**L3VPN**

Virtual Private Network — это набор технологий и протоколов, который позволяет подключить что-то к вашей частной сети через чужую сеть, в частности, через Интернет.

**Клиентский VPN** - клиент сам озабочен его конфигурацией и поднятием. Единственный его посредник — Интернет.

**Провайдерский VPN** - провайдер предоставляет клиенту несколько точек подключения, а внутри своей сети строит каналы между ними. Провайдерский VPN, в отличие от клиентского позволяет обеспечить определённое качество услуг. Обычно при заключении договора подписывается SLA (**Service** **Level** **Agreements**), где оговариваются уровень задержки, джиттера, процент потерь пакетов, максимальный период недоступности сервисов итд.

VPN 3-го уровня — L3VPN, когда нам необходимо обеспечить маршрутизацию сетевого трафика.

**VRF — Virtual Routing and Forwarding** instance. Терминология тут не устоявшаяся: в Cisco — это VRF, в Huawei — VPN-instance, в Juniper — Routing Instance. Все названия имеют право на жизнь, но суть одна — виртуальный маршрутизатор. На одном физическом маршрутизаторе есть много виртуальных маршрутизаторов. Каждый такой виртуальный маршуртизатор — это по сути отдельный VPN. Их таблицы маршрутизации, FIB, список интерфейсов и прочие параметры не пересекаются — они строго индивидуальны и изолированы. Ровно так же они обособлены и от самого физического маршрутизатора. Но, как и в случае виртуальных серверов, между ними возможна коммуникация. VRF строго локален для маршрутизатора, за его пределами VRF не существует. Соответственно VRF на одном маршрутизаторе никак не связан с VRF на другом.

MPLS VPN VRF создаётся только на тех маршрутизаторах, куда подключены клиентские сети. Любым промежуточным узлам не нужно ничего знать о VPN. А между ними нужно как-то обеспечить изолированную передачу пакетов разных VPN. Вот какой подход предлагает MPLS VPN: коммутация в пределах магистральной сети осуществляется, как мы описывали в предыдущей статье, по одной метке MPLS, а принадлежность конкретному VPN определяется другой — дополнительной меткой. Пример:

1) Клиент отправляет пакет из сети 172.16.0.0/24 в сеть 172.16.1.0/24 (в другом филиале).

2) Пока он движется в пределах своего филиала (сеть клиента), он представляет из себя самый обычный пакет IP, в котором Source IP — 172.16.0.2, Destination IP — 172.16.1.2.

3) Сеть филиала знает, что добраться до 172.16.1.0/24 можно через Сеть провайдера. До сих пор это самый обычный пакет, потому что стык идёт по чистому IP с частными адресами.

4) Дальше R1 (маршрутизатор провайдера), получает этот пакет, знает, что он принадлежит определённому VRF (интерфейс привязан к VRF TARS), проверяет таблицу маршрутизации этого VRF — в какой из филиалов отправить пакет — и инкапсулирует его в пакет MPLS.

Метка MPLS на этом пакете означает как раз его принадлежность определённому VPN. Это называется «**Сервисная метка**».

5) Далее наш маршрутизатор должен отправить пакет на R3 — за ним находится искомый офис клиента. Естественно, по MPLS. Для этого при выходе с R1 на него навешивается **транспортная метка** MPLS. То есть в этот момент на пакете две метки.

Продвижение пакета MPLS по облаку происходит по принципу базового MPLS. В частности, на R2 заменяется транспортная метка — SWAP Label.

6) R3 в итоге получает пакет, отбрасывает транспортную метку, а по сервисной понимает, что тот принадлежит к соответствующему VPN.

7) Он снимает все заголовки MPLS и отправляет пакет в интерфейс таким, какой он пришёл на R1 изначально.

Чем хорош MPLS? Тем, что никому нет дела до того, что находится под меткой. Поэтому в пределах магистральной сети не важно, какие адресные пространства у клиента, то есть, какой IP-пакет кроется под заголовком MPLS. Поскольку пакет коммутируется по меткам, а не маршрутизируется по IP-адресам — нет нужды поддерживать и таблицу маршрутизации VPN на промежуточных узлах.

Итак, в промежутке между R1 и R3 (то есть в облаке MPLS) ни у кого нет понимания, что такое VPN – пакеты VPN движутся по метками до пункта назначения, и только он уже должен волноваться, что с ними делать дальше. Это убирает необходимость поднимать VRF на каждом узле и, соответственно, поддерживать таблицу маршрутизации, FIB, список интерфейсов итд.

Учитывая, что весь дальнейший путь пакета определяется на первом MPLS-маршрутизаторе (R1), отпадает нужда и в индивидуальном протоколе маршрутизации в каждом VPN, хотя остаётся вопрос, как найти выходной маршрутизатор.

