Содержание

I. Технологии построения сетей Ethernet

1. Основы коммутации второго уровня

Коммутатор – это устройство, функционирующее на втором уровне эталонной модели ISO/OSI и предназначенное для объединения сегментов сети, работающих на основе одного протокола Канального уровня. Коммутатор принимает входящий трафик через свои порты, но в отличие от концентратора, который передает исходящий трафик через все множество своих портов, коммутатор направляет трафик только через один порт, необходимый для достижения места назначения.

Так как коммутатор работает на втором уровне модели ISO/OSI, то он считывает и обрабатывает заголовки пакетов протокола Ethernet. Следовательно, коммутатор может выполнять как минимум следующие функции:

- перенаправление трафика (является обязательной функцией);
- проверка пакетов на ошибки;
- фильтрация трафика.

1.1. Перенаправление трафика (Traffic Forwarding)

Коммутатор принимает все пакеты с присоединенных сегментов. Для каждого получаемого пакета устройство считывает адрес получателя из заголовка протокола Ethernet, и если этот пакет предназначен для системы, расположенный в другом сегменте, передает пакет в этот сегмент и только в этот сегмент. Если пакет послан системе в локальном сегменте, коммутатор отбрасывает его, поскольку данные уже достигли своего места назначения. Чтобы эффективно выполнять перенаправление получаемых пакетов, мост должен знать, какие системы в каком сегменте находятся. Мост хранит эту информацию во внутренней адресной таблице перенаправления (Forwarding Table). Механизм заполнения данной таблицы называется прозрачной маршрутизацией (Transparent Routing).

1.2. Проверка пакетов на ошибки (Packet Error Checking)

Коммутатор может работать в двух основных режимах: без буферизации и с промежуточным хранением пакетов. Коммутатор без буферизации пакетов считывает только MAC-адрес входящего пакета, ищет его в своей таблице перенаправления и немедленно начинает передавать пакет через порт, обеспечивающий доступ к месту назначения. Единственным преимуществом данного режима является скорость передачи пакета, так как не тратится дополнительное время на сохранение целого пакета в оперативной памяти коммутатора и его дополнительную обработку.

Коммутатор с промежуточным хранением пакетов целиком сохраняет входящий пакет в буферной памяти прежде, чем передать его через порт назначения. Пока пакет находится в памяти, коммутатор проверяет его на наличие ошибок циклической контрольной суммы (CRC, Cyclic Redundancy Check) и выполнение других условий, таких как недопустимо малая или большая длина пакета, а также неправильная длительность передачи. Коммутатор немедленно отбрасывает любые пакеты с ошибками.

1.3. Дополнительные функции коммутаторов второго уровня

Функции перенаправления трафика и проверки пакетов на ошибки выполняют все современные коммутаторы, как управляемые, так и неуправляемые. Данные функции являются основой работы коммутаторов. Есть также дополнительные функции, которые присущи только управляемым коммутаторам. В зависимости от уровня сложности коммутатора варьируется количество дополнительных функций, которые он поддерживает. Большинство возможных дополнительных функций описаны в следующих разделах данной главы.

2. Управление потоком в полнодуплексном режиме (IEEE 802.3x Flow Control in full-duplex compliant)

Дуплексный режим работы требует наличия такой дополнительной функции, как управление потоком. Она позволяет принимающему узлу (например, порту сетевого коммутатора) в случае переполнение буфера дать узлу-источнику (например, файловому серверу) команду приостановить передачу кадров на некоторый промежуток времени. Управление осуществляется полностью на MAC-уровне с помощью кадра-паузы (pause frame), который автоматически формируется принимающей стороной. Если переполнение будет ликвидировано до истечения периода ожидания, то для того, чтобы восстановить передачу, отправляется второй кадр-пауза с нулевым значением времени ожидания.

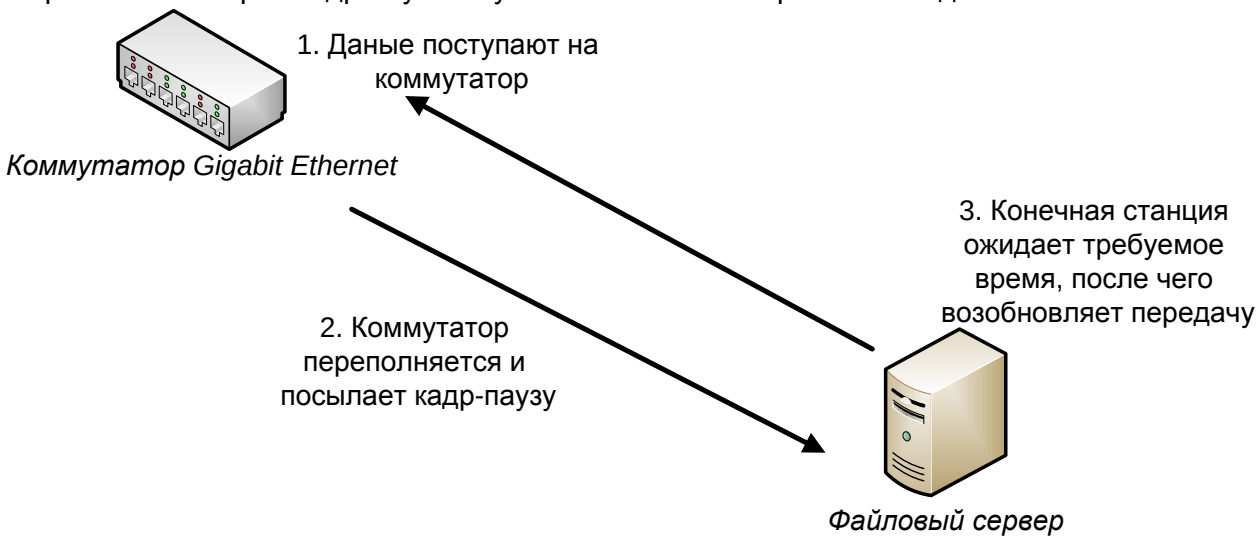


Рисунок 1.1.

Ниже приведён формат кадра-паузы (рисунок 1.2). Признаком кадра этого типа является наличие в поле «длина-тип» (LENGTH/TYPE) кода 0x8808, зарезервированного IEEE для кадров, которые используются в процедурах управления на уровне MAC. Поле «код операции» (OPCODE) содержит код 0x0001 – признак кадра управления потоком.

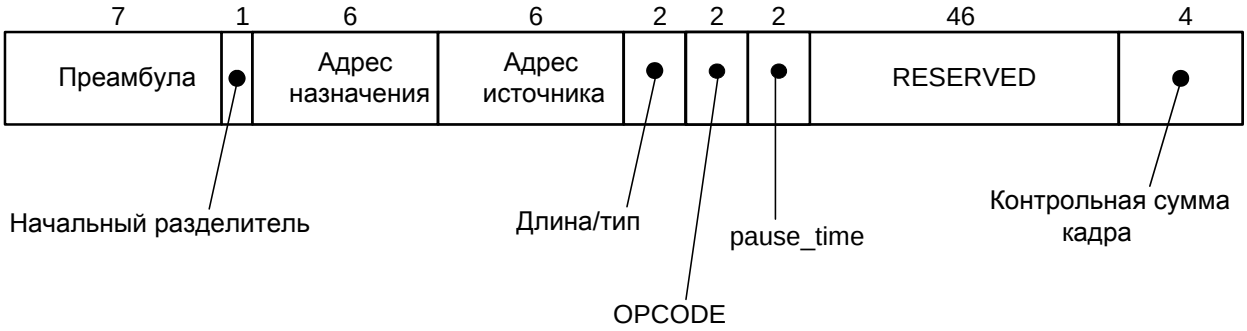


Рисунок 1.2. Формат кадра-паузы IEEE 802.3x.

В поле «адрес назначения» (Destination Address) кадра-паузы должен быть размещен код 01-80-C2-00-00-01, который представляет собой Multicast адрес станций, которые поддерживают выполнение данной процедуры, или Unicast адрес конкретного абонента в сети, формирующего избыточный трафик для данной станции. В поле «адрес источника» (Source Address) кадра-паузы помещается MAC-адрес станции, которая инициирует выполнение процедуры управления потоком. Поле «pause_time» содержит код, который соответствует размеру предлагаемой паузы, выраженному в 512 битовых интервалах (Bit Time, BT). Минимальное значение паузы равно 0, а максимальное значение равно 65535*512 = 33553920 BT. Таким образом, размер предлагаемой паузы для сети Fast Ethernet может иметь значение от 0 до 0,336 секунды. Остальные поля данного кадра зарезервированы для дальнейшего использования.

3. Зеркалирование портов (Port Mirroring)

Сегодня анализаторы сетевых протоколов эффективно используются ИТ-отделами и отделами информационной безопасности для решения широкого круга задач. С их помощью можно быстро определить причину медленной работы ИТ-сервиса или бизнес-приложения. Они позволяют документировать сетевую активность пользователей и использовать полученные данные, например, для определения источника утечки информации. Единственной «ложкой дегтя» является неудобство их использования в коммутируемых сетях. Дело в том, что анализатор протоколов должен «видеть» весь анализируемый трафик, в то время как в коммутируемой сети трафик между двумя портами коммутатора «невиден» на других его портах. Одним из способов решения данной проблемы является технология зеркалирования портов (Port Mirroring), которая поддерживается многими коммутаторами класса High End.

Зеркалирование позволяет копировать пакеты, получаемые и передаваемые с одного порта, на другой зеркальный порт (mirrored port). К зеркальному порту можно подключить устройство мониторинга (например, сетевой сниффер) для детального анализа пакетов, проходящих через первый порт.

4. Объединение портов в магистральные линии связи (Port Trunking)

Объединение портов в магистрали (Port Trunking) позволяет объединить несколько портов вместе для получения одного высокопроизводительного канала связи. Участвующие в объединение порты называются членами магистральной группы (trunk group), при этом один из портов назначается «якорем» группы. Так как все порты магистральной группы должны быть настроены одинаковым образом, то все настройки и дальнейшие их изменения копируются с порта-якоря на оставшиеся порты магистральной группы. Таким образом, необходимо сконфигурировать только порт-якорь. Коммутатор рассматривает все порты в магистральной группе как один порт. Пример применения технологии Port Trunking показан на рисунке 1.3.

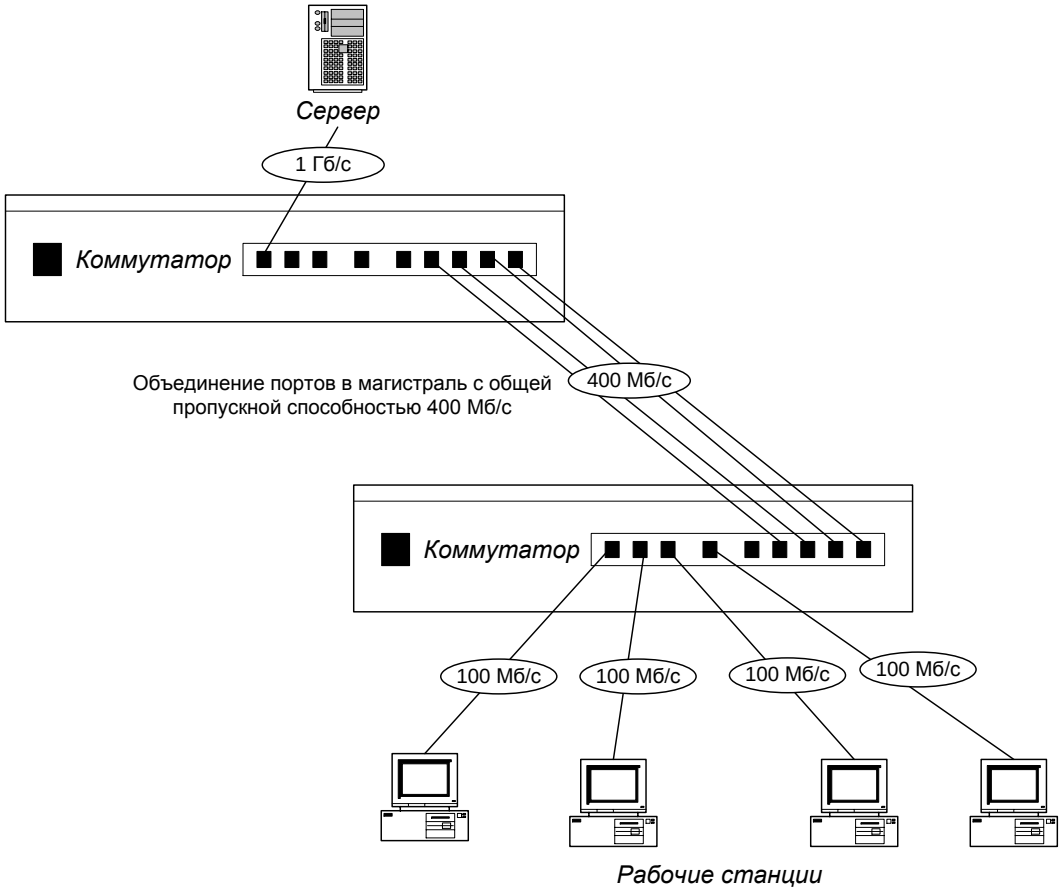


Рисунок 1.3. Применение технологии Port Trunking.

5. Виртуальные сети (Virtual LAN)

Виртуальная ЛВС (VLAN, Virtual LAN) – логическая группа компьютеров в пределах одной реальной ЛВС, за пределы которой не выходит любой тип трафика (широковещательный [broadcast], многоадресный [multicast] и одноадресный [unicast]). За счет использования VLAN администратор сети может организовать пользователей в логические группы независимо от физического расположения рабочих станций этих пользователей.

Основные цели введения виртуальных сетей в коммутируемую среду:

- повышение полезной пропускной способности сети за счет локализации широковещательного (broadcast) трафика в пределах виртуальной сети;
- повышения уровня безопасности сети за счет локализации одноадресного (unicast) трафика в пределах виртуальной сети;
- формирование виртуальных рабочих групп из некомпактно (в плане подключения) расположенных узлов;
- улучшение соотношения цены/производительности по сравнению с применением маршрутизаторов.

Классический пример, отражающий суть виртуальных сетей, приведен на рисунке 1.4. К одному коммутатору гипотетической фирмы подключены как машины бухгалтеров, так и машины бухгалтеров. При этом совершенно нет никакой необходимости во взаимодействие машин данных двух групп сотрудников. Поэтому машины бухгалтеров выделяются в одну виртуальную сеть, а машины инженеров в другую. При этом весь трафик (широковещательный, многоадресный и одноадресный) будет ограничен пределами своей виртуальной сети. Более того, повысится безопасность сети. Например, если на машине бухгалтера появится вирус «червь», то максимум он сможет заразить только машины бухгалтеров.

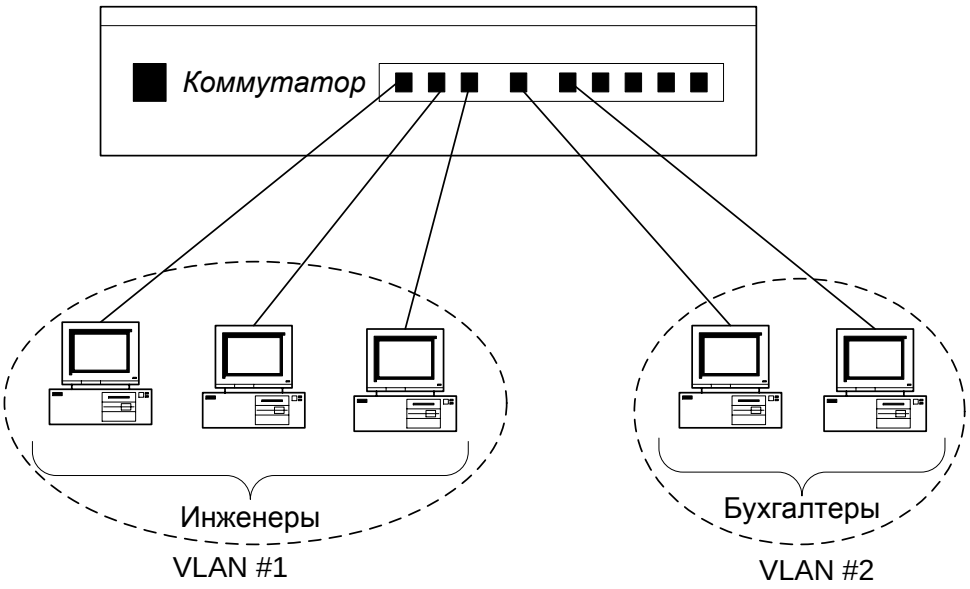


Рисунок 1.4. Пример использования технологии виртуальных сетей.

Сегодня существует достаточно много вариантов реализации VLAN. Простые варианты VLAN представляют собой набор портов коммутатора, более сложные реализации позволяют создавать группы на основе других критериев. В общем случае возможности организации VLAN тесно связаны с возможностями коммутаторов.

5.1. Сети на базе MAC-адресов (MAC-based VLANs)

Данный тип виртуальных сетей группирует устройства на основе их MAC-адресов. Для получения доступа в виртуальную сеть, устройство должно иметь MAC-адрес, который содержится в списке адресов данной виртуальной сети. Помимо прочего, отличительной особенностью данного типа виртуальных сетей является то, что они ограничивают только широковещательный трафик. Отсюда вытекает их название – широковещательные домены на базе MAC-адресов. Теоретически один MAC-адрес может являться членом не-

скольких широковещательных доменов, на практике данная возможность определяется функциональностью конкретной модели коммутатора.

Настройка виртуальной сети на основе MAC-адресов может отнять много времени. Представьте себе, что вам потребуется связать с VLAN адреса 1000 устройств. Кроме того, MAC-адреса "зашиты" в оборудование и может потребоваться много времени на выяснение адресов устройств в большой, территориально распределенной сети. Более того, существуют проблемы безопасности при работе с сетью на базе MAC-адресов. Злоумышленник может узнать MAC-адрес компьютера, входящего в ту или иную VLAN, назначить его своей сетевой карте (как минимум, это можно сделать стандартными средствами MS Windows XP) и успешно подключиться к сети.

С другой стороны, широковещательные домены на базе MAC-адресов позволяют физически перемещать станцию, позволяя, тем не менее, оставаться ей в одном и том же широковещательном домене без каких-либо изменений в настройках конфигурации. Это удобно в сетях, где станции часто перемещают, например, где люди, использующие ноутбуки, постоянно подключаются в разных частях сети.

5.2. Сети на базе портов (Port-based VLANs)

Устройства связываются в виртуальные сети на основе портов коммутатора, к которым они физически подключены. То есть каждой порт коммутатора включается в одну или более виртуальных сетей. К достоинствам данного типа виртуальных сетей можно отнести высокий уровень безопасности и простоту в настройке. К недостаткам можно отнести статичность данного типа виртуальных сетей. То есть при подключении компьютера к другому порту коммутатора необходимо каждый раз изменять настройки VLAN.

5.3. Сеть на базе маркированных кадров (IEEE 802.1Q VLANs)

В отличие от двух предыдущих типов виртуальных сетей VLAN на основе маркированных кадров могут быть реализованы на двух и более коммутаторах. В заголовок каждого кадра Ethernet вставляется маркер, который идентифицирует членство компьютера в определенной VLAN. На рисунке 1.5 показан пример такой VLAN.

Механизмы добавления идентифицирующего маркера в заголовок кадра Ethernet

Маркеры с номером VLAN в виртуальных сетях 802.1Q могут быть добавлены:

- явно, если сетевые карты поддерживают стандарт IEEE 802.1Q, и на этих картах включены соответствующие опции, то исходящие кадры Ethernet от этих карт будут содержать маркеры идентификации;
- неявно, если сетевые адаптеры, подключенные к этой сети, не поддерживают стандарт IEEE 802.1Q, то добавление маркеров выполняется на коммутаторе на основе группировки по портам.

Предположим, что некоторые порты коммутатора сгруппированы в VLAN. Коммутатор с поддержкой IEEE 802.1Q будет добавлять маркер к входящим на этот порт кадрам Ethernet с соответствующим ID VLAN. Но эти маркеры будут удаленны коммутатором из исходящих с этих же портов кадров, чтобы конечные компьютеры могли разобрать заголовок кадра Ethernet.

Если идентификация VLAN маркерами протокола 802.1Q была осуществлена обоими способами – явно и неявно, входящие кадры к портам коммутатора могут состоять из обоих типов кадров (с маркерами и без). В этой ситуации к неотмеченным входящим кадрам будут добавляться маркеры. В то время как маркированные кадры уже поддерживают членство в явно определенной виртуальной сети. Способность VLAN 802.1Q извлечения маркеров из заголовков кадров Ethernet позволяет VLAN работать с существующими коммутаторами и сетевыми адаптерами, которые не распознают маркеры. Способность добавления маркеров позволяет VLAN распространяться через множество 802.1Q-совместимых коммутаторов.

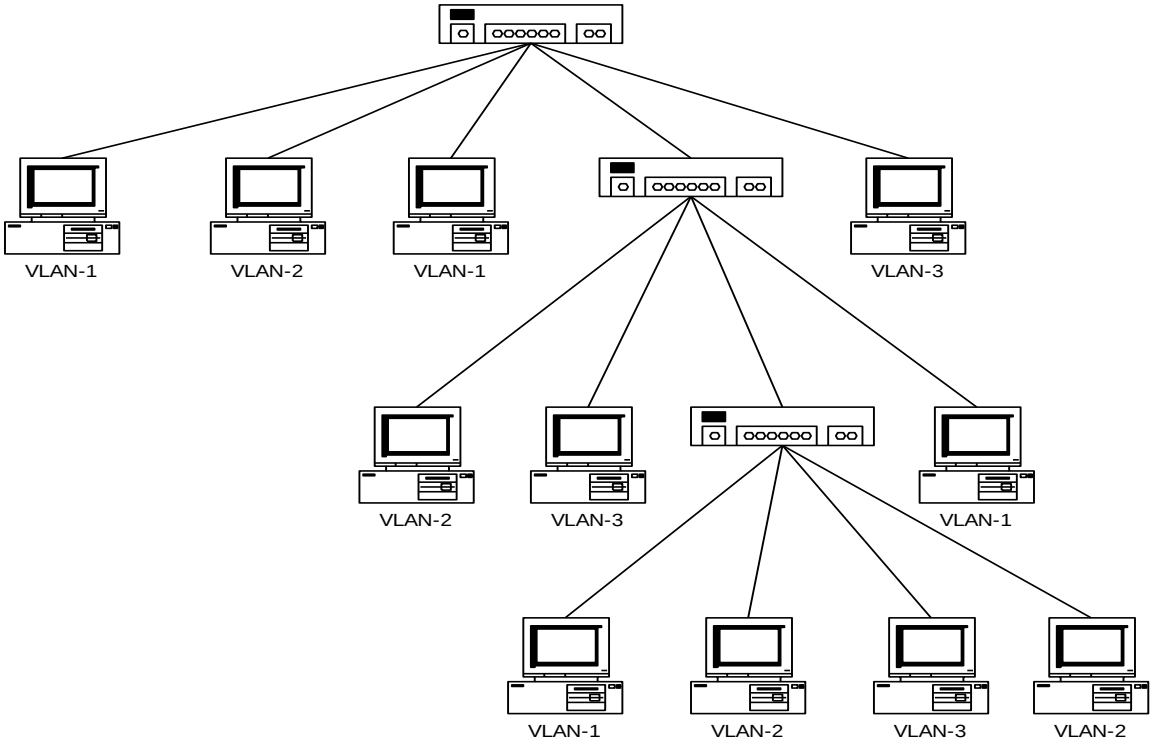


Рисунок 1.5. Виртуальная сеть на основе нескольких коммутаторов с использованием стандарта IEEE 802.1Q.

Компоненты конфигурации виртуальной сети на основе тэгов

Необходимо понять суть двух ключевых переменных для настройки VLAN 802.1Q:

- Port VLAN ID (PVID, идентификатор номера порта);
- VLAN ID (VID, идентификатор номера VLAN).

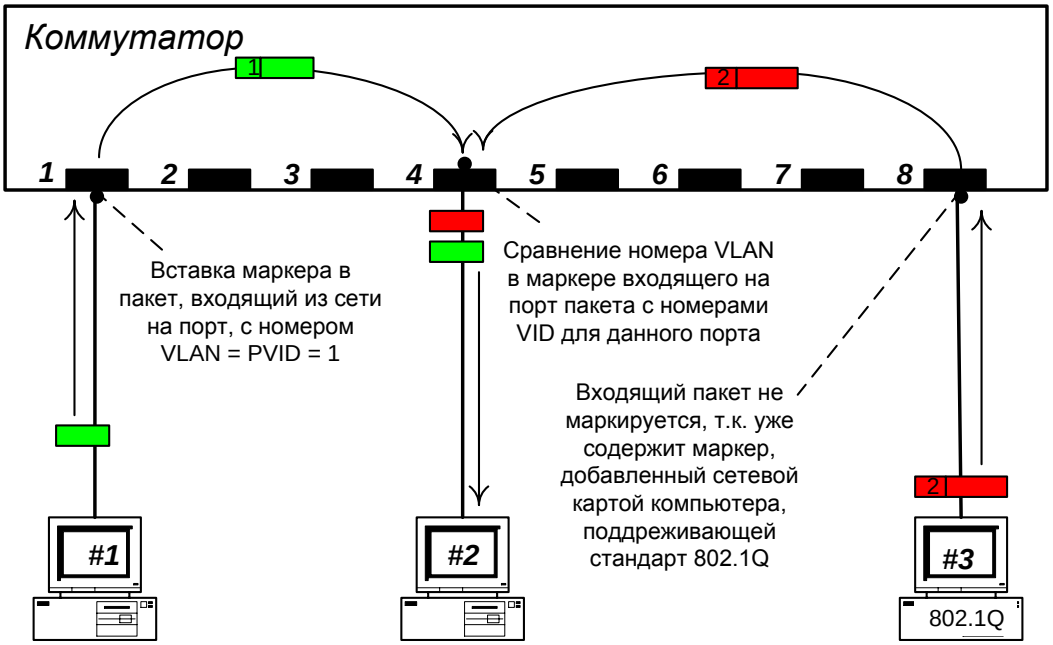
Обе переменные назначаются для одного и того же порта коммутатора, но между ними имеется существенная разница.

Пользователь может назначить ТОЛЬКО ОДИН номер PVID для каждого порта. Номер PVID определяет, к какой VLAN принадлежит данный порт. Когда ИЗ СЕТИ на порт коммутатора поступает НЕМАРКЕРОВАННЫЙ пакет, то принадлежность данного пакета к определенной VLAN осуществляется как раз на основе номера PVID порта. Номер PVID вставляется в заголовок кадра Ethernet и идентифицирует членство пакета в одной определенной VLAN. Необходимо отметить важный момент: если пакет уже содержит маркер (заданный явно сетевой картой компьютера или заданный неявно на другом коммутаторе), то повторная маркировка пакета, поступающего из вне на какой-либо порт коммутатора, НЕ производится.

С другой стороны можно задать МНОЖЕСТВО идентификаторов VID для одного и того же порта. Данные идентификаторы определяют, пакеты каких виртуальных сетей может принимать данный порт. Данная проверка осуществляется, когда пакет УЖЕ ПОЛУЧЕН ИЗ СЕТИ и МАРКИРОВАН, во время процедуры перенаправления пакета с порта на порт ВНУТРИ коммутатора.

Построение виртуальных сетей 802.1Q на одном коммутаторе

Рассмотрим следующую модель (рисунок 1.6), которая помогает понять принцип работы виртуальных сетей 802.1Q. В данном примере порт #1 является членом VLAN #1, а порты #4 и #8 являются членами VLAN #2. При этом порт #1 может принимать входящие пакеты только из своей VLAN, то есть только из первой VLAN. Порт #4 может принимать входящие пакеты из обоих VLAN (первой и второй). Порт #8, также как первый порт, может принимать входящие пакеты только из своей VLAN (второй). В результате полноценно функционировать в данной сети смогут только машины #2 и #3. Машины #1 и #2 не смогут обмениваться информацией, так как порт #4 может принимать пакеты с порта #1, но не наоборот.



PVID	Номер порта
1	1
2	4
2	8

VID	Номера портов
1	1,4
2	4,8

Рисунок 1.6. Модель функционирования виртуальной сети 802.1Q на одном коммутаторе.

Построение виртуальных сетей 802.1Q на нескольких коммутаторах

Как уже отмечалось, виртуальные сети 802.1Q могут быть построены на нескольких коммутаторах и охватывать всю локальную сеть. При этом все коммутаторы сети должны поддерживать стандарт 802.1Q. Необходимо понять следующие важные термины:

- Маркировка (Tagging) – процесс вставки информации о виртуальной сети 802.1Q в заголовок кадра Ethernet. Порт, для которого включен режим маркировки, будет оставлять без изменения заголовок ИСХОДЯЩЕГО С КОММУТАТОРА В СЕТЬ пакета Ethernet, то есть будет оставлять в нем маркер 802.1Q. Подобная возможность необходима для связи двух устройств, поддерживающих стандарт 802.1Q.
- Демаркировка (Untagging) – процесс удаления информации о виртуальной сети 802.1Q из заголовка кадра Ethernet. Порт, для которого включен режим демаркировки, будет удалять маркер из заголовка пакета, ИСХОДЯЩЕГО С КОММУТАТОРА В СЕТЬ. Подобная возможность необходима для связи коммутатора и устройства, не поддерживающего стандарт 802.1Q.
- Входящий порт (Ingress Port) – порт коммутатора, на который поступают пакеты из сети. Для каждого порта можно настроить фильтр Ingress Filter. Если этот фильтр отключен (состояние Disabled) то порт функционирует в обычном режиме, то есть проверка пакета на принадлежность какой-либо VLAN (сравнение номера VLAN пакета с номерами VID порта) производится при поступлении пакета на данный порт с другого порта ВНУТРИ коммутатора (как это показано на рисунке 1.7). При включении фильтра (состояние Enabled) данная проверка ДОПОЛНИТЕЛЬНО производится для любого маркированного пакета, входящего на данный порт ИЗ СЕТИ.
- Исходящий порт (Egress Port) – порт коммутатора, с которого пакеты выходят в сеть и при этом необходимо принять решение о маркировке/демаркировке пакета.

С учетом всего ниже приведена расширенная модель функционирования виртуальной сети 802.1Q на основе нескольких коммутаторов (рисунок 1.7).

Преимущества виртуальных сетей 802.1Q

Дополнительное преимущество VLAN 802.1Q заключается в том, что можно изменять топологию сети без физического перемещения станций или изменения кабельных соединений. Станции можно назначить другую VLAN и, соответственно, общаться и совместно использовать ресурсы с членами в новой VLAN просто изменив настройки порта с одной VLAN (например, VLAN отдела продаж) на другую (VLAN отдела маркетинга). Таким образом, VLAN обеспечивают гибкость при перемещениях, изменениях и наращивании сети. Следует отметить, что в современных коммутаторах поддерживается единственный тип VLAN – тэгированные виртуальные сети.

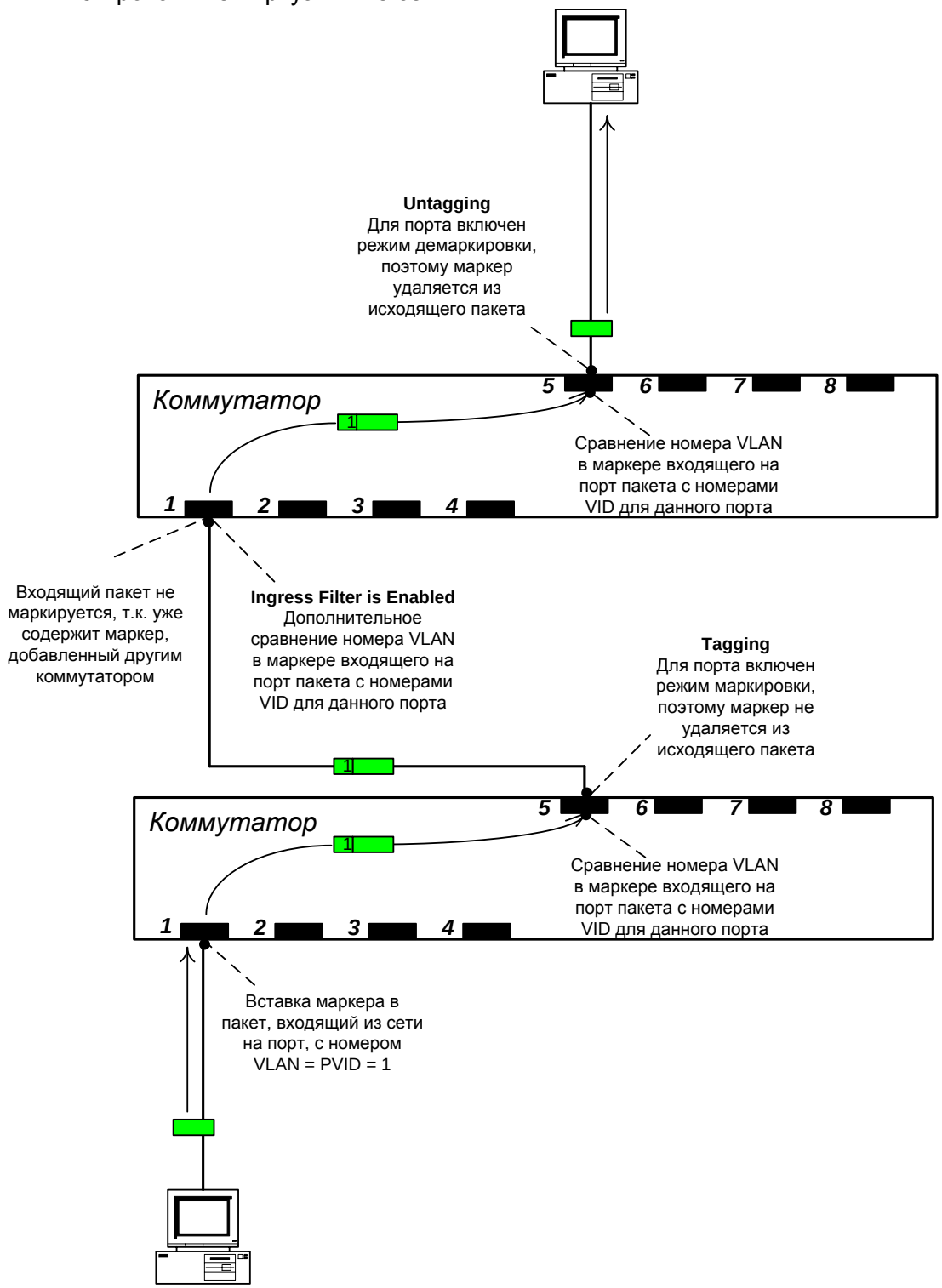


Рисунок 1.7. Модель функционирования VLAN 802.1Q на нескольких коммутаторах.

6. Протоколы связующего дерева (Spanning Tree Protocols)

Для обеспечения надежности работы сети зачастую необходимо использовать резервные линии связи. Базовые протоколы локальных сетей поддерживают только древовидные, то есть не содержащие замкнутых контуров, топологии связей. Это означает, что для организации альтернативных каналов требуются особые протоколы и технологии, выходящие за рамки базовых, к которым относится Ethernet. Для автоматического перевода в резервное состояние всех альтернативных связей, не вписывающихся в топологию дерева, в локальных сетях используются алгоритм связующего дерева (Spanning Tree Algorithm, STA) и реализующий его протокол (Spanning Tree Protocol, STP) и расширения последнего.

6.1. Алгоритм связующего дерева (Spanning Tree Algorithm, IEEE 802.1d)

Алгоритм связующего (*примечание:* иногда покрывающего, распределяющего) дерева предназначен для изменения структуры, построенной на коммутаторах вычислительной сети, таким образом, чтобы в ней отсутствовали циклы, и чтобы при этом сеть сохранила связность. Изначально алгоритм был разработан компанией Digital Equipment Corporation. Впоследствии этот алгоритм был взят за основу при разработке комитетом IEEE спецификации 802.1d, в соответствии с которой обеспечивалась автоматическое изменение структуры сети, построенной на коммутаторах.

Для того чтобы алгоритм STA мог выполняться, необходимо чтобы в сети коммутаторов были выполнены следующие предварительные установки:

- каждому коммутатору администратор должен назначить уникальный идентификатор, на основе которого будет получен приоритет коммутатора как комбинация MAC-адреса коммутатора и назначенного идентификатора;
- каждый порт коммутатора тоже должен иметь уникальный идентификатор;
- для каждого порта коммутатора должно быть определено значение относительной стоимости отправки кадров через данный порт, которое обычно обратно пропорционально пропускной способности подключенного канала.

После того, как были заданы указанные настройки, алгоритм STA предполагает выполнение нескольких последовательных этапов:

- выбор корневого моста;
- выбор корневых портов;
- выбор назначенных мостов;
- выбор назначенных портов;
- изменение конфигурации сети.

Алгоритм выбора корневого моста

Корневым в алгоритме STA называется мост, расположенный в вершине дерева, в которое преобразуется существующая структура сети. Алгоритм STA предписывает использование в качестве критерия выбора значение приоритета моста, которое представляет собой MAC адрес данного моста. Для того чтобы системный администратор мог управлять процессом выбора корневого коммутатора, он может назначать для каждого коммутатора уникальный идентификатор, который и является приоритетом. На рисунке 1.8 представлена схема сети, которая построена с использованием коммутаторов и содержит несколько петель. В качестве корневого моста в данном случае будет выбран мост №1.

Алгоритм выбора корневых портов

После определения корневого моста все оставшиеся мосты выполняют процедуру определения корневого порта (root port). Корневым считается тот из портов моста, который связан наиболее дешевым путем с корневым мостом. В процессе определения корневого порта определяется также значение стоимости корневого маршрута (root path cost). На рисунке 1.8 корневые порты мостов отмечены точкой. Число, размещенное около порта, соответствует рассчитанному значению стоимости корневого маршрута.

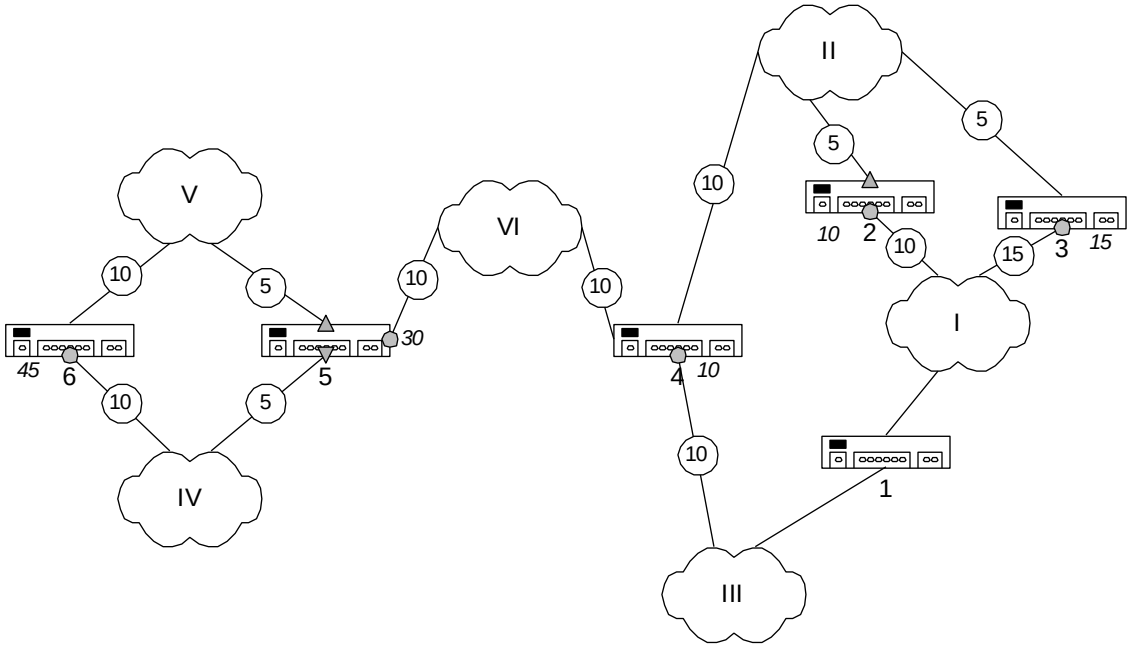


Рисунок 1.8. Пример работы алгоритма STA с сетью, содержащей петлевые структуры.

Стоимость маршрута иногда называют расстоянием. Именно по этому критерию выбирается единственный порт, соединяющий каждый коммутатор с корневым, и единственный коммутатор, соединяющий каждый сегмент сети с корневым коммутатором. Стоимость маршрута определяется как суммарное условное время на передачу одного бита данных от порта данного коммутатора до порта корневого коммутатора. При этом временем внутренних передач (с порта на порт) пренебрегают, учитывается только время передачи по сегментам сети. Условное время сегмента рассчитывается как время, затрачиваемое на передачу одного бита информации в 10 наносекундных единицах между непосредственно связанными по сегменту сети портами. Для сегмента Ethernet это время равно 10 условным единицам.

В том случае, если несколько портов моста имеют одинаковое значение величины корневого маршрута, в качестве корневого выбирается порт, имеющий меньший порядковый номер, то есть наибольший приоритет.

Алгоритм определения назначенных мостов и портов

После определения корневых портов выполняется процедура определения назначенного моста для сети. На рисунке 1.8 сегменты сети обозначены римскими цифрами. В качестве назначенного для сегмента сети выбирается мост, который имеет минимальное значение стоимости корневого маршрута. В том случае, если несколько мостов будут иметь одинаковое значение величины корневого маршрута, в качестве назначенного будет выбран мост, который имеет минимальное значение идентификатора, то есть максимальный приоритет. В частности, для рассматриваемого варианта сети, назначенным мостом для сегмента II будет выбран мост №2, которому соответствует такое же значение этой величины, что и мосту №4. Аналогичным образом мост №5 будет выбран назначенным мостом для сегментов V и VI.

Порт назначенного моста, который используется для подключения к выбранному сегменту сети, называется назначенным портом. На рисунке 1.8 назначенные порты отмечены треугольником.

У корневого моста все порты являются назначенными, а их расстояние до корня полагается равным нулю. Корневого порта у корневого моста нет.

Изменение конфигурации сети по результатам выполнения алгоритма STA

После того, как были определены назначенные мосты и порты, может быть выполнена процедура изменения структуры сети. В соответствии с этой процедурой, все порты мос-

тов, которые не получили статус корневых или назначенных, должны быть переведены в заблокированное состояние. В соответствие с этой процедурой должны быть заблокированы все порты мостов №3, 4 и 6 (рисунок 1.9). Также должны быть заблокированы порты, которые не получили статус корневых или назначенных, на корневом и назначенных мостах.

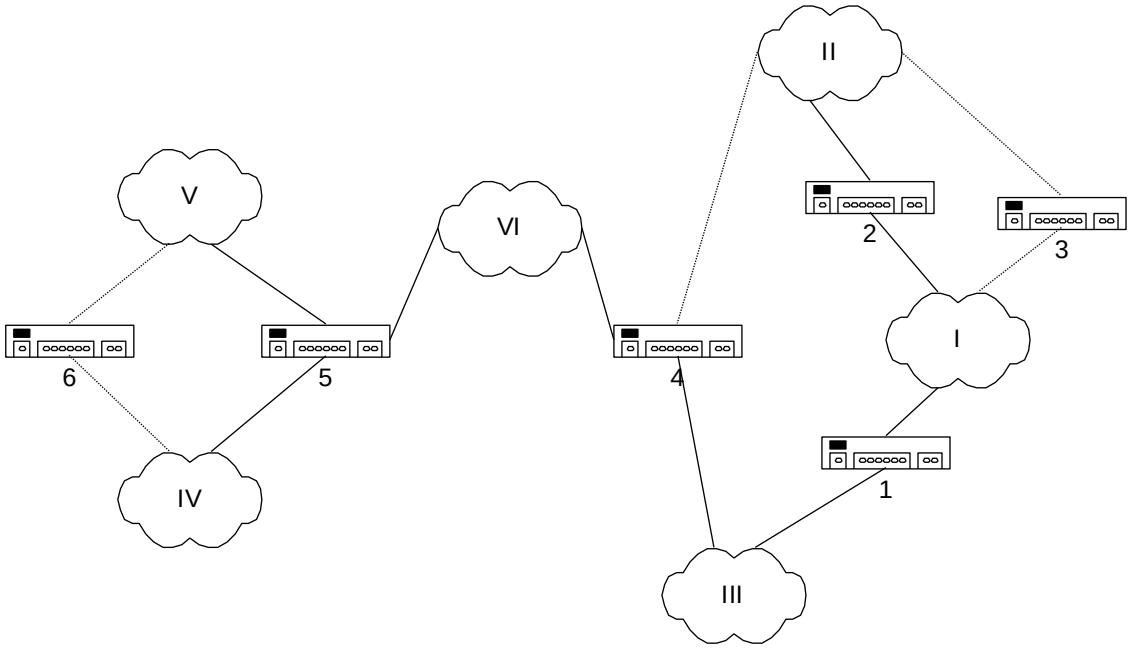


Рисунок 1.9. Топология сети после применения алгоритма связующего дерева.

Протокол связующего дерева (Spanning Tree Protocol)

Для активизации протокола STP администратор должен активировать протокол, а затем задать следующие *обязательные настройки* на каждом коммутаторе в сети:

1. Уникальный идентификатор коммутатора, на основе которого будет получен приоритет коммутатора. Чем меньше значение идентификатора, тем выше будет приоритет коммутатора.
2. Уникальный идентификатор для каждого порта коммутатора. Чем меньше значение идентификатора, тем выше будет приоритет порта.
3. Стоимость отправки кадров через каждый порт коммутатора. Обычно коммутатор автоматически вычисляет стоимость для каждого собственного порта на основе пропускной способности подключенной к порту линии связи.

Также в зависимости от модели коммутатора администратор может дополнительно вручную задать значение следующих переменных:

1. Интервал приветствия Bridge Hello Time – это интервал между двумя посылками специального пакета BPDU (см. ниже), рассылаемого корневым коммутатором для оповещения остальных коммутаторов о том, что корневой коммутатор функционирует.
2. Максимальный возраст коммутатора Bridge Max. Age. Если пакет BPDU не будет получен от корневого коммутатора по истечении максимального времени жизни, то ваш коммутатор будет считать, что корневой коммутатор перестал функционировать, то есть произошли изменения в конфигурации сети, а затем начнет посылать собственный BPDU всем другим коммутаторам для разрешения стать корневым мостом.

   Замечание: необходимо на всех коммутаторах сети значение параметра Bridge Max. Age выставлять большим значения параметра Bridge Hello Time (см. формулы ниже), иначе постоянно будут происходить ошибки конфигурации.
3. Задержка смены состояния коммутатора Bridge Forward Delay – это время, которое затрачивается не перевод любого порта из режима блокировки (blocking state) в режим функционирования (forwarding state).

Для нормального функционирования протокола STP приведенные параметры должны удовлетворять следующему условию:

$$2*(HelloTime+1) \le MaxAge \le 2*(ForwardDelay-1)$$

После активации и осуществления необходимых настроек администратором протокол STP начинает функционировать. Чтобы выполнить все шаги, заложенные в алгоритме STA, коммутаторы, работающие по протоколу STP, обмениваются между собой сообщениями в формате, определенном стандартом IEEE 802.1d. Эти сообщения называются блоками данных протокола мостов (BPDU, Bridge Protocol Data Unit) и содержат следующие поля:

- Идентификатор версии протокола (2 байта). Коммутаторы должны поддерживать одну и ту же версию протокола STP, иначе может установиться активная конфигурация с петлями.
- Тип сообщения (1 байт). Существует два типа сообщений BPDU – конфигурационный BPDU, то есть заявка на возможность стать корневым коммутатором, на основании которой происходит определение активной конфигурации, и BPDU уведомления о реконфигурации, которое посылается коммутатором, обнаружившим событие, требующее проведения реконфигурации - отказ линии связи, отказ порта, изменение приоритетов коммутатора или портов.
- Флаги (1 байт). Содержит два 1-битных флага:
  - Бит 0. Topology Change (Изменение топологии) – указывает на то, что сообщение было послано для того, чтобы сигнализировать об изменении в составе сети (сообщение реконфигурации);
  - Бит 7. Topology Change Acknowledgment (Подтверждение изменения топологии) – используется для подтверждения получения сообщения реконфигурации с установленным битом 0.
- Приоритет корневого коммутатора (8 байт) содержит в качестве 6 младших байт MAC-адрес корневого коммутатора и в качестве старших 2 байт идентификатор корневого коммутатора, назначенный администратором.
- Стоимость корневого пути (2 байта).
- Приоритет коммутатора (8 байт) содержит 6-байтовый аппаратный адрес моста, пославшего сообщение, и 2-байтовое значение заданного этому мосту идентификатора.
- Идентификатор порта (2 байта). Определяет порт, через который было послано сообщение.
- Возраст сообщения (2 байта). Указывает на время, прошедшее с момента отправки сообщения. Измеряется в единицах времени, кратных 0,5 секунды. Каждый коммутатор добавляет ко времени жизни пакета свое время задержки.
- Максимальный возраст сообщения (2 байта). Задает ограничение возраста, по достижении которого сообщение должно быть удалено.
- Интервал приветствия Bridge Hello Time (2 байта). Фиксирует временной интервал между конфигурационными сообщениями корневого моста.
- Задержка смены состояний Bridge Forward Delay (2 байта). Специфицирует промежуток времени, в течение которого коммутаторы должны ожидать завершения работы алгоритма STA после изменения топологии сети. Если еще не все коммутаторы закончили работу алгоритма, то преждевременные передачи данных могут вызвать появление циклов.

Сообщения BPDU инкапсулируются стандартными кадрами протокола Канального уровня. Желательно, чтобы все коммутаторы поддерживали общий групповой адрес, с помощью которого кадры, содержащие пакеты BPDU, могли одновременно передаваться всем коммутаторам сети. В противном случае пакеты BPDU рассылаются широковещательно. Пакеты BPDU не перенаправляются в другие сети.

Жизненный цикл протокола STP можно представить в виде следующей блок-схемы.

Рисунок 1.10.

Подробно рассмотрим каждый из этапов цикла.

Процесс конфигурации

Выбор корневого моста. После инициализации каждый коммутатор сначала считает себя корневым. Поэтому он начинает через интервал приветствия Bridge Hello Time генерировать через все свои порты сообщения BPDU конфигурационного типа. В них он указывает свой приоритет в качестве приоритета корневого коммутатора, расстояние до корня устанавливается в 0, а в качестве идентификатора порта указывается приоритет того порта, через который передается BPDU. Как только коммутатор получает BPDU, в котором имеется приоритет корневого коммутатора, больше его собственного, он перестает генерировать свои собственные кадры BPDU, а начинает ретранслировать только кадры нового претендента на звание корневого коммутатора. При ретрансляции кадров он наращивает расстояние до корня, указанное в пришедшем BPDU, на условное время сегмента, по которому принят данный кадр.

Выбор корневого порта. При ретрансляции кадров каждый коммутатор для каждого своего порта запоминает минимальное расстояние до корня. При завершении процедуры установления конфигурации покрывающего дерева, каждый коммутатор находит свой корневой порт – это порт, который ближе других портов находится к корню дерева.

Выбор назначенных мостов и портов. Кроме этого, коммутаторы выбирают для каждого сегмента сети назначенный порт. Для этого они исключают из рассмотрения свой корневой порт. Далее для всех своих оставшихся портов сравнивают принятые по ним минимальные расстояния до корня. Порт с минимальным расстоянием является назначенным портом. Когда имеется несколько портов с одинаковым кратчайшим расстоянием до корневого коммутатора, выбирается порт с наименьшим идентификатором (следовательно, с наибольшим приоритетом). Все порты, кроме назначенных и корневого, переводятся в заблокированное состояние и на этом построение покрывающего дерева заканчивается.

Функционирование сети

В процессе нормальной работы корневой коммутатор продолжает генерировать служебные пакеты BPDU через интервал Bridge Hello Time, а остальные коммутаторы продолжают их принимать своими корневыми портами и ретранслировать назначенными.

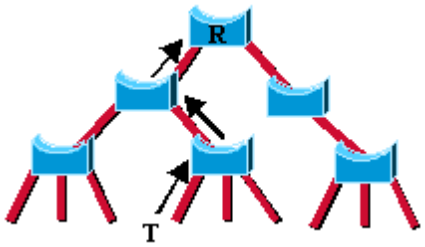
Возникновение события, требующего проведение реконфигурации

В процессе функционирования сети с установившейся древовидной топологией следующие события приводят к инициализации новой процедуры построения покрывающего дерева:

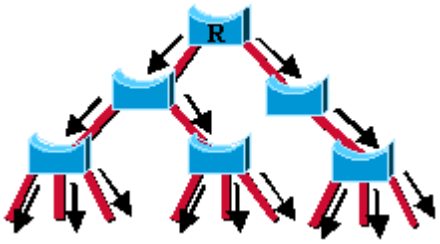
- если по истечении максимального времени жизни корневой порт любого коммутатора сети не получит пакет BPDU от корневого коммутатора.
- если пакет BPDU, генерируемый корневым коммутатором, будет получен любым не корневым портом любого назначенного коммутатора.

Рассылка сообщений о реконфигурации

Коммутатор, обнаруживший одно из вышеперечисленных событий, в первую очередь оповещает об этом корневой коммутатор (рисунок 1.11).



Изменение топологии произошло в точке Т. Шаг 1. Уведомление об изменение топологии (TCN) посылается корневому коммутатору



Шаг 2. Корневой коммутатор оповещает об этом остальные коммутаторы.

Рисунок 1.11.

Как только корень узнает о происшедшем изменении, он выставляет флаг TC в сообщениях BPDU, которые он рассылает. Затем данные сообщения распространяются по всей сети. В этом случае граф сети выстраивается заново в соответствии с вышеописанным алгоритмом.

У пакета BPDU с уведомлением о реконфигурации отсутствуют все поля, кроме первых двух.

