**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 VLANам и ряде других вопросов, но в целом решающих ту же задачу похожими способами. Все их принято обобщённо называть STP-протоколами.

Протокол работает на канальном уровне. STP позволяет делать топологию избыточной на физическом уровне, но при этом логически блокировать петли. Достигается это с помощью того, что STP отправляет сообщения BPDU и обнаруживает фактическую топологию сети. А затем, определяя роли коммутаторов и портов, часть портов блокирует так, чтобы в итоге получить топологию без петель.

STP использует алгоритм STA (Spanning Tree Algorithm), результатом работы которого является граф в виде дерева

Для того чтобы определить какие порты заблокировать, а какие будут передавать данные, STP выполняет следующее:

* Выбор корневого моста (Root Bridge)
* Определение корневых портов (Root Port)
* Определение выделенных портов (Designated Port)

**BPDU**

BPDU бывают двух видов: конфигурационные (Configuration BPDU) и TCN (Topology Change Notification BPDU). Первые регулярно рассылаются корневым свичом (и ретранслируются остальными) и используются для построения топологии, вторые, как понятно из названия, отсылаются в случае изменения топологии сети (проще говоря, подключении\отключении свича). Конфигурационные BPDU содержат несколько полей, остановимся на самых важных:

* идентификатор отправителя (Bridge ID)
* идентификатор корневого свича (Root Bridge ID)
* идентификатор порта, из которого отправлен данный пакет (Port ID)
* стоимость маршрута до корневого свича (Root Path Cost)

Устройства с включенным STP шлют BPDU из всех работающих портов на мультикастовый ethernet-адрес 01-80-c2-00-00-00 (по умолчанию каждые 2 секунды), который прослушивают все свичи с включенным STP.

**Выбор корневого моста**

Сначала выбирается так называемый корневой мост/свич (root bridge). Это устройство, которое STP считает точкой отсчета, центром сети; все дерево STP сходится к нему. Выбор базируется на таком понятии, как идентификатор свича (Bridge ID). Bridge ID это число длиной 8 байт, которое состоит из Bridge Priority (приоритет, от 0 до 65535, по умолчанию 32768+номер vlan или инстанс MSTP, в зависимости от реализации протокола), и MAC-адреса устройства. В начале выборов каждый коммутатор считает себя корневым, о чем и заявляет всем остальным с помощью BPDU, в котором представляет свой идентификатор как ID корневого свича. При этом, если он получает BPDU с меньшим Bridge ID, он начинает анонсировать полученный Bridge ID в качестве корневого. Корневым становится коммутатор с наименьшим идентификатором моста (**Bridge ID**).

Такой подход таит в себе довольно серьезную проблему. Дело в том, что, при равных значениях Priority (а они равные, если не менять ничего) корневым выбирается самый старый свич, так как мак адреса прописываются на производстве последовательно, соответственно, чем мак меньше, тем устройство старше (естественно, если у нас все оборудование одного вендора). Понятное дело, это ведет к падению производительности сети, так как старое устройство, как правило, имеет худшие характеристики. Подобное поведение протокола следует пресекать, выставляя значение приоритета на желаемом корневом свиче вручную

Только один коммутатор может быть корневым. Для того чтобы выбрать корневой коммутатор, все коммутаторы отправляют сообщения **BPDU**, указывая себя в качестве корневого коммутатора. Если коммутатор получает BPDU от коммутатора с меньшим Bridge ID, то он перестает анонсировать информацию о том, что он корневой и начинает передавать BPDU коммутатора с меньшим Bridge ID. В итоге только один коммутатор останется корневым и будет передавать BPDU.

**Определение корневых портов**

Порт коммутатора, который имеет кратчайший путь к корневому коммутатору, называется корневым портом. У любого не корневого коммутатора может быть только один корневой порт.

После того, как коммутаторы выбрали root bridge, каждый из остальных свичей должен найти один, и только один порт, который будет вести к корневому свичу. Такой порт называется корневым портом (**Root port**).

Эта стоимость определяется суммой стоимостей всех линков, которые нужно пройти кадру, чтобы дойти до корневого свича. В свою очередь, стоимость линка определяется просто- по его скорости (чем выше скорость, тем меньше стоимость). Процесс определения стоимости маршрута связан с полем BPDU “**Root Path Cost**” и происходит так:

* Корневой свич посылает BPDU с полем Root Path Cost, равным нулю
* Ближайший свич смотрит на скорость своего порта, куда BPDU пришел, и добавляет стоимость
* Далее этот второй свич посылает этот BPDU нижестоящим коммутаторам, но уже с новым значением Root Path Cost, и далее по цепочке вниз

Если стоимость линков до корневого коммутатора совпадает, то выбор корневого порта происходит на основе меньшего Bridge ID коммутатора. Если и Bridge ID коммутаторов до корневого коммутатора совпадает, то тогда корневой порт выбирается на основе **Port ID**.

**Определение назначенных портов**

Коммутатор в сегменте сети, имеющий наименьшее расстояние до корневого коммутатора, называется назначенным коммутатором. Порт этого коммутатора, который подключен к рассматриваемому сегменту сети называется назначенным портом. Так же как и корневой порт выбирается на основе:

* Меньшего Root Path Cost.
* Меньшего Bridge ID.
* Меньшего Port ID.

У корневого свича все порты — назначенные.

