**Мультикаст**

Существуют следующие типы трафика:

**Unicast** — одноадресная рассылка — один отправитель, один получатель. (Пример: запрос HTTP-странички у WEB-сервера).

**Broadcast** — широковещательная рассылка — один отправитель, получатели — все устройства в широковещательном сегменте. (Пример: ARP-запрос).

**Multicast** — многоадресная рассылка — один отправитель, много получателей. (Пример: IPTV).

**Anycast** — одноадресная рассылка ближайшему узлу — один отправитель, вообще получателей много, но фактически данные отправляются только одному. (Пример: Anycast DNS).

**Два основных принципа мультикастовой рассылки:**

1 Отправитель посылает только одну копию трафика, независимо от количества получателей.

2 Трафик получают только те, кто действительно заинтересован в нём.

В абсолютном большинстве случаев используется именно UDP. Это легко объясняется тем, что обычно с помощью многоадресной рассылки передаются данные, которые нужны здесь и сейчас.

В IPv4 был заложен блок адресов класса D: 224.0.0.0/4 (224.0.0.0-239.255.255.255). Адреса этого диапазона определяют мультикастовую группу. Один адрес — это одна группа, обычно она обозначается буквой «G».

Диапазон 224.0.0.0/24 зарезервирован под link-local коммуникации. Мультикастовые пакеты с такими адресами назначения не могут выходить за пределы одного широковещательного сегмента.

Диапазон 224.0.1.0/24 зарезервирован под протоколы, которым необходимо передавать мультикаст по всей сети, то есть проходить через маршрутизаторы.

**IGMP** — **Internet Group Management Protocol** — это сетевой протокол взаимодействия клиентов мультикастового трафика и ближайшего к ним маршрутизатора. Версий у протокола сейчас три: IGMPv1, IGMPv2, IGMPv3.

Роль **IGMP** очень проста: если клиентов нет — передавать мультикастовый трафик в сегмент не надо. Если появился клиент, он уведомляет маршрутизаторы с помощью IGMP о том, что хочет получать трафик.

**Процесс получения мультикаст-трафика**

**1.** Как только мы запустили приложение на клиенте и задали группу 224.X.X.X, в сеть будет отправлен пакет **IGMP Membership Report** — узел «рапортует» о том, что хочет получать трафик этой группы.

В IGMPv2 Report отправляется на адрес желаемой группы, и параллельно он же указывается в самом пакете. Данные сообщения должны жить только в пределах своего сегмента и не пересылаться никуда маршрутизаторами, поэтому и TTL у них 1.

Часто в литературе вы можете встретить упоминание о **IGMP Join** — это альтернативное название для IGMP Membership Report.

**2.** Маршрутизатор получает **IGMP-Report** и, понимая, что за данным интерфейсом теперь есть клиенты, заносит информацию в свои таблицы

ip igmp groups

ip mroute 224.X.X.X

Интерфейсы, в которые нужно передавать трафик, входят в список нисходящих интерфейсов — OIL — **Outbound Interface List**.

**3.** Клиент начал получать трафик. Теперь маршрутизатор должен иногда проверять, что получатели до сих пор у него есть. Для этого он периодически отправляет во все свои нисходящие интерфейсы запрос **IGMP Query**.

По умолчанию это происходит каждые 60 секунд. TTL таких пакетов тоже равен 1. Они отправляются на адрес 224.0.0.1 — все узлы в этом сегменте — без указания конкретной группы. Такие сообщений Query называются **General Query** — общие.

Получив **IGMP General Query**, любой хост, который слушает любую группу, должен отправить **IGMP Report**, как он это делал при подключении. В Report должен быть указан адрес интересующей его группы.

Если в ответ на **Query** на маршрутизатор пришёл хотя бы один **Report** для группы, значит есть ещё клиенты, он продолжает вещать в тот интерфейс, откуда пришёл этот Report, трафик этой самой группы.

Если на 3 подряд Query не было с интерфейса ответа для какой-то группы, маршрутизатор удаляет этот интерфейс из своей таблицы мультикастовой маршрутизации для данной группы — перестаёт туда посылать трафик.

По своей инициативе клиент обычно посылает Report только при подключении, потом — просто отвечает на Query от маршрутизатора.

Получив Query, он не торопится сразу же ответить Report'ом. Узел берёт тайм-аут длиной от 0 до Max Response Time, который указан в пришедшем Query.

