**MPLS**

MPLS — MultiProtocol Label Switching.

Несмотря на то, что MPLS не привязывается к типу сети, на которой он будет работать, в наше время он живёт в симбиозе только с IP. То есть сама сеть строится поверх IP, но переносить при этом она может данные многих других протоколов. MPLS не заменяет IP-маршрутизацию, а работает поверх неё.

Что происходит, если мы активируем MPLS? На маршрутизаторах заполняются таблицы меток и строятся многочисленные LSP.

Когда IP-пакет от ПК1 попадает в сеть MPLS первый маршрутизатор навешивает метку, дальше этот пакет идёт к точке назначения, а каждый следующий маршрутизатор меняет одну метку на другую. При выходе из сети MPLS метка снимается и дальше передаётся уже чистый IP-пакет, каким он был в самом начале.

Это основной принцип MPLS — маршрутизаторы коммутируют пакеты по меткам, не заглядывая внутрь пакета MPLS. Первый — добавляет, последний — удаляет.

Но, на самом деле, даже базовый MPLS дает нам преимущества, если мы провайдер. Как провайдер, мы ведь не используем протоколы IGP для маршрутизации между AS. Для этого мы используем BGP. И именно в связке с BGP станут понятны преимущества MPLS. На каждом маршрутизаторе в нашей AS должен быть настроен BGP. Иначе мы не сможем передавать трафик соседних AS и наших клиентов, через нашу AS. Каждый маршрутизатор должен знать все маршруты. Но это было до MPLS! Когда в нашей сети настроен MPLS, нам больше не обязательно настраивать BGP на каждом маршрутизаторе в сети. Достаточно настроить его только на пограничных маршрутизаторах в AS, на тех, которые подключены к другим клиентам или провайдерам.

Кроме того, что BGP теперь можно не настраивать на каждом маршрутизаторе в AS, маршрутизаторам также не нужно создавать метку для каждого префикса BGP. Достаточно знать, как добраться до IP-адреса, который указан как next-hop.

**Терминология**

**Label** — метка — значение от 0 до 1 048 575. На основе неё LSR принимает решение, что с пакетом делать: какую новую метку повешать, куда его передать. Является частью заголовка MPLS.

**Label Stack** — стек меток. Каждый пакет может нести одну, две, три, да хоть 10 меток — одну над другой. Решение о том, что делать с пакетом принимается на основе верхней метки. Каждый слой играет какую-то свою роль. Например, при передаче пакета используется транспортная метка, то есть метка, организующая транзит от первого до последнего маршрутизатора MPLS. Другие могут нести информацию о том, что данный пакет принадлежит определённому VPN.

**Push Label** — операция добавления метки к пакету данных — совершается в самом начале — на первом маршрутизаторе в сети MPLS.

**Swap Label** — операция замены метки — происходит на промежуточных маршрутизаторах в сети MPLS — узел получает пакет с одной меткой, меняет её и отправляет с другой.

**Pop Label** — операция удаления метки — выполняется последним маршрутизатором — узел получает пакет MPLS и убирает верхнюю метку перед передачей его дальше.

**LSR** — Label Switch Router — это любой маршрутизатор в сети MPLS. Называется он так, потому что выполняет какие-то операции с метками. LSR делится на 3 типа:

* **Intermediate LSR** — промежуточный маршрутизатор MPLS — он выполняет операцию Swap Label.
* **Ingress LSR** — «входной», первый маршрутизатор MPLS — он выполняет операцию Push Label.
* **Egress LSR** — «выходной», последний маршрутизатор MPLS — он выполняет операцию Pop Label (R6).
* **LER** — Label Edge Router — это маршрутизатор на границе сети MPLS. В частности, Ingress LSR и Egress LSR являются граничными, а значит они тоже LER.
* **LSP** — Label Switched Path — путь переключения меток. Это однонаправленный канал от Ingress LSR до Egress LSR, то есть путь, по которому фактически пройдёт пакет через MPLS-сеть. Иными словами — это последовательность LSR. Важно понимать, что LSP на самом деле однонаправленный. Это означает, что, во-первых, трафик по нему передаётся только в одном направлении, во-вторых, если существует «туда», не обязательно существует «обратно», в-третьих, «обратно» не обязательно идёт по тому же пути, что «туда».

На LSR вы не увидите законченной последовательности узлов от входа до выхода, по типу атрибута AS-PATH в BGP. Здесь каждый узел знает только входную и выходную метки. Но LSP при этом существует. LSR не принимает решение о каждом пакете на основе адреса назначения — путь определён заранее.

**FEC — Forwarding Equivalence Class** — это классы трафика. В простейшем случае идентификатором класса является адресный префикс назначения (грубо говоря, IP-адрес или подсеть назначения). Например, есть потоки трафика от разных клиентов и разных приложений, которые идут все на один адрес — все эти потоки принадлежат одному классу — одному FEC — используют один LSP.

