанной на рис. 17.2.

Рис. 17.2. Принципиальная древовидная схема взаимодействия серверов доменных имен, соответствующая иерархии присвоения имен. Теоретически каждый сервер в пределах домена, которым он управляет, “знает” адреса всех серверов низшего уровня, управляющих поддоменами

Корень дерева соответствует серверу, который распознает домены верхнего уровня и “знает”, какой сервер распознает каждый из доменов. Корневой сервер управляет процессом преобразования каждого имени и назначает для этого соответствующий сервер. На следующем уровне находятся серверы имен, каждый из которых отсылает ответ на запрос, относящийся к домену верхнего уровня (например, . edu). Сервер на этом уровне “знает”, какой сервер может распознать каждый из поддоменов в его домене. На третьем уровне дерева находятся серверы имен, которые отвечают на запросы, относящиеся к поддоменам (например, purdue в домене.edu). Можно сказать, что дерево домена начинается на самом верхнем уровне и продолжается вниз по уровням, на которых находятся серверы имен, относящиеся ко всем поддоменам этого домена.

Связи на этой принципиальной древовидной схеме не соответствуют физическим связям в сети. Они просто показывают серверы имен, которые известны данному серверу, и с какими серверами он напрямую взаимодействует. Серверы сами по себе могут располагаться в произвольных местах объединенной сети. Таким образом, дерево серверов имен — это абстракция, в которой для взаимодействия серверов между собой используется объединенная сеть.

Если бы серверы в доменной системе имен работали так, как в описанной только что упрощенной модели, то организовать взаимодействие серверов, принадлежащих разным административным единицам, было бы очень легко. В самом деле, если организации переданы права на управление некоторым поддоменом, то она должна запустить у себя сервер доменных имен и связать его с описанным выше деревом.

На практике же взаимосвязь между иерархией присваивания имен и деревом серверов не столь проста, как в описанной модели. Обычно дерево серверов имеет всего несколько уровней, поскольку на одном физическом сервере можно разместить информацию для больших частей иерархии пространства имен. В частности, организации часто выделяют один физический сервер имен для обслуживания всех своих поддоменов. На рис. 17.3 показана реальная схема взаимодействия серверов для иерархии присваивания имен, показанной на рис. 17.1.

Рис. 17.3. Реальная схема взаимодействия серверов для иерархии присвоения имен, показанной на рис. 17.1. Поскольку дерево широкое и неглубокое, то для преобразования практически любого доменного имени необходимо взаимодействие небольшого количество серверов имен

На корневом сервере хранится информация, необходимая для поддержки системы доменных имен верхнего уровня. Каждая организация использует один сервер для управления своим пространством имен. Поскольку дерево серверов неглубокое, для распознавания имен типа xinu.cs.purdue.edu нужно, чтобы два сервера (корневой и сервер домена purdue.edu) связались между собой. Другими словами корневой сервер “знает”, какой из серверов имен управляет доменом purdue.edu, а вся информация об этом домене находится на одном сервере.

**Распознавание доменных имен**

Хотя принципиальная древовидная схема облегчает понимание существующих взаимосвязей между серверами доменных имен, на ней не показано несколько тонких деталей.

Для пояснения рассмотрим алгоритм распознавания доменного имени. По идее распознавание должно происходить сверху вниз — от корневого сервера вниз к серверам, расположенным на листьях дерева. Клиент может использовать систему доменных имен двумя способами.

Во-первых, он может выполнять преобразование в процессе диалога с сервером, обращаясь к нему каждый раз для преобразования очередного уровня доменного имени.

Во-вторых, он может запросить сервер преобразовать имя целиком. В любом случае клиентская программа должна сформировать запрос, поместив в него имя, которое должно быть преобразовано, информацию о его классе, желаемый тип ответа и код, указывающий, должен ли сервер имен преобразовать имя полностью. После этого клиентская программа отправляет запрос серверу имен.

Получив запрос, сервер доменных имен проверяет, не относится ли указанное клиентом имя к поддомену, на который ему переданы полномочия по управлению. Если это так, то он с помощью своей базы данных преобразует имя в адрес, добавляет эту информацию в пользовательский пакет запроса и возвращает его клиенту. Если сервер не может распознать имя полностью, он проверяет, какой тип взаимодействия определен клиентом.

Если клиент затребовал полное преобразование имени (в терминологии доменных имен такое преобразование называют *рекурсивным*), сервер обращается с запросом к тому серверу доменных имен, который может распознать имя, и возвращает ответ клиенту.

Если клиент затребовал нерекурсивное (*итеративное*) преобразование, сервер имен не может отправить информацию клиенту. В этом случае он генерирует ответ, где указывает адрес другого сервера имен, с которым клиент должен связаться для распознания имени.

Как же клиент находит сервер имен, которому нужно послать первый запрос? А как один сервер имен находит адрес другого сервера имен, который может ответить на запрос клиента? Ответ прост. Клиент должен знать, как связаться по крайней мере с одним сервером имен. А для того чтобы сервер доменных имен мог связаться с другими серверами системы, каждый сервер должен знать адрес по крайней мере одного корневого сервера1. Кроме того, серверу имен может быть известен адрес другого сервера, находящегося выше его на один уровень иерархии (он называется *родительским*).

Для взаимодействия между собой серверы доменных имен используют стандартный номер порта протокола. Поэтому клиенты всегда могут связаться с сервером доменных имен при условии, что им известен только IP-адрес машины, на которой этот сервер запущен. Что касается способов поиска IP-адреса такой машины в локальном сетевом окружении, то стандартного решения пока не существует, — все зависит от разработчиков клиентского программного обеспечения.

В некоторых системах IP-адрес машины, на которой запущена служба доменных имен, встраивается в прикладные программы еще на этапе компиляции. В других системах он определяется при начальной загрузке компьютера. В-третьих — администратор прописывает вручную этот адрес и сохраняет его в файле на вспомогательном запоминающем устройстве.

1. Чтобы повысить надежность работы системы, для каждого узла в дереве доменов создается несколько серверов имен. Кроме того, для снижения нагрузки на корневой сервер имен он тиражируется на несколько физических машин.

**Эффективное преобразование имен**

Несмотря на то, что процесс распознавания имен, выполняемый сверху вниз по дереву серверов имен, кажется вполне естественным и понятным, он неэффективен по трем причинам.

Во-первых, в большинстве случаев клиенты присылают запросы на преобразование локальных имен, т.е. имен, относящихся к той же части пространства имен, что и машина, от которой приходит запрос. Поэтому прохождение пути по всей иерархии серверов сверху вниз до осуществления контакта с сервером имен, которому переданы полномочия по управлению локальным доменом, неэффективно.

Во-вторых, если бы каждый процесс преобразования доменного имени начинался с обращения к серверу имен, находящемуся на самом верхнем уровне иерархии, то это привело бы к его быстрой перегрузке.

В-третьих, отказ в работе сервера, находящегося на верхних уровнях иерархии не позволил бы распознать имя, несмотря на то, что локальный сервер имен мог бы сделать это без особых проблем.

Сказанное легко пояснить на уже использовавшемся примере иерархии телефонных номеров. Несмотря на то, что телефонные номера назначаются в соответствии с принятой иерархической структурой, их распознавание происходит снизу вверх. Поскольку большинство телефонных звонков местные, они могут быть распознаны и обработаны местной АТС без установки связи с другими АТС. Кроме того, звонки, сделанные в пределах одного региона (т.е. внутри области с одним междугородным телефонным кодом) могут быть обработаны без привлечения телефонных станций, находящихся в зоне действия другого междугородного телефонного кода. Применив описанный алгоритм к системе доменных имен, получим двухэтапный механизм распознавания имен, при использовании которого сохраняется административная иерархия и эффективно выполняется преобразование доменных имен.

Выше уже было сказано, что на сервер имен часто приходят запросы на преобразование локальных имен. Процесс распознавания имен при использовании двухэтапного алгоритма начинается на локальном сервере имен. Если же локальный сервер не может распознать имя, то запрос перенаправляется другому серверу системы доменных имен.

**Кэширование — залог эффективности**

Стоимость преобразования нелокальных имен может быть чрезвычайно высока, если клиентские программы (распознаватели) будут каждый раз посылать запрос на корневой сервер. Даже если бы запросы сразу поступали на нужный сервер, имеющий полномочия по управлению данным доменом, процесс преобразования имен представлял бы большую нагрузку на объединенную сеть. Таким образом, чтобы повысить общую производительность системы доменных имен, необходимо понизить стоимость преобразования нелокальных имен.

Для оптимизации процесса поиска имен серверы глобальной сети Internet используют механизм *кэширования имен*. То есть каждый сервер сохраняет в локальной кэш-памяти информацию о недавно использовавшихся именах, а также сведения о том, какой из серверов предоставил данные для преобразования этого имени. Получив от клиента запрос на распознание имени, сервер, согласно стандартной процедуре, сначала проверяет, имеет ли он полномочия на управление доменом, к которому принадлежит то имя. Если нет, то сервер выполняет поиск данного имени в своей кэш-памяти. В случае нахождения адресной привязки сервер имен извлекает ее из кэш-памяти и отправляет клиенту. При этом сервер уведомляет клиента, что привязка получена не из достоверного источника (*nonauthoritative*) и передает вместе с ней доменное имя сервера *S*, от которого он получил эту привязку. Локальный сервер посылает также клиенту дополнительную информацию, в которой указана привязка между именем сервера *S* и его IP-адресом. В результате клиенты получают ответы быстро, однако содержащаяся в них информация может быть устаревшей. Поэтому, если во главу угла ставится эффективность, клиент воспользуется ответом, полученным из недостоверного источника, и продолжит работу. Если же система должна работать без ошибок, клиент обязан направить запрос официальному серверу домена S и проверить достоверность полученной привязки между именем и адресом.

Кэширование хорошо работает в системе доменных имен, потому что привязки имен к адресам редко изменяются. Тем не менее рано или поздно это все-таки происходит. Поэтому если сервер поместит в кэш адресную привязку в тот момент, когда она была в первый раз затребована, и больше никогда не будет ее изменять, то содержимое кэш-памяти со временем устареет и может стать неправильным. Для поддержания в кэш-памяти актуальной информации сервер имен контролирует время нахождения каждой привязки в кэше и удаляет из него элементы, время пребывания которых превысило установленное значение. Получив запрос после того, как соответствующий элемент удален из кэш-памяти, сервер должен обратиться к достоверному источнику и заново получить привязку.

Важно то, что серверы имен не используют единое фиксированное время хранения для всех элементов кэш-памяти. Время хранения привязки в кэше определяется достоверным источником, от которого эта привязка получена. Всякий раз, когда официальный сервер имен отвечает на запрос, он помещает в отклик значение *времени жизни (Time То Live*, или *TTL)*. Это значение определяет, на какой срок привязка останется без изменений. Следовательно, для уменьшения нагрузки на сеть администратор официального сервера имен должен указать большие значения времени жизни для тех доменов, которые, как предполагается, не будут часто изменяться. Если же предполагается, что содержимое домена будет изменяться довольно часто, значение времени жизни должно быть достаточно малым, чтобы обеспечить достоверность информации.

Кэширование может осуществляться как на узлах сети, так и на локальных серверах доменных имен. Во многих системах, работающих в режиме разделения времени, используется сложная форма кода программы распознавателя, благодаря которой удается обеспечить высокую эффективность работы системы (иногда даже более высокую, чем на сервере имен). Работа таких систем основана на том, что при запуске программа распознавателя имен загружает с локального сервера доменных имен полную базу данных имен и адресов и поддерживает собственный кэш недавно использованных имен. Программа обращается к серверу только в том случае, когда не может найти имена в своей базе данных. Конечно, узел сети, на котором содержится копия базы данных локального сервера имен, должен периодически сверяться с сервером и получать от него новые привязки. Кроме того, по истечении времени жизни программа распознавателя должна удалить соответствующие привязки из своего кэша. Поскольку имена доменов изменяются нечасто, у узла сети, как правило, не возникает серьезных проблем с поддержанием согласованности этих баз данных.

Хранение копии базы данных локального сервера имен на каждом узле сети имеет несколько преимуществ. Очевидно, что благодаря этому существенно ускоряется процесс распознавания имен на локальных узлах сети, поскольку программа распознавателя может работать автономно, не обращаясь к сети. Кроме того, обеспечивается защита узла сети, если локальный сервер имен выйдет из строя. И наконец, уменьшается вычислительная нагрузка на сервер имен, что позволяет ему обслуживать большее количество машин.

**Формат сообщений системы доменных имен**

Подробное рассмотрение структуры сообщений, которыми обмениваются клиенты и серверы системы доменных имен, поможет объяснить принцип работы системы с точки зрения типичной прикладной программы. Предположим, пользователь запускает прикладную программу и задает имя машины, с которой эта программа должна взаимодействовать. Прежде чем прикладная программа сможет использовать протоколы типа TCP или UDP для обмена данными с указанной машиной, она должна определить IP-адрес этой машины. Для этого программа передает доменное имя машины локальному распознавателю и запрашивает у него ее IP-адрес. Локальный распознаватель проверяет свой кэш и возвращает ответ, если это имя в нем присутствует. Если локальный распознаватель не смог найти имя в кэше, он формирует сообщение системы доменных имен и посылает его серверу (т.е. теперь он становится клиентом). Хотя в нашем примере было использовано только одно имя, формат сообщения позволяет клиенту делать несколько запросов в одном сообщении. Каждый запрос состоит из доменного имени, для которого клиент ищет IP-адрес, спецификации класса запроса (т.е., указывается, где нужно выполнять поиск, например в *объединенной сети*) и типа нужного объекта (например, *адреса*). В качестве ответа сервер возвращает сообщение, сформированное клиентом, в которое помещает имеющуюся у него информацию о привязках. Если же сервер не может ответить на все запросы, то в ответе будет указано, с какими серверами имен клиент должен связаться, чтобы получить ответы.

В ответ сервера имен включается также информация об официальных серверах имен для данного домена и их IP-адреса. Формат сообщения системы доменных имен показан на рис. 17.4.

Рис 17.4. Формат сообщения системы доменных имен. Разделы запросов, ответов, официальных серверов и дополнительной информации имеют переменную длину

Каждое сообщение начинается с заголовка фиксированного формата. В заголовке предусмотрено специальное поле *идентификации*, куда клиент помещает уникальное значение для сопоставления запросов и ответов сервера. В поле *параметров* помещается тип запрашиваемого действия и код ответа. Значение битов этого поля приведено в табл. 17.2. Биты перечислены слева направо, начиная с нулевого.

Таблица 17.2. Значение битов поля параметров в сообщении системы доменных имен

| **Номер бита** | **Описание** |
| --- | --- |
| 0 | Тип операции: |
|  | 0 — запрос |
|  | 1 — ответ |
| 1-4 | Тип запроса: |
|  | 0 — стандартный |
|  | 1 — инверсный |
|  | 2 — окончание 1 (устарел) |
|  | 3 — окончание 2 (устарел) |
| 5 | Устанавливается, если ответ получен из ненадежного источника |
| 6 | Устанавливается, если сообщение усечено |
| 7 | Устанавливается, если желательна рекурсивная обработка |
| 8 | Устанавливается, если выполняется рекурсивная обработка |
| 9-11 | Зарезервированы |
| 12-15 | Тип ответа: |
|  | 0 — операция выполнена без ошибок |
|  | 1 — неверный формат запроса |
|  | 2 — сбой в работе сервера |
|  | 3 — имя не существует |

В четырех последующих полях указывается количество элементов в соответствующих разделах сообщения. Например, в поле *Количество запросов* указывается количество элементов, которые находятся в разделе запросов сообщения.

Клиент заполняет только раздел запросов, помещая сюда список всех своих вопросов к серверу. Сервер возвращает запросы клиента и ответы на них в одном сообщении. Каждый запрос состоит из *доменного имени*, за которым указываются *тип* запроса и его *класс*, как показано на рис. 17.5.

Рис. 17.5. Формат элемента в разделе запросов сообщения системы доменных имен. Доменное имя имеет переменную длину

Хотя поле *запрашиваемого доменного имени* имеет переменную длину, в следующем разделе мы увидим, как формат внутреннего представления доменных имен позволяет получателю узнать точную длину имени. Тип запроса (например, относится ли запрос к имени машины или к почтовому адресу) указывается в одноименном поле элемента в разделе запросов. Поле *класса запроса* позволяет использовать доменную систему имен для работы с объектами произвольного типа. Все дело в том, что при работе в глобальной сети Internet разрешен только один класс запросов, относящийся к официальным именам. Следует заметить, что, хотя на рис. 24.5 показан формат элемента запросов кратный 32-битам, поле *запрашиваемого доменного имени* может содержать произвольное количество октетов. Причем байты выравнивания на границу 32 битов не используются. Поэтому сообщения, посылаемые на серверы доменных имен или в обратном направлении, могут содержать нечетное количество октетов.

В сообщении системы доменных имен каждый из оставшихся разделов (ответов, официальных серверов и дополнительной информации) состоит из набора *записей ресурса*, которые описывают доменные имена и их соответствие адресам. В каждой записи ресурса описывается только одно имя. Формат записи показан на рис. 17.6.

Рис. 17.6. Формат записи ресурса, используемой в трех последних разделах сообщения системы доменных имен, возвращаемых серверами имен

В поле *доменного имени ресурса* указывается имя домена, к которому относится запись ресурса. Имя может иметь произвольную длину. В поле *типа* кодируется тип данных, включенных в запись ресурса, а поле *класса* определяет класс этих данных. В поле *времени жизни* сервер помещает целое 32-битовое число, определяющее время в секундах, в течении которого информация об этой записи может сохраняться в локальной кэш-памяти. Значение этого поля используется клиентами, которые запросили привязку имени и, возможно, собираются поместить ее в кэш. Последние два поля содержат результаты поиска привязки: в поле *длины данных ресурса* указывается количество октетов, находящихся в поле *данных ресурса*.

**Сжатый формат представления имен**

Доменные имена представляются в сообщении в виде последовательности меток. Каждая метка начинается с октета, который определяет ее длину. Следовательно, для восстановления доменного имени получатель должен поочередно извлечь длину метки, *n*, из октета длины, а затем — *n* октетов самой метки, расположенной следом. Конец доменного имени обозначается октетом длины, содержащим нулевое значение.

Серверы доменных имен часто возвращают несколько ответов на запрос и во многих случаях в них указываются одинаковые суффиксы домена. Для того чтобы сэкономить место в ответном пакете, серверы сжимают имена, сохраняя только одну копию каждого доменного имени. При извлечении доменных имен из сообщения клиентская программа должна проверить каждый сегмент имени на предмет того, содержит ли оно строку символов (в формате: один октет счетчика длины, за которым следуют символы, составляющие имя) или указатель на строку символов. Если обнаружен указатель, клиентская программа должна перейти на новое место в сообщении и найти остаток имени.

Указатели всегда размещаются в начале сегментов и кодируются в байте счетчика. Если два старших бита 8-битового октета счетчика равны единице, то клиент должен использовать следующие 14 битов как целочисленный указатель. Если же два старших бита равны нулю, то следующие 6 битов определяют количество символов в метке, которая следует за октетом счетчика.

**Сокращение доменных имен**

На примере иерархии присвоения телефонных номеров можно увидеть другую полезную особенность локального распознавания имен — возможность *сокращения имен*. Сокращение позволяет укорачивать имена, если в процессе распознавания часть имени может быть получена автоматически.

Звоня по местному телефонному номеру, абонент не набирает междугородный телефонный код. Предполагается, что цифры телефонного номера, вводимые абонентом и составляющие сокращенный номер, относятся к тому же междугородному коду, что и номер телефона абонента.

Описанный принцип сокращения также хорошо работает и в случае имен машин. В процессе распознавания имени типа xyz можно предположить, что это имя и машина, на которой оно распознается, относятся к одному домену. Таким образом, распознаватель может автоматически восполнять отсутствующие части имени. Например, в пределах факультета информатики университета Пердью, сокращенное имя xinu будет эквивалентно полному доменному имени xinu.cs.purdue.edu.

В большинстве клиентских программ поддерживается возможность использования *списка суффиксов домена*, на основании которого реализуется процесс сокращения имен. При формировании этого списка администратор локальной сети указывает в нем перечень возможных суффиксов домена, которые автоматически будут добавляться к концу сокращенного имени при выполнении поиска. Получив имя, распознаватель извлекает элементы этого списка и добавляет их по одному к имени, каждый раз проверяя полученное имя. Например, список суффиксов домена факультета информатики университета Пердью выглядит так:

.cs.purdue.edu

.cc.purdue.edu

.purdue.edu

*нуль*

Таким образом, при поиске имени xinu локальный распознаватель сначала добавит к нему суффикс cs.purdue.edu. Если полученное в результате имя не будет найдено, то распознаватель добавит к имени xinu другой суффикс — cc.purdue.edu — и попытается распознать это имя и т.д. Последним элементом списка в нашем примере указана пустая строка. Она означает, что, если предыдущие операции поиска оказались неудачными, распознаватель попытается распознать имя без добавления суффикса. Список суффиксов домена может использоваться сетевым администратором для того, чтобы предоставить пользователям удобные и легко запоминающиеся короткие имена, а также для возможности использования локальных имен в прикладных программах.

Выше уже отмечалось, что при использовании сокращений их расширением занимается клиентская программа. При этом стоит подчеркнуть, что сами по себе сокращения не являются частью системы доменных имен. Доменная система имен позволяет искать только полностью определенные доменные имена. Вследствие этого программы, использующие сокращенные имена, могут работать неправильно за пределами того окружения, для которого они были созданы. Можно подытожить все сказанное выше так.

Система доменных имен позволяет преобразовывать в адреса только полностью определенные доменные имена. Сокращения сами по себе не являются частью системы доменных имен, а используются в клиентских программах для удобства работы пользователей, которые привыкли к коротким локальным именам машин.

**Инверсные преобразования**

Как было сказано выше, система доменных имен позволяет преобразовывать не только имена машин в IP-адреса, но и выполнять любые другие преобразования. В данном разделе речь пойдет о так называемых *инверсных преобразованиях*, которые позволяют клиентской программе запросить сервер выполнить преобразование имени в обратном направлении. То есть, на основании готового ответа сервера восстановить запрос клиента, при обработке которого был бы получен такой же ответ. Конечно, такое преобразование не всегда дает однозначный результат, т.е. одинаковый ответ сервера может быть получен при обработке нескольких разных запросов. Поэтому при выполнении инверсных преобразований сервер может построить неправильный запрос. Несмотря на то, что механизм инверсных запросов с самого начала входил в систему доменных имен, он используется довольно редко. Причина состоит в том, что зачастую сервер не может обработать такой тип запроса без выполнения поиска по целому ряду серверов.

**Запросы указателя**

Несмотря на изложенные в предыдущем разделе недостатки инверсного преобразования, очевидно, что одна из его форм имеет право на существование, более того, просто необходима. Важность этой формы столь велика, что в системе доменных имен для ее реализации предусмотрен специальный домен и специальная форма запроса, называемая *запросом указателя (pointer query)*. При использовании запроса указателя клиентская программа формирует обычное сообщение системы доменных имен, только в раздел запросов помещает не доменное имя, а IP-адрес машины, цифры которого представлены в виде текстовой строки и разделены точками. На запрос указателя сервер имен должен возвратить доменное имя машины, соответствующей указанному IP-адресу. Запросы указателя особенно полезны для бездисковых машин, поскольку с их помощью компьютер по IP-адресу может определить свое высокоуровневое имя.

Запросы указателя нетрудно сгенерировать. Если IP-адрес записать в десятичной форме с точками в качестве разделителей, он будет иметь следующий вид:

ааа.bbb.ссс.ddd

Чтобы сформировать запрос указателя, клиент перестраивает десятичное представление адреса в строку формы:

ddd.ссс.bbb.ааа.in-addr.аrра

Новая форма адреса представляет собой имя, относящееся к специальному домену in-addr.аrра1. Поскольку локальный сервер имен может не быть официальным сервером для домена аrра, либо для домена in-addr.аrра, для обработки запроса ему может понадобиться связаться с другими серверами имен.

Чтобы повысить эффективность обработки запросов указателя, корневые серверы системы доменных имен глобальной сети Internet поддерживают базу данных распределенных IP-адресов, а также информацию о серверах имен, которые могут распознать каждый адрес.

1. При формировании доменного имени порядок октетов IP-адреса должен быть изменен на обратный. Дело в том, что в IP-адресах наиболее значимые октеты (старшие октеты адреса) находятся в начале, в то время как в доменных именах сначала указывается наименее значимая информация.

**Типы объектов и содержимое записи ресурса**

Как уже было сказано, система доменных имен может использоваться для преобразования имен машин в их IP-адреса, а также для преобразования адресов электронной почты в имена почтовых серверов, выполняющих обработку и пересылку электронной почты. Доменная система имен в целом является универсальной, поэтому ее можно применить для любых иерархий имен. Например, может понадобиться организовать справочную службу вычислительного центра, позволяющую по названию предоставляемой услуги определить номер телефона, по которому клиент может позвонить и получить дополнительную информацию о ней. Или, например, можно создать справочную службу производителей программного обеспечения, которая бы по типу используемого протокола находила названия фирм производителей и их адреса.

Напомним, что подобные преобразования возможны за счет включения поля *типа* в каждую запись ресурса. При отсылке запроса, клиент должен указать его тип1. Серверы имен указывают тип данных во всех возвращаемых записях ресурса. Тип определяет содержимое записи ресурса, как показано в табл. 17.3.