Сделано это для того, чтобы сотни клиентов все скопом не наводнили сеть своими пакетам Report, получив General Query. Более того, только один клиент обычно отправляет Report. Дело в том, что Report отсылается на адрес группы, а следовательно, доходит и до всех клиентов. Получив Report от другого клиента для этой же группы, узел не будет отправлять свой. Логика простая: маршрутизатор и так уже получил этот самый Report и знает, что клиенты есть, больше ему не надо.

Этот механизм называется **Report Suppression**.

**4.** Если клиент хочет выйти из группы, он отправляет **IGMP Leave** на адрес группы.

Маршрутизатор получает его и по идее должен отключить. Но он ведь не может отключить одного конкретного клиента — маршрутизатор их не различает — у него просто есть нисходящий интерфейс. А за интерфейсом может быть несколько клиентов.

Маршрутизатор в ответ на Leave высылает **IGMP Query** на адрес группы, для которой этот Leave пришёл в тот интерфейс, откуда он пришёл. Такой пакет называется **Group Specific Query**. На него отвечают только те клиенты, которые подключены к данной конкретной группе.

Если маршрутизатор получил ответный **Report** для группы, он продолжает вещать в интерфейс, если не получил — удаляет по истечении таймера.

**Querier**

В клиентский сегмент подключено два (или больше) маршрутизатора, которые могут вещать трафик. Если ничего не сделать, мультикастовый трафик будет дублироваться — оба маршрутизатора ведь будут получать Report от клиентов. Во избежание этого существует **механизм выбора Querier — опрашивателя**.

Рассмотрим ситуацию с момента включения маршрутизаторов R1 и R2.

1) Активировали IGMP на интерфейсах.

2) Сначала по умолчанию каждый из них считает себя Querier.

3) Каждый отправляет **IGMP General Query** в сеть. Главная цель — узнать, есть ли клиенты, а параллельно — заявить другим маршрутизаторам в сегменте, если они есть, о своём желании участвовать в выборах.

4) **General Query** получают все устройства в сегменте, в том числе и другие IGMP-маршрутизаторы.

5) Получив такое сообщение от соседа, каждый маршрутизатор оценивает, кто достойнее.

6) Побеждает маршрутизатор с меньшим IP (указан в поле Source IP пакета IGMP Query). Он становится **Querier**, все другие — **Non-Querier**.

7) **Non-Querier** запускает таймер, который обнуляется каждый раз, как приходит **Query** с меньшим IP-адресом. Если до истечения таймера (больше 100 секунд: 105-107) маршрутизатор не получит Query с меньшим адресом, он объявляет себя Querier и берёт на себя все соответствующие функции.

8) Если Querier получает Query с меньшим адресом, он складывает с себя эти обязанности. Querier'ом становится другой маршрутизатор, у которого IP меньше.

**Отличия версий**

**Версия 1** отличается тем, что в ней нет сообщения **Leave**. Если клиент не хочет больше получать трафик данной группы, он перестаёт посылать Report в ответ на Query. Когда не останется ни одного клиента, маршрутизатор по таймауту перестанет слать трафик. Кроме того, не поддерживаются выборы Querier. За избежание дублирования трафика отвечает вышестоящий протокол, например, PIM, о котором мы будем говорить далее.

**Версия 3** поддерживает всё то, что поддерживает IGMPv2, но есть и ряд изменений.

Во-первых, Report отправляется уже не на адрес группы, а на мультикастовый служебный адрес 224.0.0.22. А адрес запрашиваемой группы указан только внутри пакета. Делается это для упрощения работы IGMP Snooping, о котором мы поговорим дальше.

Во-вторых, что более важно, IGMPv3 стал поддерживать SSM в чистом виде. Это так называемый Source Specific Multicast. В этом случае клиент может не просто запросить группу, но также указать список источников, от которых он хотел бы получать трафик или наоборот не хотел бы. В IGMPv2 клиент просто запрашивает и получает трафик группы, не заботясь об источнике.

IGMP не работает для IPv6. Там существует протокол MLD.

Термины:

**IGMP** — протокол, с помощью которого маршрутизатор узнаёт о наличии получателей мультикастового трафика и об их отключении.

**IGMP Report** — посылается клиентом при подключении и в ответ на IGMP Query. Означает, что клиент хочет получать трафик конкретной группы.

**IGMP General Query** — посылается маршрутизатором периодически, чтобы проверить какие группы сейчас нужны. В качестве адреса получателя указывается 224.0.0.1.

