STP & SNR

Содержание

- Введение.
- STP (802.1D, Spanning Tree Protocol).
  - Выбор корневого коммутатора (Root Bridge).
  - Определение корневых портов (Root Port).
  - Определение назначенных портов (Designated Port).
  - Таймеры STP и изменение топологии.
  - STP Toolkit.
- RSTP (802.1w, Rapid Spanning Tree Protocol).
  - RSTP vs STP. Изменения в BPDU, новые роли и состояния портов.
  - Новые типы линков.
  - Proposal & Agreement.
- MSTP (802.1s, Multiple Spanning Tree Protocol).
  - Взаимодействие внутри региона MSTP.
  - Взаимодействие между регионами MSTP.
  - Формат MSTP BPDU.
  - Роли портов в MSTP.
  - Topology change.
  - Взаимодействие MSTP с проприетарными PVST+/RPVST+.
- Настройка и работа STP на коммутаторах SNR.
  - Настройка spanning-tree на основных сериях коммутаторов SNR.
  - Взаимодействие SNR с проприетарными PVST+/RPVST+.
  - Настройка spanning-tree на серии S5210.
- Заключение.

Введение

В любой современной компьютерной сети, будь то сеть крупного интернет-провайдера или корпоративная сеть небольшого предприятия, необходимо так или иначе обеспечить резервирование каналов связи для бесперебойной работы сети или для минимального простоя сервисов, ведь в современных реалиях, даже минутное отсутствие доступа к сети может быть очень критичным как для клиентов, так и для оператора. Под резервированием, например, понимают организацию дополнительных линков (дополнительного плеча), чтобы в случае падения линка, трафик перетек на второе плечо – тогда пользователи и не заметят простоя связи.

Помимо резервирования, также необходимо обезопасить свою сеть, в частности, от возникновения нежелательных колец/петель. Например, если кто-то случайно, а, может быть даже, специально захотел организовать дополнительный линк между коммутаторами (возможно, этот человек как раз хотел добиться некоего резервирования, либо просто хотел навредить нам) и таким образом организовал кольцо. Или даже уже был организован резервный линк, чтобы его можно было поднять в случае аварии, но кто-то по ошибке включил административно порт на коммутаторе при другом активном линке.

Все эти задачи по устранению нежелательных колец и петель коммутации в совокупности с отказоустойчивостью решает небезызвестный протокол STP (Spanning Tree Protocol). Именно о нем и его видах и пойдет речь в данной статье.

Несмотря на то, что в сети Интернет уже множество раз рассматривали работу данного протокола, мы сделаем это еще раз и постараемся сделать максимально подробно, так как, во-первых, с момента основания протокола прошло достаточно большое количество времени, вышло очень много статей, вебинаров, обзоров, где каждый по-своему рассматривал данный протокол, мы же постараемся освежить все и привести к одному виду, а во-вторых, не помешает освежить знания перед тем, как мы будем рассматривать принцип работы STP/RSTP/MSTP и их настройку и работу на коммутаторах SNR.

Но зачем вообще нам нужен STP, почему мы просто так не можем организовать дополнительный физический линк? Чем вообще может быть чревато наличие колец в сети?

Чтобы ответить на этот вопрос, вспомним принцип работы коммутатора. Коммутатор оперирует такими сущностями, как кадры (frame). Это PDU канального уровня. И при своей работе коммутатор опирается на адреса в этих PDU, которые называются MAC-адреса - адреса канального (L2) уровня.

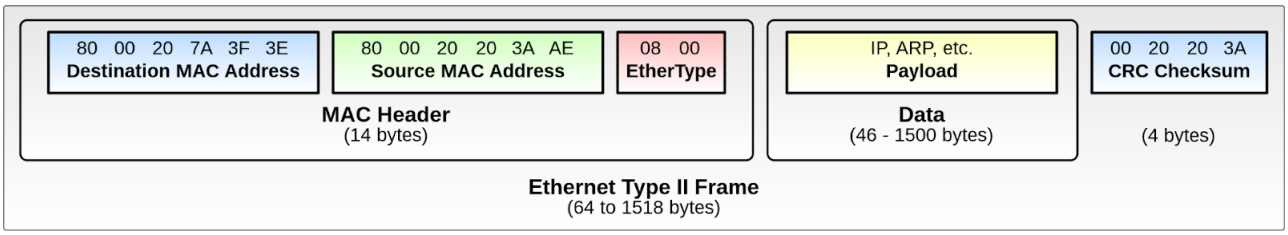

Рис. 1. Пример самого распространенного формата Ethernet кадра из Википедии

Когда на порт коммутатора приходит кадр, коммутатор смотрит в поле Ethernet-заголовка SMAC (Source MAC Address - адрес отправителя или адрес источника), извлекает оттуда MAC-адрес и записывает в свою таблицу коммутации (таблицу MAC-адресов или CAM-таблицу).

Выглядит эта таблица следующим образом: соответствие VLAN, в котором был изучен MAC-адрес, к самому MAC-адресу и к порту, за которым был изучен данный MAC-адрес (т.е. с какого интерфейса к нам пришел кадр с данным MAC-адресом отправителя). Могут встречаться еще различные поля в таблице MAC-адресов, но сейчас они нам не так интересны, мы вспомнили только самые важные для изучения STP.

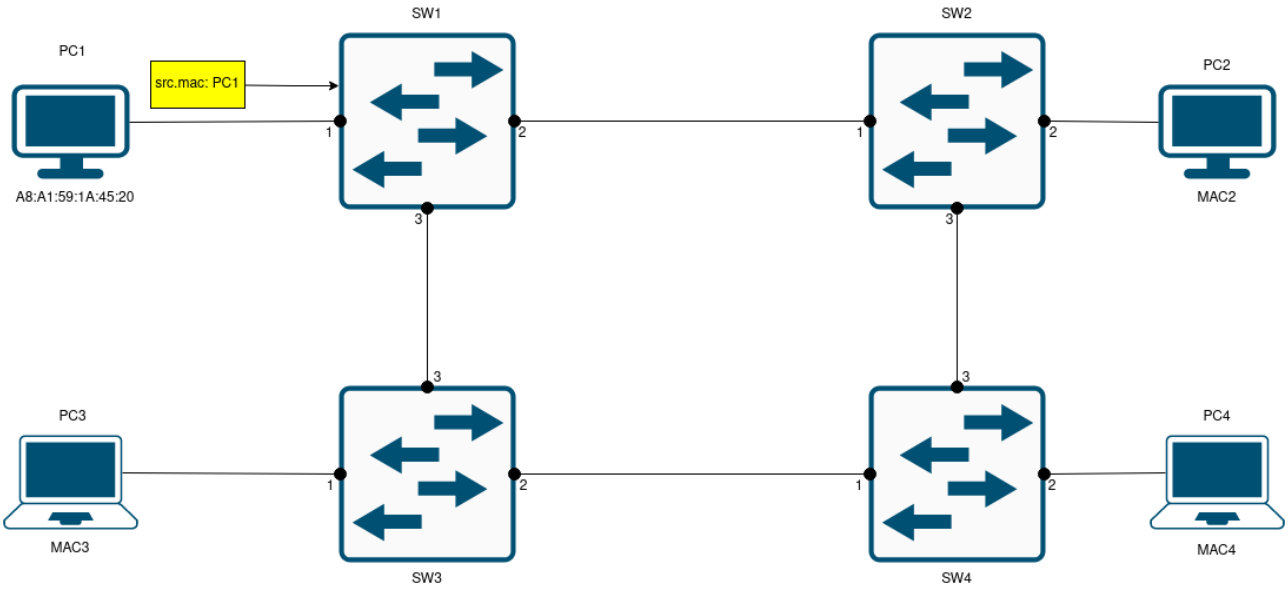

Рис. 2. Кадр пришел на один из портов коммутатора

[STP & SNR]

Рис. 3. Пример записи из таблицы MAC-адресов

Далее коммутатору необходимо перенаправить данный кадр. Для этого он смотрит в Ethernet-заголовке поле DMAC (Destination Address - адрес получателя или адрес назначения), и затем коммутатор запускает процесс поиска MAC-адреса получателя в своей таблице коммутации.

Когда коммутатор находит запись в своей таблице о MAC-адресе получателя, то без лишних раздумий коммутатор отправляет кадр на порт, соответствующий данному MAC-адресу.

Однако, если записи в таблице о данном адресе нет, то коммутатор отправит кадр на все порты, кроме того порта, откуда пришел данный кадр (такой тип трафика называется "unknown-unicast").

Конечные устройства, которым не предназначается данный кадр, т.е. которые имеют MAC-адрес отличный от MAC-адреса получателя, просто отбросят данный кадр и на этом его жизненный путь закончится.

Помимо этого, некоторые кадры заведомо могут быть широковещательными (broadcast, т.е. иметь MAC-адрес назначения FF-FF-FF-FF-FF-FF), например, при работе таких протоколов как ARP, DHCP, и т.д., т.е. когда мы не знаем, кому именно отправлять пакет, мы его отправим всем, в надежде, что наша цель, кто бы то ни был, получит данный кадр и ответит нам (при необходимости). Ну и раз уж мы вспомнили broadcast-трафик, вспомним и многоадресный или мультикаст (multicast) трафик,

когда трафик точно также разливается во все порты, кроме того, откуда он пришел.

Итого, у нас есть три метода отправки трафика, при котором трафик в любом случае будет отправляться во все порты, т.к. мы не знаем, куда именно или кому его отправлять. Такой трафик называется BUM-трафик (Broadcast, Unknown-unicast, Multicast).

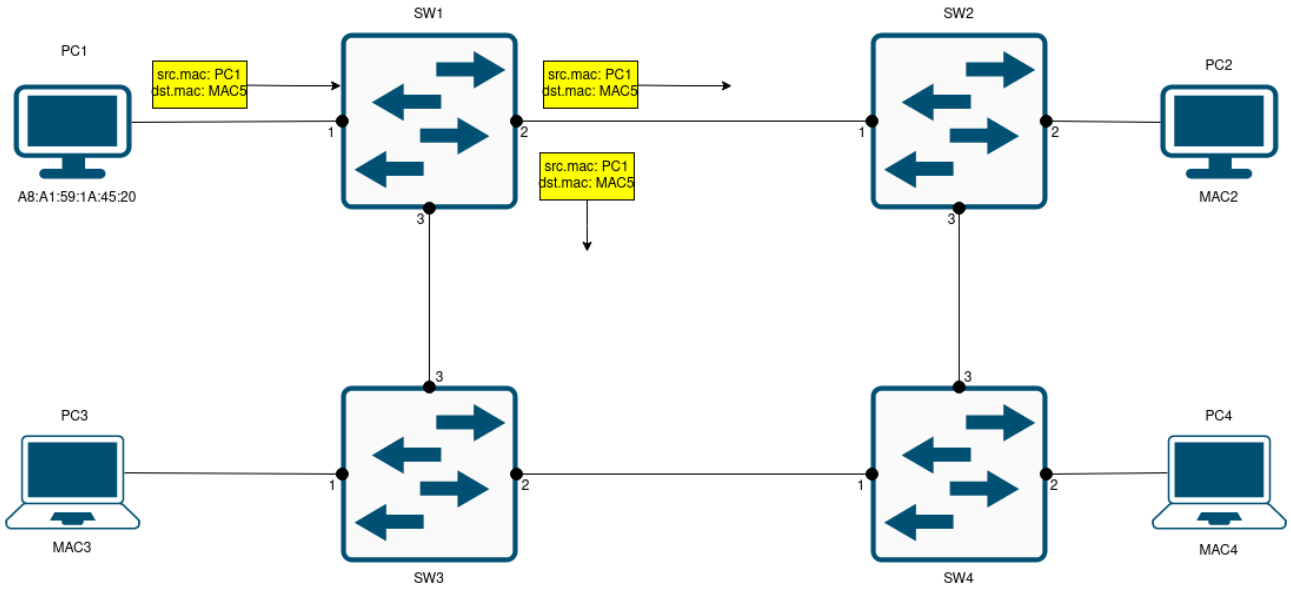

Рис. 4. Пример отправки unknown-unicast трафика

Если мы вновь обратимся к Ethernet-заголовку (неважно какой формат: Ethernet II или IEEE 802.3), в нем мы не увидим поля, которое ограничивает время жизни кадра (как, например, поле "TTL" в заголовке IPv4-пакета, или "Hop Limit" в фиксированном заголовке IPv6, т.е. на уровне L3). А это значит, что пакет будет блуждать по сети вечно, точнее до тех пор, пока кольцо не будет разорвано (имеется в виду физическое/административное отключение линка, либо выключение коммутатора).

Особо в этом ничего страшного нет, что у нас по сети будет вечно блуждать пара пакетов. Однако со временем один отправленный ранее пакет, бесконечно блуждающий в нашем широковещательном домене, размножится, плюс не стоит забывать, что большинство других современных протоколов используют широковещательные сообщения для своей работы (тот самый BUM-трафик). И если в обычной ситуации кол-во широковещательных пакетов не должно составлять более 10% от общего трафика в сети, то в данном случае их кол-во будет очень быстро расти, вследствие чего уже через несколько минут сеть окажется неработоспособной: потери трафика до 90-100%, отсутствие возможности управления сетевыми устройствами и т.д.

Описанная ситуация при неконтролируемом росте BUM-трафика в сети называется широковещательный шторм, который является следствием возникновения колец и петель в сети.

На рисунке ниже показан пример начала широковещательного шторма. То есть, коммутатор увидел, что кадр содержит dst.mac (MAC-адрес получателя), запись о котором отсутствует в его таблице, поэтому он отправит кадр во все порты. То же самое сделают и другие коммутаторы (также кадры будут отправляться и в порты, за которыми находятся конечные устройства, там они будут отбрасываться, ведь кадр предназначается не им). Причем будет уже две копии кадра, которых со временем будет становиться еще больше, ведь у нас все четыре коммутатора соединены между собой избыточными линками, а механизма TTL в Ethernet нет.

[STP & SNR]

Рис. 5. Пример начала широковещательного шторма

Даже если сеть будет работоспособна какое-то время, мы можем столкнуться с еще одним неприятным явлением - MAC Address Flapping (флаппинг MAC-адресов). Суть данного понятия в том, что кадр, который отправил какой-либо пользователь, будет бесконечно ходить по сети, и коммутаторы будут изучать его и записывать MAC-адрес отправителя (Source MAC Address) в свои таблицы. Однако через некоторое время, точно такой же кадр будет получен уже на другом порту коммутатора, соответственно коммутатор изменит запись в своей таблице, полагая, что данный отправитель находится именно за данным портом (куда пришел кадр), хотя это может быть и не так. И так будет продолжаться постоянно: один и тот же MAC-адрес будет фиксироваться на разных портах, это и называется - флаппинг MAC-адресов. В результате этого явления, данные,

предназначенные какому-либо получателю, вполне могут и не дойти до него, так как кадр будет отправлен в “неправильный” порт.

Избежать широковещательного шторма и флаппинга MAC-адресов нам поможет протокол STP.

STP (Spanning Tree Protocol) — сетевой протокол (или семейство сетевых протоколов) предназначенный для автоматического удаления циклов (петель коммутации) из топологии сети на канальном уровне в Ethernet-сетях. Первоначальный протокол STP описан в стандарте 802.1D. Позже появилось несколько новых протоколов (RSTP, MSTP, PVST, PVST+, ...), отличающихся некоторыми особенностями в алгоритме работы, в скорости, в отношении к VLAN'ам и ряде других вопросов, но в целом решающих ту же задачу похожими способами. Все их принято обобщенно называть STP-протоколами.

STP использует алгоритм STA (Spanning Tree Algorithm), результатом работы которого является граф в виде дерева (связный и без простых циклов).

Обратимся к определению из википедии: "Граф — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин."

Если переводить дословно: spanning tree - охватывающее, связующее или остовное дерево.

Связующее или остовное дерево (как его чаще называют в математике) - это подграф без циклов, содержащий все вершины исходного графа. Грубо говоря, остовное дерево получается из исходного графа удалением максимального числа рёбер, входящих в циклы, но без нарушения связности графа.

То есть, возвращаясь от математики к сетям, связующее дерево содержит все исходные коммутаторы, но при этом в нем нет циклов, то есть избыточных соединений между этими коммутаторами. Это если рассматривать логическую топологию. Из физической топологии избыточные линки никуда не исчезают. Поэтому можно сказать, что физическая топология - полноценный граф, а логическая топология, полученная в результате работы алгоритма STP - подграф, где каждые две ноды соединены только одним путем.

Поэтому, как и было сказано ранее, основная цель STP, вне зависимости от реализации протокола - автоматическое удаление циклов (петель коммутации) из топологии сети на канальном уровне.

Сперва мы рассмотрим подробно три вида (реализации) STP:

- классический STP;
- RSTP (Rapid STP);
- MSTP (Multiple STP).

Также, вскользь упомянем про семейство проприетарных Cisco протоколов PVST и их взаимодействие с мультивендорными STP-протоколами. А затем рассмотрим их настройку и работу на коммутаторах SNR.

Если кто-то уже в деталях знает и понимает работу всех протоколов семейства spanning-tree, можно перейти сразу к рассмотрению главы: "Настройка и работа STP на коммутаторах SNR".

STP (802.1D, Spanning Tree Protocol)

Первоначальный (классический) STP был разработан в 1985 году Радией Перлман (Radia Joy Perlman - американский программист и сетевой инженер), за что она в свою очередь получила прозвище «Мать Интернета», так как протокол STP сделал возможным появление массовых сетей с использованием технологии Ethernet, потому что иначе использование кольцевых топологий в Ethernet было бы невозможным.

Впервые STP был стандартизован IEEE (Institute of Electrical and Electronics Engineers) в 1990 году в стандарте IEEE 802.1D-1990, который в свою очередь был заменен на 802.1D-1998 в 1998 году. Данный стандарт описывает не только сам STP, в нем описано как в принципе нужно коммутировать трафик в Ethernet-сетях, и, соответственно, также описан протокол STP.

В 2001 году вышло дополнение к стандарту 802.1D - IEEE 802.1t-2001, где была изменена длина полей для создания уникального Bridge ID для каждого VLAN, и изменены значения path cost на линках, но об этом позднее.

В 2004 году как 802.1D-1998, так и 802.1t-2001, и опубликованный в том же 2001 IEEE 802.1w-2001 (стандарт для RSTP) были перенесены в IEEE 802.1D-2004 (причем оригинальный STP был удален из этого документа и был заменен на RSTP). Но об этом опять же позднее.

Далее углубляться в стандарты не будем, необходимо было поверхностно рассмотреть хронологию и принадлежность к стандартам, так как у тех, кто впервые разбирается с STP, может возникнуть небольшая путаница с этим. Сейчас нам важно то, что оригинальный STP принято ассоциировать со стандартом 802.1D, в котором он и был изначально описан.

Протокол STP работает на канальном (L2) уровне. STP позволяет делать топологию избыточной на физическом уровне, но при этом логически блокировать петли. Достигается это с помощью того, что коммутаторы в STP отправляют специальные сообщения BPDU и обнаруживают фактическую топологию сети. А затем, после определения ролей коммутаторов и портов, часть портов блокируется так, чтобы в итоге получилась логическая топология без петель.

Как мы уже сказали, в STP для общения и обмена информацией между собой коммутаторы используют специальные пакеты, так называемые BPDU (Bridge Protocol Data Units).