Таблица 17.3. Типы записей ресурса системы доменных имен

| **Тип** | **Значение** | **Содержимое** |
| --- | --- | --- |
| А | Host Address (адрес узла сети) | 32-битовый IP-адрес |
| СNAME | Canonical Name (каноническое имя) | Каноническое имя домена для псевдонима |
| HINFO | Host Info (информация об узле сети) | Название фирмы — производителя центрального процессора компьютера и типа операционной системы |
| MINFO | Mailbox Info (информация о почтовом ящике) | Информация о почтовом ящике или списке почтовой рассылки |
| MX | Mail Exchanger (сервер пересылки почты) | 16-битовое значение приоритета и имя узла сети, который принимает и пересылает почту для указанного домена |
| NS | Name Server (сервер имен) | Имя официального сервера для указанного домена |
| PTR | Указатель | Имя домена (тип символьной связи) |
| SOA | Start of Authority (начало полномочия) | Несколько полей, которые определяют, за какую часть иерархии назначения имен отвечает сервер |
| TXT | Text (произвольный текст) | Неинтерпретируемая ASCII-строка текста |

Большинство записей в базе данных имен относится к типу А. Они состоят из имени и соответствующего ему IP-адреса узла сети, подключенного к глобальной сети Internet.

Второй часто используемый тип ресурса, MX, предназначен для указания серверов, принимающих и обрабатывающих электронную почту для этого домена. С его помощью сетевой администратор может указать несколько почтовых серверов, обрабатывающих электронную почту для данного домена. При отправке сообщения по электронной почте, пользователь указывает адрес получателя в виде имя-пользователя@домен. Почтовая система использует систему доменных имен для преобразования доменной части адреса электронной почты получателя в адрес почтового сервера, указанного в записи типа MX. Доменная система возвращает набор записей ресурса, в каждой из которых содержится значение приоритета и доменное имя почтового сервера. При отправке сообщения почтовая система пытается сначала передать его серверу, которому назначен наивысший приоритет (меньшие числа соответствуют более высокому приоритету). Если передача не удалась, из списка выбирается сервер с меньшим приоритетом, и процесс отправки сообщения повторяется, и т.д. При этом программа доставки электронной почты извлекает из МХ-записи ресурса доменное имя очередного почтового сервера и использует запрос типа А, чтобы преобразовать его в IP-адрес. После этого она пытается связаться с узлом сети по указанному адресу и доставить почту. Если этот узел сети недоступен, программа доставки электронной почты попытается связаться с другим узлом, указанным в списке.

Чтобы сделать поиск эффективным, сервер всегда возвращает дополнительные привязки, помещая их в раздел *дополнительной информации* сообщения системы доменных имен. В случае записей типа MX сервер имен может поместить в раздел *дополнительной информации* данные, полученные из записи ресурса типа А, соответствующие доменным именам, указанным в разделе *ответа*. В результате программа доставки электронной почты получит в одном сообщении всю нужную ей информацию, что существенно уменьшит количество запросов, которые она должна отправить своему серверу имен.

1. В запросах можно указать несколько дополнительных типов (например, существует специальный тип запроса, который позволяет выбрать все записи ресурса).

**Получение полномочий для управления поддоменом**

Прежде чем получить официальные полномочия на управление доменом второго уровня, организация должна запустить у себя сервер доменных имен, соответствующий стандартам глобальной сети Internet. Естественно, что сервер имен должен удовлетворять стандартам протокола, которые определяют форматы сообщений и правила ответов на запросы. Сервер должен также располагать адресами других серверов имен, которые отвечают за работу каждого поддомена (если таковые существуют), а также адресом по крайней мере одного корневого сервера имен.

На практике структура доменной системы имен гораздо сложнее, чем описанная в этом разделе. В большинстве случаев один физический сервер может обслуживать несколько иерархий назначения имен. Например, один сервер имен, запущенный в университете Пердью, обслуживает как домен второго уровня purdue.edu, так и географический домен laf.in.us. Таким образом, поддерево имен, обслуживаемых сервером, составляет его *зону полномочий*.

Еще одно затруднение возникает из-за того, что серверы должны одновременно обрабатывать множество запросов, хотя для завершения некоторых из них требуется продолжительное время. Обычно серверы обрабатывают поступившие запросы параллельно. Это позволяет им переходить к обработке вновь поступивших запросов, в то время как запросы, поступившие ранее, еще обрабатываются. Параллельная обработка запросов особенно важна, когда сервер получает рекурсивный запрос, для завершения которого он вынужден обратиться за дополнительной информацией к другому серверу.

Реализация сервера имен осложняется еще и тем, что центральный административный орган глобальной сети Internet требует, чтобы информация на каждом сервере доменных имен дублировалась и автоматически реплицировалась на резервных серверах. Причем количество этих серверов должно быть не меньше двух, и работать они должны на разных физических машинах. Практические требования еще более строгие: выход из строя одного сервера не должен повлиять на работу другого сервера, т.е. у резервных серверов не должно быть общих предпосылок для сбоя. Устранение общих предпосылок для сбоя означает, что резервные серверы имен не могут быть подключены к одной и той же сети и получать электрическое питание из одного источника. Чтобы удовлетворить эти требования, администрация сетевого центра должна договориться с администрацией по крайней мере еще одного сетевого центра и расположить резервный сервер имен в их центре. Естественно, сервер, расположенный в любой точке дерева, должен “уметь” находить как основной, так и резервные серверы имен для данного поддомена. Если основной сервер по какой-либо причине оказывается недоступным, все запросы перенаправляются одному из резервных серверов имен.

**Проверка знаний: Система доменных имен (DNS)**

**Задание 1**

Как называется механизм преобразования доменных имен, который состоит из ряда независимых, но взаимодействующих между собой компьютерных систем?

кодировщик имен

распознаватель имен

серверы имен

**Задание 2**

Завершите утверждение корректно

Система адресации из универсальных адресов преобразует адреса сетевых адаптеров в

физические адреса.

**Задание 3**

Какие доменные имена позволяет преобразовывать в адреса система доменных имен?

имена которые еще не определены

только полностью определенные

частично определенные

**Задание 4**

Что требует сделать с информацией на каждом сервере доменных имен центральный административный орган глобальной сети Internet?

дублировать

удалять

создавать отчет

**Задание 5**

Для каких машин особо полезны запросы указателя?

бездисковых

с несколькими сетевыми адаптерами

суперкомпьютеров

**Задание 6**

Что должна определить прикладная программа, прежде чем она сможет использовать протоколы типа TCP или UDP, для обмена данными с указанной машиной?

IP-адрес этой машины

физический адрес этой машины

сетевой адрес

**Задание 7**

Иерархические имена машин в объединенной сети TCP/IP присваиваются в соответствии со структурой организации, которой переданы полномочия по управлению частью пространства имен. Как связан сам процесс назначения имени со структурой физических сетевых подключений?

согласуется с ней

полностью зависит от нее

никак

**Задание 8**

Как называется механизм, с помощью которого реализована иерархическая система именования машин в объединенной сети TCP/IP?

система сетевых адресов

система доменных имен

таблица физических узлов

**Задание 9**

Что должен указать клиент, отправляя запрос об объекте, имя которого нужно преобразовать?

предполагаемый тип объекта

нет необходимости, что либо указывать

предполагаемый размер объекта

**Безопасность в объединенной сети и брандмауэры (IPsec)**

Как известно, для надежного хранения материального имущества нужны крепкие двери и сложные замки. Точно так же, чтобы предотвратить нежелательную утечку информации, необходимо предпринять ряд мер в работе компьютеров и сетей передачи данных.

Проблема безопасности информации в объединенной сети является важной и в тоже время трудно разрешимой. Важность проблемы заключается в том, что информация на сегодняшний день имеет существенную ценность — она может быть куплена или продана как непосредственно, так и косвенно (т.е. использоваться для создания новых продуктов и услуг, приносящих высокие прибыли).

Обеспечить безопасность в объединенной очень трудно, поскольку для этого требуется не только понимание технических тонкостей работы сетевых аппаратных средств и протоколов, но и знание того, как и когда участвующие в процессе сетевого обмена пользователи, компьютеры, службы и сети могут доверять друг другу.

Средства безопасности должны соблюдаться на каждом компьютере и в каждом протоколе. Одно слабое звено может поставить под угрозу безопасность всей сети. И, что более важно, поскольку семейство протоколов TCP/IP может использоваться широким кругом разных пользователей, а также при создании разнообразных служб и сетей, и, поскольку объединенная сеть может пересекать много как политических, так и организационных границ, то задействованные индивидуумы и организации могут не договориться между собой об уровне доверия или правилах, касающихся обработки данных.

В этом разделе рассматриваются два главных принципа, положенных в основу безопасности объединенной сети: внешняя безопасность и шифрование.

Правила внешней безопасности позволяют организации определить службы и сети, которые могут быть доступны извне, а также допустимые пределы, в которых посторонние лица могут пользоваться ее ресурсами.

Шифрование покрывает большинство других аспектов безопасности.

Раздел мы начнем с рассмотрения основных концепций и терминологии.

* [Безопасность ресурсов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690628)
* [Информационная стратегия](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690629)
* [Безопасность глобальной сети Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062a)
* [Безопасность протокола IP (IPsec)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062b)
* [Заголовок аутентификации семейства протоколов IPsec](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062c)
* [Ассоциация обеспечения безопасности](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062d)
* [Инкапсуляция зашифрованных данных в протоколе IPsec](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062e)
* [Аутентификация и изменяемые поля заголовка](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69062f)
* [Туннелирование по закрытому каналу на основе протоколов IPsec](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690630)
* [Обязательные алгоритмы шифрования](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690631)
* [Механизм защищенных сокетов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690632)
* [Брандмауэры и доступ к сети Internet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690633)
* [Множественные соединения и самые слабые связи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690634)
* [Реализация брандмауэра](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690635)
* [Фильтры пакетов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690636)
* [Безопасность и параметры фильтра пакетов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690637)
* [Влияние ограничения доступа на клиентов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690638)
* [Доступ к Internet в обход брандмауэра через proxy-серверы](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690639)
* [Подробнее о структуре брандмауэра](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69063a)
* [Тупиковая сеть](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69063b)
* [Альтернативный вариант реализации брандмауэра](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69063c)
* [Текущий контроль и регистрация](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69063d)
* [Проверка знаний: Безопасность в объединенной сети и брандмауэры (IPsec)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a69063e)

**Безопасность ресурсов**

Термины *безопасность сети (network security)* и *информационная безопасность (information security)* в широком смысле относятся к секретности, т.е. гарантии того, что информация и службы, имеющиеся в сети, не будут доступны для несанкционированного использования. Безопасность подразумевает механизм защиты, гарантирующий целостность данных, невозможность несанкционированного доступа к вычислительным ресурсам, шпионажа или перехвата сообщений, а также вторжения в работу служб.

Конечно, как нельзя гарантировать абсолютную защищенность материальных ценностей от посягательств, так нельзя гарантировать и полную безопасность сети. Руководство организации должно уделять внимание безопасности своих сетей, как это делается в отношении зданий и офисов: даже несложные меры безопасности могут воспрепятствовать преступлению, значительно усложнив его совершение.

Обеспечение информационной безопасности требует охраны как физических, так и виртуальных ресурсов. К физическим ресурсам можно отнести как пассивные устройства для хранения информации, такие как жесткие диски и компакт-диски, так и активные устройства, такие как компьютеры пользователей. В сетевом окружении понятие физической безопасности относится к кабелям, мостам и маршрутизаторам, а также ко всем устройствам, составляющим инфраструктуру сети. Действительно, хотя о физической безопасности редко упоминается, она часто играет важную роль при планировании полной безопасности. Очевидно, что меры физической безопасности могут предотвратить перехват сообщений путем несанкционированного подключения к сети. Эффективные меры физической безопасности способствуют также предотвращению диверсии (например, умышленному выводу из строя маршрутизатора с целью изменения маршрута следования пакетов и их передачи по альтернативному, незащищенному пути).

Обеспечить безопасность виртуального ресурса, такого как информация, обычно более трудно, чем обеспечить физическую безопасность, поскольку информация не имеет физических границ. Информационная безопасность охватывает много аспектов защиты, основные из которых перечислены ниже.

* *Целостность данных*. Безопасная система должна защищать информацию от несанкционированного изменения или повреждения.
* *Доступность данных*. Система должна гарантировать, что несанкционированный пользователь не сможет помещать законному доступу к данным (например, никто из посторонних не должен иметь возможность блокировать клиентов фирмы от доступа к ее Web-узлу).
* *Секретность*, или *конфиденциальность*. Система не должна позволять несанкционированным пользователям создавать копии данных во время их передачи по сети, а также анализировать их содержимое в том случае, если копии все же сделаны.
* *Авторизация*. Хотя при осуществлении мер физической безопасности часто выполняют классификацию людей и ресурсов по широким категориям (например, лицам не являющимся служащими компании, запрещается доступ в некоторые помещения), меры информационной безопасности должны быть более избирательны (например, часть полей записи базы данных служащих должна быть доступна только для сотрудников отдела кадров, другая — только для начальника, а третья — для бухгалтерии).
* *Аутентификация*. Система должна позволять двум взаимодействующим между собой объектам проверять подлинность друг друга.
* *Запрещение повторного использования*. Чтобы посторонние не могли перехватывать копии пакетов с целью их дальнейшего использования, система не должна обрабатывать копии повторно переданных пакетов.

**Информационная стратегия**

Прежде чем приступить к реализации системы безопасности сети, следует оценить риски и разработать четкую стратегию доступа к информации и ее защиты. В стратегии определяется круг лиц и уровень их доступа к той или иной информации, правила, которых эти лица должны придерживаться в целях неразглашения информации, а также ответственность за нарушение этих правил.

Разработка информационной стратегии во многом зависит от человеческого фактора, потому что

люди представляют собой наиболее уязвимое звено в любой системе безопасности. Служащий компании либо по злому умыслу, либо по неосторожности, либо не зная принятой в компании стратегии, может поставить под угрозу самую совершенную систему безопасности.

**Безопасность глобальной сети Internet**

Обеспечить безопасность в глобальной сети Internet особенно трудно, поскольку дейтаграммы, передающиеся от отправителя до конечного получателя, проходят через несколько промежуточных сетей и маршрутизаторов, которые не принадлежат ни отправителю, ни конечному получателю и не контролируются ими. Таким образом, поскольку дейтаграммы могут быть перехвачены и изменены без ведома отправителя, их содержимому нельзя доверять.

В качестве примера рассмотрим сервер, который использует процедуру *аутентификации источника (source authentication)* для проверки того, что запросы поступают от авторизованных клиентов. Процедура аутентификации источника требует, чтобы сервер при получении каждой дейтаграммы проверял IP-адрес отправителя, и принимал запросы только от компьютеров, адреса которых перечислены в специальном списке. Данный вид аутентификации обеспечивает *слабую* защиту, поскольку ее можно легко обойти. В частности, один из промежуточных маршрутизаторов может контролировать трафик, идущий к нашему серверу и от него, и фиксировать IP-адреса авторизованных клиентов. После этого промежуточный маршрутизатор может воссоздать запрос и указать в нем один из зафиксированных адресов отправителя и перехватить ответ сервера. Суть сказанного можно подытожить так.

Механизм авторизации, в котором для подтверждения подлинности отправителя используется его IP-адрес, не удовлетворяет требованиям безопасности объединенной сети. Любой самозванец, получивший контроль над промежуточным маршрутизатором, может легко обойти систему защиты, и выступить в роли авторизованного клиента.

Для более строгой аутентификации требуется механизм *шифрования*. Отправитель шифрует сообщение с помощью математической функции, которая изменяет значение битов согласно ключу, известному только отправителю. Для расшифровки сообщения получатель должен использовать другую математическую функцию и ключ. Тщательный выбор алгоритма шифрования и ключа могут сделать расшифровку сообщения и его подделку на промежуточных машинах практически невозможными.

**Безопасность протокола IP (IPsec)**

Группа IETF разработала набор протоколов, которые обеспечивают безопасную связь в глобальной сети Internet. Все вместе они называются семейством протоколов *IPsec* (сокращение от *IP security*, или *защищенный протокол IP*).

В этих протоколах аутентификация и шифрование данных выполняются на уровне протокола IP. Причем они могут использоваться и с IPv4, и с IPv61.

Важно то, что вместо полного определения функциональных возможностей или алгоритма шифрования, который нужно использовать, группа IETF сделала систему одновременно и гибкой, и расширяемой. Например, при создании приложений, использующих протоколы IPsec, у программиста есть возможность выбора: применить средство аутентификации, проверяющее достоверность отправителя, или же средство шифрования, которое помимо всего прочего гарантирует конфиденциальность передаваемой информации. Причем возможность выбора является “асимметричной” (например, аутентификация может выполняться в одном направлении и не выполняться в другом).

Кроме того, в стандарте протоколов IPsec явно не оговорен тип шифрования или алгоритмом аутентификации. Вместо этого, в стандарте определена общая структура системы, которая позволяет каждой паре взаимодействующих конечных пунктов выбирать алгоритмы и параметры шифрования (например, размер ключа). Чтобы гарантировать способность к взаимодействию, в семейство протоколов IPsec все-таки включен набор алгоритмов шифрования, которые должны поддерживаться во всех реализациях этих протоколов.

Суть сказанного выше можно подытожить так.

IPsec — это не один протокол, а набор алгоритмов, обеспечивающих безопасную передачу данных и общую структуру, которая позволяет паре взаимодействующих объектов использовать любые алгоритмы, шифрования, гарантирующие соответствующую безопасность соединения.

1. В этом разделе приводятся примеры только для протокола IPv4. Подробное описание протокола IPv6 и примеры заголовков IPsec в дейтаграммах IPv6, приведены в разделе [Будущее протокола TCP/IP (IPv6)](https://it.rfei.ru/course/%7EGZqN/-/%7EvE3L).

**Заголовок аутентификации семейства протоколов IPsec**

Разработчики семейства протоколов IPsec не стали вносить изменения в формат основного заголовка IP-дейтаграммы или создавать дополнительное поле ее параметров. В протоколах IPsec используется отдельный *заголовок аутентификации (Authentication Header*, или *АН*), несущий информацию об аутентификации. На рис. 18.1 показан пример наиболее простого использования заголовка аутентификации с протоколом IPv4. Новый заголовок помещен непосредственно после IP-заголовка.

Рис. 18.1. Структура обычной дейтаграммы IPv4 (а); структура той же дейтаграммы после добавления заголовка аутентификации IPsec (б)

Как показано на рис. 18.1, заголовок аутентификации протокола IPsec помещен непосредственно после заголовка IP-дейтаграммы, но перед заголовком протокола транспортного уровня. Кроме того, значение поля *типа протокола* в заголовке IP-дейтаграммы изменено на величину 51, что указывает на присутствие заголовка аутентификации.

Возникает вопрос: как при наличии заголовка аутентификации протокола IPsec получатель может определить тип данных, передаваемых в дейтаграмме? Ведь значение поля типа протокола IP-дейтаграммы изменено и всегда равно одному и тому же значению — 51. Для решения этой проблемы в заголовке аутентификации предусмотрено поле *типа следующего заголовка*, в котором и указывается тип передаваемых в дейтаграмме данных. Программы протокола IPsec переписывают оригинальное значение поля типа протокола IP-дейтаграммы в поле типа следующего заголовка. При получении дейтаграммы анализируется информация системы безопасности из заголовка аутентификации для проверки отправителя, а затем используется значение поля *типа следующего заголовка* для дальнейшего демультиплексирования дейтаграммы. Формат заголовка аутентификации показан на рис. 18.2.

Рис. 18.2. Формат заголовка аутентификации протокола IPsec. В поле типа следующего заголовка хранится оригинальное значение поля типа протокола IP-дейтаграммы

Интересно, что в поле *длины содержимого* не указывается размер области данных дейтаграммы. Вместо этого в него помещается длина заголовка аутентификации. В остальных полях находится информация, использующаяся для гарантированной безопасной передачи данных. В поле *порядкового номера* содержится уникальный порядковый номер каждого посланного пакета. Этот номер начинается с нуля при выборе определенного алгоритма безопасности и последовательно увеличивается. Значение поля *индекса параметров системы безопасности* определяет используемую схему безопасности, а поле *данныx для аутентификации* содержит данные для выбранной схемы безопасности.

**Ассоциация обеспечения безопасности**

Чтобы понять причину использования индекса параметров системы безопасности, надо отметить, что в принятой схеме безопасности может быть задействована масса деталей, благодаря которым обеспечивается большое разнообразие параметров. Например, принятая схема безопасности включает алгоритм аутентификации, ключ (или ключи), используемые в алгоритме, время жизни, в течение которого ключ остается действительным, время жизни, в течение которого получатель может использовать данный алгоритм, список адресов отправителей, которые уполномочены использовать данную схему. Естественно, вся эта информация не может поместиться в заголовке аутентификации.

Чтобы сэкономить место в заголовке, в протоколе IPsec было введено абстрактное понятие, называемое *ассоциацией обеспечения безопасности (Security Association*, или *SA*), и оговорено, что каждый получатель может получить все сведения относительно используемой схемы безопасности из этой ассоциации.

Каждой ассоциации присваивается номер, называемый *индексом параметров системы, безопасности*, по которому она идентифицируется. Прежде чем отправитель сможет использовать протокол IPsec для связи с получателем, он должен узнать значение индекса для требуемой ассоциации. Затем отправитель помещает это значение в поле индекса параметров системы безопасности каждой исходящей дейтаграммы.

Индексы не являются глобальными. Получатель создает необходимое количество ассоциаций системы безопасности и каждой из них присваивает значение индекса. Получатель может определить время жизни для каждой ассоциации и по его истечении использовать значение индекса для другой ассоциации. Следовательно, индекс системы безопасности нельзя корректно интерпретировать без запроса к получателю (например, индекс 1 может иметь совершенно различные значения для двух получателей). Сказанное выше можно подытожить так.

Получатель использует индекс параметров системы, безопасности, чтобы, идентифицировать ассоциацию безопасности для пакета. Индексы, не являются глобальными. Для идентификации ассоциации необходима комбинация адреса получателя и индекса параметров системы, безопасности.

**Инкапсуляция зашифрованных данных в протоколе IPsec**

Чтобы можно было обрабатывать зашифрованные данные по аналогии с данными для аутентификации, в протоколе IPsec используется механизм *инкапсуляции зашифрованных данных (Encapsulating Security Payload*, или *ESP*), который намного сложнее механизма аутентификации. Если значение поля типа протокола IP-дейтаграммы равно 50, это означает, что в дейтаграмме находятся зашифрованные данные. На рис. 18.3 схематично изображена такая дейтаграмма.

Рис. 18.3. Схематичное изображение дейтаграммы (а), и та же дейтаграмма, в которую инкапсулированы зашифрованные данные протокола IPsec. Использование шифрования означает, что идентифицировать значения полей достаточно сложно (б)

Как показано на рис. 18.3, при использовании зашифрованных данных к дейтаграмме добавляются три дополнительные области. Заголовок ESP следует сразу за заголовком IP-дейтаграммы и предшествует зашифрованной области данных. Трейлер ESP зашифрован вместе с передаваемыми данными. Поле аутентификации ESP имеет переменную длину и следует сразу за зашифрованными данными.

При инкапсуляции зашифрованных данных используются практически те же элементы, которые мы рассматривали в заголовке аутентификации, только порядок их следования другой. Например, заголовок ESP состоит из 8 октетов. В нем указывается индекс параметров системы безопасности и порядковый номер пакета, как показано на рис. 18.4.

Рис. 18.4. Формат заголовка ESP

Трейлер ESP состоит из необязательного поля переменной длины, использующегося для выравнивания общей длины пакета, поля, в котором указана длина поля выравнивания, и поля *типа следующего заголовка*, за которым следует поле переменной длины, содержащее данные для аутентификации ESP (рис. 18.5).

Рис. 18.5. Формат трейлера ESP

Поле для выравнивания не является обязательным. Тем не менее оно может принудительно вводиться в пакет. Для этого имеются три причины.

Во-первых, для работы некоторых алгоритмов дешифрования требуется, чтобы после зашифрованного сообщения следовал ряд нулевых октетов.

Во-вторых, обратите внимание, что поле типа следующего заголовка выровнено по правой границе 4-октетного поля. Это выравнивание очень важно, поскольку в спецификации IPsec требуется, чтобы данные для аутентификации, которые следуют за трейлером, были выровнены на границу 4-октетного поля. Таким образом, поле для выравнивания может использоваться по своему прямому назначению.

В-третьих, в некоторых системах к каждой дейтаграмме может добавляться случайное количество октетов выравнивания, чтобы в случае ее перехвата на одной из промежуточных станций злоумышленники не смогли бы по размеру дейтаграммы догадаться о ее назначении.

**Аутентификация и изменяемые поля заголовка**

По идее механизм аутентификации протокола IPsec создавался для того, чтобы можно было гарантировать, что прибывающая дейтаграмма полностью идентична дейтаграмме, посланной отправителем. Однако на практике обеспечить это невозможно. Чтобы понять, почему, напомним, что взаимодействие по протоколу IP выполняется по принципу “от машины к машине”. Это означает, что при передаче дейтаграммы от одного маршрутизатора к другому она будет проходить по иерархии протоколов каждого устройства. В частности, каждый промежуточный маршрутизатор уменьшает на единицу значение поля времени жизни IP-дейтаграммы и пересчитывает контрольную сумму ее заголовка.