**IGMP Group Sepcific Query** — посылается маршрутизатором в ответ на сообщение Leave, чтобы узнать есть ли другие получатели в этой группе. В качестве адреса получателя указывается адрес мультикастовой группы.

**IGMP Leave** — посылается клиентом, когда тот хочет покинуть группу.

**Querier** — если в одном широковещательном сегменте несколько маршрутизаторов, которые могут вещать, среди них выбирается один главный — Querier. Он и будет периодически рассылать Query и передавать трафик.

**PIM**

**PIM** — **Protocol Independent Multicast.**

PIM имеет две версии:

**PIM Dense Mode (DM)**

**PIM Sparse Mode (SM)**

**PIM Dense Mode (DM)**

**PIM DM** пытается решить проблему доставки мультикаста в лоб. Он заведомо предполагает, что получатели есть везде, во всех уголках сети. Поэтому изначально он наводняет всю сеть мультикастовым трафиком, то есть рассылает его во все порты, кроме того, откуда он пришёл. Если потом оказывается, что где-то он не нужен, то эта ветка «отрезается» с помощью специального сообщения **PIM Prune** — трафик туда больше не отправляется.

Но через некоторое время в эту же ветку маршрутизатор снова пытается отправить мультикаст — вдруг там появились получатели. Если не появились, ветка снова отрезается на определённый период. Если клиент на маршрутизаторе появился в промежутке между этими двумя событиями, отправляется сообщение **Graft** — маршрутизатор запрашивает отрезанную ветку обратно.

После «обрезания» ненужных ветвей остаётся дерево, вдоль которого передаётся мультикастовый трафик. Это дерево называется **SPT — Shortest Path Tree**. Оно лишено петель и использует кратчайший путь от получателя до источника.

Для избежания петель используется **RPF — Reverse Path Forwarding.** Это главный принцип передачи мультикастового трафика в PIM (любого вида: и DM и SM) — трафик от источника должен приходить по кратчайшему пути.

То есть для каждого полученного мультикастового пакета производится проверка на основе таблицы маршрутизации, оттуда ли он пришёл.

1) Маршрутизатор смотрит на адрес источника мультикастового пакета.

2) Проверяет таблицу маршрутизации, через какой интерфейс доступен адрес источника.

3) Проверяет интерфейс, через который пришёл мультикастовый пакет.

4) Если интерфейсы совпадают — всё отлично, мультикастовый пакет пропускается, если же данные приходят с другого интерфейса — они будут отброшены.

Такая проверка называется **RPF-Check** и благодаря ей даже в более сложных сетях петли в MDT не возникнут.

**PIM Sparse Mode (SM)**

Заинтересованные узлы самостоятельно запрашивают подключение к дереву с помощью сообщений **PIM Join**. Если маршрутизатор не посылал **Join**, то и трафик ему отправляться не будет.

Запись вида **(\*, G)**, /читается старкомаджи/ сообщает нам о получателях. Причём не обязательно речь об одном клиенте-компьютере, вообще это может быть и, например, другой PIM-маршрутизатор. Важно то, в какие интерфейсы надо передавать трафик. Если список нисходящих интерфейсов (OIL) пуст — Null, значит нет получателей.

Запись **(S, G)**, /читается эскомаджи/ говорит о том, что известен источник.

Первое, что делает PIM — устанавливает соседство. Для этого используются сообщения **PIM Hello**. При активации PIM на интерфейсе, с него отправляется PIM Hello на адрес 224.0.0.13 с TTL = 1. Это означает, что соседями могут быть только маршрутизаторы, находящиеся в одном широковещательном домене.

**Rendezvous Point — RP**. Это центральное понятие PIM SM. Без неё ничего бы не работало. Здесь встречаются источник и получатели. Все PIM-маршрутизаторы должны знать, кто является RP в домене, то есть знать её IP-адрес.

Чтобы построить дерево **MDT (Multicast Distribution Tree)**, в сети выбирается в качестве RP некая центральная точка, которая:

1 отвечает за изучение источника,

2 является точкой притяжения сообщений Join от всех заинтересованных.

Существует два способа задания RP: статический и динамический.

Статическое задание. Чтобы увеличить надёжность, обычно выбирается адрес **Loopback**-интерфейса. Поэтому на всех маршрутизаторах выполняется команда:

RX(config)#ip pim rp-address [Lo-address]

этот адрес должен быть доступен по таблице маршрутизации со всех точек.