Сразу отметим из определения "Bridge" (сетевой мост). Если коротко, bridge - устройство, которое находится на промежуточной стадии между концентратором (он же хаб) и коммутатором. Протокол STP разрабатывался еще во времена, когда использовались сетевые мосты. Сейчас такие устройства в современных сетях, разумеется, не используются, как и хабы, поэтому под "bridge" мы в дальнейшем контексте будем понимать именно коммутатор.

Что же касается BPDU, то данные сообщения могут использоваться не только для протокола STP. Существуют еще, например, LLDP BPDU, LACP BPDU и т.д. Мы же в дальнейшем под BPDU будем понимать именно spanning-tree BPDU.

BPDU в STP бывают двух видов: конфигурационные (Configuration BPDU, также в литературе можно часто встретить определение “Hello BPDU”) и об изменении топологии - TCN (Topology Change Notification BPDU).

В классическом STP Configuration BPDU рассылаются именно корневым коммутатором (и ретранслируются остальными) и используются для построения топологии, вторые, как понятно из названия, отсылаются в случае изменения топологии сети.

Configuration BPDU имеет следующие поля:

- Protocol Identifier (Идентификатор протокола STP, 2 байта): 0x0000;
- Protocol Version Identifier (Версия протокола STP, 1 байт): 0 для STP;
- BPDU Type (Тип BPDU, 1 байт): 0x00 для Configuration BPDU;
- BPDU flags (Флаги, 1 байт): бит 0 (младший) для TC (Topology change), бит 7 (старший) для TCA (Topology Change Acknowledgement), остальные не используются;
- Root Identifier (Идентификатор корневого коммутатора, 8 байт): Root Bridge Priority + Root Bridge System ID;
- Root Path Cost (Расстояние до корневого коммутатора, 4 байта);
- Bridge Identifier (Идентификатор коммутатора, 8 байт): Bridge Priority + Bridge System ID;
- Port identifier (Идентификатор порта, 2 байта);
- Message Age (Время жизни сообщения, 2 байта);
- Max Age (Таймер - максимальное время жизни сообщения, 2 байта);
- Hello Time (Таймер - интервал отправки Hello BPDU, 2 байта);
- Forward Delay (Таймер - задержка смены состояний, 2 байта).

[STP & SNR]

Рис. 6. Пример дампа Configuration BPDU

TCN BPDU состоит только из заголовка и имеет следующие поля:

- Protocol Identifier (Идентификатор протокола STP, 2 байта): 0x0000.
- Protocol Version Identifier (Версия протокола STP, 1 байт): 0 для STP.
- BPDU Type (Тип BPDU, 1 байт): 0x80 для TCN BPDU.

[STP & SNR]

Рис. 7. Пример дампа TCN BPDU

Как мы можем заметить, BPDU сообщения STP инкапсулируются сперва в LLC (Logical Link Control -

802.2). Как мы помним, это верхний подуровень канального уровня модели OSI. LLC осуществляет управление передачей данных, а также обеспечивает проверку и правильность передачи информации по соединению.

Зачем это было сделано - непонятно, но факт остается фактом, STP инкапсулируется в LLC, даже если формат L2-кадра Ethernet - Ethernet II, а не 802.3 (по крайней мере так нам об этом сообщает Wireshark).

Сами BPDU, как правило, делятся на три условные части: это некий заголовок BPDU, Priority Vector и таймеры STP.

Заголовок BPDU занимает 5 байт и содержит такие поля как Protocol Identifier, Protocol Version Identifier, BPDU Type и BPDU Flags.

Priority vector занимает 22 байта и состоит из Root Identifier, Root Path Cost, Bridge Identifier, Port Identifier (передающей стороны), и еще при получении могут добавляться 2 байта Port Identifier (принимающей стороны), но в современных сетях этот параметр не используется, так как предназначен для разрешения неоднозначности в случае применения хабов и, соответственно, при полудуплексном режиме работы на каком-либо коммутирующем интерфейсе.

Подробно Priority Vector мы рассматривать не будем, скажем лишь, что вектор приоритета - это совокупность некой информации, которой обмениваются коммутаторы для определения активных маршрутов и вычисления топологии. Priority vector позволяет определять, какая BPDU лучше (superior), а какая хуже (inferior). Определяется это по 22 байтам содержимого.

Мы для простоты, как и все остальные, будем сравнивать BPDU не по значению вектора приоритета, а по отдельным полям, которые входят в него: Root Identifier, Root Path Cost, Bridge Identifier, Port Identifier. Как это происходит, мы узнаем немного позднее.

И последняя часть BPDU - таймеры, занимают 8 байт, используются непосредственно при работе STP, при сходимости сети и т.д.

Все коммутаторы (по умолчанию каждые 2 секунды) отправляют BPDU со всех работающих портов на мультикастовый Ethernet-адрес 01-80-C2-00-00-00 (в случае обычного мультивендорного STP) или, например, на 01-00-0C-CC-CC-CD (в случае проприетарного Cisco PVST), который прослушивают все коммутаторы, на которых включен и запущен STP. Если на коммутатор приходит Hello BPDU, то он понимает, что за этим портом также находится коммутатор, а не какое-то конечное устройство или маршрутизатор и т.д.

Для того чтобы определить какие порты заблокировать, а какие будут передавать данные, выполняется алгоритм STA, и состоит он из следующих последовательных шагов:

1. Выбор корневого коммутатора (Root Bridge).
2. Определение корневых портов (Root Port).
3. Определение назначенных портов (Designated Port).
4. Блокировка оставшихся портов (Non-Designated Port).

Выбор корневого коммутатора (Root Bridge)

Итак, сначала выбирается, так называемый, корневой мост/коммутатор (Root Bridge). Root Bridge - это устройство (коммутатор), который STP считает центральной точкой для всех коммутаторов в сети, все дерево STP сходится к нему (поэтому он и называется корнем - root). Root Bridge будет отвечать за пересылку трафика в сети.

Выбор корневого моста основан на таком понятии как идентификатор коммутатора (Bridge ID или сокращенно BID), который, напомню, передается в Configuration BPDU.

Bridge ID - это число длиной 8 байт (64 бита), которое по умолчанию в своем классическом варианте состоит из двух полей: Bridge Priority (2 байта или 16 бит) и MAC-адреса устройства (6 байт или 48 бит):

[STP & SNR]

Рис. 8. Формат классического Bridge ID

По умолчанию Bridge Priority (приоритет) равен 32768 на всех коммутаторах и может принимать значения от 0 до 65535 (2^16).

Однако, как мы помним, STP разрабатывался еще в далекие времена, когда дизайн сетей не подразумевал использование VLAN, соответственно, строилось одно общее дерево для всех

коммутаторов. Поэтому позднее сначала Cisco в своем проприетарном PVST протоколе, а затем и IEEE в стандарте 802.1t-2001 (который, скорее, являлся дополнением к 802.1D-1998, а затем добавлен в 802.1D-2004) переосмыслили этот момент и переработали формат Bridge ID (при этом сохранив совместимость с классическим STP), и в результате, в числе прочего (мы чуть позже рассмотрим, что еще поменялось в 802.1t) появилось такое понятие как “Extended System ID”, и появилась возможность добавлять его в Bridge Priority.

Поэтому формат Bridge ID в версиях STP, которые поддерживают работу с VLAN, стал выглядеть следующим образом:

[STP & SNR]

Рис. 9a. Формат обновленного Bridge ID из 802.1t

Или если подробнее, то Bridge Priority стал выглядеть так:

[STP & SNR]

Рис. 9b. Формат обновленного Bridge ID из 802.1t

Сетевые инженеры подумали, что отдавать на приоритет коммутатора 16 бит – слишком много. Это, по сути, позволит администратору распределить 65536 потенциальных Root коммутаторов, но такое количество в сетях явно будет излишним, поэтому на Bridge Priority оставили 4 бита, а остальные 12 бит оставили для Extended System ID (VLAN ID или mst instance в случае MSTP).

Понятно, почему 12 бит выделено под Extended System ID - потому что максимально возможное количество VLAN - 4096 (2^{12}).

Оставшиеся 4 бита были оставлены для Bridge Priority, который теперь мог принимать значения от 0 до 61440, причем только кратные 4096 (т.к. в определении приоритета остались играть роль только первые значащие 4 бита, что наглядно видно из рисунка выше).

Откуда получился такой диапазон? Bridge ID вычисляется по формуле: “4096*N + номер VLAN/mst instance”, где N - множитель, который может принимать значения от 0 до 15 ($2^4 = 16$, т.к. осталось 4 бита под поле приоритета), и по умолчанию он равен 8 (то есть значение Bridge Priority все также по умолчанию 32768), потому что старший бит в Bridge Priority принимает значение единицы ($1000_2 = 8_{10}$).

Это и объясняет то, почему значения Bridge Priority должны быть кратны 4096 - по значению младшего бита в Priority, то есть могут принимать значения: 0, 4096, 8192, 12288, 16384, 20480, 24576, 28672, 32768, 36864, 40960, 45056, 49152, 53248, 57344, 61440 (то есть 16 возможных значений, как раз на 4 бита).

Вместе обновленный Bridge Priority и Extended System ID и образуют новый Bridge Priority в BID. В примере выше на картинке приоритет бы был равен 32768 + 1 = 32769 (priority + vlanid).

Однако, поле Extended System ID на сегодняшний день используется преимущественно только у Cisco. Кстати, добавление данного параметра в Cisco можно вообще отключить на некоторых платформах: ‘no spanning-tree extended system-id’. То есть этот параметр, по сути, необязательный и изначально был введен исключительно для Cisco PVST для учета VLAN.

К слову, в Cisco сначала появился протокол PVST (Per-VLAN Spanning Tree) - проприетарный протокол Cisco, который для каждого VLAN строит отдельное дерево (на Cisco нет как такового классического spanning-tree, только своя вариация). Однако PVST использовал инкапсуляцию только в ISL trunk. ISL (Inter-Switch Link) - это опять же проприетарный Cisco протокол, создан до появления dot1q (мало где используется, даже на самих Cisco). Поэтому позднее компания Cisco разработала улучшение для PVST - протокол PVST+, который позволяет инкапсулировать кадры с помощью стандарта IEEE 802.1Q. По умолчанию на всех коммутаторах Cisco работает PVST+. Эта информация может пригодиться нам, когда будем рассматривать взаимодействие мультивендорных протоколов spanning-tree с семейством проприетарного PVST.

Однако мы немного отвлеклись, в STP/RSTP мы не встретим добавление номера VLAN в расширенном Bridge Priority, так как это реализовано только в тех версиях STP, которые поддерживают работу с разными VLAN (у нас для работы с VLAN будет MSTP, но о нем позднее), поэтому данный параметр используется в основном только в протоколах, которые умеют работать с VLAN, а в обычных версиях STP/RSTP, параметр Extended System ID всегда, как правило, принимает нулевое значение: даже если используется расширенный формат Bridge Priority и кадр инкапсулируется в 802.1Q, то в дампе мы все равно увидим что-то вроде: “Bridge System ID Extension: 0”. Многие вендоры, несмотря на то, что новый формат Bridge Priority был даже стандартизирован IEEE, используют 4 бита под Bridge Priority (то есть используют Priority кратные 4096) и оставляют 12 бит под VID, но не добавляют его к

значению Bridge Priority.

Это было затронуто, опять же, для того, чтобы у читателя, который только начинает свое знакомство с STP, увидевшего в Интернете такие сущности как Extended System ID или приоритет по умолчанию 32769 вместо 32768, не возник ступор: что это и почему так, почему где-то по умолчанию priority 32768 без добавления VLAN, а где-то еще и VLAN почему-то добавляется.

Вернемся к нашему обычному Bridge ID (далее для простоты будем рассматривать стандартный BID, без расширения для VLAN) и выборам корневого коммутатора.

На рисунке ниже показан пример, как BID выглядит в дампе:

[STP & SNR]

Рис. 10. Пример BID в BPDU

Сначала сверяется только Bridge Priority. Побеждает и становится корневым тот коммутатор, у которого Bridge Priority меньше (например, 4096 выигрывает у 32768). Если приоритет одинаковый, то уже сравниваются MAC-адреса (Bridge System ID), и снова: у кого MAC-адрес меньше, тот и становится корневым. По умолчанию, у коммутаторов приоритет всегда 32768, поэтому в некоторых источниках и пишут, что BID выбирается именно по MAC-адресу коммутатора, но это не совсем верно. То, что именно MAC-адрес является ключевым фактором при выборе Root Bridge, происходит только если приоритет у коммутаторов одинаковый (что и бывает в большинстве случаев, если не менять вручную настройки приоритета).

В начале выборов каждый коммутатор считает себя корневым, о чем и заявляет всем остальным с помощью BPDU, в котором представляет свой идентификатор BID как ID корневого коммутатора - Root Bridge ID. В этом же BPDU каждый коммутатор передает свой BID и указывает, что Root Path Cost = 0, потому что считает себя корневым.

То есть после включения сети абсолютно все коммутаторы наводняют сеть своими Configuration BPDU, где Bridge ID = Root Bridge ID, и отправляются эти BPDU вообще со всех портов, ведь на Root коммутаторах все порты являются Designated, а Configuration BPDU отправляются именно через Designated-порты.

[STP & SNR]

Рис. 11. Начало выборов Root Bridge

Если коммутатор получает BPDU, где Bridge ID больше, чем у него, то коммутатор отбрасывает этот BPDU и продолжает считать корневым именно себя. Если коммутатор получает BPDU от коммутатора с меньшим Bridge ID (Superior BPDU), то он перестает анонсировать информацию о том, что он корневой и начинает передавать BPDU коммутатора с меньшим Bridge ID, указывая свой BID, новый Root Bridge ID и новый Root Path Cost до этого Root Bridge. И так коммутаторы продолжают обмениваться BPDU, пока не решат единогласно, кто же будет Root коммутатором.

В итоге, корневым оказывается тот коммутатор, чей Bridge ID меньше всех.

Может быть такое, что при одинаковых приоритетах (например, везде настроен приоритет по умолчанию - 32768) корневым коммутатором может быть выбран самый старый и непроизводительный коммутатор, и тогда весь трафик пойдет через него, что может привести к определенным проблемам в работе сети. Поэтому лучше назначать приоритеты на устройствах вручную.

В STP после выбора Root коммутатора, только он продолжает рассылать мультикастом Configuration BPDU каждые 2 секунды. На Root Bridge все порты являются Designated, и находятся они всегда в состоянии Forwarding. Другие коммутаторы будут перенаправлять BPDU от корневого коммутатора, но сами они их перестают генерировать после сходимости сети.

Как уже было сказано ранее, Configuration BPDU отправляются только через Designated порты. Коммутаторы не перенаправляют BPDU от корневого коммутатора через свои Root или Non-Designated порты: с корневых портов не отправляются, потому что нет смысла отправлять BPDU обратно коммутатору, который его и сгенерировал, а с Non-Designated не отправляются, потому что такие порты находятся в состоянии Blocking, а при таком состоянии порта коммутатор не отправляет BPDU с него, а только принимает, но об этом позднее.

Роли портов в STP:

1. Root Port (корневой порт коммутатора) - порт для передачи трафика корневому коммутатору. Каждый “некорневой” коммутатор имеет только один корневой порт, выбранный из соображений минимальной стоимости пути.

2. Designated Port (назначенный порт сегмента) - “некорневой” порт коммутатора между сегментами сети, принимающий трафик из соответствующего сегмента. На “некорневом” коммутаторе может быть множество Designated портов, а вот в каждом сегменте сети Designated порт всегда один.
3. Non-Designated Port (“неназначенный” порт сегмента) - порт, не являющийся корневым, или назначенным. Передача данных через такой порт запрещена.

Определение корневых портов (Root Port)

После того, как в сети был определен коммутатор с ролью Root Bridge, каждый из остальных коммутаторов должен найти один и только один порт, который будет вести к корневому коммутатору. Такой порт называется корневым портом (Root Port). Он предоставляет единственный путь для данного коммутатора к корневому.

Корневой порт выбирается по следующим критериям (или, как их называют в англоязычной литературе - tiebreaker; при совпадении первого критерия, выбор происходит по следующему и т.д.):

- Наименьшего Root Path Cost (lowest root cost).
- Наименьшего Bridge ID соседа (lowest neighbor bridge ID).
- Наименьшего Port ID соседа (lowest neighbor port ID).
- Наименьшего локального Port ID (lowest local port ID).

Root Path Cost (RPC) - это общее значение стоимости всех линков до корневого коммутатора. Эта стоимость определяется суммой стоимостей всех линков, которые нужно пройти кадру, чтобы дойти до корневого коммутатора (lowest root cost). В свою очередь, стоимость линка определяется просто — по его пропускной способности (чем выше скорость, тем меньше стоимость).

Происходит выбор Root Port следующим образом:

- Корневой коммутатор посылает BPDU с полем Root Path Cost, равным нулю (ноль, потому что он и так корневой коммутатор, ему не нужно никуда добираться, как и напрямую подключенным к нему коммутаторам, ведь у них уже есть прямое подключение к Root Bridge).
- Ближайший коммутатор смотрит на скорость своего порта, куда BPDU пришел, и добавляет стоимость согласно таблице (если быть точнее, коммутатор смотрит не на скорость порта - speed, а именно на пропускную способность - bandwidth. Пропускная способность по умолчанию равна скорости на интерфейсе, но может быть изменена при необходимости отдельно).

Таблица 1. Рекомендуемая стоимость порта в STP (из 802.1D-1998) в зависимости от пропускной способности

Пропускная способность	Рекомендуемая стоимость
4 Mbps	250
10 Mbps	100
16 Mbps	62
100 Mbps	19
1 Gbps	4
2 Gbps	3
10 Gbps	2

Еще раз хочется подчеркнуть, что это именно рекомендуемые значения стоимости линков согласно стандарта, но настраивать cost можно и по своему усмотрению.

В стандарте 802.1t-2001 рекомендованные значения стоимости были изменены (позднее были перемещены вместе с RSTP в IEEE 802.1D-2004):

[STP & SNR]

Рис. 12. Выдержка из стандарта IEEE 802.1t-2001

Стоимость портов в 802.1D была в 16-битном формате. Причем в стандарте указывалось, что именно стоимость path cost ограничивалась 16 битами, а вот стоимость до Root – уже 32 бита, потому что в BPDU под неё было выделено сразу именно столько. Это несоответствие было решено в 802.1t – теперь любой path cost имел 32-битный формат. Рекомендуемое значение cost в 802.t получается путем деления 20 000 000 000 bit/s на пропускную способность линка в kbit/s (либо просто 20 Tbit/s на пропускную способность линка).

Помимо прочего, в 802.1t, как мы уже знаем, изменился формат BID, был добавлен Edge тип порта и т.д.

Но мы опять отвлеклись (к тому же все это мы посмотрим, когда будем разбирать RSTP), вернемся к процессу выбора Root портов в классическом STP:

- Коммутатор указывает новое значение RPC перед отправкой BPDU своим соседям: это значение состоит из стоимости порта, с которого можно добраться до Root Bridge, то есть куда пришел BPDU от этого самого Root коммутатора и из значения RPC, которое было указано в полученном BPDU.
- Далее этот коммутатор, получивший BPDU с RPC 0, посылает данный BPDU нижестоящим коммутаторам, но уже с новым значением Root Path Cost в зависимости от пропускной способности интерфейса, который у него смотрит в сторону корневого (на котором получил BPDU), и далее по цепочке вниз.