Состояния портов коммутатора

Таким образом, в процессе построения топологии сети каждый порт коммутатора проходит несколько стадий:

- Blocking (Заблокирован). При инициализации коммутатора все порты (за исключением отключенных) автоматически переводятся в состояние "Заблокирован". В этом случае порт генерирует, принимает, обрабатывает и ретранслирует только пакеты BPDU. Все остальные пакеты не передаются.
- Listening (Прослушивание). В начальный момент работы алгоритма STA порты коммутатора переходят в состояние "Прослушивание". В этот момент пакеты BPDU от других коммутаторов еще не получены и коммутатор считает себя корневым, а все свои порты – назначенными. В этом режиме порт может находиться до истечения таймера смены состояний (Forwarding Timer). Значение таймера берется из значения переменной Bridge Forward Delay и может устанавливаться администратором (см. выше). В этом режиме порт продолжает генерировать, принимать, обрабатывать и ретранслировать только пакеты BPDU. Если в течение этого времени порт получит BPDU с лучшими параметрами, чем собственные (расстояние, идентификатор коммутатора или порта), то он перейдет в состояние "Заблокирован". В противном случае порт переводится в состояние "Обучение".
- Learning (Обучение). Порт начинает принимать пакеты и на основе адресов источника строить таблицу коммутации. Порт в этом состоянии все еще не продвигает пакеты. Порт продолжает участвовать в работе алгоритма STA, и при поступлении BPDU с лучшими параметрами переходит в состояние Blocking "Заблокирован".

- Forwarding (Продвижение). Только после двукратной выдержки по таймеру порт переходит в состояние “Продвижение” и обрабатывает пакеты данных в соответствии с построенной таблицей
- Disable (Отключен). В это состояние порт переводит администратор. Отключенный порт не участвует ни в работе протокола STP, ни в продвижении пакетов данных. Порт можно также вручную включить и он сначала перейдет в состояние Blocking.

Временная диаграмма состояния портов приведена ниже.

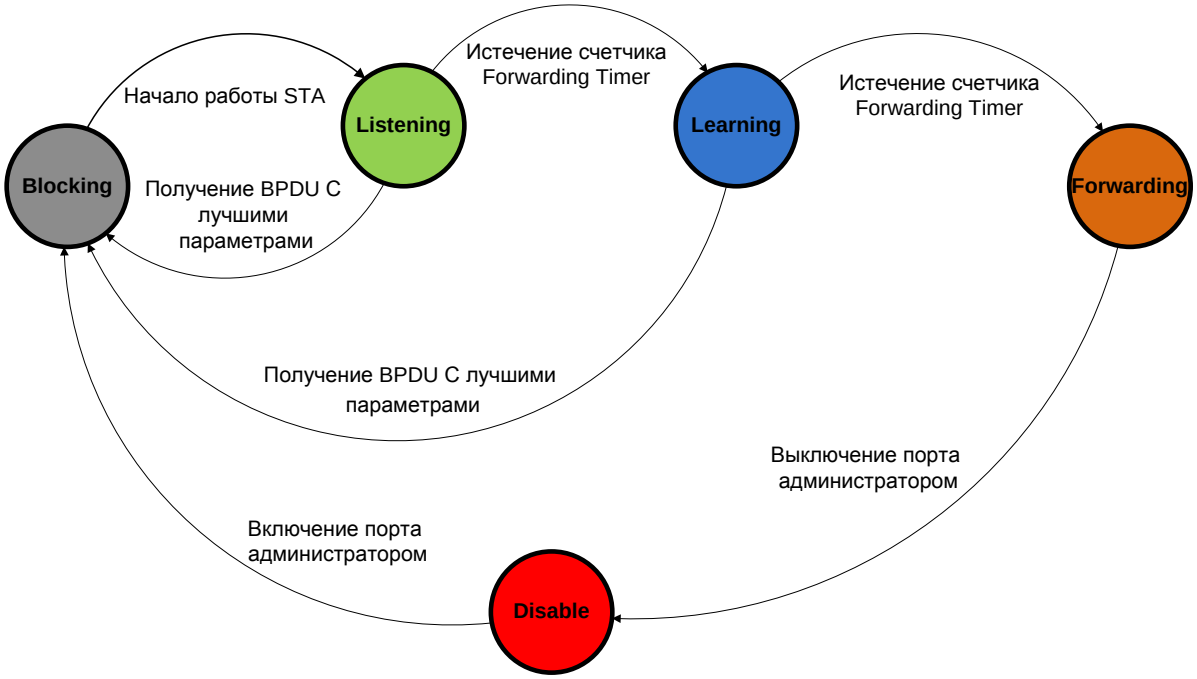


Рисунок 1.12.

Недостатки протокола STP:

Можно выделить следующие основные недостатки протокола STP:

1. Медленное время восстановления (сходимости) сети после аварии. При использовании настроек по умолчанию, это время может достигать нескольких минут в небольшой сети для восстановления после простого обрыва соединения. Пока идет процесс восстановления, пользователи оторваны от сети, и большинство приложений закрывают свои сессии по тайм-ауту до того, как работа сети восстановится. С точки зрения пользователей это большое неудобство, так как они будут вынуждены переустанавливать все сессии, открытые приложениями.
2. Большое количество соединений в сети, использующей STP, находится в заблокированном состоянии и не передает данные. Таким образом, значительная часть пропускной способности сети не используется.

Данные недостатки призваны устранить следующие поколения протокола STP – Rapid STP и Multiple STP, которые рассмотрены ниже.

6.2. Алгоритм быстрого связующего дерева (Rapid Spanning Tree Algorithm, IEEE 802.1w)

Стандарт на протокол связующего дерева IEEE 802.1d Spanning Tree Protocol был разработан в 1983 году, когда время восстановления сети после сбоя в течении 1 минуты считалось достаточно приемлемым. Протокол IEEE 802.1w Rapid Spanning Tree (RSTP) был предложен многими производителями для уменьшения времени простоя клиентских портов во время процедуры восстановления сети, использующей STP. Протокол RSTP может рассматриваться скорее как эволюция протокола STP, нежели как революция. Терминология стандарта IEEE 802.1d в основном осталась прежней. Большинство параметров остались без изменений. Таким образом, пользователи, хорошо знакомые со стандартом IEEE 802.1d, смогут быстро понять и сконфигурировать сеть на основе нового стандарта IEEE 802.1w. Стандарт IEEE 802.1w совместим со стандартом IEEE 802.1d, но при этом теряются достоинства протокола RSTP.

Основные отличия стандарта IEEE 802.1w от стандарта IEEE 802.1d

Отличие 1. Состояния портов коммутатора

Согласно стандарту на протокол RSTP существует только три состояния порта:

- Discarding (Отвергающий). Состояния «Отключен», «Заблокирован» и «Прослушивание» стандарта IEEE 802.1d объединены в стандарте IEEE 802.1w в одно состояние «Отвергающий».
- Learning (Обучение).
- Forwarding (Продвижение).

Таким образом, временная диаграмма состояния портов для протокола RSTP выглядит следующим образом.

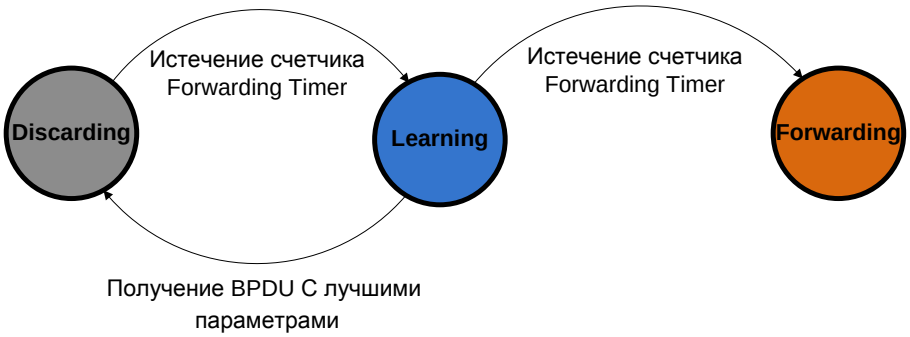


Рисунок 1.13.

Отличие 2. Роли портов

Так же как и в протоколе STP остались роли корневого и назначенного порта, но протокол RSTP вводит еще две дополнительные роли – альтернативный порт (alternative port) и резервный порт (backup port). Протокол STP определяет роль порта на основе пакетов BPDU. Грубо говоря, всегда существует метод для сравнения любых двух пакетов BPDU для того, чтобы решить, какой из них более приоритетный. Данное сравнение осуществляется на основе значений, записанных в поля «Стоимость корневого пути», «Приоритет коммутатора» и «Идентификатор порта». Корневые и назначенные порты в протоколе RSTP определяются и назначаются точно также, как и в протоколе STP. Поэтому имеет смысл показать, каким образом назначаются альтернативные и резервные порты.

Альтернативный порт. Альтернативный порт является заблокированным портом, который получает более приоритетные пакеты BPDU от другого моста (рисунок 1.14).

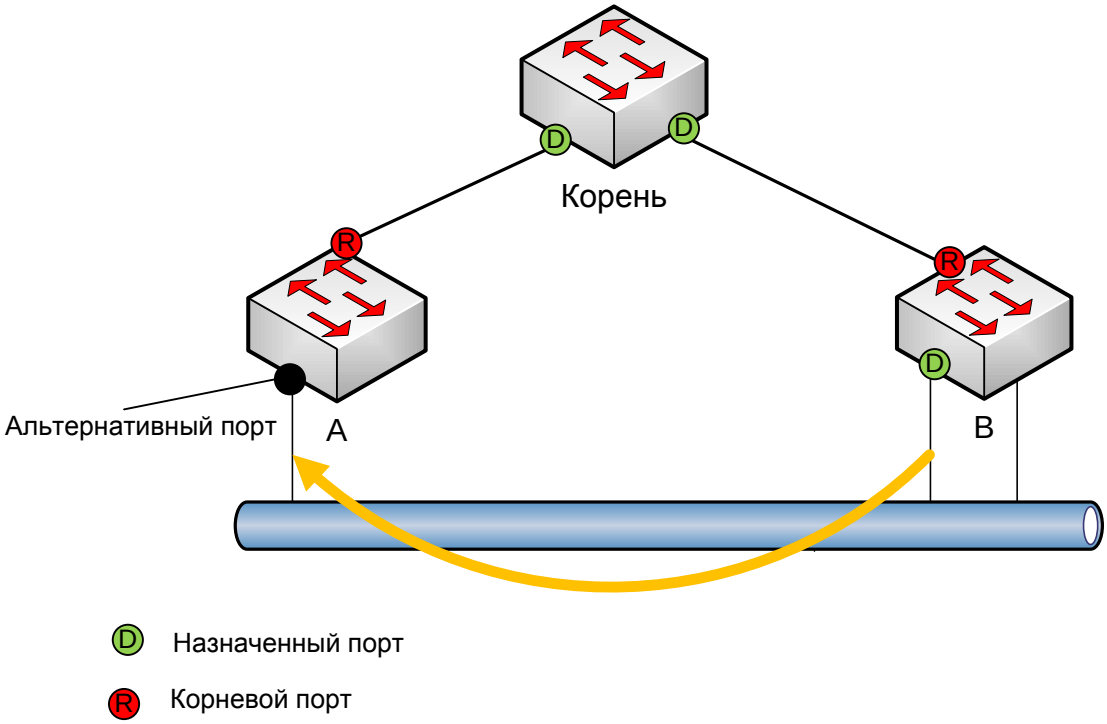


Рисунок 1.14.

Резервный порт. Резервный порт является заблокированным портом, который получает более приоритетные пакеты BPDU от своего же моста (рисунок 1.15).

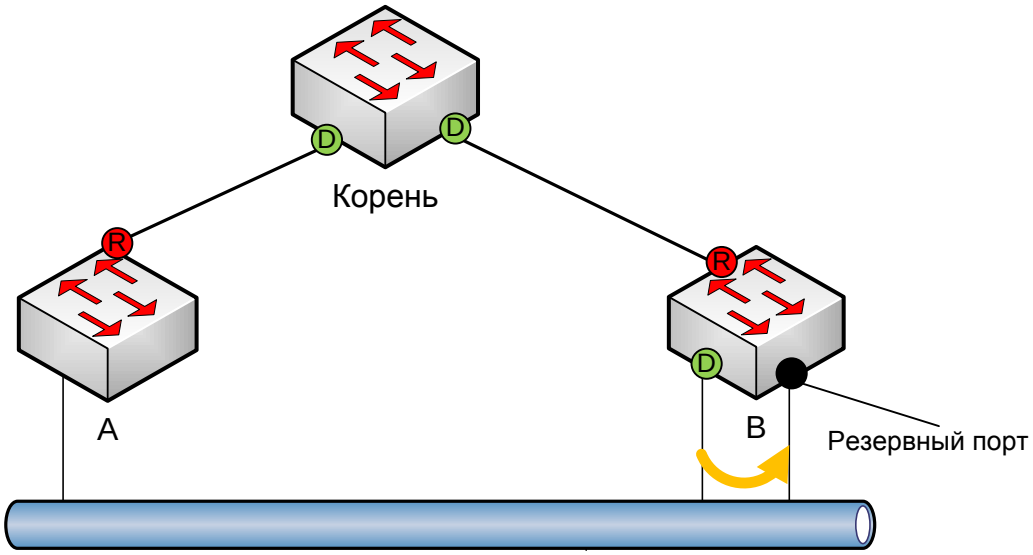


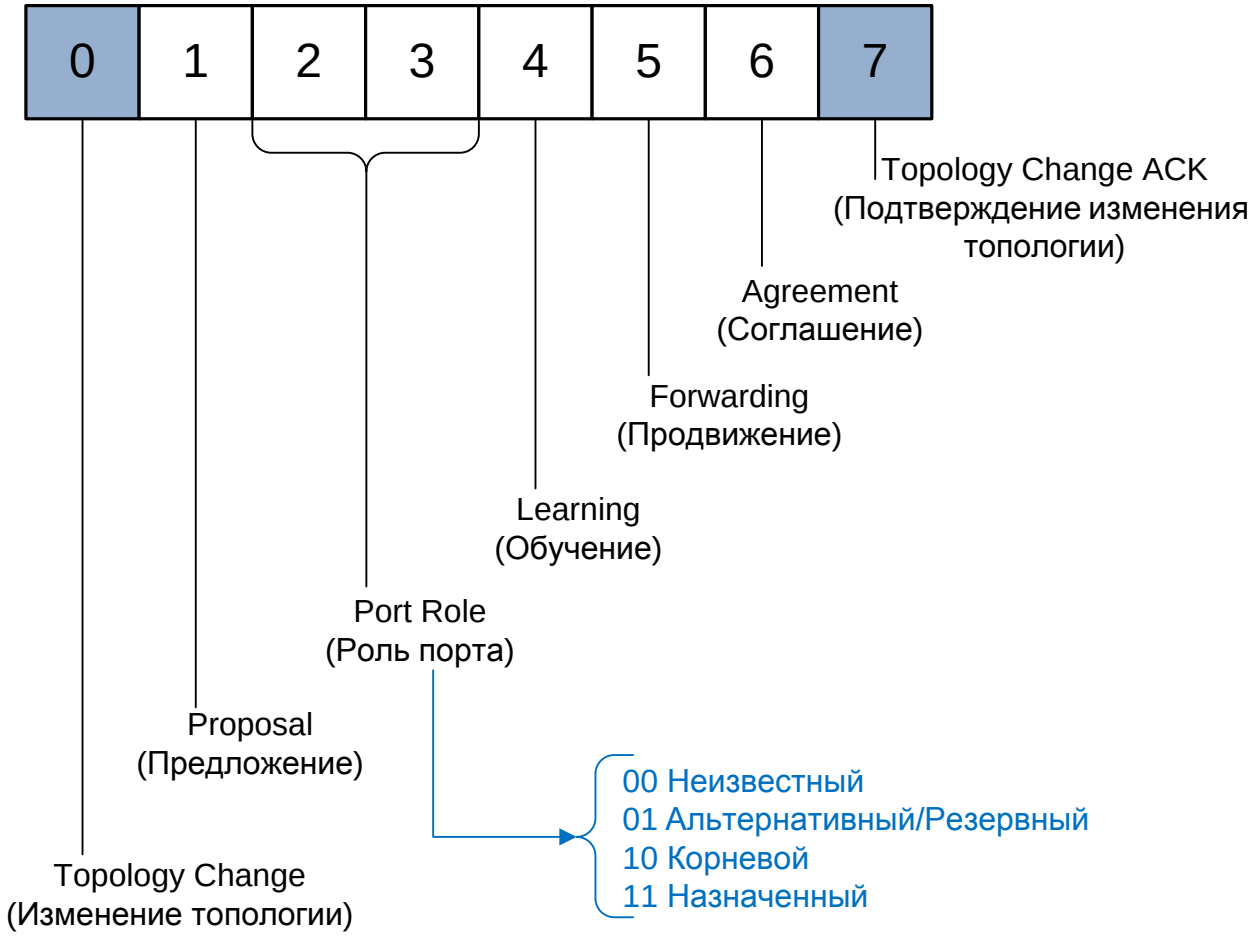
Рисунок 1.15.

Альтернативный порт обеспечивает альтернативный маршрут к корневому мосту и, следовательно, может заменить текущий назначенный коммутатор. Резервный порт обеспечивает всего лишь дополнительный канал к корневому мосту.

Отличие 3. Новый формат пакета BPDU

Некоторые изменения затронули и формат пакета DPDU. В стандарте IEEE 802.1d определено всего два флага (см. выше). Стандарт IEEE 802.1w использует все 8 бит поля «Флаги» пакета BPDU. Новые флаги обеспечивают выполнение двух основных функций:

- кодирование роли и состояния порта, который сгенерировал пакет BPDU;
- осуществление механизма «предложение/соглашение».



Другое важное изменение, привнесенное стандартом IEEE 802.1w в формат пакета DPDU, – это значение поля «Идентификатор версии протокола» содержит значение 2. Мосты, работающие по стандарту IEEE 802.1d, будут отбрасывать пакет данного типа.

Отличие 4. Новый алгоритм работы с пакетами BPDU
Пакеты BPDU рассылаются с интервалом Bridge Hello Time всеми коммутаторами (а не только корневым) и больше не ретранслируются. Согласно стандарту IEEE 802.1d пакеты BPDU генерировал только корневой коммутатор с интервалом *Hello Time*, который может задаваться администратором сети. При этом остальные не корневые коммутаторы только ретранслировали пакеты BPDU, пришедшие на их корневой порт. Стандарт IEEE 802.1w предусматривает, что все коммутаторы с интервалом Hello Time рассылают пакеты BPDU с собственной информацией через все свои порты, даже если они не получили пакет BPDU от корневого моста. Таким образом, после установления устойчивой конфигурации сети (этап «Функционирование сети», см. выше «Жизненный цикл протокола STP») служебный трафик протокола RSTP несколько больше, чем у протокола STP. Учитывая пропускную способность каналов современных ЛВС, это вообще нельзя считать недостатком.

Более быстрое устаревание информации. В протоколе RSTP следующие события могут привести к инициализации новой процедуры построения покрывающего дерева:

- если по истечении максимального времени жизни корневой порт любого коммутатора сети не получит служебный пакет BPDU от корневого коммутатора (как и в протоколе STP);
- если пакеты BPDU не будут получены три раза подряд (три интервала Hello Time) от соседних коммутаторов.

Таким образом, теперь пакеты BPDU используются в качестве механизма определения работоспособности подключенной линии связи или коммутатора (keep-alive mechanism). Коммутатор, не получивший пакет BPDU от другого коммутатора три раза подряд, счита-

ет, что связь с другим коммутатором потеряна. Подобный механизм позволяет быстрее определять неисправность в сети и запускать процедуру реконфигурации.

Обработка «худших» пакетов BPDU. Рассмотрим следующую ситуацию (рисунок 1.16). В установившейся топологии сети происходит разрыв связи между корневым коммутатором и коммутатором B. Следовательно, не получив пакет BPDU от корневого коммутатора в течении времени Max Bridge Age, коммутатор B рассылает сообщение о реконфигурации, а затем конфигурационный пакет BPDU. В случае протокола STP всё это приводит к реконфигурации всей сети. При использовании же протокола RSTP события будут развиваться иначе.

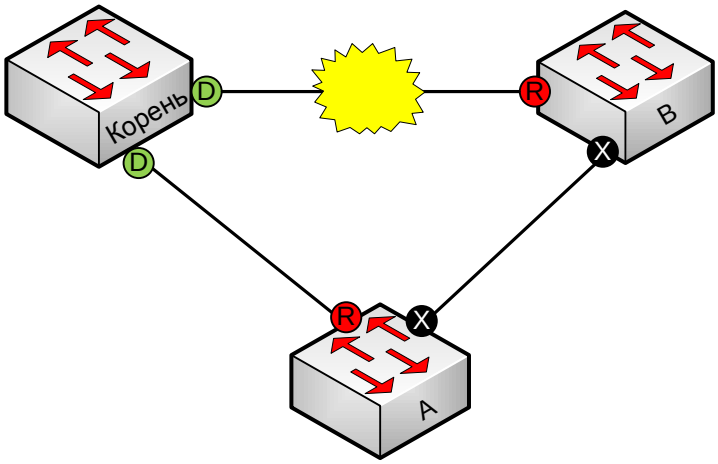


Рисунок 1.16.

Коммутатор A продолжает получать пакеты BPDU от корневого коммутатора, приоритет которых выше, чем пакеты BPDU коммутатора B. Поэтому коммутатор A сразу пересылает пакет BPDU корневого коммутатора коммутатору B. Следовательно, коммутатор B прекращает генерировать пакеты BPDU и определяет порт X как корневой.

Быстрый переход портов в состояние «Продвижения»
Быстрая конфигурация сети является самым важным нововведением в стандарте IEEE 802.1w. В сетях, в которых применяется классический протокол STP, уменьшение времени сходимости можно достичь только за счет уменьшения значений параметров Bridge Forward Delay и Bridge Max. Age, что зачастую приводит к нестабильности в работе протокола и сети. Для того, чтобы достичь более быстрой сходимости сети, протокол RSTP вводит два новых типа портов – пограничный порт (edge port) и порт точка-точка (point-to-point port, P2P port).

Пограничные порты
Пограничным портом является порт, непосредственно подключенный к сегменту, в котором не могут быть созданы петли. Например, порт непосредственно подключен к рабочей станции. Порт, который определен как пограничный, мгновенно переходит в состояние продвижения, минуя состояние обучения. Пограничный порт теряет свой статус и становится обычным портом связующего дерева в том случае, если получит пакет BPDU.

P2P порт
P2P порт обычно используется для подключения к другим мостам и также способен быстро перейти в состояние продвижения. При работе RSTP все порты, функционирующие в полнодуплексном режиме, рассматриваются как порты P2P, до тех пор, пока не будут переконфигурированы вручную.

Сходимость сети
При использовании протокола STP.
Рассмотрим пример, отражающий ситуацию, приводящую к реконфигурации сети при использовании протокола STP (рисунок 1.17).

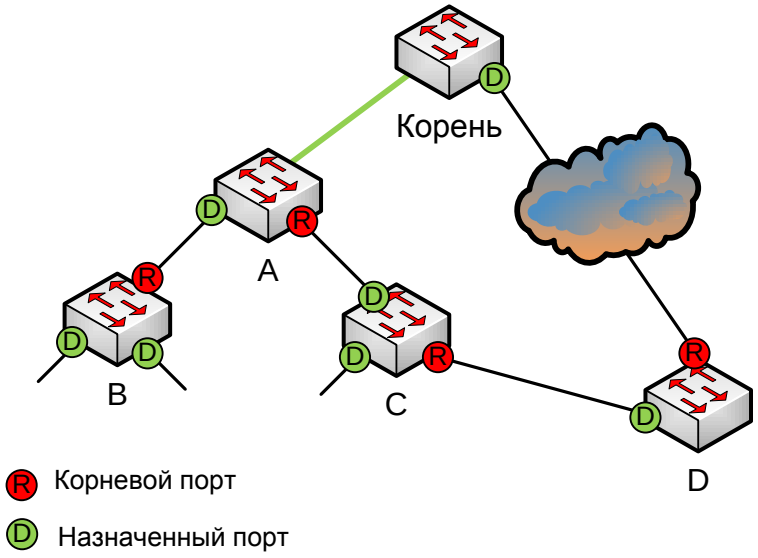


Рисунок 1.17.

На приведенной топологии добавлен новый канал между корневым коммутатором и коммутатором A. При этом между указанными коммутаторами уже есть связь через коммутаторы C и D (рисунок 1.18).

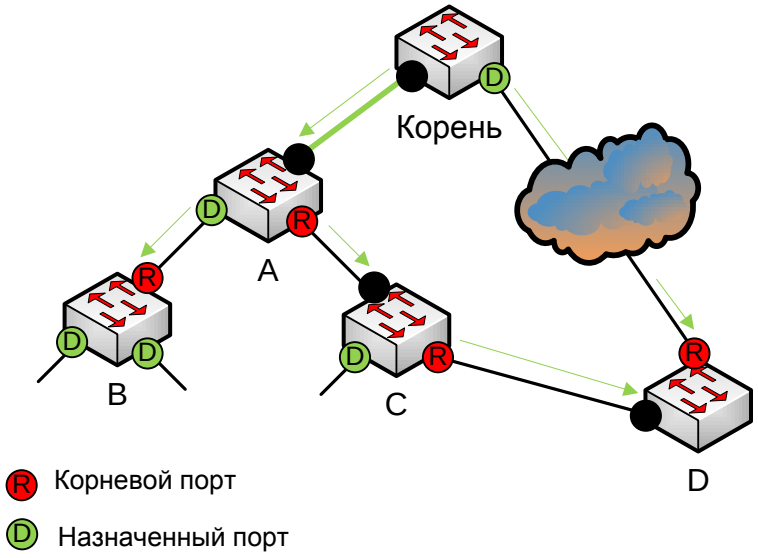


Рисунок 1.18.

Сейчас коммутатор A в состоянии прослушивать корневой порт напрямую. Таким образом, пакет BPDU от корня приходит не на корневой порт коммутатора A. А это одно из двух событий, приводящих к реконфигурации сети при использовании протокола STP. Таким образом, протокол STP блокирует порты на обоих концах нового канала, то есть порт корневого коммутатора и коммутатора A, и переводит эти порты в состояние «Прослушивание». Таким образом, исключается петля в сети.

Далее пакеты BPDU коммутатор A сразу же ретранслирует на все свои порты, и они попадают на назначенный порт коммутатора C и назначенный порт коммутатора D. Следовательно, коммутаторы C и D тоже понимают, что произошло изменение в конфигурации сети, блокируют все свои порты и рассылают сообщение о реконфигурации сети. Теперь только через двойное время Bridge Forward Delay произойдет полное перестроение сети.

Теперь посмотрим, как протокол RSTP будет работать в аналогичной ситуации. Обратите внимание на то, что конечная топология сети будет такой же, как и в случае использования протокола STP. Только при использовании протокола RSTP для достижения данной

топологии будет применена другая последовательность шагов, что займет гораздо меньше времени.

Также как и в протоколе STP сразу после добавления нового канала связи порты между коммутатором A и корневым коммутатором будут заблокированы (рисунок 1.19).

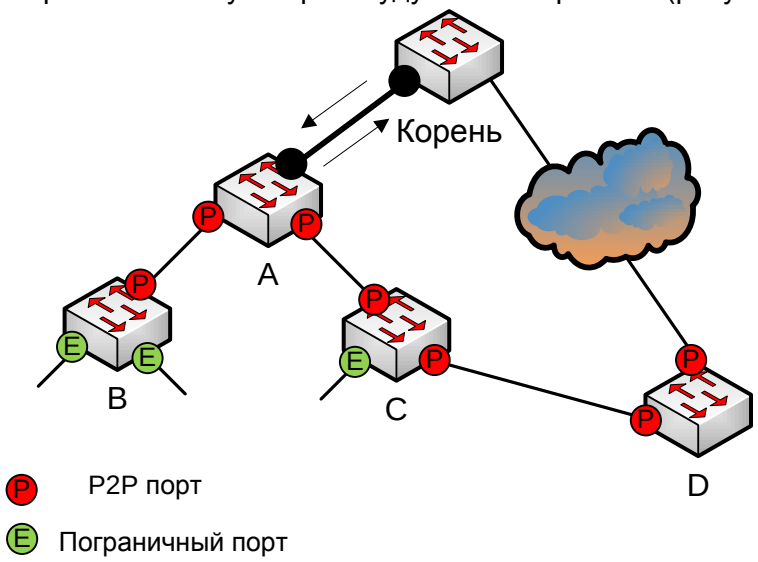


Рисунок 1.19.

После этого коммутатор A запустит процедуру sync, в рамках которой будут выполнены следующие действия:

- коммутатор A блокирует все свои P2P-порты;
- коммутатор A инициализирует процесс согласования с корнем, в рамках которого порты на обоих концах нового канала переводятся обратно в состояние Forwarding.

Далее процесс блокировки портов спускается ниже по дереву по мере того, как распространяются служебные пакеты BPDU корневого коммутатора через коммутатор A, и сеть принимает конфигурацию, показанную на рисунке 1.20.

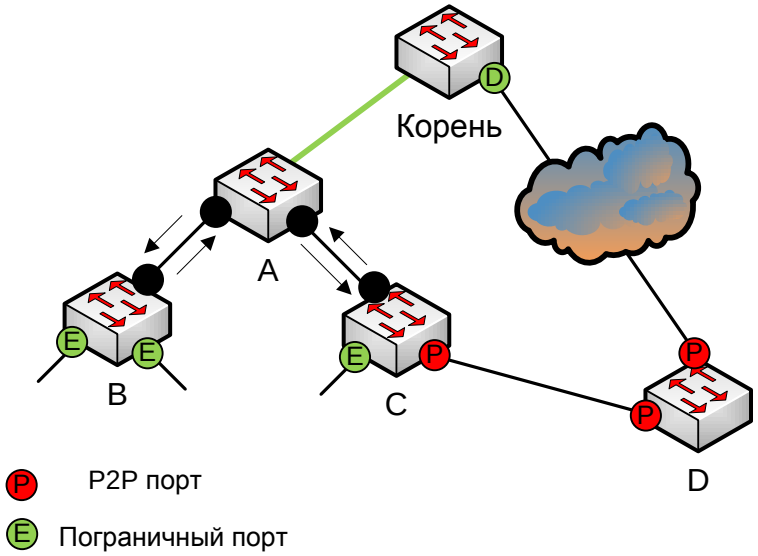


Рисунок 1.20.

Коммутаторы B и C также выполняют процедуру sync. Коммутатор B помимо заблокированного порта содержит только пограничные порты, поэтому в рамках процедуры sync он дополнительно не блокирует ни одного из своих портов, а только переводит канал между собой и коммутатором A в активное состояние. Коммутатор C блокирует порт, соединяющий его с D. В результате сеть принимает следующий вид (рисунок 1.21).

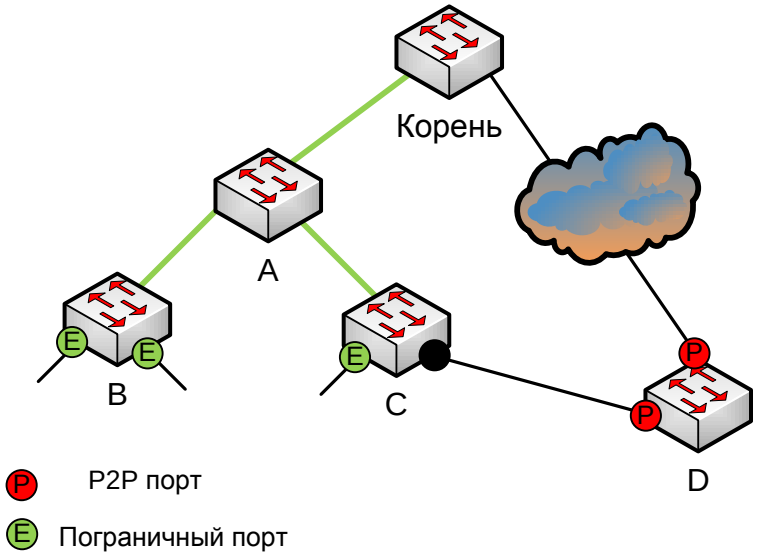


Рисунок 1.21.

В итоге достигается такая же конфигурация сети, как и при использовании протокола STP (блокируется порт коммутатора D, соединяющий его с коммутатором C). Время, затрачиваемое на перестроение сети, – это время, необходимое для спуска пакетов BPDU вниз по дереву, то есть в процессе реконфигурации не учитываются никакие таймеры.

Механизм «Предложение/Согласие»

После того, как согласно алгоритму STA порт стал назначенным, пройдёт еще двойное время Forward Delay, и только тогда порт перейдет в состояние продвижения. При использовании протокола RSTP переход портов в состояние продвижения осуществляется гораздо быстрее. Рассмотрим следующий пример (рисунок 1.22).

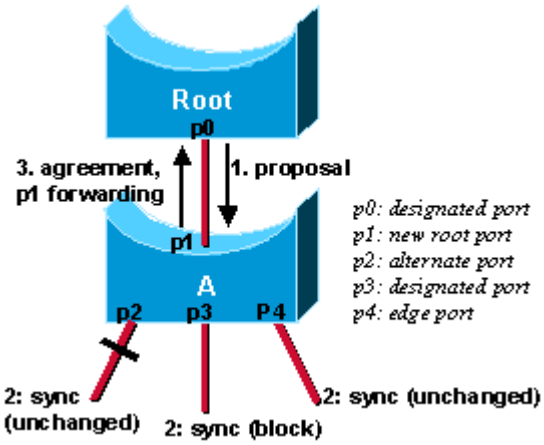


Рисунок 1.22.

Предположим, что между корневым коммутатором и коммутатором A появился новый канал связи. Соответственно, оба назначенных порта на этом канале будут переведены в состояние блокировки. При этом, согласно протоколу RSTP, когда назначенный порт находится в состоянии блокировки (Discarding) или обучения (Learning) (и только в этом случае), он выставляет бит «Предложение» (Proposal bit) в BPDU, который он генерирует. Поэтому порт p0 корневого коммутатора посылает именно такой пакет BPDU (шаг 1). Так как коммутатор A получает более приоритетную информацию (пакет BPDU, в котором расстояние до корня меньше, чем во всех пакетах BPDU, получаемых до возникновения нового канала связи), то он немедленно назначает порт p1 новым корневым портом. Далее коммутатор A запускает процедуру sync (Шаг 2). В данной процедуре не участвуют следующие типы портов:

- пограничный;
- заблокированный.

Для того, чтобы проиллюстрировать работу процедуры sync над различными типами портов, предположим, что на коммутаторе A существует альтернативный порт p2, назначенный порт p3 в состоянии продвижения и пограничный порт p4. Заметим, что в проводимой процедуре sync порты p2 и p4 участвовать не будут. Поэтому коммутатору A для удачного завершения процедуры sync необходимо заблокировать только порт p3. После этого коммутатор A может разблокировать свой новый корневой порт p1 и послать через него пакет BPDU корневому коммутатору с выставленным битом «Согласие» (Agreement bit) (Шаг 3). Сразу после получения портом p0 данного пакета, он сразу переходит в состояние продвижения (Шаг 4, рисунок 1.23).

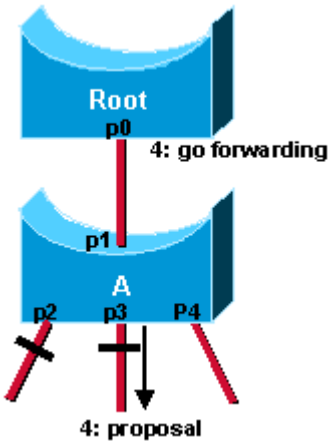


Рисунок 1.23.

После выполнения четвертого шага порт p3 находится в такой же ситуации, как и порт p0 на первом шаге. Поэтому порт p3 начнет посылать своим соседям пакеты BPDU с выставленным битом предложения.

Таким образом, можно выделить следующие основные черты механизма «Предложение/Соглашение»:

- механизм очень быстрый и не использует в своей работе никаких таймеров. «Волна рукопожатий» быстро распространяется по направлению к границе сети и быстро восстанавливает работоспособность сети после происходящих изменений в топологии;
- если заблокированный назначенный порт не получает пакет BPDU с выставленным битом «Соглашение» он выполняет стандартную процедуру перехода в состояние продвижения (discarding-learning-listen), тем самым замедляя процедуру перехода и увеличивая время реконфигурации сети. Это может случиться, если удаленный коммутатор не поддерживает протокол RSTP или порт удаленного коммутатора заблокирован.

UplinkFast

Другой формой немедленного перевода порта в состояние продвижения является механизм UplinkFast. Когда корневой порт выходит из строя, коммутатор переводит свой наилучший альтернативный порт в режим продвижения. Выбор альтернативного порта в качестве корневого изменяет топологию сети, но при этом типичная процедура реконфигурации сети путем рассылки широковещательных (или multicast) сообщений BPDU не запускается. Просто мосты, стоящие выше в дереве иерархии, очищают соответствующие записи в таблице Content Addressable Memory. Данный механизм не требует какой-либо конфигурации со стороны администратора, так как в протоколе RSTP он является встроенным.

Новый механизм изменения топологии
Механизм изменения топологии в протоколе RSTP сильно переработан.

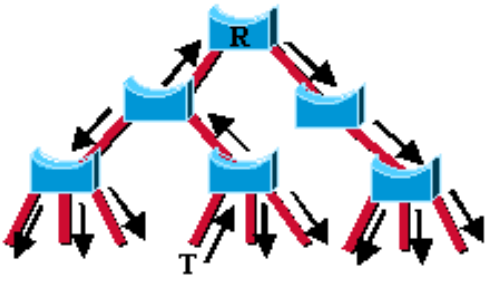
Обнаружение изменения топологии
Согласно стандарту IEEE 802.1w, только не пограничные порты, переведенные в состояние продвижения, могут генерировать события изменения топологии. При обнаружении изменения топологии протокол RSTP в первую очередь производит следующие действия:

- запускает на всех не пограничных портах и корневом порту таймер TC While Timer на время, равное двум величинам Hello Time. Во время работы таймера TC While Timer через порт посылаются сообщения BPDU с выставленным флагом TC;
- очищает все записи в таблице перенаправления, связанные с данными портами.

Распространение объявления об изменении топологии
Кода коммутатор получает сообщение BPDU с выставленным флагом TC, он производит следующие операции:

- очищает все записи в таблице перенаправления, связанные со всеми его портами за исключением того, по которому было получено сообщение BPDU;
- запускает таймер TC While Timer и рассылает сообщения BPDU с выставленным флагом TC через все свои назначенные порты и корневой порт.

Таким образом, сообщения об изменение топологии распространяются очень быстро по всей сети. Распространение данных сообщений теперь укладывается в один шаг. То есть коммутатор, обнаруживший изменение топологии, распространяет сообщения через всю сеть в отличие от протокола STP, где данную функцию может выполнять только корневой коммутатор (рисунок 1.24).



Создатель сообщений Toplogy Change напрямую распространяет их через всю сеть

Рисунок 1.24.

За несколько секунд большинство записей в таблице перенаправления сбрасываются у всех коммутаторов сети. С одной стороны это приводит к временному «флуду» в сети (который необходим для повторного заполнения таблицы), а с другой – к удалению устаревших записей, которые мешают быстрому восстановлению связности сети.

Совместимость со стандартом IEEE 802.1d
Протокол RSTP в состоянии взаимодействовать с протокол STP, но при этом теряются все достоинства стандарта IEEE 802.1w, связанные с быстрой сходимостью сетевой топологии.

Недостатки стандарта IEEE 802.1w
Хотя протокол RSTP имеет гораздо более высокие показатели по надежности и отказоустойчивости по сравнению со стандартным протоколом STP, он сохраняет такие недостатки, как долгое время восстановления сети. И поэтому любые версии Spanning Tree неприменимы в сетях, которые требуют надежности в 99,999%.

6.3. Множественные группы Spanning Tree (Multiple Spanning Tree, IEEE 802.1s)

MSTP расширяет стандарт IEEE 802.1w (RSTP)для поддержки нескольких копий STP. Это расширение обеспечивает как быструю сходимость сети, так и возможность баланса нагрузки в сети с настроенными VLAN. Стандарт 802.1s MSTP также вносит дополнения и в стандарт 802.1Q. Протокол MSTP обратно совместим с протоколами 802.1d и 802.1w и позволяет настраивать несколько независимых "деревьев" STP в разных VLAN – администратор может группировать и назначать VLAN на отдельные "связующие деревья". Каждое такое "дерево" может иметь свою независимую от других "деревьев" топологию. Подобная новая архитектура обеспечивает несколько разных вариантов для передачи данных и позволяет организовать баланс нагрузки. Это свойство улучшает отказоустойчивость сети к возможным сбоям, так как сбой соединений в отдельном "дереве" (маршруте передачи данных) не отразится на других "деревьях" и, соответственно, возможных маршрутах. Также благодаря MSTP облегчается задача администрирования и управления крупными сетями: можно использовать резервные маршруты передачи данных путем настройки нескольких VLAN и настройкой независимых "деревьев" на поучившихся сегментах сети.

Проиллюстрируем работу MSTP на простом примере (рисунок 1.25). Допустим, сеть состоит из 3 коммутаторов, соединенных между собой. В сети настроены 2 VLAN с VID 10 и 20. На коммутаторе 1 VLAN 10 и 20 настроены на разных портах таким образом, что трафик для обоих VLAN 10 и 20 передается по разным соединениям. На первый взгляд, такая конфигурация достаточно обычна и хорошо подходит для балансировки нагрузки при передаче трафика двух различных VLAN. Однако в сети настроен протокол STP.

Если коммутатор 3 будет выбран корневым коммутатором для STP, то соединение между коммутаторами 1 и 2 будет заблокировано. В этом случае трафик из VLAN 20 не сможет передаваться по сети. Эта проблема возникает потому, что коммутаторы рассматривают VLAN 10 и 20 как независимые сети, в то время как протокол STP рассматривает топологию сети как одну целую сеть.

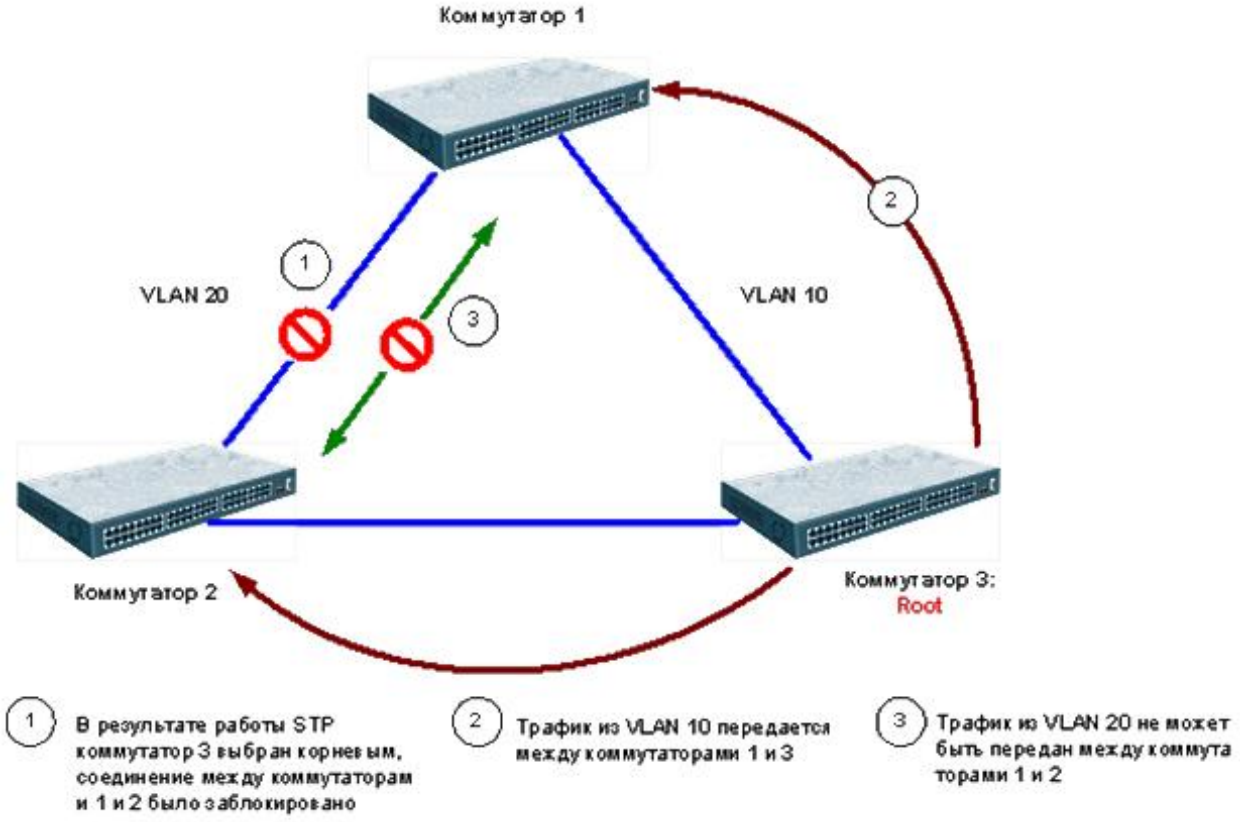


Рисунок 1.25.

Одним из решений подобной проблемы может стать настройка независимых копий STP на коммутаторах. Но настройка независимых копий STP не применяется, т.к. это может привести к значительному снижению производительности сети. К тому же, для многих сетей нет необходимости выполнять сложные настройки. Гораздо предпочтительнее настроить единую структуру STP на всей сети.

Для нормального взаимодействия нескольких устройств, они должны сопоставлять различные VLAN с несколькими возможными вариантами прохождения трафика - копиями дерева STP. Именно для этого и служит протокол 802.1S MSTP. Возвращаясь к примеру, можно увидеть, что 802.1s действительно решает поставленную задачу: если назначить VLAN 10 на копию MSTP под номером 1, а VLAN 20 сопоставить с копией 2. Т.о. получится две независимых топологии дерева STP. Коммутатор 3 становится корневым для копии MSTP номер 2 и блокирует прохождение трафика между коммутаторами 1 и 2. Но, в отличие от протокола 802.1D STP, это соединение блокируется только для прохождения трафика из VLAN 10. Трафик из VLAN 20 будет передаваться по этому соединению. Аналогичным образом копия MSTP под номером 2 выберет коммутатор 2 в качестве корневого и заблокирует соединение между коммутаторами 1 и 3 для трафика из VLAN 20.

Таким образом, достигается требуемая работа сети: осуществляется баланс нагрузки при передаче трафика нескольких VLAN по разным соединениям и в то же время в сети отсутствуют логические "петли".

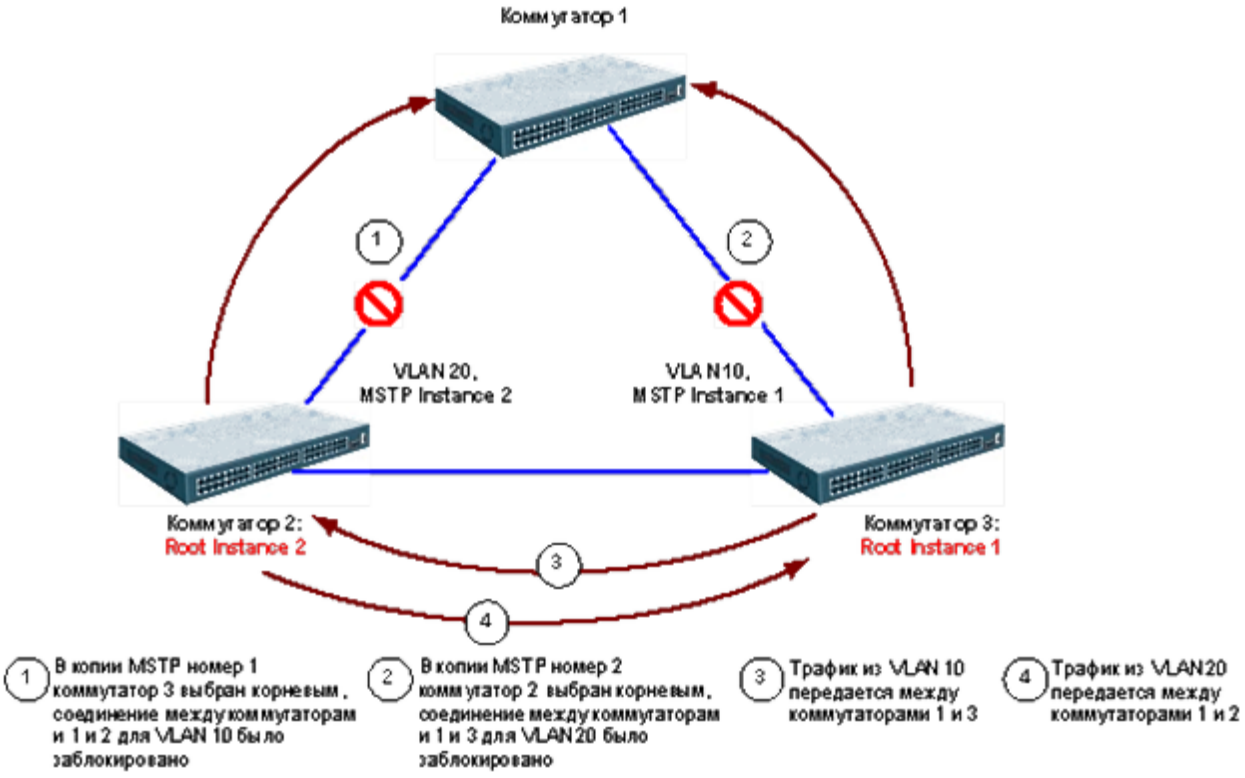


Рисунок 1.26.

7. Технология Quality of Service

7.1. Введение

В настоящее время вместе с планомерным увеличением скоростей передачи данных в телекоммуникациях увеличивается доля интерактивного трафика, крайне чувствительного к параметрам среды транспортировки. Поэтому задача обеспечения качества обслуживания (Quality of Service, QoS) становится все более актуальной.

Определений термина QoS настолько много, что мы выберем единственно верное – от Cisco: "QoS – QoS refers to the ability of a network to provide better service to selected network traffic over various underlying technologies…". Что можно литературно перевести как: "QoS – способность сети обеспечить необходимый сервис заданному трафику в определенных технологических рамках".

Необходимый сервис описывается многими параметрами, отметим среди них самые важные:

- ✓ Bandwidth (BW) – полоса пропускания, описывает номинальную пропускную способность среды передачи информации, определяет ширину канала.
- ✓ Delay – задержка при передаче пакета.
- ✓ Jitter – колебание (вариация) задержки при передаче пакетов.
- ✓ Packet Loss – потери пакетов, определяет количество пакетов, отбрасываемых сетью во время передачи.

Служба QoS имеет распределенный характер, так как ее элементы должны присутствовать во всех сетевых устройствах, продвигающих пакеты: коммутаторах, маршрутизаторах, серверах доступа. С другой стороны, служба QoS должна включать также элементы централизованного управления, поскольку работу отдельных сетевых устройств, поддерживающих QoS, нужно координировать, чтобы качество обслуживания было однородным вдоль всего пути, по которому следуют пакеты потока. Любые гарантии QoS настолько хороши, насколько их обеспечивает наиболее "слабый" элемент в цепочке между отправителем и получателем. Поэтому поддержка QoS только в одном сетевом устройстве, пусть даже и магистральном, может весьма незначительно улучшить качество обслуживания или же совсем не влиять на параметры QoS.

На рисунке 1.27 представлена базовая архитектура службы QoS с элементами трех основных типов:

- ✓ средств QoS узла;
- ✓ протоколов сигнализации QoS;
- ✓ централизованных функций политики, управления и учета QoS.

7.2. Сервисные модели QoS

Best Effort Service

Негарантированная доставка. Абсолютное отсутствие механизмов QoS. Используются все доступные ресурсы сети без какого-либо выделения отдельных классов трафика и регулирования. Считается, что лучшим механизмом обеспечения QoS является увеличение пропускной способности. Это в принципе правильно, однако некоторые виды трафика (например, голосовой) очень чувствительны к задержкам пакетов и вариации скорости их прохождения. Модель «Best Effort Service» даже при наличии больших резервов допускает возникновение перегрузок в случае резких всплесков трафика. Поэтому были разработаны и другие подходы к обеспечению QoS.

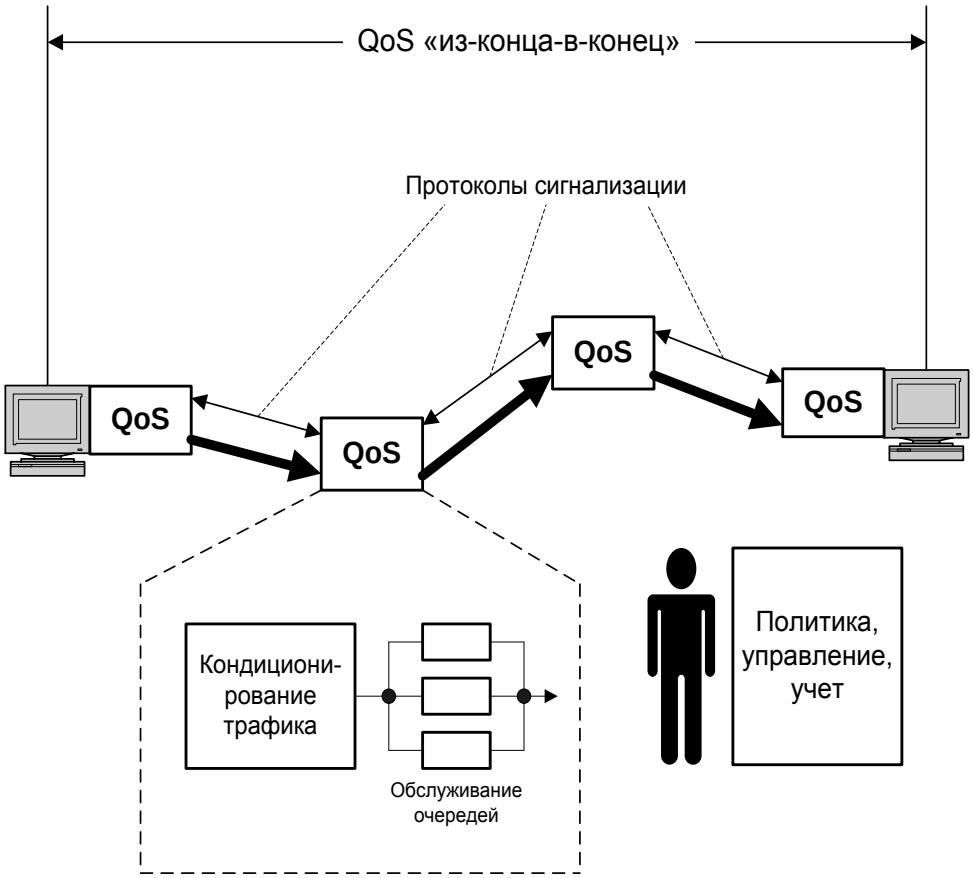


Рисунок 1.27. Базовая архитектура службы QoS.