В теории помимо адреса назначения FEC может учитывать, например, метки QoS, адрес источника, идентификатор VPN или тип приложений. Важно понимать тут, что пакеты одного FEC не обязаны следовать на один и тот же адрес назначения. И в то же время, если даже и два пакета следуют в одно место, не обязательно они будут принадлежать одному FEC. Для каждого FEC выбирается свой LSP — свой путь через сеть MPLS. И тогда, например, для WEB-сёрфинга вы устанавливаете приоритет QoS BE — это будет один FEC — а для VoIP — EF — другой FEC. И далее можно указать, что для FEC BE LSP должен идти широким, но долгим и негарантированным путём, а для FEC EF — можно узким, но быстрым. Сейчас в качестве FEC может выступать только IP-префикс. Такие вещи, как маркировка QoS не рассматриваются.

Всю работу по определению FEC и в какой LSP отправить трафик берёт на себя Ingress LSR — получив чистый пакет, он его анализирует, проверяет какому классу тот принадлежит и навешивает соответствующую метку. Пакеты разных FEC получат разные метки и будут отправлены в соответствующие интерфейсы. Пакеты одного FEC получают одинаковые метки. То есть промежуточные LSR — это молотилки, которые для всего транзитного трафика только и делают, что переключают метки. А всю интеллектуальную работу выполняют Ingress LSR.

**LIB** — Label Information Base — таблица меток. Аналог таблицы маршрутизации (RIB) в IP. В ней указано для каждой входной метки, что делать с пакетом — поменять метку или снять её и в какой интерфейс отправить.

**LFIB** — Label Forwarding Information Base — по аналогии с FIB — это база меток, к которой обращается сетевой процессор. При получении нового пакета нет нужды обращаться к CPU и делать lookup в таблицу меток — всё уже под рукой.

**Заголовок MPLS**

Весь заголовок MPLS — это 32 бита. Формат полей и их длина фиксированы. Часто весь заголовок называют меткой, хотя это не совсем и верно.



**Label** — собственно сама метка. Длина — 20 бит.

**TC** — Traffic Class. Несёт в себе приоритет пакета, как поле DSCP в IP. Длина 3 бита. То есть может кодировать 8 различных значений. Например, при передаче IP-пакета через сеть MPLS значению в поле DSCP определённым образом ставится в соответствие значение TC. Таким образом пакет может почти одинаково обрабатываться в очередях на всём протяжении своего пути, как на участке чистого IP, так и в MPLS. Но, естественно, это преобразование с потерями — шести битам DSCP тесно в 3 битах TC: 64 против 8. Поэтому существует специальная таблица соответствий, где целый диапазон — это всего лишь одно значение.

**S** — Bottom of Stack — индикатор дна стека меток длиной в 1 бит. Заголовков MPLS на пакете может быть несколько, например, внешняя для коммутации в сети MPLS, а внутренняя указывает на определённый VPN. В бит S записывается «1», если это последняя метка (достигнуто дно стека) и «0», если стек содержит больше одной метки (ещё не дно). То есть LSR не знает, сколько всего меток в стеке, но знает, одна она или больше — да этого и достаточно на самом-то деле. Ведь любые решения принимаются на основе только самой верхней метки, независимо от того, что там под ней. Зато, снимая метку, он уже знает, что дальше сделать с пакетом: продолжить работу с процессом MPLS или отдать его какому-то другому. В итоге, несмотря на то, что длина заголовка MPLS фиксированная, самих заголовков может быть много — и все они располагаются друг за другом.

**TTL** — Time To Live — полный аналог IP TTL.

**Пространство меток**

Может существовать 2^20 меток. Из них несколько зарезервировано:

**0: IPv4 Explicit NULL Label**. «Явная пустая метка». Она используется на самом последнем пролёте MPLS — перед Egress LSR — для того, чтобы уведомить его, что эту метку 0 можно снять, не просматривая таблицу меток (точнее LFIB).

**1: Метка Router Alert Label** — аналог опции Router Alert в IP — может быть где угодно, кроме дна стека. Когда пакет приходит с такой меткой, он должен быть передан локальному модулю, а дальше он коммутируется в соответствии с меткой, которая была ниже — реальной транспортной, при этом наверх стека снова должна быть добавлена метка 1.

**2: IPv6 Explicit NULL Label** — то же, что и 0, только с поправкой на версию протокола IP.

**3: Implicit Null.** Фиктивная метка, которая используется для оптимизации процесса передачи пакета MPLS на Egress LSR. Эта метка может анонсироваться, но никогда не используется в заголовке MPLS реально.