В спецификации IPsec поля заголовка IP-дейтаграммы, которые должны изменяться при ее передаче, называются *изменяемыми*. Чтобы такие поля не влияли на процесс аутентификации и не приводили к ошибкам вычислений, в спецификации IPsec говорится, что они должны быть исключены из аутентификации. Таким образом, получив дейтаграмму, программы поддержки протокола IPsec аутентифицирует только неизменяемые поля (например, адрес отправителя и тип протокола).

**Туннелирование по закрытому каналу на основе протоколов IPsec**

В разделе [Взаимодействие частных сетей (NAT, VPN)](https://it.rfei.ru/course/%7EGZqN/-/%7EyHb0) уже говорилось о том, что для обеспечения секретности при передаче данных между двумя сетевыми центрами применяется технология VPN, в которой наряду с туннелированием IP-в-IP используется шифрование. Стандарты IPsec специально разрабатывались для создания зашифрованного туннеля. В частности, в них определены специальные “туннельные” версии и для заголовка аутентификации, и для заголовка инкапсуляции зашифрованных данных (ESP). Структура дейтаграмм в режиме туннелирования показана на рис. 18.6.

Рис. 18.6. Структура дейтаграмм, использующихся в режиме туннелирования по закрытому каналу на основе протокола IPsec, содержащих данные для аутентификации (а) и зашифрованные данные (б). Содержимое внутренней дейтаграммы полностью защищено

**Обязательные алгоритмы шифрования**

В спецификации IPsec определен минимальный набор алгоритмов, поддержка которых обязательна (т.е. ими должны быть снабжены все программные реализации). Особенности реализации определены в стандарте для каждого конкретного случая. Список обязательных алгоритмов приведен в табл. 18.1.

Таблица 18.1. Алгоритмы шифрования, которые являются обязательными для реализации в IPsec

|  |  |
| --- | --- |
| ***Аутентификация*** |  |
| НМАС с MD5 | RFC 2403 |
| НМАС с SHA-1 | RFC 2404 |
| ***Инкапсуляция зашифрованных данных*** |  |
| DES в режиме цепочки связанных блоков (СВС) | RFC 2405 |
| НМАС с MD5 | RFC 2403 |
| НМАС с SHA-1 | RFC 2404 |
| Отсутствие аутентификации |  |
| Отсутствие шифрования |  |

**Механизм защищенных сокетов**

К середине 1990-ых, когда стало очевидно, что для проведения торговых операций в глобальной сети Internet нужно обеспечить безопасность передаваемых данных, несколькими группами были предложены механизмы безопасности для использования в Web. Одно из предложений стало стандартом де-факто, хотя оно и не было формально принято группой IETF.

Технология под названием *протокол защищенных сокетов (Secure Sockets Layer*, или *SSL*) была первоначально создана компанией Netscape.

Как следует из названия протокола SSL, он находится на том же уровне, что и API сокетов. Когда клиент использует SSL для связи с сервером, средства протокола SSL позволяют сторонам аутентифицировать друг друга. После этого обе стороны согласовывают алгоритм шифрования, который они будут использовать. И наконец, протокол SSL позволяет двум сторонам установить зашифрованный канал связи (т.е. канал, при передаче данных по которому они шифруются с использованием выбранного алгоритма при заданном уровне секретности).

**Брандмауэры и доступ к сети Internet**

Механизмы управления *доступом к объединенной сети* предназначены для защиты определенной сети или всех сетей организации от нежелательного подключения несанкционированных пользователей. Эти механизмы не позволяют посторонним получать или изменять внутренние данные организации, а также каким-либо образом препятствовать взаимодействию пользователей по корпоративной локальной сети. Для создания успешного механизма управления доступом нужно тщательно продумать комплекс мер: ограничения на топологию сети, промежуточное хранение информации и фильтры пакетов.

Основным компонентом для управления доступом к объединенной сети является специализированное устройство, называемое *брандмауэром (firewall*)1. Обычно брандмауэр устанавливается между внутренней сетью организации и каналом, ведущим к внешним сетям (например, к глобальной сети Internet). Брандмауэр разделяет объединенную сеть на две области, которые неофициально называются *внутренней* и *внешней*.

1. Термин *брандмауэр* позаимствован из строительства, где он обозначает толстую несгораемую стену, благодаря которой секция строения становится непроницаемой для огня.

**Множественные соединения и самые слабые связи**

Хотя сама идея брандмауэра проста, ее реализация усложняется множеством факторов. Один из них — внутренняя сеть организации может иметь несколько внешних соединений. При этом руководство организации должно сформировать *периметр безопасности (security perimeter)*, установив брандмауэр на каждое внешнее соединение.

Чтобы гарантировать эффективность периметра безопасности, во всех брандмауэрах должны использоваться одинаковые ограничения доступа1. В противном случае злоумышленники могут обойти ограничения, наложенные одним брандмауэром, и зайти в объединенную сеть организации через другой брандмауэр. Все сказанное выше можно подытожить так.

Если объединенная сеть организации имеет несколько внешних соединений, то на каждое из них необходимо установить брандмауэр и скоординировать их работу. Если на всех брандмауэрах будут использоваться разные ограничения доступа, то безопасность сети организации может оказаться под угрозой.

1. Хорошо известно, что уровень безопасности системы определяется ее самым слабым звеном. Этот принцип был назван *аксиомой самой слабой связи* по аналогии с поговоркой, что крепость цепи определяется ее самым слабым звеном.

**Реализация брандмауэра**

В каком же виде должен быть реализован брандмауэр? Теоретически брандмауэр должен попросту блокировать все несанкционированные соединения, осуществляемые между компьютерами организации и компьютерами, находящимися за ее пределами. На практике же, детали реализации зависят от применяемых сетевых технологий, пропускной способности внешнего канала связи, степени его загрузки и принятых в организации правил использования сети. Таким образом, нет единого решения, приемлемого для всех организаций. Поэтому создание эффективного брандмауэра в соответствии с техническими условиями заказчика может оказаться трудной задачей.

Чтобы брандмауэр не замедлял работу сети, его аппаратное и программное обеспечение должно быть оптимизировано на решение конкретной задачи. К счастью, в большинство коммерческих маршрутизаторов включен быстродействующий механизм фильтрации, который выполняет большую часть работы.

Администратор может сконфигурировать фильтр в маршрутизаторе так, чтобы маршрутизатор блокировал прохождение определенных дейтаграмм. После того как будет подробно описана работа фильтра, мы покажем, как на его основе создать брандмауэр.

В конце раздела будет показано, как можно использовать фильтр в сочетании с другим механизмом для обеспечения безопасного и в то же время гибкого процесса взаимодействия.

**Фильтры пакетов**

Во многих коммерческих маршрутизаторах реализован механизм, усовершенствующий обычную маршрутизацию и позволяющий администратору управлять обработкой пакетов. Этот механизм неофициально называется *фильтром пакетов*. Для его работы требуется, чтобы администратор определил, как маршрутизатор должен обрабатывать каждую дейтаграмму. Например, администратор может настроить маршрутизатор так, чтобы он *отфильтровывал* (т.е. блокировал) все дейтаграммы, полученные от определенного отправителя или предназначенные для использования в определенном приложении. При этом все другие дейтаграммы маршрутизатор будет перенаправлять к получателям как обычно.

Происхождение термина *фильтр пакетов* связано с тем, что механизм фильтрования не оставляет никаких следов о прохождении нежелательной дейтаграммы (например, записи в системном журнале или в списке полученных дейтаграмм). Фильтр обрабатывает каждую дейтаграмму отдельно. Получив дейтаграмму, маршрутизатор сначала пропускает ее через свой фильтр пакетов, а затем выполняет необходимую обработку этой дейтаграммы. Если дейтаграмма не удовлетворяет условиям фильтрации, маршрутизатор тут же ее аннулирует.

Поскольку в семействе протоколов TCP/IP ничего не сказано по поводу стандарта для фильтров пакетов, при реализации фильтра каждый разработчик маршрутизатора может создать фильтр, поддерживающий нестандартные возможности и выбрать для его конфигурации собственный интерфейс. В некоторых маршрутизаторах администратор может сконфигурировать работу фильтра для каждого интерфейса, в то время как в других одна конфигурация параметров фильтра применяется сразу для всех интерфейсов.

Обычно при определении дейтаграмм, которые фильтр должен блокировать, администратор может использовать любую комбинацию IP-адресов отправителя и получателя, типа протокола, номера портов протокола отправителя и получателя.

Пример параметров настройки фильтра для маршрутизатора с двумя интерфейсами (рис. 18.7) показан в табл. 18.2. Маршрутизатор, в котором поддерживается фильтр пакетов, составляет основной компонент брандмауэра.

Рис. 18.7. Маршрутизатор с двумя интерфейсами

Таблица 18.2. Пример параметров настройки фильтра

| **Номер интерфейсa** | **IP-aдpec отправителя** | **IP-aдpec получателя** | **Протокол** | **Порт отправителя** | **Порт получателя** |
| --- | --- | --- | --- | --- | --- |
| 2 | \* | \* | TCP | \* | 21 |
| 2 | \* | \* | TCP | \* | 23 |
| 1 | 128.5.0.0/16 | \* | TCP | \* | 25 |
| 2 | \* | \* | UDP | \* | 43 |
| 2 | \* | \* | UDP | \* | 69 |
| 2 | \* | \* | UDP | \* | 79 |

В этом примере администратор заблокировал все входящие дейтаграммы, предназначенные для популярных сетевых служб, а также один из видов отправляемых дейтаграмм. Фильтр блокирует все исходящие дейтаграммы, полученные от узла, IP-адрес сети которого соответствует 16-битовому префиксу 128.5.0.0, и предназначенные для удаленного сервера обработки электронной почты (порт 25 протокола TCP, соответствующий протоколу SMTP). Фильтр также блокирует входящие дейтаграммы, предназначенные для служб FTP (порт 21 протокола TCP), TELNET (порт 23 протокола TCP), WHOIS (порт 43 протокола UDP), TFTP (порт 69 протокола UDP) и FINGER (порт 79 протокола TCP).

**Безопасность и параметры фильтра пакетов**

Хотя в списке параметров конфигурации фильтра, приведенного в табл. 18.2, перечислено небольшое количество стандартных служб, которые должны блокиро¬ваться, такой подход не годится для создания эффективного брандмауэра по трем причинам.

Во-первых, количество портов стандартных служб очень велико, и оно постоянно растет. Таким образом, если вносить в список каждую новую службу, то администратор должен постоянно модифицировать этот список. Если он что-либо забудет, то брандмауэр может стать уязвимым.

Во-вторых, большая часть трафика объединенной сети не содержит пакеты, в которых указаны номера портов стандартных служб. Причина в том, что обычно номера портов службам присваиваются динамически, как, например, в службе *дистанционного вызова процедур (Remote Procedure Call*, или *RPC*). Кроме того, программисты могут выбирать не¬стандартные номера портов для собственных приложений типа клиент/сервер.

В-третьих, даже если внести в список номера портов всех популярных служб, брандмауэр все равно останется уязвимым для *туннелирования*. При туннелиро-вании можно обойти систему безопасности, если узел или маршрутизатор, относящиеся к внутренней части сети, будет принимать инкапсулированные дейтаграммы из внешней части сети, удалять один уровень инкапсуляции и пересылать дейтаграммы службе, которая обычно блокирована брандмауэром.

Как эффективно использовать фильтр пакетов при создании брандмауэра? Для ответа на этот вопрос необходимо рассмотреть процесс фильтрации с другой стороны. Речь о том, что вместо определения параметров дейтаграммы, которая должна быть отфильтрована, брандмауэр должен по умолчанию блокировать все дейтаграммы, кроме тех, которые предназначены для указанных сетей, узлов и портов протоколов. Для них соединение с внешним миром будет разрешено. Таким образом, при настройке брандмауэра администратор руководствуется принципом: все, что явно не разрешено, — запрещено. Затем он должен тщательно рассмотреть стратегию информационной безопасности организации и в соответствии с ней открыть нужные порты. Во многих реализациях фильтров пакетов администратор должен определить набор дейтаграмм, которые надо пропускать, вместо набора дейтаграмм, которые надо блокировать. Сказанное выше можно подытожить так.

Для того чтобы брандмауэр, использующий фильтрование дейтаграмм, был эффективным, он должен ограничивать прохождение дейтаграмм для всех IP-адресов отправителя и получателя, протоколов и номеров портов, кроме тех, которые указаны администратором явно.

Для облегчения настройки брандмауэра можно использовать такие реализации фильтра пакетов, в которых администратор должен определить набор пропускаемых дейтаграмм вместо набора блокируемых дейтаграмм.

**Влияние ограничения доступа на клиентов**

Казалось бы, многие потенциальные проблемы безопасности можно решить, запретив посторонним машинам доступ к серверу организации путем фильтрации дейтаграмм, прибывающих по неизвестному порту протокола. Однако применение такого брандмауэра имеет одно последствие: компьютер, находящийся во внутренней сети, не может стать клиентом и получить доступ к службе, запущенной на сервере, находящемся во внешней сети. Чтобы понять, почему это так, напомним, что клиент, в отличие от сервера, не использует стандартный номер порта. После запуска программа клиента обращается к операционной системе за свободным номером порта протокола (т.е. таким номером, который не относится ни к стандартным портам, ни к портам, используемым на компьютере клиента в настоящее время). Пытаясь связаться с сервером, расположенным во внешней сети, клиент скорее всего создаст одну или несколько дейтаграмм и отправит их серверу. У каждой исходящей дейтаграммы в качестве номера порта получателя указывается один из стандартных номеров портов протокола, а в качестве порта отправителя — номер порта, полученный программой клиента у операционной системы. Когда такие дейтаграммы проходят через брандмауэр во внешнюю сеть, он не блокирует их. При формировании ответа сервер поменяет номера портов местами в заголовке дейтаграммы. При этом порт клиента становится портом получателя, а порт сервера — портом отправителя. Однако когда такая дейтаграмма достигнет брандмауэра, она будет блокирована, поскольку она адресована по неизвестному порту получателя. Эту важную идею можно сформулировать так.

Если брандмауэр организации будет отфильтровывать все входящие дейтаграммы, посланные по неизвестному порту протокола, за исключением портов, соответствующих службам, которые организация сделала доступными извне, приложение, запущенное во внутренней сети, не может стать клиентом сервера, работающего во внешней сети.

**Доступ к Internet в обход брандмауэра через proxy-серверы**

Очевидно, что не все организации могут настроить свои брандмауэры так, чтобы те блокировали дейтаграммы, поступающие по неизвестному номеру порта протокола. Бывают случаи, когда необходимо создать безопасный брандмауэр, который предотвратит нежелательный доступ из вне, и в то же время позволит пользователям внутренней сети получить доступ ко внешним службам. При этом необходимо выработать специальный механизм безопасности, который и является второй основной частью структуры брандмауэра.

В общем случае организация может обеспечить безопасный доступ ко внешним службам только через защищенный компьютер. Понятно, что сделать все компьютерные системы в организации защищенными задача не из легких. Поэтому обычно с каждым брандмауэром связывают один защищенный компьютер и устанавливают на этом компьютере набор шлюзов уровня приложения. Для того чтобы такой компьютер мог служить в качестве безопасного канала связи, его степень защиты должна быть чрезвычайно высока. Поэтому такой компьютер часто называют *бастионным узлом*. Место бастионного узла в общей системе безопасности организации показано на рис. 18.8.

Бастионный узел обеспечивает безопасный доступ ко внешним службам. При этом от брандмауэра не требуется, чтобы он пропускал дейтаграммы, адресованные произвольным получателям во внутренней сети.

Рис. 18.8. Структурная схема брандмауэра, содержащего бастионный узел

Как показано на рис. 18.8, брандмауэр имеет два логических барьера. Внешний барьер блокирует весь входящий трафик, кроме (1) дейтаграмм, предназначенных для служб, запущенных на бастионном узле, которые организация хочет сделать доступными извне, и (2) дейтаграмм, предназначенных для клиентов, работающих на бастионном узле. Внутренний барьер также блокирует весь поступающий трафик, кроме дейтаграмм, отправленных бастионным узлом.

В большинстве брандмауэров предусмотрен также обходной канал, конфигурируемый вручную. Он позволяет администраторам временно передавать весь трафик или некоторую его часть между внутренним и внешним узлом сети (например, при тестировании или поиске неисправности в сети).

Чтобы понять, как работает бастионный узел, рассмотрим доступ к службе Web. Поскольку брандмауэр не допускает, чтобы компьютер, подключенный ко внутренней сети, получал входящие дейтаграммы, пользователь не может использовать броузер для прямого доступа к службе Web, находящейся во внешней сети. Поэтому на бастионном узле запускается proxy-сервер службы Web. Каждый броузер внутри организации конфигурируется так, чтобы все запросы посылались proxy-серверу. Всякий раз, когда пользователь выбирает ссылку или вводит URL, его броузер связывается с proxy-сервером. Proxy-сервер связывается с Web-сервером, получает указанную страницу, а затем доставляет ее пользователю.

**Подробнее о структуре брандмауэра**

Теперь, когда понятна основная идея построения брандмауэра, ее реализация не должна казаться сложной. Для реализации каждого из барьеров, показанных на рис. 18.8, потребуется маршрутизатор, поддерживающий фильтрацию пакетов1. Маршрутизаторы взаимодействуют с бастионным узлом с помощью отдельных сетей. Например, организация, сеть которой подключена к глобальной сети Internet, могла бы реализовать брандмауэр так, как показано на рис. 18.9.

Рис. 18.9. Брандмауэр, реализованный на основе двух маршрутизаторов и бастионного узла. Один из маршрутизаторов связан с глобальной сетью Internet

Как показано на рис. 18.9, внешний барьер реализован на маршрутизаторе *R2*. Он фильтрует весь входящий трафик, кроме дейтаграмм, предназначенных для бастионного узла, *Н*. Внутренний барьер реализован на маршрутизаторе *R1*, который изолирует остальную часть внутренней сети организации от посторонних. Он блокирует все входящие дейтаграммы, кроме тех, которые приходят от бастионного узла.

Конечно, надежность всего брандмауэра зависит от степени безопасности бастионного узла. Если злоумышленник сможет получить доступ к компьютеру бастионного узла, то по сути он получит полный доступ к внутренней сети. Кроме того, злоумышленник может воспользоваться изъянами в системе безопасности операционной системы бастионного узла или одного из сетевых приложений, которые на нем выполняются. Таким образом, администраторы должны быть особенно внимательны при выборе и конфигурировании программного обеспечения для бастионного узла. Вывод такой.

Хотя бастионный узел необходим для осуществления связи с внешней сетью в обход брандмауэра, надежность последнего зависит от степени безопасности бастионного узла. Злоумышленник, который знает изъяны в системе безопасности операционной системы бастионного узла, может получить полный доступ к внутренней сети организации.

1. В некоторых организациях используется конфигурация с одиночным брандмауэром, в которой все функциональные возможности реализованы на одном физическом маршрутизаторе.

**Тупиковая сеть**

На первый взгляд может показаться, что на рис. 18.9 изображена одна лишняя сеть, которая соединяет два маршрутизатора и бастионный узел. Такую небольшую сеть часто называют *тупиковой (stub network)*. Возникает вопрос: необходима ли тупиковая сеть, или можно подключить бастионный узел непосредственно к одной из внутренних сетей организации? Ответ зависит от величины трафика, ожидаемого извне. Тупиковая сеть изолирует внутреннюю сеть организации от трафика, создаваемого входящими дейтаграммами. В частности, поскольку маршрутизатор *R2* пропускает все дейтаграммы, предназначенные для бастионного узла, внешние узлы могут посылать произвольное количество таких дейтаграмм по тупиковой сети. Если пропускная способность внешнего канала, связывающего организацию с Internet, меньше пропускной способности тупиковой сети, то отдельная физическая сеть для тупиковой сети может быть просто не нужна. Однако, хотя применение отдельной физической сети несколько увеличивает стоимость брандмауэра, зато позволяет полностью обезопасить инфраструктуру внутренней сети от внешних вторжений.

**Альтернативный вариант реализации брандмауэра**

Брандмауэр, структура которого показана на рис. 18.9, хорошо работает, если организация имеет один выделенный последовательный канал, соединяющий ее с остальной частью глобальной сети Internet.

Однако в некоторых случаях топология сетевого центра может быть несколько иной. Предположим, у компании есть три или четыре крупных клиента, каждому из которых надо заносить в базу данных или извлекать из нее большие объемы информации. При этом администрация компании хочет иметь только один брандмауэр1 и разрешить клиентам связываться с внешним миром. Одна из возможных структур брандмауэра, которая обеспечивает несколько внешних соединений показана на рис. 18.10.

Рис. 18.10. Альтернативная структура брандмауэра, которая обеспечивает поддержку нескольких внешних соединений через один брандмауэр. Использование одного брандмауэра для нескольких соединений уменьшает стоимость системы

Как показано на рис. 18.10, в альтернативной структуре функции брандмауэра несколько расширены за счет поддержки нескольких внешних сетей, в которые ведут внешние каналы связи. Чтобы защитить внутреннюю корпоративную сеть, применяется маршрутизатор *R1*. Он выполняет те же функции, что и прежде (см. рис. 18.9), т.е. блокирует все входящие дейтаграммы за исключением тех, которые отправлены бастионным узлом. Внешние сети подключаются к брандмауэру через маршрутизаторы *R2, …, R1*.

Чтобы понять, зачем в брандмауэре, поддерживающем несколько внешних соединений, обычно устанавливают один маршрутизатор для каждого соединения, напомним, что по определению все внешние сетевые центры не доверяют друг другу. Другими словами организация, управляющая брандмауэром, не доверяет полностью никакой из внешних организаций, и никакие внешние организации не доверяют друг другу полностью. Фильтр пакетов, поддерживаемый каждым из маршрутизаторов, к которому подключен внешний канал, должен быть сконфигурирован так, чтобы он ограничивал трафик для этого соединения. В результате владелец брандмауэра может гарантировать, что, хотя все внешние каналы подключены к одной общей сети, никакая дейтаграмма не попадет из одного внешнего канала в другой.

Таким образом, организация, управляющая брандмауэром, может гарантировать своим клиентам, что их соединение безопасно. Все сказанное выше можно подытожить так.

Подключение нескольких внешних сетевых центров к одному брандмауэру обычно выполняется через выделенные маршрутизаторы, (т.е. каждый внешний канал подключается к отдельному маршрутизатору). При этом можно предотвратить нежелательные потоки данных от одного внешнего сетевого центра к другому.

1. Использование одного брандмауэра для всех подключений является более дешевым и более легким для управления, чем использование отдельного брандмауэра для каждого соединения.

**Текущий контроль и регистрация**

Текущий контроль — один из наиболее важных аспектов проектирования брандмауэра. Сетевой администратор, ответственный за брандмауэр, должен быть осведомлен о всех попытках злоумышленников взломать его систему безопасности. Если брандмауэр ничего не сообщает о произошедших инцидентах, то администратор никогда не узнает о проблемах.

Текущий контроль может быть *активным* или *пассивным*.

При активном контроле брандмауэр уведомляет администратора как только происходит инцидент. Главным преимуществом активного контроля является скорость — администратор немедленно узнает о потенциальной проблеме. Главное неудобство заключается в том, что активные системы контроля зачастую выдают так много информации, что администратор не может целиком охватить ее или вовремя выявить проблему. Поэтому большинство администраторов предпочитает использовать пассивный контроль или комбинацию пассивного контроля с сообщениями системы активного контроля о некоторых особо опасных инцидентах.

При пассивном контроле брандмауэр регистрирует каждый инцидент в файле журнала на диске. Обычно система пассивного контроля регистрирует информацию о нормальном трафике, например статистику прохождения и количество отфильтрованных дейтаграмм. При этом администратор может получить доступ к журналу в любое время. Большинство администраторов использует для этого специальную компьютерную программу. Главное преимущество системы пассивного контроля заключается в том, что она регистрирует происходящие события. Проанализировав журнал сообщений, администратор может выявить узкие места в системе безопасности, а если что-то все-таки случится, он сможет восстановить хронологию предшествующих событий. Важно то, что администратор может периодически (например, анализировать журнал раз в день), чтобы определить, увеличивается или уменьшается со временем количество попыток несанкционированного доступа к сети организации.

**Проверка знаний: Безопасность в объединенной сети и брандмауэры (IPsec)**

**Задание 1**

Почему обеспечить безопасность в глобальной сети интернет особенно тяжело?

дейтаграммы проходят через несколько сетей и маршрутизаторов, которые не принадлежат ни отправителю, ни получателю и не контролируются ими

потому, что обмен информации в сети интернет происходит открытым способом

слишком много пользователей, сетей и маршрутизаторов

**Задание 2**

Каким ресурсам сложнее обеспечить безопасность?

виртуальным

физическим

одинаково сложно любым: и физическим и виртуальным

**Задание 3**

Как называется семейство протоколов, которое обеспечивает безопасность связи в глобальной сети интернет?

IPv6

IPv4

TCP

IPsec

**Задание 4**

Что (или кто) представляет собой наиболее уязвимое звено в любой системе безопасности?