Integrated Service (IntServ)

Integrated Service (IntServ, RFC 1633) – модель интегрированного обслуживания. Может обеспечить сквозное (End-to-End) качество обслуживания, гарантируя необходимую пропускную способность. IntServ использует для своих целей протокол сигнализации RSVP (ReSerVation Protocol). Позволяет приложениям выражать сквозные требования к ресурсам и содержит механизмы обеспечения данных требований. IntServ можно кратко охарактеризовать как резервирование ресурсов (Resource reservation).

В соответствии с концепциями «Integrated Services» приложения могут выбирать для своих потоков данных любой из многочисленных контролируемых уровней качества обслуживания. Для этого к базовым IP-сервисам добавляются новые компоненты и механизмы. Обслуживание в реальном времени требует гарантий, и эти гарантии не могут быть обеспечены без резервирования. Протокол резервирования RSVP предусматривает, что ресурсы резервируются для каждого потока, требующего QoS, на каждом промежуточном маршрутизаторе на пути от отправителя к получателю с использованием сигнализации «из конца в конец». Это, в свою очередь, требует хранения на маршрутизаторах данных о состоянии каждого конкретного потока и, как следствие, подразумевает кардинальные изменения в общей модели Internet. Кроме того, такое решение предлагает способ передачи требований приложения элементам сети, расположенным вдоль пути, и обмена служебной информацией QoS между сетевыми элементами и приложением.

Недостатки использования протокола RSVP следующие:

- ✓ время на прокладку маршрута резервирования, что может быть критичным в глобальных сетях;
- ✓ загрузка промежуточных маршрутизаторов информацией о характере передаваемого трафика.

Differentiated Service (DiffServ)

Differentiated Service (DiffServ, RFC 2474/2475) – модель дифференцированного обслуживания. Определяет обеспечение QoS на основе четко определенных компонентов, комбинируемых с целью предоставления требуемых услуг. Архитектура DiffServ предполагает наличие классификаторов и формирователей трафика на границе сети, а также поддержку функции распределения ресурсов в ядре сети в целях обеспечения требуемой политики пошагового обслуживания (Per-Hop Behavior - PHB). Разделяет трафик на классы, вводя несколько уровней QoS. DiffServ состоит из следующих функциональных блоков: граничные формирователи трафика (классификация пакетов, маркировка, управление интенсивностью) и реализаторы PHB политики (распределение ресурсов, политика отбрасывания пакетов). DiffServ можно кратко охарактеризовать как приоритезацию трафика (Prioritization).

Механизмы «Differentiated Services» усовершенствуют протокол IP с целью преодолеть ограничения IntServ/RSVP и обеспечить масштабируемое избирательное обслуживание в Internet без необходимости запоминать состояние каждого потока и поддерживать сигнализацию. В отличие от RSVP, в случае DiffServ отправитель и получатель не обмениваются информацией о требованиях к качеству обслуживания, что исключает временные затраты на прокладку пути, присущие RSVP.

Механизмы DiffServ ограничиваются только установлением соответствия между услугами и различными уровнями «чувствительности» к задержкам и потерям, то есть не имеют дела с точными значениями или гарантиями. Они не рассчитаны на обеспечение того или иного уровня обслуживания. Вместо этого они стараются обеспечить относительное упорядочивание агрегированных потоков, так что с одним агрегированным потоком будут «обращаться лучше», чем с другим, в зависимости от правил обслуживания, определенных для каждого агрегированного потока.

Масштабируемость архитектуры DiffServ достигается за счет объединения классификационных признаков трафика, при этом информация о типе трафика передается в заголовке IP-датаграммы. При этом сложные операции классификации, маркировки, определения правил обслуживания и формирования трафика необходимо выполнять только на границах сети или же на хостах

Будущая модель QoS

Применяемые вместе IntServ и DiffServ могут способствовать внедрению IP-телефонии, видео по требованию и различных критически важных для предприятий не мультимедийных приложений. IntServ позволяет хостам запрашивать конкретный объем ресурсов для каждого потока и использовать обратную связь для определения возможности выполнения запросов. DiffServ обеспечивает масштабируемость для крупных сетей.

В настоящее время предложена гибридная структура, которая, по-видимому, является наиболее выигрышным компромиссом. Она предусматривает применение модели, в которой периферийные подсети поддерживают RSVP и IntServ. Эти подсети связаны сетями DiffServ. Благодаря масштабируемости сетей DiffServ данная модель позволяет расширить диапазон действия сетей IntServ/RSVP. Промежуточные сети DiffServ выглядят для сетей IntServ/RSVP как одно транзитное звено. Хосты, подключенные к периферийным сетям IntServ/RSVP, передают через сети DiffServ запросы друг другу на резервирование ресурсов для каждого отдельного потока. Внутри периферийных сетей IntServ/RSVP применяется стандартная обработка протоколов IntServ/RSVP, а сигнальные сообщения RSVP передаются через сети DiffServ прозрачным образом. Устройства на границе между сетями IntServ/RSVP и сетями DiffServ обрабатывают сообщения RSVP и обеспечивают входной контроль с учетом наличия ресурсов внутри сети DiffServ.

7.3. Базовые функции QoS

Базовые функции QoS заключаются в обеспечении необходимых параметров сервиса и определяются по отношению к трафику как:

- ✓ классификация;
- ✓ разметка;
- ✓ управление перегрузками;
- ✓ предотвращение перегрузок;
- ✓ регулирование.

Функционально классификация и разметка чаще всего обеспечиваются на входных портах оборудования, а управление и предотвращение перегрузок – на выходных.

Классификация и разметка (Classification and Marking)

Классификация пакетов (Packet Classification) представляет собой механизм соотнесения пакета к определенному классу трафика. Другой не менее важной задачей при обработке пакетов является маркировка пакетов (Packet Marking) - назначение соответствующего приоритета (метки). В зависимости от уровня рассмотрения (имеется в виду OSI) эти задачи решаются по-разному.

Layer 2 Classification and Marking

Протокол Ethernet в чистом виде не поддерживает поле приоритета. Поэтому на Ethernet портах (Access Port) возможна лишь внутренняя (по отношению к коммутатору) классификация по номеру входящего порта и отсутствует какая-либо маркировка.

Более гибким решением является использование стандарта IEEE 802.1P, который разрабатывался совместно с 802.1Q. Иерархия отношений здесь следующая: 802.1D описывает технологию мостов и является базовой для 802.1Q и 802.1P. 802.1Q описывает технологию виртуальных сетей (VLAN), а 802.1P обеспечение качества обслуживания. В целом, включение поддержки 802.1Q автоматически дает возможность использования 802.1P. Согласно стандарту используются 3 бита в заголовке второго уровня, которые называются Class of Service (CoS). Таким образом, CoS может принимать значения от 0 до 7.

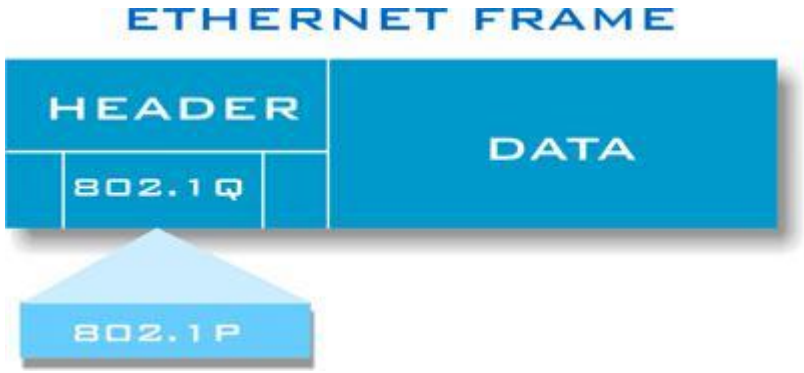


Рисунок 1.28. Дополнительное поле CoS в заголовке кадра Ethernet.

Layer 3 Classification and Marking

Маршрутизирующее оборудование (Layer 3) оперирует IP пакетами, в которых под цели маркировки предусмотрено соответствующее поле в IP-заголовке - Type of Service (ToS) размером один байт. ToS может быть заполнен классификатором IP Precedence или DSCP в зависимости от задачи. IP precedence (IPP) имеет размерность 3 бита (принимает значения 0-7). DSCP относится к модели DiffServ и состоит из 6 бит (значения 0-63).

Кроме цифровой формы, значения DSCP могут быть выражены с использованием специальных ключевых слов: доставка по возможности BE – Best Effort, гарантированная доставка AF – Assured Forwarding и срочная доставка EF – Expedited Forwarding. В дополнение к этим трем классам существуют коды селектора классов, которые добавляются к обозначению класса и обратно совместимы с IPP. Например, значение DSCP равное 26 можно записать как AF31, что полностью равнозначно.

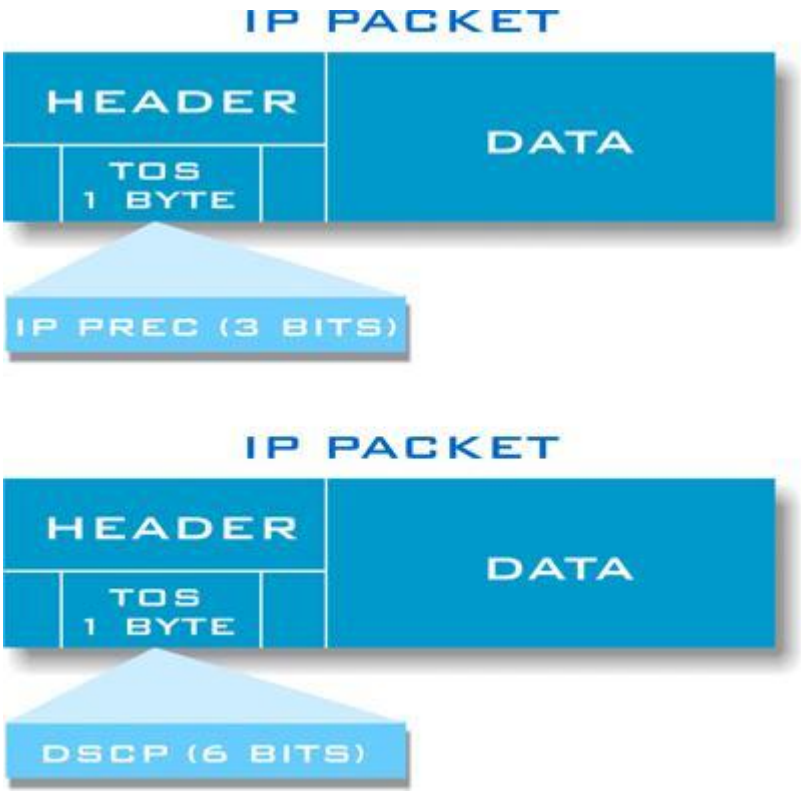


Рисунок 1.29. Поле TOS в IP-заголовке.

Управление перегрузками (Congestion Management). Механизм очередей.

Перегрузки (Congestions)

Перегрузка возникает в случае переполнения выходных буферов передающего трафик оборудования. Основными механизмами возникновения перегрузок (или, что равнозначно, скоплений – congestions) является агрегация трафика и несогласованность скоростей на интерфейсах.

Управление пропускной способностью в случае перегрузок (узких мест) осуществляется с помощью механизма очередей. Пакеты помещаются в очереди, которые упорядоченно обрабатываются по определенному алгоритму. Фактически, управление перегрузками – это определение порядка, в котором пакеты выходят из интерфейса (очередей) на основе приоритетов. Если перегрузок нет – очереди не работают (и не нужны). Перечислим методы обработки очередей.

Layer 2 Queuing

Физическое устройство классического коммутатора можно упрощенно представить следующим образом: пакет приходит на входной порт, обрабатывается механизмом коммутации, который решает, куда направить пакет, и попадает в аппаратные очереди выходного порта. Аппаратные очереди представляет собой быструю память, хранящую пакеты перед тем, как они попадут непосредственно на выходной порт. Далее, согласно определенному механизму обработки, пакеты извлекаются из очередей и покидают коммутатор. Изначально очереди равноправны и именно механизм обработки очередей (Scheduling) определяет приоритезацию. Обычно каждый порт коммутатора содержит ограниченное число очередей: 2, 4, 8 и так далее.

В общих чертах настройка приоритезации заключается в следующем:

1. Изначально очереди равноправны. Поэтому предварительно необходимо их настроить, то есть определить очередность (или пропорциональность объема) их обработки. Чаще всего это делается привязкой приоритетов 802.1P к очередям.
2. Необходимо сконфигурировать обработчик очередей (Scheduler). Чаще всего используются взвешенный циклический алгоритм (Weighted Round Robin, WRR), который работает также как настраиваемые очереди (Custom Queueing) на уровне 3 (см. ни-

же), или строгая очередь приоритетов (Strict Priority Queueing), которая работает также как приоритетное обслуживание (Priority Queueing) на уровне 3 (см. ниже).

3. Назначение приоритета поступающим пакетам: по входному порту, по CoS или, в случае дополнительных возможностей (Layer 3 switch), по каким-то полям IP.

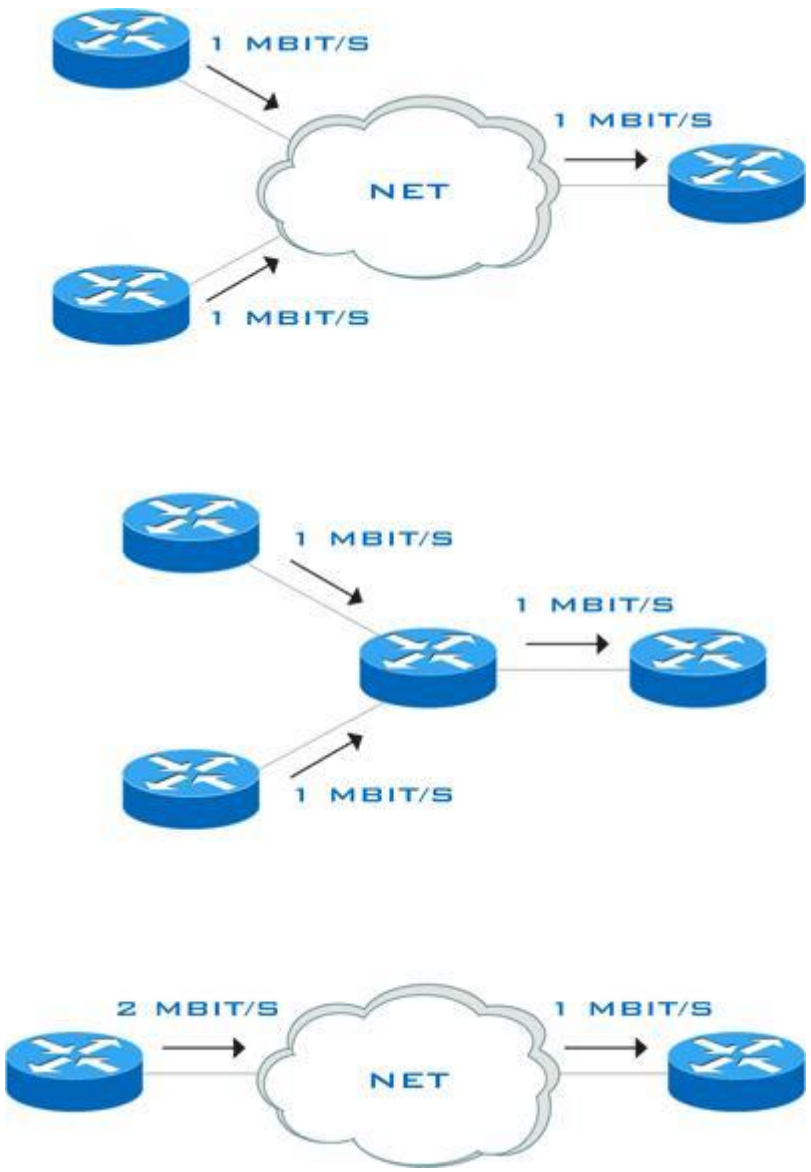


Рисунок 1.30. Примеры возникновения перегрузок.

Работает все это следующим образом:

1. Пакет попадает в коммутатор. Если это обычный Ethernet пакет, то он не имеет меток приоритета и таковая может выставляться коммутатором, например, по номеру входного порта, если это нужно. Если входной порт транковый (802.1Q или ISL), то пакет может нести метку приоритета и коммутатор может ее принять или заменить на необходимую. В любом случае пакет на данном этапе попал в коммутатор и имеет необходимую разметку CoS.
2. После обработки процессом коммутации пакет в соответствии с меткой приоритета CoS направляется классификатором (Classify) в соответствующую очередь выходного порта. Например, критический трафик попадает в высокоприоритетную, а менее важный в низкоприоритетную очереди.
3. Механизм обработки (Scheduling) извлекает пакеты из очередей согласно их приоритетам. Из высокоприоритетной очереди за единицу времени будет выдано на выходной порт больше пакетов, чем из низкоприоритетной.

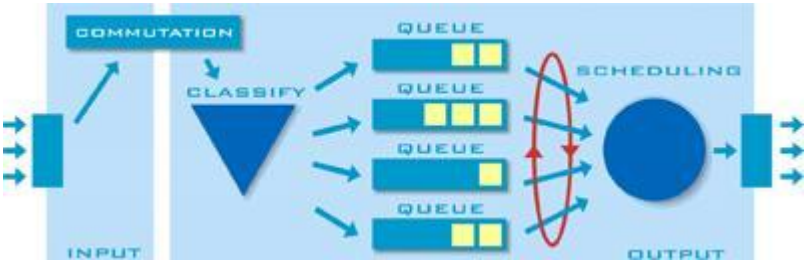


Рисунок 1.31. Механизм обработки входящего трафика согласно CoS.

Layer 3 Queuing

Маршрутизирующие устройства оперируют пакетами на третьем уровне OSI. Чаще всего поддержка очередей обеспечивается программно. Это означает в большинстве случаев отсутствие аппаратных ограничений на их число и более гибкое конфигурирование механизмов обработки. Общая парадигма QoS Layer 3 включает классификацию и маркировку пакетов на входе (Classification & Marking), распределение по очередям и их обработку (Scheduling) по определенным алгоритмам.

И еще раз подчеркнем, что приоритезация (очереди) требуется в основном только в узких, загруженных местах, когда пропускной способности канала не хватает для передачи всех поступающих пакетов и нужно каким-то образом дифференцировать их обработку. Кроме того, приоритезация необходима и в случае предотвращения влияния всплесков сетевой активности на чувствительный к задержкам трафик.

Проведем классификацию Layer 3 QoS по методам обработки очередей. Чаще всего в маршрутизаторах и коммутаторах применяются следующие алгоритмы обработки очередей:

- ✓ традиционный алгоритм FIFO;
- ✓ приоритетное обслуживание (Priority Queueing), которое также называют «подавляющим»;
- ✓ настраиваемые очереди (Custom Queueing);
- ✓ взвешенное справедливое обслуживание (Weighted Fair Queueing, WFQ).

Каждый алгоритм разрабатывался для решения определённых задач и поэтому специфическим образом воздействует на качество обслуживания различных типов трафика в сети. Возможно комбинированное применение этих алгоритмов.

FIFO

Элементарная очередь с последовательным прохождением пакетов, работающая по принципу первый пришел – первый ушел (First In First Out - FIFO. Здесь нет никакой приоритезации.

Приоритетное обслуживание

Механизм приоритетной обработки трафика предусматривает разделение всего сетевого трафика на небольшое количество классов с назначением каждому классу некоторого числового признака – приоритета. Блок классификации трафика может размещаться как в самом устройстве (рисунок 1.32), так и вне его.

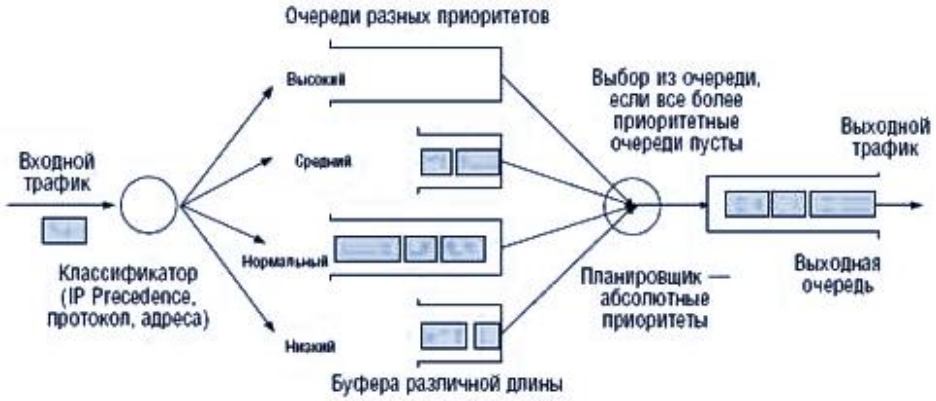


Рисунок 1.32. Организация приоритетных очередей Priority Queueing.

Независимо от выбранного способа классификации трафика, в сетевом устройстве имеется несколько очередей, в соответствии с количеством классов. Поступивший в период перегрузки пакет помещается в очередь согласно его приоритету. На рисунке 1.32 приведен пример использования четырех приоритетных очередей: с высоким, средним, нормальным и низким приоритетом. Приоритеты очередей имеют абсолютный характер предпочтения при обработке: пока из более приоритетной очереди не будут выбраны все пакеты, устройство не переходит к обработке следующей, менее приоритетной. Поэтому пакеты со средним приоритетом всегда обрабатываются только тогда, когда очередь пакетов с высоким приоритетом пуста, а пакеты с низким приоритетом – только когда пусты все вышестоящие очереди.

Конечный размер буферной памяти сетевого устройства предполагает некоторую предельную длину каждой очереди. Обычно по умолчанию всем приоритетным очередям отводятся буферы одинакового размера, но многие устройства разрешают администратору выделять каждой очереди индивидуальный буфер. Его максимальная длина определяет предельное количество пакетов, которые могут храниться в очереди данного приоритета. Пакет, поступивший в то время, когда буфер заполнен, просто отбрасывается.

Приоритетное обслуживание очередей обеспечивает высокое качество сервиса для пакетов из самой приоритетной очереди. Если средняя интенсивность их поступления в устройство не превосходит пропускной способности выходного интерфейса (и производительности внутренних блоков самого устройства, участвующих в продвижении пакетов), то пакеты с наивысшим приоритетом всегда получают ту пропускную способность, которая им необходима. Что же касается остальных классов приоритетов, то качество их обслуживания ниже, чем у пакетов с наивысшим приоритетом.

Поэтому приоритетное обслуживание обычно применяется в том случае, когда в сети есть чувствительный к задержкам трафик, но его интенсивность невелика, так что его наличие не слишком ущемляет остальной трафик. Например, голосовой трафик чувствителен к задержкам, но его интенсивность обычно не превышает 8-16 Кбит/с, и, таким образом, при назначении ему наивысшего приоритета остальные классы трафика не пострадают. Однако в сети могут наблюдаться и другие ситуации. В частности, видеотрафик тоже требует первоочередного обслуживания, но имеет гораздо более высокую интенсивность. Для таких случаев разработаны алгоритмы управления очередями, дающие низкоприоритетному трафику некоторые гарантии даже в периоды повышения интенсивности высокоприоритетного трафика.

Взвешенные настраиваемые очереди

Алгоритм взвешенных очередей (Weighted Queueing) разработан для того, чтобы для всех классов трафика можно было предоставить определенный минимум пропускной способности или удовлетворить требования к задержкам. Под весом какого-либо класса понимается доля выделяемой данному виду трафика пропускной способности выходного интерфейса. Алгоритм, в котором вес классов трафика может назначаться администратором, называется «настраиваемой очередью» (Custom Queueing). В случае, когда веса назначаются автоматически, на основании некоторой адаптивной стратегии, реализуется так называемый алгоритм «взвешенного справедливого обслуживания» (Weighted Fair Queueing, WFQ).

Как при взвешенном, так и при приоритетном обслуживании, трафик делится на несколько классов, и для каждого вводится отдельная очередь пакетов. С каждой очередью связывается доля пропускной способности выходного интерфейса, гарантируемая данному классу трафика при перегрузках этого интерфейса. В примере, приведенном на рисунке 1.33, устройство поддерживает пять очередей для пяти классов трафика. Этим очередям соответствует 10, 10, 30, 20 и 30% пропускной способности выходного интерфейса при перегрузках.

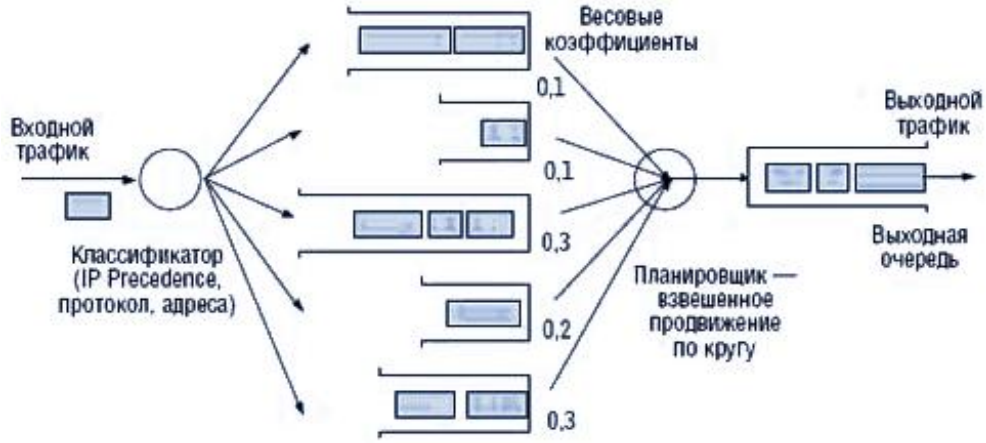


Рисунок 1.33. Настраиваемые очереди Custom Queueing.

Поставленная цель достигается благодаря тому, что очереди обслуживаются последовательно и циклически, и в каждом цикле из каждой очереди забирается такое число байт, которое соответствует весу очереди. Например, если цикл просмотра очередей в рассматриваемом примере равен 1 сек, а скорость выходного интерфейса составляет 100 Мбит/с, то при перегрузках в каждом цикле из первой очереди забирается 10 Мбит данных, из второй тоже 10 Мбит, из третьей – 30 Мбит, из четвертой – 20 Мбит, из пятой – 30 Мбит. В результате каждому классу трафика достается гарантированный минимум пропускной способности, что во многих случаях является более желательным результатом, чем подавление низкоприоритетных классов высокоприоритетным.

Точные значения параметров QoS для алгоритма взвешенного обслуживания предсказать трудно. Они существенным образом зависят от динамически изменяющихся параметров нагрузки сетевого устройства – интенсивности пакетов всех классов и вариаций промежутков времени между прибытием пакетов. В общем случае взвешенное обслуживание приводит к большим задержкам и их отклонениям, чем первоочередное обслуживание для самого приоритетного класса, даже при значительном превышении выделенной пропускной способности над интенсивностью входного потока данного класса. Но для более низких приоритетных классов взвешенное справедливое обслуживание часто оказывается более приемлемым с точки зрения создания благоприятных условий обслуживания всех классов трафика.

Взвешенное справедливое обслуживание

Взвешенное справедливое обслуживание (Weighted Fair Queuing, WFQ) – это комбинированный механизм обслуживания очередей, сочетающий приоритетное обслуживание со взвешенным. Производители сетевого оборудования предлагают многочисленные собственные реализации WFQ, отличающиеся способом назначения весов и поддержкой различных режимов работы, поэтому в каждом конкретном случае необходимо внимательно изучить все детали поддерживаемого WFQ.

Наиболее распространенная схема предусматривает существование одной особой очереди, которая обслуживается по приоритетной схеме – всегда в первую очередь и до тех пор, пока все заявки из нее не будут исполнены. Эта очередь предназначена для системных сообщений, сообщений управления сетью и, возможно, пакетов наиболее критических и требовательных приложений. Во всяком случае, предполагается, что её трафик имеет невысокую интенсивность, поэтому значительная часть пропускной способности выходного интерфейса остается другим классам трафика.

Остальные очереди устройство просматривает последовательно, в соответствии с алгоритмом взвешенного обслуживания (рисунок 1.34). Администратор может задать вес для каждого класса трафика аналогично тому, как это делается в случае взвешенного обслуживания. Вариант работы по умолчанию предусматривает для всех остальных классов трафика равные доли пропускной способности выходного интерфейса (за вычетом оставшейся от приоритетного трафика).

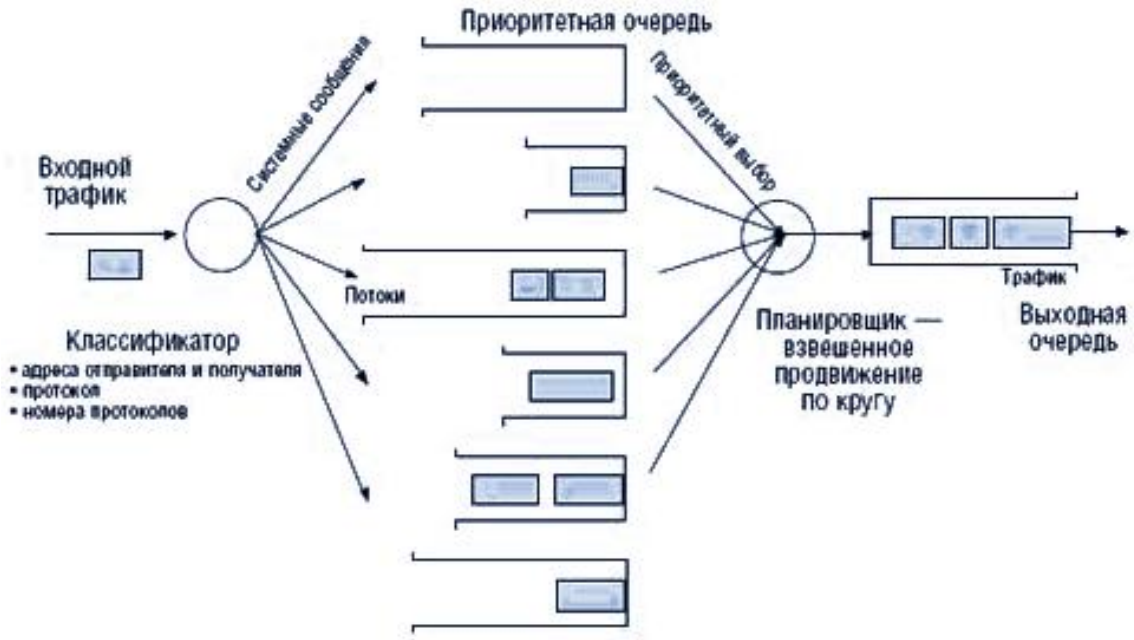


Рисунок 1.34. Взвешенное справедливое обслуживание Weighted Fair Queuing.

Производители оборудования дополняют механизм WFQ некоторыми полезными режимами. Например, в маршрутизаторах компании Cisco предусмотрено несколько разновидностей WFQ:

- ✓ основанный на потоках (Flow-based) режим WFQ (FWFQ);
- ✓ основанный на классах (Class-based) режим WFQ (CWFQ).

Для варианта FWFQ на базе потоков в маршрутизаторе создается столько очередей, сколько потоков существует в трафике. Каждому потоку соответствует отдельная выходная очередь, для которой в периоды перегрузок механизм WFQ выделяет равные доли пропускной способности порта. Поэтому иногда алгоритм FWFQ называют FQ (Fair Queuing) – справедливое обслуживание.

Вариант CWFQ на базе классов в маршрутизаторах Cisco имеет два подварианта:

- ✓ классы трафика определяются на основании так называемых групп QoS, соответствующих набору признаков из списка управления доступом (ACL), например, номеру входного интерфейса или номеру хоста или подсети;
- ✓ классы трафика определяются значениями полей ToS.

Для варианта групп QoS администратор задает веса пропускной способности, выделяемой каждой группе QoS, а также (опционально) максимальную длину очереди. Пакеты, не отнесенные ни к одной из групп, включаются в группу 0. При назначении весов WFQ нужно принимать во внимание следующее:

- ✓ группе QoS с номером 0 автоматически назначается 1% имеющейся пропускной способности;
- ✓ общий вес всех остальных групп не может быть более 99%;
- ✓ оставшаяся после назначения весов пропускная способность выделяется группе 0.

В варианте классификации на основании значении ToS предусматриваются веса классов по умолчанию. Они вступают в силу, если администратор явно не задал их с помощью команды weight. Для классификации используется два младших бита трехразрядного подполя IP Precedence из поля ToS, так что в этом варианте имеется всего четыре класса трафика. По умолчанию, классу 0 выделяется 10% выходной пропускной способности, классу 1 – 20%, классу 2 – 30% и классу 3 – 40%. Чем выше класс, тем важнее трафик, поэтому выделение большей доли пропускной способности создает для него более привилегированные условия продвижения.

8. Основы коммутации третьего уровня

8.1. Причины появления

Маршрутизаторы – устройства сложные и оказываются дороже, чем коммутаторы при том же уровне производительности. А из-за задержек на обработку информации маршрутизатор функционирует в одинаковых условиях медленнее, чем коммутатор. С другой стороны, данные, которыми располагает маршрутизатор, позволяют ему выполнять дополнительные функции помимо собственно перемещения пакетов данных, например управление передачей. К концу 90-х годов возникла необходимость в создании новых сетевых устройств, которые бы объединили функции маршрутизаторов и производительность коммутаторов. Стало понятно, что решение надо искать на пересечении технологий: объединив достоинства коммутации с умением маршрутизаторов накладывать на физическую структуру сети логические ограничения.

Идея коммутация на 3 уровне впервые была предложена компанией Ipsilon. На текущий момент времени (2008 год) так и не создано какого-либо стандарта, определяющего концепцию коммутации на 3 уровне модели OSI. Вместо этого каждый производитель трактует и реализует данную технологию по-своему. Тем не менее, можно выделить основные направления в технологии коммутации третьего уровня, которые рассмотрены в следующем разделе.

8.2. Классификация

Коммутаторы 3-го уровня делятся на две категории:

- пакетные (Packet-by-Packet Layer 3 Switches, PPL3) или коммутирующие маршрутизаторы (switching routers). PPL3 означает просто быструю маршрутизацию;
- сквозные (Cut-Through Layer 3 Switches, CTL3) или маршрутизирующие коммутаторы (routing switches). CTL3 – маршрутизацию первого пакета и коммутацию всех остальных;
- коммутаторы потоков.

8.3. Пакетные коммутаторы 3 уровня

Пакетная коммутация на третьем уровне (рисунок 1.35) на самом деле представляет собой не что иное, как аппаратную маршрутизация. Традиционные маршрутизаторы реализуют свои функции с помощью программно-управляемых процессоров, что будем называть программной маршрутизацией. Использование высокоскоростной технологии больших заказных интегральных схем (ASIC) является главной характеристикой, отличающей пакетные коммутаторы третьего уровня от традиционных маршрутизаторов.

Пакетная коммутация относительно проста, полностью поддерживает все сервисы и протоколы, хорошо изучена и протестирована в конкретных продуктах. Пакетные коммутаторы и маршрутизаторы фактически идентичны, разница заключается в их физической реализации. В стандартных маршрутизаторах обработка пакета происходит в микропроцессорном ядре, в то время как коммутаторы 3 уровня используют микросхемы ASIC (Application Specific Integrated Circuit, Специализированная прикладная интегральная схема).

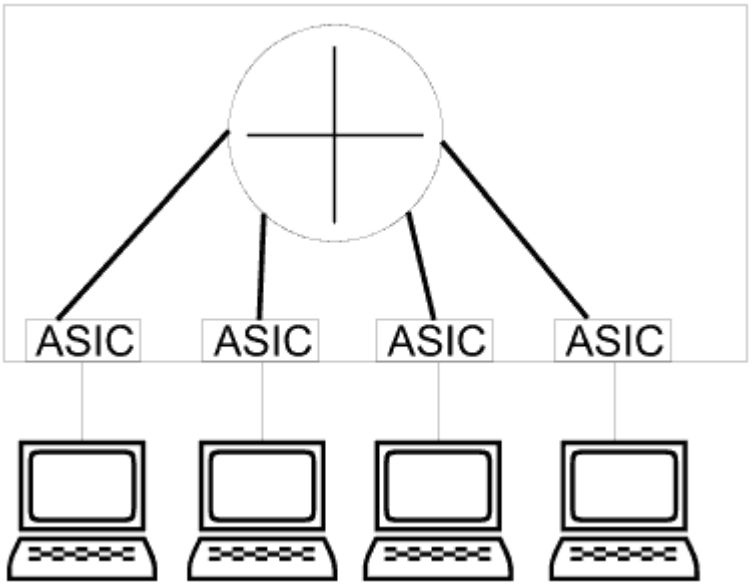


Рисунок 1.35.

8.4. Сквозные коммутаторы 3 уровня

Первыми устройствами, реализовавшими коммутацию третьего уровня, были маршрутизирующие коммутаторы или сквозные коммутаторы. Процесс обработки и передачи данных осуществляется на Канальном уровне, предварительный же поиск оптимального маршрута осуществляется при помощи 3-го уровня. Обычно задействуется внешний маршрутизатор или плата маршрутизации, которую коммутатор воспринимает как отдельное внешнее устройство.

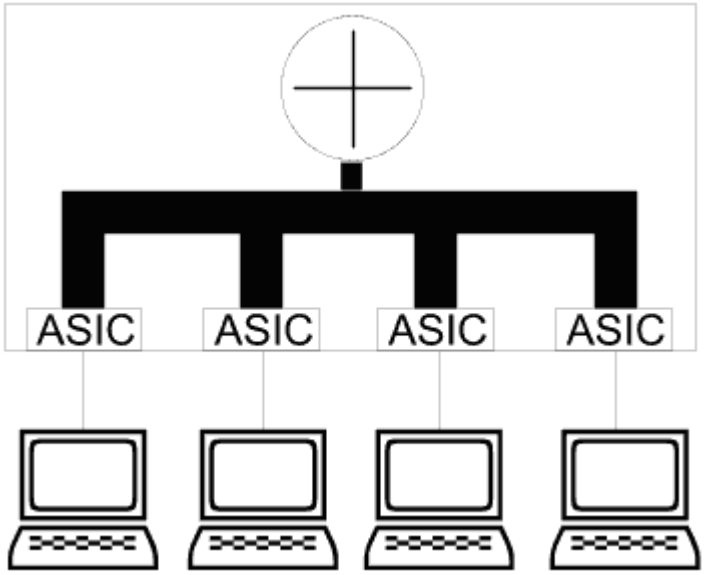


Рисунок 1.36.

Более подробно схема работы данных устройств выглядит следующим образом (рисунок 1.37). Пусть коммутатор третьего уровня построен так, что в нем имеется информация о соответствии сетевых адресов (например, IP-адресов) адресам физического уровня (например, MAC-адресам). Все эти MAC-адреса обычным образом отображены в коммутационной таблице (Forwarding Table), независимо от того, принадлежат ли они данной сети или другим сетям.

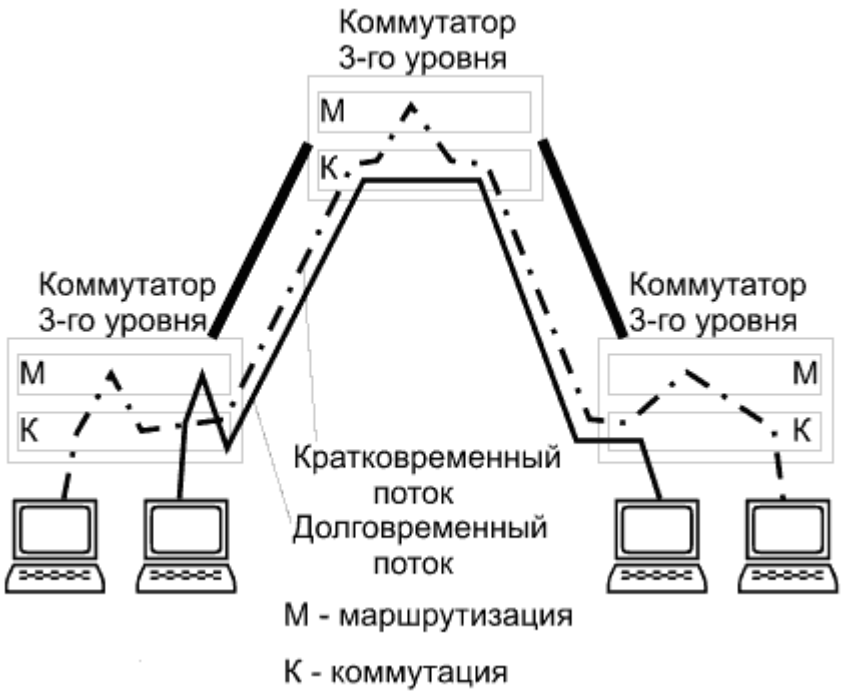


Рисунок 1.37.

Первый коммутатор, на который поступает пакет, частично выполняет функции маршрутизатора, а именно, функции фильтрации, обеспечивающие безопасность. Он решает, пропускать или нет данный пакет в другую сеть. Если пакет пропускать нужно, то коммутатор по IP-адресу назначения определяет MAC-адрес узла назначения и формирует новый заголовок второго уровня с найденным MAC-адресом. Затем выполняется обычная процедура коммутации по данному MAC-адресу с просмотром адресной таблицы коммутатора. Все последующие коммутаторы, построенные по этому же принципу, обрабатывают данный кадр как обычные коммутаторы второго уровня, не привлекая функций маршрутизации, что значительно ускоряет его обработку. Однако функции маршрутизации не являются для них избыточными, поскольку и на эти коммутаторы могут поступать первичные пакеты (непосредственно от рабочих станций), для которых необходимо выполнять фильтрацию и подстановку MAC-адресов.

Это описание носит схематический характер и не раскрывает способов решения возникающих при этом многочисленных проблем, например, проблемы построения таблицы соответствия IP-адресов и MAC-адресов.

Последним комплексным "проявлением" маршрутизирующей коммутации является технология Мультипротокольной Коммутации Меток (MultiProtocol Label Switching, MPLS), которая пытается сразу решить как задачи непосредственно быстрой передачи данных, так и качества обслуживания и приоритезации трафика (QoS) плюс изоляции и безопасности (VLAN).

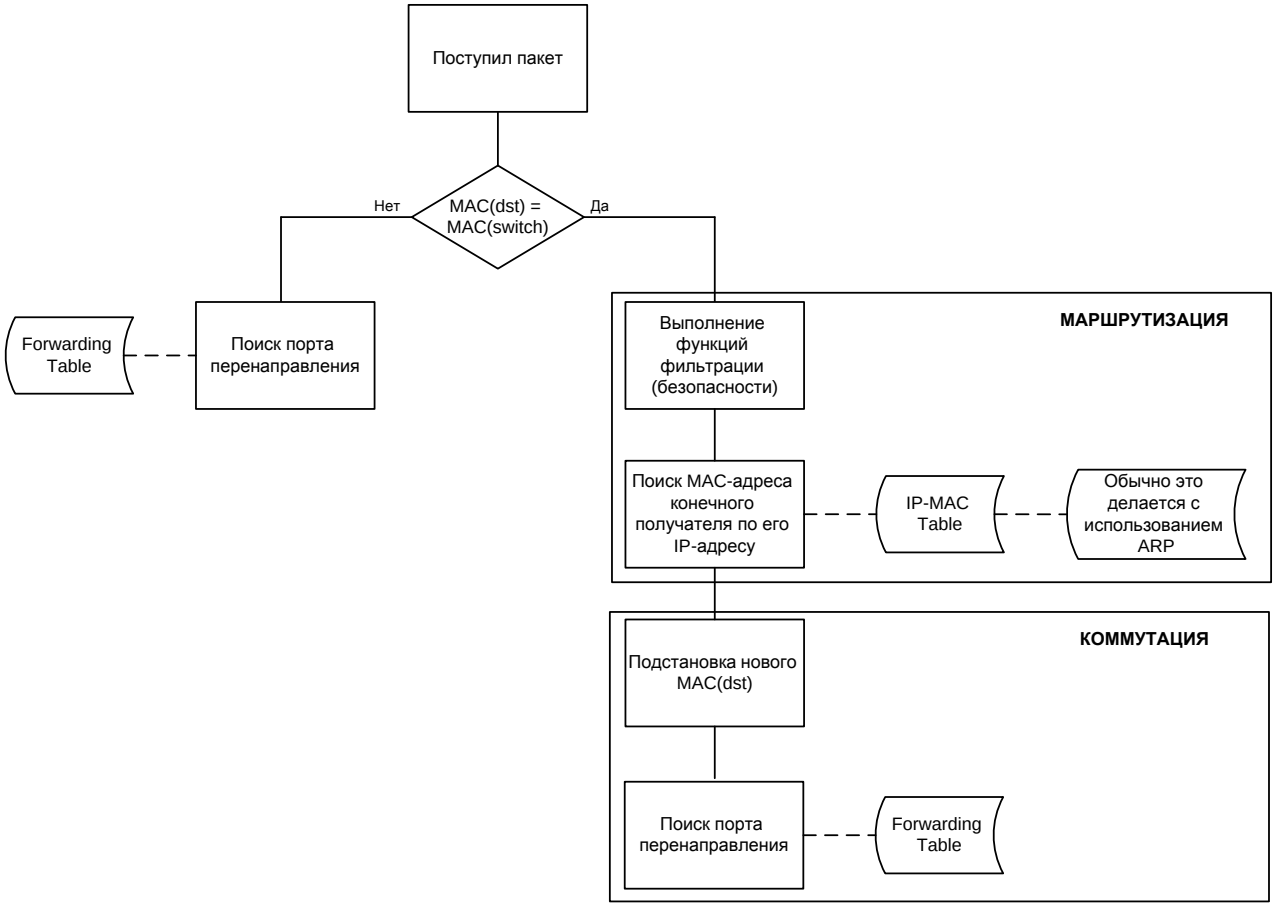


Рисунок 1.38.

Примеры реализации маршрутизирующих коммутаторов

Cabletron

Примерами коммутаторов третьего уровня, работающих по этой схеме, являются коммутаторы SmartSwitch компании Cabletron. Компания Cabletron реализовала в них свой протокол ускоренной маршрутизации SecureFast Virtual Network, SFVN.

Для организации непосредственного взаимодействия рабочих станций без промежуточного маршрутизатора необходимо сконфигурировать каждую из них так, чтобы она считала собственный интерфейс маршрутизатором по умолчанию. При такой конфигурации станция пытается самостоятельно отправить любой пакет конечному узлу, даже если этот узел находится в другой сети. Так как в общем случае (рисунок 1.39) станции неизвестен MAC-адрес узла назначения, то она генерирует соответствующий ARP-запрос, который перехватывает коммутатор, поддерживающий протокол SFVN. В сети предполагается наличие сервера SFVN Server, являющегося полноценным маршрутизатором и поддерживающего общую ARP-таблицу всех узлов SFVN-сети. Сервер возвращает коммутатору MAC-адрес узла назначения, а коммутатор, в свою очередь, передает его исходной станции. Одновременно сервер SFVN передает коммутаторам сети инструкции о разрешении прохождения пакета с MAC-адресом узла назначения через границы виртуальных сетей. Затем исходная станция передает пакет в кадре, содержащем MAC-адрес узла назначения. Этот кадр проходит через коммутаторы, не вызывая обращения к их блокам маршрутизации. Отличие протокола SFVN компании Cabletron от - описанной выше общей схемы в том, что для нахождения MAC-адреса по IP-адресу в сети используется выделенный сервер.

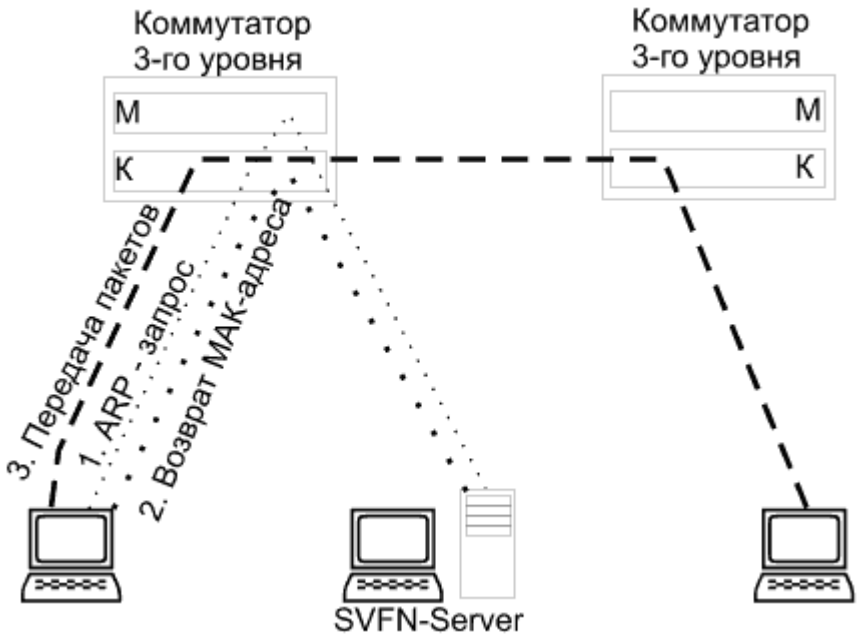


Рисунок 1.39.

3Com

Протокол Fast IP компании 3Com является еще одним примером реализации подхода с отображением IP-адреса на MAC-адрес. В этом протоколе основными действующими лицами являются сетевые адаптеры (что не удивительно, так как компания 3Com является признанным лидером в производстве сетевых адаптеров Ethernet). С одной стороны, такой подход требует изменения программного обеспечения драйверов сетевых адаптеров, и это минус. Но зато не требуется изменять все остальное сетевое оборудование.

При необходимости передать пакет узлу назначения другой сети, исходный узел в соответствии с технологией Fast IP должен передать запрос по протоколу NHRP (Next Hop Routing Protocol) маршрутизатору сети. Маршрутизатор переправляет этот запрос узлу назначения, как обычный пакет Узел назначения, который также поддерживает Fast IP и NHRP, получив запрос, отвечает кадром, отсылаемым уже не маршрутизатору, а непосредственно узлу-источнику (по его MAC-адресу, содержащемуся в NHRP-запросе). После этого обмен идет на канальном уровне на основе известных MAC-адресов. Таким образом, снова маршрутизировался только первый пакет потока, а все остальные коммутировались.

8.5. Коммутаторы потоков

Базовой концепцией коммутации потоков является обнаружение продолжительных потоков данных между двумя IP-узлами. Поток — это последовательность пакетов, имеющих некоторые общие свойства. По меньшей мере, у них должны совпадать адрес отправителя и адрес получателя, и тогда их можно отправлять по одному и тому же маршруту. Когда поток определяется программами уровня 3, между конечными точками организуется коммутируемое соединение и в дальнейшем поток управляется работающим на уровне 2 оборудованием. Копирование файлов или Web-страницы с графикой являются типичными случаями возникновения потоков. Трафик, не удовлетворяющий требованиям потока, маршрутизируется традиционными способами.

Примеры реализации коммутаторов потоков

Cisco

Рассмотрим этот подход на примере технологии NetFlow компании Cisco, реализованной в ее маршрутизаторах и коммутаторах. Для каждого пакета, поступающего на порт маршрутизатора, вычисляется хэш-функция от IP-адресов источника, назначения, портов UDP или TCP и поля TOS, характеризующего требуемое качество обслуживания. Во всех маршрутизаторах, поддерживающих данную технологию, через которые проходит данный пакет, в кэш-памяти портов запоминается соответствие значения хэш-функции и адрес-

ной информации, необходимой для быстрой передачи пакета следующему маршрутизатору. Таким образом, образуется квазивиртуальный канал, который позволяет быстро передавать по сети маршрутизаторов все последующие пакеты этого потока. При этом ускорение достигается за счет упрощения процедуры обработки пакета маршрутизатором - не просматриваются таблицы маршрутизации, не выполняются ARP-запросы. Этот прием может использоваться в маршрутизаторах, вообще не поддерживающих коммутацию, а может быть перенесен в коммутаторы. В этом случае такие коммутаторы тоже называют коммутаторами третьего уровня.

8.6. Выводы

У коммутатора третьего уровня, кроме реализации функций маршрутизации в специализированных интегральных схемах, имеется несколько особенностей, отличающих их от традиционных маршрутизаторов. Эти особенности отражают ориентацию коммутаторов 3-го уровня на работу, в основном, в локальных сетях, а также последствия совмещения в одном устройстве коммутации на 2-м и 3-м уровнях:

- поддержка интерфейсов и протоколов, применяемых в локальных сетях;
- усеченные функции маршрутизации;
- обязательная поддержка механизма виртуальных сетей.

Так как коммутаторы 3 уровня созданы для обработки интенсивного трафика локальных сетей, то они могут быть расположены в любом месте сетевого ядра или магистрали. При этом они являются эффективным и выгодным в плане стоимости решением замены традиционных магистральных маршрутизаторов. Коммутаторы 3 уровня взаимодействуют с маршрутизаторами глобальных сетей, используя стандартные протоколы маршрутизации типа RIP и OSPF. Таким образом, современные коммутаторы 3 уровня могут быть помещены в любом месте локальной сети, где стоят маршрутизаторы. То есть они являются полноценной заменой маршрутизаторов в локальных сетях, при этом последние проигрывают им как в производительности, так и в ценовом отношении.

В таблице 1.1 приведено сравнение коммутаторов 3 уровня и маршрутизаторов. Коммутатор 3 уровня оптимизирован для работы в локальных сетях и не ориентирован на работу в глобальных сетях и поддержку соединений с данными сетями. Эта оптимизация повышает производительность коммутатора 3 уровня (далее КЗУ) в среднем в 10 раз по сравнению с маршрутизаторами, в то время как КЗУ в среднем в 3 раза дешевле маршрутизатора. При этом следует учесть, что обучение администрирования КЗУ также дешевле.

Таблица 1.1 – Сравнительные характеристики коммутаторов 3-го уровня и маршрутизаторов

Характеристика	**Коммутатор 3 уровня**	**Маршрутизатор**
Поддержка протоколов Сетевого уровня	Да	Да
Архитектура коммутации	Аппаратная	Программная
Цена	Низкая	Высокая
Производительность	Высокая	Низкая
Поддержка соединений с глобальной сетью	Нет	Да

Фирма Intel произвела тестирование своих коммутаторов, результаты которого представлены на рисунке 1.40. Видно, что коммутаторы 2 уровня по производительности немного отстают скорости передачи данных в канале. В свою очередь коммутаторы 3 уровня немного отстают по производительности от коммутаторов 2 уровня.