**4-15:** Зарезервированы.

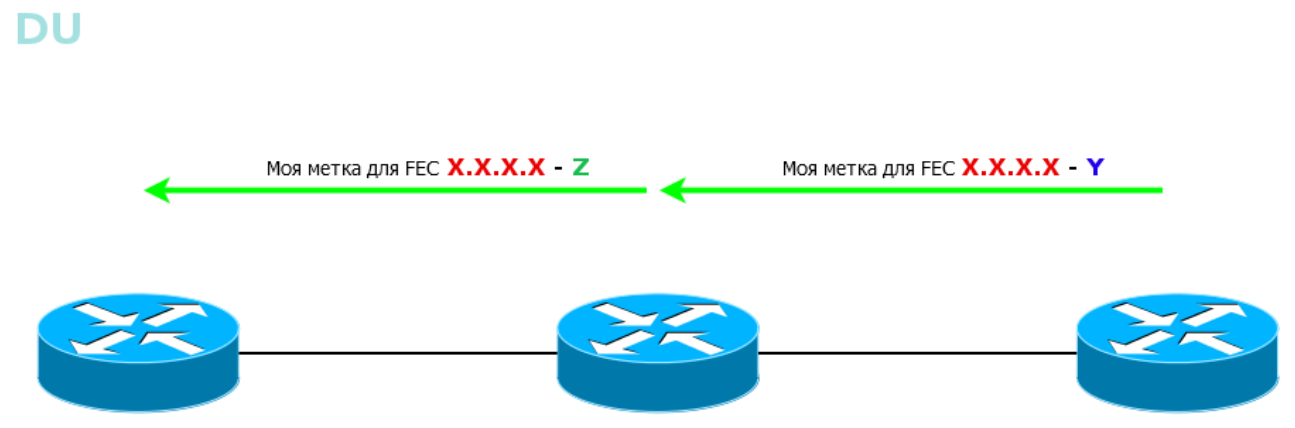
**Распространение меток**

**Методы распространения меток**

Метки распространяются в направлении от получателя трафика к отправителю, а точнее от Egress LER к Ingress LER. В MPLS Downstream — это от отправителя к получателю, а Upstream от получателя к отправителю. То есть метки распространяются навстречу трафику.

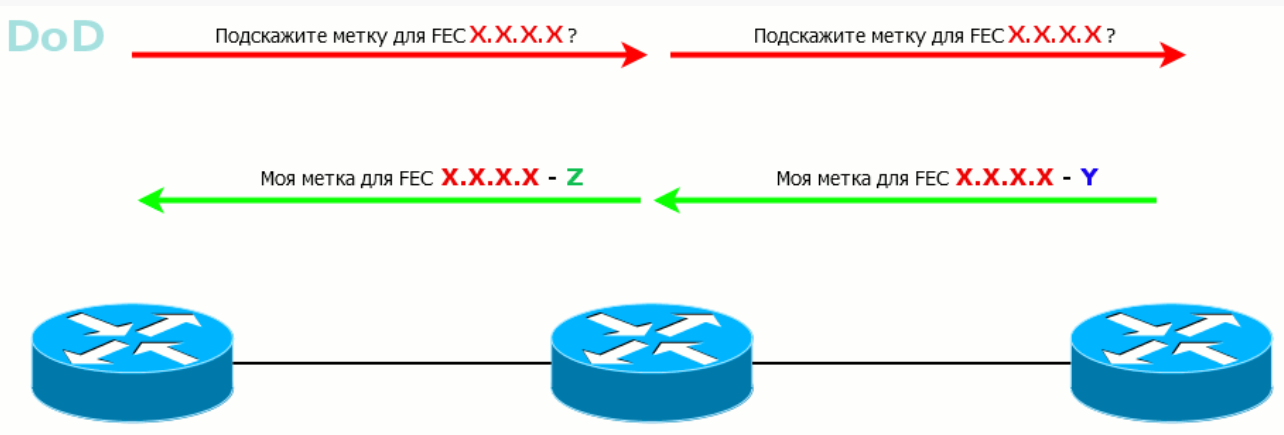
**DU против DoD**

**DU — Downstream Unsolicited**. Как только LSR узнаёт про FEC, он рассылает всем своим MPLS-соседям метки для этого FEC.



Все LSR узнают обо всех FEC по всем возможным путям. Сначала соответствие FEC-метка расходится по всей сети от соседа к соседу, почти как это происходит с сообщениями BootStrap в PIM SM. А потом каждый LSR выбирает только тот, который пришёл по лучшему пути, и его использует для LSP — точно так же работает Reverse Path Forwarding в том же PIM SM.

**DoD — Downstream-on-Demand**. LSR знает FEC, у него есть соседи, но пока они не спросят, какая для данного FEC метка, LSR сохраняет молчание.



Этот способ удобен, когда к LSP предъявляются какие-то требования, например, по ширине полосы. Режим выделения меток специфичен для интерфейса и определяется в момент установки соединения. В сети могут быть использованы оба способа, но на одной линии, соседи должны договориться только об одном конкретном.