[STP & SNR]

Рис. 13a. Выбор Root Port на основе lowest root cost

На рисунке выше видно, что коммутаторы сообщают своим соседям, кто является Root Bridge, и сколько будет стоить добраться до него через них. При этом они просто прибавляют стоимость своего интерфейса, на котором получают BPDU, и передают его дальше. И если для каких-то коммутаторов выбор корневого порта будет очевиден, например, для SW2 корневым явно будет Gi1/0/1, потому что там cost 0, в то время как на Gi1/0/3 он видит, что ему анонсируют стоимость до Root Bridge - 8, значит добраться до корневого коммутатор будет стоить 12 (8 + стоимость порта, на котором получен BPDU). 4 (0 + стоимость порта, на котором был получен BPDU) в данном контексте явно лучше (т.к. стоит меньше), чем 12. То же самое и с коммутатором SW3. А вот для других коммутаторов выбор по RPC будет невозможен. Например, SW4 в данном контексте, получает одинаковые значения стоимости до Root Bridge, и как тогда ему выбрать, какой порт делать корневым?

Посмотрим, как определились корневые порты:

[STP & SNR]

Рис. 13b. Выбор Root Port на основе lowest root cost

Как видим, на SW2 и SW3 корневые порты удалось выбрать без проблем по наименьшему RPC, а вот на SW4 на всех портах, с которых приходят BPDU - path cost одинаковый. Что же делать в таком случае?

Если стоимость линков до корневого коммутатора совпадает, то в ход вступает следующий tiebreaker, и выбор корневого порта происходит на основе меньшего Bridge ID соседа-коммутатора (lowest neighbor bridge ID).

То есть, если RPC на одном коммутаторе одинаковый для разных портов, то корневым портом станет порт, к которому подключен сосед с наименьшим Bridge ID.

[STP & SNR]

Рис. 14. Выбор Root Port на основе lowest neighbor bridge ID (при одинаковом RPC)

Если и Bridge ID коммутаторов до корневого коммутатора совпадает (например, в случае, когда между двумя коммутаторами два и более линка), то тогда корневой порт выбирается на основе

меньшего Port ID соседа (lowest neighbor port ID).

На L2-коммутаторе при работающем STP у каждого порта свой уникальный номер (идентификатор Port ID). Port ID, например, 0x8019 (hex) = 128.25 (dec), где 128 - default port priority, а 25 - ifindex порта, который определяется на коммутаторе. Причем, Port ID, как и в случае с Bridge ID, смотрится на другом конце линка (то есть не на локальном коммутаторе, а на соседе-коммутаторе, откуда получаем кадры на наш локальный коммутатор, где и происходит выбор Root port).

[STP & SNR]

Рис. 15. Выбор Root Port на основе lowest neighbor port ID (при одинаковом RPC и BID)

Но что если и на этот раз у нас не получилось выявить победителя? Например, на нашем локальном коммутаторе есть два порта, которые подключаются в хаб, линки от которого в свою очередь идут в сторону другого коммутатора. Тогда у нас будет одинаковый neighbor port ID на локальном коммутаторе, соответственно нужен еще один tiebreaker для выбора корневого порта.

Поэтому последний критерий при выборе - lowest local port ID, это порт с наименьшим ID на нашем локальном коммутаторе.

[STP & SNR]

Рис. 16. Выбор Root Port на основе lowest local port ID (при одинаковом RPC, BID, neighbor port ID)

Однако в современных сетях выбор по такому критерию маловероятен, т.к. хабы сейчас практически нигде не используются, но все равно не стоит забывать об этом.

Что важно запомнить:

- Root Port находится всегда в состоянии Forwarding.
- На линке напротив Root-порта (т.е. на коммутаторе соседе) ВСЕГДА должен быть Designated-порт, потому что путь до корневого коммутатора не может быть заблокирован.
- В сегменте всегда должен быть Designated-порт, чтобы принимать трафик, но наоборот - необязательно (порт напротив Designated может быть и заблокирован, а не быть Root).

Определение назначенных портов (Designated Port)

Далее выбираются назначенный (Designated) коммутатор в сегменте сети и соответственно назначенные (Designated) порты. Сегмент - это домен коллизий, т.е. если рассматривать коммутируемые Ethernet сети, то это, грубо говоря, патч-корд между коммутаторами. Либо это может быть несколько соединений (патч-кордов) между коммутаторами, например, при использовании хабов или общей шины в Ethernet.

Коммутатор в сегменте сети, имеющий наименьшее расстояние до корневого коммутатора, называется назначенным коммутатором или Designated bridge, а порт этого коммутатора, который подключен к рассматриваемому сегменту сети, является назначенным портом.

Назначенным (Designated) портом выбирается тот, который имеет лучшую стоимость до корневого коммутатора в данном сегменте. На “некорневом” коммутаторе всегда есть только один Root порт и при этом может быть множество Designated портов. А вот в каждом сегменте сети Designated порт всегда один.

Ранее мы уже упоминали, что у корневого коммутатора все порты являются Designated, и обычно это действительно так, но с некоторой оговоркой. Как мы уже сказали, что сегмент - домен коллизий, и в каждом домене коллизий будет выбран только один Designated порт. Но что если к корневому коммутатору двумя или более линками будет подключен хаб? Это будет означать, что порты корневого коммутатора находятся в одном сегменте (в одном домене коллизий), поэтому только один из них останется Designated, а остальные будут Non-Designated. Однако в современных сетях хабы практически не используются, поэтому и распространено говорить, что у корневого коммутатора все порты, как правило, являются Designated.

Так же, как и Root порт, Designated порт выбирается на основе определенных критериев в порядке очередности (tiebreakers):

- Наименьшего Root Path Cost коммутатора (lowest root cost).
- Наименьшего Bridge ID коммутатора (lowest bridge ID).
- Наименьшего локального Port ID (lowest local port ID).

Выбирается все именно относительно сегмента.

На примере рисунка ниже, рассмотрим, как сеть разделяется на сегменты:

[STP & SNR]

Рис. 17. Разбиение сети на сегменты

На рисунке выше, каждый сегмент выделен в красный эллипс. Для каждого сегмента существует свой назначенный порт. В сегменте, где к коммутатору подключается какое-либо конечное устройство, порт коммутатора всегда будет Designated. Также, как правило, у корневого коммутатора все порты назначенные (только если к нему не подключен хаб). Данные порты мы уже можем точно определить как Designated.

А вот как определять остальные назначенные порты для каждого сегмента описано в алгоритме выше.

Lowest Root Path Cost не означает, что Designated порт будет выбран на основе наименьшего RPC на этом линке, рассматривается именно RPC на каждом коммутаторе в сегменте, где мы и выбираем Designated порт. То есть, назначенным портом в сегменте будет выбран тот, на коммутаторе которого наименьшее значение стоимости path cost до Root Bridge.

[STP & SNR]

Рис. 18. Выбор Designated портов на основе RPC

Если Root Path Cost коммутаторов одинаковый, то Designated порт будет определен на основе меньшего BID в сегменте (DP будет выбран на коммутаторе с наименьшим BID).

[STP & SNR]

Рис. 19. Выбор Designated портов на основе BID

Если и BID в сегменте одинаковые (т.е. если это один коммутатор), то Designated порт будет выбран по наименьшему Local Port ID.

[STP & SNR]

Рис. 20. Выбор Designated портов на основе local port ID

Также Designated выбираются порты, с которых не приходят BPDU, т.е. коммутатор понимает, что за данным портом находится какое-то конечное устройство (например, ПК, телефон, сервер, которые не умеют отправлять BPDU), и он спокойно переводит его в состояние Forwarding, т.к. нет риска возникновения петли.

И также, как мы уже говорили, Designated-портом обязательно будет порт напротив Root порта в сегменте.

И вот уже после того, как выбраны корневые и назначенные порты, оставшиеся порты становятся Non-Designated и блокируются, таким образом разрывая петлю. Заблокированный порт может продолжать принимать кадры, но не будет их никуда перенаправлять, также он продолжает слушать BPDU (т.е. важно понимать, что STP не отключает линк полностью, потому что BPDU должны продолжать отправляться на случай изменения топологии, не передается только data трафик).

[STP & SNR]

Рис. 21. Блокировка оставшихся Non-Designated портов

Подводя итог, еще раз сравним для понимания Root и Designated-порты. Root порт-выбирается на коммутаторе для ОТПРАВКИ трафика корневому (Root) коммутатору, или в его в сторону (поэтому на корневом коммутаторе и нет Root портов), и на данном коммутаторе, где выбирается Root порт, этот Root порт является ближайшей точкой выхода (кратчайший путь до Root Bridge), чтобы добраться до корневого коммутатора. А Designated порт, ПРИНИМАЕТ этот трафик в сегменте (который был отправлен через Root порт предыдущего коммутатора), чтобы в конечном счете трафик дошел до корневого коммутатора, и опять же по кратчайшему пути (именно поэтому на корневом коммутаторе все порты Designated, и он сам будет являться Designated Bridge в сегменте, потому что имеет “лучший путь” до самого себя во всех сегментах).

Таймеры STP и изменение топологии

В STP существуют следующие состояния портов:

- **Disabled** - отключенный порт. Порт, выключенный административно (командой shutdown или вследствие errdisabled). Данный порт ничего не передает и не принимает: ни пользовательский трафик, ни BPDU. MAC-адреса, разумеется, не изучает.
- **Blocking** - блокирование. Данное состояние предназначено для предотвращения петель в сети. Однако, это не означает, что порт будет неактивен: STP не может выключать этот линк полностью. Блокированный порт все равно слушает и принимает BPDU (хоть сам и НЕ отправляет), так как ему нужно быть готовым перейти в другое состояние в случае работы алгоритма STA. MAC-адреса он НЕ изучает. Кадры с данными также НЕ перенаправляет;
- **Listening** - прослушивание. Порт слушает и начинает сам отправлять BPDU (только Designated порт). MAC-адреса НЕ изучает. Кадры с данными НЕ перенаправляет;
- **Learning** - обучение. Порт слушает и отправляет BPDU (отправляет только Designated порт), а также вносит изменения в CAM-таблицу (начинает изучать MAC-адреса, по трафику, который приходит на порт). Данные НЕ перенаправляет;
- **Forwarding** - пересылка. Порт слушает/отправляет BPDU (отправляет только Designated порт), участвует в поддержании таблицы MAC-адресов и перенаправляет данные. То есть это обычное состояние рабочего порта.

Blocking и Forwarding - стабильные состояния, а Listening и Learning - промежуточные.

Если порт изменяет состояние с Blocking в Forwarding, то он должен пройти через два промежуточных состояния: Listening и Learning. Переход из Forwarding в Blocking может выполняться сразу.

Root/Designated порты всегда должны оставаться в стабильном состоянии Forwarding (если только не происходит изменение топологии).

Non-Designated порты всегда остаются в стабильном Blocking состоянии.

Работа классического протокола STP основа на таймерах.

Как мы уже рассматривали выше в структуре BPDU, в STP существуют следующие таймеры:

- **Hello Time** - интервал отправки Hello BPDU. Это интервал, через который посылаются пакеты BPDU с Root Bridge. По умолчанию - 2 секунды, но можно редактировать;
- **Forward Delay** - задержка смены состояний. Определяет время, как долго Root или Designated порты коммутатора будут оставаться в состоянии Listening и Learning. По умолчанию - 15 секунд. Можно еще сказать так, что Forward Delay - это время, за которое информация в BPDU гарантированно будет передана от любого коммутатора в топологии до любого другого. Далее будет понятно, почему так можно толковать этот таймер;
- **Max Age** - максимальное время жизни сообщения. Показывает, когда текущий Hello BPDU устареет и должен быть удален. По умолчанию - это 20 секунд (10*Hello Time). Если кадр BPDU имеет время жизни, превышающее максимальное, то кадр игнорируется коммутаторами. Таймер обнуляется при получении BPDU. Если текущий Hello BPDU устарел, но новые BPDU не приходят (таймер не обнуляется), то коммутатор восстанавливает резервный линк.

При включении устройства с STP (или при подключении нового патч-корда) все порты проходят вышеприведенные состояния именно в таком порядке (за исключением состояния Disabled). Ведь при включении на другом конце патч-корда, который только что подключили в порт, может быть коммутатор, на котором может присутствовать потенциальная петля. Вот поэтому порт сначала 15 секунд по умолчанию (Forward Delay таймер) находится в состоянии прослушивания (Listening) - он смотрит BPDU, попадающие в него, выясняет свое положение в сети. Потом порт переходит к обучению еще на 15 секунд (снова таймер Forward Delay) - порт смотрит какие MAC-адреса есть на линке, и записывает в свою таблицу коммутации все MAC-адреса из кадров, поступающих на порт (Learning), и потом, убедившись, что все в порядке, начинает уже свою работу (Forwarding). Итого, мы имеем целых 30 секунд простоя, прежде чем подключенное устройство сможет обмениваться информацией со своими соседями.

Как известно, Root коммутатор отправляет Configuration BPDU каждые 2 секунды (Hello Time). И эти BPDU транслируют все коммутаторы через свои Designated порты. В STP у коммутаторов есть такой таймер, как Max Age. По умолчанию - это 20 секунд (10*Hello Time). Если кадр BPDU имеет время жизни, превышающее максимальное, то кадр игнорируется коммутаторами. Таймер обнуляется (точнее сбрасывается снова до 20 секунд по умолчанию) при получении BPDU. Например, Root Bridge отправил Hello BPDU, он пришел на коммутатор, коммутатор начинает отсчитывать по умолчанию 20, 19, 18 ... Затем, поскольку прошло 2 секунды (Hello Time), Root Bridge снова отправляет Hello BPDU, и при его получении коммутатор начинает отсчитывать заново: 20, 19, 18 ...

Если вдруг коммутатор перестает получать Hello BPDU от Root Bridge, то спустя интервал Max Age, когда отсчет дошел уже до нуля, и если Hello BPDU не пришел на коммутатор, то коммутатор восстанавливает резервный линк. Для этого он заново “проводит выборы” STP: выбор корневого

коммутатора, выбор Root, Designated и Non-Designated портов.

Если бывший Non-Designated порт после выборов станет Root или Designated портом, то из состояния Blocking, он должен будет перейти в Forwarding, через несколько промежуточных состояний согласно всем таймерам. И в конце концов ранее заблокированный порт становится Root или Designated портом, а другой порт блокируется. Итого, если коммутатор перестает получать BPDU, то он не сразу реагирует и не будет сразу же стремиться изменить топологию STP, поэтому у нас будет около 50 секунд простоя до полного перестроения топологии: 20 + 15 + 15 (Max Age + Forward Delay + Forward Delay), что есть, несомненно, долго в современных реалиях.

Однако, Max Age таймер не всегда “вступает в игру”. Например, если падает линк между коммутаторами, и коммутатор видит, что его Root порт теперь недоступен, соответственно он не знает, как теперь добраться до корневого коммутатора, но при этом он продолжает получать BPDU через Non-Designated порт, то он сразу же, не дожидаясь окончания таймера Max Age, меняет роль порта с Non-Designated на Root порт, и проходит через состояния Listening —> Learning —> Forwarding. То есть, процесс занимает 30 секунд вместо 50.

А вот если бы после падения линка перестали приходить BPDU на Non-Designated порт другого коммутатора, то тут как раз бы и “вступил в игру” таймер Max Age и простой в таком случае составит 50 секунд.

Вот поэтому и говорят, что в классическом STP простой составляет от 30 до 50 секунд, в зависимости от ситуации.

Вообще, Max Age по умолчанию - 20 секунд. Однако время, при котором BPDU устаревают, зависит от того, как далеко коммутатор находится от корневого. Когда коммутатор получает BPDU он вычитает значение Message Age из Max Age таймера. Message Age (время жизни сообщения) - количество времени, прошедшее с того момента как Root Bridge отправил BPDU (аналогично полю TTL в IP-пакете). Root коммутатор устанавливает значение 0, и это значение увеличивается на один каждый раз, когда BPDU проходит через какой-то коммутатор на своем пути. То есть, на коммутаторе за корневым Message Age будет 1, на следующем коммутаторе будет 2 и т.д., поэтому на коммутаторе за корневым, BPDU истечет через 20 секунд (20-0=20), на следующем коммутаторе устареет через 19 секунд (20-1=19) и т.д.

Все таймеры, установленные на Root Bridge: Hello Time, Forward Delay, Max Age, определяют таймеры во всей сети (т.е. на всех остальных коммутаторах).

Изменениями топологии в STP считается изменения ролей DP и RP, а именно:

- когда какой-либо порт переходит в состояние Forwarding;
- когда порт в состоянии Learning или Forwarding переходит в состояние Blocking или Disabled.

Если, например, Non-Designated порт станет Designated, то это не послужит триггером для оповещения о смене топологии, по крайней мере не сразу. Потому что, как было сказано ранее, триггером для определения смены топологии служит переход в состояние Forwarding, либо переход из состояний Learning или Forwarding в состояние Blocking или Disabled. А смена ролей с Non-Designated на Designated, это в первую очередь переход из состояния Blocking в состояние Listening, а это не является триггером для отправки сообщений, что топология поменялась.

“Некорневой” коммутатор, который обнаружил изменения в топологии, например, один из портов перешел в состояние Disabled (shutdown). отправляет Topology Change Notification (TCN) BPDU корневому коммутатору через свой Root порт.

Когда мы рассматривали формат TCN BPDU ранее, то убедились, что в нем нет информации о том, что именно и где изменилось, коммутатор просто сообщает что где-то в топологии произошли какие-то изменения.

Отправка TCN сообщения повторяется каждый Hello interval (2 секунды по умолчанию) до тех пор, пока получение сообщения не будет подтверждено, но обычно подтверждение приходит практически мгновенно, поэтому, как правило, коммутатор успевает отправить только один TCN BPDU.

Следующий коммутатор, который получил TCN BPDU, как правило, сперва также отправляет TCN BPDU через свой Root порт, и только потом отправляет назад подтверждение в ответ на предыдущий TCN, который он сам получил. Дело в том, что нужно как можно скорее уведомить корневой коммутатор об изменении топологии, поэтому отправка TCN BPDU имеет приоритет перед подтверждением. Но это зависит опять же от реализации spanning-tree на коммутаторе.

Подтверждение отправляется в следующем Configuration BPDU, которое будет отправлять коммутатор, выставлением флага TCA (Topology Change Acknowledgement).

[STP & SNR]

Рис. 22. Отправка BPDU с флагом TCA

Далее коммутаторы, у которых порт работает в роли DP для сегмента, повторяют первые два шага и отправляют TCN BPDU через свой корневой порт и ждут подтверждения.

После того, как корневой коммутатор получил TCN BPDU, он также отправляет сообщение о получении с флагом TCA, а затем начинает отправлять Configuration BPDU с установленным флагом TC (Topology Change), уже без флага TCA. Корневой коммутатор продолжает отправлять эти сообщения в течение Max Age таймера + таймера Forward Delay (20+15 = 35 секунд по умолчанию).

[STP & SNR]

Рис. 23. Отправка BPDU с флагом TC

Эти сообщения получают все коммутаторы. При получении сообщения Hello BPDU с флагом TC, коммутатор использует таймер (Forward Delay time), для того чтобы обновить записи в таблице коммутации. Для этого после получения сообщения с флагом TC он изменяет время жизни MAC-адресов (MAC Address Aging Timer) со значения по умолчанию (обычно, это 300 секунд, но в зависимости от вендора может быть по-разному) до 15 секунд (как раз на величину таймера Forward Delay). MAC-адреса устройств, которые никак не взаимодействуют ни с кем, удаляются из таблицы коммутации по истечении 15 секунд, а MAC-адреса активных устройств останутся в таблице.