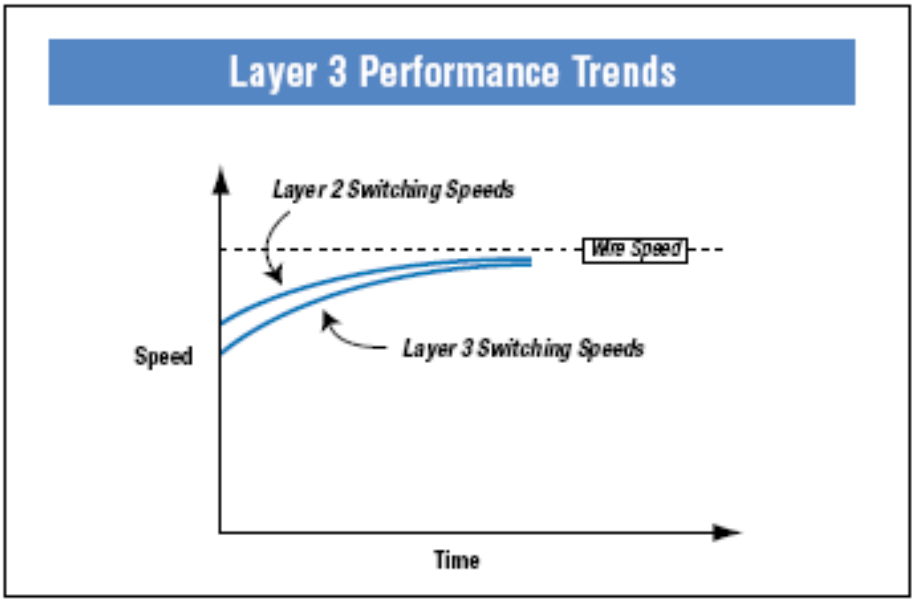

Layer 3 Performance Trends
Layer 2 Switching Speeds
Wire Speed
Layer 3 Switching Speeds
Speed
Time

Рисунок 1.40.

9. Способы управления коммутаторами

9.1. Технология Single IP Management

Введение

Объединение устройств в стек требует наличия специальных модулей и кабелей для стекирования, что ограничивает возможность включения в стек коммутаторов различных моделей, кроме того требуется установка коммутаторов в один монтажный шкаф. Устранить эти ограничения позволяет использование новой технологии Single IP Management.

Технология Single IP Management (SIM) – это технология управления виртуальным стеком через единый IP-адрес (рисунок 41).

Рисунок 1.41.

Технология SIM позволяет:

- ✓ устранить ограничения на модели коммутаторов, объединяемых в стек;
- ✓ уменьшить количество управляющих IP-адресов сети;
- ✓ устранить необходимость использования специализированных модулей и кабелей, предназначенных для стекирования;
- ✓ преодолеть ограничения, связанные с длиной кабелей в стеке.

В отличие от стеков, построенных с использованием традиционных методов стекирования, виртуальный стек на основе технологии SIM позволяет включить в группу большее количество коммутаторов. Например, компания D-Link позволяет включить до 32 коммутаторов в виртуальный стек, в то время как традиционные стеки того же производителя ограничены максимум 12 коммутаторами. При этом виртуальный стек может быть расширен коммутаторами разного типа – от недорогих коммутаторов 2-го уровня до высокопроизводительных коммутаторов на основе шасси (для ядра сети).

Объединение коммутаторов в SIM-стек не требует использования специальных соединительных кабелей. Трафик, передаваемый между устройствами стека, проходит через полнодуплексные интерфейсы Fast Ethernet, Gigabit Ethernet или 10 Gigabit Ethernet по обычным медным или оптическим кабелям. Отказ от использования специализированных стекирующих кабелей позволяет преодолеть ограничения, связанные с их длиной. В стек могут быть объединены устройства, расположенные в любом месте сети. Расстояния между узлами виртуального стека определяется лишь ограничениями соответствующего стандарта IEEE 802.3 и может достигать десятки километров.

Ниже (рисунок 1.42) приведена архитектура SIM.

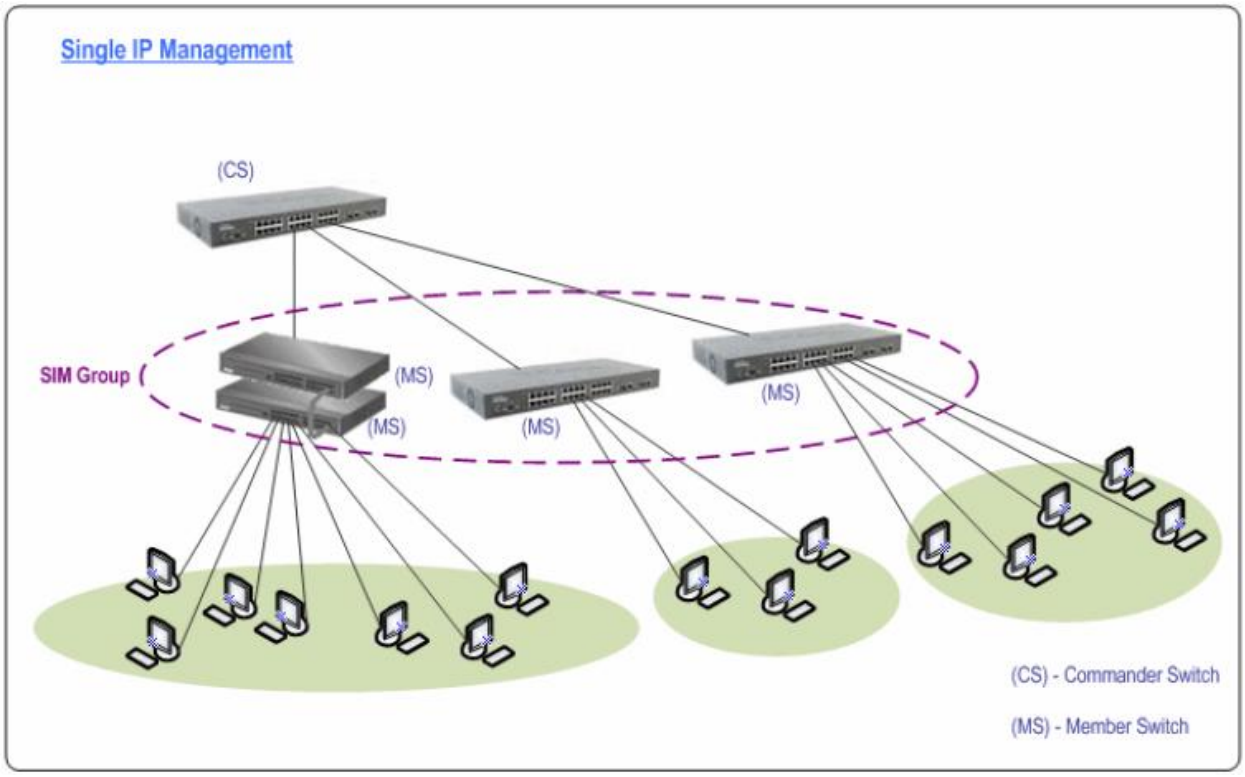


Рисунок 1.42.

Группа SIM состоит из трёх компонент:
- ✓ Commander Switch (CS) – командный коммутатор;
- ✓ Member Switch (MS) – коммутатор-участник;
- ✓ Candidate Switch (CaS) – коммутатор-кандидат.

Каждая группа SIM состоит из одного командного коммутатора и максимум 32 коммутаторов-участников.

Командный коммутатор используется для управления всеми коммутаторами в группе SIM и обладает следующими характеристиками:
- ✓ имеет назначенный IP-адрес;
- ✓ не является командным коммутатором или коммутатором участником другой группы SIM;
- ✓ подсоединён к коммутаторам-участникам через собственную управляющую VLAN.

Коммутатор-участник является коммутатором, который входит в группу SIM, доступен с командного коммутатора и обладает следующими характеристиками:
- ✓ не является командным или коммутатором-участником другой группы SIM;
- ✓ подсоединён к другим коммутаторам-участникам через общую VLAN.

Коммутатор-кандидат – это коммутатор, который готов вступить в группу SIM, но пока не является членом ни одной группы. Коммутатор-кандидат может вступить в группу SIM, используя автоматическую функцию, встроенную в SIM-коммутаторы, или путём ручной настройки. Коммутатор, сконфигурированный как CaS, не является членом SIM и обладает следующими характеристиками:
- ✓ не является командным или коммутатором-участником другой группы SIM;
- ✓ подсоединён к другим коммутаторам-участникам через общую VLAN.

После настройки одного коммутатора в качестве управляющего SIM-группы, другие коммутаторы могут стать членами группы через непосредственное подключение к управляющему коммутатору. Только управляющий коммутатор может обращаться к CaS, он является своеобразной точкой доступа к членам группы. IP-адрес управляющего комму-

татора станет адресом для всех членов группы, управление же доступом ко всем членам группы будет осуществляться через пароль администратора CS и/или аутентификацию. Когда функция SIM включена, приложения управляющего коммутатора будут перенаправлять пакеты вместо их обработки. Приложения будут декодировать пакет от администратора, видоизменять некоторые данные и затем отправлять его членам группы. После выполнения этих действий управляющий коммутатор может получить ответный пакет, который закодирует и отправит обратно администратору. После того, как управляющий коммутатор станет обыкновенным членом SIM-группы, он будет членом первой SNMP-группы (включая права чтения/записи и права только чтения), к которой принадлежал управляющий коммутатор. Однако если у коммутатора MS есть свой собственный IP-адрес, то он может принадлежать к SNMP-группе, в которой другие коммутаторы SIM-группы не состоят.

Версии SIM

Существуют следующие версии SIM: 1.0, 1.5 и 1.6. Ниже они будут рассмотрены в сравнении.

Версии 1.0 и 1.5

Версия SIMv1.5 предлагает следующие улучшения по сравнению с версией 1.0:

- ✓ Возможность сохранения списка всех коммутаторов-участников в энергонезависимой памяти командного коммутатора. В версии SIMv1.0 командный коммутатор не имел возможности сохранять информацию о коммутаторах-участниках в своей памяти. Следовательно, если коммутатор-участник перегружался, он принимал статус коммутатора CaS. В версии SIMv1.5, если информация о коммутаторе-участнике была сохранена в памяти командного коммутатора, то после перезагрузки коммутатору-участнику автоматически присваивался статус MS. Если же перегружается командный коммутатор, то он заново собирает информацию о топологии сети.
- ✓ Возможность отображения информации о магистральных группах (trunks) в топологической карте сети. В версии SIMv1.0 отображалась только пропускная способность порта вне зависимости от его принадлежности к магистральной группе. Версия SIMv1.5 показывает пропускную способность всей магистральной группы, а не порта. Когда организована магистральная группа, на топологической карте отражается множество линий.
- ✓ Меньший размер программного обеспечения.
- ✓ Поддержка масштабирования топологической карты при просмотре через web-браузер.
- ✓ Поддержка нескольких конфигурационных загрузочных файлов.

Версии 1.5 и 1.6

Несмотря на улучшения по сравнению с предыдущей версией версия SIMv1.5 имеет несколько недостатков, связанных с безопасностью. Большинство подобных недостатков связаны с передачей пакетов без возможности их шифрования. Таким образом, версия SIMv1.6 призвана улучшить безопасность технологии SIM путём добавления механизмов шифрования/дешифрования. Существует обратная совместимость версий 1.6 и 1.5.

Формат пакета

Ниже (рисунок 1.43) приведён формат пакета для версии 1.6. Данный формат пакета применим для пакетов обнаружения (Discovery Packets), пакетов отчёта (Report Packets), пакетов поддержки (Maintence Packets), конфигурационных пакетов (Configuration Packets) и пакетов перенаправления (Redirection Packets) и не применим для пакетов построения топологии (Topology Packets). Также стоит отметить, что, когда устройство, не поддерживающее шифрование, получает зашифрованный пакет, то данное устройство будет отбрасывать пакеты обнаружения и отчётов.

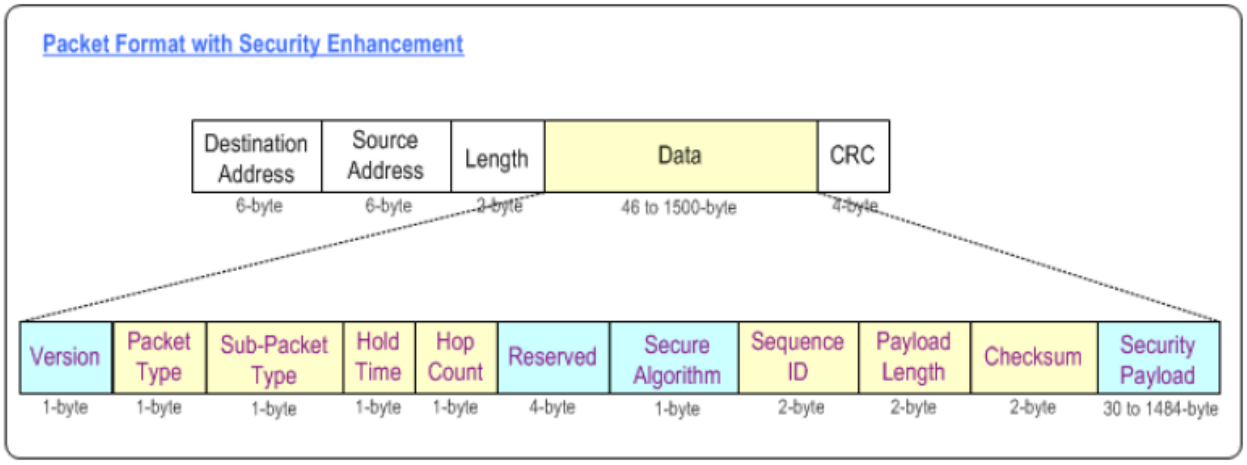


Рисунок 1.43.

Назначение полей в пакете формата secure то же самое, что в пакете формата unsecure, за исключением следующих полей:

- Version – версия пакета SIM. Версия пакета представленного формата – 0x02.
- Reserved – длина этого поля меняется от 5 до 4 байт. Значение поля всегда составляет 0x00.
- Secure Algorithm – алгоритм шифрования:
  - ✓ 0x00 – отсутствует;
  - ✓ 0x01 – внутренний алгоритм D-Link (XOR);
  - ✓ 0x01 … 0xFF – зарезервировано для дальнейшего использования.
- Payload – полезная зашифрованная нагрузка.

Операции SIMv1.6

Операции SIM версии 1.6 разделены на три уровня (этапа):

- ✓ Collection – сбор информации;
- ✓ Maintenance – поддержка;
- ✓ Managment – управление.

Этап сбора информации (Collection Stage)

На данном этапе командный коммутатор CS периодически рассылает пакеты отчёта Report (как зашифрованные, так и нет) коммутатором-кандидатам CaS, а кандидаты с той же периодичностью рассылают пакеты обнаружения Discovery. Пакеты Report/Discovery с поддержкой механизма безопасности будут зашифрованы алгоритмом по умолчанию. Дополнительно некоторая информация о механизмах безопасности включается в полезную нагрузку. Это данные о максимальных и предпочитаемых уровнях безопасности, которые поддерживает устройство.

Следующая диаграмма (рисунок 1.44) иллюстрирует операции, производимые коммутатором CS после получения данных от коммутаторов CaS:

1. Коммутатор CaS посылает одновременно два Discovery-пакета (один зашифрованный, другой – нет).
2. Когда коммутатор CS получает незашифрованный пакет, он ждёт 20 секунд до получения зашифрованного Discovery-пакета. Если в течение данного интервала времени зашифрованный пакет будет получен, то командный коммутатор определит коммутатор CaS как коммутатор, поддерживающий механизмы безопасности. Иначе коммутатор CaS будет определён как коммутатор, не поддерживающий механизмы безопасности.
3. Далее информация о коммутаторе CaS будет записана в базу данных коммутаторов CaS.

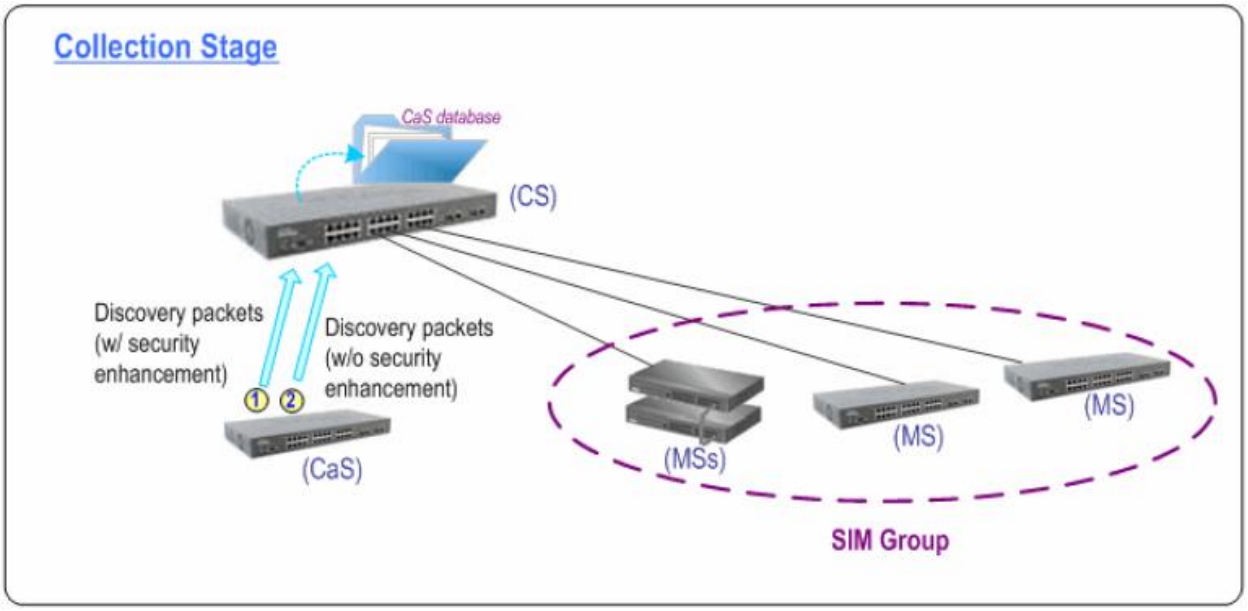


Рисунок 1.44.

Этап поддержки (Maintenance Stage)

После обмена Discovery/Report-пакетами наступает этап поддержки. Следующий рисунок 1.45 показывает операции, выполняемые в командном коммутаторе, по добавлению и конфигурированию коммутаторов CaS как членов его SIM-группы и операции, выполняемые в коммутаторе CaS, по получению команды для присоединения к SIM-группе.

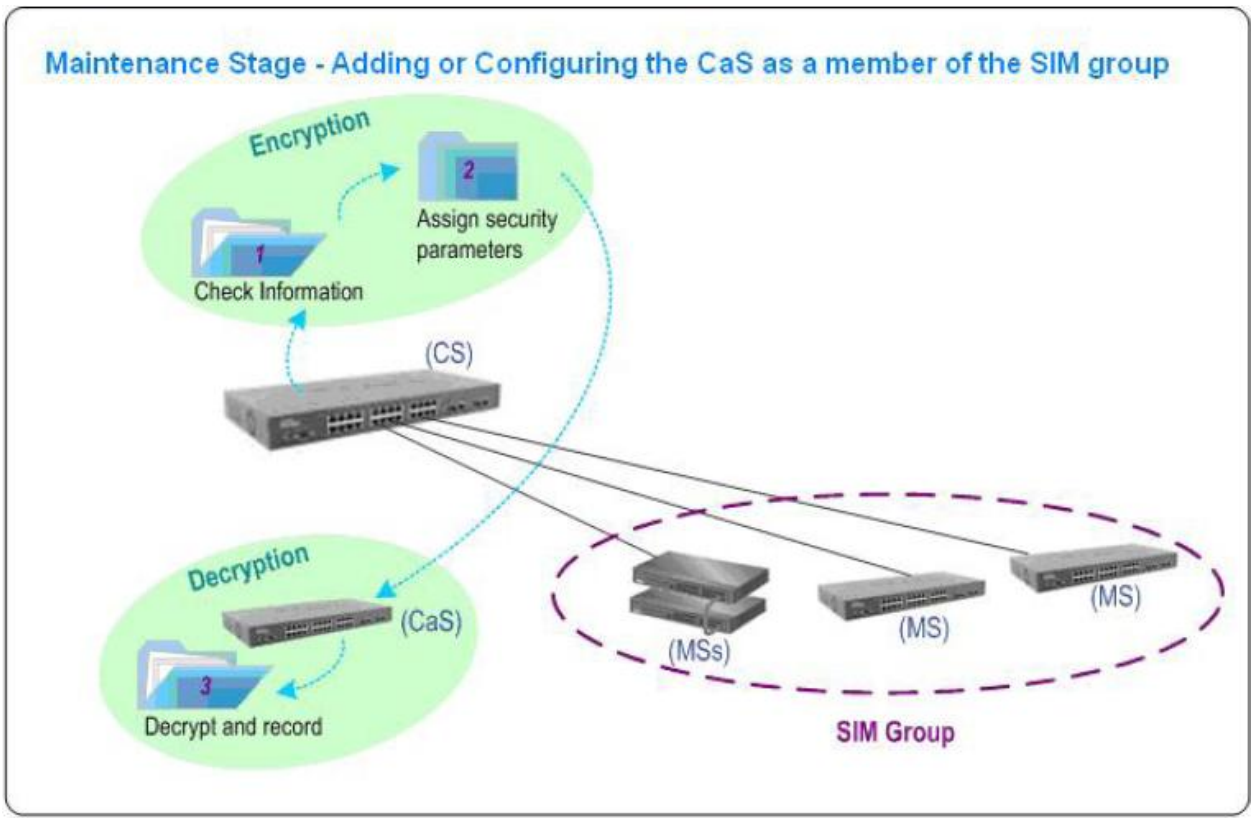


Рисунок 1.45.

Этап управления

Назначение данного этапа перевести коммутатора-кандидата в роль коммутатора-участника (MS). Новый коммутатор MS примет и последует методу обмена информацией, установленном на данном этапе, для взаимодействия с остальными коммутаторами-участниками и командным коммутатором своей группы.

Взаимодействия в топологии SIM

Взаимодействие между коммутаторами в топологии SIM могут осуществляться в 5 различных режимах (в зависимости от того, поддерживает ли командный коммутатор механизмы безопасности или нет):

1. Когда командный коммутатор не поддерживает механизмы безопасности, всё взаимодействие между ним и остальными участниками группы происходит без применения шифрования. Возможные режимы обмена для данного случая показаны на рисунке 1.46.

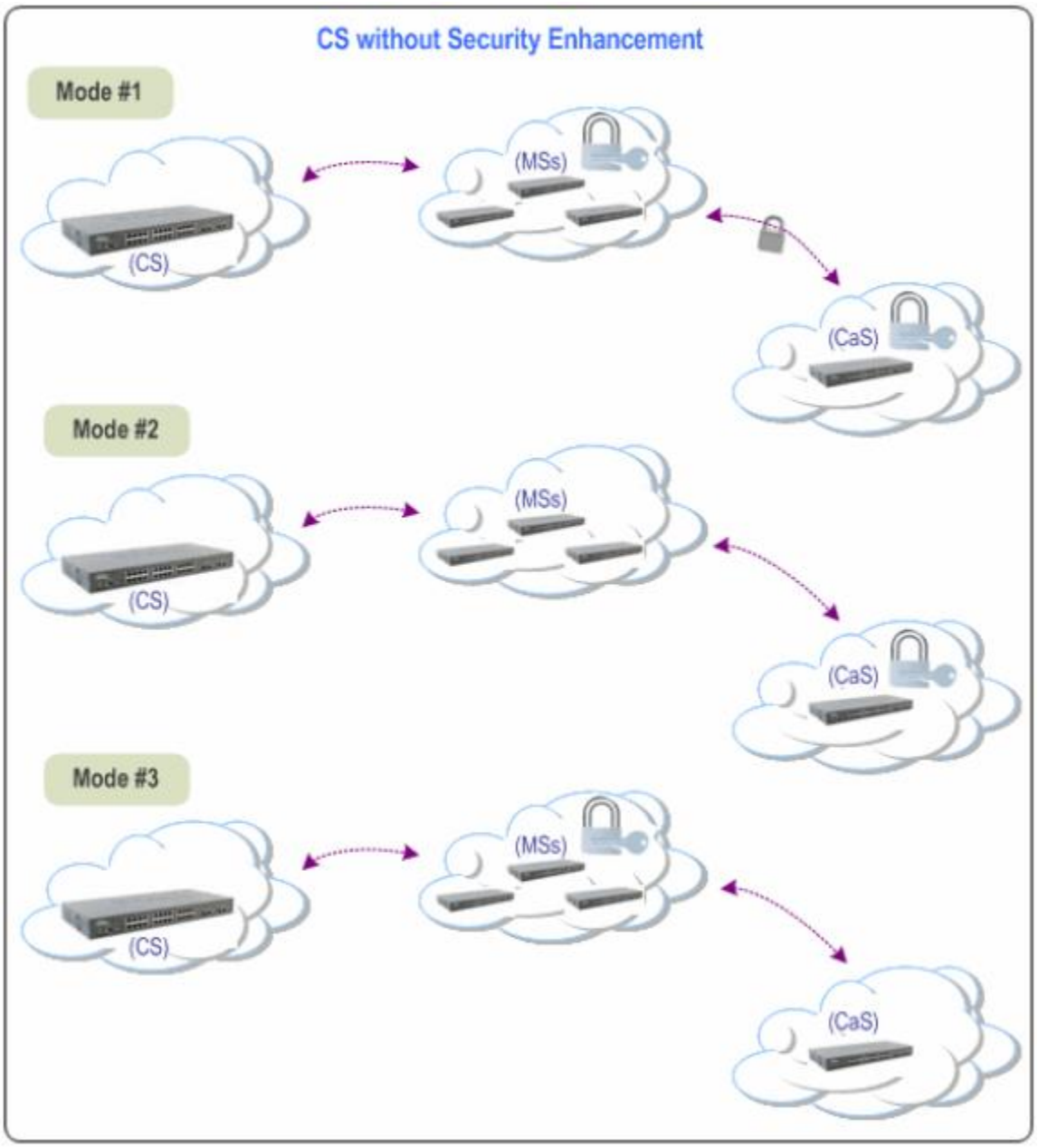


Рисунок 1.46.

2. Когда все коммутаторы группы (CS, MSs, CaSs) поддерживают механизмы безопасности, все взаимодействия между ними производятся только в безопасном режиме. Данная концепция проиллюстрирована в режиме 1 (Mode #1) на рисунке 1.47. Остальные режимы иллюстрируют ситуации, когда тот или иной член группы не поддерживает механизмы безопасности.

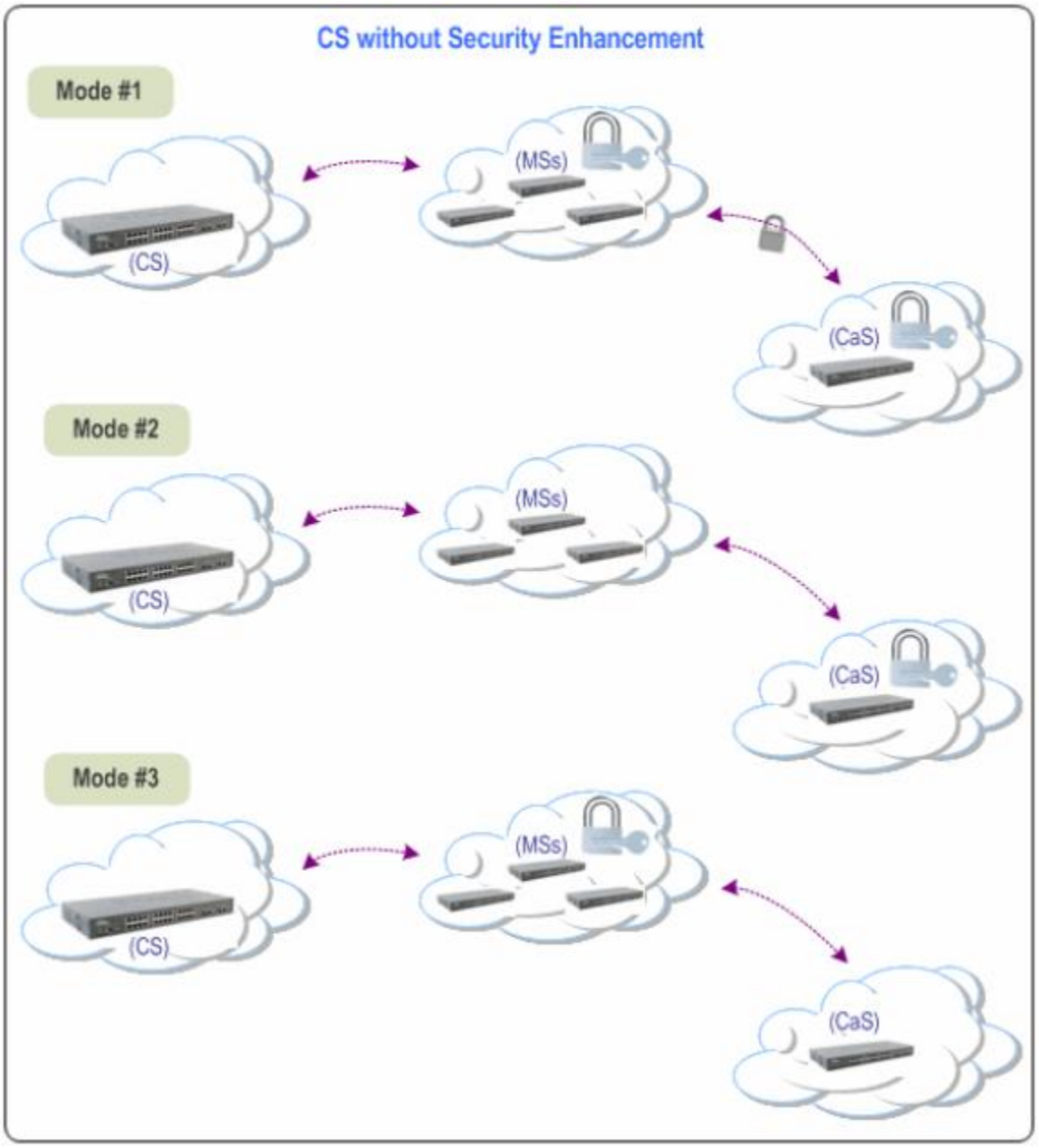

CS without Security Enhancement
Mode #1
(MSs)
(CS)
(CaS)
Mode #2
(MSs)
(CS)
(CaS)
Mode #3
(MSs)
(CS)
(CaS)

Рисунок 1.47.

II. Технологии обеспечения безопасности передачи данных в сетях Ethernet

1. Ограничение количества управляющих компьютеров

Современные коммутаторы позволяют задать конкретные компьютеры (IP-адреса), с которых будет происходить настройка данных коммутаторов по Web- или Telnet- инерфейсам или через протокол SNMP. Попытка прочих компьютеров подключиться к коммутаторам для управления ими будет отклонена.

2. Настройка безопасности индивидуального порта

Данная функция позволяет:

- заблокировать дальнейшее обновление таблицы коммутатора – если конфигурирование Вашей сети больше не изменятся, Вы блокируете таблицу коммутации, и коммутатор будет просто отбрасывать все пакеты, которые поступают с неизвестных адресов. Причём данное действие можно выбрать для конкретных портов. При этом записи в таблице коммутации не будут удаляться по тайм-ауту;
- задать максимальное количество MAC-адресов для привязки к конкретному порту.

3. Фильтрация MAC-адресов

Дополнительно можно настроить коммутатор так, чтобы он не принимал пакеты с определенным MAC-адресом (как получателя, так и отправителя). Для этого служит таблица фильтрации трафика (Filtering Table). Фактически данная таблица является «чёрным списком». После получения пакета коммутатор считывает оба аппаратных адреса, как получателя, так и отправителя. Если хотя бы один из них содержится в таблице фильтрации, то пакет отбрасывается.

4. Технология фильтрации IP-MAC Binding

Основное назначение данной технологии – это ограничить доступ к коммутатору определённого количества компьютеров. Для этого индивидуально для каждого желаемого порта создаётся таблица соответствия IP- и MAC-адресов. Далее коммутатор будет пропускать только пакеты с указанными MAC-адресами и только в том случае, если истинно соответствие IP- и MAC-адреса.

Существенные отличия данной технологии от технологии фильтрации MAC-адресов:

- фильтрация работает на всех портах, а в данной технологии можно настраивать каждый порт в отдельности;
- фильтрация содержит «чёрный» список, то есть работает по принципу: не пропускать те пакеты, MAC-адреса которых содержатся в таблице фильтрации. Данная технология, наоборот, содержит «белый» список, то есть пропускает только те пакеты, MAC-адреса которых содержатся в созданных списках;
- в отличие от фильтрации технология IP-MAC Binding проверяет не только MAC-адрес источника пакета, но и его IP-адрес.

5. Списки контроля доступа (Access Control Lists)

Списки контроля доступа обеспечивают ограничение прохождения трафика через коммутатор. Фактически технология ACL реализует на коммутаторах 3-го уровня полноценный фильтр пакетов. Приём пакетов или отказ в приёме основывается на определённых признаках, которые содержит пакет:

- ✓ IP- и MAC-адреса источника и приёмника;
- ✓ номер VLAN;
- ✓ номер порта TCP или UDP;
- ✓ тип ICMP-сообщения.

Ключевым понятием в данной технологии является понятие профиля доступа – это набор признаков, который содержит пакет, с определёнными значениями. Каждый профиль доступ имеет свой уникальный номер в пределах коммутатора. Пример профилей доступа:

1: <VLAN = Yes, Source MAC = 00-01-08-DE-FA-10, Destination MAC = 00-01-07-DE-FA-10>
2: <VLAN = Yes, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = Yes>

Помимо приведенных примеров существуют также контекстно-зависимые профили. Основное отличие контекстно-зависимых профилей от всех остальных заключается в том, что в них признаки задаются смещением относительно начала кадра Ethernet. Например, чтобы указать MAC-адрес источника пакета в контекстно-зависимом профиле необходимо указать смещение Offset = 6 байтам и далее значение MAC-адреса, так как данный адрес располагается в заголовке кадра Ethernet с 7 по 12 байт включительно.

Профиль является всего лишь образцом (некоторым эталоном), на основе которого создаются правила. Правила представляют собой профиль, заполненный конкретными значениями признаков и содержащий действие, которое необходимо выполнить над пакетом, если он попадает под эти признаки. Обычно действие – это удалить или продвинуть пакет. Для каждого профиля может быть создано более одного правила. Например, для вышеприведённого профиля с ID=2 могут быть созданы следующие правила:

Accept: <VLAN = Default, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = 514>
Deny: <VLAN = Marketing, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = 8080>

При поступлении пакета на коммутатор профили доступа применяются последовательно, в порядке возрастания их номеров. Пакет проверяется на соответствие условиям, указанным во всех правилах профиля, начиная с первого профиля. Если правило подходит, пакет в соответствие с действием обработки либо принимается, либо отбрасывается и дальше не проверяется. Если пакет не попадает ни под одно правило профиля, то пакет проверяется по правилам следующего профиля. Если не один профиль не подходит, то применяется политика по умолчанию – разрешающая или запрещающая прохождение трафика.

Очень важно правильно расставлять профили и правила внутри профилей потому, что если пакет попадает хотя бы под одно запрещающее правило, то он удаляется из коммутатора и дальше не проверяется на соответствие другим правилам, даже если среди них есть правила, которые могли бы его пропустить дальше.

6. Сегментация трафика (Traffic Segmentation)

Сегментация трафика служит для разграничения портов на Канальном уровне. Данная функция позволяет настраивать порты или группу портов таким образом, чтобы они были изолированы друг от друга, но в то же время имели доступ к разделяемым портам, используемым для подключения серверов или магистрали сети провайдера. Очень важной является возможность одновременного использования данной функции с виртуальными сетями (VLAN), что позволяет производить дальнейшее разграничение прав.

Данная технология схожа с технологией VLAN, но является более ограниченной по функциональности. К преимуществам технологии можно отнести:

- простота настройки;
- более низкая загрузка процессора коммутатора по сравнению с VLAN.

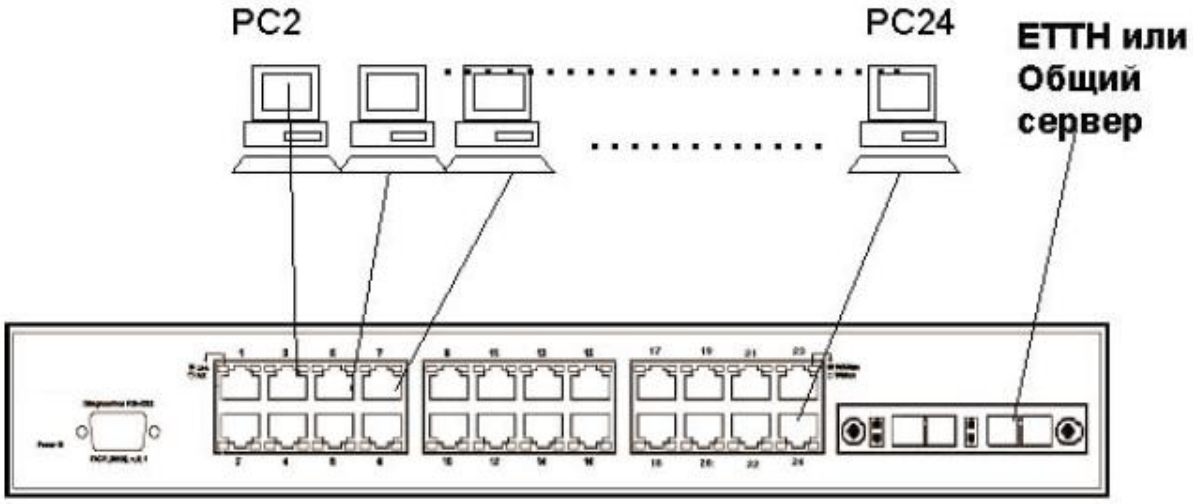


Рисунок 2.1.

Рассмотрим пример (рисунок 2.1). Все компьютеры имеют доступ к общему порту, но не имеют доступа к друг другу на Канальном уровне. Это решение может быть использовано в:

- в проектах ETTH (Ethernet To The Home, Ethernet до дома) для изоляции компьютеров конечных пользователей;
- для предоставления доступа к общему серверу.

7. Протокол IEEE 802.1x

Протокол IEEE 802.1x является механизмом безопасности, обеспечивающим аутентификацию и авторизацию пользователей и тем самым ограничивающим доступ проводных или беспроводных устройств к локальной сети. Работа протокола базируется на клиент-серверной модели контроля доступа (рисунок 2.2). В качестве сервера аутентификации используется RADIUS-сервер. При этом весь процесс аутентификации пользователя производится в проводных сетях на основе протокола EAPOL (Extensible Authentication Protocol over LAN), в беспроводных – на основе протокола EAPOW (Extensible Authentication Protocol over Wireless).

Рисунок 2.2.

До тех пор, пока клиент не будет аутентифицирован, протокол IEEE 802.1x будет пропускать через сетевой порт только трафик протокола EAPOL. После успешной аутентификации обычный трафик будет пропускаться через порт. Работа протокола IEEE 802.1x основывается на трёх компонентах (рисунок 2.2), каждая из которых подробно рассмотрена в следующем разделе.

7.1. Роли устройств

Клиент – это рабочая станция, которая запрашивает доступ к локальной сети и сервисам коммутатора и отвечает на запросы коммутатора. На рабочей станции должно быть установлено клиентское ПО, реализующее протокол 802.1x (в ОС Microsoft Windows XP данное ПО является встроенным).

Сервер аутентификации выполняет фактическую аутентификацию клиента, проверяя подлинность клиента и информируя коммутатор, предоставлять или нет клиенту доступ к локальной сети.

Коммутатор (также называется аутентификатор) управляет физическим доступом к сети, основываясь на статусе аутентификации клиента. Коммутатор работает как посредник между клиентом и сервером аутентификации, получая запрос на проверку подлинности от клиента, проверяя данную информацию при помощи сервера аутентификации, и пересылая ответ клиенту. ПО коммутатора включает клиента RADIUS, который отвечает за инкапсуляцию и деинкапсуляцию кадров EAP и взаимодействие с сервером аутентификации.

7.2. Процесс аутентификации

Инициировать процесс аутентификации может коммутатор или клиент. Клиент инициирует аутентификацию, посылая кадр EAPOL-start, который вынуждает коммутатор отправить ему запрос на идентификацию. Когда клиент отправляет EAP – ответ со своей идентификацией, коммутатор начинает играть роль посредника, передающего кадры EAP между клиентом и сервером аутентификации до успешной или неуспешной аутентификации. Если аутентификация завершилась успешно, порт коммутатора становится авторизованным.

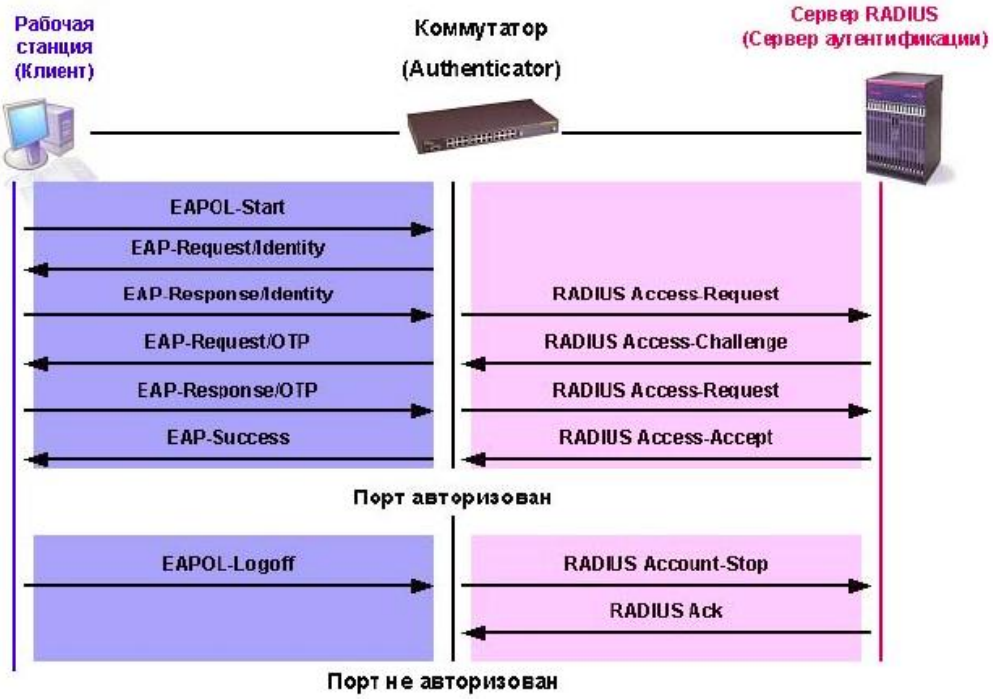


Рисунок 2.3. Временная диаграмма аутентификации клиента в сети.

Временная диаграмма обмена EAP-кадрами зависит от используемого метода аутентификации. На рисунке 2.3 показана схема обмена, инициируемая клиентом, использующая метод аутентификации с использованием одноразовых паролей (One Time Password, OTP) сервером RADIUS.

7.3. Состояние портов коммутатора

Состояние порта коммутатора определяется тем, получил или не получил клиент право доступа к сети. Первоначально порт находится в неавторизованном состоянии. В этом состоянии он запрещает прохождение всего входящего и исходящего трафика за исключением пакетов протокола IEEE 802.1x. Когда клиент аутентифицирован, порт переходит в авторизованное состояние, позволяя передачу любого трафика от него.

Возможны варианты, когда клиент или коммутатор не поддерживают протокол IEEE 802.1x. Если клиент, который не поддерживает протокол IEEE 802.1x, подключается к неавторизованному порту, коммутатор посылает клиенту запрос на аутентификацию. Поскольку в этом случае клиент не ответит на запрос, порт останется в неавторизованном состоянии и клиент не получит доступ к сети.

В другом случае, когда клиент с поддержкой протокола IEEE 802.1x подключается к порту, на котором не запущен протокол IEEE 802.1x, клиент начинает процесс аутентификации, посылая кадр EAPOL-start. Не получив ответа, клиент посылает запрос определённое количество раз. Если после этого ответ не получен, клиент, считая, что порт находится в авторизованном состоянии начинает посылать кадры.

В случае, когда и клиент и коммутатор поддерживают протокол IEEE 802.1x, при успешной аутентификации клиента, порт переходит в авторизованное состояние и начинает передавать все кадры клиента. Если в процессе аутентификации возникли ошибки, порт остаётся в неавторизованном состоянии, но аутентификация может быть восстановлена. Если сервер аутентификации не может быть достигнут, коммутатор может повторно передать запрос. Если от сервера не получен ответ после определённого количества попыток, то в доступе к сети будет отказано из-за ошибок аутентификации.

Когда клиент завершает сеанс работы, он посылает сообщение EAPOL-logoff, переводящее порт коммутатора в неавторизованное состоянии. Если состояние канала связи порта переходит из активного (up) в неактивное (down), то порт также возвращается в неавторизованное состояние.

7.4. Методы контроля доступа при использовании протокола IEEE 802.1x

Протокол IEEE 802.1x предоставляет два метода контроля доступа к сети:

1. На основе портов (Port-Based Access Control). При использовании данного метода достаточно, чтобы только один любой пользователь, подключенный к порту коммутатора, был авторизован. Тогда порт перейдёт в авторизованное состояние и доступ к сети получат любые пользователи, подключенному к данному порту.
2. На основе MAC-адресов (MAC-Based Access Control). При использовании данного метода при аутентификации также учитывается MAC-адрес клиента, подключенного к порту, и порт авторизуется только для клиента с конкретным MAC-адресом.

Контроль доступа на основе портов

Изначально протокол IEEE 802.1x разрабатывался с учётом того, что к порту коммутатора подключено не более одного устройства (рисунок 2.4). Как только устройство успешно проходило процедуру аутентификации, порт переходил в авторизованное состояние и далее пропускал весь трафик до тех пор, пока не наступало событие, которое обратно переводило его в неавторизованное состояние. Следовательно, если порт коммутатора подключен не к одному устройству, а к сегменту локальной сети, то успешная аутентификация любого устройства из этого сегмента открывает доступ в сеть всем остальным устройствам из сегмента. Естественно, это является серьёзной проблемой с точки зрения безопасности.

Контроль доступа на основе MAC-адресов

Для того, чтобы успешно использовать протокол IEEE 802.1x в распределённых локальных сетях, необходимо создавать логические порты – по одному логическому порту на каждое устройство, подключенное к физическому порту. Таким образом, физический порт представляет собой множество логических портов, каждый из которых независимо контролирует отдельное устройство-клиента с точки зрения аутентификации и авторизации. Принадлежность устройства к определённому логическому порту осуществляется на основе MAC-адреса устройства (рисунок 2.5). Таким образом, устраняется проблема безопасности доступа множества устройств через один физический порт коммутатора.

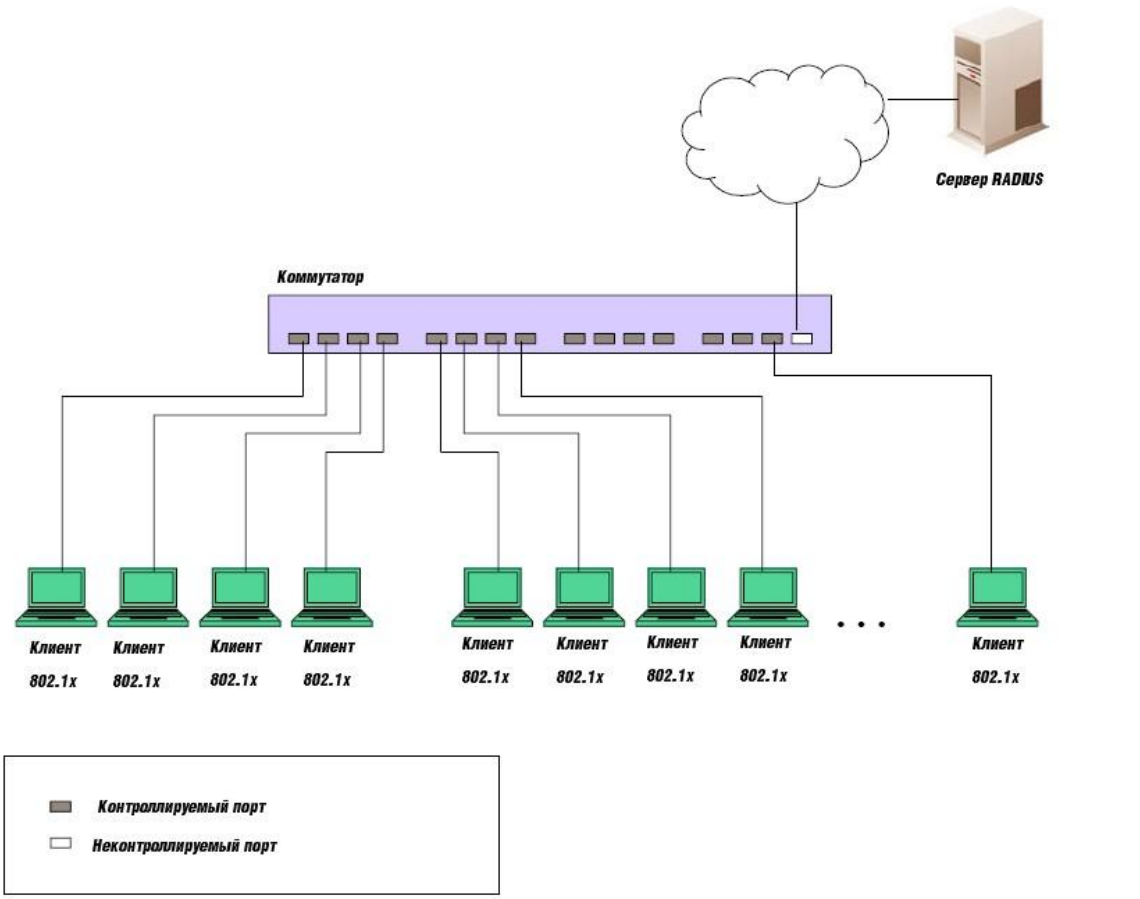


Рисунок 2.4.

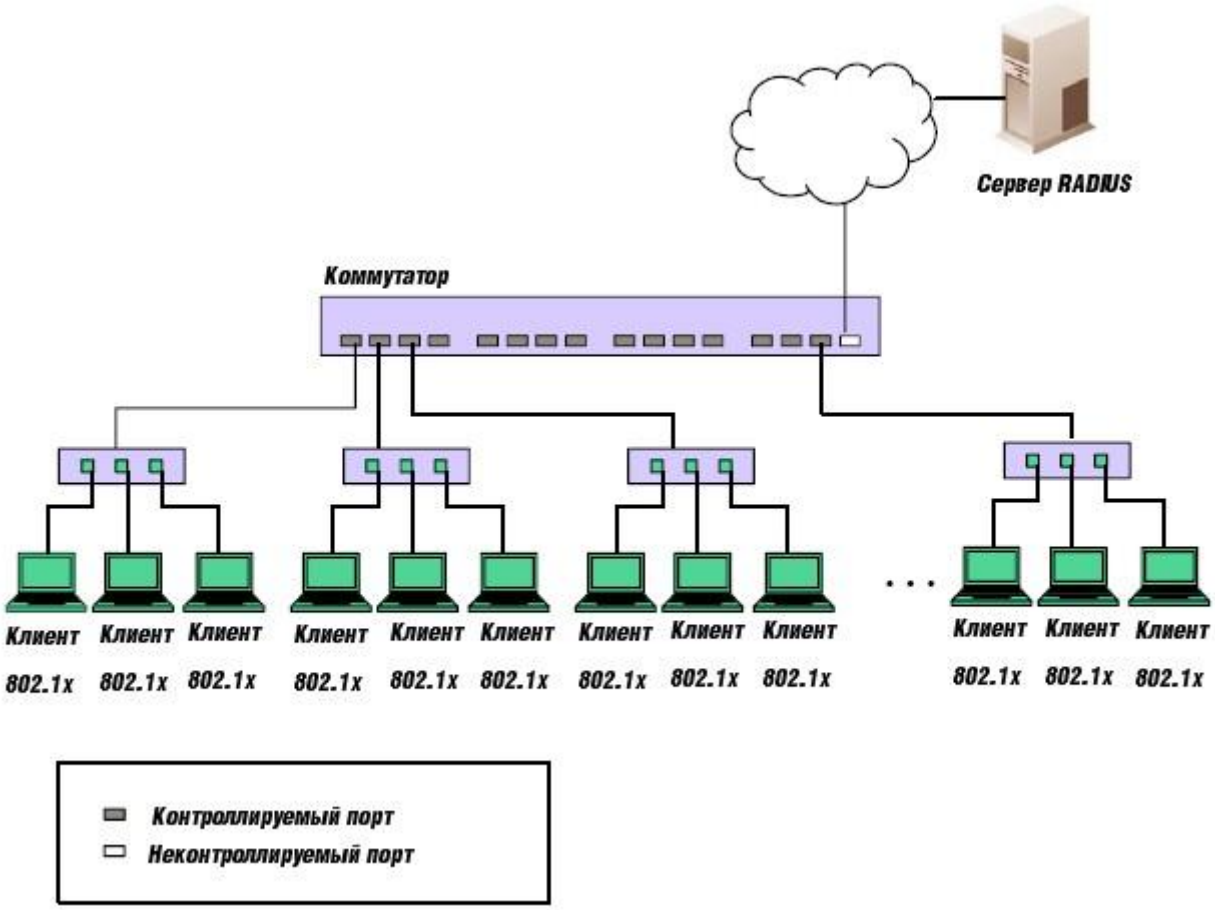


Рисунок 2.5.

III. Технологии передачи данных в сетях TCP/IP

Стек протоколов TCP/IP предшествовал эталонной модели OSI, но, тем не мене, протоколы можно разделить на четыре уровня, которые будут в общих чертах аналогичны семиуровневому стеку OSI (рисунок 3.1).