**Ordered Control против Independent Control**

LSR может дожидаться, когда со стороны **Egress LER** придёт метка данного FEC, прежде чем рассказывать вышестоящим соседям. А может разослать метки для FEC, как только узнал о нём.

Первый режим называется **Ordered Control**. Гарантирует, что к моменту передачи данных весь путь вплоть до выходного LER будет построен.

Второй режим — **Independent Control**. То есть метки передаются неупорядоченно. Удобен тем, что трафик можно начинать передавать ещё до того, как весь путь построен. Этим же и опасен.

**Liberal Label Retention Mode против Conservative Label Retention Mode**

Хранит ли LSR полученные метки от соседа, которые не являются лучшим маршрутом.

**Liberal Label Retention Mode** — метки сохраняются. В случае, когда альтернативный роутер станет следующим шагом (например, проблемы с основным путём), трафик будет перенаправлен скорее, потому что метка уже есть. То есть скорость реакции выше, но велико и количество использованных меток.

**Conservative Label Retention Mode** — лишняя метка отбрасывается сразу, как она получена. Это позволяет сократить количество используемых меток, но и MPLS среагирует медленнее в случае аварии.

**Penultimate Hop Popping (PHP)**. Нужед, чтобы не проставлять метку на предпоследнем роутере. Для PHP существует специальная метка — 3. При передаче пакета на Egress LER роутер должен назначить ему фиктивную метку — 3, но фактически она не применяется и в интерфейс отправляется голый IP-пакет (стоит заметить, что PHP работает только в сетях IP) — то есть процедура Pop Label выполняется на предпоследнем роутере.

**Протоколы распространения меток**

Существует три базовых протокола для распространения меток — LDP, RSVP-TE и MBGP.

**LDP** — самый простой и понятный способ — опирается на маршрутную информацию узлов.

**RSVP-TE** — это развитие некогда разработанного, но непопулярного протокола RSVP — используется в MPLS-TE для построения LSP, удовлетворяющих определённым условиям. Для его работы нужны IGP, поддерживающие Traffic Engineering (OSPF, ISIS).

**MBGP** — близкий родственник BGP, но это протокол из немного другой истории, он передаёт метки для других целей. Поэтому и стоит он в стороне от LDP и RSVP-TE.

Транспортные метки используются для передачи трафика по сети MPLS. Для них используются LDP и RSVP-TE.

Сервисные метки служат для разделения различных сервисов. Тут на арену выходят MBGP и отросток LDP — tLDP.

**LDP**

Протокол с очень прозрачным названием — **Labed Distribution Protocol** — имеет соответствующий принцип работы.

**1.** После включения LDP LSR делает мультикастовую рассылку UDP-дейтаграмм во все интерфейсы на адрес 224.0.0.2 и порт 646, где активирован LDP — так происходит поиск соседей. TTL таких пакетов равен 1, поскольку LDP-соседство устанавливается между непосредственно подключенными узлами. Такие сообщения называются **Hello**.

**2.** Когда соседи обнаружены, устанавливается TCP соединение с ними, тоже по порту 646 — **Initialization**. Дальнейшие сообщения (кроме Hello) передаются уже с TTL равным 255.

**3.** Теперь LSR периодически обмениваются сообщениями **Keepalive** адресно по TCP и по-прежнему не оставляют попыток найти соседей с помощью **Hello**.

**4.** В какой-то момент один из LSR обнаруживает что он — **Egress LSR** — то есть он является выходным для какого-то FEC. В зависимости от режима он ждёт запроса на метку для данного FEC, либо рассылает его сразу же. Эта информация передаётся в сообщении **Label Mapping Message**. Исходя из названия, оно несёт в себе соответствие FEC и метки.

Самое важное, что нужно понимать относительно LDP — он не использует в своей работе протоколы динамической маршрутизации — по принципу работы он похож на PIM DM: наводняет всю сеть метками, но при этом он опирается на информацию из таблицы маршрутизации LSR. И если на R1 придёт две метки для одного FEC от разных соседей, то он выберет для LSP только ту, которая получена через лучший интерфейс до этого FEC по информации из ТМ.

Это означает три вещи:

* Вы вольны выбирать IGP, который вам больше.
* LDP всегда строит только один (лучший) маршрут и не может построить, например, резервный.
* При изменении топологии сети LSP перестроится в соответствии с обновившейся таблицей маршрутизации, то есть сначала должен сойтись IGP, и только потом поднимется LSP.

По умолчанию LDP в Cisco работает в следующих режимах:

* DU
* Independent Control
* Liberal Retention Mode
* В качестве FEC выбираются все записи в таблице маршрутизации

Для каждого конкретного FEC есть один или несколько Egress LSR и множество Ingress LSR (как правило, все маршрутизаторы), до которых ведут LSP. Понятия LER возникают когда мы говорим о конкретном LSP, тогда мы можем сказать, кто является Ingress, кто Egress.