Процесс установления связи между источником и клиентами

1) Клиент 1 отправляет **IGMP Report** для группы G

2) Роутер **LHR** получает этот запрос, понимает, что есть клиент за интерфейсом FE, добавляет этот интерфейс в OIL и формирует запись **(\*, G)**

**LHR (Last Hop Router)** — последний маршрутизатор на пути мультикастового трафика, если считать от источника. Иными словами — это маршрутизатор, ближайший к получателю.

3) Поскольку на **LHR** пока нет мультикастового потока (он его не запрашивал прежде), он формирует сообщение **PIM Join** и отправляет его в сторону **RP**

**PIM Join** отправляется мультикастом на адрес 224.0.0.13. Такой Join обозначается ещё как **Join (\*,G)** и говорит: «Не важно, кто источник, мне нужен трафик группы G».

То есть каждый маршрутизатор на пути должен обработать такой Join и при необходимости отправить новый Join в сторону RP. (Важно понимать, что если на маршрутизаторе уже есть эта группа, он не будет отправлять выше Join — он просто добавит интерфейс, с которого пришёл Join, в OIL и начнёт передавать трафик).

4) RP, получив Join, формирует запись **(\*, G)** и добавляет интерфейс FE0/0 в OIL. Но Join отсылать уже некуда — он сам уже RP, а про источник пока ничего не известно.

**RPT — Rendezvous Point Tree**. Это дерево с корнем в RP, а ветви которого простираются до клиентов.

5) Источник начинает вещание. Он не волнуется о PIM, RP, IGMP — он просто вещает. А **FHR** получает этот поток. Его задача — доставить мультикаст до RP.

В PIM есть специальный тип сообщений — **Register**. Он нужен для того, чтобы зарегистрировать источник мультикаста на RP.

**FHR (First Hop Router)** — первый маршрутизатор на пути мультикастового трафика или ближайший к источнику.

6) **FHR** инкапсулирует каждый полученный от источника мультикастовый пакет в юникастовый PIM Register и отправляет его на RP.

7) RP получает PIM Register, распаковывает его и обнаруживает под обёрткой трафик для группы G.

Информацию об этом он сразу заносит в свою таблицу мультикастовой маршрутизации

Появилась запись (S, G) — (172.16.0.5, 224.2.2.4).

Распакованные пакеты RP дальше отправляет в RPT.

В принципе, на этом можно было бы и остановиться. Всё работает — клиенты получают трафик. Но есть две проблемы:

1 Процессы инкапсуляции и декапсуляции — весьма затратные действия для маршрутизаторов. Кроме того, дополнительные заголовки увеличивают размер пакета, и он может просто не пролезть в MTU где-то на промежуточном узле (вспоминаем все проблемы туннелирования).

2 Если вдруг где-то между источником и RP есть ещё получатели для группы, мультикастовому трафику придётся пройти один путь дважды.

8) Поэтому RP отправляет на FHR сообщение PIM Join. Но теперь уже в нём указывается для группы адрес не RP, а источника, изученный из сообщения Register. Такое сообщение называется **Join (S, G) — Source Specific Join**.

Цель у него точно такая же, как у PIM Join (\*, G) — построить дерево, только на этот раз от источника до RP.

Join (S, G) распространяется также узел за узлом, как обычный Join (\*, G). Только Join (\*, G) стремится к RP, а Join (S, G) к S — источнику. В качестве адреса получателя также служебный адрес 224.0.0.13 и TTL=1.

Путь, по которому прошёл Join от RP до источника, превращается в **Source Tree** — дерево от источника. Но более распространённое название — **SPT — Shortest Path Tree** — ведь трафик от источника до RP пойдёт по кратчайшему пути.

9) FHR получив Join (S, G), добавляет интерфейс, откуда пакет пришёл, в список нисходящих интерфейсов OIL и начинает туда вещать чистый мультикастовый трафик.

Но надо иметь ввиду, что сообщения Register передавались всё это время и передаются до сих пор. То есть фактически R1 отправляет две копии трафика сейчас: один — чистый мультикаст по SPT, другой — инкапсулированный в юникастовый Register.

10) Мультикаст достигает RP. Она понимает, что это тот же самый трафик, который приходит в Register, потому что одинаковый адрес группы, одинаковый адрес источника и с одного интерфейса. Чтобы не получать две копии, он отправляет на FHR юникастовый **PIM Register-Stop**.