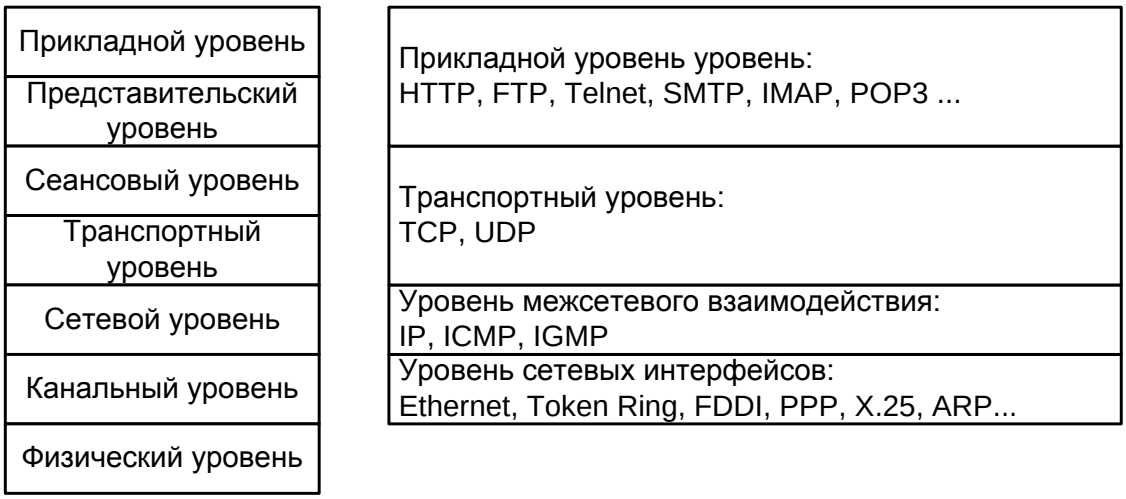


Рисунок 3.1. Архитектура стек протоколов TCP/IP.

В первую очередь отметим основные функции уровней согласно нотации 7-и уровневой модели ISO/OSI:

- ✓ Физический уровень – представляет собой среду и аппаратуру передачи данных;
- ✓ Канальный уровень – передача пакета в пределах ЛВС;
- ✓ Сетевой уровень – передача пакета до пункта назначения;
- ✓ Транспортный уровень – передача пакета конкретному приложению;
- ✓ Сеансовый, представительский и прикладной – функции размыты – конкретное приложение.

В ЛВС функциональность Канального уровня определяется не протоколом из стека TCP/IP, а стандартными протоколами Канального уровня типа Ethernet или Token Ring. Единственными протоколами из стека TCP/IP, работающими на Канальном уровне, являются ARP и протоколы для установления dial-up соединения (SLIP, PPP).

Уровень Межсетевого взаимодействия представлен протоколом IP (Internet Protocol), выступающим в качестве первичного носителя для всех остальных протоколов, оперирующих на более высоких уровнях. Также на этом уровне присутствует протокол ICMP, осуществляющий диагностику и контроль сообщений об ошибках на IP-уровне. Протокол IP, как основной протокол-носитель, является протоколом без установления соединения и, в связи с этим, ненадежным, поскольку недостающие сервисы обеспечиваются Транспортным уровнем по мере необходимости.

На транспортном уровне функционируют два протокола: Transmission Control Protocol и User Datagram Protocol. Протокол TCP ориентирован на установление соединения и надежен, в то время как UDP представляет собой протокол без установления соединения и, следовательно, не очень устойчив. Приложение может затребовать первый или второй протокол в зависимости от собственных запросов и сервисов, которые ему уже предоставлены.

Протокол TCP зачастую используется приложениями, передающими большие порции данных. Протокол UDP зачастую используется приложениями, оперирующими малым объемом данных и построенных по архитектуре «клиент-сервер», т.к. в данной архитектуре заложен механизм подтверждения доставки данных и она не нуждается в излишних средствах подтверждения нижележащего протокола.

Прикладной уровень наиболее трудно поддается описанию, так как протоколы, работающие здесь, могут быть как полностью завершенными, самодостаточными приложениями, так и всего лишь механизмами, с помощью которых другие приложения обеспечивают какие-либо услуги, как в случае DNS.

1. Адресация в IP-сетях

В стеке протоколов TCP/IP используется три типа адресов:

1. *MAC-адрес* – идентификационный номер сетевого адаптера или порта маршрутизатора, например, 00-60-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными, так как управляются централизовано. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байтов: старшие 3 байта - идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем.
2. *Доменный адрес*. Имеет иерархическую структуру и несет смысловую нагрузку, поскольку предназначен для удобства запоминания пользователем. Такой адрес состоит из нескольких частей, например, имени машины, имени организации, имени домена. Называется также DNS-именем. Например, www.susu.ac.ru.
3. *Сетевой адрес*. Для сетей, поддерживающих стек протоколов TCP/IP, это IP-адрес. IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например: 193.233.81.15 - традиционная десятичная форма представления адреса.

IP-адресация поддерживает двухуровневую иерархию. Адрес делится на две части – номер сети, и номер узла в этой сети. Для разделения двух частей адреса используется маска – двоичное число, которое содержит единицы в разрядах, интерпретируемых как номер сети. Например, маска 255.255.255.0 (11111111. 11111111. 11111111. 00000000) для адреса 193.66.39.214 означает, что в этом адресе первые три байта будут определять номер сети, а остальные – адрес узла. Таким образом, адрес сети – 193.66.39.0. Иногда для записи маски используют следующий формат: 193.66.39.214/24. Такая запись означает, что маска содержит 24 единицы, то есть под адрес сети отведено 24 разряда.

Существует соглашение о специальных адресах. Расшифровка особых адресов приведена в таблице 3.1.

Таблица 3.1 – Специальные IP-адреса

Вид адреса	Пример	Назначение
Все нули	0.0.0.0	Адрес того узла, который сгенерировал пакет
(номер сети).(все нули)	230.154.17.0	Данная IP-сеть
(все нули).(номер узла)	0.0.0.192	Узел в данной IP-сети
(номер сети).(все единицы)	230.154.17.255	Все узлы данной IP-сети. Пакет, имеющий такой адрес рассылается всем узлам сети с заданным номером. Такая рассылка называется широковещательной
Все единицы	255.255.255.255	Все узлы в той же сети, что и пославший пакет. Ограниченная широковещательная рассылка
127.(что угодно)	127.0.0.1	Петля. Адрес узла, пославшего пакет. Используется для тестирования процессов в пределах одной машины

Для установления соответствия между MAC и IP-адресами используется протокол разрешения адреса – Address Resolution Protocol, ARP.

Таблица 3.2 – Пример ARP таблицы

Адрес IP	Физический адрес	Тип
210.30.50.186	00-80-48-a3-fe-72	Динамический
210.30.50.42	00-80-48-eb-65-d6	Статический

Статические записи создаются вручную администратором, а динамические – средствами протокола ARP.

2. Протокол IGMP

2.1. Рассылка групповых сообщений в сети Internet

Рассылка групповых сообщений IP (IP-мультикастинг) предоставляет приложениям два сервиса:

1. Доставка к нескольким пунктам назначения. Существует множество приложений, которые доставляют информацию нескольким адресатам, например, диалоговые конференции, распространение почты или новостей. Если групповая адресация не используется, эти типы сервисов вынуждены использовать TCP (при этом осуществляется доставка отдельной копии на каждый пункт назначения). Даже при существовании групповой формы сообщений, некоторые приложения все-таки используют TCP из-за его надежности.
2. Запрос от клиента к серверу. Например, бездисковая рабочая станция старается определить положение сервера загрузки. В настоящее время это осуществляется с использованием широковещательных запросов (как в случае с BOOTP), однако решение с использованием групповых запросов может уменьшить загруженность хостов, которые не предоставляют этот сервис.

ЭВМ может участвовать в мультикастинг-процессе на одном из следующих уровней:

- 0 – ЭВМ не может ни посылать, ни принимать данные
- 1 – ЭВМ может только посылать пакеты в процессе IP-мултикастинга
- 2 – ЭВМ в режиме мультикастинга может передавать и принимать пакеты

2.2. Групповые адреса

На рисунке 3.2 показан формат IP-адреса класса D.

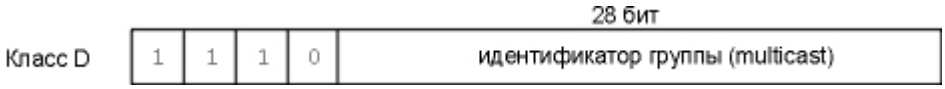


Рисунок 3.2. Формат IP-адреса класса D.

В отличие от трех других классов IP адресов (A,B, C) 28 бит, отведенных под групповой идентификатор, не подвергаются дальнейшему делению. Групповой адрес (multicast group address) состоит из четырех старших бит, установленных в 1110, и идентификатора группы. В десятичном виде групповые адреса находятся в диапазоне от 224.0.0.0 до 239.255.255.255.

Некоторое количество хостов, просматривающих определенный групповой IP адрес, называется группой хостов (host group). Группа хостов может объединять хосты в разных сетях. Членство в группе динамическое: хост может вступать в группу и выходить из группы по собственному желанию. Не существует ограничений на количество хостов в группе, и хост не должен принадлежать к группе, чтобы послать сообщение в эту группу.

Некоторые адреса групп назначаются как заранее известные адреса от IANA (Internet Assigned Numbers Authority) (таблица 3.3). В этом случае группа считается постоянной группой хостов (permanent host group). Заранее известные групповые адреса приведены в последних RFC назначенных номеров (Assigned Numbers RFC). Обратите внимание на то, что постоянным в данном случае является групповой адрес, а не членство в группе.

Таблица 3.3 – Зарегистрированные мултикаст-адреса IANA

Мультикастинг адрес	Описание
224.0.0.0	Зарезервировано
224.0.0.1	Все системы данной субсети
224.0.0.2	Все маршрутизаторы данной субсети
224.0.0.4	Все DVMRP-маршрутизаторы
224.0.0.5-224.0.0.6	OSPFIGP (MOSPF)

224.0.0.9	Маршрутизаторы RIP2
224.0.0.10	IGRP маршрутизаторы
224.0.1.0	VMTP-группа менеджеров
224.0.1.1	NTP-network time protocol - сетевая службы времени
224.0.1.6	NSS - сервер имен
224.0.1.7	Audionews - audio news multicast (аудио служба новостей)
224.0.1.9	MTP (multicast transport protocol) - транспортный протокол мультикастинга
224.0.1.10	IETF-1-low-audio
224.0.1.11	IETF-1-audio
224.0.1.12	IETF-1-video
224.1.0.0-224.1.255.255	ST мультикастинг-группы
224.2.0.0-224.2.255.255	Вызовы при мультимедиа- конференциях
232.0.0.0-232.255.255.255	VMTP переходные группы

IANA владеет блоком Ethernet адресов, которые в шестнадцатеричном представлении выглядят как 00:00:5e. Это старшие 24 бита Ethernet адреса, означающие, что блок включает адреса в диапазоне от 00:00:5e:00:00:00 до 00:00:5e:ff:ff:ff. IANA отвела половину этого блока для групповых адресов. Установлено правило, что первый байт Ethernet адреса равный 01 указывает на групповой адрес. Это означает, что Ethernet адреса, соответствующие групповым адресам IP, должны находиться в диапазоне от 01:00:5e:00:00:00 до 01:00:5e:7f:ff:ff.

Приведенные здесь выражения используют стандартную последовательность битов для Internet, для сетей CSMA/CD или Token bus, а именно такую, как биты располагаются в памяти. Это как раз то, с чем сталкивается большинство программистов и системных администраторов. IEEE документация использует порядок бит, который используется при передаче. Assigned Numbers RFC предоставляет дополнительные подробности о различиях между этими представлениями. Подобное расположение позволяет 23 битам в Ethernet адресе соответствовать идентификатору группы IP. В процессе преобразования адресов 23 младших бита идентификатора группы помещаются в 23 бита Ethernet адреса (рисунок 3.3). Старшие 5 бит в идентификаторе группы игнорируются, так как они не уникальны. Каждому Ethernet адресу соответствует 32 различных идентификатора группы. Например, групповой адрес 224.128.64.32 (в шестнадцатеричном представлении e0.80.40.20) и 224.0.64.32 (в шестнадцатеричном представлении e0.00.40.20) оба будут трансформированы в Ethernet адрес 01:00:5e:00:40:20. Так как подобное сопоставление не уникально, предполагается, что драйвер устройства или IP модуль должен осуществить фильтрацию, так как сетевая плата может получить групповой фрейм, который хосту не предназначен. Если сетевая плата не осуществляет адекватную фильтрацию групповых фреймов, драйвер устройства, вполне возможно, должен будет получать все групповые фреймы и сам осуществлять фильтрацию.

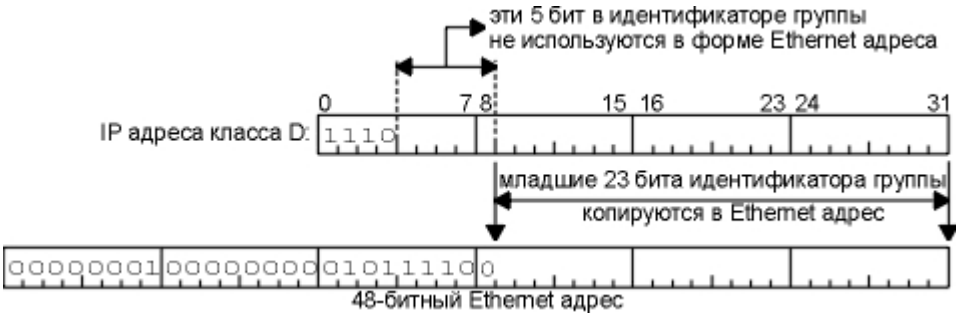


Рисунок 3.3. Соответствие между IP адресами класса D и групповыми адресами Ethernet.

Существует два варианта реализации групповой адресации в сетевых платах, использующиеся в локальных сетях. Одни осуществляют групповую фильтрацию, основанную на значении аппаратного группового адреса, что означает, что некоторые нежелательные фреймы могут пройти. В другом случае имеется небольшое фиксированное количество групповых адресов, принимаемых платой, при этом, если хосту необходимо принять больше групповых адресов, чем поддерживается, интерфейс должен быть помещен в режим "разных групп" (multicast promiscuous). Однако, оба типа интерфейсов все еще требуют, чтобы драйвер устройства осуществлял проверку на предмет того, необходимо ли дальше обрабатывать принятый фрейм. Даже если интерфейс осуществляет идеальную групповую фильтрацию (основанную на 48-битном аппаратном адресе) фильтрация все еще необходима, так как сопоставление IP адресов класса D и 48-битных аппаратных адресов осуществляется не один к одному. Однако, если абстрагироваться от несовершенства преобразования адресов и аппаратной фильтрации, групповая адресация все же лучше, чем широковещательная.

Осуществить групповой запрос в единственную физическую сеть довольно просто. Отправляющий процесс указывает IP адрес назначения, который является групповым адресом, драйвер устройства конвертирует это в соответствующий Ethernet адрес и отправляет. Принимающие процессы должны указать своим IP модулям, что они хочет получать датаграммы, предназначенные определенному групповому адресу, и драйвера устройств должен каким-либо образом получать эти групповые фреймы. Все это называется "вступлением в группу". (Причина, по которой используется выражение "принимающие процессы" во множественном числе, объясняется тем, что обычно существует несколько получателей, которым предназначено групповое сообщение, либо на одном, либо на разных хостах.) Когда групповая датаграмма получена хостом, копия доставляется всем процессам, которые принадлежат к группе. Это отличается от UDP, где единственный процесс получает входящую персональную UDP датаграмму. Несколько процессов на одном хосте могут принадлежать к одной группе.

Однако сложности растут как снежный ком, когда группа распространяется на несколько сетей, и групповые пакеты должны проходить через маршрутизаторы. Маршрутизаторам необходимо знать, принадлежат ли какие-либо хосты в данной физической сети к определенной группе. Для определения этого, существует протокол, называемый протоколом управления группами Internet (IGMP - Internet Group Management Protocol).

2.3. Назначение протокола IGMP

В этой главе мы рассмотрим протокол управления группами Internet, который используется хостами, маршрутизаторами и коммутаторами, для того чтобы поддерживать групповую рассылку сообщений. Он позволяет всем системам физической сети знать, какие хосты в настоящее время объединены в группы и к каким группам они принадлежат. Эта информация необходима для групповых маршрутизаторов, именно так они узнают, какие групповые датаграммы необходимо перенаправлять и на какие интерфейсы. Не все коммутаторы поддерживают протокол IGMP. Функция, реализующая возможность перехвата и анализа IGMP пакетов, называется "igmp snooping". Именно благодаря анализу проходящих IGMP пакетов, коммутатор узнает, к каким портам подключены абоненты и в каких группах они состоят. В дальнейшем коммутаторы и маршрутизатору, поддерживающие протокол IGMP будем называть групповыми маршрутизаторами.

2.4. Идеология и алгоритм работы протокола IGMP

Как и ICMP, IGMP является частью IP уровня. Так же как ICMP, IGMP сообщения передаются в IP-датаграммах. Протокол IGMP имеет сообщение фиксированного размера, без необязательных данных. На рисунке 3.4 показана инкапсуляция IGMP сообщения в IP-датаграмму.

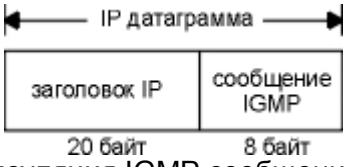


Рисунок 3.4. Инкапсуляция IGMP сообщения в IP датаграмму.

На то, что в IP-датаграмме находится IGMP сообщение, указывает величина в поле протокола равная 2.

Вступление в группу
Фундаментальной концепцией для работы группы является то, что процесс вступает в группу с определенным интерфейсом хоста. Членство в группе динамическое: со временем процесс может как вступать, так и выходить из группы. Естественно, процесс должен иметь возможность вступить в группу с указанным интерфейсом. Процесс также может покинуть группу, в которую он до этого вступил. Для вступления в группу и выхода из группы требуется, чтобы какой-либо API на хосте поддерживал групповую рассылку. Используется выражение "интерфейс", потому что членство в группе связано с интерфейсом. Процесс может вступить в одну и ту же группу с разных интерфейсов. Релиз IP, поддерживающий групповую рассылку в Berkeley Unix Стэндфордского университета, детализирует эти изменения сокетов API. Поэтому идентификатор хоста в группе это адрес группы и интерфейс. Хост должен помнить таблицу всех групп, к которым принадлежит хотя бы один процесс, и счетчик обращений, то есть количество процессов, принадлежащих к группе.

IGMP запросы и отчеты
IGMP сообщения используются групповыми маршрутизаторами, чтобы поддерживать членство в группах для каждой сети, физически подключенной к маршрутизатору. Существуют следующие правила.

1. Хост отправляет первый IGMP отчет, когда первый процесс вступает в группу. Если несколько процессов на данном хосте вступили в одну и ту же группу, отправляется только один отчет, в тот момент, когда процесс первый раз вступил в группу. Отчет посылается на тот же интерфейс, с которым процесс вступил в группу.
2. Хост не посылает отчет, когда процесс выходит из группы, даже когда последний процесс вышел из группы. Хост знает, что в этой группе больше нет членов, поэтому когда он получает следующий запрос (следующий шаг), он не отправляет отчет.
3. Групповой маршрутизатор отправляет IGMP запрос с регулярными интервалами, чтобы выяснить, принадлежат ли процессы каких-либо хостов к каким-либо группам. Маршрутизатор посылает один запрос на каждый интерфейс. Групповой адрес в запросе установлен в 0, так как маршрутизатор ожидает приход одного отклика от хоста для каждой группы, к которой от хоста принадлежит один или несколько членов.
4. Хост отвечает на IGMP запрос посылкой одного IGMP отчета для каждой группы, которая содержит хотя бы один процесс.

С использованием этих запросов и отчетов групповой маршрутизатор поддерживает таблицу, содержащую информацию о том, на котором из его интерфейсов имеется один или несколько хостов в группе. Когда маршрутизатор получает групповую датаграмму, которую необходимо перенаправить, он перенаправляет ее (с использованием соответствующего группового адреса канального уровня) только на тот интерфейс, на котором до сих пор есть хосты, процессы которых принадлежат к этой группе.

На рисунке 3.5 показаны два типа IGMP сообщений, отчеты, отправленные хостом, и запросы, отправленные маршрутизатором. Маршрутизатор опрашивает каждый хост, чтобы тот идентифицировал каждую группу для данного интерфейса.

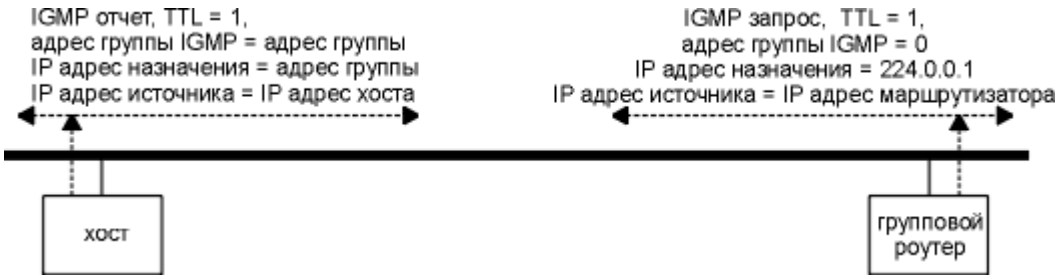


Рисунок 3.5. IGMP отчеты и запросы.

Детали реализации

В протоколе существуют некоторые аспекты, которые улучшают его производительность. Во-первых, когда хост посылает исходный IGMP отчет (когда первый процесс вступил в группу), не существует гарантии, что этот отчет будет доставлен (так как используется средство доставки IP). Позже отправляется еще один отчет. Время, когда будет отправлен следующий отчет, выбирается хостом случайным образом, причем значение времени находится в диапазоне от 0 до 10 секунд.

Когда хост получает запрос от маршрутизатора, он не отвечает сразу же, а откладывает ответы на более позднее время. (Мы используем множественное число "ответы", потому что хост должен послать один отчет для каждой группы, которая содержит одного или несколько членов.) Так как несколько хостов могут отправить отчет для одной и той же группы, каждый отправляет свой отчет с задержкой, выбранной случайным образом. Также обратите внимание на то, что все хосты подключенные к одной физической сети получают все отчеты от других хостов, находящихся в той же группе, потому что адрес назначения отчета (рисунок 3.5) – это групповой адрес. А это, в свою очередь, означает, что если хост отложил момент отправки отчета, однако получил копию того же самого отчета от другого хоста, ответ может быть отменен. Это объясняется тем, что групповому маршрутизатору нет необходимости знать, сколько хостов принадлежит к группе - ему достаточно знать, что по крайней мере один хост принадлежит к группе.
На одном физическом кабеле без групповых маршрутизаторов, трафик, принадлежащий IGMP, – это отчеты, которые отправляются хостами, поддерживающими групповую адресацию IP, когда они вступают в новую группу.

Поле времени жизни

На рисунке 3.5 видно, что поле TTL в отчете и запросе установлено в 1. Это напоминает обычное TTL поле в IP заголовке. Групповая датаграмма с TTL исходно равным 0 не "уйдет" дальше своего хоста. По умолчанию групповые датаграммы рассылаются с TTL равным 1. Это позволяет датаграммам распространяться только в своей подсети. Значение TTL больше единицы может быть установлено групповым маршрутизатором. ICMP ошибка никогда не генерируется в ответ на датаграмму, направляемую на групповой адрес. Групповые маршрутизаторы не генерируют ICMP ошибку "время истекло" (time exceeded), когда значение TTL становится равным 0.

Обычно, пользовательский процесс не заботится о значении исходящего TTL. Одно исключение, пожалуй, программа Traceroute, принцип работы которой основан как раз на изменении значения поля TTL. Однако, приложения, которые работают с групповой адресацией, должны иметь возможность установить исходящее поле TTL. Это означает, что программный интерфейс должен предоставлять эту возможность пользовательским процессам.

Путем увеличения TTL приложение может осуществить расширенный поиск (expanding ring search) конкретного сервера. В этом случае первая групповая датаграмма посылается с TTL равным 1, если ответ не получен, посылается датаграмма с TTL равным 2, затем 3 и так далее. В этом случае приложение определяет положение ближайшего сервера в количествах пересылок. Специальный диапазон адресов 224.0.0.0 – 224.0.0.255 отводится для приложений, которые не будут рассылать групповые запросы дальше чем на одну

пересылку. Групповые маршрутизаторы не должны перенаправлять датаграммы с такими адресами назначения, вне зависимости от TTL.

Группа всех хостов (All-Hosts)
На рисунке 3.5 мы видели, что IGMP запрос от маршрутизатора отправляется на IP адрес назначения 224.0.0.1. Этот адрес называется адресом группы всех хостов (all-hosts). Он имеет отношение ко всем хостам и маршрутизаторам подключенным к физической сети и поддерживающим групповую адресацию. Каждый хост автоматически вступает в эту группу со всеми интерфейсами, которые поддерживают групповую адресацию, при инициализации интерфейса. О членстве в этой группе никогда не сообщается (рассылкой отчетов).

Краткие выводы
Групповая рассылка это способ разослать сообщения нескольким получателям. Для многих приложений это вариант предпочтительней, нежели рассылка широковещательных запросов, так как с помощью групповой рассылки можно уменьшить загруженность хостов, которые не заинтересованы в принятии сообщений. Простой протокол, определяющий принадлежность хостов к группе (IGMP),– это краеугольный камень, на котором строится групповая адресация.

Так как рассылка широковещательных запросов обычно ограничивается одной локальной сетью, групповые запросы могут быть использованы вместо широковещательных для различных предложений, которые в настоящее время используют широковещательные запросы. Однако, для глобальных сетей проблема групповых запросов до сих пор до конца не решена.

2.5. Формат IGMP-сообщений
В настоящий момент существует три версии этого протокола: IGMP v1 (RFC 1112), IGMP v2 (RFC-2236), IGMP v3. Наиболее распространена версия 2.

Формат IGMPv2 пакета приведен ниже (рисунок 3.6).

<table>
<tr><td colspan="8">0 1 2 3 4 5 6 7</td><td colspan="8">8 9 10 11 12 13 14 15</td><td colspan="16">16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="8">Тип сообщения</td><td colspan="8">Максимальное время ответа</td><td colspan="16">Контрольная сумма</td></tr>
<tr><td colspan="32">Широковещательный адрес группы</td></tr>
</table>

Рисунок 3.6. Формат пакета протокола IGMP версии 2.

В IGMPv2 существуют следующие типы сообщений (таблица 3.4).

Таблица 3.4 – Возможные значения поля Type

TYPE	Источник	Тип сообщения	Назначение сообщения
0x11	Маршрутизатор	Membership Query	Запрос о составе группы
0x16	Узел	Membership Report (IGMPv2)	Отчет о составе группы
0x17	Узел	Leave Group	Сообщение о выходе из группы

Рассмотрим назначение этих сообщений более детально. При подключении к IGMP-группе абонентское устройство посылает в сеть IGMP пакет с типом "Отчет о составе группы", тем самым давая понять, что желает получать пакеты для данной группы. При выходе из группы абонентское устройство посылает в сеть IGMP пакет с типом "Сообщение о выходе из группы". Маршрутизатор периодически посылает в сеть IGMP запрос с типом "Запрос о составе группы" для выяснения состава группы на текущий момент времени. Если ответа от абонентских устройств не последовало, то маршрутизатор отключает эту группу и более не пересылает пакеты для данной группы.

Содержимое поля «Максимальное время ответа» имеет смысл только для сообщений Membership Query. В это поле групповой маршрутизатор, использующий IGMPv2, поме-

щает максимальное время задержки ожидания ответа на сообщение, выраженное в десятых долях секунды. По умолчанию значение поля считается равным 100 (10 сек). В остальных сообщениях это поле не анализируется.

Контрольная сумма (checksum) рассчитывается так же, как контрольная сумма ICMP.

Групповой адрес (group address) это IP адрес класса D. В запросе поле группового адреса устанавливается в 0. В отчете оно содержит групповой адрес.

3. IP-маршрутизация

Основная задача любой сети – транспортировка информации от ЭВМ-отправителя к ЭВМ-получателю. В большинстве случаев для этого нужно совершить несколько пересылок. Проблему выбора пути решают алгоритмы маршрутизации. Алгоритмы для своей работы используют таблицу маршрутизации. В зависимости от того, каким образом заполняется и корректируется таблица – администратором или автоматически, – маршрутизация делится на статическую и динамическую (рисунок 3.7). При динамической маршрутизации к системе добавляется еще один компонент – протокол маршрутизации, который собственно и отвечает за заполнение и корректировку таблицы маршрутизации. Но модификация таблицы происходит не напрямую. Протокол поддерживает собственную базу данных, которая может содержать следующую информацию по маршрутам:

- загруженность сетевого интерфейса;
- задержка в подключенных каналах передачи данных;
- стоимость передачи данных по каналам;
- прочее.

Данная информация формируется на основе двух источников:

- измерение и оценка параметров сети самим протоколом;
- взаимодействие с протоколами, работающими на других маршрутизаторах.

На основе содержимого базы данных протокола формируется таблица маршрутизации.

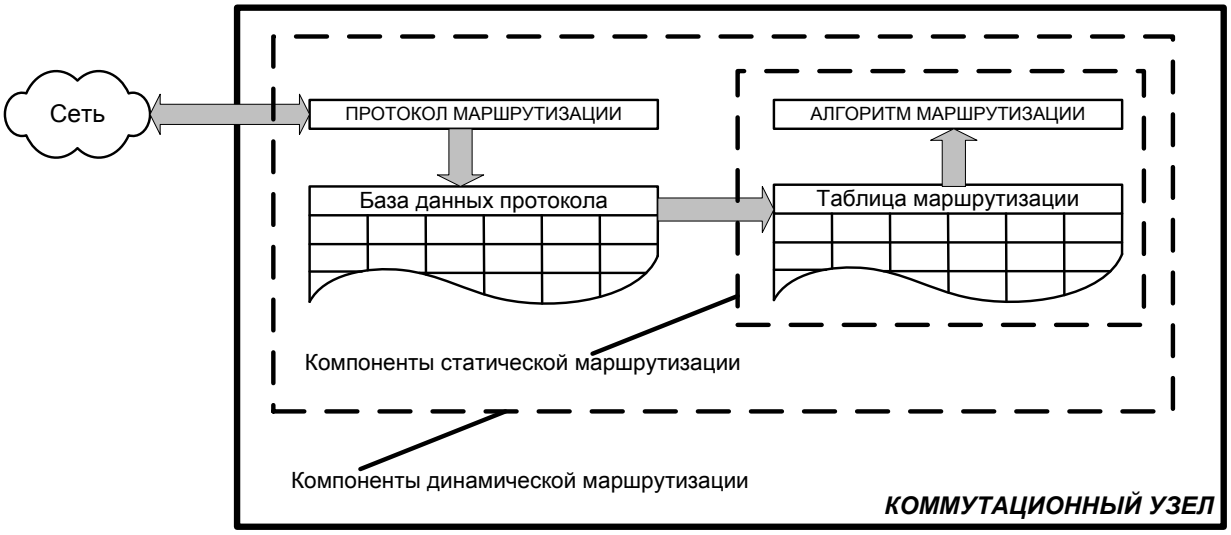


Рисунок 3.7. Компоненты системы маршрутизации.

Таким образом, можно четко выделить функции протокола и алгоритма маршрутизации. Если у алгоритма всего лишь одна задача – выбор следующего маршрута для обрабатываемого пакета, то протокол маршрутизации включает в себя следующие процедуры:

- измерение и оценка параметров сети;
- взаимодействие с протоколами на других узлах сети путем рассылки и принятия служебной информации;
- формирование собственной базы данных;
- модификация таблицы маршрутизации.

То есть, в системе маршрутизации алгоритм является пассивным компонентом, который для выполнения своей функции использует информацию, предоставляемую либо администратором, либо протоколом – активными управляющими компонентами. Данное различие хорошо подчеркивается терминами, которыми обозначаются сетевые протоколы, реализующие алгоритм и протокол маршрутизации. Термин «Routed protocol» обычно применяется для обозначения протокола, в рамках которого реализован алгоритм маршрутизации, например, протокол IP. Термин «Routing protocol» обычно применяется для обозначения протокола маршрутизации.

2.1. Общая концепция IP-маршрутизации

Основным документом, описывающим алгоритм IP-маршрутизации, является RFC-1812. Согласно ему не выработано единого стандарта на какой-либо из существующих алгоритмов IP-маршрутизации. Все они являются так называемым Internet Folklore. Тем не менее, документ RFC-1812 выделяет основополагающий алгоритм, так называемый Classic Algorithm, которые должны использовать IP-маршрутизаторы. Суть его работы следующая.

Канальный уровень предоставляет сетевому уровню пакет (IP-датаграмму) и дополнительную информацию о нем. IP-протокол считывает IP-заголовок из датаграммы и начинает его поэтапную обработку. Основные шаги следующие:

1. Проверка корректности датаграммы (Packet Validation). На данном этапе проверяется общая длина пакета, длина заголовка, версия IP-протокола и контрольная сумма.
2. Обработка IP-опций.
3. Определение места назначения пакета. В первую очередь проверяется, предназначен ли пакет локальной системе или нет. Если нет, то маршрутизатор должен определить маршрут, по которому следует переслать пакет дальше, – за это отвечает алгоритм определения следующего хопа (hop). Если подходящего маршрута не обнаруживается, то пакет подлежит сбросу.
4. Заключительный этап имеет место быть, если же все-таки маршрутизатор определил, куда направить пакет. На данном шаге выполняется коррекция IP-заголовка (декрементация поля TTL, обработка оставшихся IP-опций), фрагментация датаграммы (если необходимо), маппирование адреса канального уровня (вызов протокола ARP) и передача датаграммы на канальный уровень.

Нас интересует алгоритм определения следующего хопа, так как именно он отвечает за выбор сетевого интерфейса, на который будет послан пакет. При определении адреса следующего узла (хопа) применяется алгоритм просмотра маршрутов (route lookup algorithm). Данный алгоритм работает с множеством маршрутов-кандидатов. Обозначим это множество как $\{R_K\}$. Изначально множество $\{R_K\}$ заполняется содержимым таблицы маршрутизации. Цель алгоритма – выбрать из данного множества наилучший маршрут путем отброса остальных. Выбор происходит за ряд шагов, которые называются «правилами отсечения» (pruning rules). Должны обязательно применяться следующие правила в строго приведенном порядке:

1. *Basic Match.*
   Каждый маршрут имеет следующий вид:
   route.dest/route.length
   route.dest – атрибут назначения;
   route.length – префикс длины (или маска подсети);
   ip.dest – IP-адрес назначения в заголовке пакета.
   Данное правило оставляет только те маршруты, для которых наложение маски подсети на атрибут назначения дает IP-адрес назначения, то есть:
   route.dest & route.length = ip.dest
   В результате работы правила получаем множество $\{R_{K2}\}$.
2. *Longest Match*.
   Работает с множеством $\{R_{K2}\}$. В множестве остаются маршруты, которые имеют самую длинную маску подсети. Остальные маршруты отбрасываются. В результате работы правила получаем множество $\{R_{K3}\}$.
3. *Weak TOS (опционально).*
   Данное правило выполняется только в том случае, если маршрутизатор учитывает значение поля TOS в IP-заголовке при выборе маршрута.
   Работает с множеством $\{R_{K3}\}$. Каждая запись о маршруте содержит поле «tos». Из множества маршрутов $\{R_{K3}\}$ выбираются те, у которых величина «tos» совпадает с величиной «tos», взятой из IP-заголовка пакета. Если таковых маршрутов не нахо-

дится, то из маршрутов выбираются те, у которых величина «tos» = 0000 (значение по умолчанию). В результате работы правила получаем множество $\{R_{K4}\}$.

4. *Best Metric.*
   Работает с множеством $\{R_{K4}\}$. Каждая запись о маршруте содержит поле метрики «metric». Из множества маршрутов выбираются те, у которых значение метрики более приоритетно. Вопрос приоритета – отдельный вопрос. В результате работы правила получаем множество $\{R_{K5}\}$.
5. *Vendor Policy.*
   Здесь могут применяться правила, заданные разработчиком программного продукта. Работает с множеством $\{R_{K5}\}$. В результате работы правила получаем конечное множество {R}.

Считается, что алгоритм отработал нормально, если конечное множестве {R} содержит один маршрут. Если множество {R} оказалось пустым, то пакет отбрасывается. Также возможен вариант, когда по окончанию работы алгоритма, множество {R} содержит более одного маршрута. Тогда алгоритм в классическом варианте алгоритма выбирается любой один маршрут. Следует также отметить, что на практике зачастую используются правила «Basic Match» → «Longest Match» → «Best Metric».

Помимо приведенного классического алгоритма (Classic Algorithm) существует ряд других, широко используемых в Интернете. Отличаются они только количеством и порядком применения правил отсечений и ориентированны в основном на поддержку работы протоколов маршрутизации, таких как OSPF, BGP и т.д.

2.2. Основные термины IP-маршрутизации

Автономные системы

Автономной системой называют такую локальную сеть или систему сетей, которые имеют единую администрацию и общую маршрутную политику. Концепция автономных систем предполагает разбиение сети на отдельные области, в пределах которых управление процессами определения маршрута производится автономно. На (рисунок 3.8) приведена структурная схема организации информационного взаимодействия в соответствии с концепцией автономных систем.

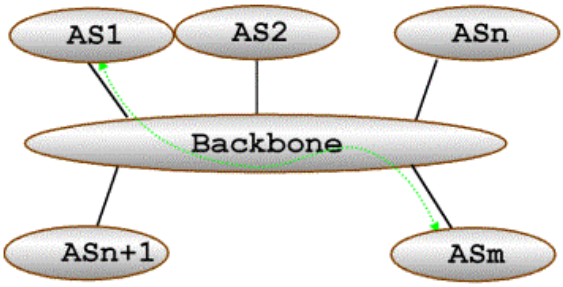


Рисунок 3.8. Структурная схема информационного взаимодействия на основе автономных систем.

Взаимодействие между локальными сетями в пределах автономной системы AS_N производится в соответствии с правилами определения маршрута, которые приняты в данной автономной системе. Внутри каждой автономной системы могут быть использованы различные правила определения маршрута и протоколы маршрутизации. Информационное взаимодействие между компонентами различных автономных систем может быть выполнено только через специальную область, которая предназначена для интеграции всей системы в целом. Такая область, в соответствии с терминологией, которая принята в данной концепции, называется Backbone Area.

Процесс определения маршрута в информационных сетях, которые построены в соответствии с данной концепции, состоит из двух процессов:

- определение маршрута внутри автономной системы
- определение маршрута между автономными системами

Для обеспечения этого взаимодействия коммуникационные узлы автономной системы должны предоставлять информацию о состоянии маршрутов к сетям данной автономной

системы. Кроме того, коммуникационные узлы должны обеспечивать предоставление информации о маршрутах, по которым должны быть переданы данные для достижения компонентов, расположенных в других автономных системах. В том случае, когда автономная система использует несколько физических каналов для организации информационного обмена с остальным миром, такая задача является особенно актуальной.
В зависимости от своего расположения в общей структуре Интернет автономная система может быть тупиковой (stub) – имеющей связь только с одной АС, или многопортовой (multihomed), т.е. имеющей связи с несколькими АС. Если административная политика автономной системы позволяет передавать через свои сети транзитный трафик других АС, то такую автономную систему можно назвать транзитной (transit).
Внедрение идеологии автономных систем сделали возможным существенно облегчить процедуру маршрутизации, сократить требуемое число IP-адресов и создать гибкую и эффективную схему описания маршрутной политики.

Метрики, используемые алгоритмами маршрутизации
Когда алгоритм маршрутизации обновляет таблицу маршрутизации, его главной целью является определение наилучшей информации для включения в таблицу. Каждый алгоритм маршрутизации интерпретирует понятие “наилучшая” по-своему. Для каждого пути в сети алгоритм генерирует число, называемое метрикой. Определение оптимальности путей при формировании и обновлении таблицы маршрутизации может производиться в соответствии с такими наиболее употребительными метриками или их комбинациями, как:

1. Длина маршрута является наиболее общим показателем маршрутизации. Единица измерения длины маршрута в числе промежуточных узлов, включая также узел назначения – это количество хопов (от англ. hop – “скачок”). Длина маршрута от себя до себя равна 0, от себя до другого узла в этой же сети - 1 хоп. Как правило, чем меньше величина скачков, тем лучше путь.
2. Пропускная способность канала связи. Единицами измерения пропускной способности при различных каналах связи могут быть килобиты, мегабиты и гигабиты в секунду. Как правило, чем больше пропускная способность, тем предпочтительней маршрут, хотя не всегда большая полоса пропускания, обязательно будет лучше маршрутов, проходящих через менее быстродействующие каналы.
3. Задержка - прогнозируемое суммарное время пересылки пакета от отправителя получателю. Задержка зависит от многих факторов, включая полосу пропускания промежуточных каналов сети, загруженность очередей маршрутизаторов на пути продвижения пакета, физическое расстояние, на которое необходимо переместить пакет, и прочее. Так как здесь имеет место совокупность нескольких важных показателей, задержка является наиболее общим и полезным показателем.
4. Стоимость канала связи – произвольное значение, обычно основанное на величине полосы пропускания, денежной стоимости или результате других измерений, которые назначаются сетевым администратором.
5. Загрузка – объем действий, выполняемый сетевым ресурсом, например маршрутизатором или каналом:
   - процент пакетов, сброшенных из-за нехватки памяти в буферах;
   - средняя длина очередей в системе;
   - число пакетов, для которых наступил тайм-аут и для которых были сделаны повторные передачи;
   - средняя задержка пакета при доставке и среднее отклонение задержки при доставке пакета.
6. Надежность – темп возникновения ошибок в каждом сетевом канале. Некоторые каналы сети могут отказывать чаще, чем другие. Отказы одних каналов сети могут быть устранены легче или быстрее, чем отказы других каналов. При назначении оценок надежности могут быть приняты в расчет любые факторы надежности. Оценки надежности обычно назначаются администраторами сети.

2.3. Свойства алгоритма маршрутизации

Алгоритм маршрутизации должен обладать рядом свойств:

1. Корректность.
2. Оптимальность.
3. Простота и низкие непроизводительные затраты.
4. Живучесть и стабильность.
5. Справедливость.
6. Быстрая сходимость.
7. Гибкость.

Если корректность комментариев не требуют, то остальные свойства надо разъяснить.

Оптимальность

Оптимальность, вероятно, является самой общей целью разработки. Она характеризует способность алгоритма маршрутизации выбирать "наилучший" маршрут. Наилучший маршрут зависит от показателей и от "веса" этих показателей, используемых при проведении расчета. Сформулируем принцип оптимальности. Этот принцип утверждает, что если маршрутизатор J находится на оптимальном пути между маршрутизаторами I и K, то оптимальный маршрут между J и K принадлежит этому оптимальному пути. Это так, поскольку если существование между J и K оптимального маршрута отличного от части маршрута между I и K противоречил бы утверждению об оптимальности маршрута между I и K.

Следствием из принципа оптимальности является утверждение, что все маршруты к заданной точке сети образуют дерево с корнем в этой точке. Это дерево называется деревом погружения и его иллюстрирует (рисунок 3.9).

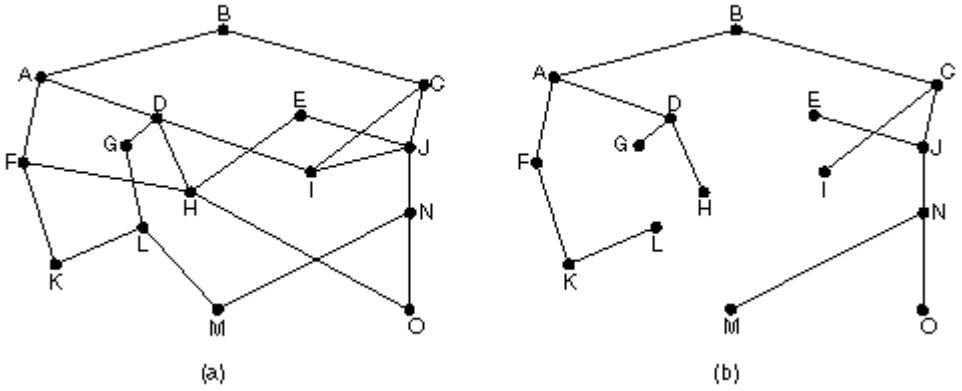


Рисунок 3.9. (a) Подсеть. (b) Дерево погружения для маршрутизатора B.

Поскольку дерево погружения – это дерево, то там нет циклов так, что каждый пакет будет доставлен за конечное число скачков. На практике все может оказаться сложнее. Маршрутизаторы могут выходить из строя и наоборот появляться новые, каналы могут выходить из строя, разные маршрутизаторы могут узнавать об этих изменениях в разное время и т.д. и т.п.

Простота и низкие непроизводительные затраты

Алгоритмы маршрутизации разрабатываются как можно более простыми. Другими словами, алгоритм маршрутизации должен эффективно обеспечивать свои функциональные возможности, с минимальными затратами программного обеспечения и коэффициентом использования. Особенно важна эффективность в том случае, когда программа, реализующая алгоритм маршрутизации, должна работать в компьютере с ограниченными физическими ресурсами.

Живучесть и стабильность

Алгоритмы маршрутизации должны обладать живучестью. Другими словами, они должны четко функционировать в случае неординарных или непредвиденных обстоятельств, таких как отказы аппаратуры, условия высокой нагрузки и некорректные реализации. Т.к.

маршрутизаторы расположены в узловых точках сети, то их отказ может вызвать значительные проблемы.
Часто наилучшими алгоритмами маршрутизации оказываются те, которые выдержали испытание временем и доказали свою надежность в различных условиях работы сети.

Справедливость
Справедливость значит, что все пакеты будут обслуживаться равномерно, ни какому направлению не будет отдаваться предпочтение, для всех абонентов будет всегда выбираться оптимальный маршрут. Надо отметить, что справедливость и оптимальность часто могут вступать в противоречие друг с другом. На рисунке 3.10 приведен пример такого противоречия.

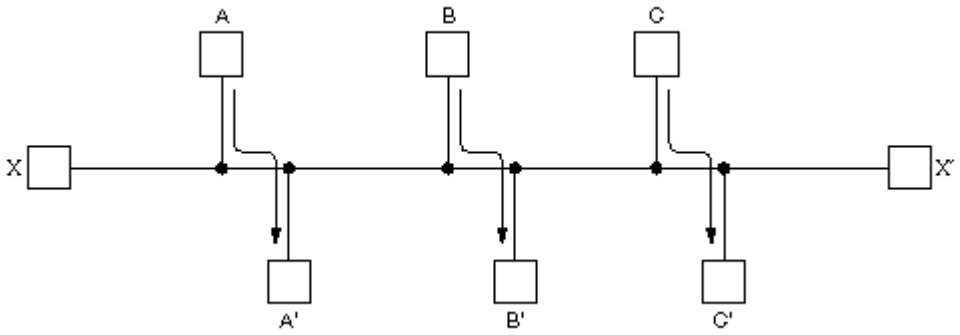


Рисунок 3.10. Конфликт между справедливостью и оптимальностью.

Трафики между A-A”, B-B”, C-C” могут уже забить канал между X-X”. Поэтому вместо кратчайшего маршрута между X и X” надо будет выбирать какой-то другой маршрут.

Быстрая сходимость
Алгоритмы маршрутизации должны быстро сходиться. Сходимость – это процесс соглашения между всеми маршрутизаторами по оптимальным маршрутам. Когда какое-нибудь событие в сети приводит к тому, что маршруты или отвергаются, или становятся доступными, маршрутизаторы рассылают сообщения об обновлении маршрутизации. Сообщения об обновлении маршрутизации пронизывают сети, стимулируя пересчет оптимальных маршрутов и, в конечном итоге, вынуждая все маршрутизаторы придти к соглашению по этим маршрутам. Алгоритмы маршрутизации, которые сходятся медленно, могут привести к образованию петель маршрутизации или выходам из строя сети.

Гибкость
Алгоритмы маршрутизации должны быть также гибкими. Другими словами, алгоритмы маршрутизации должны быстро и точно адаптироваться к разнообразным обстоятельствам в сети. Например, предположим, что сегмент сети отвергнут. Многие алгоритмы маршрутизации, после того как они узнают об этой проблеме, быстро выбирают следующий наилучший путь для всех маршрутов, которые обычно используют этот сегмент. Алгоритмы маршрутизации могут быть запрограммированы таким образом, чтобы они могли адаптироваться к изменениям полосы пропускания сети, размеров очереди к маршрутизатору, величины задержки сети и других переменных.

2.4. Классификация алгоритмов маршрутизации
Все алгоритмы маршрутизации можно классифицировать как:
1. Статическими или динамическими.
2. Одномаршрутными или многомаршрутными.
3. Одноуровневыми или иерархическими.
4. С интеллектом в главной вычислительной машине или в маршрутизаторе.
5. Внутридоменными и междоменными.
6. Алгоритмами состояния канала или вектора расстояний.

Статические или динамические алгоритмы

Статические алгоритмы представляют свод правил работы со статическими таблицами маршрутизации, которые настраиваются администраторами сети. Хорошо работают в случае предсказуемого трафика в сетях стабильной конфигурации. Статическая маршрутизация уменьшает количество передаваемой служебной информации, поскольку в этом случае не посылается информация об изменениях в маршрутном расписании. К существенным недостаткам статической маршрутизации можно отнести то, что, так как статические системы маршрутизации не могут реагировать на изменения в сети, они, как правило, считаются непригодными для современных крупных, постоянно изменяющихся сетей.

Динамическая маршрутизация выполняется несколько иначе. После того, как сетевой администратор введет конфигурационные команды для начала динамической маршрутизации, маршрутная обстановка изменяется автоматически при каждом получении из сети информации об изменениях в ее топологии. Динамические алгоритмы маршрутизации подстраиваются к изменяющимся обстоятельствам сети в масштабе реального времени. Они выполняют это путем анализа поступающих сообщений об обновлении маршрутизации. Если в сообщении указывается, что имело место изменение сети, программы маршрутизации пересчитывают маршруты и рассылают новые сообщения о корректировке маршрутизации. Такие сообщения пронизывают сеть, стимулируя маршрутизаторы заново прогонять свои алгоритмы и соответствующим образом изменять таблицы маршрутизации. Динамические алгоритмы маршрутизации могут дополнять статические маршруты там, где это уместно.

Эффективность динамической маршрутизации зависит от выполнения маршрутизатором двух своих основных функций: поддержка таблицы маршрутизации и своевременное распределение информации о топологии сети. В процессе обмена информацией о топологии сети динамическая маршрутизация опирается на протокол маршрутизации, который представляет собой набор правил, используемых маршрутизатором при обмене информацией с соседними маршрутизаторами.

Немаловажный недостаток динамической маршрутизации – она имеет тенденцию к полной открытости всей информации о сети.

Адаптивные алгоритмы различаются тем, где и как они получают информацию (локально от соседних маршрутизаторов или глобально ото всех), когда они меняют маршрут (каждые Т секунд, когда меняется нагрузка, когда меняется топология), какая метрика используется при оптимизации (расстояние, число скачков, ожидаемое время передачи).

Одномаршрутные или многомаршрутные алгоритмы

Некоторые сложные протоколы маршрутизации обеспечивают множество маршрутов к одному и тому же пункту назначения. Такие многомаршрутные алгоритмы делают возможной мультиплексную передачу трафика по многочисленным линиям. Другими словами данные алгоритмы осуществляют балансировку загрузки сети. Одномаршрутные алгоритмы не могут делать этого. Преимущества многомаршрутных алгоритмов очевидны – они могут обеспечить значительно большую пропускную способность и надежность.

Одноуровневые или иерархические алгоритмы

Некоторые алгоритмы маршрутизации оперируют в плоском пространстве, в то время как другие используют иерархии маршрутизации. В одноуровневой системе все маршрутизаторы равны по отношению друг к другу. В иерархической системе некоторые маршрутизаторы формируют то, что составляет основу (backbone - базу) маршрутизации. Пакеты из внебазовых маршрутизаторов перемещаются к базовым маршрутизаторам и пропускаются через них до тех пор, пока не достигнут общей области пункта назначения. Начиная с этого момента, они перемещаются от последнего базового маршрутизатора через один или несколько внебазовых маршрутизаторов до конечного пункта назначения.

Системы маршрутизации часто устанавливают логические группы узлов, называемых доменами, или автономными системами (Autonomous Systems), или областями. В иерархических системах одни маршрутизаторы какого-либо домена могут сообщаться с маршрутизаторами других доменов, в то время как другие маршрутизаторы этого домена могут поддерживать связь только в пределах своего домена. В очень крупных сетях могут существовать дополнительные иерархические уровни. Маршрутизаторы наивысшего иерархического уровня образуют базу маршрутизации.

Основным преимуществом иерархической маршрутизации является то, что она имитирует организацию большинства компаний и следовательно, очень хорошо поддерживает их схемы трафика. Большая часть сетевой связи имеет место в пределах групп небольших компаний (доменов). Внутридоменным маршрутизаторам необходимо знать только о других маршрутизаторах в пределах своего домена, поэтому их алгоритмы маршрутизации могут быть упрощенными. Соответственно может быть уменьшен и трафик обновления маршрутизации, зависящий от используемого алгоритма маршрутизации.

Внутридоменные или междоменные алгоритмы

Некоторые алгоритмы маршрутизации действуют только в пределах доменов (автономных систем). Такие алгоритмы называются внутридоменными и относятся к классу interior gateway protocol – IGP. Другие алгоритмы маршрутизации используются для определения маршрута за пределами домена (автономной системы). Данные алгоритмы называются междоменными и относятся к классу exterior gateway protocol – EGP. Природа этих двух типов алгоритмов различная. Поэтому понятно, что оптимальный алгоритм внутридоменной маршрутизации не обязательно будет оптимальным алгоритмом междоменной маршрутизации.

Алгоритмы с маршрутизацией от источника или в роутере

В системах маршрутизации от источника роутеры действуют просто как устройства хранения и пересылки пакета, без всяких раздумий отсылая его к следующей остановке, они предполагают, что отправитель рассчитывает и определяет весь маршрут сам. Другие алгоритмы предполагают, что хост отправителя ничего не знает о маршрутах. При использовании такого рода алгоритмов маршрутизаторы определяют маршрут через сеть, базируясь на своих собственных расчетах.

Алгоритмы (протоколы) вектора расстояния или состояния канала

В зависимости от способа, который используется для обеспечения обмена информацией о маршрутах в сети между узлами, различают два типа протоколов маршрутизации:

- Протоколы вектора расстояния (Distance Vector).
- Протоколы состояния канала (Link State).
- Гибридный подход, объединяющий аспекты алгоритмов с определением вектора расстояния и оценки состояния канала.

4. Алгоритмы маршрутизации

4.1. Алгоритмы вектора расстояния

Алгоритмы вектора расстояния построены на основе алгоритмов Бэлмана-Форда и Форда-Фолкерсона. В основе их лежит идея, что у каждого маршрутизатора в подсети есть:

- таблица расстояний до каждого маршрутизатора в подсети. Каждый элемент таблицы состоит из двух полей: первое – номер линии, по которой надо отправлять пакеты, чтобы достичь нужного места, второе – величина задержки до места назначения. Эта величина задержки может быть измерена в разных единицах: скачках, миллисекундах, длине очереди на линии и т.д.
- Кроме этого, каждый маршрутизатор в подсети постоянно замеряет задержки до своих соседей.