Обновление выполняется из-за того, что после изменений в топологии STP, в таблице коммутации могут храниться неправильные записи. Поэтому, чтобы устройства могли общаться между собой после изменения топологии и используется Forward Delay таймер (15 секунд), в противном случае без немедленной очистки записей придется ждать 5 минут (300 секунд, значение по умолчанию), пока время жизни записей в таблице коммутации не истечет, и коммутатор только тогда узнает за каким портом теперь находится получатель (после перестроения топологии), за счёт отправки unknown-unicast сообщения. Да, сходимость STP будет все также до 50 секунд, однако это лучше, чем невозможность устройств общаться между собой в течение 5 минут.

STP Toolkit

Говоря об STP (впрочем, как и о любой другой сущности в сетевых технологиях), не стоит забывать о безопасности. Дело в том, что в самом алгоритме spanning-tree нет никакой проверки аутентификации или проверки от кого именно пришел BPDU-пакет, чем легко могут воспользоваться злоумышленники. Поэтому в STP есть полезный функционал или расширение, который иногда также называют STP Toolkit (в терминологии Cisco), он позволит обеспечить дополнительную защиту при использовании STP, а также содержит дополнительный функционал, который может быть полезен при определенных условиях.

В него входят:

- Portfast.
- BPDU Guard / BPDU Filter.
- Root Guard / Loop Guard.
- и т.д. (есть еще функционал, который входит в STP Toolkit, но мы рассмотрим наиболее популярный).

1. Portfast.

На коммутаторе при использовании STP можно применить на порту режим portfast. Данный режим позволяет переводить порт сразу же в состояние Forwarding, минуя Listening и Learning (то есть без простоя в 30 секунд). Причем несмотря на это, порт все равно будет слушать приходящие BPDU.

Данный режим можно включать на портах, к которым подключены конечные устройства, и также, на некоторых вендорах (например, на Cisco) это не будет работать на trunk портах (только на access, потому что другие коммутаторы подключаются, как правило, через trunk порты).

На портах, к которым подключаются другие коммутаторы, данный функционал включать крайне не рекомендуется, так как есть риск возникновения петли.

Также риск и есть даже при включении portfast на портах, куда подключены конечные устройства. Что если, кто-то вместо конечного устройства, пусть даже непредумышленно, подключит коммутатор и соединит порты, где было прописано portfast в этот коммутатор. Образуется петля.

2. BPDU Guard / BPDU Filter.

Также на порту можно включить BPDU Guard, и если на данный порт придет STP BPDU, то интерфейс будет заблокирован для предотвращения образования петли (err-disabled). Также функционал может использоваться для защиты от злоумышленников, которые, отправляя BPDU, хотят выдавать себя за Root коммутатор, чтобы перехватывать весь трафик, либо просто хотят сделать сеть неработоспособной из-за бесконечных перестроений, так как коммутаторы будут постоянно получать TCN BPDU и будут реагировать на них.

BPDU Filter имеет схожий функционал, но не отключает порт, а просто отбрасывает все BPDU (как на IN, так и на OUT).

3. Root Guard / Loop Guard.

Если на порту применить функционал Root Guard, то даже при поступлении superior BPDU (с меньшим BID), коммутатор не будет воспринимать данный коммутатор как новый Root Bridge, а просто заблокирует интерфейс. Это позволит поддерживать, точнее даже удерживать топологию STP, как нам необходимо. Функционал будет полезен, например, если какой-либо злоумышленник захочет включить еще один коммутатор в топологию и сделать его корневым, чтобы весь трафик шел через него.

Loop Guard же позволяет не переводить порт в состояние Forwarding, даже если коммутатор перестал получать BPDU на этом порту. То есть коммутатор при отсутствии BPDU может посчитать, что там находится теперь не коммутатор, а, вероятно, какое-то конечное устройство, значит можно разблокировать порт. А на деле, например, по какой-то причине данные на линке перестали передаваться в одну сторону, т.е. линк стал однонаправленный, поэтому BPDU и не проходят. Поэтому если применить Loop Guard на порту, то когда при отсутствии BPDU порт попытается перейти в Forwarding, он будет сразу же заблокирован.

RSTP (802.1w, Rapid Spanning Tree Protocol)

Как мы уже заметили ранее, время сходимости STP было очень большим, даже в контексте того времени. 50 секунд простоя - целая вечность, особенно в масштабе современных сетей. Поэтому нужна была какая-то альтернатива, которая не заставила себя ждать. Так впервые в 2001 году появился протокол RSTP (из названия “Rapid” можно догадаться, что этот протокол должен работать гораздо быстрее).

RSTP впервые был описан в стандарте IEEE 802.1w-2001, затем он перекочевал в IEEE 802.1D-2004, где полностью заменил собой классический STP. В Cisco аналог мультивендорного RSTP называется PVRST+ (Per-Vlan Rapid Spanning Tree) или Rapid PVST+ (RPVST+). Rapid PVST+ в каждом VLAN строит дерево. В каждом VLAN работает RSTP.

RSTP vs STP. Изменения в BPDU, новые роли и состояния портов

Сразу же остановимся на основных отличиях STP и RSTP, а позднее рассмотрим их все подробнее:

Таблица 2. Сравнение ключевых особенностей в работе STP и RSTP

Критерий	STP	RSTP
Кто отправляет BPDU	В уже сложившейся топологии только корневой коммутатор отправляет BPDU, остальные ретранслируют.	Все коммутаторы отправляют BPDU в соответствии с hello-таймером (2 секунды по умолчанию)
Соответствие между состояниями портов в STP и RSTP	- Disabled - Blocking - Listening - Learning - Forwarding	- Discarding - Discarding - Discarding - Learning - Forwarding

Роли портов	- Root (корневой). Участвует в пересылке данных, ведет к корневому коммутатору; - Designated (назначенный). Ведет от корневого коммутатора; - Non-Designated (неназначенный). Не участвует в пересылке данных.	- Root (корневой). Участвует в пересылке данных, ведет к корневому коммутатору. - Designated (назначенный). Ведет от корневого коммутатора. - Alternate (дополнительный). Не участвует в пересылке данных. - Backup (резервный). Не участвует в пересылке данных.
Механизмы работы	Использует таймеры: - Hello (2 секунды); - Max Age (20 секунд); - Forward delay timer (15 секунд).	Использует процесс “Proposal and Agreement” (предложение и соглашение).
Изменение топологии	Коммутатор, обнаруживший изменение топологии, извещает корневой коммутатор. Тот в свою очередь, требует от всех остальных очистить их записи о текущей топологии в течение Forward Delay timer.	Обнаружение изменений в топологии влечет немедленную очистку записей.
	Если “некорневой” коммутатор не получает Hello BPDU от корневого в течение Max Age, он начинает новые выборы.	Начинает действовать, если не получает BPDU в течение 3 hello-интервалов.
	Изменением в топологии считается, когда какой-либо порт переходит в состояние Forwarding, либо когда порт в состоянии Learning или Forwarding переходит в состояние Blocking или Disabled.	Изменением в топологии считается переход порта в состояние Forwarding.
Смена состояний порта	Последовательное прохождение порта через состояния Blocking - Listening (15 сек) - Learning (15 сек) - Forwarding.	Возможность быстрого перехода к Forwarding для p2p и Edge-портов.

Рекомендуемая стоимость порта в RSTP (802.1w) в зависимости от пропускной способности линка (аналогична обновленной стоимости в стандарте 802.1t для классического STP, рассчитывается также по формуле: 20 Tbit/s / bandwidth):

Таблица 3. Рекомендуемая стоимость порта в RSTP в зависимости от пропускной способности

Пропускная способность	Стоимость
4 Mbps	5 000 000
10 Mbps	2 000 000
16 Mbps	1 250 000
100 Mbps	200 000
1 Gbps	20 000
2 Gbps	10 000
10 Gbps	2 000

100 Gbps	200
1 Tbps	20
10 Tbps	2

Состояния портов:

- Discarding - порт только получает BPDU: сам BPDU не отправляет, MAC-адреса не изучает, данные не перенаправляет. Заменяет Disabled, Blocking и Listening состояния из классического STP. Справедливее даже сказать, что Discarding объединил в себе Disabled и Blocking из STP, а Listening просто перестал использоваться и был удален;
- Learning - порт слушает и отправляет BPDU, а также начинает изучать MAC-адреса, данные НЕ перенаправляет;
- Forwarding - порт слушает/отправляет BPDU, участвует в поддержании таблицы MAC-адресов, данные перенаправляет.

Роли портов RSTP:

- Root - тот же корневой порт, что и в классическом STP;
- Designated - тот же назначенный порт, что и в классическом STP;
- Non-Designated делится на следующие:
  - Alternate - альтернативный путь к корневому коммутатору. Это, можно сказать, резервный ("лучший второй", хотя их может быть и несколько) корневой порт. Путь отличается от того, который использует Root порт. Регулярный трафик на порту блокируется. Порт находится в состоянии Discarding и принимает превосходящие (superior) BPDU от другого коммутатора (т.е. либо с меньшим RPC, BID или Port ID).
  - Backup - запасной/резервный путь в сегмент, предоставляющий сегменту линк к корневому коммутатору, большей стоимости (т.е. хуже), чем «Designated» порт. То есть, по сути, резервный ("лучший второй", опять же может быть несколько) Designated порт. Регулярный трафик на порту блокируется. Порт находится в состоянии Discarding и принимает превосходящие (superior) BPDU от другого интерфейса на одном коммутаторе, т.е. BID в BPDU один и тот же. Это происходит только если интерфейсы коммутатора объединены в один домен коллизий, например, через хаб или через общую шину, например, коаксиальный кабель. Однако такие технологии практически не используются в современных сетях, поэтому Backup порт встречается достаточно редко.

В концепции новых резервных портов RSTP и кроется одна из причин быстрого переключения в случае отказа. Вместо того, чтобы искать решение проблемы уже после того, как она случилась (причем делать это очень долго), все “пути отхода” нам уже известны заранее, еще до появления проблемы. Грубо говоря, в случае отказа основного линка, произойдет быстрое переключение на резервный линк, и для этого RSTP не нужно заново просчитывать топологию, он просто переключится на запасной, заранее просчитанный путь.

В RSTP есть только один тип BPDU: он называется “Rapid/Multiple Spanning Tree” (аналог Configuration BPDU в обычном STP) и имеет следующие поля:

- Protocol Identifier (Идентификатор протокола STP, 2 байта): 0x0000.
- Protocol Version Identifier (Версия протокола STP, 1 байт): 2 для RSTP.
- BPDU Type (Тип BPDU, 1 байт): 0x02 для Rapid/Multiple Spanning Tree.
- BPDU flags (Флаги, 1 байт): бит 0 (младший) - TC (Topology Change), бит 1 - Proposal, биты 2-3 - роль порта: 00 (0) - unknown, 01 (1) - Alternate/Backup, 10 (2) - Root port, 11 (3) - Designated port, бит 4 - Learning, бит 5 - Forwarding, бит 6 - Agreement, бит 7 (старший) - TCA (Topology Change Acknowledgement).
- Root Identifier (Идентификатор корневого коммутатора, 8 байт): Root Bridge Priority + Root Bridge System ID.
- Root Path Cost (Расстояние до корневого коммутатора, 4 байта).
- Bridge Identifier (Идентификатор коммутатора, 8 байт): Bridge Priority + Bridge System ID.
- Port identifier (Идентификатор порта, 2 байта).
- Message Age (Время жизни сообщения, 2 байта).
- Max Age (Таймер - максимальное время жизни сообщения, 2 байта).
- Hello Time (Таймер - интервал отправки Hello BPDU, 2 байта).
- Forward Delay (Задержка смены состояний, 2 байта). Минимальное время перехода коммутатора в новое состояние. По умолчанию - 15 секунд. Используется при переходе из состояния Blocking в состояние Listening, и затем из состояния Listening в состояние Learning (используется на shared-портах вместо Proposal/Agreement механизма, либо если сосед поддерживает только классический STP).

[STP & SNR]

Рис. 24. Пример дампа RSTP BPDU

Если на коммутаторе запущен классический STP (802.1D) и если к нему придет RSTP BPDU, то коммутатор не поймет, что с ним делать. Поэтому, если коммутатор, на котором запущен RSTP определит, что на другом конце линка к нему поступает BPDU формата 802.1D, он также начнет работать в режиме классического STP. Это было сделано для обратной совместимости STP и RSTP.

Также можно заметить, что в RSTP нет отдельного BPDU для анонсирования изменений в топологии (Topology Change Notification - TCN BPDU). Вместо этого протокол использует флаг Topology Change (TC) в обычном BPDU, для того чтобы указать на изменения, произошедшие в топологии. Однако, опять же, для обратной совместимости с коммутаторами, которые используют классический STP, коммутаторы, использующие RSTP, могут обрабатывать и генерировать TCN BPDU.

В RSTP, в отличии от классического STP, все коммутаторы генерируют и отправляют свои BPDU через Designated порты через каждый Hello Time. Благодаря этому быстрее может быть обнаружено изменение топологии.

В RSTP при изменении топологии коммутатор отправляет Configuration BPDU с установленным флагом TC, и, подобно классическому STP, его сосед отправляет назад подтверждение в Configuration BPDU с установленным флагом TCA. А затем сообщает об изменении топологии уже своим соседям.

Изменением топологии в RSTP считается только переход non-edge порта в состояние Forwarding (в STP, как мы помним, были немного другие критерии).

Новые типы линков

Ранее в STP, для того чтобы убедиться, что порт может участвовать в передаче данных, требовались таймеры, т.е. коммутатор пассивно ждал в течение назначенного времени, слушая BPDU. Ключевой особенностью RSTP стало введение концепции типов портов, основанных на режиме работы линка: full-duplex или half-duplex, типы портов point-to-point (p2p) или shared соответственно.

К p2p-порту обычно подключается только один коммутатор, т.е. это прямое соединение между двумя коммутаторами. Данные порты всегда работают в режиме full-duplex. Дополнительно не нужно обозначать порт как p2p, коммутатор сделает это самостоятельно. Однако все равно, при желании, можно указать на порту явно, чтобы он был point-to-point.

К shared-порту подключается один или несколько RSTP коммутаторов через хаб, либо одно или несколько устройств через тот же хаб. Shared-порты должны всегда работать в режиме half-duplex для избежания коллизий. Аналогично point-to-point типу порта, shared-порт не нужно настраивать вручную, коммутатор сам определит тип, но опять же, как и в случае с p2p-портом, shared-порт можно задать принудительно.

Также появилось понятие пограничного порта - edge-port (которое также появилось одновременно с этим в стандарте 802.1t), к которому, как правило, подключаются конечные устройства.

Вообще в RSTP есть два типа портов: Edge-port и Non-edge-port. Edge-port - порт, подключенный к конечному устройству (например, ПК, IP-телефон, сервер и т.д.), на котором отсутствует возможность создания петлевой структуры. Non-edge-port - “непограничный” порт, к которому подключаются другие коммутаторы, участвующие в топологии RSTP. Если конечное устройство подключается к коммутатору напрямую, то на коммутаторе edge-порт должен работать в режиме full-duplex и быть point-to-point (то есть порт является одновременно пограничным, и тип линка является point-to-point. Такой тип соединения называется edge p2p). Но если конечные устройства подключаются к коммутатору через хаб, то тип порта на коммутаторе будет также edge, ведь к нему подключаются конечные устройства, но в то же время он будет shared и будет работать в режиме half-duplex, т.к. к нему подключен хаб (такой тип порта будет edge shared).

Как правило, Edge-порты назначаются вручную командой “spanning-tree portfast”, т.е. как в классическом STP настраивается режим portfast на порту. Коммутатор не умеет самостоятельно определять, кто к нему подключен. Конечно, он мог бы ориентироваться на наличие BPDU на этом порту, но по стандарту коммутатор должен обязательно подождать минимум 15 секунд (Forward delay таймер), прежде чем решить, что на его порт так и не пришло ни одно сообщение (а это слишком долго для RSTP), и при подключении устройств, Edge-порты сразу же, без какого-либо согласования, переходят из состояния Discarding в Forwarding, пропуская состояние Listening. Добавление/удаление edge-портов, или изменение их состояний никак не влияет на топологию: никаких перестроений не вызывает.

Shared-порты работают по старой схеме с прохождением через состояния Blocking - Listening -

Learning - Forwarding, и поэтому в случае использования shared-порта преимущества использования RSTP отсутствуют. Однако, как уже было сказано множество раз в этой статье, в современных сетях хабы практически не используются, поэтому навряд ли мы встретим такой тип соединения.

А вот на p2p-портах (т.е. на портах между двумя коммутаторами) RSTP использует процесс предложения и соглашения: Proposal/Agreement. Коммутатор справедливо считает, что если линк работает в режиме полного дуплекса и не обозначен как пограничный, значит на нем только два устройства: он и другой коммутатор. И вместо того, чтобы ждать входящих BPDU, он сам пытается связаться с коммутатором на другом конце линка с помощью специальных Proposal BPDU (обычный RSTP BPDU с установленным флагом Proposal), в которых есть информация о стоимости маршрута к корневому коммутатору. Коммутатор устанавливает флаг "Proposal" в RSTP BPDU для того чтобы предложить себя на роль выделенного (Designated) коммутатора в сегменте. Роль порта в Proposal сообщении всегда установлена в "Designated". Второй коммутатор сравнивает полученную информацию со своей текущей и принимает решение, о чем извещает первый коммутатор посредством Agreement BPDU (тоже обычный RSTP BPDU с установленным флагом Agreement). Коммутатор устанавливает флаг "Agreement" в RSTP BPDU для того чтобы принять предыдущее предложение. Роль порта в Agreement сообщении всегда установлена в "Root".

Рассмотрим данный процесс более детально.

Proposal & Agreement

Допустим, у нас есть некая топология без петель.

[STP & SNR]

Рис. 25. Пример топологии в RSTP без избыточных линков

Мы подключили к коммутатору sw3 два конечных устройства, обозначили порты подключения как edge-порты, и поэтому коммутатор сразу же перевел их в состояние Forwarding.

[STP & SNR]

Рис. 26. Пример топологии в RSTP без избыточных линков с добавлением endpoint устройств

Все работает как надо, все предельно просто.

Теперь представим, что мы добавили еще один коммутатор и подключили его двумя линками в нашу топологию.

[STP & SNR]

Рис. 27. Добавили еще один коммутатор в RSTP топологию

Рассмотрим сначала линк sw1 - sw4.

При добавлении коммутатора в дерево, когда поднимается какой-либо линк, порты на этом линке сразу становятся Designated и переходят в состояние Discarding. Затем коммутатор sw1 отправляет со своего Designated порта (который находится в статусе Discarding) superior BPDU с установленным флагом Proposal. В данном BPDU коммутатор также анонсирует себя как Root коммутатор, сообщает стандартно Root Bridge ID, RPC и т.д. Тем самым коммутатор sw1 предлагает (proposal): “Привет, я корневой коммутатор, предлагаю, чтобы мой порт был Designated, а твой порт должен быть Root портом” (на самом деле, отправить BPDU с установленным флагом Proposal будут оба устройства, но, в конечном итоге, на Proposal будет отвечать inferior устройство, т.к. будет видеть, что полученный BPDU соседа - superior).

[STP & SNR]

Рис. 28. Отправка BPDU с установленным флагом Proposal

Когда коммутатор sw4 на другом конце линка получает данный BPDU, он видит, что это superior BPDU, и он переводит свой порт (на котором он получил BPDU от Root Bridge) из Designated Discarding в Root Discarding.