**Register-Stop** не означает, что RP отказывается от трафика или не признаёт больше этот источник, это говорит лишь о том, что надо прекратить посылать инкапсулированный трафик.

Таким образом мы имеем одно большое дерево **MDT** для группы G от источника до клиентов. И это **MDT** составлено из двух кусков, которые строились независимо друг от друга: **Source Tree** от источника до RP и **RPT** от RP до клиентов. Вот оно отличие MDT от RPT и SPT. MDT — это довольно общий термин, означающий дерево передачи мультикаста вообще, в то время, как RPT/SPT — это его очень конкретный вид.

Если на RP начали приходить сообщения **Register** для какой-то группы, а для неё ещё нет получателей, то **RP** не заинтересован в получении этого трафика, поэтому, не отправляя PIM Join (S, G), RP сразу посылает **Register-Stop** на FHR.

FHR, получив Register-Stop и видя, что дерева для этой группы пока нет (нет клиентов), начинает отбрасывать мультикастовый трафик от сервера.

То есть сам сервер по этому поводу совершенно не беспокоится и продолжает посылать поток, но, дойдя до интерфейса маршрутизатора, поток будет отброшен.

При этом RP продолжает хранить запись (S, G). То есть трафик он не получает, но где находится источник для группы знает. Если в группе появляются получатели, RP узнаёт о них и посылает на источник **Join (S, G)**, который строит дерево.

Кроме того, каждые 3 минуты FHR будет пытаться повторно зарегистрировать источник на RP, то есть отправлять пакеты **Register**. Это нужно для того, чтобы уведомить RP о том, что этот источник ещё живой.

Важно понимать, что для каждой группы может быть своя RP. То есть в сети их может быть и две, и три, и сто — одна RP отвечает за один набор групп, другая — за другой. Более того, есть такое понятие, как Anycast RP и тогда разные RP могут обслуживать одну и ту же группу.

**Отключение ненужных ветвей**

После того, как последний клиент в сегменте отказался от подписки, PIM должен отрезать лишнюю ветку RPT.

Пусть, например, единственный клиент на LHR выключил компьютер. Маршрутизатор по сообщению **IGMP Leave** или после трёх безответных **IGMP Query** понимает, что клиентов за FE больше нет, и отправляет в сторону RP сообщение **PIM Prune**. По формату оно точно такое же, как Join, но выполняет противоположную функцию. Адрес назначения также 224.0.0.13, и TTL равен 1.

Но маршрутизатор, получивший **PIM Prune**, перед тем, как удалить подписку, ждёт некоторое время (обычно 3 секунды — **Join Delay Timer**). Это сделано для нижеследующей ситуации.

В одном широковещательном домене 3 маршрутизатора. Один из них стоит выше и именно он передаёт в сегмент мультикастовый трафик. Это R1. Для обоих маршрутизаторов (R2 и R3) его OIL содержит только одну запись.

Если теперь R2 решит отключиться и отправит PIM Prune, то он может подставить своего коллегу R3 — R1 ведь перестанет вещать в интерфейс вообще.

Так вот, чтобы этого не произошло, R1 и даёт таймаут в 3 секунды. За это время R3 должен успеть среагировать. Учитывая широковещательность сети, он тоже получит Prune от R2 и поэтому, если хочет продолжать получать трафик, он мгновенно отправляет обычный PIM Join в сегмент, уведомляя R1, что не надо удалять интерфейс.

Этот процесс называется — **Prune Override**.

**SPT Switchover** — переключение RPT-SPT Процесс оптимизации дерева.

Дело в том, что пока мы привязаны к RP — она корень RPT, только она поначалу знает, где кто находится. Однако после первого же мультикастового пакета все маршрутизаторы по пути трафика будут знать адрес источника, ведь он указан в заголовке IP. Почему бы кому-нибудь не отправить самому Join в сторону источника и оптимизировать маршрут? Такое переключение может инициировать LHR (Last Hop Router)

После получения первого мультикастового пакета **LHR** отправляет уже знакомый нам **Source Specific Join (S, G)** в интерфейс, который указан в его таблице маршрутизации, как исходящий.

Получив такой **Join**, следующий роутер отправляет его не на RP, как делал это с обычным **Join (\*, G)**, а в сторону источника (через интерфейс согласно таблице маршрутизации).