Каждые T секунд маршрутизатор шлет своим соседям свой вектор задержек до всех маршрутизаторов в подсети. В свою очередь он получает такие же вектора от своих соседей. Поэтому, имея вектора расстояний от соседей и зная расстояние до соседей, маршрутизатор всегда может вычислить кратчайший маршрут.

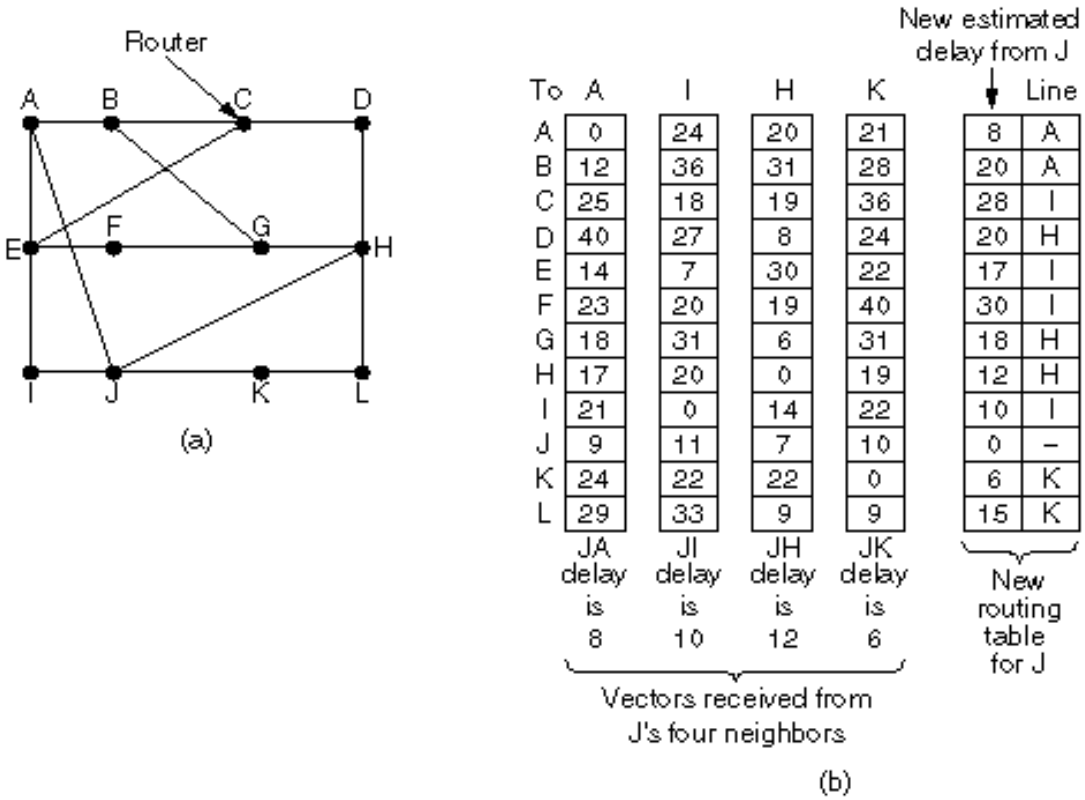


Рисунок 3.11. Работа алгоритма вектора расстояния.

Рассмотрим пример на рисунке 3.11. На рисунке 3.11 (a) показана подсеть. На рисунке 3.11 (b) показаны вектора, которые J маршрутизатор получил от своих соседей и его замеры задержек до соседей. Там же показана итоговая таблица маршрутизации, которую J маршрутизатор вычислит на основании этой информации.

Таким образом, алгоритмы вектора расстояния характеризуются следующим:

- требуют от каждого маршрутизатора посылки всей или части своей маршрутной таблицы, но только своим соседям;
- предусматривают передачу информации о маршрутах периодически через установленные интервалы времени.

Опишем проблемы, характерные для всех протоколов, базирующихся на векторе расстояния, и способы их решения:

1. *Проблема бесконечного счетчика.*
   Алгоритм маршрутизации по вектору расстояния теоретически работает хорошо, но у него есть один недостаток – он очень медленно сходится к правильному значению. Информация о появлении хорошего маршрута в подсети распространяется более или

менее быстро, а вот данные о потере, разрушении какого-либо маршрута распространяются не столь быстро. Рассмотрим пример на рисунке 3.12. Дана линейная подсеть. Пусть изначально маршрутизатор A не работал. Поэтому у всех маршрутизаторов в подсети для него стояла бесконечность. Пусть в какой-то момент времени A был включен. По истечении определенного времени маршрутизаторы начнут обмениваться векторами и B узнает об A. Еще через один обмен векторами об A узнает C и т.д.

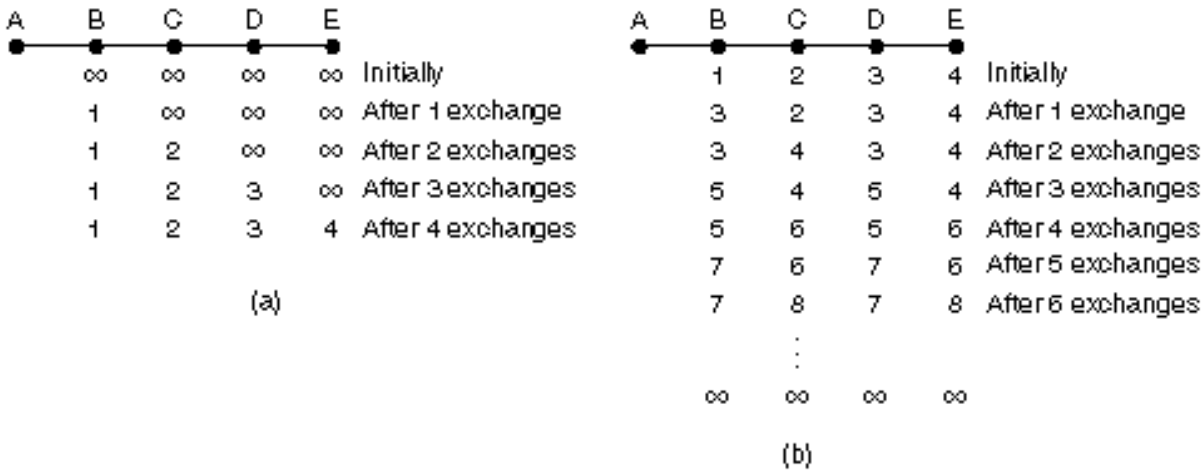


Рисунок 3.12. Проблема бесконечного счетчика.

Таким образом, информация о новом маршруте будет распространяться линейно шаг за шагом за каждый обмен векторами. Если самый длинный маршрут в подсети имеет длину N, то потребуется N обменов векторами, пока информация о новом маршруте дойдет до самого удаленного узла в подсети.
Теперь рассмотрим обратную ситуацию на рисунке 3.12 (b). Здесь изначально все маршрутизаторы и линии были работоспособны. Пусть в какой-то момент времени линия между A и B разрушена. B перестает видеть A, но C говорит B: «Не беспокойся у меня есть маршрут до A». При этом B не подозревает, что маршрут от C до A идет через него же. При этом маршрутизаторы D и E своих таблиц не меняют. Их расстояния до A на единицу больше, чем у C. Плохая весть будет распространяться медленно пока счетчики задержек не примут значения бесконечности для данной сети. Только после этого станет ясно, что A не достижимо ни через C, ни через D, ни через E. Сколько времени на это потребуется, зависит от конкретного значения бесконечности в данной подсети.

Алгоритм Split Horizon (разделение направлений) предназначен для решения выше описанной проблемы. Маршрутизатор, используя данное правило, разделяет свои маршруты на столько групп, сколько у него есть активных интерфейсов. Далее обновления (вектора) для маршрутов, которые были получены через некоторый интерфейс, не должны передаваться через этот же интерфейс. Если теперь рассмотреть, как будет работать подсеть на рисунке 3.12 (b), то там проблем возникать не будет.
Однако, рассмотрим подсеть на рисунке 3.13. Если линия между C и D будет разрушена, то C сообщит об этом A и B. Однако, A знает что у B есть маршрут до D, а B знает что такой маршрут есть и у A. И мы приходим к проблеме циклических маршрутов, которую алгоритм Split Horizon решить не в состоянии.

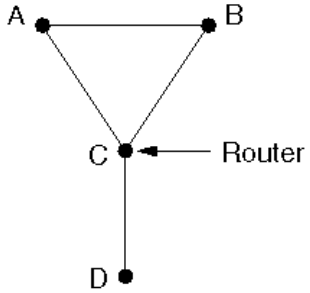


Рисунок 3.13. Пример неработоспособности алгоритма Split Horizon.

Алгоритм Poison Reverse (отравленный реверс) является другим способ решения проблемы бесконечного счетчика (или маршрутной петли). Алгоритм гласит – однажды изучив маршрут через какой-либо интерфейс, объявляй этот маршрут как «недоступный» обратно через тот же самый интерфейс.

2. *Проблема циклических маршрутов.*
   Данную проблему можно разрешить с помощью механизма Triggered Update (управляемые обновления). Использование данного механизма предписывает необходимость формирования мгновенных обновлений в том случае, когда происходит изменение состояния сети. Благодаря тому, что управляемые обновления передаются по сети с высокой скоростью, использование этого механизма может предотвратить появление циклических маршрутов. Однако, поскольку процесс передачи управляемых обновлений имеет вполне определенную конченую скорость, сохраняется возможность, что в процессе передачи регулярного обновления циклический маршрут все-таки возникнет.

4.2. Алгоритмы состояния канала

Алгоритмы состояния канала известны как алгоритмы "первоочередности наикратчайшего маршрута". Идея этих алгоритмов состоит в построении графа подсети, где вершины – маршрутизаторы, а дуги – линии связи. Алгоритм находит для любой пары абонентов наикратчайший маршрут в этом графе.
Проиллюстрируем концепцию наикратчайшего пути на рисунке 3.14 (стрелками указаны рабочие узлы).

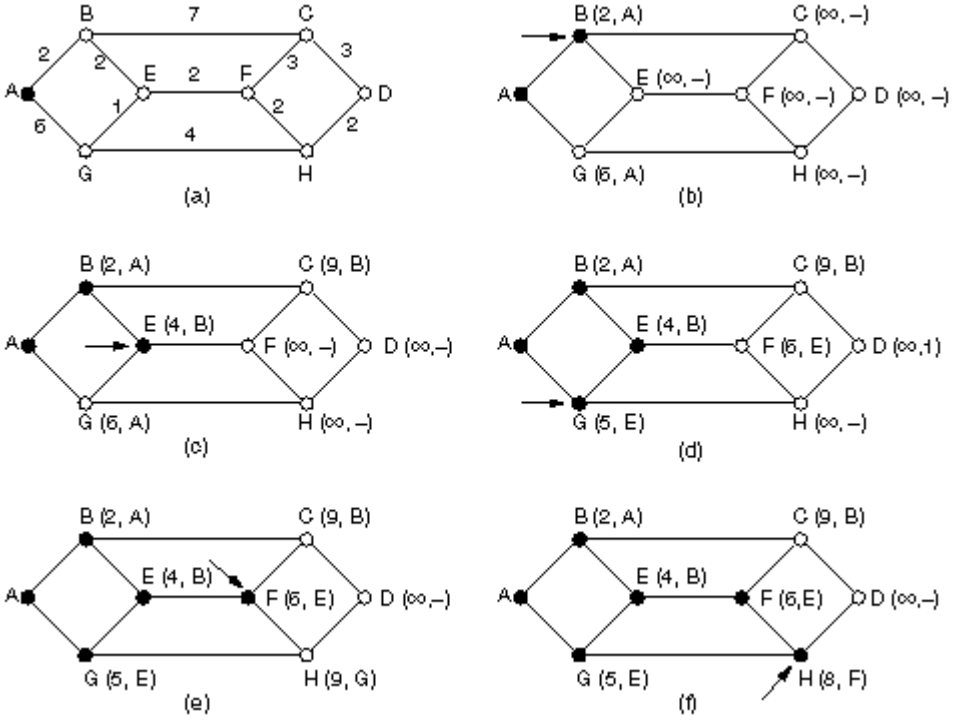


Рисунок 3.14. Первые пять шагов, используемые при вычислении кратчайшего пути от A до D.

Расстояние можно измерять в скачках, а можно в километрах. Тогда маршрут ABC длиннее маршрута ABE. Возможны и другие меры. Например, дуги графа могут быть размечены величиной средней задержки для пакетов. В графе с такой разметкой наикратчайший путь – наибыстрейший путь, который не обязательно имеет минимальное число скачков или километров.

В общем случае метки на дугах могут быть функциями от расстояния, пропускной способности, среднего трафика, стоимости передачи, средней длины очереди и других факторов. Изменяя весовую функцию, алгоритм будет вычислять наикратчайший путь в смысле разных мер.

Известно несколько алгоритмов вычисления наикратчайшего пути в графе. Один из них был предложен Дейкстра. Все вершины в графе помечаются в скобках расстоянием до исходной вершины вдоль наилучшего известного пути. Изначально никаких путей не известно и все вершины помечены бесконечностью. По мере работы алгоритма и нахождения путей метки могут меняться. Все метки могут быть либо пробными, либо постоянными. Изначально все метки пробные. Когда обнаруживается, что метка представляет наикратчайший возможный путь до исходной вершины, она превращается в постоянную и никогда более не меняется.
На рисунке 3.14 показан процесс построения маршрута. Самым интересным является момент на рисунке 3.14 (e). Здесь в окрестности вершины H оказывается вершина F, у которой пробная метка показывает, что путь через F короче. Поэтому метка у вершины H меняется так, как если бы маршрут шел через вершину H. Надо сделать оговорку, что алгоритм строит наикратчайший путь, начиная от точки доставки, а нет от точки отправления. Поскольку граф не ориентированный, то это никакого влияния на построение пути не оказывает.

Шлюзы, работающие по алгоритму Дейкстры, сначала определяют кратчайшие маршруты по всем сетям автономной системы. Для этого в каждом шлюзе строится полное дерево кратчайших путей с корнем в данном шлюзе. После этого изменения характеристик линий связи, определяющих длины соответствующих дуг графа, либо изменения топологии сети приводят лишь к небольшим дополнительным вычислениям для корректирования дерева кратчайших путей.

Шлюзы обмениваются только сведениями о длинах исходящих линий связи, а не векторами длин маршрутов, как в случае алгоритма Беллмана-Форда. Размер корректирующих пакетов со служебной информацией для маршрутизации мал и не зависит от числа сетей в автономной системе. Каждый шлюз посылает такие пакеты с помощью лавинной маршрутизации. При появлении в сети нового шлюза или включении новой линии связи изменения в топологии сети не учитываются при маршрутизации в течение некоторого времени для того, чтобы информация о происшедших изменениях успела достигнуть всех шлюзов автономной системы.
Алгоритмы состояния канала характеризуются следующим:

- направляют потоки маршрутной информации во все узлы объединенной сети. Однако каждый маршрутизатор посылает только ту часть маршрутной таблицы, которая описывает состояние его собственных каналов;
- предусматривают передачу информации о маршрутах в момент первоначального включения или при возникновении изменений в существующей структуре информационных связей.

4.3. Сравнение маршрутизации по вектору расстояния и маршрутизации с учетом состояния канала связи

Дистанционно-векторные алгоритмы хорошо работают только в небольших сетях. В больших сетях они засоряют линии связи интенсивным широковещательным трафиком. К тому же изменения конфигурации могут отрабатываться по этому алгоритму не всегда корректно, так как маршрутизаторы не имеют точного представления о топологии сети, а располагают лишь обобщенной информацией – вектором дистанций, к тому же полученной через посредников.
Отличаясь более быстрой сходимостью, алгоритмы состояния каналов несколько меньше склонны к образованию петель маршрутизации, чем алгоритмы вектора расстояния. С другой стороны, алгоритмы состояния канала характеризуются более сложными расчетами в сравнении с алгоритмами вектора расстояний, требуя большей процессорной мощности и памяти, чем алгоритмы вектора расстояний. Вследствие этого, реализация и поддержка алгоритмов состояния канала может быть более дорогостоящей.
Итоговое сравнение представлено в таблице 3.5.

Таблица 3.5 – Сравнение маршрутизации по вектору расстояния и маршрутизации с учетом состояния канала связи

Маршрутизация по вектору расстояния	**Маршрутизация с учетом состояния канала связи**
Видит топологию сети «глазами» соседних маршрутизаторов	Получает общий вид топологии всей сети
Суммирует вектор расстояния от одного маршрутизатора к другому	Вычисляет кратчайший путь до других маршрутизаторов
Частые периодические обновления топологической информации, медленная сходимость	Обновления инициируются фактом изменения топологии; быстрая сходимость
Передает копии таблицы маршрутизации только соседним маршрутизаторам	Передает пакеты с информацией об актуальном состоянии канала связи всем другим маршрутизаторам

4.4. Протоколы маршрутизации, используемые в IP-сетях

В современных IP-сетях наиболее широко используются следующие динамические протоколы маршрутизации:

- RIP
- OSPF
- IGRP
- EIGRP
- BGP

5. Протокол маршрутизации RIP

Протокол RIP является одним из первых, которые были использованы в информационно – вычислительных сетях вообще и в сети Internet – в частности. Этот протокол маршрутизации предназначен для сравнительно небольших и относительно однородных сетей (алгоритм Бэлмана-Форда). Протокол разработан в университете Калифорнии (Беркли), базируется на разработках фирмы Xerox и реализует те же принципы, что и программа маршрутизации «routed», используемая в ОС Unix (4BSD). Последняя версия протокола – RIP Version 2 [RFC 2453, 1721, 1722, 1723, 1724].

Маршрут здесь характеризуется вектором расстояния до места назначения. Предполагается, что каждый маршрутизатор является отправной точкой нескольких маршрутов до сетей, с которыми он связан. Каждый маршрутизатор, на котором запущен протокол RIP, содержит таблицу маршрутизации. Данная таблица содержит по одной записи для каждого места назначения, которое доступно через систему, на которой запущен протокол RIP. Каждая запись содержит, по крайней мере, следующие поля:

- IP-адрес места назначения.
- Маска подсети места назначения.
- IP-адрес ближайшего шлюза по пути до места назначения. Если узел назначения находится в одной из непосредственно подключенных сетей, то информация в данном поле является необязательной.
- Метрика, которая представляет конечную стоимость доставки датаграммы от текущего маршрутизатора до места назначения. Данная метрика является суммой стоимостей, назначенных для каждой из сетей, через которые пройдет датаграмма. Метрика является целым числом от 1 до 15 включительно.
- Флаг, указывающий, что информация о маршруте недавно изменилась;
- Различные таймеры, связанные с маршрутом.

Записи о подключенных сетях устанавливаются маршрутизатором, используя информацию, собранную средствами, которые протокол RIP не регламентирует. Системный администратор также может добавлять дополнительные записи о маршрутах. Данные записи являются «статическими маршрутами».

Первым двум полям записи мы обязаны появлению термина вектор расстояния (место назначение – направление; метрика – модуль вектора). Периодически (раз в 30 сек.) каждый маршрутизатор посылает широковещательно (мультикастово в RIP-2) копию своей таблицы маршрутизации всем соседям-маршрутизаторам, с которыми связан непосредственно. Маршрутизатор-получатель просматривает таблицу. Если в таблице присутствует новый путь или сообщение о более коротком маршруте, или произошли изменения длин пути, эти изменения фиксируются получателем в своей маршрутной таблице.

Протокол RIP должен обрабатывать два типа ошибок:

1. Бесконечный счетчик. Медленное распространение маршрутной информации по сети создает проблемы при динамичном изменении маршрутной ситуации (система не поспевает за изменениями). Малое предельное значение метрики улучшает сходимость, но не устраняет проблему. Проблема решается с помощью алгоритма Split Horizon.
2. Циклические маршруты. Проблема решается с помощью механизма Triggered Update.

Несоответствие маршрутной таблицы реальной ситуации типично не только для RIP, но характерно для всех протоколов, базирующихся на векторе расстояния, где информационные сообщения актуализации несут в себе только пары кодов: адрес места назначение и расстояние до него. Но все перечисленные методы и некоторые другие известные алгоритмы, решая одну проблему, часто вносят другие. Многие из этих методов могут при определенных условиях вызвать лавину широковещательных сообщений, что также дезорганизует сеть. Именно малая скорость установления маршрутов в RIP (и других протоколах, ориентированных на вектор расстояния) и является причиной их постепенного вытеснения другими протоколами.

В RIP сообщения инкапсулируются в udp-дейтограммы, при этом передача осуществляется через порт 520. В качестве метрики RIP использует число шагов до цели. Если между отправителем и приемником расположено три маршрутизатора, то считается, что между ними 4 шага. Такой вид метрики не учитывает различий в пропускной способности или загруженности отдельных сегментов сети. Применение вектора расстояния не может гарантировать оптимальность выбора маршрута, ведь, например, два шага по сегментам сети Ethernet обеспечат большую пропускную способность, чем один шаг через последовательный канал на основе интерфейса RS-232.

Маршрут по умолчанию имеет адрес 0.0.0.0. Каждому маршруту ставится в соответствие таймер тайм-аута и "сборщика мусора". Тайм-аут-таймер сбрасывается каждый раз, когда маршрут инициализируется или корректируется. Если со времени последней коррекции прошло 3 минуты или получено сообщение о том, что вектор расстояния равен 16, маршрут считается закрытым. Но запись о нем не стирается, пока не истечет время "уборки мусора" (2 мин). При появлении эквивалентного маршрута переключения на него не происходит, таким образом, блокируется возможность осцилляции между двумя или более равноценными маршрутами.

При реализации RIP можно выделить следующие режимы:

- Инициализация, определение всех "живых" интерфейсов путем посылки запросов, получение таблиц маршрутизации от других маршрутизаторов. Часто используются широковещательные запросы.
- Получен запрос. В зависимости от типа запроса высылается адресату полная таблица маршрутизации, или проводится индивидуальная обработка.
- Получен отклик. Проводится обновление таблицы маршрутизации (удаление, исправление, добавление).
- Регулярные обновления. Каждые 30 секунд вся или часть таблицы маршрутизации посылается всем соседним маршрутизаторам. Могут посылаться и специальные запросы при локальном изменении таблицы.

5.1. Область применения протокола, достоинства, недостатки

Протокол RIP является внутридоменным алгоритмом для работы в современных (в плане размера) автономных областях. Он не предназначен для использования в более сложных по архитектуре сетях. С появлением протоколов OSPF и IS-IS начало бытовать мнение о том, что RIP устарел. Несмотря на то, что новые внутридоменные протоколы превосходят RIP по большинству позиций, данный протокол все-таки имеет ряд преимуществ. В основном они проявляются при его использовании в сетях малого размера. При этом RIP налагает небольшие расходы в плане используемой пропускной способности канала и времени, используемого для конфигурации и управления. Протокол RIP также легок для внедрения, особенно по сравнению с новыми внутридоменными протоколами.

Протокол RIP обладает рядом специфических ограничений:

- Применение протокола ограничено сетями, в которых максимальный сегмент (диаметр сети) составляет 15 хопов.
- Протокол зависит от механизма "счет до бесконечности" для решения необычных ситуаций. Таким образом, если система состоит из нескольких сотен подсетей, и при этом возникает зацикливание процесса маршрутизации (который вовлекает все подсети), тогда решение зацикливания потребует либо много времени (если частота маршрутных пакетов обновления ограничена), либо пропускной полосы (если обновления рассылаются всякий раз, когда происходят изменения в сети).
- Протокол использует фиксированные метрики для сравнения альтернативных маршрутов. Это не подходит для случаев, когда маршруты необходимо выбирать, основываясь на параметрах реального времени – измеряемая задержка, надежность или загруженность канала.
- Число шагов важный, но не единственный параметр маршрута, да и 15 шагов не предел для современных сетей.

5.2. Классификация

Протокол RIP является:

- динамическим;
- одномаршрутным;
- одноуровневым;
- с интеллектом в маршрутизаторе;
- внутридоменным;
- алгоритмом вектора расстояний.

6. Протокол маршрутизации OSPF

Протокол OSPF (Open Shortest Path First, RFC-1245-48, RFC-1370, RFC-1583-1587, RFC-1850, RFC-2328-29, RFC-3137) является стандартным протоколом маршрутизации для использования в системах сетей IP любой сложности и альтернативой RIP в качестве внутреннего протокола маршрутизации. OSPF представляет собой протокол состояния связей и реализован в демоне маршрутизации «gated», который поддерживает также RIP и внешний протокол маршрутизации BGP.

6.1. Термины и определения протокола OSPF

При описании алгоритма OSPF используются несколько специальных терминов и понятий:

- **Автономная система (Autonomous System)**

Автономной системой (АС) называется группа маршрутизаторов, которая для обеспечения взаимного обмена информацией о маршрутах использует единый протокол маршрутизации.

АС может быть разбита на области. Протокол OSPF допускает совместную группировку множеств непрерывных сетей и хостов. Такая группа совместно с маршрутизатором, у которого интерфейсы подключены к любой из включенных в группу сетей, называется областью (area). Каждая область имеет собственную базу данных состояний связей и соответствующий граф.

Топология области неизвестна снаружи области. И наоборот, маршрутизаторы, находящиеся внутри области, ничего не знают о детальной топологии сети, внешней по отношению к области. Пример разбиения АС на области приведен на рисунке 3.15.

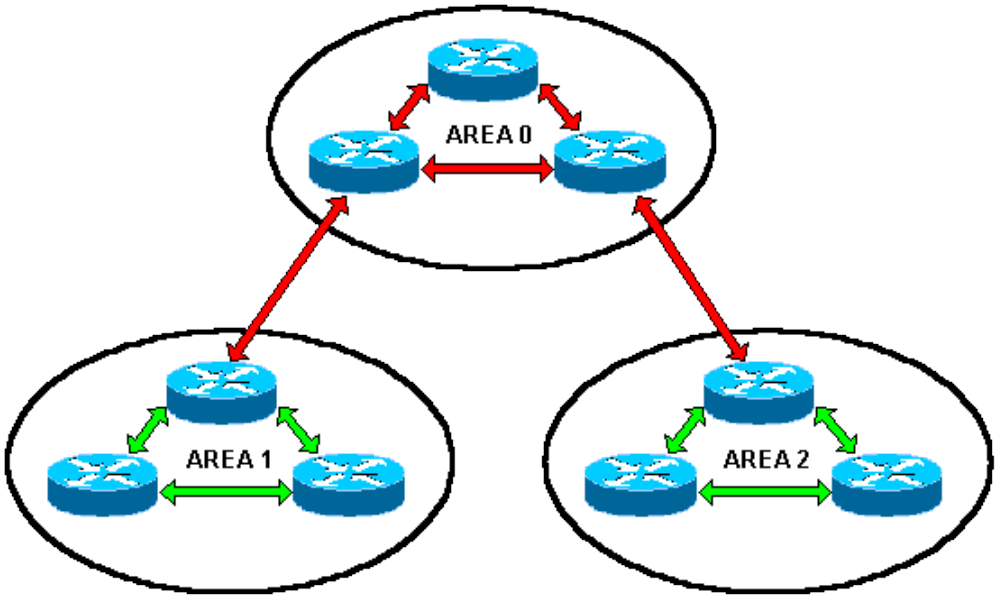


Рисунок 3.15. Разбиение автономной системы на области.

При разбиение АС на области вводится понятие основной OSPF Области (OSPF Backbone Area), или специальной OSPF Области 0 (зачастую пишут Область 0.0.0.0). Основная область ответственна за распределение маршрутной информации между остальными областями.

- **Типы маршрутизаторов OSPF**

Протокол OSPF относится к протоколам, которые обеспечивают иерархическую маршрутизацию. Как уже было выше отмечено, Область 0 используется для обеспечения информационного взаимодействия между остальными областями.

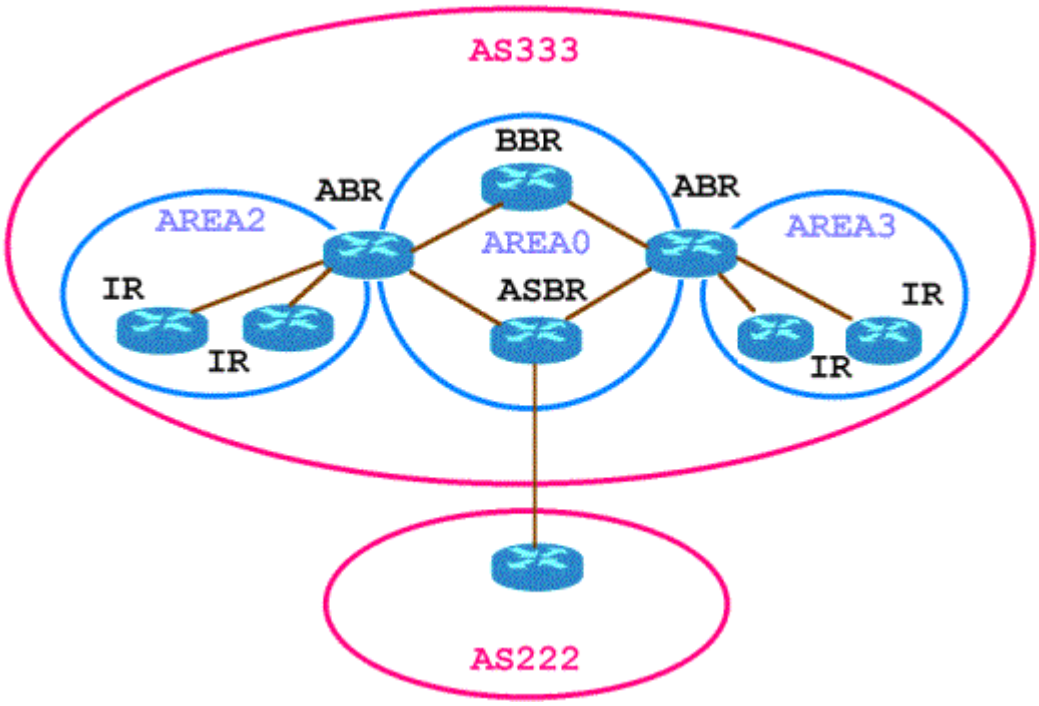


Рисунок 3.16. Типы маршрутизаторов OSPF.

В зависимости от того, к какой области принадлежит маршрутизатор, и какие информационные потоки через него проходят, различают четыре типа маршрутизаторов OSPF (рисунок 3.16):

- Внутренний маршрутизатор (Internal Router – IR).
  Маршрутизаторы типа Internal Router размещаются внутри автономной системы и не имеют интерфейсов, которые выходят за пределы этой автономной системы.
- Пограничный маршрутизатор области (Area Border Router – ABR).
  Маршрутизаторы типа Area Border Router размещаются на границе между несколькими областями в пределах автономной системы. Такие маршрутизаторы имеют интерфейсы, которые связывают их с маршрутизаторами, находящимися в других областях. Маршрутизаторы данного типа предназначены для того, чтобы передавать информацию о маршрутах между различными областями.
- Основной маршрутизатор (Backbone Router – BBR).
  К данному типу относятся все маршрутизаторы, которые имеют интерфейсы в нулевую область.
- Пограничный маршрутизатор АС (AS Boundary Router – ASBR).
  Маршрутизаторы типа AS Boundary Router обеспечивают информационный обмен с маршрутизаторами, которые расположены в других автономных системах.

- **Идентификатор маршрутизатора (Router ID)**

32 разрядный номер, присваиваемый каждому маршрутизатору, использующему протокол OSPF. Этот номер является уникальным в пределах автономной системы.

- **Назначенный маршрутизатор (Designated Router)**

Возможно возникновение ситуации, когда к одной сети типа broadcast окажутся подключенными несколько маршрутизаторов, входящих в один домен маршрутизации OSPF. Для того, чтобы избежать дублирования представления сети типа broadcast несколькими маршрутизаторами, в протоколе OSPF используется специальный алгоритм, с помощью которого выбирается назначенный маршрутизатор. В этом случае только один маршрутизатор обеспечивает передачу информации о маршрутах в сегменте сети.

- **Соседние маршрутизаторы (Neighboring Routers)**

Маршрутизаторы, которые подключены к одной и той же сети называются соседними маршрутизаторами.

- **Смежность (Adjacency)**

Два маршрутизатора из числа соседних могут быть выбраны для установления близких отношений, которые предполагают обмен информацией о маршрутах. Близкие отношения устанавливаются не в каждой паре соседствующих маршрутизаторов.

- **Базы данных состояний связей маршрутизаторов OSPF (Link-State Database)**

База данных Link-State Database отображает текущую структуру информационных связей в рассматриваемой области маршрутизации. Эти базы данных должны быть идентичными у всех маршрутизаторов, которые расположены в пределах одной области. Базы данных состоят из сообщений, которые называются Link – State Advertisement и формируются всеми активными маршрутизаторами данной области. Активным в данном случае считается маршрутизатор, который имеет хотя бы один подключенный канал в данной области.

- **Объявление состояния связи (Link State Advertisement, LSA)**

Блок данных, который содержит информацию о состоянии маршрутизатора или сети, называется объявлением о состоянии связи. В том случае, если данное объявление представляет состояние маршрутизатора, оно должно содержать информацию о статусе его интерфейсов и близких ему маршрутизаторов. Каждое такое объявление распространяется по всей автономной системе. Совокупность таких LSA формирует базу данных маршрутизации в каждом из маршрутизаторов. При этом используются различные типы LSA:

  - Сообщения LSA типа 1 – router link advertisement.
    Сообщения типа «router link advertisement» (состояния связей маршрутизатора) формируются каждым маршрутизатором для каждой области, в которой он имеет активные интерфейсы. Сообщения LSA типа 1 содержат объединенную информацию о состоянии связей, которые имеет маршрутизатор в данной области. Сообщения этого типа распространяются только в пределах одной области.
  - Сообщения LSA типа 2 – network link advertisement.
    Сообщения LSA типа «network link advertisement» (состояние связей сети) формируется только в сетях, которые могут быть отнесены к классу broadcast или NBMA (Non Broadcast Multi Access). В сообщении LSA типа 2 указываются идентификаторы всех маршрутизаторов, подключенных к данной сети. Формирование сообщений данного типа выполняется назначенным маршрутизатором.
  - Сообщения LSA типа 3, 4 – summary link advertisement.
    Сообщения LSA типа «summary link advertisement» формируются пограничными маршрутизаторами области (ABR) и направляются за пределы области, в которой они сформированы. Каждое сообщение данного типа содержит маршрут, который может быть использован для информационного обмена между различными областями в пределах одной автономной системы. В частности, LSA типа 3 описывают маршруты к сетям, LSA типа 4 описывают маршруты к пограничным маршрутизаторам АС (ASBR).
  - LSA типа 5 – external link advertisement.
    Сообщения LSA типа «external link advertisement» формируются пограничным маршрутизаторам АС (ASBR) и содержат информацию о маршрутах, которые являются внешними по отношению к данной автономной системе. Сообщения данного типа распространяются по всем областям автономной системы за исключением отдельных специально сконфигурированных областей, которые называются stub-areas.

- **Затопление (Flooding)**

Процесс распространения LSA в пределах автономной системы называется затоплением.

- **Протокол приветствия (Hello Protocol)**

Одним из компонентов протокола OSPF является протокол Hello, с помощью которого маршрутизаторы устанавливают и обслуживают соседские отношения. В частности, с помощью этого протокола производится выбор назначенного маршрутизатора для некоторых сетей.

6.2. Процесс сбора и обмена маршрутной информацией

Метрика

Метрика представляет собой оценку качества связи на данном физическом канале. Чем меньше метрика, тем лучше качество соединения. Метрика маршрута равна сумме метрик всех связей (сетей), входящих в маршрут.

Поскольку при работе алгоритма SPF ситуации, приводящие к счету до бесконечности, отсутствуют, значения метрик могут варьироваться в широком диапазоне. Кроме того, протокол OSPF позволяет определить для любой сети различные значения метрик в зависимости от типа сервиса (тип сервиса запрашивается дейтаграммой в соответствии со значением поля TOS ее заголовка). Для каждого типа сервиса будет вычисляться свой маршрут, и дейтаграммы, затребовавшие наиболее скоростной канал, могут быть отправлены по одному маршруту, а затребовавшие наименее дорогостоящий канал – по другому.

Метрика для различных типов сервиса может отвечать следующим параметрам:

1. Пропускная способность канала.
   Вычисляется следующим образом:
   $$\text{Метрика} = 10^8/\text{ifSpeed}$$
   где ifSpeed – пропускная способность текущего интерфейса.
   Таким образом, получаем типичные метрики для большинства каналов (таблица 3.6).

Таблица 3.6 – Значения метрики «Пропускная способность канала» протокола OSPF для различных типов каналов.

Тип канала (пропускная способность)	*Значение метрики*
≥ 100 Мб/с	1
Ethernet/802.3	10
E1	48
T1 (ESF)	65
64 Кб/с	1562
56 Кб/с	1785
19.2 Кб/с	5208
9.6 Кб/с	10416

2. Величина задержки распространения сигнала в канале
3. Надежность канала
4. Загруженность канала
5. Стоимость канала.
6. Числом датаграмм, стоящих в очереди для передачи
7. Размер максимального блока данных, который может быть передан через данный канал
8. Требованиями безопасности
9. Числом шагов до цели

Порядок расчета метрик, оценивающих задержку, надежность и стоимость, не определен. Администратор, желающий поддерживать маршрутизацию по этим типам сервисов, должен сам назначить разумные и согласованные метрики по этим параметрам.

Если не требуется маршрутизация с учетом типа сервиса (или маршрутизатор ее не поддерживает), используется метрика по умолчанию, равная метрике по пропускной способности (нулевой тип сервиса, TOS 0). Следует заметить, что маршрутизация по типам сервиса крайне редка, более того, она исключена из последних версий стандарта OSPF.

Построение БД состояний связей (Link-State Database)

Как уже было отмечено, для работы алгоритма SPF на каждом маршрутизаторе строится база данных состояния связей, представляющая собой полное описание графа OSPF-системы. При этом вершинами графа являются маршрутизаторы, а ребрами – соединяющие их связи. Для простоты будем рассматривать OSPF-систему, состоящую только из маршрутизаторов, соединенных линиями связи типа "точка-точка".

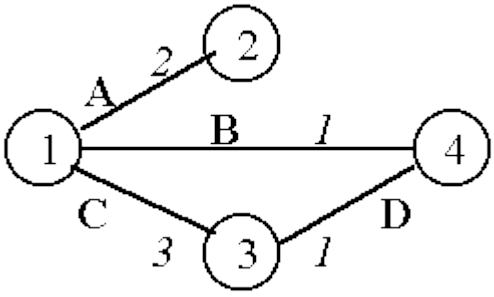


Рисунок 3.17. Пример OSPF-системы.

Обозначения на рисунке 3.17: ①, ②, ③, ④ – маршрутизаторы; A,B,C,D – линии связи (или просто связи), цифры означают метрику каждой связи. База данных состояния связей представляет собой таблицу, где для каждой пары смежных вершин графа (маршрутизаторов) указано ребро (связь), их соединяющее, и метрика этого ребра. Граф считается ориентированным. База данных состояния связей в нашем примере выглядит следующим образом (рисунок 3.18):

От→до	Сеть	Метрика
①→②	A	2
①→③	C	3
①→④	B	1
②→①	A	2
③→①	C	3
③→④	D	1
④→①	B	1
④→③	D	1

Рисунок 3.18. База данных состояния связей.

После инициализации модуля OSPF (например, после подачи питания на маршрутизатор) через все интерфейсы, включенные в OSPF-систему, начинают рассылаться Hello-сообщения. Задача протокола Hello – обнаружение соседей и установление с ними отношений смежности. Другая задача протокола Hello – выбор выделенного маршрутизатора в сети с множественным доступом, к которой подключено несколько маршрутизаторов.
Hello-пакеты продолжают периодически рассылаться и после того, как соседи были обнаружены. Таким образом, маршрутизатор контролирует состояние своих связей с соседями и может своевременно обнаружить изменение этого состояние (например, обрыв связи или отключение одного из соседей). Обрыв связи может быть также обнаружен и с помощью протокола канального уровня, который просигнализирует о недоступности канала.
В сетях с возможностью широковещательной рассылки Hello-пакеты рассылаются по мультикастинговому адресу 224.0.0.5 ("Всем OSPF-маршрутизаторам"). В других сетях все возможные адреса соседей должны быть введены администратором.

После установления отношений смежности для каждой пары смежных маршрутизаторов происходит синхронизация их баз данных. Эта же операция происходит при восстановлении ранее разорванного соединения, поскольку в образовавшихся после аварии двух изолированных подсистемах базы данных развивались независимо друг от друга. Синхронизация баз данных происходит с помощью протокола обмена (Exchange protocol).

Сначала маршрутизаторы обмениваются только описаниями своих баз данных (Database Description), содержащими идентификаторы записей и номера их версий, это позволяет

избежать пересылки всего содержимого базы данных, если требуется синхронизировать только несколько записей. Во время этого обмена каждый маршрутизатор формирует список записей, содержимое которых он должен запросить (то есть эти записи в его базе данных устарели, либо отсутствуют), и соответственно отправляет пакеты запросов о состоянии связей (Link State Request). В ответ он получает содержимое последних версий нужных ему записей в пакетах типа "Обновление состояния связей (Link State Update)". После синхронизации баз данных производится построение маршрутов.

Каждый маршрутизатор отвечает за те, и только те записи в базе данных состояния связей, которые описывают связи, исходящие от данного маршрутизатора. Это значит, что при образовании новой связи, изменении в состоянии связи или ее исчезновении (обрыве), маршрутизатор, ответственный за эту связь, должен соответственно изменить свою копию базы данных и немедленно известить все остальные маршрутизаторы OSPF-системы о произошедших изменениях, чтобы они также внесли исправления в свои копии базы данных. Эту задачу выполняет протокол затопления (Flooding protocol). При работе этого протокола пересылаются сообщения типа "Обновление состояния связей (Link State Update)", получение которых подтверждается сообщениями типа "Link State Acknowledgment". Каждая запись о состоянии связей имеет свой номер (номер версии), который также хранится в базе данных. Каждая новая версия записи имеет больший номер. При рассылке сообщений об обновлении записи в базе данных номер записи также включается в сообщение для предотвращения попадания в базу данных устаревших версий.
Маршрутизатор, ответственный за запись об изменившейся связи, рассылает сообщение "Обновление состояния связи" по всем интерфейсам. Однако новые версии состояния одной и той же связи должны появляться не чаще, чем оговорено определенной константой.

Далее на всех маршрутизаторах OSPF-системы действует следующий алгоритм.

1. Получить сообщение. Найти соответствующую запись в базе данных.
2. Если запись не найдена, добавить ее в базу данных, передать сообщение по всем интерфейсам.
3. Если номер записи в базе данных меньше номера пришедшего сообщения, заменить запись в базе данных, передать сообщение по всем интерфейсам.
4. Если номер записи в базе данных больше номера пришедшего сообщения и эта запись не была недавно разослана, разослать содержимое записи из базы данных через тот интерфейс, откуда пришло сообщение. Понятие "недавно" определяется значением константы.
5. В случае равных номеров сообщение игнорировать.

Протокол OSPF устанавливает также такую характеристику записи в базе данных, как возраст. Возраст равен нулю при создании записи. При затоплении OSPF-системы сообщениями с данной записью каждый маршрутизатор, который ретранслирует сообщение, увеличивает возраст записи на определенную величину. Кроме этого, возраст увеличивается на единицу каждую секунду. Из-за разницы во времени пересылки, в количестве промежуточных маршрутизаторов и по другим причинам возраст одной и той же записи в базах данных на разных маршрутизаторах может несколько различаться. Это нормальное явление. При достижении возрастом максимального значения (60 минут), соответствующая запись расценивается маршрутизатором как просроченная и непригодная для вычисления маршрутов. Такая запись должна быть удалена из базы данных.
Поскольку базы данных на всех маршрутизаторах системы должны быть идентичны, просроченная запись должна быть удалена из всех копий базы данных на всех маршрутизаторах. Это делается с использованием протокола затопления: маршрутизатор затапливает систему сообщением с просроченной записью. Соответственно, в описанный выше алгоритм обработки сообщения вносятся дополнения, связанные с получением просроченного сообщения и удалением соответствующей записи из базы данных.
Чтобы записи в базе данных не устаревали, маршрутизаторы, ответственные за них, должны через каждые 30 минут затапливать систему сообщениями об обновлении записей, даже если состояние связей не изменилось. Содержимое записей в этих сообщениях неизменно, но номер версии больше, а возраст равен нулю.

Вышеописанные протоколы обеспечивают актуальность информации, содержащейся в базе данных состояния связей, оперативное реагирование на изменения в топологии системы сетей и синхронизацию копий базы данных на всех маршрутизаторах системы.
Для обеспечения надежности передачи данных реализован механизм подтверждения приема сообщений, также для всех сообщений вычисляется контрольная сумма.
В протоколе OSPF может быть применена аутентификация сообщений, например, защита их с помощью пароля.

6.3. Алгоритм маршрутизации

Формат таблицы маршрутизации.

Структура таблицы маршрутизации (ТМ) содержит всю информацию, необходимую для перенаправления IP датаграммы к месту назначения. Каждая запись ТМ описывает набор лучших маршрутов к какому-либо месту назначения. При маршрутизации IP датаграммы выбирается запись ТМ, которая обеспечивает наилучший маршрут. Протокол OSPF обеспечивает также наличие маршрута по умолчанию (Идентификатор места назначения [Destination ID] = DefaultDestination, Маска адреса [Address Mask] = 0x00000000). Каждый маршрутизатор содержит одну ТМ. Каждая запись содержит следующие поля:

- *Тип места назначения (Destination Type)*

Это может быть либо «сеть», либо «маршрутизатор». При маршрутизации трафика актуальны только записи с типом «сеть». Записи с типом «маршрутизатор» используются исключительно как промежуточные шаги в процессе построения ТМ и хранятся для граничных маршрутизаторов области (Area Border Router) и граничных маршрутизаторов автономной системы (AS Boundary Routers).
Сеть (в терминах протокола OSPF) – диапазон IP адресов, к которым может быть направлен трафик. Данный диапазон может включать IP сети (класса A, B или C), IP подсети, IP суперсети и одиночные IP хосты. Маршрут по умолчанию также попадает в эту категорию.

- *Идентификатор места назначения (Destination ID)*

Это может быть либо идентификатор, либо имя места назначения. Данный факт зависит от типа места назначения. Для сетей идентификатором является их IP адрес. Для маршрутизаторов идентификатором является идентификатор маршрутизатора протокола OSPF (OSPF Router ID).

- *Маска адреса (Address Mask)*

Данное поле определено только для сетей. Для подсетей поле содержит маску подсети, для хостов маска – «все единицы» (0xffffffff).

- *Опциональные возможности (Optional Capabilities)*

Когда местом назначения является маршрутизатор, данное поле указывает опциональные OSPF возможности, поддерживаемые этим маршрутизатором. Единственной опциональной возможностью, определенной в RFC-2328, является возможность обработки сообщений типа AS-external-LSA.

- *Область (Area)*

Данное поле указывает область, связь с которой ведет к месту назначения.

- *Тип маршрута (Path-type)*

Существует четыре возможных типа маршрутов, используемых для маршрутизации трафика к месту назначения. Данные типы перечислены в порядке убывания степени их предпочтения: внутриобластной (intra-area), межобластной (inter-area), внешний типа 1 (type 1 external) и внешний типа 2 (type 2 external).
Внутриобластные маршруты указывают место назначения, принадлежащее одной из прикрепленных к текущему маршрутизатору областей. Межобластные маршруты – маршруты к местам назначения в других OSPF областях. Они обнаруживаются путем анализа полученных сообщений summary-LSA. Внешние маршруты автономной области указывают место назначения за пределами данной автономной области. Они обнаруживаются путем анализа полученных сообщений AS-external-LSA.

- *Стоимость (Cost)*

Поле содержит стоимость маршрута к месту назначения. Для всех за исключением внешних маршрутов второго типа данное поле содержит полную стоимость пути. Для внешних маршрутов второго типа данное поле содержит стоимость части пути,

который является внутренним по отношению к текущей автономной области. Данная стоимость рассчитывается как сумма стоимостей каналов маршрута.

- *Стоимость внешних маршрутов второго типа (Type 2 cost)*

Для данных маршрутов поле содержит стоимость внешней части пути. Данная стоимость объявляется граничным маршрутизатором автономной области и является наиболее значимой частью общей стоимости пути.

- *Источник состояния связей (Link State Origin)*

Действительно только для внутриобластных маршрутов. Данное поле содержит LSA (router-LSA или network-LSA), которое напрямую указывает на место назначения. Например, если местом назначения является транзитная сеть, тогда данное поле содержит network-LSA транзитной сети. LSA анализируется во время построения дерева наикратчайшего пути. Значение в данном поле используется протоколом MOSPF (Multicast OSPF).

Когда к месту назначения существует несколько маршрутов одинакового типа и стоимости (называются маршруты "одинаковой стоимости", “equal-cost” paths), тогда они хранятся в одной записи ТМ. Каждый из подобных маршрутов различается следующими полями:

- *Следующий хоп (Next hop)*

Интерфейс маршрутизатора, который необходимо использовать для перенаправления трафика к месту назначения. В широковещательных, Point-to-MultiPoint и NBMA сетях следующий хоп также включает IP адрес следующего маршрутизатора на пути к месту назначения.

- *Объявляющий маршрутизатор (Advertising router)*

Поле действительно только для межобластных и внешних маршрутов и содержит идентификатор маршрутизатора, объявляющего сообщения summary-LSA или AS-external-LSA, которые ведут к данному пути.

Формирование таблицы маршрутизации (алгоритм SPF).

Используя свои базы данных связей, маршрутизатор выполняет алгоритм построения таблицы маршрутизации шаг за шагом. На каждом шаге, маршрутизатор должен обращаться к отдельной части базы данных состояний связей. Данное обращение выполняется с помощью специальной функции просмотра.

Процесс построения таблицы маршрутизации может быть разбит на следующие шаги:

1. Текущая таблица маршрутизации признается недействительной и сохраняется для того, чтобы изменения в новой ТМ могли быть идентифицированы. Новая ТМ строится с нуля.
2. Внутриобластные маршруты вычисляются путем построения дерева наикратчайших путей для каждой присоединенной области. В частности, все записи таблицы маршрутизации, у которых Тип Места Назначения есть "Пограничный Маршрутизатор Области", вычисляются на данном этапе. Данный этап разбивается на две части. В первую очередь, дерево строится только с учетом связей (каналов) между маршрутизаторами и промежуточными сетями. Затем сети «заглушки» (stub networks) включаются в дерево. Во время построения дерева наикратчайших путей области также вычисляется свойство области «Возможность Транзита» для дальнейшего использования в шаге 4.
3. Межобластные маршруты вычисляются путем анализа сообщений summary-LSA. Если маршрутизатор подсоединен к множеству областей (например, пограничный маршрутизатор области), то анализируются сообщения summary-LSA только из основной области.
4. В пограничных маршрутизаторах области, подсоединенных к одной или более промежуточной области, анализируются сообщения summary-LSA из промежуточных областей. Цель анализа – выяснить, существуют ли лучшие маршруты через промежуточные области, чем те, которые были найдены на этапах 2 и 3.
5. Маршруты к внешним местам назначения вычисляются путем анализа сообщений AS-external-LSA. Расположение пограничных маршрутизаторов АС (которые генерируют сообщения AS-external-LSA) определяется на этапах 2-4.

Изменения, сделанные в ТМ, могут инициировать OSPF протокол к принятию дополнительных действий. Например, изменение в внутриобластном маршруте заставит пограничный маршрутизатор области генерировать новое сообщение summary-LSA.

Алгоритм маршрутизации.
Когда получена IP дейтаграмма, OSPF маршрутизатор должен найти запись в ТМ, которая содержит место назначения, наиболее подходящее для дейтаграммы. Данная запись должна обеспечить исходящий интерфейс и маршрутизатор следующей пересылки.
Выбирается запись ТМ, которая отвечает правилу «Longest Match». Предположим, что маршрутизатор переправляет пакет к месту назначения с адресом 192.9.1.1. При этом запись 192.9.1/24 всегда будет предпочтительней записи 192.9/16 вне зависимости от типа маршрута.
Если не найдено подходящей записи ТМ, или наиболее подходящая найденная запись входит в список «отвернутых», то тогда считается, что пакет невозможно доставить к месту назначения. Пакет должен быть сброшен. ICMP-сообщение «Получатель недоступен» должно быть сгенерировано для источника пакета.

6.4. Область применения, достоинства, недостатки

Благодаря использованию более совершенного алгоритма протокол OSPF работает в сетях любой сложности и не имеет ограничений и побочных эффектов протокола RIP. При этом время, используемое на построение таблицы маршрутов, и нагрузка на сеть для передачи служебной информации в среднем меньше по сравнению с тем, что потребовал бы RIP для такой же системы.
Преимущества протокола OSPF:

- Отсутствие ограничений на размер сети.
- Достаточно большую скорость установления маршрута.
- Процедура установления подлинности источника информации при передаче и получении обновлений маршрутов.
- Иерархическая маршрутизация.
- Может быть несколько маршрутных таблиц, по одной на каждый тип сервиса (TOS).
- При существовании эквивалентных маршрутов OSFP распределяет поток равномерно по этим маршрутам (Load balancing).
- При связи точка-точка не требуется IP-адрес для каждого из концов (экономия адресов).

Недостатки:

- Трудно получить информацию о предпочтительности каналов для узлов, поддерживающих другие протоколы, или со статической маршрутизацией.
- Протокол OSPF гораздо сложнее протокола RIP.

6.5. Классификация

Протокол OSPF является:

- динамическим;
- многомаршрутным;
- иерархическим;
- с интеллектом в маршрутизаторе;
- внутридоменным;
- алгоритмом состояний канала.

7. Протокол маршрутизации BGP

Протокол BGP (RFC-1771, BGP-4; RFC-1772-74, 1997, 2918, 3392) разработан сотрудниками компаний IBM и CISCO и представляет собой протокол внешней маршрутизации автономных систем.

Протокол BGP может быть использован для определения различных типов маршрутов:

- ✓ Inter-autonomous system routing маршруты, которые соединяют данную автономную систему с одной или несколькими другими автономными системами.
- ✓ Intra-autonomous system routing маршруты – протокол может быть использован для определения маршрута внутри автономной системы, в том случае, когда несколько маршрутизаторов участвуют в процессе определения маршрута BGP.
- ✓ Pass-through autonomous system маршруты – протокол может быть использован для определения маршрутов, которые проходят через автономную систему, которая не участвует в процессе BGP.