**Роль меток MPLS**

**Ingress LSR** на пакет навешивает специальную метку MPLS — Сервисную — она идентификатор VPN. **Egress LSR** (последний маршрутизатор — R3) по этой метке понимает, что IP-пакет принадлежит VPN и просматривает соответствующий FIB. То есть очень похоже на VLAN, с той разницей, что только первый маршрутизатор должен об этом заботиться.

Но на основе сервисной метки пакет не может коммутироваться по MPLS-сети — если мы где-то её поменяем, то Egress LSR не сможет распознать, какому VPN она принадлежит. И тут на выручку приходит стек меток. Сервисная метка оказывается внутренней — первой в стеке, а сверху на неё ещё навешивается транспортная. То есть по сети MPLS пакет путешествует с двумя метками — верхней — транспортной и нижней — сервисной).

**Транспортная метка**

Она определяет LSP и меняется на каждом узле. Она добавляется (PUSH) Ingress LSR и удаляется (POP) Egress LSR (или Penultimate LSR в случае PHP). На всех промежуточных узлах она меняется с одной на другую (SWAP). Распространением транспортных меток занимаются протоколы LDP и RSVP-TE.

В целом транспортная метка нам мало интересна, поскольку всё и так уже понятно, за исключением одной детали — FEC. FEC здесь уже не сеть назначения пакета (приватный адрес клиента), это адрес последнего LSR в сети MPLS, куда подключен клиент. Это очень важно, потому что LSP не в курсе про всякие там VPN, соответственно ничего не знает о их приватных маршрутах/префиксах. Зато он хорошо знает адреса интерфейсов Loopback всех LSR. Так вот к какому именно LSR подключен данный префикс клиента, подскажет BGP — это и будет FEC для транспортной метки.

**Сервисная метка**

Нижняя метка в стеке — сервисная. Она является уникальным идентификатором префикса в конкретном VPN. Она добавляется Ingress LSR и больше не меняется нигде до самого Egress LSR, который в итоге её снимает. FEC для Сервисной метки — это префикс в VPN, или, грубо говоря, подсеть назначения изначального пакета.

**Терминология**

**CE — Customer Edge router** — граничный маршрутизатор клиента, который подключен в сеть провайдера.

**PE — Provider Edge router** — граничный маршрутизатор провайдера. Собственно к нему и подключаются CE. На PE зарождается VPN, на нём они и кончаются. Именно на нём расположены интерфейсы, привязанные к VPN. Именно PE назначает и снимает сервисные метки. Именно PE являются Ingress LSR и Egress LSR.

PE должны знать таблицы маршрутизации каждого VPN, ведь это они принимают решение о том, куда посылать пакет, как в пределах провайдерской сети, так и в плане клиентских интерфейсов.

**P — Provider router** — транзитный маршрутизатор, который не является точкой подключения — пакеты VPN проходят через него без каких-либо дополнительных обработок, иными словами просто коммутируются по транспортной метке. P нет нужды знать таблицы маршрутизации VPN или сервисные метки. На P нет интерфейсов привязанных к VPN.

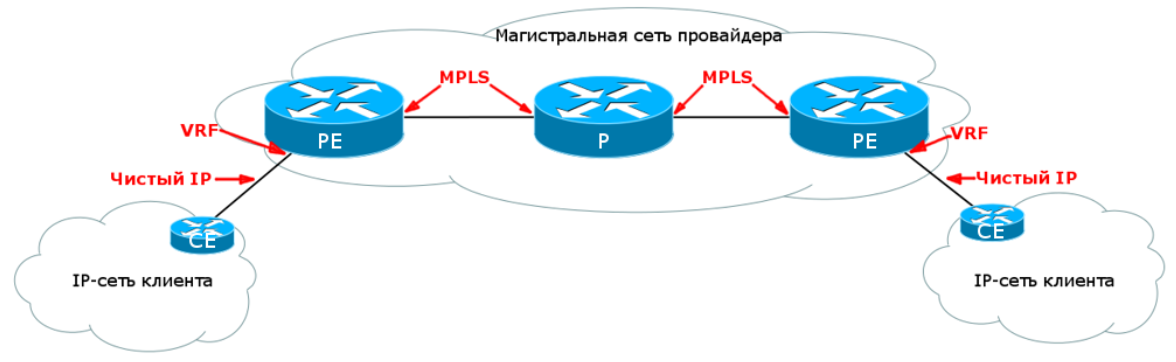
**Стек меток** — набор MPLS заголовков, навешанных на один пакет. Каждый из них выполняет какую-то свою роль. В нашей реальности мало кто из вендоров поддерживает больше шести меток в стеке.