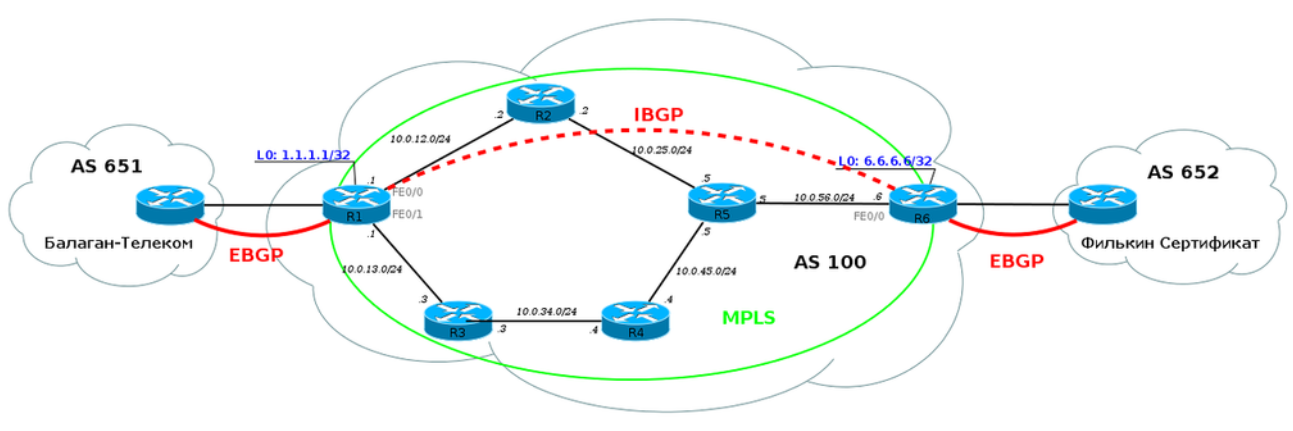
**MPLS и BGP**

Главное отличие BGP от IGP заключается в том, что MPLS не создает метки для маршрутов BGP. Если вспомнить о том, какое количество маршрутов передает BGP, то становится понятно, что это очень хорошая идея. Как же тогда состыковать MPLS и BGP?

Все просто:

* Создаем метки только для адресов, которые будут указаны как next-hop для маршрутов, которые мы получаем по BGP (тут надо не забыть про next-hop-self для IBGP-соседей).
* Когда нашему Ingress LSR понадобится передать пакеты по маршруту, который был получен по BGP, отправляем их к next-hop, который указан в маршруте и используем ту метку, которая была создана для него.

Теперь, вместе того чтобы настраивать BGP на каждом маршрутизаторе в нашей AS, мы можем настраивать его только на пограничных маршрутизаторах, к которым подключены клиенты или другие провайдеры.



**RSVP-TE**

LDP хорош. Работает он просто и понятно. Но есть такая технология, как **MPLS TE — Traffic Engineering**. И ей недостаточно лучшего маршрута, который может обеспечить LDP. Управление трафиком подразумевает, что вы можете направить трафик между узлами как вам угодно, учитывая различные ограничения.

Протокол **RSVP (Resource ReSerVation Protocol)** изначально (в 1993-м году) был задуман для организации IntServ в IP-сетях. Он должен был донести информацию о QoS для какого-то конкретного потока данных до каждого узла и заставить его зарезервировать ресурсы.

В первом приближении работает он просто.

1) Узел-источник хочет отправить поток данных со скоростью 5 Мб/с. Но перед этим он посылает RSVP запрос на резервирование полосы до получателя — **Path Message**. Это сообщение содержит некие идентификаторы потока, по которым узел потом сможет идентифицировать принадлежность полученных IP-пакетов потоку, и требуемую ширину полосы.

2) Сообщение **Path** передаётся от узла к узлу до самого получателя. Куда его послать, определяется на основе таблицы маршрутизации.

3) Каждый маршрутизатор, получив **Path**, проверяет свои ресурсы. Если у него есть достаточно свободной полосы, он настраивает свои внутренние алгоритмы так, чтобы пакеты потока были обработаны как следует и пропускной способности всегда хватало.

4) Если же у него нет необходимых 5 Мб/с (занято другими потоками), он отказывает в выделении ресурсов и возвращает соответствующее сообщение отправителю.

5) Пакет Path доходит до получателя потока, тот отправляет назад сообщение **Resv**, подтверждая выделение ресурсов на всём протяжении пути.

6) Изначальный отправитель, получив **Resv**, понимает, что всё для него готово, и он может отправлять данные.

**RSVP для Traffic Engineering**, или проще — RSVP TE, было разработано специально для **MPLS TE**. Его задача такая же, как у LDP — распределить метки между LSR и скомпилировать в итоге LSP от получателя до отправителя. Но теперь появляются нюансы — LSP должен удовлетворять определённым условиям.