Далее этот Join попадет на **FHR.** А FHR нет разницы, кто его отправлял — RP или кто-то другой — он просто добавляет интерфейс, через который пришел Join, в свой OIL для группы 224.2.2.4.

Далее идет обрезание лишних потоков и построение SPT согласно таблице маршрутизации.

**SPT Switchover** может происходить:

1 Вообще никогда (команда ip pim spt-threshold infinity).

2 При достижении определённой утилизации полосы пропускания (команда ip pim spt-threshold XХХ).

3 Безусловно — сразу после получения первого пакета (действие по умолчанию или no ip pim spt-threshold XХХ)

**DR — Designated Router**.

Это выделенный маршрутизатор, который ответственен за отправку служебных пакетов на RP.

**Source DR** — отвечает за принятие мультикастовых пакетов непосредственно от источника и его регистрацию на RP.

**Receiver DR** — то же, что Source DR, только для получателей мультикастового трафика — LHR (Last Hop Router).

**Assert и PIM Forwarder**

Проблема двух одновременно передающих маршрутизаторов может возникнуть и в середине сети, где нет ни конечных клиентов, ни источников — только маршрутизаторы.

**PIM Forwarder** — тот маршрутизатор, который вправе вещать в данном сегменте.

**PIM Assert** позволяет маршрутизаторам, которые передают трафик одной группы в одну и ту же сеть, согласовать кто из них будет передавать трафик в сеть, для того чтобы не отправлять копии пакетов.

**Динамический выбор RP**

Динамический выбор RP позволяет и избежать ручной работы и обеспечить надёжность

В данный момент существует один общепризнанный протокол, позволяющий это сделать — **Bootstrap**.

Кандидаты на роль RP называются **C-RP**.

RX(config)#ip pim rp-candidate loopback 0

Чтобы информировать все мультикастовые маршрутизаторы домена о существующих RP вводится механизм **BSR — BootStrap Router**. Претендентов может быть несколько, как и **C-RP**. Они называются соответственно **C-BSR**. Настраиваются они похожим образом.

R1(config)#ip pim bsr-candidate loopback 0

Сначала из всех **C-BSR** выбирается один главный **BSR**. Для этого каждый C-BSR отправляет в сеть мультикастовый **BootStrap Message (BSM)** на адрес 224.0.0.13 — это тоже пакет протокола PIM. Его должны принять и обработать все мультикастовые маршрутизаторы и после разослать во все порты, где активирован PIM. BSM передаётся не в сторону чего-то (RP или источника), в отличии, от PIM Join, а во все стороны. Такая веерная рассылка помогает достигнуть BSM всех уголков сети, в том числе всех C-BSR и всех C-RP. Для того, чтобы BSM не блуждали по сети бесконечно, применяется всё тот же механизм **RPF** **Reverse Path Forwarding** — если BSM пришёл не с того интерфейса, за которым находится сеть отправителя этого сообщения, такое сообщение отбрасывается.

С помощью этих BSM все мультикастовые маршрутизаторы определяют самого достойного кандидата на основе приоритетов. Как только C-BSR получает BSM от другого маршрутизатора с большим приоритетом, он прекращает рассылать свои сообщения. В результате все обладают одинаковой информацией.

На этом этапе, когда выбран **BSR**, благодаря тому, что его **BSM** разошлись уже по всей сети, **C-RP** знают его адрес и юникастом отправляют на него сообщения **Candidte-RP-Advertisement**, в которых они несут список групп, которые они обслуживают —**group-to-RP mapping**. BSR все эти сообщения агрегирует и создаёт **RP-Set** — информационную таблицу: какие RP каждую группу обслуживают.

Далее BSR в прежней веерной манере рассылает те же **BootStrap Message**, которые на этот раз содержат **RP-Set**. Эти сообщения успешно достигают всех мультикастовых маршрутизаторов, каждый из которых самостоятельно делает выбор, какую RP нужно использовать для каждой конкретной группы.

BSR периодически делает такие рассылки, чтобы с одной стороны все знали, что информация по RP ещё актуальна, а с другой C-BSR были в курсе, что сам главный BSR ещё жив.

RP, кстати, тоже периодически шлют на BSR свои анонсы **Candidate-RP-Advertisement**.

Фактически всё, что нужно сделать для настройки автоматического выбора RP — указать C-RP и указать C-BSR — не так уж много работы, всё остальное за вас сделает PIM.

Как всегда, в целях повышения надёжности рекомендуется указывать интерфейсы Loopback в качестве кандидатов.