Помимо всего прочего, когда inferior коммутатор получает Proposal на своем порту, прежде чем отправить какой-то ответ, он начинает процесс синхронизации. Сперва коммутатор переводит все свои non-edge порты в состояние Discarding (edge порты не блокируются!). Это нужно для того, чтобы уберечься от возможного образования петель в момент работы процесса Proposal/Agreement. Non-edge порты переводятся в Discarding именно на inferior коммутаторе.

[STP & SNR]

Рис. 29. Начало процесса синхронизации для линка sw1-sw4

После того, как процесс синхронизации начался, коммутатор sw4 отправит ответ с установленным флагом Agreement в сторону Root Bridge (соглашение на предложение) и переводит свой порт из Root Discarding в Root Forwarding.

[STP & SNR]

Рис. 30. Отправка BPDU с установленным флагом Agreement

Когда корневой коммутатор sw1 получает положительный ответ (agreement) на свое предложение (proposal), он переводит свой порт из Designated Discarding в Designated Forwarding.

[STP & SNR]

Рис. 31. Синхронизация для линка sw1-sw4 закончена

После этого синхронизация для этого линка заканчивается (можно сказать, что синхронизация - это процесс, который инициализирует коммутатор при поступлении к нему BPDU с установленным Proposal битом. В этом процессе коммутатор переводит все non-edge порты в состояние Discarding и отправляет ответ на Proposal, который и запустил этот процесс, и это даже необязательно должен быть Agreement. Процесс заканчивается сразу же, как только коммутатору, который и запустил этот процесс, приходит ответ, и порты переходят в стабильное состояние).

А вот на коммутаторе sw4, где в результате процесса синхронизации мы перевели non-edge порты в Discarding, с этих портов начнут отправляться BPDU с флагом Proposal, т.е. так называемый “разрыв” топологии сместился с линка между коммутаторами sw1 и sw4, под коммутатор sw3, на следующий линк. И для этого линка точно также будет проходить процесс синхронизации, далее процесс сместится на следующий линк и т.д.

Однако, не всегда коммутатор будет отправлять в ответ на Proposal свой Agreement.

[STP & SNR]

Рис. 32. Пример ситуации, в которой не будет отправлен Agreement в ответ на Proposal

Например, в данном случае, у нас на линках sw1-sw4, sw4-sw3 прошел процесс синхронизации и добрался до линка между коммутаторами sw3 и sw2. В ответ на Proposal от коммутатора sw3 коммутатор sw2 не отправит Agreement, т.к. он не согласен с данным предложением. У него уже есть лучший маршрут до корневого коммутатора, и Root порт у него уже выбран. Поэтому коммутаторы решают, что линк между ними будет заблокирован. Роли портов и кто будет заблокирован, выбираются стандартно: один из портов будет Designated Forwarding, а другой Alternate Discarding. На этом процесс Proposal/Agreement заканчивается. У нас получилась новая топология без петель.

[STP & SNR]

Рис. 33. Процесс Proposal/Agreement завершен для новой RSTP топологии

Так как весь процесс теперь не привязан к таймерам, происходит он очень быстро. Как только мы подключили новый коммутатор, он практически сразу вписывается в общую топологию и приступает к работе, весь процесс занимает буквально доли секунд, т.к. опять же коммутаторы не ждут истечения каких-либо таймеров, а обмениваются сообщениями между собой очень быстро (но все зависит, конечно, еще и от топологии сети).

Как уже было сказано ранее, RSTP имеет обратную совместимость с STP.

RSTP в своей работе использует обычные STP таймеры в следующих случаях:

1. К порту подключается устройство, поддерживающее только классический STP. В этом случае порт коммутатора из режима работы RSTP переходит в режим STP. А значит проходит все стандартные для STP стадии: Blocking, Listening, Learning, Forwarding (в зависимости от того, с какими значениями корневого коммутатора и стоимости будет получен BPDU).
2. Коммутатор определил соединение как shared. В этом случае используются стандартные таймеры RSTP. Так как к такому порту подключено два и более коммутатора, задействовать механизм Proposal/Agreement невозможно.
3. Подключено устройство, не участвующее в STP, и на порту не задан тип edge. В этом случае порт пройдёт следующие стадии RSTP: Discarding (15 секунд), Learning (15 секунд), Forwarding. Таким образом, он перейдёт в режим передачи только через 30 секунд.

Итак, вкратце подытожим о принципах работы протокола RSTP в виде следующих тезисов:

1. В RSTP 802.1w по сравнению с классическим STP был изменен формат cost (который был позаимствован из 802.1t, вышедшего в том же году), изменен формат BPDU, был удален тип TCN BPDU, появились типы линков: point-to-point, shared и edge-порт, новые роли портов: Alternate (запасной корневой порт) и Backup (запасной Designated порт), появилось состояние Discarding, которое заменило Disabled, Blocking, Listening;
2. Несмотря на все это, RSTP сохранил обратную совместимость с STP. Если коммутатор на другом конце линка не умеет работать с RSTP, или если тип линка между устройствами shared, то RSTP будет работать по классическому алгоритму 802.1D со всеми таймерами и состояниями. В RSTP корневой коммутатор и роли портов, все выбирается аналогично;
3. Однако ключевое преимущество RSTP как раз-таки в том, что он не привязан к каким-либо таймерам, а работает за счет механизма Proposal/Agreement (на p2p линках);
4. Если коммутатор перестает получать BPDU (которые в RSTP отправляют все коммутаторы в сети, а не только корневой) от соседа в течение трех Hello Time (6 секунд по умолчанию), то он считает, что сосед “мертв”, уведомляет всех своих соседей путем лавинной рассылки BPDU с установленным TC флагом (и соседи делают тоже самое) и запускает перестроение топологии: бывший Alternate порт станет Root портом и запустит механизм Proposal/Agreement для определения состояний портов.

MSTP (802.1s, Multiple Spanning Tree Protocol)

MSTP впервые был описан в стандарте IEEE 802.1s (дополнение к 802.1Q) в 2002 году, а в последствии перекочевал в IEEE 802.1Q-2003.

MSTP работает по тому же принципу, что и RSTP (т.е. со всеми Proposal/Agreement механизмами и прочим), только теперь еще и с учетом VLAN.

Казалось бы, у нас есть усовершенствованный механизм STP (в лице RSTP), который в случае чего срабатывает мгновенно, зачем тогда нам еще нужен какой-то MSTP?

Изначально для защиты сетей с кольцевыми топологиями использовались STP и его модификация RSTP, которые строили единое для всех VLAN покрывающее дерево. Это было просто и удобно с точки зрения эксплуатации, но не позволяло гибко управлять трафиком, разнося разные VLAN по разным физическим каналам. С учетом того, что VLAN - достаточно удобный инструмент, который используется практически во всех современных сетях, такая технология построения дерева для разных VLAN была просто необходима.

К тому же, может быть и такое, что у нас петля существует только физически, а логически петли между коммутаторами нет, потому что на портах выданы разные VLAN, поэтому трафик кольцеваться не будет, но, к сожалению, классических STP/RSTP это не особо интересует, ведь они не умеют работать с VLAN.

Конечно, проблема гибкости была решена в проприетарных Cisco PVST и PVST+, которые создавали отдельный процесс построения покрывающего дерева для каждого VLAN. Данный подход был довольно прост с точки зрения эксплуатации и очень гибок, но, во-первых, это, все-таки, проприетарные протоколы, которые полноценно поддерживаются только на оборудовании Cisco, а, во-вторых, при большом количестве VLAN будет тратиться больше времени на расчет и построение топологии, а также мы можем столкнуться с переутилизацией CPU коммутатора из-за большого количества STP процессов: если, например, у нас существует 3000 штук VLAN на транковом порту, то именно столько BPDU и будет поступать на CPU коммутатора каждый Hello Time.

Так как трафик большинства VLAN использовал идентичные пути, смысла в разделении процессов STP для них не было, соответственно концепция протокола была доработана. Так появился более совершенный протокол Multiple Spanning Tree Protocol (MSTP), который абстрагируется от отдельных VLAN, а использует такие понятия как: instance и region.

Как мы уже сказали, MSTP строит дерево не для каждого VLAN, а для каждого MST Instance (MSTI).

Если переводить дословно с английского, instance - экземпляр, копия, образец, пример. То есть в MSTP, instance - некая сущность, которая может объединять в себе несколько VLAN, и для каждого MSTI будет свое дерево RSTP (своя RSTP топология) и своя копия трафика.

Новая концепция вызывала сложности в эксплуатации, так как было необходимо идентично конфигурировать соответствие VLAN-MSTI на всех коммутаторах. Для упрощения и поддержания обратной совместимости с STP и RSTP была разработана концепция регионов.

Регион MSTP может быть образован из нескольких смежных коммутаторов с одинаковыми MSID (MST

Configuration Identification), можно сказать, с одинаковыми настройками, состоящими из:

- Имя региона MSTP.
- Ревизия конфигурации (некое значение, которое должно меняться при изменении конфигурации. Назначается и меняется вручную, но на практике этот параметр почти не используется и не изменяется).
- Дайджест соответствий VLAN-MSTI (маппинг VLAN к MST Instance).

Грубо говоря, каждый регион объединяет в себе коммутаторы, которые имеют одинаковые соответствия VLAN к Instance.

Соответствие VLAN-MSTI не передается в BPDU, потому что если данные вдруг будут переданы неверно, то коммутаторы будут находиться в разных регионах. Поэтому вместо этого, у коммутаторов условно есть некая таблица маппинга VLAN-MSTI, состоящая из 4096 ячеек (максимального кол-ва VLAN), где каждому VLAN будет соответствовать какой-то MSTI, и коммутаторы будут вычислять MD5-хеш от этого соответствия VLAN-MSTI и отправлять его соседям, и уже это значение будет сравниваться коммутаторами.

Если все параметры MSID (MST Config name, MST Config revision, MST Config digest) совпадают у коммутаторов, значит они будут находиться в одном регионе MSTP, но если хоть один из параметров будет не совпадать, то коммутаторы окажутся в разных MSTP регионах.

Существует два основных взаимодействия в MSTP:

1. внутри региона;
2. между регионами.

Взаимодействие внутри региона MSTP

Внутри каждого MST региона есть два типа instance:

- Internal Spanning Tree Instance (IST Instance).
- Multiple Spanning Tree Instance (MSTI).

IST (Internal Spanning Tree Instance, либо еще можно услышать такое определение как “контрольный instance”) - это специальная копия связующего дерева, которая, по умолчанию, существует в каждом MST-регионе и не может быть удалена. IST присвоен номер 0 (Instance 0). Только IST дерево внутри региона может отправлять и получать кадры BPDU, а также служит для управления топологией внутри этого региона. Все VLAN, настроенные на коммутаторах MST-региона, по умолчанию привязаны именно к IST.

Также, IST определяет Root Bridge для всего региона. Root bridge для IST называется Regional Root Bridge.

Именно посредством Regional Root Bridge в IST передаются кадры BPDU, через которые строится дерево для каждого instance. То есть, для снижения нагрузки на коммутаторы, все BPDU различных MSTI коммутатора объединяются в один BPDU, и рассылаются в instance 0, но не во всех остальных instance.

MSTI (Multiple Spanning Tree Instance) в отличии от IST включает в себя VLAN, которые были добавлены в него статически (вручную), и должен содержать как минимум один VLAN. MSTI существуют только в пределах региона.

VLAN, которые назначаются в MSTI, должны изначально существовать в IST из того же региона MST. Когда мы назначаем VLAN статически в MSTI, коммутатор сперва удаляет этот VLAN из IST, и только затем добавляет его в MSTI. Таким образом, можно назначить VLAN только одному MSTI в регионе.

В MSTP может существовать от 0 до 64 деревьев MSTI, точнее нулевой instance для IST, и от 1 до 64 для MSTI. То есть, всего 65 (ограничение стандарта). Но у каждого вендора это может быть реализовано по-разному, например, на Cisco на некоторых платформах может существовать всего до 16 MSTI (0 - IST, 1-15 - MSTI). Но максимальное кол-во для всех одно - 65.

В пределах одного региона, в плане блокирования портов и сходимости, MSTP повторяет принципы RSTP, на основе которого и работает. То есть в IST (Instance 0) выбирается свой корневой коммутатор внутри региона - Regional Root Bridge**.** Региональный корневой коммутатор выбирается по следующим критериям:

- Lowest External path cost to CIST Root bridge (минимальная стоимости пути до корневого коммутатора всей сети: тот коммутатор, у которого наименьший Root Path Cost до CIST Root Bridge и станет Regional Root Bridge);
- Lowest Regional Bridge Identifier (наименьший BID в регионе);
- Lowest CIST Bridge Identifier (из BPDU, полученного из региона, где находится CIST Root Bridge).

(понятия CIST и CST мы рассмотрим ниже).

Далее, строятся деревья внутри региона для IST и каждого MSTI. Для каждого MSTI точно так же выбирается свой региональный корневой коммутатор (он может совпадать с региональным коммутатором IST, но необязательно, и обычно называется просто, Root коммутатор для этого MSTI, либо Regional Root для этого MSTI, но главным региональным коммутатором все равно будет regional root именно в IST), и строится своя топология без избыточных линков.

Взаимодействие между регионами MSTP

На самом деле, в современных сетях достаточно редко используется концепция с несколькими регионами, по крайней мере в небольших сетях точно нет смысла разделять MSTP на регионы.

Однако все равно разбиение на регионы - это очень важная часть MSTP, и нужно понимать, как происходит взаимодействие между регионами, что мы и рассмотрим далее.

Для этого введем такие понятия как CST и CIST.

CST (Common Spanning Tree) - дерево, соединяющее между собой MST регионы и коммутаторы, на которых не запущен MSTP, а настроен, например, STP, RSTP и т.д.

CIST (Common and Internal Spanning Tree) - полное покрывающее дерево сети, объединяющее в себе CST и IST всех MST-регионов.

На первый взгляд не очень понятно, поэтому рассмотрим схему.

[STP & SNR]

Рис. 34. Пример IST/CST/CIST топологий

На рисунке выше рассмотрено три разных региона. В каждом регионе строятся свои топологии: разные MSTI деревья, ну и, конечно, одно главное дерево внутри региона - IST. Для связи между регионами используется дерево CST (оно выделено красным). Ну а все вместе: IST+CST - это уже будет CIST дерево.

Для защиты топологий соединения различных регионов и отдельных коммутаторов строится общее покрывающее дерево (CST). Каждый регион MSTP представляется для CST как отдельный виртуальный коммутатор, и при взаимодействии с другими регионами для нас существует только дерево CST. Мы не показываем соседям в других регионах, что существует и происходит внутри нашего региона, нашу топологию.

Соответствия VLAN-MSTI (точнее хеш от этих соответствий) также не передаются за пределы региона. Поэтому даже если в соседнем регионе будет точно такое же соответствие, между ними не будет ничего общего, т.к. коммутаторы находятся в разных регионах.

Для взаимодействия с другими регионами с пограничных портов CST будут отправляться BPDU с одинаковым Root Bridge ID, с одинаковым sender Bridge ID, с одинаковым RPC и т.д., даже если BPDU отправляются разными коммутаторами в регионе, поскольку, как уже было сказано, регион представляется в виде одного большого виртуального коммутатора.

И само дерево CST функционирует по протоколу STP/RSTP.

В качестве корневого коммутатора для всей сети (т.е. CST + IST) выбирается CIST/CST Root Bridge (CIST Root Identifier). Выбирается CIST Root Bridge стандартно по наименьшему Bridge ID (либо наименьший приоритет, либо наименьший идентификатор - MAC-адрес).

Если этот CIST Root Bridge будет находиться внутри какого-то региона, то остальные коммутаторы в этом же регионе будут понимать, что среди них всех данный коммутатор будет являться CIST Root Bridge. А коммутаторы в других регионах просто видят данный регион как один виртуальный коммутатор, и этот виртуальный коммутатор и будет являться для них всех корневым (несмотря на то, что в этом регионе есть и другие коммутаторы).

В MSTP BPDU (подробно формат BPDU мы рассмотрим позднее) для CIST Root Identifier существует так называемый “внешний” RPC - “CIST External Root Path Cost”. Он показывает нам, сколько будет стоить добраться до этого единого корневого коммутатора в сети. Соответственно, если этот коммутатор

находится внутри региона, то на всех MST-коммутаторах в этом регионе External Root Path Cost будет равен нулю, т.к. до него не стоит ничего добраться из этого же региона. А если же нам поступил BPDU от соседа (из другого региона), что добраться до CIST Root Bridge будет стоить нам столько-то, то опять же коммутаторы внутри региона ничего не прибавляют к значению External Root Path Cost, а передают его без изменений внутри региона (единственное, что прибавится сама стоимость пограничного интерфейса, на котором получили BPDU).

При передаче информации за пределы региона в BPDU также указывается CIST Regional Root Bridge (CIST Regional Root Bridge = Regional Root Bridge). Как мы уже разобрались, это корневой коммутатор для дерева IST в пределах одного региона. И как мы уже говорили, с какого бы коммутатора в регионе мы не отправляли BPDU за пределы этого региона, Regional Root Bridge всегда будет указан один и тот же.

Таким образом, для одного региона в BPDU, отправляемом соседям за пределы региона с разных коммутаторов, всегда будут одинаковые поля CIST Root Bridge (в BPDU просто Root Identifier), CIST External Root Path Cost (в BPDU просто Root Path Cost), CIST Regional Root ID (в BPDU это просто Bridge Identifier при взаимодействии между регионами). А вот CIST Port ID (Port Identifier в BPDU) может и отличаться, смотря с какого порта будет отправлен BPDU.

В дереве IST, как мы помним, корневым коммутатором является Regional Root Bridge. Расстояние до него в пределах региона указывается в CIST Internal Root Path Cost. Ну а CIST Bridge Identifier - это, по сути, указывается BID в IST того, кто отправил BPDU.

Единственное, что не дублируется в BPDU - Port ID. Поскольку в MSTP отправляется всегда одна BPDU вне зависимости от принадлежности к дереву, поэтому и поле Port ID нет смысла дублировать для разных деревьев.

Формат MSTP BPDU

Формат MSTP BPDU очень схож с форматом RSTP BPDU:

- Protocol Identifier (Идентификатор протокола STP, 2 байта): 0x0000;
- Protocol Version Identifier (Версия протокола STP, 1 байт): 3 для MSTP;
- BPDU Type (Тип BPDU, 1 байт): 0x02 для Rapid/Multiple Spanning Tree;
- BPDU flags (CIST) (Флаги, 1 байт);
- Root Identifier (CIST Root ID) (Идентификатор корневого CIST коммутатора, 8 байт);
- Root Path Cost (CIST External RPC) (Расстояние до корневого CIST коммутатора, 4 байта);
- Bridge Identifier (либо это BID коммутатора, который непосредственно отправил BPDU (например, если коммутатор, не поддерживает MSTP), либо это Regional Root Identifier (при взаимодействии между регионами), 8 байт);
- Port identifier (CIST Port ID) (Идентификатор порта, 2 байта);
- Message Age (Время жизни сообщения, 2 байта);
- Max Age (Таймер - максимальное время жизни сообщения, 2 байта);
- Hello Time (Таймер - интервал отправки Hello BPDU, 2 байта);
- Forward Delay (Задержка смены состояний, 2 байта);
- Version 1 Length (Всегда нулевое значение, 1 байт);
- Version 3 Length (Размер MST Extension, 2 байта);
- MST Extension:
- Configuration Identifier (51 байт):
- MST Config ID format selector (1 байт);
- MST Config name (32 байта);
- MST Config revision (2 байта);
- MST Config digest (16 байт);
- CIST Internal Root Path Cost (расстояние до Regional Root Bridge внутри региона, 4 байта);
- CIST Bridge Identifier (на самом деле, это IST Bridge Identifier, 8 байт);
- CIST Remaining hops (кол-во коммутаторов, которые прошел BPDU между коммутаторами внутри региона. По умолчанию максимальное кол-во - 20, затем BPDU будет отброшен. 1 байт);
- MSTID (M-record, one per-instance) (N*16 байт).