мосты маршрутизации

люди

протоколы

**Задание 5**

Как обычно выполняется подключение нескольких внешних сетевых центров к одному брандмауэру?

через бастионный узел

через выделенные маршрутизаторы

через глобальную сеть интернет

**Задание 6**

Какой компанией было первоначально создана технология под названием *«протокол защищенных сокетов»*?

Netscape

Microsoft

Bios

**Задание 7**

Как называется специализированное устройство, предназначенное для защиты сети от нежелательного подключения несанкционированных пользователей?

брандмауэр

антивирус

сетевой экран

**Задание 8**

Какой механизм дает возможность в протоколе IPseс обрабатывать зашифрованные данные?

раскодировка зашифрованных данных

компиляция зашифрованных данных

инкапсуляция зашифрованных данных

**Задание 9**

Как, при наличии заголовка аутентификации протокола IPsec, получатель может определить тип данных, передаваемых в дейтаграмме?

по полю «длина содержимого»

по полю «тип следующего заголовка»

по полю «зарезервировано»

**Будущее протокола TCP/IP (IPv6)**

Тесная связь развития технологии TCP/IP и глобальной сети Internet обусловлена несколькими причинами.

Во-первых, Internet — наибольшая из созданных глобальных сетей, в которой используется семейство протоколов TCP/IP. Поэтому многие проблемы, связанные с размером сети, в Internet возникают до того, как они появятся в других сетях, использующих протоколы семейства TCP/IP.

Во-вторых, исследования и разработка новых протоколов семейства TCP/IP финансируются частными и государственными компаниями, деятельность которых зависит от стабильной работы Internet. Поэтому они склонны финансировать связанные с этим проекты.

В-третьих, поскольку большинство исследователей ежедневно пользуются Internet, они непосредственно заинтересованы в решении проблем. И эта заинтересованность приводит к улучшению качества предоставляемых услуг и расширению функциональных возможностей.

Учитывая то, что работа миллионов пользователей и десятков тысяч сетевых центров, разбросанных по всему миру, зависит от ежедневного использования глобальной сети Internet, может показаться, что Internet является полностью стабильным средством производства. Уже пройдена ранняя стадия развития, когда каждый пользователь был также экспертом. На современном этапе мало кто из пользователей до конца понимает технологию Internet. Но несмотря на видимость стабильности, ни Internet, ни набор протоколов TCP/IP не являются неизменными. Организации находят новые применения технологии, исследователи решают возникающие проблемы сети, инженеры совершенствуют используемое оборудование. Короче говоря, технология продолжает развиваться.

Цель этого раздела — обсудить происходящий эволюционный процесс, а также рассмотреть одно из наиболее значительных инженерных достижений: проект изменения протокола IP. Если проект будет внедрен поставщиками Internet-услуг, то это окажет значительное влияние на протоколы семейства TCP/IP и глобальную сеть Internet.

* [Зачем что-то менять?](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690654)
* [Новые правила](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690655)
* [Причины пересмотра стандарта IPv4](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690656)
* [Путь к новой версии протокола IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690657)
* [Название следующей версии протокола IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a31001183c7a690658)
* [Свойства протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690659)
* [Общая структура дейтаграммы протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065a)
* [Формат основного заголовка дейтаграммы IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065b)
* [Дополнительные заголовки дейтаграммы протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065c)
* [Анализ дейтаграммы протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065d)
* [Фрагментация и сборка дейтаграммы в протоколе IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065e)
* [Важность сквозной фрагментации](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69065f)
* [Маршрутизация от источника в протоколе IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690660)
* [Параметры протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690661)
* [Размер адресного пространства протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690662)
* [Шестнадцатеричная запись с разделением двоеточием](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690663)
* [Три основных типа адресов протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690664)
* [Взаимозаменяемость широковещательной и многоадресатной передачи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690665)
* [Проблема выбора, стоящая перед разработчиками](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690666)
* [Проект распределения адресного пространства протокола IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690667)
* [Кодирование адресов протокола IPv4 и переход к адресам IPv6](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690668)
* [Неопределенные адреса и адреса петли обратной связи](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690669)
* [Иерархия одноадресатных адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066a)
* [Структура сгруппированных глобальных одноадресатных адресов](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066b)
* [Идентификаторы интерфейса](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066c)
* [Дополнительный уровень иерархии](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066d)
* [Локальные адреса](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066e)
* [Автоконфигурация и перенумерация сетей](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69066f)
* [Проверка знаний: Будущее протокола TCP/IP (IPv6)](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690670)

**Зачем что-то менять?**

Базовая технология TCP/IP хорошо работала более двух десятилетий. Зачем же ее менять? Необходимость пересмотра стандартов протоколов связана с изменением технологий, лежащих в их основе, и распространением этих протоколов.

* *Новые компьютерные и коммуникационные технологии*. Компьютерное и сетевое оборудование продолжает развиваться. И как только новые технологии появляются, они сразу же внедряются в Internet.
* *Новые приложения*. Так как программисты постоянно придумывают новые способы использования протоколов TCP/IP, возникает необходимость в поддержке дополнительных протоколов. Например, привлечение внимания к IP-телефонии за последние годы дало толчок к созданию протоколов передачи данных в реальном масштабе времени.
* *Увеличение размеров и нагрузки*. Уже в течение многих лет происходит непрерывный экспоненциальный рост Internet. Сейчас размер объединенной сети удваивается каждые девять месяцев или даже быстрее. В 1999 году новый узел в Internet появлялся примерно каждые две секунды. Так же быстро вырос трафик, в связи с широким распространением анимированных изображений, видео- и аудиоклипов.

**Новые правила**

Распространяясь в новые страны, Internet существенно меняется: право на управление частью объединенной сети получают новые административные органы. Смена управления приводит к изменениям в административных правилах и делегирует механизмы внедрения этих правил.

Как мы выяснили, и структура объединенной сети Internet, и используемые в ней протоколы, развиваются, все более удаляясь от централизованной модели. Развитие продолжается за счет того, что к Internet подключается большее количество национальных магистральных сетей. Это приводит к созданию все более сложных правил регулирования взаимодействием. С подобными проблемами сталкиваются многочисленные корпорации при объединении внутренних локальных сетей на основе протокола TCP/IP, поскольку они пытаются сначала определить правила взаимодействия, а затем развить механизмы внедрения этих правил.

Таким образом, многие усилия исследователей и инженеров, связанные с протоколами семейства TCP/IP продолжают сосредотачиваться на поиске способов поддержки работы новых административных групп.

**Причины пересмотра стандарта IPv4**

В настоящее время в семействе протоколов TCP/IP и в глобальной сети Internet основной механизм передачи информации обеспечивается за счет протокола IP версии 4 (*IPv4*), который почти не изменился с момента его создания1 в конце 1970-х годов. Успешное использование этой версии на протяжении многих лет свидетельствует о ее надежности и гибкости ее структуры. Со времени разработки стандарта IPv4 производительность процессоров увеличилась более чем на два порядка, стандартные объемы памяти возросли более чем в 100 раз, пропускная способность магистральной сети Internet увеличилась более чем в 7000 раз, появились новые технологии построения локальных сетей, количество узлов в объединенной сети Internet возросло от ничтожно малого до более чем 56 миллионов. Поскольку изменения произошли не в один момент, адаптация к ним была непрерывным процессом.

Несмотря на логичную структуру стандарта IPv4 в ближайшее время он должен быть заменен. Речь идет о том, что в ближайшем будущем возникнет проблема нехватки адресного пространства. Во времена разработки протокола IP 32-битового адресного пространства было вполне достаточно. Локальные сети использовались только в небольшом количестве организаций; еще меньше организаций было подключено к глобальным сетям. Сейчас же в большинстве корпораций средних размеров используются многочисленные локальные сети, а в наиболее крупных из них имеются корпоративные распределенные сети. Следовательно, даже при аккуратном распределении и использовании технологии NAT теперешнее 32-битовое адресное пространство не сможет обеспечить потребности всемирной сети Internet с учетом ее ожидаемого роста после 2020 года.

Необходимость увеличения адресного пространства — не единственная (хотя и главная) причина пересмотра структуры семейства протоколов TCP/IP. В частности, чтобы протокол IP можно было использовать в приложениях, работающих в реальном масштабе времени, были предложены системы, в которых потоковая передача дейтаграмм осуществляется после предварительного резервирования ресурсов. С целью повышения надежности приложений для электронной коммерции, следующая версия протокола IP разрабатывается с уже встроенной поддержкой средств безопасности, таких как аутентификация.

1. Версии с 1 по 3 никогда не были формально определены, а версия номер 5 была присвоена протоколу ST.

**Путь к новой версии протокола IP**

Для того чтобы сформулировать новую версию стандарта протокола IP, проблемной группе IETF потребовалось много лет. Так как IETF создает *открытые* стандарты, она предложила поучаствовать в этом процессе всему сообществу Internet. Производители компьютеров, комплектующих и программного обеспечения, пользователи, менеджеры, программисты, работники телефонных компаний и индустрии кабельного телевидения — все излагали свои требования к следующей версии протокола IP и обсуждали различные предложения.

Многие из предложенных решений удовлетворяли отдельным целям или устраивали определенную часть сообщества. В одном из основных предложений предполагалось сделать протокол IP более гибким за счет увеличения его сложности и увеличения накладных расходов на обработку дейтаграмм. В другом решении предлагалось использовать модификацию протокола OSI CLNS. В третьем (кстати, одном из самых интересных проектов) предполагалось сохранение большинства принципов старого протокола IP и создание простых расширений для поддержки *большего* адресного пространства. Это проект назвали *SIP1* (*Simple IP*). Он стал основой для расширенного варианта стандарта, в который включили идеи из других предложений. Расширенная версия была названа *Simple IP Plus (SIPP)* и, в конечном счете, легла в основу проекта следующей версии протокола IP.

Выбрать новую версию протокола IP было непросто. В результате огромной популярности глобальной сети Internet во всем мире возрос спрос на программные продукты, в которых используется протокол IP. Поэтому многие деловые круги видят открывающиеся экономические возможности и надеются, что новая версия протокола IP даст им преимущества в конкурентной борьбе. Кроме того, в этот процесс были вовлечены конкретные люди: некоторые в силу своего авторитета в технических вопросах, другие же воспринимали активное участие в нем, как веху в своей карьере. Поэтому дискуссии были жаркими.

1. Аббревиатура *SIP* сейчас относится к протоколу инициализации сеанса связи (*Session Initiation Protocol*), который используется для установки связи (например, в 1Р-телефонии).

**Название следующей версии протокола IP**

Проблемная группа IETF решила присвоить пересмотренной версии протокола IP номер 6 и назвать его *IPv6*1, чтобы отличать ее от современной версии *IPv4*. Решение пропустить версию номер 5 было принято из-за ряда возникших ошибок и недоразумений. Например, IAB опубликовала документ, в котором ошибочно называла следующую версию протокола *IP версии 7*, что многих ввело в заблуждение. Экспериментальному протоколу, называемому *Stream Protocol (ST*, или *потоковый протокол)*, присвоили номер версии 5, что также привело к недоразумению. Некоторые подумали, что протокол ST был выбран в качестве замены протокола IP. В конечном счете, чтобы устранить путаницу, IETF выбрала номер версии 6.

1. В некоторых документах этот проект называют IPng, или IP — The Next Generation (протокол IP следующего поколения).

**Свойства протокола IPv6**

Предложенный протокол IPv6 сохранил многие особенности, которые способствовали успеху версии IPv4. Фактически разработчики считают протокол IPv6 модифицированной версией протокола IPv4. Например, в протоколе IPv6 по-прежнему используется режим доставки дейтаграмм, не требующий установки соединения с получателем (т.е. каждая дейтаграмма маршрутизируется независимо); передающая сторона может выбирать размер дейтаграммы, а также должна установить максимальное количество переходов, которое может совершить дейтаграмма до того, как она будет аннулирована одним из промежуточных маршрутизаторов. Как будет показано ниже, протокол IPv6 также сохранил большинство возможностей, обеспечиваемых параметрами протокола IPv4, включая возможности фрагментации и маршрутизации от источника.

Несмотря на многие принципиальные сходства, в IPv6 изменена большая часть внутренней структуры протокола. Например, в нем используются адреса с увеличенной разрядностью и добавлено несколько новых свойств.

Более важным является то, что в протоколе IPv6 полностью пересмотрен формат дейтаграммы. Вместо полей параметров переменной длины в нем используется набор заголовков фиксированного формата. Подробности будут изучены нами после рассмотрения структуры основных изменений и причин их принятия.

Изменения, внесенные в протокол IPv6, можно разделить на семь категорий.

* *Увеличение адресного пространства*. Пожалуй, самое существенное изменение коснулось увеличения количества битов, занимаемых адресом. В протоколе IPv6 размер адреса увеличен в четыре раза, с 32 до 128 бит. Адресное пространство этого протокола настолько велико, что в обозримом будущем оно не может исчерпаться.
* *Расширенная иерархия адресов*. В IPv6 используется большее адресное пространство для создания дополнительных уровней иерархии адресов. В частности, в IPv6 можно определить иерархию провайдеров Internet, а также иерархическую структуру в пределах данного сетевого центра.
* *Гибкий формат заголовка*. В протоколе IPv6 используется совершенно новый и несовместимый со старым формат дейтаграммы. В отличие от заголовка фиксированного формата протокола IPv4, в протоколе IPv6 определен ряд дополнительных заголовков.
* *Улучшенный механизм параметров*. Как и в протоколе IPv4, в IPv6 в дейтаграмму может включаться дополнительная управляющая информация. В IPv6 добавлен ряд новых параметров, которые обеспечивают дополнительные возможности, недоступные в IPv4.
* *Поддержка возможности расширения протокола*. Возможно, наиболее существенным изменением является то, что, в отличие от протокола IPv4, в котором жестко установлена внутренняя структура протокола, в IPv6 заложена возможность расширения. Благодаря этому проблемная группа IETF может адаптировать этот протокол к любому сетевому оборудования или к новым приложениям.
* *Поддержка автоконфигурации и перенумерации*. В протоколе IPv6 предусмотрены средства, позволяющие автоматически назначать адреса компьютерам в изолированной сети. Это позволяет начать процесс сетевого взаимодействия без обращения к серверу за параметрами конфигурации или предварительной ручной настройки. В протоколе IPv6 также предусмотрены средства, позволяющие администратору динамически перенумеровывать сети.
* *Поддержка распределения ресурсов*. В протоколе IPv6 предусмотрено два средства для предварительного распределения сетевых ресурсов: концепция потока и спецификация дифференцированных служб. В последнем случае используется подход, аналогичный тому, что был в стандарте IPv4.

**Общая структура дейтаграммы протокола IPv6**

В протоколе IPv6 полностью изменился формат дейтаграммы. Дейтаграмма протокола IPv6 состоит из основного заголовка фиксированного размера, за которым может следовать несколько дополнительных заголовков, а после них — данные (19.1).

Рис. 19.1. Общая структура дейтаграммы протокола IPv6 с несколькими заголовками. Обязательным является присутствие только основного заголовка; дополнительные заголовки могут быть опущены

**Формат основного заголовка дейтаграммы IPv6**

Несмотря на поддержку *большего* адресного пространства, в основном заголовке протокола IPv6 содержится меньше информации, чем в заголовке дейтаграммы IPv4. Параметры и некоторые фиксированные поля, находящиеся в заголовке дейтаграммы IPv4, в IPv6 перенесены в дополнительные заголовки. В общем можно сказать, что изменения в формате заголовка дейтаграммы отражают изменения, сделанные в стандарте протокола.

* Изменилась граница выравнивания полей дейтаграммы (с 32 до 64 бит).
* Удалено поле длины заголовка, а вместо поля длины дейтаграммы используется поле *длины передаваемых данных*.
* Размер полей, содержащих адреса отправителя и получателя, увеличен до 16 октетов каждый.
* Информация о фрагментации перенесена из фиксированных полей основного заголовка в дополнительный заголовок.
* Поле, в котором хранилось *время жизни дейтаграммы*, заменено полем, содержащим *максимальное количество переходов*.
* Поле *типа обслуживания* заменено полем *класса трафика* и расширено полем, содержащим *метку потока*.
* Поле *типа протокола* заменено полем, определяющим тип следующего заголовка.

Формат основного заголовка протокола IPv6 показан на рис. 33.2. Некоторые поля основного заголовка IPv6 непосредственно соответствуют полям заголовка IPv4. Так, в первом 4-битовом поле указывается версия протокола. В дейтаграмме протокола IPv6 в этом поле всегда содержится значение 6. Как и в протоколе IPv4, адреса отправителя и получателя указываются в специальных полях. Однако в протоколе IPv6 под каждый адрес выделено не 4, а 16 октетов. Поле *максимального количества переходов* соответствует полю времени жизни дейтаграммы протокола IPv4. В отличие от IPv4, где значение времени жизни дейтаграммы интерпретируется как комбинация количества переходов и максимального времени передачи, в протоколе IPv6 соответствующее значение строго интерпретируется как максимальное число переходов, которое дейтаграмма может совершить, до того как она будет аннулирована.

Рис. 19.2. Формат 40-октетного основного заголовка, расположенного в начале каждой дейтаграммы протокола IPv6

Длина дейтаграммы в протоколе IPv6 указывается по-другому. Во-первых, так как размер основного заголовка фиксирован и равен 40 октетам, то в поле, содержащем длину самого заголовка, нет необходимости. Во-вторых, в протоколе IPv6 вместо поля, содержащего длину дейтаграммы (как в протоколе IPv4), используется 16-битовое поле, в котором указывается количество октетов, содержащихся в дейтаграмме, за исключением заголовка. Таким образом, в дейтаграмме протокола IPv6 может передаваться до 64К октетов данных.

Два поля в основном заголовке предназначены для принятия решений по пересылке дейтаграммы. Поле типа обслуживания, которое было в заголовке дейтаграммы протокола IPv4 заменено на поле класса трафика. Кроме того, в IPv6 появились новые механизмы, поддерживающие резервирование ресурсов и позволяющие маршрутизатору связывать каждую дейтаграмму со сделанным распределением ресурсов. В основе этих механизмов лежит абстрактное понятие *потока (flow)*. Поток представляет собой маршрут, проложенный в объединенной сети, вдоль которого промежуточными маршрутизаторами гарантируется связь определенного качества. В поле метки потока основного заголовка содержится информация, используемая маршрутизаторами для сопоставления дейтаграммы с определенным потоком, а также величина приоритета. Например, если двум приложениям необходимо передать видеоданные, то они могут создать поток, в пределах которого будет обеспечиваться соответствующие задержка и пропускная способность. С другой стороны, провайдер может запросить у абонентов спецификацию необходимого им качества связи, а затем с помощью потока ограничивать трафик, исходящий от определенного компьютера или приложения. Заметим, что потоки можно также использовать в пределах организации для управления сетевыми ресурсами и контроля над их равномерным распределением между приложениями. При сопоставлении дейтаграммы с определенным потоком маршрутизатором используется и адрес отправителя дейтаграммы, и идентификатор потока. Подведем итог.

Каждая дейтаграмма протокола IPv6 начинается с 40-октетного основного заголовка, в котором содержатся поля адресов отправителя и получателя, поле максимального количества переходов, поле класса трафика, поле метки потока и поле типа следующего заголовка. Таким образом, в дейтаграмме IPv6, кроме данных, должно содержаться по крайней мере еще 40 октетов информации.

**Дополнительные заголовки дейтаграммы протокола IPv6**

В качестве компромисса между универсальностью и эффективностью разработчиками была выбрана следующая структура заголовка: за основным заголовком фиксированного формата следует набор необязательных дополнительных заголовков. Действительно, чтобы протокол IPv6 был универсальным, в него должны быть включены механизмы, поддерживающие такие функции, как фрагментация, маршрутизация от источника и аутентификация. С другой стороны, расположить поля фиксированного формата, поддерживающие все механизмы, в заголовке дейтаграммы было бы неэффективно, поскольку в большинстве случаев они не будут использоваться. А использование в IPv6 адресов большего размера делало бы это еще более неэффективным. Например, при передаче дейтаграмм в пределах одной локальной сети, заголовок, содержащий пустые поля адресов, может занимать значительную часть каждого фрейма. Разработчики понимали, что невозможно заранее предсказать, какие средства могут понадобиться в будущем.

Принцип дополнительных заголовков протокола IPv6 аналогичен полям параметров дейтаграммы IPv4. При этом отправитель может выбирать, какие дополнительные заголовки следует включить в дейтаграмму, а какие — опустить. В результате обеспечивается максимальная универсальность. Подведем итог.

Дополнительные заголовки протокола IPv6 напоминают поля параметров дейтаграммы IPv4. В каждую дейтаграмму включаются только те дополнительные заголовки, которые необходимы для ее передачи.

**Анализ дейтаграммы протокола IPv6**

И в основном, и в дополнительных заголовках есть поле типа следующего заголовка. Значения этих полей используются программным обеспечением, которое установлено на промежуточных маршрутизаторах и у конечного получателя, для анализа содержимого дейтаграммы во время ее обработки. Для извлечения информации, содержащейся в заголовках дейтаграммы протокола IPv6, необходимо последовательно проанализировать все заголовки. На рис. 19.3 показаны значения полей типа следующего заголовка для трех дейтаграмм, одна из которых не содержит дополнительного заголовка, вторая и третья содержат один и два дополнительных заголовка, соответственно.

Рис. 19.3. Три дейтаграммы: только с основным заголовком (а); с основным и одним дополнительным заголовками (б); с основным заголовком и двумя дополнительными (в). Определить тип следующего заголовка можно по значению одноименного поля текущего заголовка

Очевидно, что анализ дейтаграммы протокола IPv6, состоящей только из основного заголовка и данных, не займет много времени; он столь же эффективен, как и анализ дейтаграммы протокола IPv4. Более того, промежуточным маршрутизаторам нужно анализировать только дополнительный заголовок, который был добавлен предыдущим маршрутизатором (*hop-by-hop*), а остальные дополнительные заголовки обрабатываются только конечными получателями.

**Фрагментация и сборка дейтаграммы в протоколе IPv6**

Как и в протоколе IPv4, в IPv6 сборка фрагментированной дейтаграммы выполняется на машине конечного получателя. Тем не менее разработчики внесли изменения, которые позволяют избежать фрагментации дейтаграмм маршрутизаторами. Напомним, что в протоколе IPv4 любая дейтаграмма может быть разбита промежуточным маршрутизатором на фрагменты, если ее размер превышает значение MTU сети, через которую она будет передаваться. В протоколе IPv6 фрагментация является сквозной (end-to-end); промежуточным маршрутизаторам нет необходимости производить фрагментацию. Отправитель, выполняющий фрагментацию, может воспользоваться двумя возможностями.

Во-первых, он может разбить дейтаграмму на фрагменты в соответствии с *минимальным гарантированным значением MTU*, которое равно 1280 октетам.

Во-вторых, он может выполнить *поиск минимального значения* MTU по пути следования пакетов.

В любом случае дейтаграмма разбивается отправителем таким образом, чтобы каждый фрагмент был меньше ожидаемого значения MTU по пути следования дейтаграммы.

В основном заголовке протокола IPv6 не содержится полей, аналогичных тем, которые используются для поддержки фрагментации в заголовке IPv4. При выполнении фрагментации отправитель помещает небольшой дополнительный заголовок после основного заголовка в каждый фрагмент. На рис. 19.4 показан формат *дополнительного заголовка фрагментации (Fragment Extension Header)*.

В протоколе IPv6 сохранились основные функциональные возможности протокола IPv4 в плане поддержки фрагментации. Как и раньше, смещение каждого фрагмента выражается числом, кратным *8* октетам (т.е. для определения истинного смещения фрагмента это число надо умножить на *8*). Для поиска последнего фрагмента предназначено однобитовое поле М, аналогичное биту More fragments (дополнительные фрагменты) протокола IPv4. В поле *идентификации дейтаграммы* содержится уникальное значение1</sup, используемое получателем для группировки фрагментов. Наконец, поле *RS* в настоящее время зарезервировано; его два бита устанавливаются в ноль во время передачи и игнорируются получателем.

1. В протоколе IPv6 размер этого поля увеличен до 32 бит в связи с поддержкой высокоскоростных сетей.

**Важность сквозной фрагментации**

Введение сквозной фрагментации позволяет снизить нагрузку на промежуточные маршрутизаторы. При этом каждый из них может обрабатывать большее количество дейтаграмм за единицу времени. Действительно, для фрагментации дейтаграммы протокола IPv4 маршрутизатор должен затратить некоторое количество процессорного времени. Если большая часть получаемых дейтаграмм должна быть фрагментирована, степень использования центрального процессора маршрутизатора может достигать 100%. Введение сквозной фрагментации имеет одно важное следствие. Оно означает, что при этом не выполняется основополагающее предположение протокола IPv4 о том, что маршруты могут изменяться динамически.

Чтобы стала понятна важность сквозной фрагментации, напомним, что при разработке протокола IPv4 специально предусматривалась возможность изменения маршрута дейтаграмм. Например, если часть сети или один из маршрутизаторов вышли из строя, то трафик может направляться по другому пути. Основное преимущество такой системы — гибкость: изменение направления трафика может осуществляться без разрыва ТСР-соединения и без уведомления об этом отправителя и получателя. Однако в протоколе IPv6 маршруты так легко изменяться не могут, поскольку при этом меняется значение MTU маршрута. Если MTU нового маршрута будет меньше исходного, то либо дейтаграммы должны разбиваться промежуточным маршрутизатором на фрагменты, либо отправитель должен быть проинформирован об этом. Проблему можно сформулировать следующим образом.

