Метим кадры на канальном уровне

atraining.ru/vlan-tag-switching-in-detail

Ruslan V. Karmanov

Привет.

Коммутация – одна из самых нелюбимых тем у слушателей, готовящихся к CCIE Routing and Switching. Бывают люди, которые с ходу разбираются с динамической маршрутизацией (таких большинство), кто-то сразу вкуривает, как работает Frame Relay (их меньше), кто-то интересуется взаимодействием TokenRing и IPX (это некрофилы). Свичинг не любит практически никто – за его “мутность”.

В случае, например, динамической маршрутизации, весь траблшутинг и анализ ситуации достаточно понятен и осознаваем мозгом – можно посмотреть, от кого к кому пришли какие маршруты и достаточно быстро сделать выводы о типовых проблемах в конфигурировании. В случае с траблшутингом работ на канальном уровне дело обычно хуже – надо лезть в отладку и вычислять “кто на кого почему каким портом в каком статусе смотрит и не напутали ли ничего с виланами-stp-прочим”. В принципе, со временем уже становится проще, но на психически здорового человека, попавшего на курс SWITCH с детальным разбором работы протоколов семейства STP, производит специфическое впечатление. Усугубляется это наличием книжки от русской редакции Cisco Press с названием “Коммутация в локальных сетях”, в которой полторы тысячи страниц, примеры про CatOS года так от 2003-2004, жуткий язык изложения достаточно несложных тем и постоянный подтекст а-ля Братья Гримм “но это ещё фигня, дальше только страшнее будет” (у меня в своё время сложилось впечатление, что цель этой серии – нагон слушателей на курсы путём внедрения мысли вида “видите, как всё страшно, без курсов никак не справитесь”). Поэтому на коммутацию можно и нужно потратить время – именно детальное знание таких “очевидных” штук и отличает профессионала от “мы с ребятами вчера убунту поставили – вроде всё понятно, теперь мы специалисты по суперкомпьютерной ОС”.

Сегодняшний разговор – про метки кадров.

Как обычно, я предполагаю, что Вы знаете коммутацию на уровне CCNA.

Одиночный вариант: метим кадры 802.1Q

Достаточно часто возникает задача логического мультиплексирования нескольких потоков данных в пределах одной физической среды передачи. Для её [задачи] реализации в стандартах семейства 802.x даже есть специальный стандарт – 802.2. Но в самом распространённом протоколе канального уровня – 802.3 – вместо данной схемы формирования заголовка LLC-уровня используется Ethernet II (т.е. когда после SRC MAC сразу идёт 2 байта с кодом протокола (поле EtherType)).

Конечно, есть и другие варианты – например, любимый фирмой Novell “raw ethernet”, когда сразу после двухбайтового поля Length идёт заголовок IPX, но на данный момент это решение уже не используется.

Что же используется? В основном используется 802.1Q – достаточно простой стандарт, подразумевающий добавление в кадр дополнительного заголовка размером в 4 байта, и имеющего такой формат:

- Поле TPID – Tag Protocol ID – два байта, идентифицирующие тип доп.заголовка – всегда `0x8100`.
- Поле PCP – Priority Code Point – три бита, рождённые в муках комитетом 802.1p, которому вообще-то предписывали заниматься динамической фильтрацией мультикастового трафика, но, видимо, увидев КПД (три бита), расстреляли комитет в полном составе. Эти три бита также часто называются CoS – Class of Service.
- Поле CFI – Canonical Format Indicator – 1-битовый флаг, показывающий формат MAC-адресов. В авторизованных курсах Cisco игриво называется “Признак Token Ring”, хотя говорит чуть о другом. Это поле всегда должно быть в нуле (т.е. показывать нормальность формата MAC’ов в кадре), а если оно в единице – это говорит о том, что MAC’и в кадре нетрадиционной ориентации и не могут дружить с обычными коммутаторами. По сути, найдя единицу в этом поле, коммутатор не должен отдавать указанный кадр в untagged-виде (т.е. например в транк с native-vlan’ом, совпадающим с указанным в заголовке), т.к. в этом случае не гарантируется корректная обработка кадра получателем.
- Поле VID – Virtual LAN ID – 12 бит, содержащих наконец-то главное – номер VLAN’a.