RSVP TE позволяет строить основной и запасной LSP, резервировать ресурсы на всех узлах, обнаруживать аварии на сети, строить заранее обходные пути, делать быстрое перенаправление трафика, избегать каналов, которые физически проходят по одному пути.

Рассмотрим простейший вариант.

0) R1 нужен LSP до FEC 6.6.6.6/32. Это выглядит как интерфейс Tunnel на R1, у которого адрес назначений 6.6.6.6 и тип Traffic Engineering.

1) Он посылает сообщение **RSVP Path** в направлении 6.6.6.6. В этом сообщении появляется новый объект — **Label Request**. Сообщение Path провоцирует узел выделить метку для данного FEC — то есть это запрос метки.

2) Следующий узел формирует новое сообщение **Path** и также отправляет его в сторону 6.6.6.6. И т.д.

3) Path достигает **Egress LSR**. Тот видит, что пакет адресован ему, выделяет метку и отправляет источнику сообщение **Resv**. В последнем тоже добавлен новый объект — Label. В нём Egress LSR передаёт свою метку для этого FEC предпоследнему, предпоследний предпредпоследнему свою итд.

4) Resv достигает источника, распределяя по пути метки. Таким образом создан LSP, а источник уведомлён, что всё готово.

Метки запрашиваются вниз по течению (сообщение Path от отправителя к получателю), а передаются вверх по течению (сообщение Resv от получателя к отправителю).

При этом обратите ваше внимание на то, что это уже самый что ни на есть Downstream on Demand + Ordered Control. Path выступает запросом метки, а Resv идёт от получателя шаг за шагом и, пока метку не выслал нижестоящий LSR, вышестоящий не может её отправить своим соседям.

**RSVP TE** очень тесно связан с протоколами динамической маршрутизации, он не просто опирается на результат их работы — он адаптирует их под себя, эксплуатирует в прямом смысле слова.

Во-первых, годятся только протоколы, основанные на алгоритмах по состоянию связи (link-state), то есть OSPF и ISIS.

Во-вторых, OSPF и ISIS расширяются введением новых элементов в протоколы. Так в OSPF появляется новый тип LSA — Opaque LSA, а в ISIS — новые TLV IS Neighbor и IP Reachability.

В-третьих, для расчёта пути между Ingress LSR и Egress LSR используется специальная модификация алгоритма SPF — CSPF (Constrained Shortest Path First).

Сообщение Path в принципе передаётся юникастом адресно. То есть адрес отправителя у него — адрес R1, а получателя — 6.6.6.6. И оно могло бы дойти и просто по таблице маршрутизации.

Но фактически Path передаётся по сети не как FIB на душу положит на каждом узле, ведь мы тогда не сможем ни резервирование обеспечить, ни поиск запасных маршрутов. Нет, он следует определённому пути. Этот путь определяется на Ingress LSR с точностью до каждого узла. Чтобы построить этот путь, RSVP TE нужно знать топологию сети.

То есть **RSVP TE** может обращаться к **CSPF** с просьбой вычислить для него какой-либо путь между двумя узлами.

**RSVP TE** может предъявлять требования к пути — ширина полосы пропускания, минимально доступная ширина, тип линии или даже узлы, через которые LSP должен пролегать. Чтобы CSPF мог учитывать ограничения, он должен знать и о них, и о доступных ресурсах на узлах всей сети. Входными данными для него являются ограничения, заданные в туннеле и топология сети — логично будет, если топология будет содержать кроме префиксов и метрик ещё и информацию о доступных ресурсах.

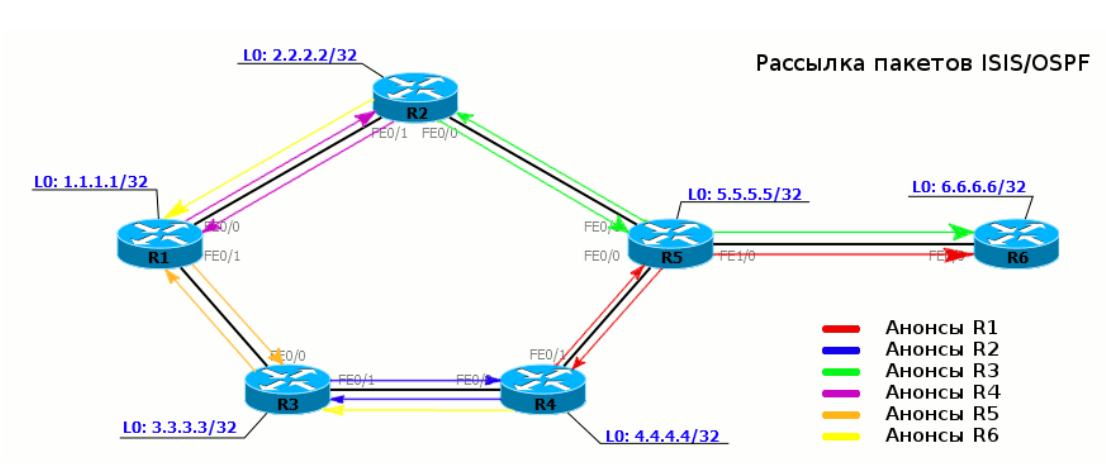
Для этой цели маршрутизаторы обмениваются между собой через сообщения OSPF и ISIS не только базовой информацией, но и характеристиками линий, интерфейсов итд. Как раз в новых типах сообщений и передаётся эта информация. Например, в OSPF для этого ввели 3 дополнительных типа LSA:

* **Type 9** — link-local scope
* **Type 10** — area-local scope
* **Type 11** — AS scope

**Opaque** значит непрозрачный (для OSPF). Это специальные типы LSA, которые никак не учитываются в самом OSPF при расчёте таблицы маршрутизации. Их могут использовать любые другие протоколы для своих нужд. Так TE их использует для построения своей топологии (она называется **TED — Traffic Egineering Database**).

Этапы построения маршрута:

0) На R1 мы включили MPLS TE и настроили ISIS (OSPF) на передачу данных для поддержки TE. Маршрутизаторы обменялись информацией о доступных ресурсах. На этом шаге сформирована TED. RSVP молчит.



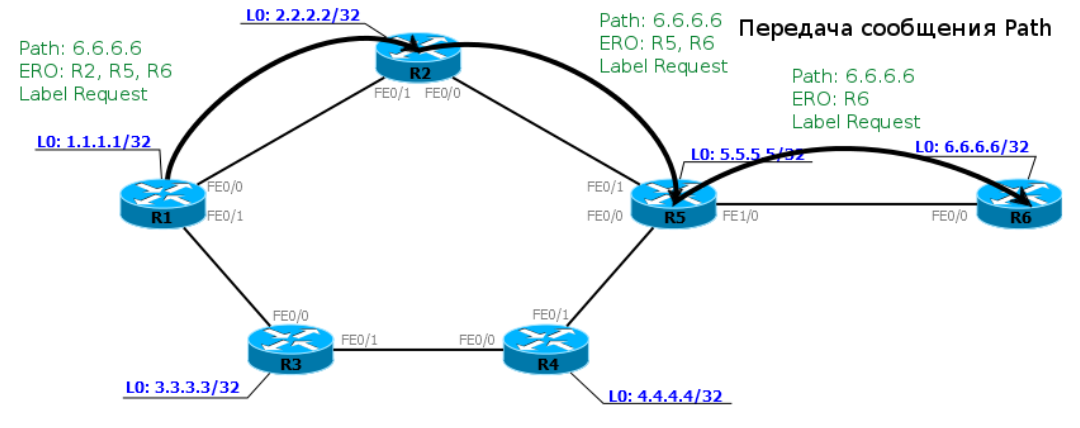
1) Мы создали туннельный интерфейс, где указали его тип (Traffic Engineering), адрес назначения (6.6.6.6) и необходимые требования по ресурсам. LSR запускает CSPF: нужно вычислить кратчайший путь от R1 до 6.6.6.6 с учётом накладываемых условий. На этом шаге мы получаем оптимальный путь — список узлов от источника до получателя (R2, R5, R6).

2) Результат предыдущего шага скармливается RSVP и трансформируется в объект **ERO** — **Explicit Route Object** — специальный объект сообщения RSVP Path. Он содержит список узлов, через которые суждено пройти этому сообщению. R1 компилирует RSPV Path, куда, естественно, добавляет ERO. Адресат пакета — 6.6.6.6. Кроме того, имеется и объект Label Request, сообщающий о том, что при получении пакета нужно выделить метку для данного FEC (6.6.6.6/32).

3) Сообщение RSVP Path передаётся специальным образом — не по таблице маршрутизации, а по списку ERO. В нашем случае лучший маршрут IGP и ERO совпадают, поэтому пакет посылается на R2.

4) R2, получив RSVP Path, проверяет наличие требуемых ресурсов и, если они есть, выделяет метку MPLS для FEC 6.6.6.6/32. Старый пакет Path уничтожается и создаётся новый, причём объект ERO меняется — из него удаляется сам R2. Делается это для того, чтобы следующий узел не пытался вернуть пакет на R2. То есть новый ERO выглядит уже так: (R5, R6). В соответствии с ним R2 отправляет обновлённый Path на R5.