**Протоколы маршрутизации**

Итак, мы имеем два типа сетей и стыки между ними:

* Клиентская IP-сеть;
* Магистральная сеть провайдера с запущенным на ней MPLS.

Граница этих сетей находится на PE. MPLS настраивается только на магистральных интерфейсах.



Во-первых, нужно обеспечить базовую IP-связность внутри магистральной сети провайдера. Чтобы были известны все Loopback-адреса, линковые сети, служебные префиксы, возможно, какие-то выходы вовне. Для этого запускается IGP (ISIS, OSPF). Уже поверх связной сети поднимается MPLS. Так мы обеспечили работоспособность магистральной сети.

Во-вторых, у клиента в филиалах может стоять не по одному маршрутизатору, а сети. Эти сети тоже надо маршрутизировать внутри себя как минимум. Очевидно, что внутри своей собственной сети клиент волен распространять маршрутную информацию, как ему угодно. Так обеспечивается передача маршрутов в пределах сетей клиентов.

В-третьих, клиенту нужно как-то сообщать своим маршруты провайдеру. На стыке CE-PE клиенту и провайдеру уже нужно договариваться о том, какой протокол будет использоваться. Хотя, едва ли у клиента какой-то свой самописный протокол IGP. Наверняка, это OSPF/ISIS/RIP. Поэтому обычно провайдер идёт навстречу и выбирает тот, который удобен клиенту. Тут надо понимать, что вот этот протокол взаимодействия с клиентом работает в VPN и никак не пересекается с IGP самого провайдера. Это разные независимые процессы. Зачастую на этом стыке работает BGP, поскольку позволяет гибко фильтровать префиксы по различным атрибутам. Так провайдер получает маршруты клиентов.

В-четвёртых, и это самое интересное — осталось передать маршруты одного филиала другому через магистральную сеть. При этом их надо по пути не потерять, не перепутать с чужими, доставить в целости и сохранности. Тут нам поможет расширение протокола BGP — MBGP — Multiprotocol BGP (Часто его называют MP-BGP).



**MBGP**

Сейчас ответим на два вопроса: как маршруты передаются в провайдерской сети от одного PE к другому и как обеспечивается изоляция.

В общем-то до сих пор не придумано ничего лучше для передачи маршрутов на удалённые узлы, чем BGP: и гибкость передачи самих маршрутов, и масса инструментов по влиянию на выбор маршрута, и политики получения и передачи маршрутов, и Community, сильно упрощающие групповые действия над маршрутами.

Нужно, чтобы выполнялось два условия:

1) Маршруты разных VPN были уникальными и не смешивались при передаче между PE.

2) Маршруты в конечной точке должны быть переданы правильному VRF.

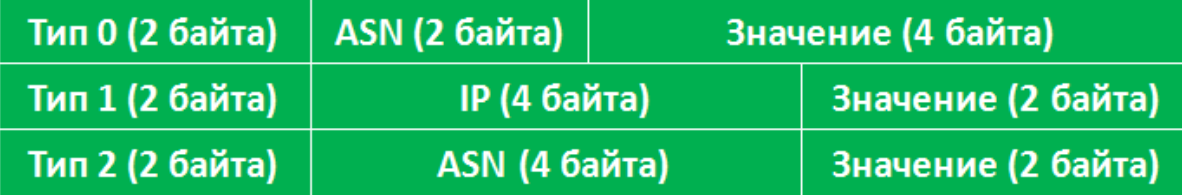
Этим проблемам было найдено элегантное решение.

**Route Distinguisher**

Для того, чтобы различать маршруты различных VPN, обычный IPv4 префикс дополняется специальной приставкой длиной 8 байтов — RD — **Route Distinguisher**.

Тогда маршрут от клиента 1 будет выглядеть так: 64500:**100**:10.10.10.10/32, а от клиента 2 так: 64500:**200**:10.10.10.10/32. И теперь это совершенно разные вещи, которые процесс BGP сможет друг от друга отличить.

Существует 3 типа RD:



RD состоит из трех частей:

Первая часть — сам тип (0, 1 или 2);

Вторая часть — Административное поле (Administrator field) — это всегда публичный параметр — публичный IP-адрес или публичный номер AS. Она необходима для того, чтобы ваши RD были уникальны не только в пределах сети, но и в пределах планеты.

То есть в Административной части не должны появиться случайно IP адрес 172.16.127.2 или AS 65001. Это может пригодиться в том случае, когда VPN нужно передать в сеть другого провайдера (а такое тоже не исключено в наше безумное время, и оно даже носит название Inter-AS VPN).

Третья часть — Выделенный номер (Assigned Number) — это уже то, что назначаете вы. Эта часть позволяет RD быть уникальным в пределах вашей сети и, собственно, определять VPN.