При использовании сквозной фрагментации требуется, чтобы отправитель заранее определял значение MTU маршрута, ведущего к каждому получателю. Если размер дейтаграммы, больше, чем значение MTU пути, то исходящая дейтаграмма должна быть разбита отправителем на фрагменты. При сквозной фрагментации не допускается динамическое изменение маршрута следования дейтаграмм.

Для решения проблемы, связанной с изменением значения MTU при изменении маршрутов, в протокол IPv6 добавлено новое ICMP-сообщение об ошибке. Когда маршрутизатор обнаруживает, что необходима фрагментация, отправителю посылается это сообщение. Получив такое сообщение, отправитель вновь определяет минимальное значение MTU маршрута. Затем дейтаграммы разбиваются на фрагменты в соответствии с новым значением.

**Маршрутизация от источника в протоколе IPv6**

В протоколе IPv6 за отправителем сохранена возможность определять неточный маршрут от источника. Однако в отличие от протокола IPv4, в котором маршрутизация от источника задается с помощью параметров дейтаграммы, в протоколе IPv6 используется отдельный дополнительный заголовок. Как показано на рис. 19.5, первые четыре поля маршрутного заголовка фиксированы. Значение поля *типа маршрута* определяет тип предоставляемой маршрутной информации. Единственный определенный на данный момент тип *0* соответствует неточной маршрутизации от источника. В области *данных, зависящих от типа маршрута*, указывается список адресов маршрутизаторов, через которые должна пройти дейтаграмма. Общее количество оставшихся в списке адресов маршрутизаторов указывается в специальном фиксированном поле заголовка. Общая *длина дополнительного заголовка* указывается в одноименном фиксированном поле маршрутного заголовка.

Рис. 19.5. Формат маршрутного заголовка протокола IPv6. В настоящее время определен только тип 0 (неточная маршрутизация от источника)

**Параметры протокола IPv6**

Может показаться, что в протоколе IPv6 параметры протокола IPv4 полностью заменены дополнительными заголовками. Однако разработчиками были предложены еще два дополнительных заголовка, содержащие различную информацию, не вошедшую в другие дополнительные заголовки. Этими заголовками являются *переходной (Hop By Нор) дополнительный заголовок и сквозной (End То End) дополнительный заголовок*. Из их названий следует, что в одном из них содержатся параметры, которые должны анализироваться при каждом переходе, а в другом — те, которые обрабатываются в конечном пункте назначения.

Несмотря на то, что коды типов этих заголовков различаются, оба они имеют идентичный формат (рис. 19.6).

Рис. 19.6. Формат дополнительного заголовка, содержащий параметры, протокола IPv6. Формат обоих типов заголовков (переходного и сквозного) одинаков. Различить их можно только по значению поля типа следующего заголовка, содержащегося в предыдущем заголовке

Как обычно, тип следующего заголовка указывается в специальном поле, расположенном в начале дополнительного заголовка. Поскольку размер заголовка, содержащего параметры протокола, имеет переменную длину, то она указывается в специальном поле. Набор различных параметров протокола указывается в области данных дополнительного заголовка. На рис. 19.6 она обозначена как *Один или несколько параметров протокола*. На рис. 19.7 показано, как кодируется тип, длина и значение каждого отдельного параметра протокола1. Поля параметров никак не выравниваются и не дополняется пустыми символами.

Рис. 19.7. Формат кодирования отдельного параметра в дополнительном заголовке протокола IPv6. Каждый параметр состоит из 1-октетного поля типа и 1-октетного поля длины, за которыми следует ноль или более октетов данных

Как показано на рис. 19.7, параметры протокола IPv6 имеют такой же формат, как и параметры протокола IPv4. В начале каждого параметра располагается 1-октетное поле типа, за которым следует 1-октетное поле длины. Если для параметра предусмотрены дополнительные данные, они помещаются после поля длины.

Два старших бита поля типа каждого параметра определяют реакцию узла сети или маршрутизатора на получение дейтаграммы, содержащей неизвестный параметр (табл. 19.1).

Таблица 19.1. Значение двух старших битов поля типа параметра

| **Значение** | **Описание** |
| --- | --- |
| 00 | Пропустить этот параметр |
| 01 | Аннулировать дейтаграмму, не посылать ICMP-сообщение |
| 10 | Аннулировать дейтаграмму, послать ICMP-сообщение отправителю |
| 11 | Аннулировать дейтаграмму; послать ICMP-сообщение для немногоадресатного адреса |

Кроме того, от значения третьего бита поля типа зависит, будет ли параметр изменяться в процессе передачи дейтаграммы. Такая информация очень важна при аутентификации. Если содержимое параметра может изменяться в процессе передачи дейтаграммы, то при аутентификации значение ее полей полагаются равными нулю.

1. В литературе такой тип кодирования называют TLV-кодированием. *TLV* — это сокращение от *Type-Length-Value (Тип-Длина-Значение)*.

**Размер адресного пространства протокола IPv6**

В соответствии с протоколом IPv6, каждый адрес занимает 16 октетов, что в четыре раза больше размера адреса, определяемого протоколом IPv4. Такой гигантский размер адресного пространства позволяет создать практически любую рациональную схему распределения адресов. Действительно, если разработчики позднее решат изменить схему адресации, то при таком размере подобное перераспределение сделать будет достаточно легко.

Чтобы представить себе размер этого адресного пространства, попробуем сравнить его с численностью населения Земли. Адресное пространство, определяемое протоколом IPv6, настолько велико, что каждому человеку на планете можно выделить такое количество адресов, что он сможет организовать собственную глобальную сеть, по величине не уступающую современной Internet. Другой путь — соотнести количество адресов с площадью поверхности Земли, которая примерно равна 5,1×108 квадратных километров. Таким образом, на каждый квадратный метр земной поверхности приходится более 1024 адресов.

Еще один способ представить себе этот размер — оценить время, через которое может исчерпаться такое количество адресов. Рассмотрим, например, в течение какого времени можно будет распределить все возможные адреса. Известно, что 16-октетное число может принимать 2128 значений. Таким образом, размер адресного пространства превышает 3,4×1038 адресов. Если каждую микросекунду распределять один миллион адресов, то адресное пространство исчерпается через 1020 лет.

**Шестнадцатеричная запись с разделением двоеточием**

Применение адресов больших размеров решает проблему нехватки адресного пространства, но при этом возникает другая проблема — человеческий фактор. Дело в том, что люди, сопровождающие глобальные сети, должны анализировать, вводить и изменять адреса. Очевидно, что двоичная форма записи для этих целей непригодна. Запись же таких адресов в виде десятичных чисел, разделенных точками, которая использовалась в протоколе IPv4, также недостаточно компактна. Чтобы понять, почему, рассмотрим в качестве примера 128-битовое число, записанное в таком виде:

104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255

Чтобы сделать длинный адрес несколько компактнее и легче для ввода, разработчики протокола IPv6 предложили использовать шестнадцатеричную форму записи с разделением двоеточиями. Адрес представляется в виде последовательности 16-битовых чисел, представленных в шестнадцатеричной системе счисления и разделенных двоеточиями. Например, если значение адреса, записанное выше десятичными числами, перевести в шестнадцатеричную форму записи с разделением двоеточиями, то получится вот что:

68Е6:8С64:FFFF:FFFF:0:1180:96А:FFFF

Очевидное преимущество шестнадцатеричной формы записи по сравнению с десятичной состоит в том, что при этом используется меньше цифр и разделительных знаков. Кроме того, в шестнадцатеричной форме записи с разделением двоеточиями предусмотрены два приема, повышающие удобочитаемость адреса. Первый прием называется *сжатие нулей*. Суть его состоит в том, что ряд повторяющихся нулей можно заменить парой двоеточий. Например, адрес:

FF05:0:0:0:0:0:0:B3

можно записать как:

FF05::В3

В стандарте указывается, что к любому адресу процедура сжатия нулей может применяться только один раз. В противном случае возможна неоднозначная интерпретация адреса. Сжатие нулей особенно эффективно при использовании с предложенной в стандарте схемой распределения адресов, поскольку при этом во многих адресах будут содержаться длинные цепочки нулей. Второй прием — включение в шестнадцатеричную форму записи с разделением двоеточиями десятичных суффиксов, разделенных точками. Ниже мы покажем, что во время перехода от протокола IPv4 к IPv6 предполагается использование такой комбинированной формы записи. Например, следующая строка является корректной формой записи адреса протокола IPv6:

0:0:0:0:0:0:128.10.2.1

Обратите внимание, что каждое из чисел, разделенных двоеточием, является 16-битовой величиной, тогда как разделяемые точкой десятичные числа представляют величину длиной в один октет. К вышеуказанному числу, конечно же, можно применить процедуру сжатия нулей, в результате чего получится эквивалентная строка, которая будет очень похожа на запись обычного адреса протокола IPv4:

::128.10.2.1

И наконец, протокол IPv6 позволяет применять для представления адреса расширенную форму записи CIDR-адресов. Другими словами, за строкой адреса через косую черту указывается десятичное число, которое определяет количество бит, составляющих адрес. Например, запись

12АВ::CD30:0:0:0:0/60

определяет первые 60 бит адреса, или в шестнадцатеричной системе

12AB00000000CD3

**Три основных типа адресов протокола IPv6**

Как и в протоколе IPv4, в IPv6 адрес соответствует определенному сетевому соединению, а не компьютеру. Поэтому процесс назначения адресов в протоколе IPv6 такой же, как и в IPv4. Маршрутизатору выделяется два или более адреса IPv6 (по количеству его сетевых интерфейсов), а узлу с одним сетевым соединением необходим только один адрес IPv6. В IPv6 также сохраняется (и расширяется) иерархия адресов IPv4, в которой часть адреса (префикс) идентифицирует физическую сеть. Однако, чтобы сделать распределение и изменение адресов более простым, в протоколе IPv6 можно назначить для одной сети несколько префиксов, а один компьютер может иметь несколько различных адресов, соответствующих одному и тому же интерфейсу. Кроме того, в протоколе IPv6 расширены, а в некоторых случаях и унифицированы, специальные адреса протокола IPv4. В общем случае адрес получателя дейтаграммы можно отнести к одной из трех категорий (табл. 19.2).

Таблица 19.2. Три типа адресов протокола IPv6

| **Тип адреса** | **Описание** |
| --- | --- |
| Одноадресатный (unicast) | Определяет одного получателя в сети (компьютер или маршрутизатор). Дейтаграмма должна передаваться к месту назначения по кратчайшему маршруту |
| Альтернативный (anycast) | Определяет группу компьютеров, возможно, расположенных в разных местах объединенной сети, которым назначен один общий адрес. Дейтаграмма должна передаваться по кратчайшему маршруту только одному (ближайшему) члену группы1 |
| Многоадресатный (multicast) | Определяет группу компьютеров, возможно, расположенных в различных местах объединенной сети. Дейтаграмма, посланная по многоадресатному адресу, будет доставлена каждому члену группы. При этом используется многоадресатный или широковещательный режим передачи сетевого оборудования (если, конечно, он поддерживается) |

1. Альтернативные адреса ранее назывались *кластерными (cluster)* адресами.

**Взаимозаменяемость широковещательной и многоадресатной передачи**

В протоколе IPv6 не используются термины *широковещательная (broadcast)* и *направленная широковещательная (directed broadcast)* передача по отношению к передаче информации всем компьютерам физической сети или логической IP-подсети. Вместо этого используется термин *многоадресатная (multicast)* передача, а широковещательная передача рассматривается как частный случай многоадресатной передачи. Такой подход может показаться странным каждому, кто хоть немного знаком с принципами работы сетевого оборудования, поскольку режим широковещательной передачи поддерживается на уровне оборудования чаще, чем многоадресатной. На самом деле разработчики оборудования обычно рассматривают многоадресатную передачу как ограниченную форму широковещательной передачи. На аппаратном уровне многоадресатный пакет рассылается всем компьютерам локальной сети точно так же, как и широковещательный пакет. После этого все многоадресатные пакеты, кроме тех, о которых со стороны программного обеспечения компьютера поступило указание принять, отфильтровываются интерфейсным оборудованием сетевой платы.

Теоретически выбор между многоадресатной передачей и ограниченными формами широковещательной передачи не принципиален, поскольку одна может эмулироваться с помощью другой. Другими словами, они в некоторой степени дублируют друг друга, обеспечивая одинаковые функциональные возможности. Чтобы понять это, рассмотрим эмуляцию одного режима передачи с помощью другого. Если поддерживается режим широковещательной передачи, то пакет может быть доставлен группе получателей следующим образом. Он посылается всем машинам, на каждой из которых программным обеспечением принимается решение о том, принять или аннулировать пришедший пакет. Когда поддерживается многоадресатная передача, пакет может быть доставлен всем машинам, если предварительно объединить их в одну многоадресатную группу (группу всех компьютеров данной подсети).

**Проблема выбора, стоящая перед разработчиками**

То, что широковещательный и многоадресатный режимы передачи теоретически взаимозаменяемы, никак не облегчает разработчикам задачу выбора одного из них. Чтобы стало понятно, почему разработчики протокола IPv6 в качестве основной модели выбрали режим многоадресатной, а не широковещательной передачи, рассмотрим эту проблему с точки зрения создания приложений, а не на уровне используемого оборудования. Приложению необходимо обмениваться данными либо с другим приложением, либо с группой приложений. Непосредственный обмен данными с одним приложением наилучшим образом обеспечивается посредством одноадресатной передачи, а с группой приложений — с помощью широковещательной или многоадресатной передачи. Для обеспечения максимальной универсальности принадлежность к группе не должна определяться по адресу сетевого интерфейса, поскольку компьютеры, входящие в группу, могут располагаться в любом месте объединенной сети. Режим широковещательной передачи нельзя использовать для обмена данными со всеми компьютерами группы, которые разбросаны по глобальной сети, особенно когда размер сети сопоставим с размером современной сети Internet.

Поэтому неудивительно, что разработчики заранее зарезервировали некоторые многоадресатные адреса для использования вместо сетевых широковещательных адресов протокола IPv4. Таким образом, от каждого маршрутизатора требуется, чтобы он кроме собственного адреса принимал все дейтаграммы, адресованные локальной многоадресатной группе, в которую входят *все маршрутизаторы*.

**Проект распределения адресного пространства протокола IPv6**

Вопрос о том, как разделить на части адресное пространство протокола IPv6, вызвал много дискуссий. Существует два основных спорных вопроса: как управлять распределением адресов и как определять по адресу маршрут. При обсуждении первого вопроса внимание уделяется практической проблеме планирования иерархии административных единиц. В отличие от современной сети Internet, в которой используется двухуровневая иерархия (сетевые префиксы распределяются центральным административным органом Internet, а суффиксы сетевых адресов назначаются локальными организациями), огромное адресное пространство протокола IPv6 подразумевает создание одной или нескольких многоуровневых иерархий. При обсуждении второго вопроса основное внимание уделяется эффективности вычислительных алгоритмов. Дело в том, что маршрутизаторы должны анализировать каждую проходящую через них дейтаграмму и выбирать для нее кратчайший маршрут к получателю независимо от того, к какой из иерархий административных органов принадлежит этот адрес. Чтобы снизить стоимость высокоскоростных маршрутизаторов, необходимо оптимизировать алгоритм выбора кратчайшего пути. Тогда время обработки каждой дейтаграммы в маршрутизаторе будет минимальным.

Разработчики протокола IPv6 предложили разбить все адресное пространство на классы, по аналогии со схемой, используемой в протоколе IPv4 (табл. 19.3). Как и в протоколе IPv4 тип адреса определяется по его префиксу. Следует отметить, что, хотя первых восьми битов адреса вполне хватило бы для определения его типа, разработчики не стали разбивать адресное пространство на блоки одинакового размера.

Таблица 19.3. Проект распределения адресного пространства протокола IPv6

| **Двоичный префикс** | **Тип адреса** | **Часть адресного пространства** |
| --- | --- | --- |
| 0000 0000 | Зарезервировано (для совместимости с IPv4) | 1/256 |
| 0000 0001 | Не распределено | 1/256 |
| 0000 001 | Адреса NSAP | 1/128 |
| 0000 010 | Адреса IPX | 1/128 |
| 0000 011 | Не распределено | 1/128 |
| 0000 1 | Не распределено | 1/32 |
| 0001 | Не распределено | 1/16 |
| 001 | Сгруппированные глобальные одноадресатные адреса | 1/8 |
| 010 | Не распределено | 1/8 |
| 011 | Не распределено | 1/8 |
| 100 | Не распределено | 1/8 |
| 101 | Не распределено | 1/8 |
| 110 | Не распределено | 1/8 |
| 1110 | Не распределено | 1/16 |
| 1111 0 | Не распределено | 1/32 |
| 1111 10 | Не распределено | 1/64 |
| 1111 110 | Не распределено | 1/128 |
| 1111 1110 0 | Не распределено | 1/512 |
| 1111 1110 10 | Одноадресатные адреса, локальные в пределах канала передачи данных | 1/1024 |
| 1111 1110 11 | Одноадресатные адреса, локальные в пределах сетевого центра | 1/1024 |
| 1111 1111 | Многоадресатные адреса | 1/256 |

Как видно из табл. 19.3, на сегодняшний день распределено только 15% адресного пространства. Оставшаяся часть будет задействоваться IEFT по мере необходимости. Несмотря на кажущуюся разбросанность, при распределении блоков адресов учитывалась возможность ускорения их обработки в маршрутизаторе. Например, многоадресатный адрес легко отличить от одноадресатного адреса по значению старшего октета. В многоадресатном адресе все биты старшего октета равны единице, тогда как в одноадресатном адресе в старшем октете содержится смесь нулей и единиц.

**Кодирование адресов протокола IPv4 и переход к адресам IPv6**

Хотя в табл. 19.3 против префикса 0000 0000 отмечено, что он *зарезервирован*, небольшую часть адресов из этого блока разработчики планируют использовать для кодирования адресов, определенных протоколом IPv4. В частности, любой адрес, начинающийся с 80 нулевых бит, за которыми следуют 16 бит из единиц или 16 бит из нулей, содержит адрес IPv4 в младших 32 битах. Значение соответствующего 16-битового поля показывает, есть ли у данного узла еще и одноадресатный адрес протокола IPv6. На рис. 19.8 показаны оба возможных варианта.

Рис. 19.8. Кодирование адресов IPv4 в протоколе IPv6. В 16-битовом поле содержится 0000, если узел имеет также одноадресатный адрес IPv6, и FFFF — в противном случае

Описанное выше кодирование будет использоваться при переходе от протокола IPv4 к IPv6 по двум причинам.

Во-первых, программное обеспечение компьютера, использующее протокол IPv4, может быть обновлено на программное обеспечение, использующее протокол IPv6, еще до того, как машина получит адрес IPv6.

Во-вторых, у компьютера с действующим программным обеспечением на основе протокола IPv6 может возникнуть необходимость связаться с компьютером, на котором используется только программное обеспечение на основе протокола IPv4.

Кодирование адресов IPv4 в адреса протокола IPv6 не решает проблему взаимодействия программного обеспечения, поддерживающего разные версии протокола, необходим еще механизм их преобразования. При использовании транслятора адресов компьютер, использующий протокол IPv6, должен сгенерировать дейтаграмму, в которой содержится закодированный IPv4 адрес получателя. Затем компьютер посылает дейтаграмму транслятору адресов, который, используя протокол IPv4, отправляет ее получателю. Получив ответ, транслятор адресов преобразовывает дейтаграмму протокола IPv4 в дейтаграмму протокола IPv6 и отсылает ее отправителю с помощью протокола IPv6.

На первый взгляд может показаться, что преобразование протокольных адресов будет приводить к некорректной работе системы, поскольку протоколы более высокого уровня проверяют неизменность адреса. В частности, в протоколах TCP и UDP при вычислении контрольной суммы используется *псевдозаголовок*. В нем содержатся адреса как отправителя, так и получателя, поэтому изменение этих адресов может повлиять на вычисление контрольной суммы. Поэтому разработчики должны были предусмотреть возможность взаимодействия протоколов TCP и UDP на машинах, использующих IPv4, с соответствующими транспортными протоколами на машинах, использующими IPv6. Чтобы избежать несовпадения контрольных сумм, при кодировании адресов протокола IPv4 в протоколе IPv6 алгоритм вычисления контрольной суммы выбирается так, чтобы 16-битовое дополнение контрольной суммы до единицы было одинаково для обоих типов адресов. Выделяя главное, можно сказать вот что.

Кроме выбора технических деталей нового протокола Internet, работа проблемной группы. IEFT над протоколом IPv6 была сосредоточена на поиске способа перехода от текущего протокола к новому. В частности, в текущем варианте протокола IPv6 предусмотрена возможность кодирования адреса протокола IPv4 в адрес IPv6 таким образом, что преобразование адреса не изменяет контрольной суммы, псевдозаголовка.

**Неопределенные адреса и адреса петли обратной связи**

Как и в протоколе IPv4, некоторым адресам протокола IPv6 придается специальное значение. Например, адрес, состоящий из одних 0:

0:0:0:0:0:0:0:0

является неопределенным. Его нельзя назначить ни одному компьютеру или использовать в качестве адреса получателя. Он используется как адрес отправителя только во время начальной загрузки компьютера, которому еще не назначен реальный адрес.

Подобно протоколу IPv4, у IPv6 есть *адрес петли обратной связи*, который используется для тестирования программного обеспечения. В протоколе IPv6 таким адресом является

0:0:0:0:0:0:0:1

Любая дейтаграмма, посланная по адресу петли обратной связи, не покидает пределов локальной машины. Этот адрес никогда не следует использовать как адрес получателя в исходящей дейтаграмме.

**Иерархия одноадресатных адресов**

Одним из наиболее важных изменений в протоколе IPv6, по сравнению с IPv4, является стратегия распределения одноадресатный (unicast) адресов и полученная в результате иерархия адресов. Напомним, что в оригинальной системе адресации протокола IPv4 использовалась двухуровневая иерархия, согласно которой адрес делился на глобально уникальный префикс и суффикс. В протоколе IPv6 данная концепция расширена за счет принятия иерархии адресов с тремя концептуальными уровнями, как показано в табл. 19.4. На практике эта иерархическая структура адреса содержит дополнительные уровни.

Таблица 19.4. Три концептуальных уровня иерархии одноадресатных адресов протокола IPv6

| **Уровень** | **Назначение** |
| --- | --- |
| 1 | Глобальная открытая топология |
| 2 | Отдельный сетевой центр |
| 3 | Отдельный сетевой интерфейс |

Два нижних уровня концептуальной иерархии наиболее легки для понимания, поскольку они соответствуют реально существующим объектам. Самому нижнему уровню иерархии соответствует одиночное соединение между компьютером и сетью. Среднему уровню иерархии соответствует несколько компьютеров и сетей, объединенных в один *сетевой центр (site)*, подразумевающий как тесную физическую взаимосвязь, так и организацию, которая владеет и управляет оборудованием. Как будет показано ниже, такую схему адресации используют как большие, так и малые сетевые центры, поскольку она подходит для сложной внутренней структуры.

Для обеспечения гибкости верхний уровень иерархии, обозначенный в табл. 19.4 как *глобальная открытая топология*, точно не определяется. В общем, его можно понимать как “сегмент” глобальной сети Internet, открытый для общего доступа. Вырисовываются два типа открытой топологии.

Один из них соответствует крупному провайдеру Internet, который предоставляет своим потребителям, называемым *абонентами*, услуги по передаче данных на большие расстояния.

Второй тип соответствует недавно образованной структуре, называемый *коммутатором (exchange)*. По замыслу разработчиков, коммутаторами будут выполняться две функции: выступать в качестве точки доступа к сети (NAP), т.е. связывать крупных провайдеров Internet и обеспечивать передачу трафика между ними, и обслуживать отдельных абонентов (в отличие от современных NAP, обслуживающих только крупных провайдеров). Это означает, что коммутаторами будут выделяться адреса абонентам. Главное преимущество выделения адресов коммутатором в том, что такие адреса не зависят от провайдера Internet, т.е. абоненты смогут свободно менять провайдеров.

**Структура сгруппированных глобальных одноадресатных адресов**

Права на распределение адресов в протоколе IPv6 плавно переходят от одного уровня к другому вниз по иерархии. Каждой организации верхнего уровня (например, провайдеру Internet или коммутатору) выделяется уникальный префикс. Когда некоторая организация становится абонентом провайдера верхнего уровня, ее сетевому центру присваивается уникальный номер. В конце концов, сетевой администратор должен назначить этот номер каждому сетевому соединению. Чтобы добиться эффективной маршрутизации, для каждого назначения в адресе резервируются последовательные наборы битов. На рис. 19.9 показан формат *сгруппированных глобальных одноадресатных адресов (aggregatable global unicast address)*.