Чуть уточнение в плане терминологии. Когда упоминается о “чистом 802.1p”, то обычно имеется в виду вариант 802.1Q, в котором VID = 0. Тогда значимым считается только поле PCP, и вся конструкция называется Priority Tag. В остальных случаях – 802.1Q. В случае явного указания “Заголовок 802.1Q/p” обычно подразумевается, что обрабатываются оба поля – и номер VLAN, и приоритет канального уровня. Поле CFI сейчас фактически не используется.

В стандартах семейства 802.11 местная реализация записи данных QoS канального уровня называется 802.11e и технически представляет из себя тот же заголовок 802.1Q.

Конечно, есть ещё вариант логического мультиплексирования с использованием ISL-инкапсуляции, но про него, если нужно, я напишу подробнее отдельно. Сейчас основное и стандартное решение всё ж 802.1Q.

Соответственно, вкратце про то, как объединить/разделить несколько потоков данных внутри одной Ethernet-сети (да и WiFi) мы разобрались, можно подытожить:

- Можно сделать несколько независимых потоков данных, разделив их на канальном уровне

- Для протоколов 802.x для этого существует SNAP-инкапсуляция
- Кроме протокола 802.3, для него практикуется использование 802.1Q, который заодно умеет передавать данные о приоритете кадра через поле 802.1p
- Ну и кроме 802.11, для которого есть 802.11e, который по сути тот же 802.1Q

Теперь удвоим ставки.

Технология 802.1ad – двойная метка

Как указывалось выше, у данной технологии есть и другие названия. QinQ, к примеру, или Stacked VLAN – всё это тоже имеет отношение к 802.1ad. Суть же достаточно проста. 12 бит для номера VLAN хватает только на 4 с небольшим тысячи отдельных сред передачи данных, а когда надо больше – то нужно придумывать что-то другое. Сценарии могут быть различны – провайдеру надо “пробросить” транк клиента, не затрагивая схему нумерации VLAN’ов, надо балансировать нагрузку между субинтерфейсами внутри сети провайдера, либо просто – маловато номеров. Самое простое – сделать ещё одну такую же метку (tag). В этом случае внешней (OuterTag) будет называться метка, которая ближе к заголовку кадра (т.е. которая сразу за SRC MAC), а внутренней (InnerTag) – следующая за ней. Также изменения будут и в названиях VID – VID во внешней метке будет называться SP-VLAN (Service Provider), во внутренней – CE-VLAN (Customer Edge).

Сразу, чтобы не запутаться:

- Тэгом – InnerTag / OuterTag – называется весь доп.заголовок, который 802.1Q, все его 4 байта
- OuterTag идёт первым (левее, если представить кадр как битовую строку), а InnerTag вторым (соответственно, правее)
- Когда всё это используется для проброса клиентских транков через сеть провайдера, то поле VID внутри OuterTag называют SP-VLAN, а поле VID внутри InnerTag называют CE-VLAN

Самих меток коснутся изменения – если внутренняя сохранит свой вид, то у внешней будет использоваться другой EtherType – вместо `0x8100` там могут быть:

- `0x88a8`
- `0x9100`
- `0x9200`

Притом стандартным является `0x88a8`, остальные же два распознаются только частью оборудования (например, старшими линейками роутеров Cisco – значение `0x9200` будет являться стандартным для организации туннелей Q-in-Q между роутерами семейства Cisco 10000). Дополнительным изменением является смена роли “бестолкового” бита CFI у OuterTag. Вместо CFI бит станет называться DEI – Drop Eligible Indicator, и по сути является дополнительным флагом для управления механимом CoS и работает аналогично подобному у Frame Relay. Над данной

структурой определены 2 операции – push tag и pop tag. Операция push tag над обычным 802.1Q кадром добавляет новый OuterTag, операция pop tag над QinQ-кадром удаляет OuterTag и возвращает кадр в вид 802.1Q (соответственно, его 802.1Q является бывшим InnerTag'ом). Давайте перейдём к практической части – как это настраивается.