Главная задача RD — разделить префиксы.

RD настраивается вручную.

1) Приходит от CE анонс новой сети. Пусть это будет 10.10.10.10/32. PE добавляет этот маршрут в таблицу маршрутизации конкретного VRF. Заметьте, что в таблице маршрутизации хранится обычный IPv4 маршрут — никаких VPNv4. А это и не нужно: VRF изолированы друг от друга, как мы уже говорили раньше — это отдельный, пусть и виртуальный маршрутизатор.

2) BGP заметил, что появился новый префикс в VPN. Из конфигурации VRF он видит какой RD нужно использовать. Компилирует из RD и нового IPv4-префикса, VPNv4-префикс. Получается так: 64500:100:10.10.10.10/32

3) Создавая BGP Update, маршрутизатор вставляет туда полученный VPNv4-префикс, адрес Next Hop и прочие атрибуты BGP. Но кроме всего прочего, он добавляет в поле NLRI информацию о метке. Эта метка привязана к маршруту, или точнее говоря, VPNv4-префикс — это FEC, а в NLRI передаётся связка данного FEC и метки.

По-английски это называется Labeled Route. Так данный PE уведомляет своих соседей, что если те получили от CE IP-пакет в эту сеть, ему нужно назначить такую сервисную метку.

4) Дальше BGP Update передаётся всем соседям, настроенным в секции VPNv4 family.

5) Удалённый PE получает этот Update, видит в NLRI, что это не обычный IPv4 маршрут, а VPNv4. Далее Egress PE определяет, в какой VRF этот маршрут нужно экспортировать и, собственно, делает это. Так маршрут появляется в таблице маршрутизации и FIB нужного VRF, а оттуда уходит в сеть клиента.

**Route Target**

Не совсем в традициях BGP — разбирать переданный адрес, анализировать его перед тем, как куда-то анонсировать. Для этих целей у нас есть политики.

То есть в классическом BGP, пришлось бы вешать политики на экспорт маршрутов в VRF для каждого отдельно. И мы бы вручную отфильтровывали куда нужно пристроить каждый маршрут.

Один шаг в сторону упрощения — использование community. При отправке маршрута с одного PE на другой можно устанавливать определённый community — свой для каждого VRF, а на удалённой по этому community уже настраивать экспорт в соответствующий VRF. Это уже выглядит удобно и убедительно.

В MBGP зашли ещё немного дальше — идею community развили до понятия Route Target. По сути, это то же community — RT даже передаётся в атрибуте Extended Community, только все политики работают автоматически. Формат RT, точно такой же, как у обычного Extended Community. Например: 64500:100. То есть он похож на первый тип RD. Отчасти поэтому RD и RT так часто путают.

На одной стороне в VRF настраивается RT на экспорт маршрута — тот RT, с которым он будет путешествовать к удалённому PE. На другой именно это же значение RT устанавливается на импорт. И наоборот. Обычно, если задача — просто организовать услугу VPN для одного клиента, то RT на экспорт и на импорт совпадают на всех точках подключения.

**Давайте подытожим шаги настройки L3VPN с нуля в правильном порядке от общего к частному**.

* Настроить IP-адреса провайдера: линковые и лупбэк. Все узлы.
* Настроить IGP в сети провайдера, чтобы обеспечить внутреннюю связность. Все узлы.
* Настроить MPLS + LDP (или RSVP TE, если необходимо). Все узлы.
* Настроить MBGP внутри сети провайдера. Только те PE, где есть клиенты.
* Настроить клиентские VRF, назначить RD, RT. Только те PE, где есть клиенты, настраиватся персонально для каждого.
* Добавить в VRF клиентские интерфейсы, настроить на них IP-адреса. Только те PE, где есть клиенты, настраиватся персонально для каждого.
* При необходимости поднять IGP/BGP с клиентом для обмена маршрутами. Только те PE, где есть клиенты, настраиватся персонально для каждого.

**Трассировка в MPLS L3VPN**

Пока пакет коммутируется по меткам значения каких-бы то ни было полей любых заголовков глубже MPLS не имеют никакого значения. В том числе и TTL. Маршрутизаторы ориентируются на TTL в заголовке MPLS.

И вот когда PE получает пакет от CE, есть два варианта:

* Скопировать значение TTL из заголовка IP в MPLS (Это режим Uniform).
* Записать в поле TTL заголовка MPLS 255 (Это режим Pipe или Short-Pipe).

В первом сценарии вы сможете увидеть каждый маршрутизатор на пути к получателю. А что, если у меня есть закономерное желание, чтобы клиенты не видели топологию моей сети своими трассировками? Я просто не буду внутри своей сети уменьшать TTL MPLS до нуля.