После того, как выбраны корневые и назначенные порты, оставшиеся блокируются, таким образом разрывая петлю.

**Роли и состояния портов**

Роли портов:

* Root Port — корневой порт коммутатора
* Designated Port — назначенный порт сегмента
* Nondesignated Port — неназначенный порт сегмента
* Disabled Port — порт который находится в выключенном состоянии.

Состояния портов:

* блокировка (**blocking**): блокированный порт не шлет ничего. Это состояние предназначено, как говорилось выше, для предотвращения петель в сети. Блокированный порт, тем не менее, слушает BPDU (чтобы быть в курсе событий, это позволяет ему, когда надо, разблокироваться и начать работать)
* прослушивание (**listening**): порт слушает и начинает сам отправлять BPDU, кадры с данными не отправляет.
* обучение (**learning**): порт слушает и отправляет BPDU, а также вносит изменения в CAM- таблицу, но данные не перенаправляет.
* перенаправление\пересылка (**forwarding**): этот может все: и посылает\принимает BPDU, и с данными оперирует, и участвует в поддержании таблицы mac-адресов. То есть это обычное состояние рабочего порта.
* отключен (**disabled**): состояние administratively down, отключен командой shutdown.

Порядок перечисления состояний не случаен: при включении (а также при втыкании нового провода), все порты на устройстве с STP проходят вышеприведенные состояния именно в таком порядке (за исключением disabled-портов). Возникает закономерный вопрос: а зачем такие сложности? А просто STP осторожничает. Ведь на другом конце провода, который только что воткнули в порт, может быть свич, а это потенциальная петля. Вот поэтому порт сначала 15 секунд (по умолчанию) пребывает в состоянии прослушивания — он смотрит BPDU, попадающие в него, выясняет свое положение в сети, потом переходит к обучению еще на 15 секунд — пытается выяснить, какие mac-адреса “в ходу” на линке, и потом, убедившись, что ничего он не поломает, начинает уже свою работу. Итого, мы имеем целых 30 секунд простоя, прежде чем подключенное устройство сможет обмениваться информацией со своими соседями.

**Portfast**

Для более быстрого подключения оконечных устройств используется особый режим порта — portfast. При подключении устройства к такому порту, он, минуя промежуточные стадии, сразу переходит к forwarding-состоянию. Само собой, portfast следует включать только на интерфейсах, ведущих к конечным устройствам (рабочим станциям, серверам, телефонам и т.д.), но не к другим свичам.

Есть очень удобная команда режима конфигурации интерфейса для включения нужных фич на порту, в который будут включаться конечные устройства: switchport host. Эта команда разом включает PortFast, переводит порт в режим access (аналогично switchport mode access), и отключает протокол PAgP (об этом протоколе подробнее в разделе агрегация каналов).

**Виды STP**

CST (Common Spanning Tree)

PVST (Per-VLAN Spanning Tree)

RSTP (802.1w, Rapid Spanning Tree Protocol)

PVRST (Per-Vlan Rapid Spanning Tree)

**RSTP (802.1w)**

Роли портов:

* Root
* Designated
* Alternate — альтернативный путь к корневому коммутатору. Путь отличается от того, который использует корневой порт.
* Backup — запасной путь в сегмент.

Состояния портов:

* Discarding
* Learning
* Forwarding

В RSTP остались такие роли портов, как корневой и назначенный, а роль заблокированного разделили на две новых роли: Alternate и Backup. Alternate — это резервный корневой порт, а backup — резервный назначенный порт. Как раз в этой концепции резервных портов и кроется одна из причин быстрого переключения в случае отказа. Это меняет поведение системы в целом: вместо реактивной (которая начинает искать решение проблемы только после того, как она случилась) система становится проактивной, заранее просчитывающей “пути отхода” еще до появления проблемы. Смысл простой: для того, чтобы в случае отказа основного переключится на резервный линк, RSTP не нужно заново просчитывать топологию, он просто переключится на запасной, заранее просчитанный.

Ранее, для того, чтобы убедиться, что порт может участвовать в передаче данных, требовались таймеры, т.е. свич пассивно ждал в течение означенного времени, слушая BPDU. Ключевой фичей RSTP стало введение концепции типов портов, основанных на режиме работы линка- full duplex или half duplex (типы портов p2p или shared, соответственно), а также понятия пограничный порт (тип edge p2p), для конечных устройств. Пограничные порты назначаются, как и раньше, командой spanning-tree portfast, и с ними все понятно- при включении провода сразу переходим к forwarding-состоянию и работаем. Shared-порты работают по старой схеме с прохождением через состояния BLK — LIS — LRN — FWD. А вот на p2p-портах RSTP использует процесс предложения и соглашения (proposal and agreement). Его можно описать так: свич справедливо считает, что если линк работает в режиме полного дуплекса, и он не обозначен, как пограничный, значит, на нем только два устройства- он и другой свич. Вместо того, чтобы ждать входящих BPDU, он сам пытается связаться со свичом на том конце провода с помощью специальных proposal BPDU, в которых, конечно, есть информация о стоимости маршрута к корневому свичу. Второй свич сравнивает полученную информацию со своей текущей, и принимает решение, о чем извещает первый свич посредством agreement BPDU. Так как весь этот процесс теперь не привязан к таймерам, происходит он очень быстро.