Базовые операции с настройкой 802.1Q на оборудовании Cisco

Начинаем настройку всегда с предположения, что ничего не настроено и не работает. Поэтому заходим на интерфейс и пробуем ввести команду `host(config-if)# switchport` В случае, если наш коммутатор умный и поддерживает L3, то эта команда в явном виде укажет, что порт должен работать в режиме порта коммутатора, а не routed. Если глупый – пошлёт, сказав, что такой команды нет. Это не страшно. Далее, мы с Вами должны зафиксировать тип транкинга – чтобы не тратилось время на автосогласование типа транка. Мы выберем 802.1Q. `host(config-if)# switchport trunk encapsulation dot1q` Если надо вернуть автосогласование типа транкинга, то: `host(config-if)# switchport trunk encapsulation negotiate`
Примечание: Обратите внимание – это не согласование транк/не-транк, это согласование какой именно тип транка.

Опять же – если коммутатор просто не знает других типов транкинга, кроме 802.1Q, он не сможет выполнить эту команду. Что тоже, в общем-то, не страшно. Эту команду надо делать до переключения интерфейса в транк по той причине, что иначе переключение просто не произойдёт. Теперь фиксируем EtherType транка – помним, что он может не всегда быть `0x8100`: `host(config-if)# switchport dot1q ethertype` *`тип`* И включаем в явном виде транк: `host(config-if)# switchport mode trunk` В общем-то интерфейс теперь работает в режиме транка, и если с другой стороны будет стоять динамическое согласование либо тоже транк – всё будет ОК. Если есть уверенность, что с той стороны тоже в явном виде будет транк, а не динамическое согласование, можно убрать обработку кадров протокола согласования транка (DTP). Это чуть уменьшит трафик (чисто символически, в общем), плюс приведёт к тому, что когда с другой стороны будет динамически настроенный порт, то всё закончится плохо – связь не будет установлена, т.к. наша сторона будет молчать, как пленный партизан, на все запросы. Выключить динамическое согласование транкинга можно так: `host(config-if)# switchport nonegotiate` Теперь настройка для роутеров. В случае PE-роутера, который будет принимать поток со Stacked VLAN, ему надо будет по внешней метке понять, на какой субинтерфейс отдать поток клиентского трафика. Поэтому на родительском интерфейсе (не на субинтерфейсе!) надо для начала указать используемый EtherType. Это необходимо, чтобы работать с не-Cisco оборудованием, которое использует EtherType = `0x9100` да и вообще – в явном виде указать не помешает. Убирайте разложенные грабли заранее. Команда `host(config-if)# dot1q tunneling ethertype` *`значение`* устанавливает на интерфейсе используемый

EtherType, а команда `host(config-if)# no dot1q tunneling ethertype` возвращает тип к дефолтному `0x8100`. После этого нам, скорее всего, захочется создать субинтерфейс, на котором надо будет принимать трафик с нужными SP-VLAN и CE-VLAN. Это делается командой: `host(config-subif)# encapsulation dot1q` *`SP-VLAN`* `second-dot1q` *`CE-VLAN`* Вместо CE-VLAN можно указать слово any, тогда на этот интерфейс будут приходить все “остальные” кадры, для которых не нашлось явно сконфигурированных субинтерфейсов. Также надо отметить, что SP-VLAN будет задаваться явным значением, а CE-VLAN может быть и диапазоном (вида `40-50,70,80`).

Заключение

Тема решений класса Metro Ethernet и Carrier достаточно интересна, по крайней мере – надо хорошо представлять базовые вопросы, принципы и терминологию. Как всегда я пишу в этом месте статьи – в случае проявления интереса данный материал может быть (и будет) расширен и дополнен.