Рис. 19.9. Разделение сгруппированного глобального одноадресатного адреса, соответствующего протоколу IPv6, на поля с указанием их соответствия трехуровневой модели иерархии

Трехбитовое поле, обозначенное на рис. 19.9 как Р, соответствует префиксу формата, значение которого для сгруппированного глобального одноадресатного адреса равно 001. Восьмибитовое поле RES зарезервировано для дальнейшего использования, и в нем содержатся нули. Оставшиеся поля адреса предназначаются для обеспечения эффективной маршрутизации. В частности, поля, соответствующие самому высокому уровню иерархии, сгруппированы, и в них включены старшие биты адреса. В поле TLA ID содержится идентификатор *группировки верхнего уровня (Top-Level Aggregation)*, т.е. уникальный идентификатор, назначаемый владельцам адреса (т.е. провайдеру Internet или коммутатору). Владелец адреса использует поле NLA для обозначения *группировки следующего уровня (Next-Level Aggregation)*, т.е. для идентификации отдельного абонента.

Шестнадцатибитовое поле SLA ID *(Site-Level Aggregation*, или *группировка на уровне сетевого центра)* предназначено для использования отдельным сетевым центром. Разработчики предполагают, что оно будет использоваться подобно полю подсети, предусмотренному протоколом IPv4. Таким образом, сетевой центр, состоящий из небольшого количества сетей, может трактовать значение этого поля как идентификатор сети. Сетевой центр с большим количеством сетей может использовать его для разбиения сетей на группы, которые затем могут быть организованы в иерархию. Для создания одноуровневой иерархии в сетевом центре организация должна использовать префикс для идентификации группы, и суффикс — для идентификации отдельной сети в этой группе. Как и в случае организации подсетей протокола IPv4, деление на группы улучшает эффективность маршрутизации, поскольку в таблице маршрутизации указываются только маршруты к другим группам, а не к индивидуальным сетям.

**Идентификаторы интерфейса**

Как показано на рис. 19.9, младшие 64 бита сгруппированного одноадресатного адреса протокола IPv6 идентифицируют конкретный сетевой интерфейс. В отличие от протокола IPv4, суффикс в IPv6 выбран достаточно большим, чтобы можно было непосредственно закодировать в протокольном адресе физический адрес платы сетевого интерфейса. Кодирование физического адреса в протокольном адресе IPv6 имеет два преимущества. Во-первых, в протоколе IPv6 для преобразования IP-адреса в физический адрес больше не нужно использовать протокол ARP. Вместо него IPv6 предусматривает использование *протокола обнаружения соседних узлов (neighbor discovery protocol)*, который входит в новую версию протокола ICMP (*ICMPvd*). Он позволяет узлу сети определить, какие компьютеры подключены вместе с ним к одной физической сети. Во-вторых, для обеспечения взаимодействия всеми компьютерами должен использоваться один и тот же алгоритм кодирования физических адресов. Следовательно, в стандарте протокола IPv6 должно быть четко указано, как кодируются различные формы физических адресов. В простейшем случае физический адрес непосредственно размещается в адресе протокола IPv6. В отдельных случаях для кодирования адреса используются более сложные алгоритмы.

Ниже приведены два примера кодирования физических адресов, которые помогут понять принцип. Например, Институтом инженеров по электротехнике и электронике (ШЕЕ) определен стандартный 64-битовый формат глобально уникального адреса, называемый *EUI-64*. Единственное изменение, которое необходимо выполнить при кодировании адреса стандарта ЕUI-64 в адрес протокола IPv6 состоит в изменении 6-го бита в старшем октете адреса. Он показывает, является ли адрес глобально уникальным.

Для обычного 48-битового адреса Ethernet необходимы более сложные преобразования (рис. 19.10). Как видно из рисунка, биты исходного физического адреса в закодированной форме не располагаются рядом. Вместо этого посередине результирующего адреса помещается 16-битовое шестнадцатеричное значение 0xFFFE. Кроме того, 6-ой бит, который указывает, является ли адрес глобальным, изменен с 0 на 1. Оставшиеся биты адреса, включая бит группы (обозначенный д), идентификатор компании — производителя интерфейса (обозначенный с) и поле расширения производителя, копируются, как показано на рис. 33.10. Расширение, выбранное производителем, для однозначной идентификации устройства помещается в младшие 24 бита адреса.

Рис. 19.10. Формат 48-битового адреса стандарта IEEE 802, используемого в сети Ethernet (а); кодирование этого адреса в младших 64-х битах одноадресатного адреса в соответствии с требованиями протокола IPv6 (б)

**Дополнительный уровень иерархии**

Несмотря на то, что в формате одноадресатного адреса, показанного на рис. 19.9, уже заложена строгая иерархия адресного пространства, в ней можно создать много дополнительных уровней. Например, биты поля NLA ID могут использоваться для создания иерархии провайдеров. Аналогично, 16 бит поля SLA ID можно разделить на части для создания иерархии в пределах организации. Поскольку эти поля имеют достаточно большой размер, то по сравнению с механизмом подсетей протокола IPv4, обеспечивается большая универсальность.

Организация может принять решение создать двухуровневую иерархию, состоящую из зон и подсетей, входящих в каждую зону. В другом случае организация может выбрать трехуровневую иерархию, состоящую из зон, подзон и подсетей, образующих подзоны.

**Локальные адреса**

Кроме глобальных одноадресатных адресов, описанных выше, в протоколе IPv6 предусмотрены дополнительные префиксы для одноадресатных адресов, имеющих локальную область действия. Как показано в табл. 19.3, в стандарте определены два типа локальных адресов: *по отношению к каналу передачи данных* (т.е. ограниченные одной физической сетью) и *по отношению к сетевому центру* (т.е. ограниченные одним сетевым центром). Область действия этих адресов учитывается маршрутизаторами: дейтаграммы, содержащие локальные адреса, не передаются ими за пределы указанной области.

Использование локальных адресов способствует решению двух проблем. Адреса, локальные по отношению к каналу передачи данных, обеспечивают взаимодействие машин в одной физической сети. Тем самым устраняется опасность передачи дейтаграммы через внешнюю объединенную сеть. Например, при обнаружении соседних узлов маршрутизатор, поддерживающий протокол IPv6, должен использовать локальный адрес по отношению к каналу связи. Правила области действия адресов определяют, что сообщения протокола определения соседних узлов будут получать только компьютеры, находящиеся в одной сети с отправителем. Точно так же локальные по отношению к каналу связи адреса могут использоваться для связи компьютеров, подключенных к *изолированной сети* (т.е. сети, к которой не подключены маршрутизаторы).

В отличие от дейтаграммы, содержащей локальный по отношению к каналу связи адрес, маршрутизаторы могут передавать дейтаграммы, содержащие локальные по отношению к сетевому центру адреса, по всем сетям организации. Однако маршрутизаторы препятствуют передаче таких дейтограмм в глобальную сеть Internet. Таким образом, эти адреса соответствуют тому, что в протоколе IPv4 называется *частными*, или *немаршрутизируемыми* адресами. Организация может назначать и использовать локальные по отношению к сетевому центру адреса в пределах всех своих внутренних сетей без официального получения глобально уникального префикса.

**Автоконфигурация и перенумерация сетей**

При разработке протокола IPv6 был предусмотрен режим автоконфигурации, не требующий сервера1. Благодаря этому сетевым администраторам нет необходимости предварительно назначать адреса компьютерам для того, чтобы они могли начать взаимодействовать. Режим автоконфигурации оказался возможным и эффективным благодаря описанным выше двум механизмам: локальной адресации относительно канала связи и кодированию физических адресов в протокольном адресе IPv6. Перед посылкой первого пакета в сеть компьютер генерирует свой локальный относительно канала связи адрес путем комбинирования локального префикса 1111 1110 10 с 54-мя нулевыми битами и 64-битовым идентификатором своей платы сетевого интерфейса.

После проверки локального адреса на уникальность он используется компьютером для отправки специального пакета маршрутизатору, в котором запрашивается дополнительная информация. Если к данной физической сети подключен маршрутизатор, то в ответ на запрос он пришлет *извещение о маршрутизаторе*. В этом сообщении указываются префиксы, которые могут быть использованы в локальных относительно сетевого центра и глобальных адресах. После получения извещения о маршрутизаторе, компьютер назначает его отправителя своим стандартным маршрутизатором. Наконец, в извещении есть специальный флажок, указывающий на то, может ли компьютер воспользоваться полученными данными для выполнения автоконфигурации, либо он должен продолжить настройку своих параметров с помощью обычного режима *управляемого конфигурирования* (т.е. протокола DHCP).

Для упрощения процесса перенумерации сетей в протоколе IPv6 маршрутизатор может ограничить время использования префикса компьютером. Поэтому в извещении для каждого префикса указываются два значения: действительное и предпочтительное время жизни. При этом узел сети должен анализировать дополнительные сообщения об извещении маршрутизатора. По истечении предпочтительного времени жизни префикса он остается работоспособным, но узел по возможности должен воспользоваться другим префиксом для продолжения сеанса связи. По истечении действительного времени жизни префикса узел сети должен немедленно прекратить его использование, даже если в этот момент происходит обмен данными с другими компьютерами.

1. Такой режим также называется автоконфигурацией без учета состояния.

**Проверка знаний: Будущее протокола TCP/IP (IPv6)**

**Задание 1**

Какой протокол использует IPv6 вместо ARP?

simple IP

обнаружение соседних узлов

протокол EUI-64

**Задание 2**

Для чего были придуманы дополнительные заголовки дейтаграммы протокола IPv6?

для безопасности

для универсальности

для повышения эффективности

**Задание 3**

Исключите лишний тип адресов протокола IPv6?

одноадресатный

многоадресатный

многопользовательский

альтернативный

**Задание 4**

Где выполняется сборка фрагментированной дейтаграммы для протокола IPv6?

на машине конечного получателя

на машине отправителя

в процессе маршрутизации

**Задание 5**

Какая версия преобразования протокола IP, легла в основу его будущей версии?

предполагалось сделать протокол IP более гибким за счет увеличения его сложности и увеличения накладных расходов на обработку дейтаграмм

модификация протокола OSI CLNS

SIP (Simple IP)

**Задание 6**

Почему пересмотренной версии протокола IP присвоили номер 6 (IPv6)?

что бы избежать путаниц

группа IETF использовала особую закономерность при выборе номера

присвоили следующий по порядку номер

**Задание 7**

Использование какого адреса обеспечивает взаимодействие машин в одной физической сети, тем самым устраняя опасность передачи дейтаграммы через внешнюю объединенную сеть?

локального

виртуального

IPv6-адреса

**Задание 8**

Каждой организации верхнего уровня выделяется...

уникальный суффикс

уникальный адрес

уникальный префикс

**Задание 9**

С какого размера основного заголовка начинается каждая дейтаграмма?

36 октетного

40 октетного

56 октетного

**Задание 10**

Какой дополнительный заголовок, содержащий различную информацию, не вошедшую в другие дополнительные заголовки, был предложен разработчиками?

переходный и сквозной

переходный

сквозной

**Задание 11**

По сравнению с IPv4 (размер адреса которого 32 бита), во сколько раз была увеличен размер адреса в протоколе IPv6?

в 2 раза

в 4 раза

в 6 раз

**Задание 12**

Сколько раз может применятся процедура сжатия нулей?

1

неограниченное количество

8

**Практическая часть: настройки протокола IP на маршрутизаторе Cisco**

В данном разделе, показано, как применить полученные ранее знания на практике. За основу нами взят маршрутизатор Cisco. Ниже приведены основные настройки и конфигурация протокола IP а так же рассмотрены основные вопросы безопасности и управления доступом на маршрутизаторах Cisco.

* [Настройка протокола IP на маршрутизаторе Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068a)
* [IP и интерфейсы Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068b)
* [ICMP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068c)
* [Использование утилит ICMP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068d)
* [Ping](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068e)
* [traceroute](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69068f)
* [Telnet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690690)
* [Удаленное администрирование с использованием Telnet](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690691)
* [rlogin](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690692)
* [Резюме](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690693)
* [Основы обеспечения безопасности Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69069d)
* [Списки доступа IP](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69069e)
* [NAT](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a69069f)
* [Проверка знаний: Настройка протокола IP на маршрутизаторе Cisco](https://it.rfei.ru/course/%7EGZqN/bundle#item-540712a41001183c7a690694)

**Настройка протокола IP на маршрутизаторе Cisco**

В предыдущих разделах мы обсуждали один из наиболее популярных сегодня протоколов в маршрутизации — IP. Но существует также ряд других протоколов и инструментальных средств, непосредственно связанных с IP, о которых еще не говорилось.

Стек протоколов TCP/IP включает в себя не только те два протокола, которые дали ему имя (TCP и IP). В этом разделе будут рассмотрены два таких дополнительных средства: ICMP (Internet Control Message Protocol — протокол управления сообщениями в Интернете) и Telnet. После знакомства с ними и изучения команд конфигурирования IP на маршрутизаторах Cisco можно будет считать, что вы полностью освоили использование и работу IP. Чтобы достичь этой цели, мы рассмотрим такие темы:

* IP и интерфейсы Cisco
* ICMP
* Использование утилит ICMP
* Telnet
* Удаленное администрирование с использованием Telnet
* riogin

ICMP — это протокол, тесно связанный с IP. ICMP является мощным инструментом диагностики в мире маршрутизации. Освоение ICMP и его утилит поможет вам обнаружить большую часть проблем, которые могут случиться с маршрутизаторами Cisco, например таких, как ошибки при соединении типа «точка-точка». Часть этого раздела посвящена непосредственно ICMP.

Еще один широко используемый протокол, который также напрямую связан с IP, — это Telnet. Если вы имеете опыт работы с компьютерами, Telnet может быть вам знаком. Маршрутизаторы Cisco используют этот протокол в двух целях.

Во-первых, ПК может использовать Telnet для взаимодействия с маршрутизатором Cisco, чтобы получить доступ к командной строке.

Второе (и наиболее важное) предназначение Telnet заключается в том, что он предоставляет маршрутизаторам Cisco возможность удаленно администрировать другие маршрутизаторы Cisco. То есть, используя Telnet, один маршрутизатор Cisco может удаленно администрировать несколько других маршрутизаторов Cisco.

Возможность администрировать несколько серверов, не меняя при этом местоположения, становится бесценной при работе в больших и очень больших средах маршрутизации.

Этот раздел завершает наш разговор об IP. Подразделы — это кусочки той большой картины, которой является стек протоколов TCP/IP. IP – это, безусловно, самый важный протокол в современной маршрутизации, и осмысление каждого аспекта его работы необходимо для понимания внутренних механизмов маршрутизации. Когда вы изучите раздел до конца, то будете обладать хорошим запасом базовых знаний.

Теперь, когда все вводные слова сказаны, давайте перейдем к конфигурированию маршрутизатора для работы с IP. Затем мы сможем погрузиться в изучение связанных с ним программ и их использование.

**IP и интерфейсы Cisco**

IP как протокол уже должен быть установлен на вашем маршрутизаторе. Все функциональные пакеты Cisco IOS, кроме IPX Basic, содержат IP. Поэтому для того, чтобы IP заработал, не требуется устанавливать никакого программного обеспечения. В этом разделе мы поговорим об использовании возможностей IP.

Все настройки IP производятся в режиме конфигурирования интерфейсов. Чтобы войти в режим конфигурирования интерфейсов маршрутизатора Cisco, выполните такие команды:

Router>\*enable\*

Router#\*configure terminal\*

Router(config)#\*interface ethernet 0\*

Router(config-if)#

Команда enable активирует привилегированный режим работы командного интерпретатора Exec. В нем мы выполняем команду configure terminal. Маршрутизатор переходит в режим глобального конфигурирования.

Используем команду interface ethernet 0 для перехода от глобального конфигурирования к конфигурированию интерфейсов (будем настраивать интерфейс Ethernet 0). В режиме конфигурирования интерфейсов достаточно одной простой команды, и IP уже может использоваться на вашем маршрутизаторе:

Router(config-if)#\*ip address 10.156.4.16 255.255.0.0\*

Команда ip имеет ряд параметров. Ключевое слово address указывает, что будет введен IP-адрес для интерфейса и маска подсети. Такая команда разрешает работу IP для одного интерфейса маршрутизатора. Повторите ее для всех физических интерфейсов, которые должны работать с IP.

Когда интерфейсы настроены, вы можете включить маршрутизацию, опять-таки, посредством одной простой команды:

Router(config)#\*ip routing\*

Команда ip routing включает маршрутизацию IP-пакетов между всеми сконфигурированными интерфейсами. Может также возникнуть необходимость разрешить бесклассовую маршрутизацию IP с помощью отдельной команды.

Если вы знаете, что в вашей сети будет использоваться бесклассовый IP, то следует разрешить маршрутизатору работу с ним. (Бесклассовый IP обсуждался в разделе [Введение в маршрутизируемые протоколы](https://it.rfei.ru/course/%7EGZqN/-/%7EwO5W) и в разделе [Изучение основ IP](https://it.rfei.ru/course/%7EGZqN/-/%7EBT9K)). Команда, разрешающая маршрутизатору работать в бесклассовом режиме, выглядит так:

Router(config)#\*ip classless\*

Заметьте, что команда ip classless выполняется в режиме глобального конфигурирования, так как возможность бесклассовой маршрутизации IP затрагивает все интерфейсы.

Когда маршрутизатор получает пакет, он пытается установить соответствующую сеть по своей таблице маршрутов. Если в таблице для пункта назначения пакета не найдено соответствия, он пересылается в надсеть. Например, если у маршрутизатора существуют определенные маршруты для сетей 128.46.69.0 и 128.45.77.0 и поступает пакет с адресом назначения 128.46.68.15, то он будет перенаправлен в сеть 128.46.68.0, так как 128.46.x.x — это надсеть наиболее близкого соответствия.

Наконец, если ваш маршрутизатор не является маршрутизатором «последней надежды» сети, то следует его задать. То есть конфигурируемый маршрутизатор должен знать, что ему делать с пакетами, для которых не определены маршруты. Чаще всего такие пакеты отправляют в другую сеть, для чего необходимо использовать шлюз. Поэтому маршрутизатор последней надежды также может быть шлюзом по умолчанию.

Маршрутизатор последней надежды — это тот, на который все остальные маршрутизаторы сети пересылают пакеты, для которых не заданы маршруты. Такой маршрутизатор чаще всего является единственным, имеющим прямое соединение с Интернетом.

Можно с уверенностью предположить, что в большинстве случаев, когда на маршрутизатор приходит пакет, для которого нет предопределенного маршрута, его следует переслать в Интернет. Поэтому маршрутизатор, соединяющий вашу сеть с Интернетом, должен быть настроен как маршрутизатор последней надежды для всех остальных устройств сети.

Чтобы установить адрес шлюза по умолчанию, используйте команду ip default-gateway:

Router(config)#\*ip default-gateway 198.2.65.1\*

Отметьте, что и эта команда выполняется в режиме глобального конфигурирования, следовательно, будет применена ко всем интерфейсам. В заключение используйте команду ip route для определения статических путей IP:

Router(config)#\*ip route 198.52.2.0 255.0.0.0 Ethernet 0 100 perm\*

Команда ip route определяет для маршрутизатора прямой путь к указанной сети. Устанавливая статические пути, вы можете управлять потоком данных вне вашего маршрутизатора. Но команда ip route требует указания нескольких параметров. Давайте немного поговорим о них.

За ip route должно следовать определение статического пути, то есть должны быть указаны IP-адрес сети и маска подсети.

Так как эти настройки производятся в режиме глобального конфигурирования, необходимо указать, для какого интерфейса маршрутизатора определяется путь. Поэтому следующий параметр задает физический интерфейс, на который будут пересылаться все пакеты, соответствующие данному статическому пути: Ethernet 0.

Следующий параметр выбирается в некоторой степени случайно, и, если хотите, то можете его не включать. Этот параметр, 100, определяет административную метрику маршрута. Административная метрика используется в тех случаях, когда для некоторой сети определено несколько путей. Тогда используется путь с наименьшей метрикой, а другие становятся резервными.

Последний параметр, perm, определяет весь статический путь как постоянный (permanent). Поэтому он будет сохранен в конфигурационном файле startup-config. Чтобы определить временные пути, не применяйте параметр perm. Временные пути не сохраняются в конфигурационном файле, и после перезагрузки маршрутизатор о них забывает.

Давайте проработаем несколько сценариев IP-маршрутизации. Например, как следует сконфигурировать маршрутизатор сети, изображенной на рис. 20.1, чтобы он использовал IP и маршрутизировал его между сетями A и B?

Рис. 20.1. Пример IP-сети

Чтобы разрешить маршрутизацию, необходимо выполнить такие команды:

Router>\*enable\*

Router#\*configure terminal\*

Router(config)#\*interface ethernet 0\*

Router(config-if)#\*ip address 10.13.50.1 255.255.255.0\*

Router(config-if)#\*~Z\*

Router#\*configure terminal\*

Router(config)#\*interface ethernet 1\*

Router(config-if)#\*ip address 10.13.60.1 255.255.255.0\*

Router(config-if)#\*~Z\*

Router#\*configure terminal\*

Router(config)#\*ip routing\*

Router(config)#\*ip route 10.13.50.0 255.255.255.0 e0 perm\*

Router(config)#ip route 10.13.60.0 255.255.255.0 e1 perm\*

Хотя такой план действий может показаться слишком простым с практической точки зрения, по существу, именно он лежит в основе функционирования любой сети.

Однако для того чтобы обладать всеми базовыми знаниями, необходимыми для конфигурирования IP на маршрутизаторах Cisco, вам не помешает знать о существовании еще одного сценария. Следующий пример иллюстрирует часто встречающийся вариант сетевой среды. На рис. 20.2 изображены три сети, объединенные вместе.

Рис. 20.2. Сеть, объединяющая три подсети

В данном случае существующая сеть приобретает третий, более мелкий сегмент. Проблема в том, что два физических интерфейса маршрутизатора уже использованы.

Решение состоит в использовании параметра secondary команды ip address для указания вторичного IP-адреса для одного из интерфейсов.

Интерфейсам маршрутизатора Cisco можно установить вторичный адрес, что будет способствовать разрешению ситуаций, подобных описанной выше. Но любой пакет, отправленный с этого интерфейса, будет иметь в качестве адреса отправителя первичный адрес интерфейса.

Чтобы сконфигурировать вторичный адрес для маршрутизатора, изображенного на рис. 10.2, используйте такую команду:

Router(config-if)#\*ip address 10.98.20.1 255.255.0.0 secondary\*

Итак, некоторое количество команд, необходимых для настройки IP на маршрутизаторе, вы уже изучили. Теперь можно спокойно приступить к обсуждению остальных связанных с IP программ и функций.

**ICMP**

ICMP очень тесно связан с IP. Многие устройства используют сервисы ICMP для текущего контроля над сетью и получения информации о сетевом окружении. Тесная связь с IP имеет свои плюсы и минусы. Преимущество в том, что благодаря распространенности IP многие устройства допускают применение ICMP. С другой стороны, все же остается немного персональных компьютеров и других устройств, которые не используют IP, и, к сожалению, ICMP не в состоянии обнаружить и продиагностировать такие устройства.

Большая часть ПК и других пригодных для работы в сети устройств обладает способностью использовать такие средства, как ping и traceroute. Эти программы применяют ICMP-пакеты для проверки возможности соединения между устройствами. Если вы будете знать, как работают данные программы, и сумеете использовать их с выгодой для себя, это поможет содержать вашу сеть и маршрутизатор в хорошим состоянии.

Далее в разделе рассказывается о технологии и принципах ICMP. Для установки ICMP на маршрутизаторе не требуется никакого дополнительного конфигурирования, но вы должны знать, как им пользоваться.

Хотя для установки ICMP или сервисов ICMP на маршрутизаторе не требуется никакого дополнительного конфигурирования, необходимо наличие хотя бы одного работающего интерфейса, использующего IP. Если ни один из интерфейсов не сконфигурирован для работы с IP, вы все равно получите доступ к командам ICMP, только они не будут работать до тех пор, пока протокол IP для интерфейса не будет включен.

Утилиты ICMP обеспечивают возможность достаточно простой диагностики сетевых проблем. Поэтому для того чтобы научиться работать с маршрутизатором Cisco, вам необходимо знать эти команды и понимать, что они выводят.

**Использование утилит ICMP**

ICMP — это протокол, который используется почти исключительно для проверки связности узлов сети и диагностики возможных проблем. Из-за сложности маршрутизации без соответствующих средств тяжело разобраться даже с простыми вопросами. Давайте рассмотрим на примере, каким сложным может оказаться выявление проблем в маршрутизируемой среде.

Сетевые окружения становятся все больше и сложнее. Уже нередки случаи, когда в одной сети работают от 5 до 10 независимых маршрутизаторов. Но многие сети не ограничены стенами комнат. Соединения с Интернетом добавляют в среду маршрутизации сотни, если не тысячи маршрутизаторов. Типичная маршрутизируемая среда представлена на рис. 20.3.

Рис. 20.3. Характерная среда маршрутизации

Уровень сложности сети возрастает с увеличением количества маршрутизаторов. Одно часто встречающееся осложнение, возникающее в результате добавления новых маршрутизаторов, — это необходимость отслеживания пакетов внутри сети. Используя ICMP, администратор может следить за маршрутизацией пакетов от начала и до конца. Это не только помогает инженерам и проектировщикам определить, есть ли проблемы с соединениями в сети, но и обеспечивает предоставление информации о том, правильные ли пути используют маршрутизаторы.

Маршрутизаторы Cisco могут применять две утилиты ICMP (ping и traceroute) для всеобъемлющей оценки маршрута. Обе эти программы обладают мощными диагностическими возможностями.