Принципиальным отличием внешней маршрутизации от внутренней является наличие маршрутной политики, то есть при расчете маршрута рассматривается не столько метрика, сколько политические и экономические соображения. Это обстоятельство не позволяет адаптировать под задачу внешней маршрутизации готовые протоколы внутренней маршрутизации, просто применив их к графу автономных систем, как раньше они применялись к графу сетей. По той же причине существующие подходы – дистанционно-векторный и состояния связей – непригодны для решения поставленной задачи.

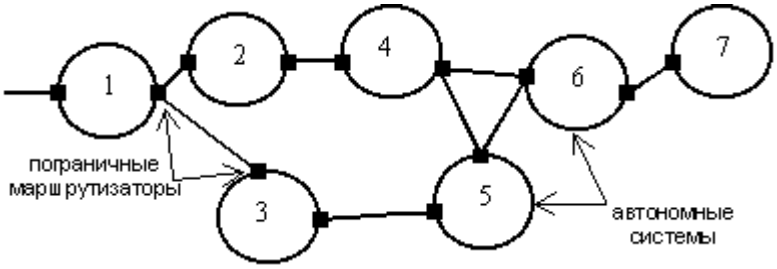


Рисунок 3.19. Автономные системы.

Например, рассмотрим систему маршрутизаторов, аналогичную графу автономных систем, изображенному на рисунке 3.19. Алгоритм SPF гарантирует, что если узел (1) вычислил маршрут в узел (6) как (1)-(3)-(5)-(6), то узел (3) также будет использовать маршрут (3)-(5)-(6). Это происходит потому, что все узлы одинаково интерпретируют метрики связей и используют одни и те же математические процедуры для вычисления маршрутов. Однако если применять протокол состояния связей на уровне автономных систем может произойти следующее: АС1 установила, что оптимальный маршрут по соображениям пропускной способности сетей в АС6 выглядит 1-3-5-6. Но АС3 считает, что выгодней добираться в АС6 по маршруту 3-1-2-4-6, так как АС5 дорого берет за передачу транзитного трафика. В итоге, из-за того, что у каждой АС свое понятие метрики, то есть качества маршрута, происходит зацикливание.

Казалось бы, дистанционно-векторный подход мог бы решить проблему: АС3, не желающая использовать АС5 для транзита в АС6, просто не прислала бы в АС1 элемент вектора расстояний для АС6, следовательно первая система никогда не узнала бы о маршруте 1-3-5-6. Но с другой стороны при использовании дистанционно-векторного подхода получателю вектора расстояний неизвестно описание маршрута на всей его протяженности. Рассмотрим другую политическую ситуацию на примере того же Рис. 20: АС1 получает от АС3 вектор АС6=2 и направляет свой трафик в АС6 через АС3, не подозревая, что на этом маршруте располагается АС5, которая находится с АС1 в состоянии войны и, естественно, не пропускает транзитный трафик от и для АС1. В результате АС1, установив внешне безобидный маршрут в АС6, осталась без связи с этой автономной системой. Если бы АС1 получила полное описание маршрута, то она несомненно бы выбрала альтернативный вариант 1-2-4-6, однако в дистанционно-векторных протоколах такая информация не передается.

7.1. Общая схема работы

BGP-маршрутизаторы соседних АС, решившие обмениваться маршрутной информацией, устанавливают между собой соединения по протоколу BGP и становятся BGP-соседями (BGP-peers). Имеется 4 типа сообщений, которым обмениваются BGP-соседи:

- ✓ OPEN – посылается после установления TCP-соединения. Ответом на OPEN является сообщение KEEPALIVE, если вторая сторона согласна стать BGP-соседом; иначе посылается сообщение NOTIFICATION с кодом, поясняющим причину отказа, и соединение разрывается.
- ✓ KEEPALIVE – сообщение предназначено для подтверждения согласия установить соседские отношения, а также для мониторинга активности открытого соединения: для этого BGP-соседи обмениваются KEEPALIVE-сообщениями через определенные интервалы времени.
- ✓ UPDATE – сообщение предназначено для анонсирования и отзыва маршрутов. После установления соединения с помощью сообщений UPDATE пересылаются все маршруты, которые маршрутизатор хочет объявить соседу (full update), после чего пересылаются только данные о добавленных или удаленных маршрутах по мере их появления (partial update).
- ✓ NOTIFICATION – сообщение этого типа используется для информирования соседа о причине закрытия соединения. После отправления этого сообщения BGP-соединение закрывается.

BGP использует подход под названием «path vector», являющийся развитием дистанционно-векторного подхода. BGP-соседи рассылают друг другу векторы путей (path vectors). Вектор путей, в отличие от вектора расстояний, содержит не просто адрес сети и расстояние до нее, а адрес сети и список атрибутов (path attributes), описывающих различные характеристики маршрута от маршрутизатора-отправителя в указанную сеть. Например, наиболее важным атрибутом маршрута является AS_PATH – список номеров автономных систем, через которые должна пройти дейтаграмма на пути в указанную сеть. В дальнейшем для краткости мы будем называть набор данных, состоящих из адреса сети и атрибутов пути до этой сети, маршрутом в данную сеть.
Данных, содержащихся в атрибутах пути, должно быть достаточно, чтобы маршрутизатор-получатель, проанализировав их с точки зрения политики своей АС, мог принять решение о приемлемости или неприемлемости полученного маршрута.

Очевидно, что BGP-маршрутизаторы, находящиеся в одной АС, также должны обмениваться между собой маршрутной информацией. Это необходимо для согласованного отбора внешних маршрутов в соответствии с политикой данной АС и для передачи транзитных маршрутов через автономную систему. Такой обмен производится также по протоколу BGP, который в этом случае часто называется IBGP (Internal BGP), (соответственно, протокол обмена маршрутами между маршрутизаторами разных АС обозначается EBGP –External BGP).

Отличие IBGP от EBGP состоит в том, что при объявлении маршрута BGP-соседу, находящемуся в той же самой АС, маршрутизатор не должен добавлять в AS_PATH номер своей автономной системы. Действительно, если номер АС будет добавлен, и сосед анонсирует этот маршрут далее (опять с добавлением номера той же АС), то одна и та же АС будет перечислена AS_PATH дважды, что расценивается как цикл.
Это очевидное правило влечет за собой интересное следствие: чтобы не возникло циклов, маршрутизатор не может анонсировать по IBGP маршрут, полученный также по IBGP, поскольку нет способов определить зацикливание при объявлении BGP-маршрутов внутри одной АС.

Следствием этого следствия является необходимость полного графа IBGP-соединений между пограничными маршрутизаторами одной автономной системы: то есть каждая пара маршрутизаторов должна устанавливать между собой соединение по протоколу IBGP. При этом возникает проблема большого числа соединений (порядка N^2, где N-число BGP-маршрутизаторов в АС). Для уменьшения числа соединений применяются различные ре-

шения: разбиение АС на конфедерации (подсистемы), применение серверов маршрутной информации и др.

Сервер маршрутной информации (аналог выделенного маршрутизатора в OSPF), обслуживающий группу BGP-маршрутизаторов, работает очень просто: он принимает маршрут от одного участника группы и рассылает его всем остальным. Таким образом, участникам группы нет необходимости устанавливать BGP-соединения попарно; вместо этого каждый участник устанавливает одно соединение с сервером. Группой маршрутизаторов могут быть, например, все BGP-маршрутизаторы данной АС, однако маршрутные серверы могут применяться для уменьшения числа соединений также и на внешних BGP-соединениях – в случае, когда в одной физической сети находится много BGP-маршрутизаторов из различных АС (например, в точке обмена трафиком между провайдерами, рисунок 3.20).

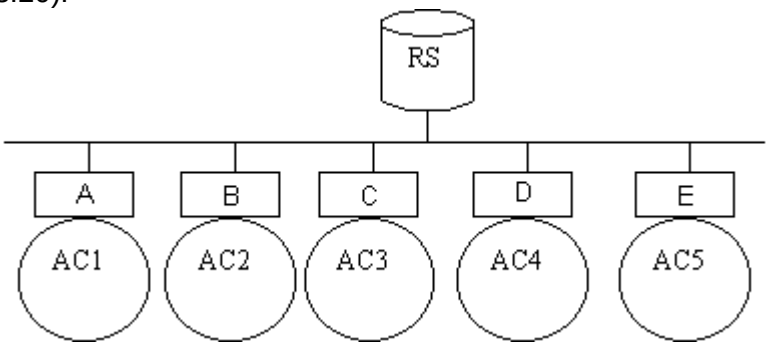


Рисунок 3.20. Точка обмена трафиком (Internet Exchange Point).
A-E – пограничные BGP-маршрутизаторы, AC1-AC5 – сети автономных систем,
RS – сервер маршрутной информации.

Следует особо отметить, что сервер маршрутной информации обслуживает только объявления маршрутов, а не сам трафик по этим маршрутам. Например, маршрутизатор A объявляет серверу RS маршруты в сети AC1. Маршрутизатор E получает информацию об этих маршрутах от сервера RS, но при этом в таблице маршрутов узла E следующим маршрутизатором на пути в AC1 значится узел A, что абсолютно разумно, поскольку эти узлы могут передавать данные друг другу непосредственно. Исключение маршрутного сервера из маршрута производится путем установки значения атрибута NEXT_HOP: анонсируя маршруты в сеть AC1, сервер RS указывает NEXT_HOP=A. Таким образом, маршрутизатор, получивший и принявший к использованию такой маршрут, будет пересылать данные, предназначенные для AC1, непосредственно маршрутизатору A.

Из вышесказанного можно сделать два важных вывода. Во-первых, узел, указанный как NEXT_HOP, должен быть достижим, то есть в таблице маршрутов маршрутизатора, принявшего маршрут с этим атрибутом, должна быть запись об узле NEXT_HOP или его сети. Если такой записи нет, то маршрутизатор должен забраковать полученный маршрут, потому что он не знает, как отправлять дейтаграммы к узлу NEXT_HOP.
Во-вторых, очевидно, что сервер маршрутной информации не является маршрутизатором. То есть в общем случае узел, на котором работает модуль BGP, – не обязательно маршрутизатор. В технических документах этот факт подчеркивается тем, что для обозначения BGP-узла используется термин BGP-speaker (не router).

7.2. Процесс обмена маршрутной информацией

Процесс отбора (Decision Process)

Рассмотрим действия BGP-маршрутизатора при получении и анонсировании маршрута (рисунок 3.21). Маршрутизатор использует три базы данных: Adj-RIBsIn, Loc-RIB и Adj-RIBsOut, в которых содержатся маршруты, соответственно, полученные от соседей, используемые самим маршрутизатором и объявляемые соседям. Также на маршрутизаторе сконфигурированы две политики: политика приема маршрутов (accept policy) и политика объявления маршрутов (announce policy). Для обработки маршрутов в базах данных в соответствии с имеющимися политиками маршрутизатор выполняет процедуру под названием процесс отбора (decision process), описанную ниже.

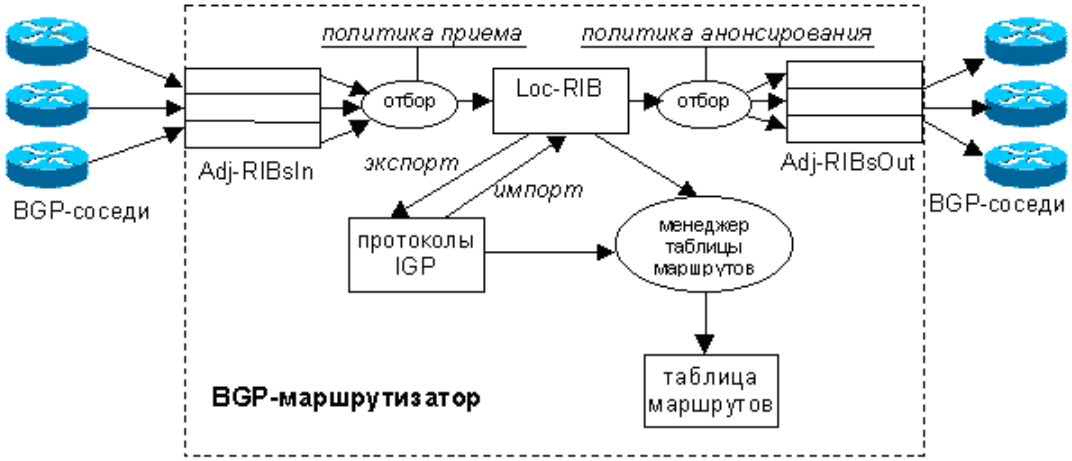


Рисунок 3.21. Обработка маршрутной информации модулем BGP

Маршруты, полученные от BGP-соседей, помещаются в базу данных Adj-RIBsIn. В соответствии с политикой приема для каждого маршрута в Adj-RIBsIn вычисляется приоритет (это называется фазой 1 процесса отбора). В результате этих действий некоторые маршруты могут быть отбракованы (признаны неприемлемыми).

Далее (фаза 2) для каждой сети из всех имеющихся (полученных и неотбракованных) вариантов выбирается маршрут с большим приоритетом. Результаты заносятся в базу Loc-RIB, откуда менеджер маршрутной таблицы IP-модуля может их взять для установки в таблицу маршрутов маршрутизатора и для экспорта во внутренний протокол маршрутизации с тем, чтобы и другие узлы автономной системы имели маршруты к внешним сетям. И наоборот, чтобы другие автономные системы имели маршруты к сетям данной АС, из таблиц протокола (протоколов) внутренней маршрутизации могут извлекаться номера сетей своей АС и заноситься в Loc-RIB.

Задача третьей фазы – отбор маршрутов для анонсирования (рассылки соседям). Из LocRIB выбираются маршруты, соответствующие политике анонсирования, и результат помещается в базу Adj-RIBsOut, содержимое которой и рассылается BGP-соседям. Возможно, что маршрутизатор имеет разные политики анонсирования для каждого соседа.
Важным свойством процесса отбора является то, что BGP-маршрутизатор объявляет только те маршруты, которые он сам использует. Это обстоятельство является следствием природы IP-маршрутизации: при выборе маршрута для дейтаграммы учитывается только адрес получателя и никогда – адрес отправителя. Таким образом, если маршрутизатор сам использует один маршрут в сеть X, а соседу объявил другой, то дейтаграммы от соседа все равно будут пересылаться в сеть X по тому маршруту, который использует сам маршрутизатор, поскольку адрес отправителя при выборе маршрута IP-модулем не рассматривается.

7.3. Конфликты маршрутных политик

Рассмотрим пример (рисунок 3.22).

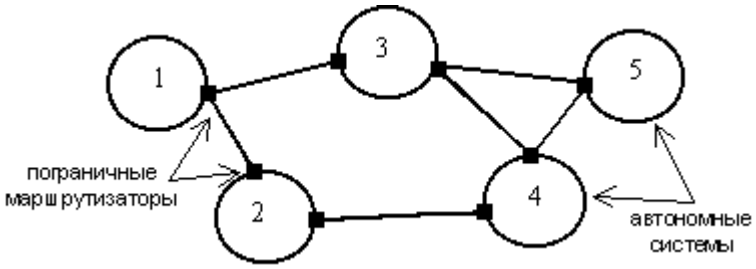


Рисунок 3.22. Автономные системы.

Предположим, автономная система 5 формирует маршрут до сетей автономной системы 1. Политическая ситуация такова, что АС3 берет с АС5 существенно б**о**льшие деньги за транзитный трафик, чем АС4 и АС2, поэтому АС5 формирует политику приема маршру-

тов, в соответствии с которой на маршруты, полученные от АС3, но ведущие в сети других АС (АС1 и АС2), назначается меньший приоритет, чем на маршруты в те же сети, но полученные от АС4. Таким образом, АС5 ожидает, что ее маршрутизатор установит маршрут в АС1 как 5-4-2-1 (естественно, что к сетям самой АС3 маршрут будет кратчайший: 5-3).

Однако выясняется, что маршрутизатор АС5 выбрал маршрут 5-3-1, то есть политика не действует, хотя все настройки в АС5 произведены верно. Дело в том, что к четвертой системе АС5 относится гораздо более дружелюбно и тарифицирует ее трафик по обычным расценкам, вследствие чего для АС4 нет разницы, как добираться в АС1: 4-3-1 или 4-2-1. АС4 по какой-то причине выбирает маршрут 4-3-1, и поскольку маршрутом 4-2-1 она не пользуется, она его и не объявляет. В результате у АС5 нет вариантов: эта автономная система просто не получает анонс маршрута 4-2-1 и вынуждена использовать маршрут 3-1.

Для решения проблемы АС5 должна обратиться к АС4 с просьбой установить политику приема маршрутов так, чтобы использовать маршрут 4-2-1. Однако, если АС3 тарифицирует трафик АС4 не просто по обычным тарифам, а со скидками, то для АС4 невыгодно вносить требуемые изменения, и АС5 остается перед выбором: либо дорого платить за трафик через АС3, либо не иметь связи с АС1.

Описанная ситуация представляет собой конфликт интересов, который не может быть разрешен программным обеспечением протокола BGP. Отметим, что в этом нет вины самого протокола; мы наблюдаем здесь побочный эффект технологии маршрутизации "только по адресу получателя", используемой протоколом IP.

7.4. Формулировка маршрутных политик

Способы описания маршрутных политик не являются частью протокола BGP и отличаются в различных реализациях BGP. Однако в любом случае политики базируются на критериях отбора маршрутов и модификации атрибутов маршрутов, попавших под критерии отбора. Модификация атрибутов маршрута в свою очередь влияет на приоритет этого маршрута при отборе нескольких альтернативных маршрутов во время фазы 2.

7.5. Классификация

Протокол BGP является:

- ✓ динамическим;
- ✓ одномаршрутным;
- ✓ одноуровневым;
- ✓ с интеллектом в маршрутизаторе;
- ✓ междоменным;
- ✓ протоколом вектора путей.

8. Протокол UDP

Протокол User Datagram Protocol представляет собой:

1. Ненадежный протокол без установления соединения. Поэтому наиболее часто применяется при взаимодействии клиента и сервера, когда клиент посылает запрос, а сервер отвечает сообщением, которое и играет роль подтверждения сообщения.
2. Отсутствие механизма, позволяющего разбивать поток данных на фрагменты и последующей их сборки. Это делает возможным его применение только при коротких транзакциях, где все данные не превышают по объему размера одной дейтаграммы. Хотя ничто не мешает дейтаграмме быть впоследствии разбитой на фрагменты протоколом IP.
3. Обеспечивает единственную транспортную услугу для протоколов Прикладного уровня – проверка данных на целостность.
4. Поддержка широковещания из-за отсутствия механизма подтверждения получения пакетов. Поэтому наиболее часто сервисы также применяют UDP для выполнения функции широковещания. По этой причине UDP (в отличие от TCP) используется в приложениях реального времени.

Заголовок UDP сообщения представлен на рисунке 3.23.

Рисунок 3.23.

Максимальный объем переносимых данных – 65507 байт.

9. Протокол TCP

9.1. Функции протокола

Протокол Transmission Control Protocol представляет собой надежный протокол с установлением соединения, являющийся альтернативой UDP, и отвечающий за большинство передач пользовательских данных по сетям TCP/IP. Основные функции протокола TCP:

1. Гарантия доставки данных – подтверждение получения пакетов.
2. Отслеживание и исправление ошибок.
3. Управление потоком.

9.2. Заголовок протокола TCP

Рисунок 3.24.

Для обеспечения указанных сервисов TCP-заголовок должен быть значительно больше, чем UDP-заголовок, и имеет такой же размер, что и IP-заголовок (20 байт без опций).

- ✓ *Порт источника.* Идентифицирует номер порта передающей системы.
- ✓ *Порт назначения*. Указывает номер порта системы назначения. Номера портов перечислены в документе «Assigned Numbers», а также в файле SERVICES каждой TCP/IP-системы.
- ✓ *Порядковый номер.* Определяет положение конкретного сегмента по отношению ко всей последовательности данных. Информация, поставляемая Транспортному уровню, в терминологии протокола TCP рассматривается как последовательность (sequence), которую протокол разбивает на сегменты для передачи по сети.
- ✓ *Номер подтверждения.* Задает максимальный номер байта в сегменте, увеличенный на единицу, который подтверждающая система ожидает получить от отправителя.
- ✓ *Смещение данных*. Задает длину в 4-байтных словах TCP-заголовка (до 60 байт).
- ✓ *Биты управления.* Содержит шесть 1-битных флагов, выполняющих следующие функции:
  - *URG* – показывает, что последовательность содержит срочные данные и активирует поле указателя срочности.
  - *ACK* – отмечает, что сообщение является подтверждением ранее полученных данных и активирует поле номера подтверждения.
  - *PSH* – предписывает системе-получателю передать всю информацию текущей последовательности, полученной на данный момент, приложению, не дожидаясь поступления остальных фрагментов.
  - *RST* – инструктирует систему-получателя отбросить все сегменты текущей последовательности, полученные к настоящему моменту, и начать установление tcp-соединения заново.
  - *SYN* – используется во время процедуры установления соединения для синхронизирования нумераторов передачи данных между взаимодействующими системами.
  - *FIN* – извещает другую систему, что передача данных закончена и соединение должно быть разорвано.
- ✓ *Окно*. Реализует механизм управления потоком данных путем объявления количества байтов, которые получатель может принять от источника.
- ✓ *Указатель срочности*. Задействуется совместно с битом URG, определяет данные последовательности, которые должны рассматриваться получателем как срочные.
- ✓ *Данные.* В пакетах SYN и ACK это поле остается пустым.

9.3. Алгоритм работы протокола TCP

Установление соединения

TCP-соединение является логическим, и система может одновременно активировать несколько TCP-соединений с одним или несколькими пунктами назначений.

В качестве примера можно рассмотреть базовую транзакцию типа клиент-сервер между web-браузером и web-сервером. Когда пользователь набирает url в браузере, программа открывает tcp-соединение с сервером для передачи html-файла по умолчанию, используемого браузером для показа начальной страницы. Соединение продолжается ровно столько времени, сколько требуется, чтобы передать одну страницу. Когда пользователь щелкает на гиперссылке для того, чтобы перейти к другой странице, требуется создание нового tcp-соединения.

1. Процесс коммуникации между клиентом и сервером стартует с того, что клиент создает первое tcp-сообщение, инициируя трехстороннее квитирование установления связи. Сообщение не содержит данных, в нем выставлен бит SYN, и в поле порядкового номера содержится целое число, называемое *начальным порядковым номером (ISN).* Система берет за основу алгоритм постоянного инкрементирования для определения ISN, который будет являться уникальным идентификатором для данного tcp-соединения. Основная цель сообщения – запрос на соединение.
2. Когда серверу приходит данное сообщение, он отвечает собственным контрольным сообщением. Ответное сообщение выполняет две функции: подтверждает получение клиентского запроса (выставлен бит ACK) и инициирует свое собственное соединение (выставлен бит SYN). Поле номера подтверждения содержит присланный клиентом ISN+1. Поле порядкового номера содержит ISN, вычисленный сервером.
3. Когда клиент получает подтверждение от сервера, он посылает свое собственное подтверждение (выставленный бит ACK), завершая тем самым процесс установления соединения. Порядковый номер содержит клиентский ISN+1. Поле номера подтверждения содержит серверный ISN+1.

Передача данных

1. Системы, используя полей опций, договариваются о максимальном размере сегмента, который каждая из них в состоянии принять. Конкретное значение зависит от протокола Канального уровня.
2. Процесс обмена данными на примере http-транзакции. Клиент посылает серверу желаемый url в отдельном пакете. Следует отметить, что порядковый номер этого пакета равен ISN+1. Помимо этого важно заметить, что выставлен бит PSH, указывающий, что сервер должен передать всю информацию пакета приложению.
3. После получения сообщения клиента сервер отправляет ему подтверждение. Номер подтверждения = порядковый номер посланного клиентского сообщения + размер переданных клиентом данных в байтах.
4. Следующий шаг – это ответ сервера на запрос клиента путем отправки требуемого html-файла.

Завершение соединения

1. Процесс начинается, когда одна из взаимодействующих сторон посылает сообщение, в котором установлен контрольный бит FIN. То, какая из систем инициирует процесс завершения соединения, зависит от приложения, генерировавшего трафик. В случае html-транзакции сервер может включить бит FIN в последний пакет данных, или же это может быть отдельный пакет.
2. Далее клиент подтверждает разрыв сообщением ACK.
3. Клиент посылает аналогичное сообщение на разрыв FIN.
4. Сервер подтверждает разрыв сообщением ACK.

Только после этого соединение считается полностью завершенным.

Полная диаграмма переходов протокола TCP показана на рисунке 3.25.

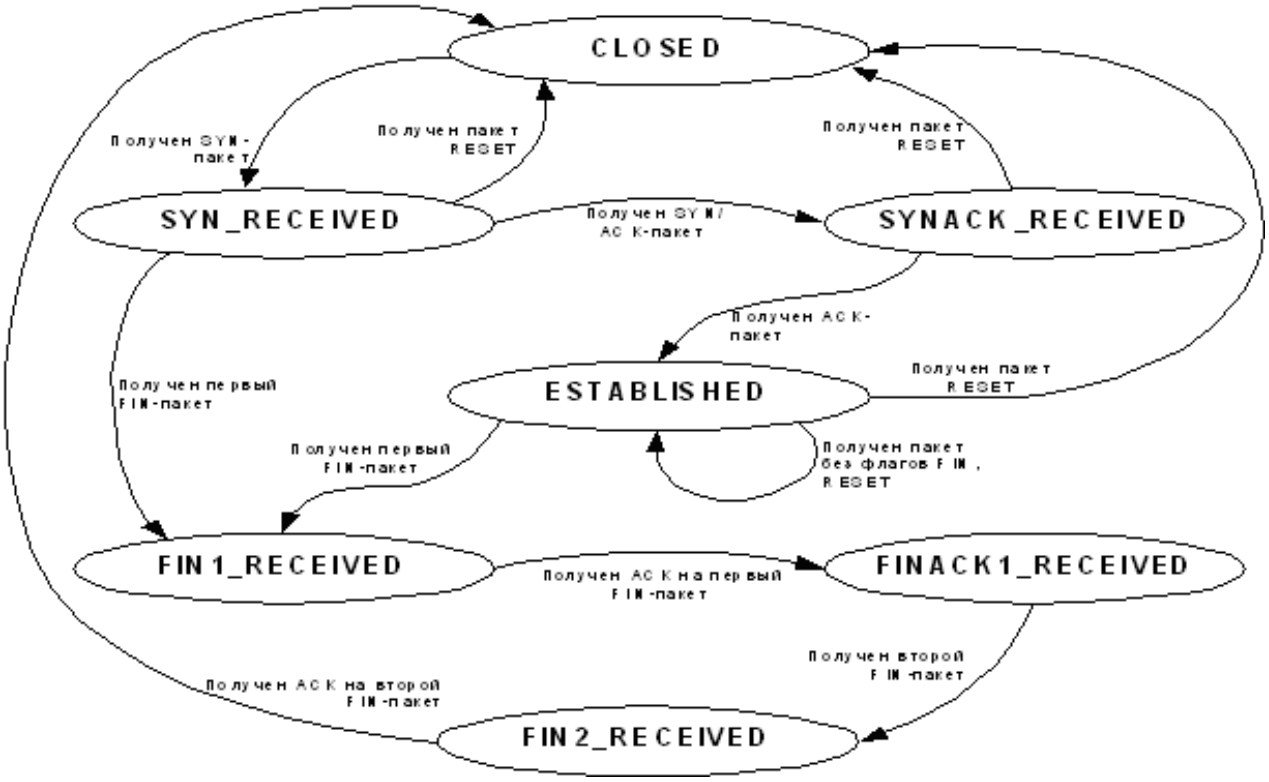

CLOSED
Получен SYN-пакет
Получен пакет RESET
Получен пакет RESET
SYN_RECEIVED
Получен SYN/ACK-пакет
SYNACK_RECEIVED
Получен ACK-пакет
Получен первый FIN-пакет
ESTABLISHED
Получен пакет RESET
Получен первый FIN-пакет
Получен пакет без флагов FIN, RESET
FIN1_RECEIVED
Получен ACK на первый FIN-пакет
FINACK1_RECEIVED
Получен второй FIN-пакет
Получен ACK на второй FIN-пакет
FIN2_RECEIVED

Рисунок 3.25.

10. Протокол SNMP

10.1. Определение и функции протокола

Протокол SNMP (Simple Network Management Protocol, простой протокол управления сетью) является протоколом Прикладного уровня, разработанный для выполнения двух задач:

- мониторинг сетевых устройств и сети в целом;
- управление сетевыми устройствами.

Протокол SNMP предоставляет возможность станциям управления считывать и изменять настройки шлюзов, маршрутизаторов, коммутаторов и прочих сетевых устройств.

10.2. Версии протокола SNMP

Опишем различия между версиями протокола SNMP и документы, определяющие эти версии. По состоянию на 2006 год единственной не устаревшей версией SNMP является SNMPv3, определённая в RFC 3411-3418.

SNMPv1

Первые RFC, описывающие стандарты SNMP, появились в 1988 году. Версия 1 подверглась критике за её посредственную модель безопасности на основе сообществ. В то время безопасность в Интернете не входила в круг первоочередных задач рабочих групп IETF.

SNMPv2

Версия 2, известная так же, как Party-based SNMPv2, или SNMPv2p, не получила широкого распространения из-за серьёзных разногласий по поводу инфраструктуры безопасности в стандарте. SNMPv2 улучшал версию 1 в области быстродействия, безопасности, конфиденциальности и взаимодействий «менеджер-менеджер». Он представил новый тип PDU Get-Bulk-Request, альтернативу Get-Next-Request для получения больших объёмов информации при помощи одного запроса. Тем не менее, новая система безопасности на основе сторон выглядела для многих как чересчур сложная и не была широко признана.

SNMPv2c

Community-based SNMPv2, или SNMPv2c, представил SNMPv2 без новой модели безопасности версии 2. Вместо неё предлагалось использовать старую модель безопасности версии 1 на основе сообществ. Соответствующее предложение RFC было принято только как черновик стандарта, однако стало де факто стандартом SNMPv2. Безопасность SNMP снова оказалась нерешённым вопросом.

SNMPv2u

User-based SNMPv2, или SNMPv2u, является компромиссом между незащищённостью SNMPv1 и чрезмерной сложностью SNMPv2p. Предложенная модель безопасности на основе пользователей была положена в основу SNMPv3.

SNMPv3

SNMPv3 наконец-то решил проблемы с безопасностью способом, который многие посчитали приемлемым. Версия 3 SNMP принята IETF как стандарт Интернета (IETF STD 62). Почти все предыдущие RFC признаны устаревшими. Документы, описывающие протокол SNMPv3, приведены ниже:

- *Общая информация*.
  RFC 3411. An Architecture for Describing SNMP Management Frameworks.
- *Обработка сообщений*.
  - Привязки к транспорту.
    RFC 3417. Transport Mappings for the SNMP.
  - Разбор и диспетчеризация сообщений.
    RFC 3412. Message Processing and Dispatching for the SNMP.
  - *Безопасность*.
    RFC 3414. User-based Security Model (USM) for SNMPv3.
- Обработка PDU.

  - Операции протокола.
    RFC 3416. Version 2 of the Protocol Operations for SNMP.
  - Приложения SNMP.
    RFC 3413. SNMP Applications.
  - Управление доступом.
    RFC 3415. View-based Access Control Model (VACM) for the SNMP.
- Модули MIB.
  RFC 3418. MIB for the SNMP.

10.3. Модель протокола SNMP

Общая модель

Модель приведена на рисунке 3.26. Основными взаимодействующими элементами протокола являются агенты (agent) и системы управления сетью (NMS, network management system). С точки зрения концепции «клиент-сервер» роль сервера выполняют агенты, то есть те самые устройства, для опроса состояния которых используется протокол SNMP. Соответственно, роль клиентов отводится системам управления – сетевым приложениям, необходимым для сбора информации о функционировании агентов. Взаимодействие агентов и систем управления осуществляется на основе сообщений протокола SNMP.

Агентами в SNMP являются программные модули, которые работают в управляемых устройствах. Агенты собирают информацию об управляемых устройствах, в которых они работают. Агент содержат всю информация об управляемом сетевом устройстве в базе управляющей информации (MIB, management information base). MIB представляет собой совокупность объектов, доступных для операций записи–чтения.

В любой управляемой сети может иметься одна или более NMS. NMS выполняют прикладные программы сетевого управления, которые представляют информацию управления конечному пользователю.

Структура базы MIB

На данный момент существует четыре типа информационной базы MIB:

1. Internet MIB – информационная база объектов для обеспечения диагностики ошибок и конфигураций. Включает в себя 171 объект (в том числе и объекты MIB I).
2. LAN manager MIB – база из 90 объектов – пароли, сессии, пользователи, общие ресурсы.
3. WINS MIB – база объектов, необходимых для управления и диагностики WINS-сервера (в серверах Microsoft Windows физически находится в файле WINSMIB.DLL).
4. DHCP MIB – база объектов, необходимых для управления и диагностики DHCP-сервера (в серверах Microsoft Windows физически находится в файле DHCPMIB.DLL).

Структуру MIB определяет документ, называемый SMI (Structure of Management Information, структура управляющей информации). Все MIB имеют иерархическую древовидную структуру. Все базы содержат десять корневых алиасов (ветвей), представленных на рисунке 3.27:

1. *System* – данная группа MIB II содержит в себе семь объектов, каждый из которых служит для хранения информации о системе (версия ОС, время работы и т.д.).
2. *Interfaces* – содержит 23 объекта, необходимых для ведения агентами статистики по сетевым интерфейсам управляемого устройства (количество интерфейсов, размер MTU, скорость передачи данных, физические адреса и т.д.).
3. *AT* – содержит 3 объекта, отвечающих за трансляцию адресов. Более не используется. Была включена в MIB I. Примером использования объектов AT может послужить простая ARP таблица соответствия физических (MAC) адресов сетевых карт IP адресам машин. В SNMP v2 эта информация была перенесена в MIB для соответствующих протоколов.
4. *IP* – содержит 42 объекта, в которых хранятся данные о проходящих IP пакетах.
5. *ICMP* – содержит 26 объектов со статистикой об ICMP-сообщениях.

6. *TCP* – содержит 19 объектов, хранящих статистику по протоколу TCP (соединения, открытые порты и т.д.).
7. *UDP* – содержит 6 объектов, хранящих статистику по протоколу UDP (входящие/исходящие датаграммы, порты, ошибки).
8. *EGP* – содержит 20 объектов – данные о трафике Exterior Gateway Protocol.
9. *Transmission* – зарезервирована для специфических задач.
10. *SNMP* – содержит 29 объектов, в которых хранится статистика по SNMP-протоколу (входящие/исходящие пакеты, ограничения пакетов по размеру, ошибки, данные об обработанных запросах и многое другое).

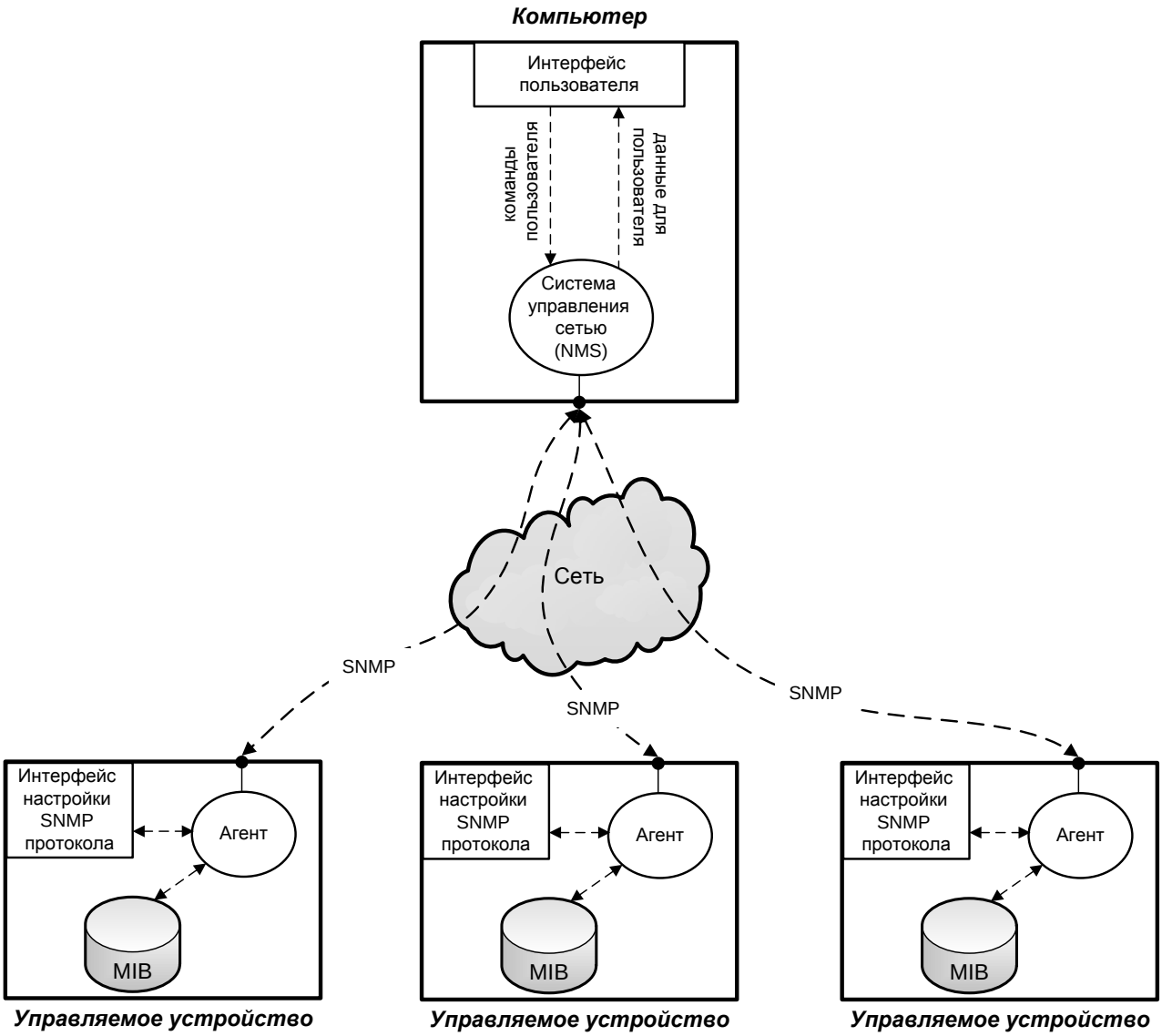


Рисунок 3.26.

Каждая из ветвей в свою очередь также представима в виде дерева. Например, к адресу администратора мы можем обратиться посредством такого пути: system.sysContact.0, ко времени работы системы system.sysUpTime.0. С другой стороны те же данные могут задаваться и в точечной нотации. Так system.sysUpTime.0 соответствует значение 1.3.0, так как system имеет индекс «1» в группах MIB II, а sysUpTime – «3» в иерархии группы system. Ноль в конце пути говорит о скалярном типе хранимых данных. В процессе работы SNMP-протокол использует точечную нотацию, то есть если менеджер запрашивает у агента содержимое параметра system.sysDescr.0, то в строке запроса ссылка на объект будет преобразована в «1.1.0».

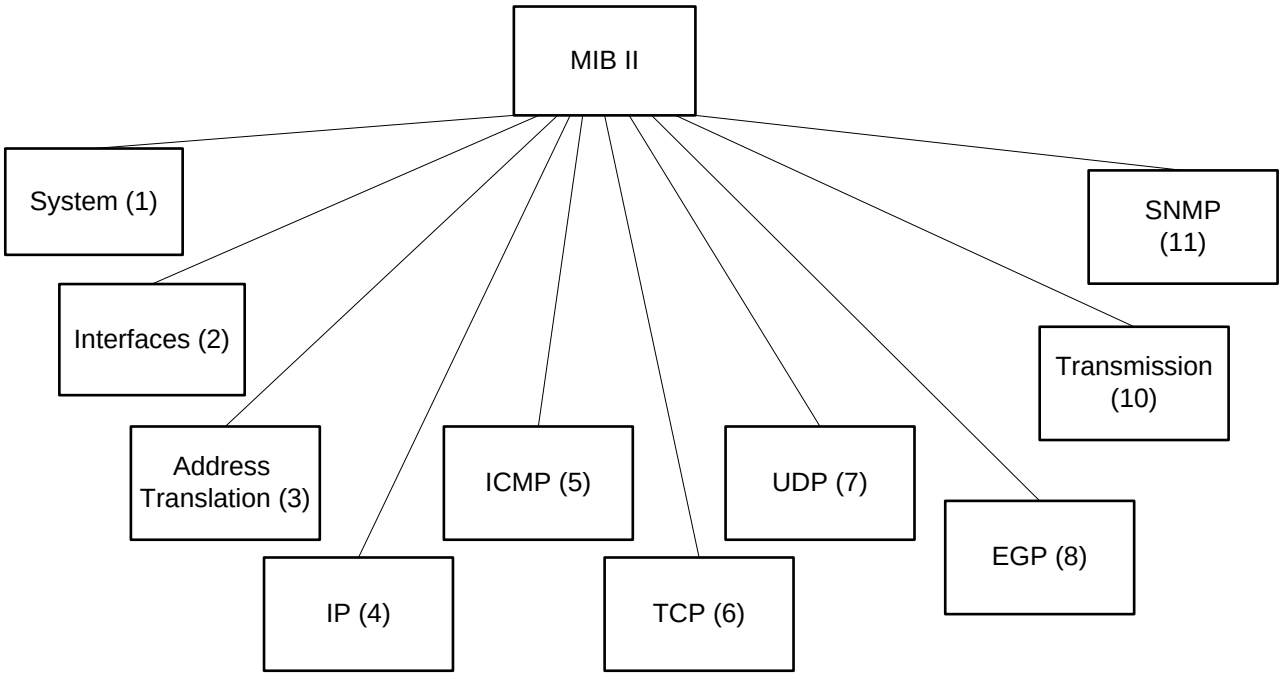


Рисунок 3.27.

Дерево MIB расширяемо благодаря экспериментальным и частным ветвям. Например, поставщики могут определять свои собственные ветви для включения реализаций своих изделий. В настоящее время вся работа по стандартизации ведется на экспериментальной ветви.

SMI определяет следующие типы данных MIB:

1. Network addresses (сетевые адреса) – символьные строки, представляющие адреса из конкретного стека протоколов. В настоящее время единственным примером сетевых адресов являются 32-битовые IP-адреса.
2. Counters (счетчики) – неотрицательные целые числа, которые монотонно увеличиваются до тех пор, пока не достигнут максимального значения, после чего они сбрасываются до нуля. Примером счетчика является общее число байтов, принятых интерфейсом.
3. Gauges (измерители) – неотрицательные целые числа, которые могут увеличиваться или уменьшаться, но фиксируются при достижении максимального значения. Примером типа «gauges» является длина очереди, состоящей из выходных пакетов.
4. Ticks (тики) – сотые доли секунды, прошедшие после какого-нибудь события. Примером типа «ticks» является время, прошедшее после вхождения интерфейса в свое текущее состояние.
5. Opaque (непрозрачный) – произвольное тип данных. Используется для передачи произвольных информационных последовательностей, находящихся вне пределов точного печатания данных, которое использует SMI.

Основные команды системы NMS

Если NMS хочет проконтролировать какое-либо из управляемых устройств, она делает это путем отправки ему сообщения с указанием об изменении значения одной из его переменных. В целом управляемые устройства отвечают на четыре типа команд (или инициируют их):

1. Reads.
   Для контролирования управляемых устройств NMS считывают переменные, поддерживаемые этими устройствами.
2. Writes.
   Для контролирования управляемых устройств NMS записывают переменные, накопленные в управляемых устройствах
3. Traversal operations.

NMS используют операции прослеживания, чтобы определить, какие переменные поддерживает управляемое устройство, а затем собрать информацию в таблицы переменных.

4. Traps.
   Управляемые устройства используют «ловушки» для асинхронных сообщений в NMS о некоторых событиях.

10.4. Протокол SNMPv3

Начиная с января 1998 года, выпущен набор документов, посвященных SNMPv3. В этой версии существенно расширена функциональность, разработана новая система безопасности.

Протокол обмена данными

Ниже на рисунке 3.28 приведена временная диаграмма, на которой в общем виде представлен протокол обмена SNMP-сообщениями. Для своей работы протокол SNMP использует транспортный протокол UDP, в основном 161 порт. Но для trap-сообщений используется 162 порт. Возможные команды протокола SNMPv3 приведены в таблице 3.7.

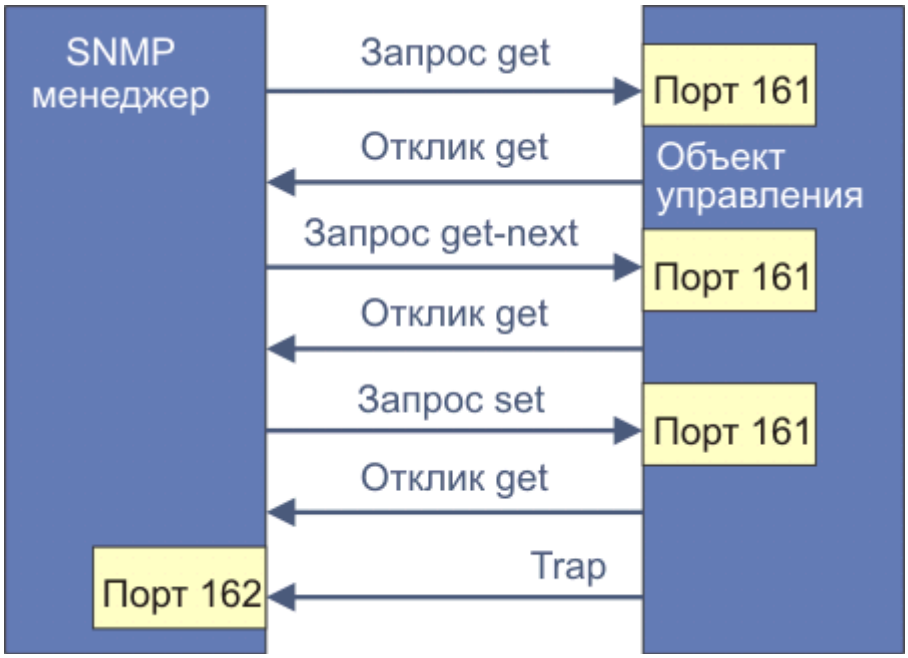


Рисунок 3.28.

Таблица 3.7 – Основные команды протокола SNMPv3.

Команда SNMP	Тип PDU	Назначение
GET-request	0	Получить значение указанной переменной или информацию о состоянии сетевого элемента.
GET-next-request	1	Получить значение переменной, не зная точного её имени (следующий логический идентификатор на дереве MIB).
SET-request	2	Присвоить переменной соответствующее значение. Используя для описания действия, которое должно быть выполнено.
GET-response	3	Отклик на GET-request, GET-next-request и SET-request. Содержит также информацию о состоянии (коды ошибок и другие данные).
TRAP	4	Отклик сетевого объекта на событие или на изменение состояния.
GetBulkRequest	5	Запрос пересылки больших объемов данных, например, таблиц.
InformRequest	6	Менеджер обращает внимание партнёра на определенную информацию в MIB.

SNMPv3-Trap	7	Отклик на событие (расширение по отношению к v1 и v2).
Report	8	Отчёт (функция пока не задана).

Формат SNMP-сообщения

Полное описание формата сообщения протокола SNMPv3 дано в документе RFC-3412 в разделе 6 «The SNMPv3 Message Format». Формат сообщения представлен на рисунке 3.29.

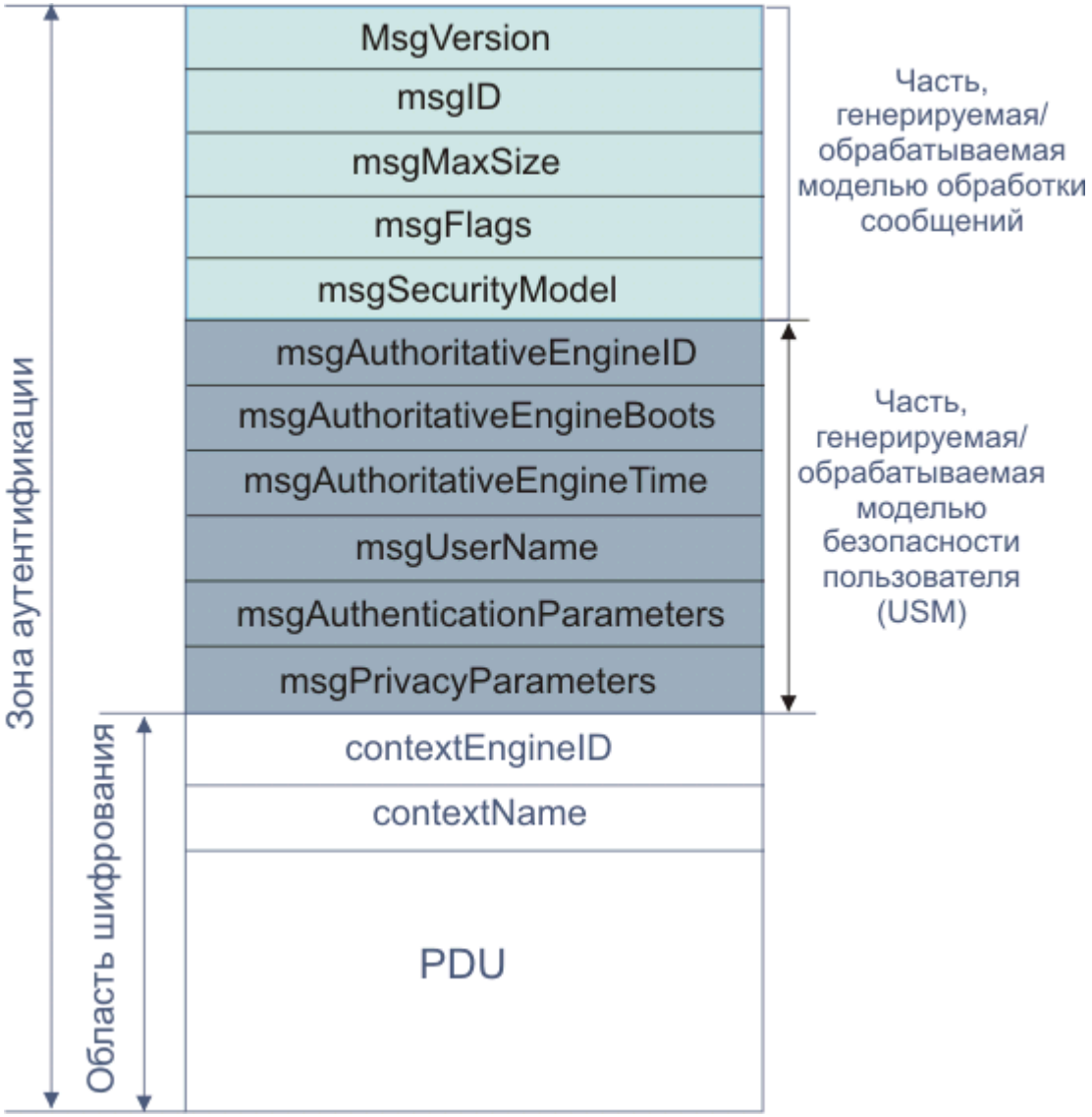


Рисунок 3.29.

SNMP-сообщение логически разделено на три части:

1. Часть, которая формируются отправителем в рамках модели обработки сообщений и обрабатываются получателем.
2. Часть, отвечающая за функции безопасности.
3. Собственно поле данных.

В данном разделе дано описание полей первой и третей частей. Описание полей безопасности дано в разделе 6.5.

Для реализации модели обработки сообщений используются следующие поля:

- **msgVersion**. Версия протокола. Для протокола SNMPv3 значение в поле равно 3.
- **msgID.** Уникальный идентификатор, используемый SNMP-сущностями для установления соответствия между запросом и откликом. Значение msgID лежит в диапазоне 0 - (2^{31}-1).

- **msgMaxSize.** Максимальный размер сообщения в октетах, поддерживаемый отправителем. Его значение лежит в диапазоне 484 - (2^{31}-1) и равно максимальному размеру сегмента, который может воспринять отправитель.
- **msgFlags.** Однобайтовая строка, содержащая три флага в младших битах:
  - *reportableFlag*. Если reportableFlag=1, то должно быть прислано сообщение с отчётом (команда Report). Флаг reportableFlag устанавливается отправителем во всех сообщениях запроса (команды Get, Set, Inform). Флаг устанавливается равным нулю в откликах и Trap-уведомлениях;
  - *privFlag*;
  - *authFlag*.

  Флаги privFlag и authFlag устанавливаются отправителем для индикации уровня безопасности для данного сообщения. Для privFlag=1 используется шифрование, а для authFlag=0 - аутентификация. Допустимы любые комбинации значений флагов кроме privFlag=1 AND authFlag=0 (шифрование без аутентификации).
- **msgSecurityModel**. Идентификатор со значением в диапазоне 0 - (2^{31}-1), который указывает на модель безопасности, используемую при формировании данного сообщения. Зарезервированы значения 1 – для SNMPv1, 2 и 3 – для SNMPv3.

10.5. Безопасность протокола SNMPv3

Модели безопасности протоколов SNMPv1-v3

Перечислим модели безопасности, применяющиеся в соответствующих версиях протокола SNMP:

1. *SNMPv1*
   SNMPv1 – Community-based Security Model
2. *SNMPv2*
   SNMPv2p – Party-based Security Model
   SNMPv2c – Community-based Security Model
   SNMPv2u – User-based Security Model
3. *SNMPv3*
   SNMPv3 – USM User-based Security Model

Модель безопасности на основе сообществ

Модель безопасности на основе сообществ (Community-based Security Model) была первой, самой простой и самой небезопасной. Она подразумевает лишь аутентификацию на основе «строки сообщества», фактически, пароля, передаваемого по сети в теле сообщения SNMP в открытом тексте. Эта модель безопасности не в состоянии бороться ни с одной из угроз информационной безопасности. Тем не менее, она часто используется до сих пор в связи со своей простотой, а также благодаря наличию внешних, не связанных с SNMP систем безопасности, например, межсетевых экранов.

Модель безопасности на основе сторон

Модель безопасности на основе сторон (Party-based Security Model) подразумевает введение понятие стороны. Сторона — это виртуальное окружение исполнения, в котором набор допустимых операций ограничен административно. Сущность SNMP при обработке сообщения действует как сторона, поэтому ограничена операциями, определёнными для этой стороны. Сторона определяется следующими параметрами:

1. Уникальный идентификатор стороны.
2. Логический сетевой адрес (адрес транспортного протокола).
3. Протокол аутентификации и параметры, требующиеся для аутентификации всех сообщений стороны.
4. Протокол шифрования и параметры, требующиеся для шифрования всех сообщений стороны.