  [STP & SNR]

Рис. 35. Пример дампа MSTP BPDU

Как видим, до начала поля MST Extension, MSTP BPDU очень трудно отличить от RSTP BPDU.

Но, исходя из описанного нами формата BPDU и из скриншота дампа, возникает ряд вопросов. Почему присутствуют такие расхождения: почему в описанном нами (да и не только нами, а много кем в сети Интернет) формате BPDU указывается CIST Root ID, а в дампе - это просто Root Identifier, почему у нас есть CIST External RPC, а по факту в BPDU - это обычный Root Path Cost, и т.д.?

Дело в том, как мы уже говорили, формат MSTP BPDU до поля “MST Extension” является полной копией RSTP BPDU для обратной совместимости с другими версиями протокола, и неудивительно, ведь MSTP основан на работе RSTP (например, между регионами используется уже знакомый нам RSTP со всеми его механизмами). И все, до поля “MST Extension”, по сути является CST и рассматривается это все как взаимодействие между регионами, ведь RSTP работает как раз между регионами. И для взаимодействия между регионами нужен CIST Root Bridge, поэтому описанный в BPDU “Root Identifier” - это не что иное как CIST Root ID, а Root Path Cost до него - это CIST External RPC. При взаимодействии между регионами Bridge Identifier будет на самом деле CIST Regional Root ID, потому что, как мы уже разбирали, при отправке BPDU за пределы региона в качестве BID отправителя будет всегда указан Regional Root ID, ведь BPDU отправляются именно в Instance 0, где и находится regional root. А вот все, что уже находится в “MST Extension” - это взаимодействие коммутаторов в пределах одного региона. Итого, в BPDU поля называются по-старому для обратной совместимости с другими версиями spanning-tree, но несут в себе немного иной смысл. И плюс, значения некоторых полей зависят от того, где передается BPDU - внутри региона или между регионами.

MSID добавляется к MSTP BPDU так, что сохраняется совместимость с STP и RSTP (поскольку сначала BPDU идет полностью идентичным RSTP, то и сохраняется обратная совместимость - STP и RSTP коммутаторы могут принимать и обрабатывать MSTP BPDU). При этом MSTP BPDU, отправленные разными коммутаторами одного региона, воспринимаются смежными STP/RSTP-коммутаторами как RSTP BPDU одного коммутатора. Таким образом кольцевая топология на разных коммутаторах по-прежнему поддерживается и в регионе MSTP сохраняется гибкость управления трафиком.

Совпадение полей MST Config name, MST Config revision, MST Config digest обозначает принадлежность к одному региону.

Роли портов в MSTP

Как мы уже заметили, MSTP по принципу работы очень схож с его прародителем - RSTP (ведь именно на RSTP и основана работа MSTP). Однако, несмотря на их схожесть, разумеется, есть некоторые различия, например, рассмотрим роли портов.

Как и в классическом STP/RSTP в MSTP существуют уже знакомые нам Root, Designated, Alternate и Backup порты. Однако, стоит заметить, что в MSTP роли портов для CIST дерева и для MSTI немного отличаются.

Итак:

1. CIST:

- Root - порт, предоставляющий наименьшую стоимость пути до CIST Root через regional root коммутатор;
- Designated - порт, предоставляющий к подключенному к нему сегменту сети самый дешевый путь до CIST Root коммутатора;
- Alternate - резервный корневой порт;
- Backup - резервный назначенный порт.

2. MSTI:

- Root - порт, предоставляющий наименьшую стоимость пути до regional root коммутатора (регионального в IST/MSTI).
- Designated - порт, предоставляющий к подключенному к нему сегменту сети самый дешевый путь до regional root коммутатора (регионального в IST/MSTI).
- Alternate - резервный корневой порт;
- Backup - резервный назначенный порт.

Также, в MSTP появилась новая роль - Master порт.

- Master - это порт с наименьшей стоимостью пути из региона до CIST Root коммутатора, который находится за пределами данного региона. Иными словами, если CIST Root находится в другом регионе, то в текущем регионе корневой порт на Regional Root Bridge будет master-портом. Порт до CIST Root коммутатора на Regional Root Bridge будет являться master-портом для всех MSTI.
  Master-порт переходит в состояние Forwarding только тогда, когда внутри региона сошлись и синхронизировались все instance. Поэтому если в регионе будут постоянные перестроения топологии, может оказаться так, что регион будет изолирован от всей остальной топологии.
  И также появилась новая роль линков - P2p Bound, или пограничный линк.
  Тип Bound (или Boundary) присваивается тем портам, которые стоят на границе с другим

регионом или другой разновидностью протокола STP, т.е. соединяют один MST регион с другим, или соединяют регион с коммутатором, на котором запущен не MSTP.

Topology change

Изменение топологии внутри какого-либо instance (IST/MSTI) влияет только на этот самый instance в пределах региона. Например, в instance 1 выданы VLAN 11, 12, 13, и произошло изменение топологии в 11 VLAN. В таком случае, все VLAN в instance (11, 12, 13) очистят свои таблицы коммутации.

При изменении топологии между регионами (CST), сообщения об изменении топологии распространяются между всеми регионами, но применяются только к boundary линкам.

Взаимодействие MSTP с проприетарными PVST+/RPVST+

Как и MSTP, проприетарные Cisco протоколы тоже умеют строить топологию для разных VLAN, позволяя балансировать трафик. Однако проблема в том, что это разные стандарты, и к тому же PVST может использоваться только на Cisco оборудовании. И в современном мире, наиболее вероятно, что в большинстве сетей используется оборудование не одного вендора, а самых разных. Поэтому необходимо как-то обеспечить взаимодействие между разными вендорами и их протоколами, как например, MSTP и PVST.

Возможно два варианта развития событий:

1. Root Bridge принадлежит MSTP (более предпочтительный вариант);
2. Root Bridge принадлежит PVST (более сложный и менее предпочтительный вариант).

Первый вариант является более предпочтительным, в то время как второй - более сложным и менее предпочтительным. Но в любом случае, оба варианта имеют право на существование. В сети Интернет можно найти множество статей о том, как можно организовать это взаимодействие, и объясняется, почему первый вариант более предпочтительный. Но тем не менее, везде есть свои нюансы (имеется в виду для оборудования разных вендоров). И в частности, как происходит взаимодействие между коммутаторами SNR и проприетарным PVST+ (Rapid PVST+) и все нюансы, мы рассмотрим позднее, поэтому сейчас не будем на этом останавливаться.

На этом можно завершить обзор протокола MSTP (и в целом обзор мультивендорных spanning-tree протоколов), все равно все нюансы рассмотреть не получится, как бы нам того не хотелось. Информации было очень много, поэтому опять же вкратце подытожим основные моменты:

1. MSTP в отличии от стандартных STP/RSTP умеет работать с VLAN, что позволяет добиться балансировки и гибкого управления трафиком;
2. Однако, в отличии от Cisco PVST+, дерево строится не для каждого VLAN, а для каждого instance. Соответственно появилась необходимость маппинга VLAN-MSTI, а для этого уже была разработана концепция регионов (которая редко где, но все равно используется);
3. Коммутаторы из одного региона должны иметь одинаковые параметры, в том числе тот же маппинг VLAN к MSTI. Внутри региона помимо пользовательских MSTI деревьев, существует основное дерево IST. Именно это дерево отвечает за прием и отправку BPDU со всех MSTI, и корневой коммутатор из этого дерева станет Regional Root Bridge для всего региона;
4. Для связи между регионами, а также коммутаторами, на которых запущен не MSTP, существует дерево CST. В данном случае, регион выступает как единый логический коммутатор. Взаимодействие будет вестись с Regional Root Bridge, а тот уже в свою очередь будет передавать данные всем в своем регионе;
5. Полное дерево CIST состоит из CST и IST. Для CIST точно также выбирается корневой коммутатор, который будет главным во всем сегменте сети;
6. Если CIST Root находится в отличном от нас регионе, то в нашем регионе корневой порт на regional root коммутаторе будет являться master портом - новая роль в MSTP.

Настройка и работа STP на коммутаторах SNR

Теперь, когда мы обладаем всеми необходимыми теоретическими знаниями, мы можем перейти к рассмотрению настройки и работы STP протоколов на коммутаторах SNR.

На всех линейках коммутаторов SNR существует поддержка всех типов мультивендорного spanning-tree: STP, RSTP, MSTP, и работает функционал везде одинаково, вне зависимости от чипсета и серии коммутаторов.

Однако, на серии S5210, в которую входят ТОРП-коммутаторы с российским программным обеспечением, есть свои небольшие нюансы, поэтому сперва мы рассмотрим настройку и работу на основных сериях коммутаторов SNR, а затем отдельно на серии с российским ПО - S5210.

Также есть и нюансы при взаимодействии коммутаторов SNR, где запущены мультивендорные протоколы защиты от петель, с коммутаторами вендора Cisco, где используются проприетарные протоколы.

Все это обязательно рассмотрим, но давайте по-порядку.

Настройка spanning-tree на основных сериях коммутаторов SNR

Настройка STP

По умолчанию на всех основных линейках коммутаторов SNR протоколы защиты от петель, в том числе и spanning-tree, отключены.

Чтобы их включить, необходимо в режиме глобальной конфигурации прописать “spanning-tree”:

[STP & SNR]

Как видим, spanning-tree запустился, причем не STP, а MSTP.

По умолчанию на коммутаторах SNR включается именно MSTP, однако можно выбрать и другой режим, например, STP или RSTP:

[STP & SNR]

После этого в текущей конфигурации увидим такие настройки:

После этого в текущей конфигурации увидим такие настройки:

!

spanning-tree mode stp

spanning-tree

!

Причем даже если бы мы выбрали сперва режим spanning-tree, отличный от MSTP, и только потом включили сам процесс spanning-tree, то у нас бы все равно было написано, что запущен MSTP, хотя на самом деле, в конфигурации и по факту будет тот режим, что мы выбрали.

Если бы был выбран режим MSTP, то это бы не отображалось в конфигурации, т.к. это режим по умолчанию, а все, что настроено по умолчанию, не отображается в конфигурации на коммутаторах SNR (обозначил явно, на всякий случай, т.к. в дальнейшем еще можем с этим столкнуться).

Например, это же касается и приоритета коммутатора. По умолчанию, как мы знаем, используется priority 32768, поэтому он и не отображается в конфигурации.

Но давайте попробуем назначить приоритет вручную, попробуем сделать так, чтобы этот коммутатор стал корневым.

[STP & SNR]

И как помним, приоритет должен быть кратным 4096, мы рассматривали это в формате Bridge ID. Поэтому при попытке назначить значение некратное 4096, коммутатор выдаст ошибку и попросит выбрать корректное значение.

И после этого настройка приоритета появится в конфигурации:

И после этого настройка приоритета появится в конфигурации:

!

spanning-tree mode stp

spanning-tree

spanning-tree priority 4096

!

К слову, после того, как мы включили глобально spanning-tree, он автоматически включается на всех портах.

Чтобы исключить какой-либо порт из spanning-tree домена необходимо в режиме конфигурации порта прописать: “no spanning-tree”.

Соберем простейшую схему из четырех коммутаторов:

[STP & SNR]

Запустим на них STP и посмотрим, что получится.

Связности между коммутаторами нет (пока не соединили их патч-кордами), каждый считает себя корневым, и картина на всех будет примерно такая:

[STP & SNR]

Теперь соединим коммутаторы патч-кордами, для простоты переименуем их в sw1, sw2, sw3, sw4, и посмотрим, как построится дерево STP:

sw1:

[STP & SNR]

sw2:

[STP & SNR]

sw3:

[STP & SNR]

sw4:

[STP & SNR]

Все коммутаторы обменялись BPDU, согласились с тем, что корневым будет коммутатор sw2, так как у него наименьший BID (засчет выставленного вручную priority).

Причем стоит обратить внимание, что MAC-адрес, который является частью BID, это CPU MAC коммутатора, а не VLAN MAC (можно увидеть это в выводе команды ‘show version’).

Давайте еще раз взглянем на схему, проверим, что алгоритм spanning-tree отработал верно.

[STP & SNR]

Должна получиться такая топология. И, как видим, алгоритм spanning-tree на коммутаторах отработал именно так.

На что можно обратить внимание в выводе “sh spanning-tree”? Давайте быстро пробежимся по-порядку по всем полям этого вывода.

- “Standard”. Думаю, здесь и так все понятно, в данном поле указывается, какой именно тип spanning-tree мы используем;
- “Bridge MAC”. Здесь все так же предельно понятно - это MAC-адрес текущего коммутатора;
- “Bridge Times”. Таймеры, используемые spanning-tree;
- “Force Version”. Аналог поля “Protocol Version Identifier” в STP пакете (0 - STP, 2 - RSTP, 3 - MSTP);
- “Self Bridge Id”. Bridge ID текущего коммутатора (priority + MAC);
- “Root Id”. Bridge ID корневого коммутатора (priority + MAC). Может совпадать с “Self Bridge Id”, но тогда значения поля будет: “this switch”;
- “Ext.RootPathCost”. Стоимость пути до корневого коммутатора. В этом поле указывается наибольшее значение RPC, с “некорневого” порта;
- “Root Port ID”. Идентификатор корневого порта коммутатора;
- Далее идет уже сам вывод и распределение (топология) портов в STP. Единственное, что здесь стоит пояснить, на мой взгляд, последние два поля: “DsgBridge” и “DsgPort”. В них указываются Bridge ID (priority + MAC) назначенного коммутатора в сегменте (т.е. того коммутатора, который

в сегменте ближе всего к корневому), и, соответственно, порт на нем. Отсюда нетрудно догадаться, что у корневого коммутатора будет указан его собственный BID и его же Port ID.

Внимательный читатель мог заметить, что в выводе “sh spanning-tree” используются такие поля как “Ext.RootPathCost”, хотя у нас не используется на данный момент MSTP, поэтому о каком внешнем RPC идет речь? Также сама стоимость этого RPC не соответствует стандарту 802.1D, а похожа скорее на 802.1t формат или RSTP. Также присутствует роль Alternate порта, хотя у нас классический STP. Bridge ID может принимать значения от 0 до 61440 и должен быть кратным 4096, т.е. новый формат BID присутствует даже в классическом STP 802.1D, и т.д.

Дело в том, что на коммутаторах SNR по умолчанию используется MSTP, поэтому остальные виды spanning-tree унаследовали некоторые особенности MSTP. Плюс ко всему сделано это для унификации и единообразия, которые позволяют добиться определенной совместимости с другими версиями spanning-tree и с другими вендорами, а также обеспечивает удобство работы, приводя все к одному виду.

Поэтому не нужно переживать, все работает согласно стандартам, хоть и может называться по-другому в классическом STP: ExtRootPathCost - обычный RPC, Alternate порт будет Non-Designated портом в состоянии Blocking. Да и формат path cost всегда можно поменять, для этого используется следующая команда:

Изменение формата cost:

SNR-S2982G-24TE(config)#spanning-tree cost-format ?

dot1d Path cost with dot1d format

dot1t Path cost with dot1t format

По умолчанию, как вы уже поняли, используется dot1t (более современный), поэтому и не отображается в конфигурации, но если выберем dot1d, то конфигурация будет следующая:

!

spanning-tree mode stp

spanning-tree

spanning-tree priority 4096

spanning-tree cost-format dot1d

!

Итак, мы без особого труда собрали схему и настроили классический STP на коммутаторах SNR. Рассматривать настройку и как работает RSTP на коммутаторах SNR нет особого смысла, т.к. для настройки достаточно сменить режим на RSTP: “spanning-tree mode rstp” на всех коммутаторах. Поэтому далее сразу перейдем к настройке MSTP.

Но для начала посмотрим, какие еще есть команды в режиме глобальной конфигурации, связанные с STP.

Команды spanning-tree из режима глобальной конфигурации

[STP & SNR]

1. agreement-check-cancel.

Ключевой особенностью RSTP является механизм Proposal/Agreement. Мы выше подробно рассмотрели этот механизм в соответствующем разделе в RSTP. Однако стоит упомянуть, что существует два режима работы данного механизма:

- Common mode;
- Enhanced mode.

Common mode соответствует режиму работы, который мы описывали ранее. И поддерживается он

как в RSTP, так и в MSTP.

А вот enhanced mode, как правило, можно встретить только в MSTP, и работает он немного иначе.

В enhanced mode сначала точно так же, как и в common mode, вышестоящее устройство (которое генерирует superior BPDU) отправляет предложение нижестоящему (inferior) устройству с установленным флагом Proposal, указывая, что порт, к которому подключается нижестоящее устройство, “хочет” как можно скорее перейти в состояние Forwarding. После этого нижестоящее устройство устанавливает порт, на котором получило BPDU, как корневой и блокирует все свои non-edge порты. Далее вышестоящее устройство отправляет второй BPDU, уже с флагом Agreement, и после этого нижестоящее устройство отвечает ему тоже BPDU с флагом Agreement. И тогда вышестоящее устройство переводит свой порт в Designated Forwarding, и на этом Proposal/Agreement заканчивается.

На коммутаторах SNR существует только enhanced метод, причем работает он и для MSTP, и для RSTP. И для совместимости с оборудованием, которое поддерживает и работает в режиме common mode, и используется функционал “agreement-check-cancel”.

Функционал, грубо говоря, отключает проверку флага Agreement, прежде чем отправить свой ответ. Поскольку на коммутаторах SNR используется enhanced метод, он будет ждать BPDU с флагами и Proposal, и Agreement, и только тогда будет отправлять свой Agreement в ответ. Но если он не получит Agreement, то не отправит свой ответ, соответственно механизм Proposal/Agreement будет невозможен, и сеть будет работать по алгоритмам классического STP. Но если применить данную команду, то коммутатор сможет отправить Agreement, даже если пришел Proposal без Agreement (т.е. как в классическом common mode).

По умолчанию функционал agreement-check-cancel отключен.

```
SNR-S2982G-24TE(config)#spanning-tree agreement-check-cancel ?

  <cr>
```

2. cost-format.

Уже упоминали ранее такую команду. Она позволяет выбрать формат path cost: классический укороченный (из стандарта 802.1D - 16 бит), или новый (из стандарта 802.1t - 32 бита).

По умолчанию используется формат dot1t.

```
SNR-S2982G-24TE(config)#spanning-tree cost-format ?

  dot1d  Path cost with dot1d format

  dot1t  Path cost with dot1t format

SNR-S2982G-24TE(config)#spanning-tree cost-format dot1t ?

  <cr>
```

3. forward-time.

Установка значения таймера “Forward Delay Time” (задержка смены состояний). Как мы помним, данный таймер определяет время, как долго Root или Designated порты коммутатора будут оставаться в состоянии Listening и Learning.