**Ping**

Команда ping используется для проверки существования конечной системы. Пусть, например, вы пытаетесь передать по TFTP конфигурационный файл от маршрутизатора к TFTP-серверу и в ответ получаете такое сообщение:

%Error opening tftp://10.16.4.153/

Вам сообщают, что сервер, с которым вы пытаетесь вступить в контакт, не отвечает. Очевидно, что существует миллиард причин, по которым сервер может не отвечать. Чтобы уменьшить это количество, можно применить команду ping.

Программа ping использует запросы отклика ICMP для проверки доступности сервера, аналогично тому, как гидролокатор подводной лодки использует звуковые импульсы (которые долго называли «pings») для проверки наличия объектов в близлежащих водах. Обращаясь к IP-адресу сервера с запросом отклика и ожидая ответа, вы можете узнать, работает ли машина и правильно ли функционирует протокол. Так вы сужаете круг возможных проблем.

Ping работает, передавая выбранному IP-адресу запросы отклика ICMP. Когда устройство с указанным адресом получает ICMP-пакеты, оно отражает их обратно отправителю.

Однако единственное, в чем вы можете быть уверены, используя ping, – это в том, что устройство включено и работает правильно; кроме этого мало что можно узнать. Например (возвращаясь к последнему примеру), если вы посылаете ping-запрос на IP-адрес TFTP-сервера и не получаете ответа, это может указывать на несколько проблем. Сервер может быть выключен, сервер может быть не сконфигурирован для работы с IP, а может быть, дело совсем не в нем, и неправильно сконфигурирован маршрутизатор, с которого вы посылаете запрос. Узнать, что именно произошло, пока невозможно, но, по крайней мере, есть с чего начать.

Будучи командой Cisco IOS, ping имеет ряд дополнительных возможностей, которыми может не обладать стандартная версия. Именно благодаря этим дополнительным свойствам ping относится к тем немногим командам, которые можно исполнять и в пользовательском, и в привилегированном режимах. Давайте посмотрим на команду ping в пользовательском режиме.

Помните, что для использования любых программ ICMP на вашем маршрутизаторе должен работать хотя бы один интерфейс с установленным IP. Если не будет доступна ни одна линия передачи данных, то при попытке выполнить команду вы увидите такое сообщение об ошибке:  
% Unrecognized host or address, or protocol not running.

**ping (пользовательский режим)**

Стандартная версия команды ping доступна в пользовательском режиме работы командного процессора. В результате ввода ping после приглашения на ввод пользовательского режима появится сообщение:

Router> ping

% Incomplete command.

Дело в том, что команде ping необходим как минимум один параметр. Используя справочную систему Cisco IOS, можно вывести список параметров ping:

Router>ping ?

WORD Ping destination address or hostname

ip IP echo

tag Tag encapsulated IP echo

С двумя дополнительными параметрами: ip и tag мы не будем иметь дела. Первый из них, ip, указывает на то, что при отправке ping-запроса будет использоваться протокол IP. Cisco разрешает использовать для этой цели и другие протоколы, указывая параметры apollo, appletalk, clns, decnet, ipx, vines и xns`. Мы сосредоточимся на IP (значение по умолчанию).

Параметр tag применяется для определения инкапсуляции пакета с тегами. Это используется главным образом при коммутации и не будет описано в данном курсе.

Чтобы команда была полной, следует указать IP-адрес устройства, которому вы хотите отправить ping-запрос:

Router>\*ping 10.16.4.152\*

Выходные данные стандартной команды ping приведены ниже. На первый взгляд, сообщение может показаться непонятным, но если разделить его на части, то все прояснится.

Router>ping 10.16.4.152

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.16.4.152, timeout is 2 seconds:

!!!!!

Success rate is 100 percent, round-trip min/avg/max = 1/3/4 ms

Первая строка вывода (Type escape sequence to abort.) на самом деле генерируется IOS до того, как выполняется команда. Это сообщение о состоянии, информирующее пользователя о том, что процесс выполнения команды можно прервать. Чтобы прекратить выполнение ping, используйте комбинацию клавиш Ctrl+Shift+6.

Вторая строка — это тоже сообщение о состоянии, генерируемое Cisco IOS. Строка повторяет смысл команды ping. В нашем случае команда ping собирается отправить пять 100-байтных пакетов по IP-адресу 10.16.4.152. Тайм-аут установлен в 2 секунды.

Тайм-аут, или TTL (Time To Live, время жизни), — это часы, используемые для определения момента, когда можно отбросить пакет. Когда время жизни пакета истекает, он удаляется хранящим его устройством. Поэтому отправляющее устройство, не получив ответа в течение 2 секунд, делает вывод о том, что адрес назначения недостижим. Иначе отправитель бесконечно ждал бы ответ, который мог бы не прийти никогда.

Следующая строка выведенного сообщения иллюстрирует исполнение ping. Каждый символ (в нашем примере — восклицательный знак) представляет один пакет, отправленный маршрутизатором по адресу назначения. Восклицательные знаки означают, что ICMP-пакеты были успешно возвращены обратно маршрутизатору. Если бы вместо восклицательных знаков стояли точки, это бы означало потерю пакетов; точка показывает, что адрес назначения не откликнулся на команду ping.

В данной строке могут появиться восемь символов:

* ! (успех)
* . (неудача)
* U (конечный узел недостижим)
* N (сеть конечного узла недостижима)
* P (несовместимость протоколов)
* Q (замедление источника)
* M (пакет слишком велик для маршрутизации и не может быть фраг-ментирован)
* ? (неизвестный ответ)

Последняя выводимая строка — сводная информация. IOS известит вас об успешном выполнении ping и о том, через какое время был получен отклик (если получен).

Хотя мы еще не затрагивали понятие «перехода» (hop), отметим, что ping имеет ограничение на количество переходов. Маршрутизаторы в сети часто называют переходами (говоря о количестве маршрутизаторов, через которое должен пройти элемент информации, чтобы достичь своего конечного назначения). Например, говорят, что пакет, который перед доставкой прошел через три маршрутизатора, преодолел три перехода.

Команда ping (в пользовательском режиме) проверяет не более девяти переходов. Поэтому для любого проверяемого вами адреса, который находится от вас на расстоянии более девяти маршрутизаторов-переходов, будет выведен ответ «адрес назначения недостижим», даже если он полностью функционален.

Переходы еще будут обсуждаться далее в разделах, связанных с протоколами маршрутизации.

Может показаться, что выводится масса полезной информации, но на самом деле это самое элементарное использование команды. Чтобы получить доступ к более мощной версии ping, войдите в привилегированный режим.

**ping (привилегированный режим)**

Cisco IOS содержит немного более мощную, чем стандартная, версию команды ping, доступную в привилегированном режиме. Версия привилегированного режима позволяет администраторам оценить несколько переменных, а не просто проверить, отвечает ли узел.

Сначала можно подумать, что версии ping пользовательского и привилегированного режимов совпадают. Если вы воспользуетесь справочной системой Cisco IOS, то увидите, что две команды принимают одни и те же параметры:

Router#\*ping ?\*

WORD Ping destination address or hostname ip

IP echo

tag Tag encapsulated IP echo

(Пояснения параметров были приведены в предыдущем разделе.) Но выполнение ping пользовательского режима без параметров приводит к ошибке:

Router>\*ping\*

% Incomplete command.

А если попытаться выполнить без параметров ping привилегированного режима, то вы будете вовлечены в расширенный диалог ping. Приведем отрывок из такого диалога:

Router# \*ping\*

Protocol [ip]:

Target IP address: \*10.16.5.152\*

Repeat count [5]:

Datagram size [100]:

Timeout in seconds [2]:

Extended commands [n]:

Sweep range of sizes [n]:

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.16.5.152, timeout is 2 seconds:

!!!!!

Success rate is 100 percent, round-trip min/avg/max = 2/3/4 ms

Просматривая параметры – строка за строкой, – легче понять, какую информацию собирает ping в привилегированном режиме.

Первый вопрос, задаваемый диалогом: какой протокол вы хотите использовать для отправки ping-запроса. По умолчанию предлагается значение IP, на которое и следует согласиться. (Ранее уже говорилось, что ping Cisco может использовать и другие протоколы, но в данном курсе об этом не будет рассказано.) После того как в ответ на первый вопрос будет принято значение по умолчанию, появится вторая строка диалога.

Теперь ping интересуется адресом пункта назначения. Это самый важный элемент диалога. Если вы введете только адрес пункта назначения и пропустите все остальные вопросы, команда ping все равно будет выполнена. (Очевидно, что для проведения проверки адрес необходим.)

Далее вам следует ввести число повторений, то есть указать количество эхо-пакетов, которые вы хотите отправить получателю. По умолчанию их будет пять, но бывают ситуации, когда хотелось бы, чтобы их было больше или меньше. Например, если проводится проверка на неисправность соединения или его отсутствие, то пяти пакетов может быть недостаточно для выявления проблемы. Поэтому для периодически возникающих проблем с линией, может быть, лучше отправить не пять, а больше пакетов. Посылая больше пакетов, вы сможете дольше «пинговать» нужный адрес, повышая свои шансы на столкновение с периодической проблемой.

Затем ping-диалог предлагает задать размер эхо-пакета. Это полезно в сетях с большой рабочей нагрузкой. Хотя и кажется, что 100 байтов – это совсем немного данных, но вы можете захотеть уменьшить размер пакета. Например, если вы проверяете чрезвычайно загруженную линию (вероятно, чтобы определить причину перегруженности), пропускной способности может не хватить для эффективной отсылки пяти 100-байтных эхо-пакетов. Поэтому вы можете решить использовать пакеты меньшего размера.

Следующая строка диалога связана с установкой времени жизни эхо-пакетов. Обычно нет причин изменять значение по умолчанию, равное 2 секундам. Так у маршрутизатора будет достаточно времени для того, чтобы определить, откликается устройство или нет.

Потом ping-диалог спрашивает, хотите ли вы разрешить расширенные команды. Ответ "yes" откроет новый диалог; если же вы ответите "no", то будет продолжен текущий диалог. В текущем диалоге будет предложено ввести диапазон изменения размеров.

Когда указан диапазон изменения размеров, ping меняет размер отправляемых эхо-пакетов. Это помогает определить, влияет ли размер пакета на возможность соединения ваших конечных систем. Обычно на выполнение ping воздействует размер дейтаграммы, принуждая все пакеты быть одинакового размера, но если указан данный параметр, то создаются пакеты разных размеров.

После этого команда ping выполняется. Если же вы разрешите расширенные команды, то увидите такие приглашения:

Extended commands [n]: \*y\*

Source address or interface: \*10.153.16.4\*

Type of service [0]:

Set DF bit in IP header? [no]:

Validate reply data? [no]:

Data pattern [0xABCD]:

Loose, Strict, Record, Timestamp, Verbose[none]: \*Record\*

Number of hops [ 9 ]:

Loose, Strict, Record, Timestamp, Verbose[RV]:

Sweep range of sizes [n]:

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.153.16.5, timeout is 2 seconds:

Packet has IP options: Total option bytes= 39, padded length=40

Расширенные команды позволяют получить еще больший контроль над командой ping и ее выполнением.

Первый вопрос диалога расширенных команд касается адреса отправителя запросов отклика. Получив возможность указать адрес источника, вы сможете следить за тем, какой интерфейс маршрутизатора выполняет команду. Такая возможность особенно полезна, если у вас несколько интерфейсов и все они потенциально могут обращаться к одним и тем же адресатам. Задание адреса интерфейса отправителя поможет вам уменьшить количество проблем с соединениями. В сценариях с несколькими интерфейсами маршрутизатора, имеющими возможность контактировать с некоторым адресом пункта назначения, можно выполнять расширенную команду ping для каждого интерфейса. Так вы определите, какой путь к устройству создает проблемы.

Все остальные приглашения относятся к более сложным вещам. Они связаны со способом форматирования пакетов, то есть спрашивается, какая структура будет использоваться в отдельных частях пакета. Это очень специфические параметры, но о них стоит поговорить.

Бит DF в заголовке IP указывает, что пакет не подлежит фрагментированию (don't fragment — DF). Маршрутизаторы и другие IP-устройства могут быть сконфигурированы так, чтобы перемещать только пакеты определенных размеров. Поэтому когда поступает пакет с размером, превышающим установленный, устройство просто фрагментирует его на более мелкие пакеты, которые оно может обрабатывать. Обычно это делается незаметно для пользователя. Устанавливая бит DF, вы сообщаете всем остальным устройствам сети: «Если вы сконфигурированы так, что можете перемещать только более мелкие, чем данный, пакеты, оставьте этот пакет и отправьте мне сообщение об ошибке». Такая возможность может быть полезна, если вы пытаетесь найти устройство, которое может быть ошибочно сконфигурировано для работы с маленькими пакетами, но обычно нет необходимости изменять эту настройку.

Два следующих параметра — это «validate replies» (подтверждать ответы) и «set data pattern» (определить структуру данных). Подтверждение ответов может быть избыточным, так как оно делает только одну вещь: создает ответ на ответ. Однако если у вас возникает периодическая проблема, этот параметр может быть полезен для определения того, правильные ли ответы вы получаете.

Определение структуры данных позволяет изменить существующую структуру битов эхо-пакета. Изменив структуру битов пакета, вы можете использовать анализатор пакетов для исследования структуры отправленного пакета в сравнении с полученным ответом. Это помогает при проверках на наличие помех в линиях и перекрестных наводок. Если структура битов ответа отличается от структуры отправленного пакета, значит, вы улавливаете какие-то электронные шумы.

Следующая строка выглядит немного загадочно. Требуется ввести один из перечисленных параметров:

Loose, Strict, Record, Timestamp, Verbose[none]:

Но непонятно, что означают эти параметры. «Loose», «Strict», «Times-tamp» и «Verbose» — это те способы, которыми маршрутизатор может исследовать заголовок пакета (пояснения даны в табл. 20.1). Ответ по умолчанию — «none».

Помните, что параметры Loose, Strict, Timestamp, Record и Verbose вводятся в поле «options» IP-заголовка. Нет гарантии, что каждый маршрутизатор сети сможет понять, как нужно обрабатывать эти команды. Это главным образом относится к не-Cisco-маршрутизаторам.

Таблица 20.1. Дополнительные параметры ping

| **Параметры** | **Описание** |
| --- | --- |
| Loose source routing (свободное исполнение маршрута отправителя) | Указывает список устройств, которые должны участвовать в работе трассировщика. Другими словами, трассировщик может проводить пакеты через разные маршрутизаторы, при этом те, для которых указан параметр «loose», должны быть включены в путь |
| Strict source routing (строгое исполнение маршрута отправителя) | Аналогичен параметру «loose», но в путь трассировщика должны быть включены только указанные устройства |
| Record (запись маршрута) | Позволяет указать требуемое количество переходов |
| Timestamp (временные метки) | Позволяет указать требуемое количество временных меток |
| Verbose (подробное описание) | Выбор любого из параметров (кроме «none») переводит пакет в режим «verbose». Результаты применения дополнительных параметров выводятся в пользовательском интерфейсе |

Установив для пакета параметр «Record», вы будете получать эхо от каждого устройства, которое пакет встретит на своем пути к месту назначения. Если выбран этот параметр, выводится такое приглашение:

Number of hops [ 9 ]:

В указании количества переходов, которые вы хотите записать, проявляется отличие между трассировкой и записью маршрута. Тогда как ping ограничивается максимум девятью переходами, для трассировщика такого ограничения нет. Данный параметр позволяет указать количество переходов, не превосходящее 9 (значение по умолчанию).

Далее версия ping привилегированного режима выполняется точно так же, как и версия пользовательского режима. Команда ping хороша для определения доступности адресата, а для выявления проблем с путями необходима команда traceroute.

**traceroute**

Как вы уже знаете, команда ping успешно определяет, отвечает ли конечная система на запросы отклика ICMP. Но она не может объяснить, почему какой-то узел не отвечает.

Есть еще одна утилита, использующая ICMP-пакеты для проверки доступности конечного узла, — traceroute. Она посылает эхо-пакеты не только с адреса назначения, но и со всех устройств, которые встречаются на пути продвижения к конечной цели.

Команда traceroute, как и ping, имеет в Cisco IOS две версии: для пользовательского и для привилегированного режимов. Пользовательская версия имеет те же параметры, что и пользовательская версия ping (и работает аналогично). В пользовательской версии traceroute вы можете указать цель, путь к которой хотите отследить. Будучи выполненной, traceroute возвращает имена и адреса всех устройств, через которые она прошла, чтобы достичь указанного объекта:

Router>\*trace 10.16.4.153\*

Type escape sequence to abort.

Tracing the route to 10.16.4.153

1 Router.testnode.com (10.16.4.199) 62 msec 82 msec 78 msec

2 RouterB.testnode.com (10.16.4.189) 80 msec 99 msec 117 msec

3 RouterC.testnode.com (10.16.4.177)100 msec 110 msec 124 msec

Обратите внимание на то, что в конце каждого ответа стоит время, которое потребовалось на получение отклика от данного устройства. Как видите, по умолчанию traceroute посылает три эхо-пакета (в отличие от пяти для ping).

Что касается параметров выполнения программ, пользовательская версия traceroute мало отличается от пользовательской версии ping. Для обеих команд можно определить адрес пункта назначения и протокол, все остальное они делают сами.

Если вы хорошо разобрались в пользовательской версии команды ping, то пользовательская версия traceroute не заcтанет вас врасплох, поэтому мы не будем останавливаться на ней и перейдем к версии привилегированного режима.

**traceroute (привилегированный режим)**

Как и в случае ping, все самые важные конфигурационные свойства traceroute доступны в привилегированном режиме. Так администраторы маршрутизаторов могут сохранить более полный контроль над сетью.

Выполнение команды traceroute в привилегированном режиме без указания адресата приведет к выводу такого диалога:

Router#\*trace\*

Protocol [ip]:

Target IP address: \*10.16.4.153\*

Source address: Numeric display [n]: \*n\*

Timeout in seconds [3]:

Probe count [3]:

Minimum Time to Live [1]:

Maximum Time to Live [30]:

Port Number [33434]:

Loose, Strict, Record, Timestamp, Verbose[none]:

Type escape sequence to abort.

Tracing the route to 10.16.4.153

Сразу бросается в глаза, что многие приглашения на ввод повторяют приглашения версии ping привилегированного режима. Так как их назначение аналогично параметрам ping, то Protocol, Target, Timeout, Probe count, Loose, Strict, Record, Timestamp и Verbose не будут рассматриваться в данном разделе. Чтобы вспомнить возможные значения и свойства этих параметров, вернитесь к разделу «ping (привилегированный режим)».

Давайте поговорим о тех параметрах, которые доступны только в версии traceroute привилегированного режима. Первый из них (хотя может показаться, что этот параметр тоже подобен своему аналогу в ping) – это адрес источника.

В отличие от ping, в параметре адреса источника нельзя указать имя интерфейса. Адрес источника должен быть протокольным адресом, назначенным физическому интерфейсу. Не считая этого отличия, в остальном параметр адреса источника используется так же, как и в команде ping.

Два следующих параметра, «Minimum Time to Live» (минимальное время жизни) и «Maximum Time to Live» (максимальное время жизни) связаны со временем жизни пакета. Так как при использовании traceroute мы имеем дело с многочисленными устройствами, то время задержки может меняться от одного к другому. Поэтому вместо установки статического TTL для всех устройств traceroute создает для времени жизни пороговые значения.

Утилита traceroute может охватывать больше переходов, чем ping, поэтому ей необходимо больше времени для того, чтобы пакеты достигли конечного адресата. В результате двухсекундный тайм-аут, используемый ping, может оказаться недостаточным для traceroute. Устанавливая минимальное и максимальное значения, traceroute делает поправку на задержки между переходами. По умолчанию пороги равны 1 и 30 секундам. Этого времени должно хватить для компенсации незначительных сетевых задержек.

Последний параметр traceroute — это номер порта. Утилита traceroute позволяет пользователю указать, с какого IP-порта (протокольного порта) он хотел бы отправлять traceroute-пакеты. Это может быть очень важно для диагностики возможных проблем с безопасностью. Указывая порты для отправки traceroute-пакетов, вы можете определить, открыты ли порты, которые должны быть открыты, и, наоборот, закрыты ли порты, которые не должны быть открыты. Так как многие сети работают по незащищенным протокольным портам, отсылая traceroute-пакеты с разных портов, вы сможете уточнить архитектуру вашей сети.

Эти простые команды (ping и traceroute) очень помогают в диагностике и поиске неисправностей при маршрутизации. Не жалейте времени на изучение этих команд, их параметров и форматов вывода. Зная, как работают ping и traceroute, вы сможете значительно сократить время, необходимое для проверки и создания новых маршрутов и схем сети.

Теперь, когда мы поговорили об «ICMP-родственниках» IP, пришло время познакомиться с «родной» утилитой IP — Telnet. Даже если вы уже работали с IP, материал следующего раздела будет полезен для вас.

**Telnet**

Имея некоторый опыт работы с ПК и сетями, вы, наверное, уже использовали Telnet (или хотя бы слышали о нем). Telnet — это мощное средство для регистрации на удаленных устройствах и выполнения на них команд, особенно это касается работы в рамках операционной системы UNIX.

Маршрутизаторы Cisco могут использовать возможности Telnet двумя способами.

Во-первых, Telnet позволяет администраторам получить удаленный доступ к маршрутизаторам (командным интерпретаторам и конфигурационным файлам). У администратора пропадает необходимость физически находиться у консолей маршрутизаторов для работы с ними. Наиболее распространенная форма администрирования маршрутизатора — это доступ к одному или нескольким устройствам по Telnet с персонального компьютера.

Вторая возможность (чем-то напоминающая первую) использования Telnet заключается в доступе администраторов, работающих с консолью маршрутизатора, к другим маршрутизаторам Cisco. То есть сами маршрутизаторы Cisco способны использовать Telnet-клиент для удаленного администрирования других маршрутизаторов Cisco. Это означает, что инженер Cisco может войти в систему на одном маршрутизаторе, а затем, используя командную строку, зайти на другие маршрутизаторы Cisco.

Однако эта невероятно полезная возможность представляет собой одну из самых больших проблем Cisco в плане обеспечения безопасности. Во многом так же, как римская система дорог была обращена захватчиками против римлян («все дороги ведут в Рим»), сетевые злоумышленники, получившие доступ к одному маршрутизатору, могут легко переместиться и на другие, используя Telnet. Поэтому будьте внимательны при защите своих соединений. Позже мы обсудим основы безопасности Cisco.

**Удаленное администрирование с использованием Telnet**

Для работы Telnet требуется некоторое дополнительное конфигурирование. То есть если вы хотите удаленно администрировать маршрутизатор по Telnet, необходимо настроить его соответствующим образом. Если же вы хотите просто использовать маршрутизатор в качестве Telnet-клиента (соединяться с него с другими маршрутизаторами и администрировать их), в дополнительной настройке нет необходимости.

По умолчанию сервисы Telnet-сервера на маршрутизаторах Cisco запрещены. Это сделано для предотвращения использования маршрутизаторов в то время, когда администраторы и не подозревают об активности протокола. Если вы хотите включить Telnet на маршрутизаторе, разрешив Telnet-клиентам удаленно администрировать маршрутизатор, вам необходимо выполнить ряд операций.

До этого наши попытки конфигурирования относились к интерфейсам (Ethernet0 или Ethernet1). Telnet же необходимо настраивать как линию передачи данных. Используйте команду line для конфигурирования линии (не забудьте указать, какую линию или линии вы хотите сконфигурировать; введя line 0 5, вы одновременно будете настраивать линии с 0 по 5).

Войдя в режим конфигурирования линии, запросите справку по Telnet, чтобы посмотреть, какие параметры конфигурирования Telnet можно задать на вашем маршрутизаторе:

Router#\*configure terminal\*

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#\*line 0 5\*

Router(config-line)#\*Telnet ?\*

break-on-ip Send break signal when interrupt is received

ip-on-break Send interrupt signal when break is received

refuse-negotiations Suppress negotiations of Telnet Remote Echo and

SuppressGo Ahead options

speed Specify line speeds

sync-on-break Send a Telnet Synchronize signal after receiving a

Telnet Break signal

transparent Send a CR as a CR followed by a NULL instead of a CR

followed by a LF

Для каждого параметра даны пояснения. Но вам в настоящее время нужен один параметр — speed. Параметр speed должен подготовить ваш маршрутизатор к принятию удаленных соединений.

В этом примере использована команда line 0 5, которая настраивает все доступные линии передачи данных на прием Telnet-соединений. Вы можете сделать еще один шаг и установить для каждой из линий отдельный пароль, но это может быть и не совсем продуктивно.  
При обращении к маршрутизатору Cisco по Telnet нет возможности указать терминальную линию, которую вы хотели бы использовать. Поэтому если вы установите шесть разных паролей для шести линий, то может оказаться, что вам придется перепробовать их все, прежде чем получить доступ к системе. На это накладывается еще тот факт, что Cisco IOS предоставляет только три попытки для ввода правильного пароля, а затем блокирует соединение.

Чтобы завершить конфигурирование Telnet, введите следующую команду в режиме конфигурирования линии:

Router(config-line)#\*telnet speed 9600\*

Эта команда разрешит (в нашем случае) линиям 0-5 принимать Telnet-содинения на скорости 9600. Теперь вы можете как с удаленного ПК, так и с другого маршрутизатора, установить соединение по Telnet с любой из этих линий.