**Важнейшие моменты PIM SM**

1 Должна быть обеспечена обычная юникастовая связность с помощью IGP или статических маршрутов. Это лежит в основе алгоритма RPF.

2 Дерево строится только после появления клиента. Именно клиент инициирует построение дерева. Нет клиента — нет дерева.

3 RPF помогает избежать петель.

4 Все маршрутизаторы должны знать о том, кто является RP — только с её помощью можно построить дерево.

5 Точка RP может быть указана статически, а может выбираться автоматически с помощью протокола **BootStrap**.

6 В первой фазе строится RPT — дерево от клиентов до RP — и Source Tree — дерево от источника до RP. Во второй фазе происходит переключение с построенного RPT на SPT — кратчайший путь от получателя до источника.

**Типы сообщений PIM Sparse Mode:**

**Hello** — для установления соседства и поддержания этих отношений. Также необходимы для выбора DR.

**Join (\*, G)** — запрос на подключение к дереву группы G. Не важно кто источник. Отправляется в сторону RP. С их помощью строится дерево **RPT**.

**Join (S, G)** — Source Specific Join. Это запрос на подключение к дереву группы G с определённым источником — S. Отправляется в сторону источника — S. С их помощью строится дерево **SPT**.

**Prune (\*, G)** — запрос на отключение от дерева группы G, какие бы источники для неё не были. Отправляется в сторону RP. Так обрезается ветвь RPT.

**Prune (S, G)** — запрос на отключение от дерева группы G, корнем которого является источник S. Отправляется в сторону источника. Так обрезается ветвь SPT.

**Register** — специальное сообщение, внутри которого передаётся мультикаст на RP, пока не будет построено SPT от источника до RP. Передаётся юникастом от FHR на RP.

**Register-Stop** — отправляется юникастом с RP на FHR, приказывая прекратить посылать мультикастовый трафик, инкапсулированный в **Register**.

**Bootstrap** — пакеты механизма BSR, которые позволяют выбрать маршрутизатор на роль BSR, а также передают информацию о существующих RP и группах.

**Assert** — сообщение для выбора **PIM Forwarder**, чтобы в один сегмент не передавали трафик два маршрутизатора.

**Candidate-RP-Advertisement** — сообщение, в котором RP отсылает на BSR информацию о том, какие группы он обслуживает.

**RP-Reachable** — сообщение от RP, которым она уведомляет всех о своей доступности.

**SSM — Source Specific Multicast**

Проблема SSM в том, что клиенты должны заранее знать адреса источников — никакой сигнализацией они им не сообщаются.

Поэтому SSM хорош в тех ситуациях, когда в сети определённый набор источников, их адреса заведомо известны и не будут меняться. А клиентские терминалы или приложения жёстко привязаны к ним.

Иными словами IPTV — весьма пригодная среда для внедрения SSM. Это хорошо описывает концепцию One-to-Many — один источник, много получателей.

**BIDIR PIM**

двунаправленный PIM (Bidirectional PIM, BIDIR PIM). В отличие от SSM в нём напротив полностью отказываются от SPT и записей (S,G) — остаются только Shared Tree с корнем в RP.

И если в обычном PIM, дерево является односторонним — трафик всегда передаётся от источника вниз по SPT и от RP вниз по RPT — есть чёткое деление, где источник, где клиенты, то в двунаправленном от источника трафик к RP передаётся также вверх по Shared Tree — по тому же самому, по которому трафик течёт вниз к клиентам.

Это позволяет отказаться от регистрации источника на RP — трафик передаётся безусловно, без какой бы то ни было сигнализации и изменения состояний. Поскольку деревьев SPT нет вообще, то и SPT Switchover тоже не происходит.

**IGMP Snooping**

**Multicast VLAN Replication**

**MSDP — Multicast Source Discovery Protocol**

Протокол для обмена информацией об источниках между различными RP.

Вообще говоря, RP знает обо всех источниках только внутри своего PIM SM домена. Естественно, он не может знать о Источниках в других доменах.

**MSDP** позволяет обеспечить обмен информацией об Источниках между RP в разных PIM SM доменах, в частности между различными ISP.

Кроме того, MSDP используется в Anycast RP. В этом случае множество RP, которые обcлуживают одну группу, обмениваются друг с другом информацией об источнике именно по протоколу MSDP.

Соединение устанавливается на TCP-порт 639.