Значение таймера по умолчанию - 15 секунд.

```
SNR-S2982G-24TE(config)#spanning-tree forward-time ?

  <4-30>  Delay time <4-30>s

SNR-S2982G-24TE(config)#spanning-tree forward-time 12 ?

  <cr>

SNR-S2982G-24TE(config)#spanning-tree forward-time 12

SNR-S2982G-24TE(config)#
```

Однако может возникнуть и такая ошибка при настройке:

SNR-S2982G-24TE(config)#spanning-tree forward-time 10

ERROR: 2*(Bridge Forward Delay - 1) should be no less than Bridge Max Age!

Данная ошибка говорит нам о том, что значение Forward Delay Timer, которое мы указываем, нельзя настроить, потому что иначе 2*(Bridge Forward Delay - 1) будет меньше, чем значение таймера Max Age. В противном случае мы не дождемся смены состояний, а BPDU просто устареет и будет отброшен.

4. hello-time.

Тут все просто. Это настройка интервала отправки Hello BPDU.

Значение по умолчанию - 2 секунды.

SNR-S2982G-24TE(config)#spanning-tree hello-time ?

<1-10> Time value <1-10>s

SNR-S2982G-24TE(config)#spanning-tree hello-time 3 ?

<cr>

5. max-hop.

Установка значения счетчика Max-Hop в MSTP, который определяет какое количество коммутаторов может пройти BPDU, до того как будет отброшен.

Количество хопов по умолчанию - 20.

SNR-S2982G-24TE(config)#spanning-tree max-hop ?

<1-40> Max hop value <1-40>

SNR-S2982G-24TE(config)#spanning-tree max-hop 21 ?

<cr>

6. maxage.

Это максимальное время жизни сообщения. Данный таймер показывает, когда текущий Hello BPDU устареет и должен быть удален.

По умолчанию - это 20 секунд.

SNR-S2982G-24TE(config)#spanning-tree maxage 22

SNR-S2982G-24TE(config)#

Этот таймер также связан с таймерами Hello time и Forward Delay time. И чтобы MSTP работал корректно, должны выполняться следующие условия:

SNR-S2982G-24TE(config)#spanning-tree maxage 25

Bridge Max Age should be no less than 2*(Bridge Hello Time + 1) and no greater than 2*(Bridge Forward Delay - 1)!

7. mode.

Мы уже рассмотрели эту команду - она позволяет выбрать режим spanning-tree (классический STP, RSTP или MSTP, который после включения spanning-tree всегда выбран по умолчанию).

SNR-S2982G-24TE(config)#spanning-tree mode ?

mstp MSTP(802.1s)

rstp RSTP(802.1w)

```
  stp   STP(802.1D)

SNR-S2982G-24TE(config)#spanning-tree mode rstp ?

  <cr>
```

8. mst.

Данная команда позволяет нам:

a) перейти к настройке регионов MSTP на коммутаторе:

```
SNR-S2982G-24TE(config)#spanning-tree mst ?

  <0-64>     Instance value

  configuration  Region config

SNR-S2982G-24TE(config)#spanning-tree mst configuration ?

  <cr>

SNR-S2982G-24TE(config)#spanning-tree mst configuration

SNR-S2982G-24TE(config-mstp-region)#?

MSTI configuration commands:

  show        Show running system information
  abort       Exit mstp config mode without saving
  exit        Exit mstp config mode after saving
  instance    Set vlan map to instance
  name        Set mstp region name
  no          Negate a command or set its defaults
  revision-level  Set mstp revision level
```

Здесь мы можем настроить маппинг vlan-instance:

```
SNR-S2982G-24TE(config-mstp-region)#instance ?

  <0-64>  Instance value

SNR-S2982G-24TE(config-mstp-region)#instance 1 ?

  vlan  Vlan

SNR-S2982G-24TE(config-mstp-region)#instance 1 vlan 10-15 ?

  <cr>

SNR-S2982G-24TE(config-mstp-region)#instance 1 vlan 10-15

SNR-S2982G-24TE(config-mstp-region)#
```

Настроить revision-level при желании (хотя, как я уже говорил, зачастую этот параметр не используется):

```
SNR-S2982G-24TE(config-mstp-region)#revision-level ?

  <0-65535>  Level <0-65535>

SNR-S2982G-24TE(config-mstp-region)#revision-level 2 ?

  <cr>
```

Чтобы выйти из настроек региона без сохранения изменений, используется команда ‘abort’.

b) настроить приоритет коммутатора для определенного instance:

```
SNR-S2982G-24TE(config)#spanning-tree mst ?

  <0-64>     Instance value
```

configuration Region config

SNR-S2982G-24TE(config)#spanning-tree mst 1 ?

priority Bridge priority

SNR-S2982G-24TE(config)#spanning-tree mst 1 priority ?

<0-61440> Priority value <0-61440>

SNR-S2982G-24TE(config)#spanning-tree mst 1 priority 4096 ?

<cr>

Единственный нюанс: данный instance должен уже существовать на коммутаторе (то что мы как раз создавали в предыдущем подпункте в конфигурации региона), иначе команда не применится.

9. priority.

Если в предыдущем пункте была возможность настроить приоритет для какого-то определенного instance, т.е. для MSTP. То данная команда позволяет настроить приоритет для всего моста (т.е. для STP, RSTP, но также и для MSTP - приоритет будет задан для instance 0, т.к. это instance по умолчанию).

Значение приоритета по умолчанию - 32768.

SNR-S2982G-24TE(config)#spanning-tree priority ?

<0-61440> Priority value <0-61440>

SNR-S2982G-24TE(config)#spanning-tree priority 4096

SNR-S2982G-24TE(config)#

10. process.

Данный функционал используется в MSTP и позволяет разбить MSTP на несколько независимых процессов. В каждом процессе могут быть задействованы свои порты, и построена своя топология. Иными словами, очень гибкий функционал управления MSTP.

По умолчанию на коммутаторе только один процесс - нулевой.

SNR-S2982G-24TE(config)#spanning-tree process ?

<1-31> Process Value <1-31>

SNR-S2982G-24TE(config)#spanning-tree process 1

SNR-S2982G-24TE(config-mstp-process-1)#?

commands:

show Show running system information
end End current mode and change to EXEC mode
exit End current mode and down to previous mode
help Description of the interactive help system
no Negate a command or set its defaults

spanning-tree Spanning-tree config

SNR-S2982G-24TE(config-mstp-process-1)#spanning-tree ?

forward-time Forward delay time

hello-time Hello time

max-hop Max hop

maxage Max age

mode Set MSTP mode

mst Mstp instance

priority Bridge priority

process Mstp process

tc-notify TC notification

tcflush Set global top change flush method

transmit-hold-count Transmit hold count

11. tcflush.

Установка режима перестроения топологии spanning-tree. Как было рассмотрено ранее, при перестроении топологии происходит очистка таблицы MAC-адресов. Да, это очень важно, однако не всегда и не во всех сетях требуется постоянно очищать записи при каждом изменении топологии.

Поэтому это можно настроить:

SNR-S2982G-24TE(config)#spanning-tree tcflush ?

disable Disable flush on top change

enable Enable flush each top change

protect Flush limit mode

disable - FDB не сбрасывается при перестроении топологии;

enable - FDB сбрасывается при перестроении топологии;

protect - FDB сбрасывается не чаще чем раз в 10 секунд при перестроении топологии.

По умолчанию настроен режим enable.

SNR-S2982G-24TE(config)#spanning-tree tcflush protect ?

<cr>

12. transmit-hold-count.

Установка максимального количества BPDU, отправляемых в течение интервала Hello Time.

SNR-S2982G-24TE(config)#spanning-tree transmit-hold-count ?

<1-255> Transmit hold count value <1-255>

SNR-S2982G-24TE(config)#spanning-tree transmit-hold-count 20 ?

<cr>

Значение по умолчанию - 200.

Посмотрим теперь, что есть интересного в режиме конфигурации порта.

Команды spanning-tree из режима конфигурации порта

[STP & SNR]

1. spanning-tree.

При применении просто команды ‘spanning-tree’ в режиме конфигурации порта ничего не произойдет, потому что, как мы уже отметили ранее, по умолчанию, при включении spanning-tree в глобале, STP включается на всех портах коммутатора. То есть, ‘spanning-tree’ на порту - команда по умолчанию. А вот командой ‘no spanning-tree’ мы можем исключить порт из STP процесса. Таким образом, BPDU не будут отправляться на OUT в данный порт (т.к. порт исключен из STP домена). А если BPDU все-таки придет на порт, то он не будет обработан, а будет отброшен. А сам порт будет сразу же переведен в состояние Forwarding.

2. binding-process.

При просмотре настроек глобальной конфигурации мы упоминали о том, что в MSTP можно создать несколько независимых процессов. И каждый порт можно привязать к какому-либо процессу, что и делает данная команда.

По умолчанию порт не привязан к какому-либо процессу.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree binding-process ?

<1-31> Process Value <1-31>

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree binding-process 1 ?

link-share Link-share port

<cr>

3. bpdu.

По умолчанию BPDU отправляются нетегированными: без тега если access порт, или в native vlan, если порт в trunk.

Данная команда позволяет отправлять BPDU с порта в определенном VLAN (с тегом, даже если сам порт находится в режиме access).

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree bpdu ?

vlan Vlan id

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree bpdu vlan ?

<1-4094> Vlan id value <1-4094>

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree bpdu vlan 10 ?

<cr>

4. cost.

Мы обсуждали ранее, что значения костов прописаны в стандарте, но их можно и изменять, как раз данной командой.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree cost ?

<1-200000000> Path cost value (dot1t <1-200000000>)(dot1d <1-65535>)

Здесь мы видим, что можно задать значение как из стандарта dot1d, так и из dot1t.

По умолчанию кост равен значению согласно пропускной способности порта, т.е. согласно стандарта.

Однако на коммутаторах SNR есть особенность, которая заключается в том, что на портах с пропускной способностью 10/100 мбит/с, по умолчанию значение согласно стандарту будет только в режиме работы линка half-duplex. Если режим работы линка будет full-duplex, то значение коста будет меньше на единицу. Это актуально и для старого (dot1d), и для нового формата (dot1t).

На таблице ниже это наглядно продемонстрировано. Найти эту таблицу можно также в наших command guide.

[STP & SNR]

5. digest-snooping.

Включение/выключение аутентификации spanning-tree на порту.

В протоколе MSTP используется концепция регионов. Мы уже рассматривали ранее, какие должны быть одинаковые параметры коммутаторов, чтобы они оказались в одном регионе:

- Имя региона MSTP;

- Ревизия конфигурации;
- Дайджест соответствий VLAN-MSTI (маппинг VLAN к MST Instance).

Но коммутаторы не передают и не сравнивают соответствие VLAN-MSTI, имя и т.д. Вместо этого они будут генерировать MD5-хеш от этих соответствий с публичным ключом аутентификации, этот хеш вместе с ключом будет передаваться соседям, и уже это значение будет сравниваться коммутаторами.

Однако некоторые вендоры не используют публичный ключ аутентификации, отсюда возникает несовместимость устройств, и они окажутся в разных регионах. Благодаря команде ‘spanning-tree digest-snooping’ на порту, будет копироваться хеш STP-соседа, находящегося за данным портом. И тогда устройства смогут оказаться в одном регионе.

По умолчанию аутентификация порта соседа не используется.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree digest-snooping ?

<cr>

6. format.

Установка формата BPDU.

standard - стандарт IEEE;

privacy - CIsco-совместимый формат;

auto - автоматическое определение формата по поступающим BPDU.

По умолчанию выбран формат auto.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree format ?

auto Auto recognized format

privacy Privacy format

standard IEEE Standard format

<cr>

7. link-type p2p.

Выбор механизма определения типа подключенной к порту сети.

auto - автоматическое определение типа соединения;

force-true - всегда point-to-point;

force-false - всегда shared.

По умолчанию тип соединения auto.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree link-type ?

p2p Point-to-point config

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree link-type p2p ?

auto Auto detect link type

force-false Force to non-point-to-point link

force-true Force to point-to-point link

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree link-type p2p force-true ?

<cr>

8. loopguard.

Функционал loopguard позволяет не переводить порт в состояние Forwarding, даже если коммутатор перестал получать BPDU на этом порту. При настройке loopguard на порту, если при отсутствии BPDU порт попытается перейти в Forwarding, он будет сразу же заблокирован.

По умолчанию loopduard отключен.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree loopguard ?

<cr>

9. mcheck.

Отслеживание режима подключенной сети и переключение на STP при необходимости.

По умолчанию данный функционал отключен и на порту используется MSTP. Если прописать данную команду на порту, то при получении на порту BPDU формата классического STP 802.1D, например, порт будет переведен в режим STP, но в противном случае, порт будет продолжать работать как MSTP порт.

Команда по умолчанию отключена. Может быть применена только при работающем MSTP, но не классическом STP.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree mcheck ?

<cr>

10. mst.

Настройки порта в определенном mst.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree mst ?

<0-64> Instance value

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree mst 0 ?

cost Path cost

loopguard Set loopguard port

port-priority Port priority

rootguard Set rootguard port

По умолчанию все находится в нулевом mst.

11. port-priority.

Установить приоритет порта вручную.

Значение по умолчанию - 128.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree port-priority ?

<0-240> Priority value <0-240>

Причем необходимо, чтобы значение было кратно 16, т.е. все как в стандарте. Если попытаемся выбрать другое значение - получим ошибку.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree port-priority 10 ?

<cr>

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree port-priority 10

ERROR: incorrect port priority 10.

Please check whether its value is between 0 and 240, and increased with step 16.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree port-priority 64

SNR-S2982G-24TE(config-if-ethernet1/0/5)#

12. portfast.

Здесь все предельно понятно: команда используется на портах подключения конечных устройств. Делается это для того, чтобы порт переходил сразу же в состояние Forwarding, минуя все промежуточные состояния.

По умолчанию portfast выключен.

Также с помощью данной команды можно активировать на порту либо функционал bpdufiler, либо bpduguard, которые мы рассматривали также ранее.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree portfast ?

bpdufilter Set edge port filter bpdu

bpduguard Set edge port guard bpdu

<cr>

Bpdufiler будет отбрасывать все BPDU (как на IN, так и на OUT), а bpduguard будет блокировать порт при поступлении BPDU на него. Можно даже задать время восстановления порта после этого. Аналогично errdisabled recovery:

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree portfast bpduguard ?

recovery Auto recovery

<cr>

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree portfast bpduguard recovery ?

<30-3600> Recovery interval <30-3600>s

13. rootguard.

Если применить данный функционал на порту, то даже при поступлении superior BPDU (с меньшим BID), коммутатор не будет воспринимать данный коммутатор как новый Root Bridge, соответственно порт делать корневым и пересчитывать топологию он не будет, а просто заблокирует интерфейс.

По умолчанию функционал отключен.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree rootguard ?

<cr>

14. tcflush.

Функционал аналогичен функционалу в глобале, только в данном случае это установка режима перестроения топологии spanning-tree для порта. То есть, необязательно применять это вообще на все порты.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree tcflush ?

disable Disable flush on top change

enable Enable flush each top change

protect Flush limit mode

15. tcnguard / tc-guard.

В протоколе STP есть BPDU, которые рассылаются устройствами в случае обнаружения изменения в сети. Эти BPDU называются Topology Change Notification - TCN.

Некоторые сетевые инженеры не хотят, чтобы эти BPDU рассылались на каких-нибудь участках сети, чтобы сеть не перестраивалась и снова сходилась.

Поэтому есть фильтрация таких пакетов - tcnguard.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree tcnguard ?

<cr>

В протоколе RSTP нет отдельных пакетов BPDU TCN. Однако в самом BDPU есть флаг TC. Значение, которое установлено в этом флаге, сообщает об изменении в сети. Для фильтрации таких пакетов есть tc-guard.

SNR-S2982G-24TE(config-if-ethernet1/0/5)#spanning-tree tc-guard ?

<cr>

И tcnguard, и tc-guard могут применяться только на выход с интерфейса (на OUT). На вход (на IN) фильтровать TCN/TC-flag невозможно, можно только с помощью различных ACL.

Теперь, когда мы рассмотрели все команды, можем переходить к рассмотрению MSTP на коммутаторах SNR.

Настройка MSTP

Топология у нас осталась та же самая, что и была при настройке STP.

[STP & SNR]

Только теперь у нас везде настроен MSTP.

Посмотрим, как сошлось дерево spanning-tree.

sw1:

[STP & SNR]

sw2:

[STP & SNR]

sw3:

[STP & SNR]

sw4:

[STP & SNR]

Что изменилось по сравнению с выводом классического STP?

- версия протокола сменилась на MSTP (“Force Version: 3”);
- появилось разделение на instance;
- появились разделение на External RPC и Internal RPC, и с ними соответственно добавились поля Int.RootPathCost и IntRPC;
- появился идентификатор регионального коммутатора;
- появилось отображение принадлежности портов к instance.

[STP & SNR]

Согласно выводам со всех коммутаторов они верно определили CIST Root Bridge, он же будет Regional Root Bridge, с чем коммутаторы также согласились. Коммутаторы попали в единый регион, ведь настройки у них у всех одинаковые, дефолтные. Поэтому ExtRPC у всех равен нулю, а работает здесь только IntRPC. Потому что, опять же, все находятся в одном регионе.

Теперь немного изменим конфигурацию так, чтобы коммутатор sw2, точно так же был CIST Root, но находился за пределами региона, и посмотрим, что изменится.

sw1:

[STP & SNR]

sw2:

[STP & SNR]

sw3:

[STP & SNR]

sw4:

[STP & SNR]

Давайте разберемся, что же тут вообще произошло.

Для начала посмотрим настройки, которые мы прописали на sw2:

spanning-tree mst configuration

instance 0 vlan 1-9;21-4094

instance 1 vlan 10-20

exit

!

spanning-tree

spanning-tree mst 0 priority 4096

!

!

!

!

!

!

!

vlan 1;10-50

!

Interface Ethernet1/0/1

switchport access vlan 10

То есть мы просто, создали еще один instance - MSTI1, тем самым изменив настройки региона, поэтому этот коммутатор и отделился от остальных.

Появился вывод информации для нового instance, и здесь точно так же описывается: кто является root bride, у кого какой идентификатор, cost и т.д.

Ну и поскольку мы ассоциировали порт 1/0/1 с VLAN 10, у нас в выводе “sh spanning-tree” порт 1/0/1 стал числиться сразу в нескольких instance.

Но MSTI1 нас не особо интересует, он был добавлен скорее для теста. Смотрим, что происходит в IST0.

Поскольку у коммутатора sw2 по-прежнему самый низкий приоритет, он будет являться Root Bridge, точнее CIST Root Bridge. И поскольку он теперь находится в отдельном регионе, и он там один, он же и будет для своего региона Regional Root Bridge. И все косты соответственно у него по нулям, ведь он

является CIST и Regional Root.

Что касается IST0 для первоначального региона, CIST Root все признали sw2, что несомненно верно, не думаю, что тут нужны какие-то комментарии.

А вот Regional Root Bridge был выбран коммутатор sw1. Почему так?

Ранее мы рассматривали принцип выбора регионального коммутатора. Повторим еще раз, чтобы далеко не листать:

- Lowest External path cost to CIST Root bridge (минимальная стоимости пути до корневого коммутатора всей сети: тот коммутатор, у которого наименьший Root Path Cost до CIST Root Bridge и станет Regional Root Bridge);
- Lowest Regional Bridge Identifier (наименьший BID в регионе);
- Lowest CIST Bridge Identifier (из BPDU от региона с CIST Root Bridge).

Коммутатор sw1 выигрывает по косту и маку, именно поэтому он и выбран Regional Root Bridge.

И все остальные коммутаторы в регионе знают, как до него добраться, они указывают стоимость в поле “IntRPC”.