Могут использоваться различные алгоритмы для протоколов аутентификации и шифрования. Обычно в качестве алгоритма для протокола аутентификации используют хэш-функцию Message Digest 5 (MD5), а для протокола шифрования — алгоритм Data Encryption Standard (DES) в режиме Cipher Block Chaining (CBC). При использовании соответствующих протоколов аутентификации и шифрования модель успешно справляется

с большинством угроз безопасности. Данная модель безопасности не была широко принята, поскольку показалась многим слишком сложной и запутанной.

Модель безопасности на основе пользователей
Модель безопасности на основе пользователей (User-based Security Model) вводит понятие пользователя, от имени которого действует сущность SNMP. Этот пользователь характеризуется именем пользователя, используемыми протоколами аутентификации и шифрования, а также закрытым ключом аутентификации и шифрования. Аутентификация и шифрование являются необязательными. Модель безопасности во многом похожа на модель на основе сторон, но она упрощает идентификацию пользователей, распределение ключей и протокольные операции.

Модель безопасности USM
Модель безопасности USM (User-Based Security Model) использует концепцию авторизованного сервера (authoritative Engene). При любой передаче сообщения одна или две сущности, передатчик или приемник, рассматриваются в качестве авторизованного SNMP-сервера. Это делается согласно следующим правилам:

1. Когда SNMP-сообщение содержит поле данных, которое предполагает отклик (например, Get, GetNext, GetBulk, Set или Inform), получатель такого сообщения считается авторизованным.
2. Когда SNMP-сообщение содержит поле данных, которое не предполагает посылку отклика (например, SNMPv2-Trap, Response или Report), тогда отправитель такого сообщения считается авторизованным.

Таким образом, сообщения, посланные генератором команд, и сообщения Inform, посланные отправителем уведомлений, получатель является авторизованным. Для сообщений, посланных обработчиком команд или отправителем уведомлений Trap, отправитель является авторизованным. Такой подход имеет две цели:

1. Своевременность сообщения определяется с учетом показания часов авторизованного сервера. Когда авторизованный сервер посылает сообщение (Trap, Response, Report), оно содержит текущее показание часов, так что неавторизованный получатель может синхронизовать свои часы. Когда неавторизованный сервер посылает сообщение (Get, GetNext, GetBulk, Set, Inform), он помещает туда текущую оценку показания часов места назначения, позволяя получателю оценить своевременность прихода сообщения.
2. Процесс локализации ключа, описанный ниже, устанавливает единственного принципала, который может владеть ключом. Ключи могут храниться только в авторизованном сервере, исключая хранение нескольких копий ключа в разных местах.

Когда исходящее сообщение передается процессором сообщений в USM, USM заполняет поля параметров безопасности в заголовке сообщения. Когда входное сообщение передается обработчиком сообщений в USM, обрабатываются значения параметров безопасности, содержащихся в заголовке сообщения. В параметрах безопасности содержатся следующие поля:

- **msgAuthoritativeEngeneID.** Идентификатор авторизованного сервера, участвующего в обмене. Это значение идентификатора отправителя для Trap, Response или Report или адресата для Get, GetNext, GetBulk, Set или Inform.
- **msgAuthoritativeEngeneBoots.** snmpEngeneBoots авторизованного сервера, участвующего в обмене. Объект snmpEngeneBoots содержит целочисленные значения в диапазоне 0 - (2^{31}-1). Это поле содержит число, показывающее сколько раз SNMP-сервер был перезагружен с момента конфигурирования.
- **msgAuthoritativeEngeneTime.** Время работы авторизованного сервера, участвующего в обмене. Значение этого поля лежит в диапазоне 0 - (2^{31}-1). Это поле характеризует число секунд, которое прошло с момента последней перезагрузки сервера. Каждый авторизованный сервер должен инкрементировать это поле один раз в секунду.
- **msgUserName.** Имя пользователя, который послал сообщение.

- **msgAuthenticationParameters.** Поле содержит ноль, если при обмене не используется аутентификация. В противном случае данное поле содержит аутентификационный параметр.
- **msgPrivacyParameters.** Поле содержит ноль, если не требуется соблюдения конфиденциальности. В противном случае данное поле содержит параметр безопасности. В действующей модели USM используется алгоритм шифрования DES.

Механизм аутентификации в SNMPv3 предполагает, что полученное сообщение действительно послано пользователем, имя которого содержится в заголовке сообщения, и это имя не было модифицировано во время доставки сообщения. Для реализации аутентификации каждый из пользователей, участвующих в обмене должен иметь секретный ключ аутентификации, общий для всех участников (определяется на фазе конфигурации системы). В посылаемое сообщение отправитель должен включить код, который является функцией содержимого сообщения и секретного ключа. Одним из принципов USM является проверка своевременности сообщения, что делает маловероятной атаку с использованием копий сообщения.

Модель управления доступом

Система конфигурирования агентов позволяет обеспечить разные уровни доступа к базе MIB для различных SNMP-менеджеров. Это делается путем ограничения доступа некоторым агентам к определенным частям MIB, а также с помощью ограничения перечня допустимых операций для заданной части MIB. Такая схема управления доступом называется VACM (View-Based Access Control Model). В процессе управления доступом анализируется контекст (vacmContextTable), а также специализированные таблицы vacmSecurityToGroupTable, vacmTreeFamilyTable и vacmAccessTable.

IV. Технологии обеспечения безопасности передачи данных в сетях TCP/IP

1. Модель сетевой безопасности

Обобщенную модель сетевой безопасности можно представить в следующем виде (рисунок 4.1).

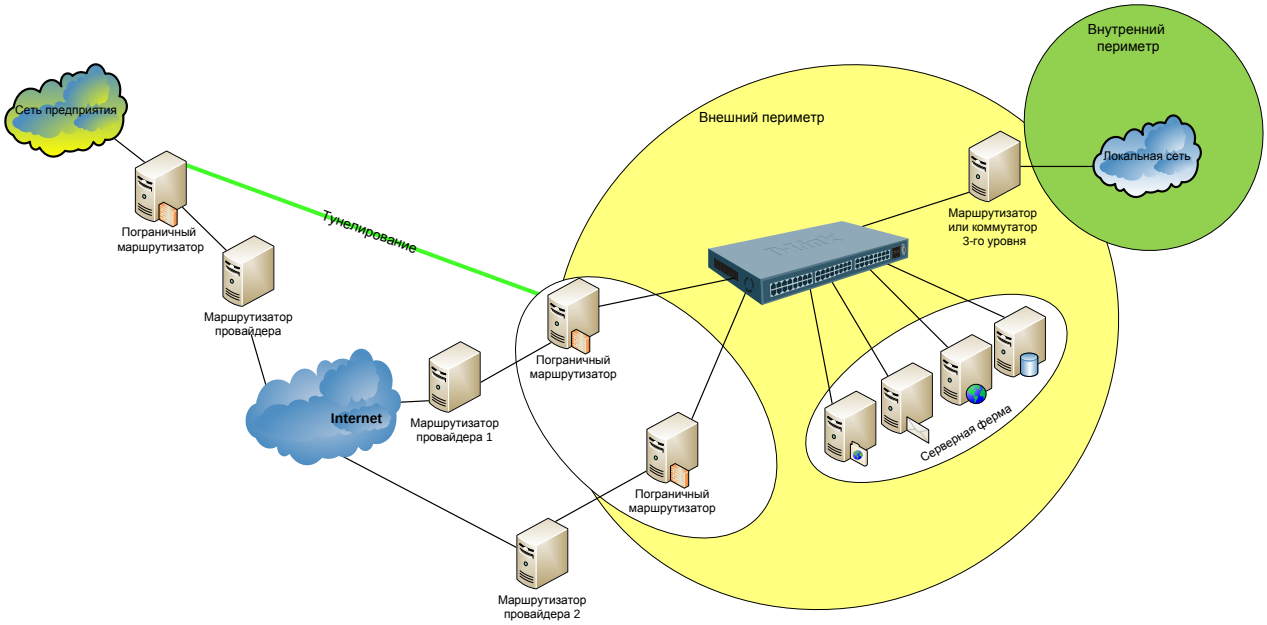


Рисунок 4.1. Обобщенная архитектура сетевой безопасной сети.

Грамотно построенная сеть (с точки зрения безопасности) должна содержать как минимум два периметра:

- ✓ Внутренний. В данном периметре располагаются только компьютеры пользователей, которые входят в сеть незарегистрированных IP-адресов. Также данная сеть может содержать беспроводные точки доступа для подключения мобильных пользователей. Во внутреннем периметре применяются механизмы внутрисетевой безопасности.
- ✓ Внешний периметр. Обычно содержит сервера предприятия (серверная ферма) и пограничные маршрутизаторы, обеспечивающие доступ во внешние сети. Все компьютеры во внешнем периметре обычно включаются в IP-сеть зарегистрированных адресов. В данном периметре применяются механизмы межсетевой безопасности, к которым относятся:
  - межсетевые экраны (МЭ);
  - системы трансляции адресов и портов;
  - системы обнаружения атак и вторжений (СОАВ);
  - системы туннелирования.

Между периметрами устанавливается маршрутизатор или коммутатор L3, на котором также могут применяться технологии межсетевой безопасности. Все маршрутизаторы могут как программным, так и аппаратными.

Данная концепция является расширяемой. Если необходимо повысить уровень безопасности сети, то можно увеличить количество внутренних периметров. Следующим этапом развития модели является добавление ещё одного внутреннего периметра. В этом случае во внешнем периметре устанавливаются сервера-ловушки, в первом внутреннем периметре – сервера предприятия и только во втором внутреннем периметре – компьютеры пользователей.

2. Межсетевые экраны

2.1. Архитектура межсетевого экрана

Межсетевой экран (брандмауэр, файерволл) – это комплекс программных или аппаратных средств, который позволяет реализовать набор правил, определяющих условия прохождения пакетов между сетями. Межсетевой экран включает следующие средства (рисунок 4.2):

- фильтр пакетов;
- фильтр состояний;
- транслятор адресов;
- транспортный шлюз;
- шлюз приложений (прокси-сервер).

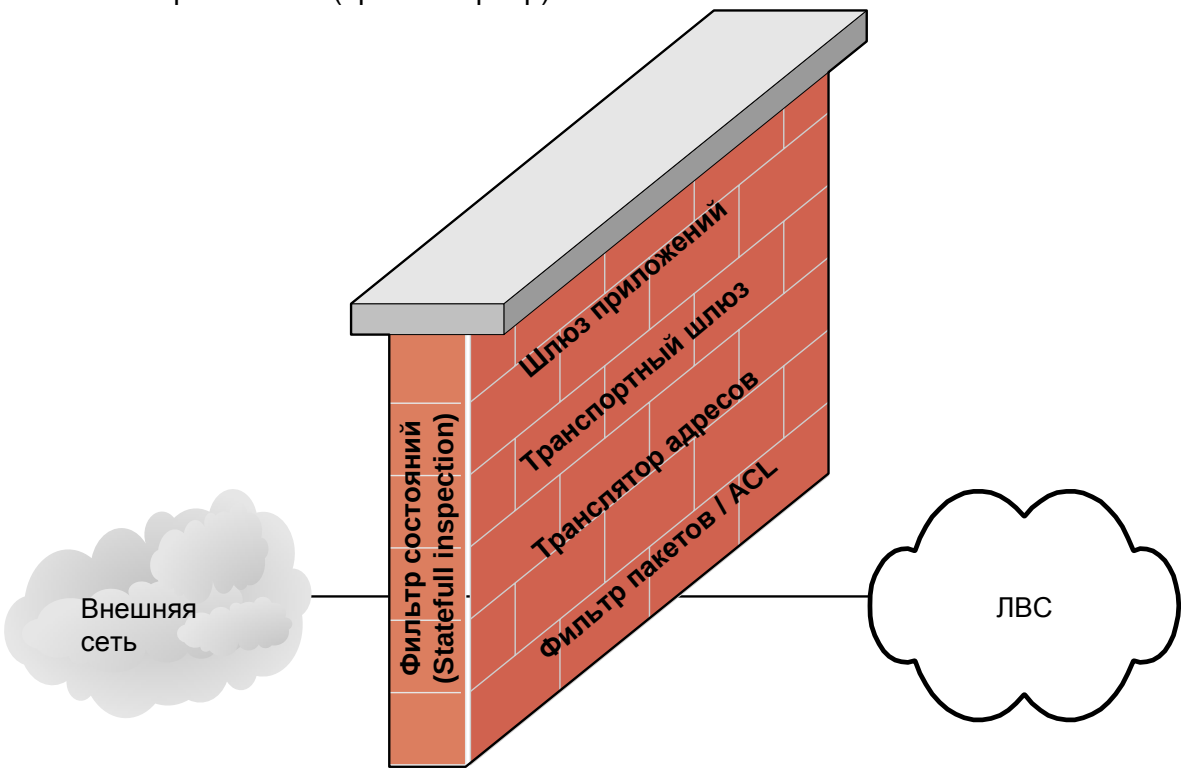


Рисунок 4.2. Упрощенная архитектура межсетевого экрана.

2.2. Фильтр пакетов

ФП разрешает или запрещает прохождение пакета в соответствии с заданными администратором правилами. Эти правила опираются на информацию, имеющуюся в заголовке каждого пакета и в сетевой системе:

- ✓ IP-адрес источника и назначения;
- ✓ протокол Транспортного уровня;
- ✓ порт источника и назначения;
- ✓ Флаги;
- ✓ Тип сообщения ICMP;
- ✓ Входящий и исходящий сетевой интерфейс.

Таким образом, для описания правил прохождения пакетов составляются таблицы типа:

Действие	Тип пакета	Адрес источника	Порт источника	Адрес назначения	Порт назначения	Флаги	Входящий интерфейс	Исходящий интерфейс
DROP	TCP	194.186.1.2	80	-	-	-	-	
ACCEPT	UDP	83.142.162.49	53	-	-	-	-	
ACCEPT	ICMP	-	-	-	-	-	-	

DROP	-	-	-	-	-	-	eth1	eth0

Фильтры работают быстро, поскольку они просто просматривают информацию о пакете при принятии решения. Но фильтры пакетов имеют один существенных недостаток – они не в состоянии отслеживать конкретный сетевой сеанс и не в состоянии анализировать содержимое сообщения конкретного приложения, которое в свою очередь может содержать зловредный код.

2.3. Фильтр инспекции состояний (statefull фильтры)

Технология инспекции состояний (statefull inspection) используется многими разработчиками, но, поскольку наименование запатентовано компанией Check Point, они вынуждены присваивать ему различные наименования (expert inspection, smart filtering, adaptive screening, multilevel inspection и др.).

Данная технология является дальнейшим развитием фильтра пакетов и включает следующую дополнительную функцию – поддержка таблицы состояний TCP-соединений и контроль сессий на её основе.

Жизненый цикл TCP-соединения имеет несколько фаз, отраженных на рисунке 4.3. Таблица состояний (netstat) содержит информацию о соединение и состояние, в котором находится данное соединение. На основе этой информации происходит фильтрация пакетов. Например, если не был произведн запрос, то пакет-ответа не будет пропушен в локальную сеть.

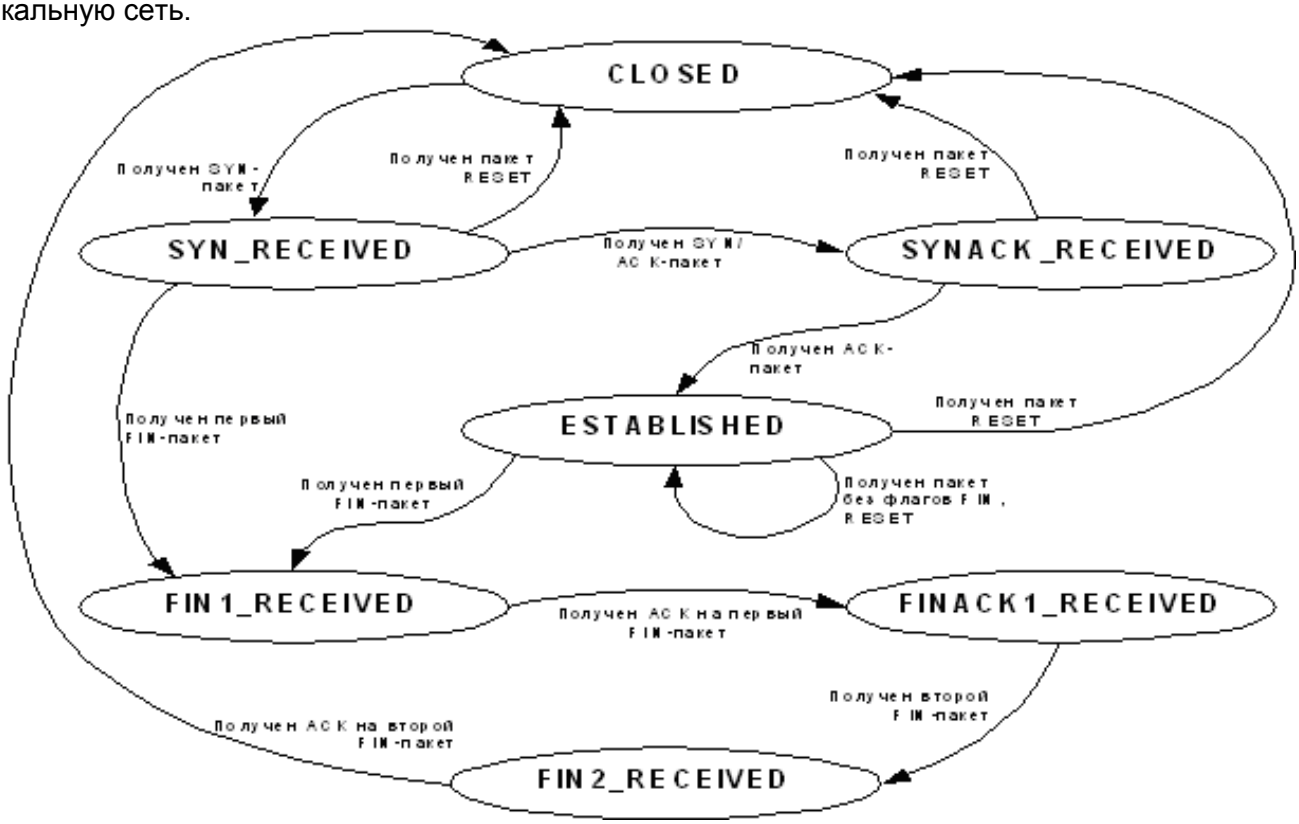


Рисунок 4.3. Диаграмма состояний протокола TCP.

2.4. Транслятор адресов

Трансляция адресов – технология, позволяющая изменять такие параметры IP-пакета, как адрес и порт источника пакета, адрес и порт назначения. При этом ведётся таблица измененных данных, и при получении обратного пакета происходит обратная трансляция адреса, если она нужна. Существуют 2 причины, по которым имеет смысл осуществлять трансляцию адресов:

- ✓ предоставление компьютерам локальной сети полноценного доступа в Интернет при условии, что имеющееся число внешних адресов или подключений к провайдеру меньше, чем число компьютеров, которым необходим доступ в Интернет. Стоит

учесть, что некоторые протоколы не поддерживают трансляцию адресов (например, PPTP), либо использование трансляции накладывает ограничения на использование служб (например, пассивный режим FTP был разработан для решения проблемы NAT). Для обхода таких проблем маршрутизатор должен иметь возможность разбирать пакеты, вносить изменения и собирать их снова;

- ✓ скрытие внутренней структуры локальной сети. За счёт выполнения трансляции адресов запросы компьютеров локальной сети выглядят так, будто выполняются с одного и того же компьютера. Однако на самом деле это могут быть запросы от разных машин. Также можно с помощью трансляции адреса совместно с портом осуществлять скрытие серверной инфраструктуры.

Существуют 2 способа трансляции адресов:

- ✓ Network Address Translation (NAT) – замена адрес источника на адрес маршрутизатора. При этом порт остаётся неизменным;
- ✓ Static Address Translation (SAT) – замена адрес источника или приёмника на некоторый адрес. Возможна одновременная замена порта. SAT подразделяется на 2 типа:
  - Source SAT – трансляция адреса источника
  - Destination SAT – трансляция адреса назначения

Типичный пример применения NAT

Некоторой организации необходимо предоставить сотрудникам доступ в интернет. Приобретается выделенная линия до провайдера, настраивается трансляция адресов на маршрутизаторе из частных во внешний

Типичный пример применения DSAT

Необходимо перенаправить соединение по порту 80/tcp (HTTP-трафик) на кэширующий сервер

Типичный пример применения SSAT

Организация приобрела не один, а 2 канала, но маршрутизатор один. Тогда настраивается SSAT, преобразование производится в адрес интерфейса, подключенного к нужному каналу. Второй канал может использоваться, например, для подключения серверной фермы.

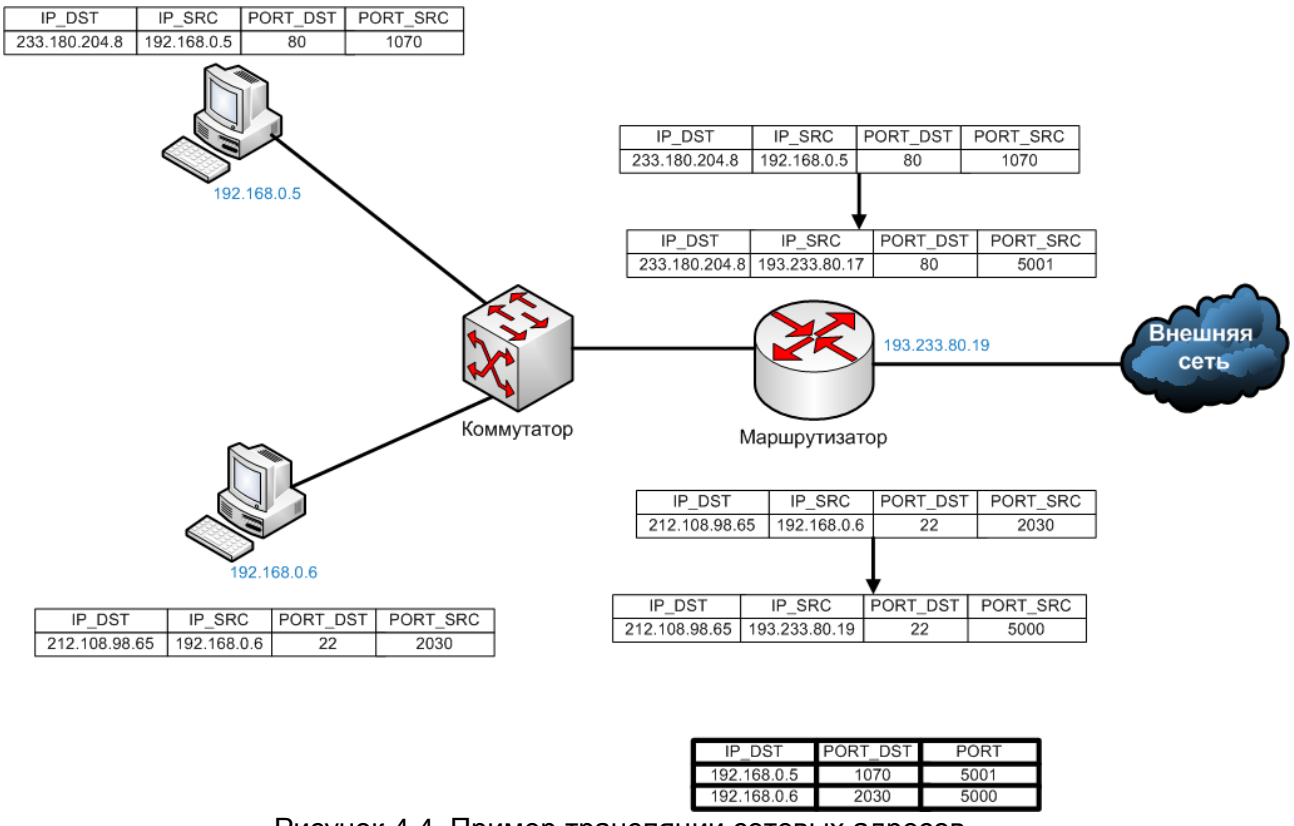


Рисунок 4.4. Пример трансляции сетевых адресов.

ОС Linux содержит системную службу netfilter, которая реализует функции фильтра пакетов (в том числе statefull), трансляции адресов, прозрачного прокси и журналирования трафика. Для управления этой службой имеется утилита iptables.

3. Виртуальные частные сети (Virtual Private Network, VPN)

3.1. Введение

Виртуальные частные сети, или защищенные виртуальные сети (Virtual Private Network, VPN), – это подключение, установленное по существующей общедоступной инфраструктуре и использующее шифрование и аутентификацию для обеспечения безопасности содержания передаваемых пакетов. Виртуальная частная сеть создает виртуальный сегмент между любыми двумя точками доступа к сети. Она может проходить через общедоступную инфраструктуру локальной вычислительной сети, подключения к глобальной сети (Wide area Network, WAN) или Интернет.

Рассмотрим VPN, организующие безопасные каналы передачи данных, использующие общедоступную инфраструктуру или Интернет. Все VPN по конфигурации можно подразделить на три основных типа:

1. узел-узел (host-to-host);
2. узел-шлюз (host-to-gateway);
3. шлюз-шлюз (gateway-to-gateway).

Для организации канала связи, проходящего через Интернет, можно использовать VPN любого типа. Основной концепцией VPN является защита шифрованием канала связи на различных уровнях модели TCP/IP, а именно:

- ✓ прикладном (5-й уровень);
- ✓ транспортном (4-й уровень);
- ✓ сетевом (3-й уровень);
- ✓ канальном (2-й уровень).

Схема расположения протоколов VPN по уровням модели приведена в таблице 4.1.
На прикладном уровне шифрование можно применять при помощи программ, подобных пакету Pretty Good Privacy (PGP), или через каналы типа Secure Shell (SSH). Такие программы работают на участке сети от узла до узла, что означает, что они предлагают защиту только душ содержимого (payload) пакета, а не всего пакета в целом. Исключение составляет протокол SSH, который может использовать режим port-forwarding для создания туннеля.

Таблица 4.1 – Схема расположения протоколов VPN по уровням модели.

Уровни TCP/IP	Основные протоколы
Прикладной	PGP, S/MIME SSH, Kerberos, RADIUS
Транспортный	SSL, TSL SOCKS v5
Уровень межсетевого взаимодействия (сетевой)	IPSec (AH, ESP)
Уровень сетевых интерфейсов (канальный)	L2TP, PPTP

На транспортном уровне для защиты содержимого пакетов конкретного сеанса между двумя сторонами можно использовать протоколы, аналогичные протоколу защищенных сокетов (Secure Sockets Layer, SSL). Обычно такой метод используется при соединениях, установленных посредством Web-браузера. При этом также защищается только содержательная часть передаваемых пакетов, а IP-датаграммы, которые несут эту информацию, доступны для просмотра.
На сетевом уровне протоколы, подобные IPSec, не только зашифровывают содержательную часть пакета (полезную нагрузку), но и зашифровывают информацию заголовков протоколов TCP/IP.

На канальном уровне протокол туннелирования (Layer 2 Tunneling Protocol, L2TP) является расширением протокола соединения типа «точка-точка» (Point-to-Point Protocol, PPP), который допускает шифрование пакетов, посланных по PPP-протоколу на канальном уровне передачи данных.

Несмотря на то, что эти технологии шифрования применяются на разных уровнях, они все могут быть частью VPN. Необходимо отметить, что некоторые из этих технологий не могут обрабатывать все режимы работы VPN без дополнительной помощи со стороны других приложений или протоколов.

3.2. Обзор протоколов PPTP и L2TP

На канальном уровне существуют два протокола для реализации VPN: протокол туннелирования типа «точка-точка» (Point-to-point Tunneling Protocol, PPTP) и протокол туннелирования второго уровня (Layer Two Tunneling Protocol, L2TP).

Протокол PPTP

Протокол PPTP является дальнейшим развитием протокола PPP, который распространился в связи с появлением модемного доступа к сети Интернет. Протокол PPTP был одним из первых протоколов для создания VPN через коммутируемое соединение и был разработан консорциумом таких производителей, как Microsoft, US Robotics, Ascend и 3Com. Для шифрования протокола PPP был использован протокол двухточечного шифрования Microsoft (Microsoft Point-to-Point Encryption, MPPE), который использует алгоритм RC4. Однако большинство проблем в области безопасности было связано с ненадежностью используемого метода аутентификации — протокола Microsoft аутентификации с предварительным согласованием вызова (Microsoft Challenge/Reply HandShake Protocol, MSCHAP). Для устранения недостатков был выпущен протокол MSCHAP версии 2. Протокол PPTP использует все связанные протоколы, которые подобны протоколу MSCHAP, протоколу аутентификации пароля (Password Authentication Protocol, PAP), протоколу аутентификации с предварительным согласованием вызова (Challenge Handshake Authentication Protocol, CHAP), расширенному протоколу аутентификации (Extensible Authentication Protocol, EAP).

Протокол PPTP использует два канала, работающих совместно. Первый — канал управления (порт 1723/tcp). Этот канал посылает в обе стороны все команды, которые управляют сеансом подключения. Второй — инкапсулированный канал передачи данных, являющийся вариантом протокола общей инкапсуляции для маршрутизации (Generic Routing Encapsulation, GRE, IP-протокол 47). Преимуществом туннеля протокола общей инкапсуляции для маршрутизации является то, что он может инкапсулировать и передавать протоколы, отличающиеся от протокола IP. Первоначально клиент устанавливает соединение с провайдером с помощью PPP, а затем устанавливает TCP/IP-соединение с PPTP-сервером через интернет.

В отличие от L2TP PPTP не требует развёртывания инфраструктуры для сертификатов, а использует парольную аутентификацию, что упрощает развёртывание VPN, но в некоторых случаях снижает безопасность. PPTP-сервер имеет ограничение по количеству одновременных подключений. PPTP не использует IPSec, поэтому проблем с трансляцией адресов не возникает (не требуется механизм NAT Traversal).
Протокол PPTP реализован во многих аппаратных устройствах. Протокол PPTP при инициализации связи использует протокол PPP, по этому может оказаться уязвимым к атакам типа spoofing и «человек посередине».

Протокол L2TP

Протокол L2TP (Layer 2 Tunneling Protocol) – открытый стандарт IETF, который решает многие проблемы PPTP, и определен в документе RFC 2661. Протокол L2TP фактически является гибридом двух предыдущих протоколов туннелирования: протокола пересылки второго уровня (Layer Two Forwarding, L2F) компании Cisco и протокола PPTP.

Протокол L2TP, подобно протоколу PPTP, использует при аутентификации пользователя возможности протокола PPP (протоколы MSCHAP, CHAP, EAP, PAP и т.д.). Аналогично протоколу PPTP протокол L2TP использует два канала связи: сообщения управления и сообщения туннеля для передачи данных. Первый бит заголовка протокола PPTP служит для опознания этих типов сообщений (1 — для сообщений управления, 0 — для сооб-

щений данных). Сообщениям управления дается более высокий приоритет по отношению к сообщениям данных, чтобы гарантировать, что важная информация администрирования сеанса будет передана настолько быстро, насколько это возможно.

Подключение канала управления устанавливается для туннеля, который затем сопровождается инициированием сеанса протокола L2TP. После завершения инициирования обоих подключений информация в виде кадров протокола PPP начинает передаваться по туннелю.

Формирование защищенного канала происходит в три этапа:
1. Установление соединения клиента с сервером удаленного доступа.
2. Аутентификация пользователя.
3. конфигурирование защищенного туннеля.

Для установления соединения с сервером удаленного доступа (сетевой сервер L2TP) удаленный пользователь связывается по протоколу PPP с концентратором доступа L2TP (Local Access Concentrator, LAC), обычно функционирующем на сервере провайдера. Концентратор доступа может выполнить аутентификацию пользователя от имени провайдера. По заданному имени получателя концентратор доступа определяет адрес сетевого сервера L2TP (L2TP Network Server, LNS), который защищает сеть с заданным адресом. Между концентратором доступа и сервером L2TP устанавливается соединение. Далее производится аутентификация пользователя сервером L2TP. В случае успешной аутентификации устанавливается защищенный туннель между концентратором доступа и сервером L2TP. С помощью управляющих сообщений производится настройка параметров туннеля, причем в одном туннеле может быть несколько сеансов пользователя. При использовании IPSec пакеты L2TP инкапсулируются в UDP-па-кеты, которые передаются концентратором доступа и сервером L2TP через IPSec-туннель (порт 1701/tcp). В большинстве случаев клиент сам выступает в роли LAC.

L2TP использует сертификаты для аутентификации, за счёт чего упрощается администрирование сервера с большим количеством клиентов и повышается безопасность. В отличие от PPTP имеется возможность создания нескольких виртуальных сетей через один туннель. В случае, когда L2TP использует IPSec, при необходимости трансляции адресов требуется NAT Traversal на LNS.

Сравнение PPTP/L2TP
1. В PPTP шифрование данных начинается после установления PPP-соединения. В L2TP/IPSec шифрование начинается до установления такого соединения.
2. В PPTP для шифрования применяется протокол MPPE, основанный на алгоритме RSA RC4, который обеспечивает возможность применения 40-, 56- и 128-битных ключей. В L2TP/IPSec возможно применение любых алгоритмов шифрования, поддерживаемых IPSec.
3. В PPTP используется аутентификация PPP (сервер аутентифицирует клиента по паре логин-пароль). В L2TP/IPSec наряду с учётной парой применяется и аутентификация по сертификатам.

3.3. Протокол IPSEC

Несмотря на то, что протокол IP стал наиболее используемым протоколом связи во всем мире и базовой технологией Интернета, он имеет множество существенных недостатков. Среди них необходимо отметить ограниченность адресного пространства и недостатки внутренней безопасности. Главной причиной этих недостатков является то, что IP-протокол изначально не был предназначен для массового использования.

В качестве развития IP-протокола была разработана его новая версия IPv6 (IP-протокол версии 6), в которой пытались решить проблемы предшествующих версий. Поскольку принятие новой версии IP-протокола затруднительно, что вызвано постоянным ростом Интернета и громадным разнообразием установленной аппаратуры и программного обеспечения, то меры безопасности, включенные в IPv6, были перенесены в текущую

версию IPv4 в виде дополнительного комплекта протоколов. Этот набор протоколов назвали комплектом протоколов IPSec (IPSec Protocol Suite).

Подключение по протоколу IPSec имеет два основных режима: транспортный (transport) и туннельный (tunnel).Транспортный режим — это форма связи типа узел-узел, где применяется шифрование только содержательной части пакета. Из-за двухточечного характера связи соответствующее программное обеспечение необходимо загрузить (установить) на все связывающиеся между собой узлы сети, что представляет собой достаточно серьезную проблему. Этот режим VPN удобно использовать для зашифрованной связи между узлами одной сети.

Режим туннелирования применяется при создании большинства VPN; потому что он шифрует весь оригинальный пакет. Режим туннелирования может применяться для организации связи типа узел-узел, узел-шлюз или шлюз-шлюз. При организации связи типа шлюз-шлюз значительно упрощается связь между сетями, не требуется установка специального ПО на узлах сети.

Первая цель семейства протоколов IPSec состоит в обеспечении конфиденциальности, целостности и аутентификации информации, передаваемой посредством IP-протокола. Это достигается с помощью протокола обмена интернет-ключами (Internet Key Exchange, IKE), протокола ESP и протокола AH. Комбинация этих трех протоколов обеспечивает безопасный обмен информацией.

Второй целью семейства протоколов IPSec является предоставление разработчикам ПО набора стандартов. Установление безопасного соединения начинается с формирования ассоциации обеспечения безопасности (Security Association, SA) между двумя сообщающимися сторонами.

Ассоциация обеспечения безопасности
Основа ассоциации обеспечения безопасности заключается в соглашении двух сторон о том, как они могут безопасно передавать свою информацию. В процессе соглашения стороны оговаривают детали защищенного обмена. Результатом такого соглашения и является ассоциация обеспечения безопасности. Каждому сеансу связи сопоставляются две ассоциации — по одной на каждого партнера связи.

Положительными чертами протокола IPSec являются открытость его стандарта для поддержки множества протоколов и режимов связи, а также поддержка различных алгоритмов шифрования и различных хэш-функций. Прежде чем договариваться об ассоциации обеспечения безопасности, необходимо локальное конфигурирование элементов протокола IPSec, которые данный партнер собирается поддерживать. Эти параметры настройки хранятся в базе данных политики безопасности (Security Policy Database, SPD).

После согласования ассоциация обеспечения безопасности содержится в базе данных ассоциации обеспечения безопасности (Security Association Database, SAD). Это необходимо, поскольку узел сети может инициализировать несколько сеансов, каждому из которых может соответствовать своя SA.

Поскольку для каждого сетевого устройства доступно множество сеансов протокола IPSec, для правильного функционирования процесса необходимо, чтобы каждый сеанс согласования SA имел свой собственный уникальный идентификатор. Этот идентификатор составляется из уникального индекса параметра обеспечения безопасности (Security Parameter Index, SPI), который определяет, какая запись БД SA соответствует рассматриваемому подключению. Кроме того, учитывается адрес назначения и идентификатор используемого протокола (ESP или AH).

Протокол обмена интернет-ключами
Протокол обмена интернет-ключами предназначен для аутентификации и согласования параметров обмена протокола IPSec. Протокол IKE представляет собой комбинацию двух

протоколов: протокола управления ассоциациями и протокола управления ключами обеспечения безопасности в сети Интернет (Internet Security Association and Key Management Protocol, ISAKMP), называемых фазами установления. Управление ключами можно выполнять вручную или используя альтернативы протокола IKE, такие как безопасная служба доменных имен (Secure DNS), Photuris или простой протокол обмена интернет-ключами (Simple Key Internet Protocol, SKIP).

Первая фаза протокола IKE. На первой фазе протокола IKE удаленный пользователь начинает сеанс со шлюзовым устройством VPN. Первая фаза выполняет две функции: аутентификацию удаленного пользователя и обмен информацией об открытых ключах, которые будут использоваться во второй фазе.

Аутентификацию можно выполнить несколькими различными способами. Наиболее часто используются технология предварительно распространяемых ключей (pre-shared keys) и технология цифровых сертификатов. Термин «предварительно распространяемые ключи» означает, что значения ключей предварительно задаются на всех компьютерах, которые собираются устанавливать соединения через VPN (что является существенным недостатком). При втором способе используются цифровые удостоверения (цифровые сертификаты, digital certificates), которые могут назначаться отдельно для каждого объекта, который соединяется с VPN. Цифровыми сертификатами можно удаленно управлять и администрировать из уполномоченного центра сертификации (Certificate Authority, CA).

Сертификационная служба является центральным элементом структуры, называемой инфраструктурой с открытым ключом (Public Key Infrastructure, PKI). За инфраструктурой PKI стоит концепция публично доступной структуры, распределяющей информацию об открытых (публичных) ключах.

В первой фазе при обмене аутентификационной информацией и параметрами безопасности могут использоваться два режима: основной (main mode) и агрессивный (aggressive mode). Различия между ними заключаются в количестве сетевых пакетов, которыми обмениваются стороны, и во времени, за которое генерируется открытый ключ. Агрессивный режим использует дополнительный заголовок меньшего размера, но основной режим обладает большей безопасностью и используется чаще всего.

Вторая фаза протокола IKE. Во второй фазе протокола IKE согласовываются конкретные параметры ассоциации обеспечения безопасности IPSec. Данное согласование подобно агрессивному режиму обмена информацией первой фазы. После завершения второй фазы формируется SA и пользователь получает подключение к VPN.

Во второй фазе возможен единственный режим согласования – быстрый режим (quick mode). Быстрый режим представляет собой короткий обмен, использующий три пакета. Все обмены второй фазы зашифрованы с помощью согласованных во время первой фазы протоколов и типов кодирования. При этом используется только защита, основанная на использовании хэш-функции и нонсе (nonce), включаемых в сетевые пакеты для подтверждения их оригинальности (нонсе является подтверждением того факта, что информация о ключе исходит из ожидаемого источника. Нонсе – это случайное число, генерируемое инициатором связи, которое заверяется респондентом цифровой подписью и посылается обратно).

Данная реализация включает в себя также идентификатор поставщика (vendor ID, VID), позволяющий участникам межплатформенных взаимодействий делать предположения о возможностях и конфигурации их партнеров, которые могут иметь различных изготовителей. После создания ассоциации обеспечения безопасности, используемой протоколом IKE, можно применять протоколы обеспечения безопасности. При построении VPN, основанной на протоколе IPSec, можно выбрать использование одного из протоколов (AH или ESP) или использовать их одновременно.

Управление ключами. Применяемый по умолчанию для IPSec протокол автоматизированного управления ключами называется ISAKMP/Oakley и состоит из следующих элементов:

- ✓ протокол определения ключей Oakley – протокол на основе алгоритма Диффи-Хеллмана, но обеспечивающий дополнительную защиту. Протокол Oakley называется общим, так как он не диктует использования каких-либо конкретных форматов;
- ✓ протокол защищенных связей и управления ключами в Интернете – обеспечивает основу схемы управления ключами и поддержку специального протокола и необходимых форматов процедуры согласования атрибутов защиты.

ISAKMP не заставляет использовать какой-то конкретный алгоритм обмена ключами, а предлагает использовать любой подобный алгоритм.
Протокол Oakley разработан в целях сохранения преимуществ алгоритма Диффи-Хеллмана и устранения его недостатков. Алгоритм Oakley характеризуется следующими особенностями:

- ✓ использование механизмов рецептов (cookies) для защиты от атак засорения;
- ✓ соглашение двух сторон о группе, которая определяет параметры алгоритма обмена ключами Диффи-Хеллмана;
- ✓ использование оказий для противостояния атакам повтора сообщений;
- ✓ обмен открытыми ключами Диффи-Хеллмана;
- ✓ аутентификация обмена для противостояния атакам «человек посередине».

Протокол аутентификации заголовка

Протокол AH – это IP-протокол 51. Он поддерживает функциональные возможности аутентификации и проверки целостности, но не поддерживает конфиденциальность содержимого пакета.

Обеспечение аутентификации и защиты целостности достигается добавлением дополнительного заголовка к IP-пакету. Этот заголовок содержит цифровую подпись, называемую значением проверки целостности (Integrity Check Value, ICV), которая является в основном значением хэш-функции, подтверждающей, что пакет не был изменен во время транспорта.

Информация IP-протокола, содержащаяся в пакете, гарантирует правильность содержимого пакета, но она передается в открытом виде. Поскольку протокол AH просматривает заголовок IP-пакета при вычислении цифровой подписи, можно убедиться, что IP-адрес отправителя подлинный и что пакет исходит от того, кого требуется.

Протокол AH также поддерживает использование порядковых номеров (sequence numbers), помогающих предотвращать нападения, основанные на повторном использовании пакета (replay attack). Эти номера используются коммуникационными устройствами для отслеживания потока сетевых пакетов сеанса связи.

Следующий заголовок	Длина содержимого пакета	Зарезервировано
Индекс параметра обеспечения безопасности (SPI)		
Порядковый номер		
Информация аутентификации (переменная длина, кратная 32 байтам)		

Рисунок 4.5. Структура заголовка AH-пакета.

Использование информации IP-заголовка протоколом AH делает его несовместимым с использованием NAT. Структура заголовка AH-пакета представлена на рисунке 4.5.

Поле следующего заголовка содержит идентификатор, определяющий тип заголовка пакета, следующего за заголовком АН-пакета. Поле длины содержимого пакета определяет длину заголовка АН-пакета. Индекс параметра обеспечения безопасности показывает, частью какого уникального потока связи ассоциации обеспечения безопасности является рассматриваемый пакет.

Рисунок 4.6. Схема преобразования исходного пакета в АН-пакет.

Порядковый номер – уникальное увеличивающееся значение, которое предназначено для противодействия повторному использованию пакета. Поле информации аутентификации содержит значение проверки целостности и цифровую подпись, подтверждающую подлинность рассматриваемого пакета. Схема преобразования исходного пакета в АН-пакет для различных режимов приведена на рисунке 4.6.

Протокол безопасной инкапсуляции содержимого пакета

Протокол ESP – это IP-протокол 50. Он обеспечивает конфиденциальность при помощи полного шифрования содержимого IP-пакетов. Протокол ESP реализован в виде модулей и может использовать любое количество доступных симметричных алгоритмов шифрования, в числе которых DES, 3DES, IDEA. Применение протокола ESP различается в зависимости от используемого режима протокола IPSec.

В транспортном режиме протокол ESP просто добавляет свой собственный заголовок после IP-заголовка и зашифровывает остальную часть сетевого пакета начиная с транспортного уровня. Если при этом определена служба аутентификации, то протокол ESP добавляет концевую метку (trailer). Концевая метка предназначена для подтверждения целостности пакета и аутентификации (в отличие от протокола АН значение проверки целостности вычисляется без использования информации из IP-заголовка).

При использовании туннельного режима протокол ESP инкапсулирует оригинальный пакет полностью, шифруя его целиком и создавая новый IP-заголовок и ESP-заголовок в устройстве туннелирования. Концевая метка также добавляется в случае выбора аутентификационного сервиса протокола ESP.

В любом режиме протокол ESP использует в каждом сетевом пакете порядковые номера. При работе протокола ESP в туннельном режиме можно использовать NAT. Структура заголовка пакета ESP приведена на рисунке 4.7.

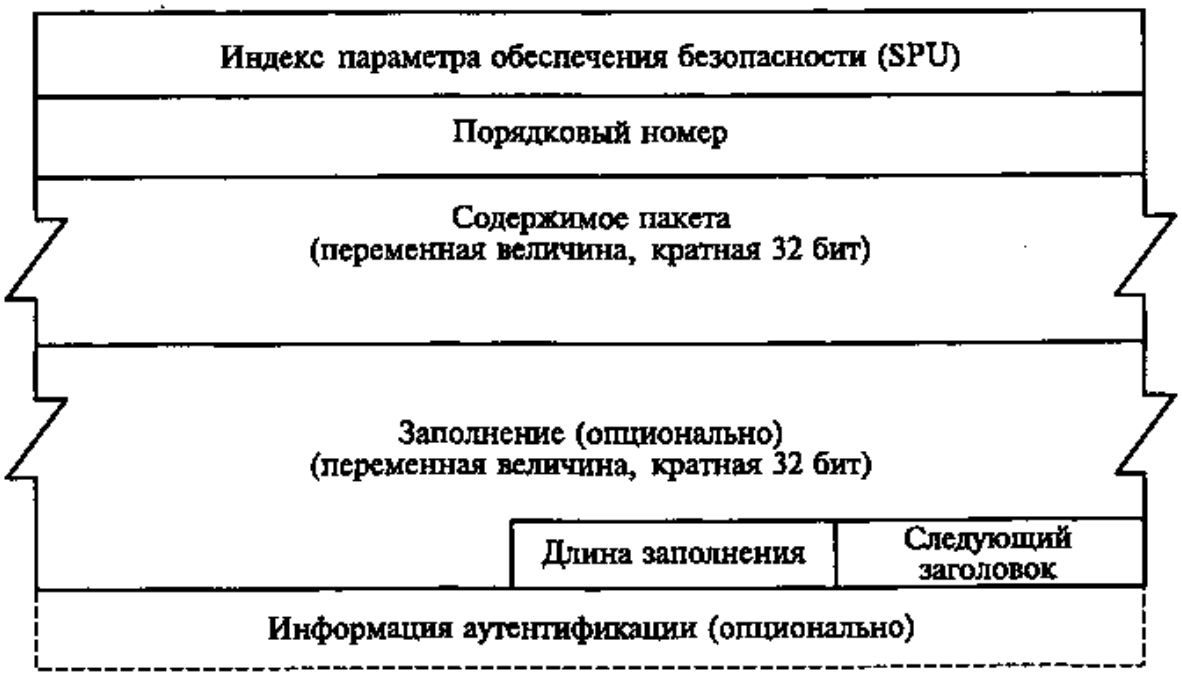


Рисунок 4.7. Структура заголовка пакета ESP.

Поле длины заполнения указывает, насколько заполнено содержимое пакета (если такое вообще имеет место), чтобы длина содержимого пакета и последующие поля заголовка соответствовали требованию выравнивания длины сетевого пакета.

Поле следующего заголовка сообщает номер протокола пакета, который инкапсулирован внутри пакета протокола ESP. Поле информации аутентификации содержит дополнительное значение проверки целостности, которое доступно для пакетов протокола ESP.

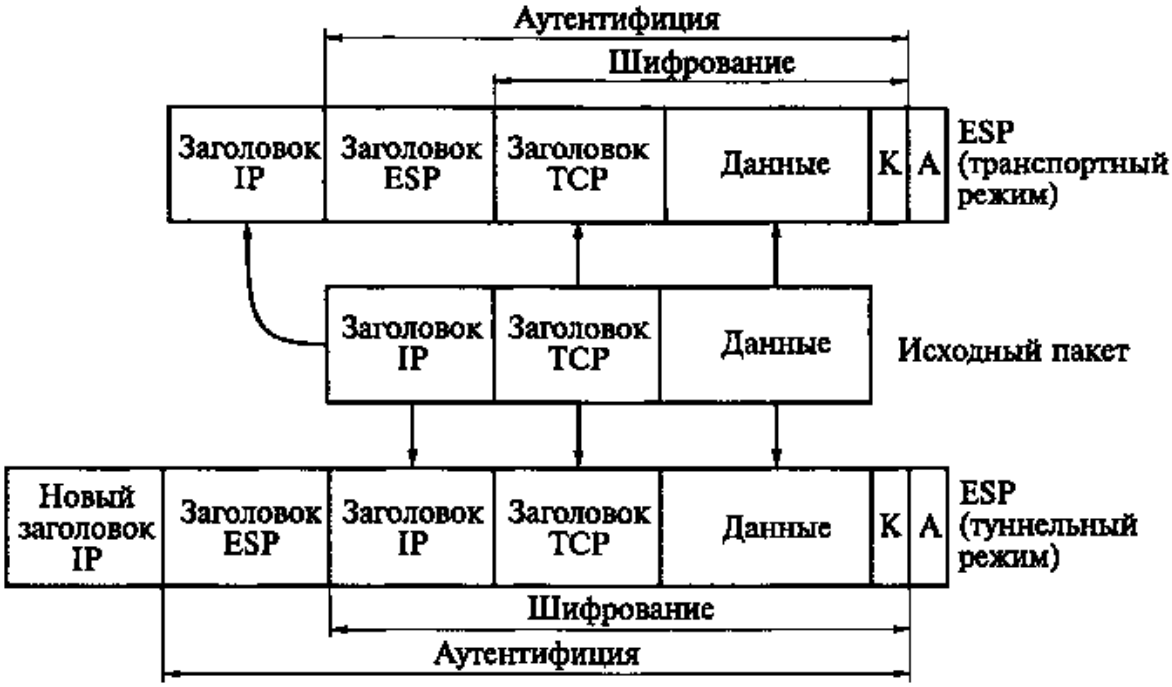


Рисунок 4.8. Схема преобразования исходного пакета в пакет ESP.

Схема преобразования исходного пакета в пакет ESP для различных режимов приведена на рисунке 4.8.

Совместное использование протоколов ESP и AH

Отдельная защищенная связь может использовать либо протокол AH, либо протокол ESP, но никак не оба эти протокола одновременно. Тем не менее, иногда конкретному потоку данных может требоваться и сервис AH, и сервис ESP. Во всех случаях одному потоку для получения комплекса услуг IPSec требуется несколько защищенных связей. Такой набор защищенных связей называется пучком защищенных связей (security association bundle), посредством которого потоку должен предоставляться необходимый набор услуг IPSec. При этом защищенные связи в пучке могут завершаться в различных конечных точках.

Защищенные связи могут быть объединены в пучки следующими двумя способами:

1. Транспортной смежности – в этом случае применяются несколько протоколов защиты к одному IP-пакету без создания туннеля. Эта комбинация AH и ESP эффективна только для одного уровня вложенности, так как обработка выполняется в одной точке — IPSec конечного получателя.
2. Повторного туннелирования – в этом случае применяются несколько уровней протоколов защиты с помощью туннелирования IP. Этот подход допускает множество уровней вложения, поскольку туннели могут начинаться и завершаться в разных использующих IPSec узлах сети вдоль маршрута передачи данных.

Кроме того, эти два подхода можно объединить. Тогда при рассмотрении пучков защищенных связей возникает вопрос – в каком порядке могут применяться аутентификация и шифрование между данной парой конечных узлов. Рассмотрим несколько подходов к такому комбинированию.

В случае применения ESP с опцией аутентификации пользователь сначала применяет ESP к требующим защиты данным, а затем добавляет поле данных аутентификации. В зависимости от используемых режимов возможны следующие варианты (в обоих вариантах аутентификации подлежит шифрованный текст):

- ✓ транспортный режим ESP – аутентификация и шифрование применяются к полезному грузу IP, доставляемому узлу адресата, но при этом заголовок IP не защищается;
- ✓ туннельный режим ESP – аутентификация применяется ко всему пакету IP, доставляемому по адресу IP внешнего получателя (например, МЭ), и выполняется этим получателем. Весь внутренний пакет IP защищается механизмом секретности, поскольку предназначен для доставки внутреннему адресату IP.

В случае транспортной смежности используется пучок из двух защищенных связей в транспортном режиме, где внутренняя связь является защищенной связью ESP, а внешняя — защищенной связью AH. В этом случае ESP используется без опции аутентификации. Поскольку внутренняя защищенная связь используется в транспортном режиме, шифрованию подлежит полезный груз IP. Получаемый в результате пакет состоит из заголовка IP, за которым следуют данные ESP. Затем применяется AH в транспортном режиме, так что аутентификация будет охватывать данные ESP и оригинальный заголовок IP, за исключением изменяемых полей. Достоинством данного подхода является то, что при комбинированном подходе аутентификация охватывает больше полей, включая поля адресов IP источника и назначения. Недостаток связан с использованием двух защищенных связей вместо одной.

Рассмотрим транспортно-туннельный режим. В некоторых случаях целесообразным является применение аутентификации до шифрования. Для этого используется пучок защищенных связей, состоящий из внутренней защищенной транспортной связи AH и внешней защищенной туннельной связи ESP. В этом случае аутентификация охватывает полезный груз IP вместе с заголовком IP (и расширениями), за исключением изменяемых полей. Полученный таким образом пакет затем обрабатывается в туннельном режиме ESP, в результате чего внутренний пакет вместе с данными аутентификации оказывается зашифрованным и имеющим новый внешний заголовок IP (с необходимыми расширениями.