После успешной настройки маршрутизатора, вы можете захотеть проверить возможности Telnet. Для этого сначала убедитесь, что хотя бы один интерфейс маршрутизатора подключен к сети. Затем попытайтесь подключиться к маршрутизатору с Telnet-клиента.

Для этого попробуйте с Telnet-клиента подключиться к одному из установленных адресов маршрутизатора. Вы должны увидеть такое сообщение:

\*C:\Telnet 10.16.4.153\*

Line 0

Password required but none set

Disconnecting

Как видите, недостаточно просто разрешить линиям работу с Telnet; Cisco не разрешает использовать незащищенное Telnet-соединение. Поэтому последний шаг в настройке Telnet заключается в установке пароля для тех линий, которым разрешено принимать соединения.

Router#\*configure terminal\*

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#\*line 0 5\*

Router(config-line)#\*password Telnet\*

Router(config-line)#\*^Z\*

Снова попробуем подключиться к маршрутизатору. Вас должно встретить приглашение на вход в систему:

\*C:\Telnet 10.16.4.153\*

login

User Access Verification

Password: \*Telnet\*

Router>

Маршрутизатор успешно сконфигурирован для принятия входящих Telnet-соединений. Давайте перейдем к рассмотрению использования вашего маршрутизатора в качестве Telnet-клиента.

Использование маршрутизатора в качестве Telnet-клиента не требует дополнительного конфигурирования (не надо настраивать маршрутизатор на прием Telnet-соединений). Доступ по Telnet с одного маршрутизатора Cisco на другой очень прост, он выполняется посредством одной из трех команд. Для открытия сеансов связи между маршрутизаторами (в административных целях) можно использовать команды telnet, connect и rlogin.

Помните, что все три команды telnet, connect и rlogin — это стандартные команды Unix. Поэтому их использование в рамках Cisco не ограничивается соединением с маршрутизаторами. Администраторы могут с маршрутизатора Cisco обращаться по Telnet к Unix-серверу (или почти любым другим серверам, совместимым с Telnet). Но в данной книге использование Telnet будет рассмотрено только в связи с маршрутизаторами Cisco.  
Команды telnet и connect работают одинаково и являются взаимозаменяемыми. Чтобы инициировать сеанс Telnet с маршрутизатора Router на другой маршрутизатор RouterB, используйте такой формат команды:

Router>\*telnet RouterB\*

Translating "RouterB"...domain server (10.16.4.188) [OK]

Trying Server3-RouterB.STYSCisco.com (10.16.4.188). . . Open

login: \*userforRouter\*

Password:

RouterB)

Ввод команды connect приводит к такому же результату. Обе команды доводят вас до приглашения на вход в систему на нужном сервере. Если вы знакомы с командой Telnet в Unix, знайте, что она отличается от Cisco Telnet. Cisco Telnet не поддерживает доверительные соединения. То есть Cisco Telnet не может аутентифицировать Telnet-соединение, основанное на данных авторизации текущего соединения. Поэтому каждый инициированный сеанс Telnet выводит приглашение на регистрацию.

Установить Telnet-соединение можно и вообще без использования команд. Например, если ввести в командной строке несуществующую команду, то в ответ будет выведено такое сообщение:

Router>\*RouterB\*

Translating "RouterB". . . domain server (255.255.255.255)

% Unknown command or computer name, or unable to find computer address

Такое сообщение об ошибке генерируется Cisco IOS при попытке установить Telnet-соединение с указанным сервером (не осознавая, что вы хотели выполнить команду). В данном случае не существовало Telnet-сервера с именем RouterB (как и команды RouterB), поэтому было сгенерировано сообщение об ошибке. Если бы Telnet-сервер с именем RouterB существовал, вы увидели бы следующее сообщение:

Router>\*RouterB\*

Translating "RouterB"...domain server (10.16.4.188) [OK]

Trying Server3-RouterB.STYSCisco.com (10.16.4.188) . . . Open

login:

Необходимо сделать одно замечание: если имя Telnet-сервера совпадает с названием команды Cisco, то для обращения к такому серверу необходимо предварить его имя ключевым словом telnet или connect. Например, чтобы подключиться к Telnet-серверу Enable, нужно использовать такое обозначение:

Router> \*connect enable\*

Translating "enable"...domain server (10.16.4.189) [OK]

Trying Server3-enable.STYSCisco.com (10.16.4.189) . . . Open

После того как Telnet-соединение успешно установлено, вы увидите новое приглашение на ввод команды, которое состоит из имени удаленного сервера и правой скобки:

RouterB)

Такое приглашение показывает, что вы выдаете команды удаленному устройству. Каждый удаленный маршрутизатор, с которым вы установите соединение, будет иметь свое уникальное приглашение на ввод. Но большое количество приглашений на ввод может сбивать с толку, а переключение с одного на другое может быть затруднительным. Поэтому Cisco создала команду resume. Чтобы перейти от одного сеанса к другому, используйте в любой командной строке синтаксис resume <connection name>:

Router>\*resume RouterB\*

RouterB)

Команда resume переместит вас в указанную сессию. Чтобы завершить сессию, используйте команду exit:

RouterB)\*exit\*

Router>

Есть еще две команды, которые могут оказаться очень полезными для тех, кто планирует использовать несколько Telnet-соединений. Это команды notify и refuse-message.

Команда notify (после того как она будет разрешена) будет извещать вас о том, что выходные данные удаленного сеанса (отличного от того, в котором вы работаете в данный момент) требуют вашего внимания. В приведенном ниже примере показано, как разрешить использование свойства notify:

Router#\*configure terminal\*

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#\*line 0 5\*

Router(config-line)#\*notify\*

Router(config-line)#\*^Z\*

Router#

Команда refuse-message будет уведомлять всех, пытающихся установить Telnet-соединение с маршрутизатором, о том, что линия уже используется:

Router#\*configure terminal\*

Enter configuration commands, one per line. End with CNTL/Z. Router(config)#\*line 0 5\*

Router(config-line)#\*refuse-message ! line in use !\*

Router(config-line)#\*^Z\*

Router#

Команда refuse-message имеет такие же параметры, как и большинство баннерных сообщений.

Команды telnet и connect могут использоваться для доступа к удаленным маршрутизаторам (и другим Telnet-серверам). Для этого же можно использовать и еще одну команду, имеющую ограниченное (специализированное) применение: rlogin.

**rlogin**

Команда rlogin работает так же, как telnet и connect. Основное ее отличие заключается в том, что не все устройства, поддерживающие Telnet, поддерживают и rlogin. Изначально rlogin создавалась для BSD Unix. Поэтому, если вы хотите инициировать сеанс rlogin, то сервер, принимающий соединение, должен быть сконфигурирован для работы с rlogin.

Синтаксис команды очень напоминает синтаксис telnet:

Router>\*rlogin RouterB\*

Преимуществом rlogin является то, что она поддерживает возможность отладки вывода. Указав, что соединение должно быть установлено в режиме отладки, вы получите более полный контроль над потоком вывода rlogin.

Router>\*rlogin RouterB debug\*

Использование таких команд, как ping, traceroute, Telnet, connect и rlogin, предоставляет вам возможность управления вашей средой маршрутизации. Если вы освоите эти команды, то будете готовы к восприятию более сложных свойств, с которыми еще предстоит познакомиться.

**Резюме**

* Поддержка IP по умолчанию устанавливается на маршрутизаторах Cisco в каждом из функциональных пакетов Cisco IOS, кроме IPX Basic Feature Pack.
* IP-адреса устанавливаются в режиме конфигурирования интерфейсов привилегированного режима работы командного интерпретатора.
* Чтобы войти в режим конфигурирования интерфейсов, используйте команду interface.
* Статические IP-пути записываются в файл startup-config при помощи параметра perm.
* В состав стека протоколов TCP/IP входит несколько утилит ICMP.
* Команда ping проверяет существование объекта, отправляя ему запросы отклика ICMP.
* Команда traceroute работает аналогично ping. Но traceroute посылает эхо-пакеты всем устройствам на пути ее следования к пункту назначения.
* Еще одной важной утилитой IP является Telnet.
* Маршрутизаторы Cisco могут устанавливать несколько Telnet-соединений и как клиенты, и как серверы.

**Основы обеспечения безопасности Cisco**

Далее мы поговорим о двух основных методах обеспечения безопасности, реализованным в маршрутизаторах Cisco. Спросите любого, кто знаком с маршрутизаторами Cisco, что сегодня считается наиболее важным в электронном бизнесе, и ответ будет один — безопасность. Связано ли это с предотвращением утечки ваших внутренних данных или с минимизацией рисков путем ограничения доступа к внутренней сети, вопросы безопасности остаются на первом месте.

Понятие безопасности, даже в применении к Cisco, является весьма обширным и сложным. Далее представлены два основных метода, знакомство с которыми обязательно даже для начинающего пользователя Cisco: списки доступа IP и трансляция сетевых адресов (NAT -Network Address Translation). Комбинируя NAT и списки доступа, можно создать простую и в то же время достаточно безопасную систему. Обе эти технологии доступны на большинстве маршрутизаторов Cisco и в большинстве версий IOS.

Трансляция сетевых адресов выполняется обычно на шлюзе, то есть на маршрутизаторе, имеющем непосредственный выход в Интернет. Принцип действия NAT заключается в том, что одному из интерфейсов маршрутизатора назначается маршрутизируемый IP-адрес (или группа адресов), а другой интерфейс получает немаршрутизируемый адрес, принадлежащий внутренней сети. Задачей NAT является трансляция внутреннего немаршрутизируемого трафика во внешний, маршрутизируемый. Трансляция адресов хорошо подходит для обеспечения начального уровня безопасности, так как значительно усложняет задачу злоумышленникам, решившим «подсесть» на входящий и исходящий трафик сети.

Трансляция адресов работает рука об руку с еще одним базовым методом обеспечения сетевой безопасности: списками доступа IP. Списки доступа предоставляют администратору механизм для разрешения и запрещения трафика в соответствии с установленными им правилами. Эти правила могут учитывать порт, используемый входящим трафиком, адрес отправителя пакета и т. п.

Списки доступа должны быть предметом каждодневной заботы администратора. Возвращаясь к предыдущим разделам этого курса, можно заметить, что в большинстве протоколов, используемых маршрутизаторами, применяются списки доступа. Списки доступа могут быть двух типов: IP и IPX.

**Списки доступа IP**

Список доступа (access list) представляет собой набор инструкций, сообщающих маршрутизатору, как он должен обращаться с различными пакетами. Маршрутизаторы Cisco используют списки доступа для управления входящими и исходящими потоками данных. Хотя в этом разделе рассматриваются только два типа списков доступа (стандартный IP и расширенный IP), всего на маршрутизаторах Cisco их может быть настроено до 11 видов:

* Стандартный IP
* Расширенный IP
* Код типа протокола
* DECnet
* Appletalk
* MAC (Media Access Control layer)
* Стандартный IPX
* Расширенный IPX
* IPX SAP
* Расширенный MAC
* IPX summary address

Управление доступом может быть настроено на маршрутизаторах Cisco практически для всех протоколов. Однако наиболее популярным остается IP.

Конфигурирование списков доступа на маршрутизаторе Cisco осуществляется в два приема. Сначала создается собственно список доступа. Другими словами, формулируются общие правила, определяющие, что маршрутизатор должен делать с определенными пакетами. На следующем этапе список доступа связывается с некоторым интерфейсом.

Таким образом, разные списки доступа могут быть связаны с разными интерфейсами.

Каждый список доступа идентифицируется номером. Этот номер, присвоенный списку доступа, частично определяется его типом. Каждому типу списков доступа назначен диапазон из ста номеров (в предположении, что может быть создано до ста списков доступа каждого типа). В табл. 20.1 приведены диапазоны номеров, выделенные каждому из типов списков доступа.

Таблица 20.1. Номера списков доступа

| **Тип списка доступа** | **Диапазон номеров** |
| --- | --- |
| Стандартный IP | 1-99 |
| Расширенный IP | 100-199 |
| Код типа протокола | 200-299 |
| DECnet | 300-399 |
| Appletalk | 600-699 |
| MAC (media access control sublayer) | 700-799 |
| Стандартный IPX | 800-899 |
| Расширенный IPX | 900-999 |
| IPX SAP | 1000-1099 |
| Расширенный MAC | 1100-1199 |
| IPX summary address | 1200-1299 |

Стандартный список доступа IP всегда получает номера от 1 до 99, а расширенный — от 100 до 199. Эти два типа списков доступа мы и рассмотрим в данном разделе.

Для конфигурирования списка доступа на маршрутизаторе Cisco используйте команду access-list в режиме глобального конфигурирования. Режим глобального конфигурирования используется потому, что на данном этапе лишь определяется набор правил. А вот связывание списка доступа с интерфейсом должно выполняться в режиме конфигурирования интерфейса. Приведенная ниже последовательность команд создает стандартный список доступа IP:

Router#\*configure terminal\*

Router(config)#\*access-list 1 permit 198.42.16.1\*

Router(config)#\*^Z\*

Маршрутизаторы Cisco (в том, что касается списков доступа) придерживаются правила, известного как неявное запрещение. При неявном запрещении считается запрещенным все, что не упомянуто в списке доступа. Следуя этому правилу (на основании только что созданного списка доступа), всем адресам, кроме 186.42.16.1, запрещается доступ к ресурсам, расположенным за данным маршрутизатором.

Команда для конфигурирования стандартного списка доступа IP имеет такой формат:

#\*access-list\* <access-list number> <action> <source address>

Список доступа, созданный в предыдущем примере, устанавливает правило: «Разрешить прохождение трафика, приходящего с адреса 198.42.16.1». Другое возможное значение параметра <action> — это deny. Следующая команда создает стандартный список доступа IP, запрещающий прохождение трафика с адреса 10.36.149.8:

Router#\*configure terminal\*

Router(config)#\*access-list 1 deny 10.36.149.8\*

Router(config)#\*^Z\*

Стандартные списки доступа прекрасно подходят для небольших сред, в которых разрешение или запрет одного-двух адресов может способствовать повышению безопасности сети. Стандартные списки позволяют создавать общие правила, разрешающие или запрещающие весь трафик с определенного адреса, но такое решение нельзя назвать очень гибким.

Расширенные списки доступа предлагают пользователям Cisco значительно более широкие возможности в назначении правил в зависимости от типа пакетов. Команда создания расширенного списка доступа имеет такой формат:

#\*access-list\* <access-list number> <action> <protocol> <source address> <destination address> <port>

При конфигурировании расширенных списков доступа пользователю доступно большее количество параметров, чем для стандартных списков. Эти дополнительные параметры позволяют маршрутизаторам Cisco фильтровать трафик, основываясь на адресах отправителя и получателя, типах протоколов и номерах портов. Например, чтобы запретить трафик протокола Telnet (порт 23) с адреса 10.98.12.1 на 10.99.36.5, используем следующие команды:

Router#\*configure terminal\*

Router(config)#\*access-list 100 deny IP host 10.98.12.1 host 10.99.36.5 23\*

Router(config)#\*^Z\*

Протокол Telnet использует TCP-порт 23. В табл. 20.2 перечислены наиболее распространенные сервисы и соответствующие им номера портов.

Таблица 20.2. Общепринятые номера портов

| **Сервис** | **Порт** |
| --- | --- |
| SSH | 22 |
| Telnet | 23 |
| SMTP | 25 |
| HTTP | 80 |
| LDAP | 389 |
| MS NetMeeting | 1024,1503 |
| HTTPs | 443 |
| SOCKS | 1080 |
| MS NetShow | 1755 |
| MSN Messenger | 1863 |
| Mirabilis ICQ | 1024 |
| AOL Instant Messenger | 5190 |
| AOL ICQ | 5190 |
| AOL | 5190-5193 |
| Dialpad.com | 5354, 7175, 8680-8890, 9000, 9450-9460 |
| pcAnywhere | 5631 |
| VNC | 5800+,5900+ |
| Netscape Conference | 6498, 6502 |
| Common IRC | 6665-6669 |
| RealAudio and Video | 7070 |
| VocalTec Internet Conference | 22555 |
| MSN Gaming Zone | 28800-29000 |
| DirectX Gaming | 47624, 2300-2400 |

В предыдущем примере использовано ключевое слово host перед адресами отправителя и получателя. Если необходимо расширить правило, например запретить HTTP-трафик для всей сети, то можно использовать ключевое слово any:

Router#\*configure terminal\*

Router(config)#\*access-list 147 deny IP any 10.0.0.0 0.255.255.255 any 0.0.0.0 255.255.255.255 80\*

Router(config)#\*^Z\*

Данное правило запрещает всем локальным пользователям (сети 10.0.0.0) доступ к любым веб-сайтам (во всем диапазоне 0.0.0.0 -255.255.255.255 на 80 порту). Однако использование ключевого слова any таит в себе и некоторую опасность. Пользователи и сервисы, нуждающиеся в легитимном доступе к некоторым адресам, могут быть заблокированы. Составление списков доступа потребует от вас большой тщательности.

Эти два типа списков доступа, стандартный и расширенный, могут комбинироваться произвольным способом, позволяя фильтровать трафик так, как администратор сочтет наиболее подходящим. Однако создание списка доступа — это лишь половина дела; после этого список должен быть связан с одним или несколькими интерфейсами.

При связывании списка доступа с интерфейсом используется ключевое слово access-group команды ip в режиме конфигурирования интерфейса. Такая команда допускает только один параметр: in или out. Этот параметр определяет, к каким пакетам применяется указанное правило — входящим или исходящим. В приведенной ниже последовательности команд создаются два списка доступа — один стандартный и один расширенный, затем они связываются с интерфейсом маршрутизатора — ethernet 0.

Router#\*configure terminal\*

Router(config)#\*access-list 13 permit 128.53.12.1\*

Router(config)#\*access-list 108 deny IP host 198.20.13.118 host 128.53.12.1 80\*

Router(config)#\*^Z\*

Router#\*configure terminal\*

Router(config)#\*interface ethernet 0\*

Router(config-if)#\* ip address 198.20.13.115\*

Router(config-if)# \*ip access-group 108 out\*

Router(config-if)# \*ip access group 13 in\*

В этом примере созданы два списка доступа. Первый разрешает входящий трафик с адреса 128.53.12.1, а второй запрещает весь трафик с адреса 198.20.13.118 на 80 порт (HTTP) адреса 128.53.12.1.Таким образом, пользователю с адресом 198.20.13.118 запрещен просмотр веб-страниц по адресу 128.53.12.1:80, и в то же время сервисы этого сайта имеют доступ к локальной сети.

В процессе работы со списками доступа важно постоянно помнить о правиле неявного запрещения. Любой пакет, не соответствующий правилам, установленным в списке доступа, будет отброшен.

Списки доступа — простой и в то же время эффективный способ управления входящим и исходящим трафиком сети. Еще одним, столь же эффективным инструментом, использующим списки доступа, является трансляция сетевых адресов.

**NAT**

Трансляция сетевых адресов (NAT, Network Address Translation) — это средство, позволяющее шлюзу отображать внешние IP-адреса на множество внутренних IP-адресов. Этот способ обычно используется на шлюзах, обеспечивающих доступ в Интернет. В связи со все возрастающей нехваткой IP-адресов многие сетевые администраторы предпочитают использовать общедоступные IP-адреса. Проблема с использованием таких адресов заключается в том, что они не могут маршрутизироваться в Интернете.

При использовании NAT один выделенный организации IP-адрес присваивается шлюзу, выполняющему преобразование адресов всего входящего и исходящего трафика таким образом, что весь исходящий трафик помечается одним внешним адресом, а входящий корректно перенаправляется на соответствующие внутренние адреса.

Трансляция сетевых адресов может служить эффективным инструментом обеспечения безопасности благодаря тому, что в процессе преобразования изменяются адреса отправителя и получателя, что существенно осложняет злоумышленникам доступ к сети через службы IP. Помимо этого, NAT использует в своей работе списки доступа, что также повышает безопасность трансляции. В основе работы NAT, как и у списков доступа, лежит концепция «внутреннего» и «внешнего» трафиков.

Чтобы полностью сконфигурировать NAT на маршрутизаторе Cisco, необходимо определить внутренние и внешние характеристики используемых интерфейсов. Обычно на шлюзе хотя бы один интерфейс имеет адрес во внутренней сети и один — в Интернете (или в другой внешней сети). Внутренний интерфейс конфигурируется по «внутренним» правилам NAT, а внешний — по «наружным» правилам. Конфигурирование NAT состоит из четырех основных шагов:

* Определение интерфейсов NAT
* Конфигурирование адресов интерфейсов
* Конфигурирование пула адресов NAT
* Сопоставление списков доступа пулу адресов

Первый шаг конфигурирования NAT состоит в назначении внутренних и внешних интерфейсов. Эта операция выполняется в режиме конфигурирования интерфейса. В следующем примере порт ethernet 0 конфигурируется в качестве внутреннего, а порт ISDN — в качестве внешнего интерфейса.

Router#\*configure terminal\*

Router(config)#\*interface ethernet 0\*

Router(config-if)#\*ip nat inside\*

Router(config-if)#\*interface bri 0

Router(config-if)#\*ip nat outside\*

Router(config-if)#\*^Z\*

Второй шаг конфигурирования NAT заключается в назначении выбранным интерфейсам соответствующих IP-адресов. Другими словами, внутренний интерфейс должен получить адрес во внутренней сети, а внешний — во внешней.

Обычно адрес для назначения внешнему интерфейсу предоставляется интернет-провайдером.

Используйте команды, подобные приведенным ниже, для установки внутреннего и внешнего адресов:

Router#\*configure terminal\*

Router(config)#\*interface ethernet 0\*

Router(config-if)#\*ip address 198.65.1.1 255.255.0.0\*

Router(config-if)#\*interface bri 0\*

Router(config-if)#\*ip address 186.91.108.1 255.255.0.0\*

Router(config-if)#\*^Z\*

Здесь интерфейсы маршрутизатора конфигурируются для работы с NAT. Однако маршрутизатор еще не знает, что ему делать с этой информацией. Ему необходимо указать пул адресов, подлежащих преобразованию. То есть маршрутизатору надо объяснить, по каким адресам он должен отправлять прибывающие данные.

Формирование адресного пула NAT выполняется в режиме глобального конфигурирования. Команда создания адресного пула NAT имеет такой формат:

#\*ip nat pool\* <name> <address range> netmask <subnet mask>

Приведенный ниже код иллюстрирует создание адресного пула NAT:

Router#\*configure terminal\*

Router(config)#\*ip nat pool my\_pool 198.65.1.1 198.65.254.254 netmask 255.255.0.0\*

Router(config)#\*^Z\*

Здесь использована команда nat\_pool для создания адресного пула с именем my\_pool, который будет использоваться при трансляции адресов. Данный пул использует адреса от 198.65.1.1 до 198.65.254.254. Наличие у адресных пулов имен позволяет назначать разные пулы разным интерфейсам.

Прежде чем использовать пул my\_pool для трансляции адресов, следует создать список доступа, содержащий правила обработки входящих в этот пул адресов. После этого при конфигурировании NAT можно будет указать, что адреса пула используются согласно правилам, установленным списком доступа. В данном примере список доступа просто разрешает трафик для адресов нашего пула. Список доступа создается командой следующего вида:

Router#\*configure terminal\*

Router(config)#\*access-list 1 permit 198.65.0.0 0.0.255.255\*

Router(config)#\*^Z\*

Последнее, что надо сделать, — это сопоставить список доступа адресному пулу NAT. Эта операция выполняется такой командой:

Router#\*configure terminal\*

Router(config)#\*ip nat inside source list 1 pool my\_pool\*

Router(config)#\*^Z\*

Команда nat inside source определяет, что весь входящий трафик, соответствующий правилам списка доступа 1, может транслироваться в адресный пул my\_pool. После того как все правила установлены, маршрутизатор готов к работе. Любой входящий пакет, запрещенный списком доступа, не обрабатывается NAT, а отбрасывается.

**Проверка знаний: Настройка протокола IP на маршрутизаторе Cisco**

**Задание 1**

Что делает команда «traceroute» ?

посылает пакеты со всех устройств, которые встречаются на пути продвижения к конечной цели

посылает эхо-пакеты с адреса назначения

посылает эхо-пакеты хаотичным способом во всех направлениях

**Задание 2**

В чем явный недостаток Telnet?

проблема в обеспечении безопасности

сложность в обслуживании

нестабильность в работе

**Задание 3**

Что означает символ «!», который представляет один пакет, в исполнении команды «ping»?

неудача

конечный узел недостижим

успех

**Задание 4**

В чем преимущество команды rlogin?

она поддерживает возможность отладки ввода

она поддерживает возможность отладки вывода

нет значительных преимуществ

**Задание 5**

Как необходимо настраивать Telnet?

как стандартный маршрутизатор

как линию передачи данных

как подсеть

**Задание 6**

Что называется *маршрутизатором последней надежды*?

маршрутизатор, на который все остальные маршрутизаторы сети пересылают пакеты, для которых не заданы маршруты

маршрутизатор, который всегда должен оставаться исправен при любых обстоятельствах

маршрутизатор, на котором завязана вся работа сетей

**Задание 7**

Что такое списки доступа IP?

избранные ip-адреса

метод безопасности

диапазон доступных ip-адресов

**Задание 8**

Почему NAT может служить эффективным инструментом безопасности?

изменяет адреса отправителя и получателя

изменяет адрес получателя

изменяет адрес отправителя