Но на самом деле, логически все выглядит как коммутатор sw1 подключенный к коммутатору sw2 (ведь мы помним, что коммутаторы одного региона представляют из себя один большой виртуальный коммутатор).

Такая схема маловероятна в современных сетях, но все же возможна. Мы просто рассмотрели простейшую настройку MSTP, а также как все будет выглядеть, если коммутаторы будут в одном регионе, или в разных.

Чтобы коммутаторы были в одном регионе, внимательно сверяйте конфигурацию, настройки региона должны совпадать.

Также подробный пример настройки MSTP на коммутаторах SNR у нас описан здесь в нашей базе знаний. Рекомендуем также ознакомиться.

Взаимодействие SNR с проприетарными PVST+/RPVST+

В ходе проведения тестов и исследований совместимости мультивендорных STP/RSTP/MSTP, которые используются на SNR, с проприетарными Cisco протоколами PVST+/RPVST+ было установлено, что корректная сходимость данных протоколов будет только в следующих сценариях:

1. PVST+/RPVST+ (Cisco) с STP/RSTP/MSTP (SNR), но корневым коммутатором должен быть именно SNR;
2. MST (Cisco) с STP/RSTP/MSTP (SNR).

Во всех остальных случаях возможны проблемы в работе: неправильный расчет костов, меняющиеся статусы портов и т.д.

Причина заключается в том, что Cisco отправляет два типа BPDU:

1. PVST BPDU на MAC-адрес 01:00:0C:CC:CC:CD. Кол-во отправляемых BPDU соответствует кол-ву VLAN на порту. Такие сообщения коммутаторы SNR пропускают прозрачно через себя, потому что не могут обработать их из-за проприетарности протокола;
2. В untagged VLAN (по умолчанию VLAN 1) также отправляется BPDU мультивендорного STP (если на Cisco настроен PVST) или RSTP (если на Cisco настроен RPVST) на MAC-адрес 01:80:C2:00:00:00. Отправляется данный BPDU в одном экземпляре. Это сообщение и должно в теории создавать совместимость между протоколами. Однако Cisco в своём Bridge ID (а если считает себя Root Bridge, то и в Root Bridge ID) использует Extended BID, т.е. добавляет значение VLAN в Extended System ID (как описано в стандарте 802.1t), и тогда поле priority в BID не будет кратным 4096. В этом случае SNR не может корректно обработать такой пакет, и STP начинает функционировать непредсказуемо.

В итоге, именно поэтому мы имеем “прыгающие” значения костов, или то, что Cisco в качестве корневого коммутатора периодически признает себя Root Bridge, но потом отказывается от этого, потом снова говорит, что он корневой, хотя блокирует свои порты и устанавливает cost невероятных размеров, и т.д.

Можно, конечно, использовать заглушку на портах коммутаторов SNR, которая будет отбрасывать BPDU от Cisco:

mac-access-list extended PVST

deny any-source-mac host-destination-mac 01-00-0c-cc-cc-cd

exit

!

Interface Ethernet1/0/X

mac access-group PVST in

Но это решает только одну проблему - прозрачную передачу PVST BPDU через SNR, но никак не поможет с обработкой пакетов, в которых используется Extended System ID.

Или можно попробовать изменить формат, отправляемых BPDU, но вероятнее всего, это не даст никакого результата.

Также все зависит от схемы. В какой-то схеме да, все может работать как часы, а в какой-то могут возникнуть вышеописанные проблемы, и сеть будет нестабильна.

Поэтому будьте внимательнее при настройке, планируйте свою сеть именно по принципам, описанным выше. Сценарии описанные выше - рабочие кейсы, все остальное - не рекомендуется.

Настройка spanning-tree на серии S5210

На данный момент по умолчанию на коммутаторах серии S5210 spanning-tree уже включен (в отличии от остальных линеек коммутаторов SNR). И включен по умолчанию режим RSTP, а не MSTP.

Поэтому если вам по какой-то причине не нужен spanning-tree на S5210, то его необходимо отключить глобально командой ‘spanning-tree shutdown’, либо исключить только какой-то конкретный порт из STP домена командой ‘spanning-tree disable’ в режиме конфигурации порта.

Можно заметить, что формат CLI на S5210 немного отличается от привычного нам Cisco-like CLI, но на самом деле это все тот же формат, просто с небольшими отличиями. Да и весь функционал работает аналогично.

Рассмотрим, какие команды здесь есть для настройки spanning-tree.

Большинство команд мы уже рассматривали на основной линейке коммутаторов SNR, но тем не менее еще раз пройдемся по всем командам, здесь могут быть некоторые отличия.

Команды spanning-tree из режима глобальной конфигурации

[STP & SNR]

1. enable.

Данная команда включает глобально spanning-tree. На данный момент, по умолчанию spanning-tree на коммутаторах данной серии и так включен, поэтому эта команда в стандартной конфигурации не будет иметь никакого эффекта.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree enable ?

<cr>

2. errdisable-timeout.

Данная команда является аналогом recovery на обычных коммутаторах SNR, когда порт переходит в состояние disable, например, после работы функционала bpduguard.

По умолчанию функционал отключен. При включении время восстановления по умолчанию равняется 5 минутам (300 секунд).

SNR-S5210G-24TX-UPS-R(config)#spanning-tree errdisable-timeout ?

enable enable the timeout mechanism for the port to be enabled back

```
  interval  interval after which port shall be enabled
SNR-S5210G-24TX-UPS-R(config)#spanning-tree errdisable-timeout enable ?
  <cr>
SNR-S5210G-24TX-UPS-R(config)#spanning-tree errdisable-timeout interval ?
  <10-1000000>  errdisable-timeout interval in seconds
```

3. forward-time.

Настройки данного таймера и остальных аналогичны тем, которые мы рассматривали ранее.

Установка значения таймера “Forward Delay Time” (задержка смены состояний).

Значение таймера по умолчанию - 15 секунд.

```
SNR-S5210G-24TX-UPS-R(config)#spanning-tree forward-time ?
  <4-30>  forward delay time in seconds
SNR-S5210G-24TX-UPS-R(config)#spanning-tree forward-time 16
SNR-S5210G-24TX-UPS-R(config)#
```

Можно отменить установку таймера командой: ‘no spanning-tree forward-time’ - она вернет значение по умолчанию.

4. hello-time.

Настройка интервала отправки BPDU.

Значение по умолчанию - 2 секунды.

```
SNR-S5210G-24TX-UPS-R(config)#spanning-tree hello-time ?
  <1-10>  seconds <1-10> - Hello BPDU interval
SNR-S5210G-24TX-UPS-R(config)#spanning-tree hello-time 4
SNR-S5210G-24TX-UPS-R(config)#
```

Можно отменить установку таймера командой: ‘no spanning-tree hello-time’ - она вернет значение по умолчанию.

5. instance.

В данном режиме мы можем настраивать bridge priority для разных instance на коммутаторе.

Естественно, по умолчанию приоритет для всех - 32768.

```
SNR-S5210G-24TX-UPS-R(config)#spanning-tree instance ?
  <1-63>  instance id
SNR-S5210G-24TX-UPS-R(config)#spanning-tree instance 1 ?
  priority  bridge priority for the common instance
SNR-S5210G-24TX-UPS-R(config)#spanning-tree instance 1 priority ?
  <0-61440>  bridge priority in increments of 4096 (Lower priority indicates
        greater likelihood of becoming root)
SNR-S5210G-24TX-UPS-R(config)#spanning-tree instance 1 priority 8192 ?
  <cr>
```

6. maxage.

Максимальное время жизни сообщения. Данный таймер показывает, когда текущий Hello BPDU устареет и должен быть удален.

По умолчанию - это 20 секунд.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree max-age ?

<6-40> seconds <6-40> - Maximum time to listen for root bridge in seconds

SNR-S5210G-24TX-UPS-R(config)#spanning-tree max-age 25

SNR-S5210G-24TX-UPS-R(config)#

Можно отменить установку таймера командой: ‘no spanning-tree max-age’ - она вернет значение по умолчанию.

7. max-hops.

Установка значения счетчика Max-Hop в MSTP, который определяет какое количество коммутаторов может пройти BPDU, до того как будет отброшен. Причем коммутатор должен работать именно в режиме MSTP, иначе коммутатор вернет ошибку, и команда не будет применена.

Количество хопов по умолчанию - 20.

SNR-S5210G-24TX-UPS-R(config)# spanning-tree max-hops ?

<1-40> hops <1-40> - Maximum hops the BPDU will be valid

SNR-S5210G-24TX-UPS-R(config)# spanning-tree max-hops 10

%Bridge is not MSTP

SNR-S5210G-24TX-UPS-R(config)#spanning-tree mode mstp

SNR-S5210G-24TX-UPS-R(config)# spanning-tree max-hops 10

SNR-S5210G-24TX-UPS-R(config)#

Можно отменить установку таймера командой: ‘no spanning-tree max-hops’ - она вернет значение по умолчанию.

8. mode.

Выбор режима работы spanning-tree.

На данный момент по умолчанию используется rstp.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree mode ?

mstp MSTP mode

rstp RSTP mode

stp STP mode

9. mst.

Конфигурация региона MSTP. Все это уже ранее рассматривали:

SNR-S5210G-24TX-UPS-R(config)#spanning-tree mst ?

configuration Configuration

SNR-S5210G-24TX-UPS-R(config)#spanning-tree mst configuration

SNR-S5210G-24TX-UPS-R(config-mst)#?

MSTI configuration commands:

exit End current mode and down to previous mode

help Description of the interactive help system

instance Instance

no Negate a command or set its defaults

region MST Region

revision Revision Number for configuration information

show Show running system information

10. portfast.

В данной команде мы можем глобально задать выбор защитного функционала от BPDU на всех портах в режиме portfast, где функционал фильтрации BPDU никак не настроен вручную. Когда будем рассматривать команды режима конфигурации порта, станет понятнее.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree portfast ?

bpdu-filter Filter the BPDUs on portfast enabled ports

bpdu-guard guard the portfast ports against bpdu receive

11. priority.

Если в команде ранее ‘instance X priority’ мы устанавливали bridge priority для различных MSTI, то в данном случае приоритет устанавливается для моста в контрольном instance, или IST (либо просто для моста в STP, RSTP).

Можно даже обратить внимание на описание данных команд для сравнения, чтобы было понятно:

instance Change priority for a particular instance

priority priority - bridge priority for the common instance

Значение по умолчанию - 32768.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree priority ?

<0-61440> bridge priority in increments of 4096 (Lower priority indicates

greater likelihood of becoming root)

SNR-S5210G-24TX-UPS-R(config)#spanning-tree priority 4096

SNR-S5210G-24TX-UPS-R(config)#

12. shutdown.

Выключить процесс spanning-tree на коммутаторе глобально.

На данный момент, по умолчанию включен.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree shutdown ?

<cr>

Отмена данной команды: ‘no spanning-tree shutdown’ будет эквивалентна команде ‘spanning-tree enable’, рассмотренной ранее.

13. transmit-holdcount.

Как и ранее, эта команда устанавливает максимальное количество BPDU, отправляемых с коммутатора за один интервал Hello time.

По умолчанию отправляется 6 BPDU.

SNR-S5210G-24TX-UPS-R(config)#spanning-tree transmit-holdcount ?

<1-10> range of the transmitholdcount

SNR-S5210G-24TX-UPS-R(config)#spanning-tree transmit-holdcount 10

SNR-S5210G-24TX-UPS-R(config)#

Команды spanning-tree из режима конфигурации порта

[STP & SNR]

1. bpdu-filter / bpdu-guard.

При включенном spanning-tree portfast на порту, можно настроить функционал фильтрации BPDU.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree bpdu-filter ?

default default

disable disable

enable enable

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree bpdu-guard ?

default default

disable disable

enable enable

Мы можем принудительно вручную активировать функционал bpdu-filter или bpdu-guard на порту с помощью опции enable. Также можем принудительно отключить функционал для данного порта с помощью опции disable. А вот если выбрано значение по умолчанию - default, то будет активен функционал или нет, определяется командой из режима глобальной конфигурации, которую мы рассматривали ранее: ‘spanning-tree portfast’.

2. disable / enable.

Тут все просто - отключение / включение spanning-tree на порту.

По умолчанию, spanning-tree включен на всех портах.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree disable ?

<cr>

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree enable ?

<cr>

3. guard.

Включение защиты root guard на порту (чтобы при получении superior BPDU, порт не становился корневым).

По умолчанию функционал отключен.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree guard ?

root disable reception of superior BPDUs

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree guard root ?

<cr>

4. hello-time.

Установка значения интервала hello-time для BPDU, отправляемых с какого-то конкретного порта.

Значение по умолчанию - 2 секунды.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree hello-time ?

<1-10> seconds <1-10> - Hello BPDU interval

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree hello-time 3

SNR-S5210G-24TX-UPS-R(config-if)

5. instance.

В данном режиме можно применить настройки для порта по отношению к какому-либо MSTI.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree instance ?

<1-63> identifier

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree instance 1 ?

path-cost path cost for a port

priority port priority for a bridge

restricted-role restrict the role of the port

Здесь можно выбрать стоимость path-cost, port priority и установить опцию restricted-role (о ней мы поговорим немного позже).

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree instance 1 path-cost ?

<1-200000000> path cost in range <1-200000000> (lower path cost indicates

greater likelihood of becoming root)

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree instance 1 priority ?

<0-240> port priority in increments of 16 (lower priority indicates greater

likelihood of becoming root)

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree instance 1 restricted-role ?

<cr>

6. link-type.

Выбор типа линка.

Auto - коммутатор будет определять тип линка автоматически, исходя из настроек дуплекса на порту;

Point-to-point - будет принудительно выбран p2p тип линка;

Shared - будет принудительно выбран shared тип линка, соответственно RSTP/MSTP будет работать по принципам классического STP.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree link-type ?

auto auto - will be set to either p2p or shared based on duplex

state

point-to-point point-to-point - enable rapid transition

shared shared - disable rapid transition

7. path-cost.

Установить path-cost для STP/RSTP. То есть опять же в отличии от команды ‘spanning-tree instance 1 path-cost’, мы выбираем path-cost не для какого-то MSTI, а глобально для IST или STP/RSTP.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree path-cost ?

<1-200000000> path cost in range <1-200000000> (lower path cost indicates

greater likelihood of becoming root)

8. portfast.

Принудительно обозначить порт как “edge-порт”, т.е. порт, который будет сразу же переходить в состояние Forwarding.

По умолчанию функционал отключен.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree portfast ?

<cr>

9. restricted-role.

Запретить порту становиться root портом.

Казалось бы, в чем тогда отличие от команды ‘spanning-tree guard root’? А отличие в механизмах реализации.

В функционале “root guard” при получении на порт лучшего BPDU, чем текущий корневой коммутатор, порт переходит в состояние root-inconsistent (эквивалентно состоянию Listening).

А вот в функционале “restricted-role” при получении на порт лучшего BPDU, чем текущий корневой коммутатор, порт меняет роль на Alternate, Backup или Designated.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree restricted-role ?

<cr>

По умолчанию функционал отключен.

10. restricted-tcn.

Функционал похож на тот, который мы рассматривали на обычных коммутаторах SNR (tcguard, tcn-guard). Однако там происходило отбрасывание таких BPDU, а здесь же, при поступлении BPDU в нем будут проигнорированы флаги об изменении топологии, и BPDU будет обработан без них (не будет отброшен).

По умолчанию функционал отключен.

SNR-S5210G-24TX-UPS-R(config-if)#spanning-tree restricted-tcn ?

<cr>

Заключение

Итак, в данной статье у нас получилось рассмотреть подробно каждый протокол из семейства spanning-tree. Понятно, что, в любом случае, не удастся учесть все детали и нюансы, но мы постарались это сделать.

Также мы рассмотрели нюансы настройки и работы spanning-tree на коммутаторах SNR.

В качестве заключения предлагаю рассмотреть такие тезисы:

- Почему НЕ spanning-tree (а, условно, какой-нибудь ERPS)?
- Почему spanning-tree?

1. **Почему НЕ spanning-tree** (а, условно, какой-нибудь ERPS).

- Время сходимости: тот же ERPS сходится гораздо быстрее практически во всех случаях (однако в случае восстановления канала, быстрее может все-таки оказаться RSTP).
- Spanning-tree расходует больше ресурсов CPU коммутатора, чем ERPS;
- В ERPS есть наличие таймера "hold-off", который исключает полную неработоспособность сети в случае частого изменения состояния порта, чего нельзя сказать про семейство spanning-tree.

2. **Почему spanning-tree.**

- Протоколы STP достаточно распространенные: используются практически на любом современном вендоре, при каких-либо проблемах/вопросах, всегда можно быстро найти информацию по работе spanning-tree в сети интернет: обучающие видео, курсы, статьи, форумы и т.д.;
- Поскольку spanning-tree очень распространен, он весьма прост в настройке и в понимании логики его работы;
- Может применяться на любой топологии, необязательно даже в кольцевой.

Как видите, есть свои плюсы и минусы. Поэтому какой именно протокол использовать - все зависит от потребностей и особенностей сети.

Напоследок еще рассмотрим возможные проблемы в работе STP и best practice:

1. Для включения spanning-tree на порту spanning-tree должен быть включен глобально;
2. Не всегда стоит полагаться на автоматику, лучше управлять своей сетью самому: Bridge Priority, path cost и port priority лучше настраивать вручную;
3. Параметры spanning-tree взаимосвязаны и необходимо соблюдать следующие соответствия, иначе spanning-tree может работать некорректно:
   2 x (Bridge_Forward_Delay - 1 sec) >= Bridge_Max_Age
   Bridge_Max_Age >= 2 x (Bridge_Hello_Time + 1 sec)
4. Если по какой-то причине не работает алгоритм rapid spanning-tree - proposal/agreement, а сам протокол работает по классическим таймерам STP, то необходимо проверять типы подключенных линков (p2p или shared), какой протокол spanning-tree работает на соседе (может быть, на соседе запущен STP);
5. Лучше использовать RSTP, если нет необходимости осуществлять балансировку в множестве VLAN, или соответственно MSTP для этой цели. Классический STP в современных сетях использовать не рекомендуется;
6. Если используется spanning-tree на коммутаторах SNR вместе с PVST, то необходимо, чтобы корневым коммутатором был коммутатор SNR. В противном случае, рекомендуется использовать MST, а не PVST;
7. Предпочтительнее, чтобы в сети при использовании MSTP существовал только один регион. Это упрощает конфигурацию, управление и уменьшает вероятность возникновения ошибок;
8. Если коммутаторы оказываются в разных регионах, то необходимо проверить на них консистентность настроек: имя региона, маппинг vlan-instance, номер ревизии. Если все одинаково, то необходимо смотреть какое значение MD5-хеша отправляют коммутаторы, и при необходимости использовать digest-snooping;
9. При диагностике всегда можно запустить сбор дебага spanning-tree и, при необходимости, выбрать какую-либо опцию:

debug spanning-tree

SNR-S2982G-24TE#debug spanning-tree ?

boundary-change Boundary-change info

bpdu Bpdu info

change Change info

error Error info

event Event info

fsm-port Port state machine

proposals Proposals

reroot Reroot

show Show running system information

sync Sync

timer Timer

<cr>

(не забыть прописать ‘terminal monitor’, чтобы сообщения дебага появились в терминале).

И нужно всегда помнить, что изменение параметров spanning-tree может вызвать изменение топологии.