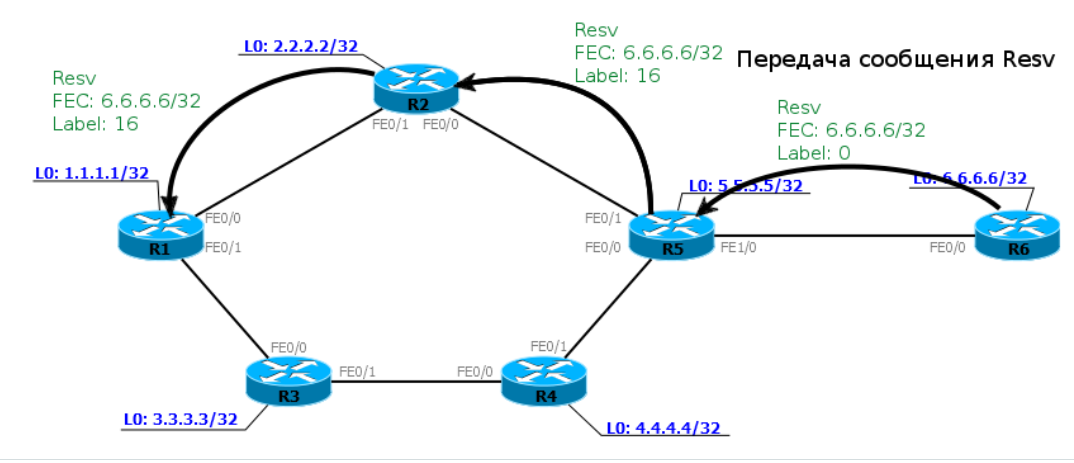
5) R5 совершает аналогичные операции: проверяет ресурсы, выделяет метку, удаляет себя из ERO, пересоздаёт пакет Path и передаёт в интерфейс, через который ему известен следующий объект ERO — R6.



6) R6, получив пакет, понимает, что он виновник всей суматохи. Он уничтожает Path, выделяет метку для FEC 6.6.6.6 и вставляет её как объект Label в ответное сообщение Resv.

Заметьте, до этого шага метки только выделялись, но не распространялись, теперь же они начинают анонсироваться тем LSR, которые их запрашивали.

7) Сообщение RESV продвигается к R1 (Ingress LSR), оставляя за собой растущий хвост LSP. Resv должно пройти через те же узлы, что Path, но в обратном порядке.



8) В конце концов LSP от R1 до 6.6.6.6 сформирован. Данные по нему могут передаваться только от R1 к R6. Чтобы позволить передачу данных в обратном направлении, нужно создать туннельный интерфейс на R6 с адресом назначения 1.1.1.1 — все действия будут точно такими же.

Возникает вопрос — почему адресат пакета Path 6.6.6.6, если передаётся он узел за узлом и их адреса известны? Вопрос этот не праздный — он ведёт нас к одной важной особенности. Объект ERO может на самом деле содержать не все узлы от Ingress LSR до Egress LSR — некоторые могут быть опущены. Поэтому каждый LSR должен знать, куда в итоге направляется сообщение. И происходить это может не потому что Ingress LSR лень просчитать весь путь.

Проблема в зонах IGP. Вы знаете, что и в OSPF и в ISIS существует это понятие для того, чтобы упростить маршрутизацию. В больших сетях (сотни и тысячи узлов) встаёт проблема широковещательных рассылок служебных пакетов и просчёт огромного количества комбинация алгоритмом SPF. Поэтому один глобальный домен делится на зоны маршрутизации.

И вся загвоздка в том, что если внутри зоны IGP и является протоколом по состоянию связи (link-state), то между ними — он самый настоящий дистанционно-векторный — топология сети строится только внутри зоны, любые внутренние маршрутизаторы не знают, как устроены другие зоны — они только поставлены в известность, что для того, чтобы попасть в ту или иную сеть, им нужно отправлять пакеты на конкретный ABR.

Иными словами, если у вас сеть поделена на зоны, то с MPLS TE возникают затруднения — CSPF не может просчитать весь путь, потому что в его топологии адресат из другой зоны — это облако, а не конкретный узел.

И тут на помощь приходит **Explicit Path** (не путать с объектом ERO). Это самый, что ни на есть, прямой способ управления путём построения LSP — администратор может самостоятельно и явно задать те узлы, через которые нужно проложить LSP. Ingress LSR должен точно следовать таким указаниям.

Важно понимать, что технология MPLS никак не регламентирует протокол распространения меток, но конечные результаты на конкретной сети вполне могут различаться при использовании разных протоколов. Вышестоящие протоколы и приложения используют LSP безотносительно того, кем и как они построены.

Кстати нередко в современных сетях встречается сценарий LPD over TE. В этом случае RSVP-TE используется для организации транспорта и реализации Traffic Engineering, а LDP для обмена метками VPN, например.

Egress LSR, записывая в заголовок MPLS первую метку, определяет весь путь пакета. Промежуточные маршрутизаторы просто меняют одну метку на другую. Содержимое может быть совершенно любым. Как раз вот эта мультипротокольность позволяет MPLS служить основой для разнообразных сервисов VPN.