Network Working Group
Request for Comments: 3031
Category: Standards Track

E. Rosen
Cisco Systems, Inc.
A. Viswanathan
Force10 Networks, Inc.
R. Callon
Juniper Networks, Inc.

January 2001

Архитектура MPLS

Multiprotocol Label Switching Architecture

Статус документа

Данный документ содержит спецификацию протокола, предложенного сообществу Internet, и служит запросом к дискуссии в целях развития протокола. Информацию о статусе данного протокола можно найти в текущей редакции документа «Internet Official Protocol Standards» (STD 1). Документ может распространяться свободно.

Аннотация

В этом документе приведена спецификация архитектуры многопротокольной коммутации пакетов по меткам (MPLS[1]).

Оглавление

[1]Multiprotocol Label Switching.

1. Соглашение

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с RFC 2119.

2. Введение в MPLS

В этом документе приведена спецификация архитектуры многопротокольной коммутации по меткам (MPLS).

Отметим, что использование MPLS для групповой адресации требует дополнительного исследования.

2.1. Обзор

Когда пакет протокола сетевого уровня без организации явных соединений проходит от одного маршрутизатора к следующему, каждый маршрутизатор на пути принимает решение о пересылке пакета независимо. Т. е., каждый маршрутизатор анализирует заголовок пакета и на каждом работает алгоритм маршрутизации сетевого уровня. Каждый маршрутизатор независимо выбирает для пакета следующий интервал пересылки (next hop) на основе анализа заголовка пакета и результата работы алгоритма маршрутизации.

Заголовки пакетов содержат существенно больше информации, нежели требуется для выбора следующего интервала. Процесс выбора следующего интервала пересылки можно рассматривать, как композицию двух функций. Первая функция делит весь набор возможных пакетов по классам эквивалентной пересылки (FEC[1]). Вторая функция отображает каждый класс FEC на следующий интервал. В контексте принятия решения о пересылке различные пакеты, отображающиеся на один класс FEC, являются неразличимыми. Все пакеты, относящиеся к определённому FEC и идущие от одного узла, будут следовать по одному пути (или, при использовании маршрутизации по множеству путей, - по одному набору путей), связанному с FEC.

При традиционной пересылке IP отдельный маршрутизатор обычно будет относить два пакета к одному FEC, если в его таблице маршрутизации имеется некий префикс X, который даёт самое длинное совпадение с адресом получателя в каждом из пакетов. По мере прохождения пакета через сеть на каждом интервале будет повторяться проверка пакета и отнесение его к определённому FEC.

В MPLS отнесение пакета к определённому классу FEC происходит однократно при входе пакета в сеть. FEC, к которому отнесён пакет, кодируется коротким целым числом фиксированного размера, которое называют меткой. При пересылке пакета на следующий интервал метка передаётся вместе с пакетом (т. е. пакет «помечается» до пересылки).

На последующих интервалах пересылки анализ заголовков сетевого уровня не выполняется. Взамен этого используется метка в качестве индекса таблицы, задающей следующий интервал пересылки и новую метку. Старая метка заменяется новой и пакет передаётся на следующий интервал.

В парадигме пересылки MPLS после того, как пакет отнесён к FEC, дальнейшего анализа заголовка на следующих маршрутизаторах не требуется - вся пересылка осуществляется по меткам. Это даёт ряд преимуществ по сравнению с традиционной пересылкой на сетевом уровне.

- MPLS-пересылка может выполняться коммутаторами, которые умеют просматривать и менять метки, но не способны анализировать заголовки сетевого уровня или не могут обеспечить достаточной производительности при таком анализе.
- Поскольку пакет связывается с классом FEC на входе в сеть, маршрутизаторы на входе могут использовать для классификации пакетов любую информацию из этих пакетов, даже если эти данные не относятся к заголовку сетевого уровня. Например, пакеты, приходящие на разные порты, могут быть отнесены к разным FEC. При традиционной же пересылке может рассматриваться только информация из заголовков сетевого уровня.
- Пакет, входящий в сеть через определённый маршрутизатор, может классифицироваться иначе, чем такой же пакет, входящий через другой маршрутизатор. В результате легко реализуется пересылка в зависимости от точки входа в сеть. Такое решение невозможно при традиционной пересылке, поскольку информация о точке входа в сеть не передаётся с пакетом.
- Процедуры классификации пакетов по FEC могут усложняться и совершенствоваться без какого-либо влияния на маршрутизаторы, выполняющие пересылку помеченных пакетов.
- Иногда желательно отправить пакет по конкретному маршруту, который явно выбирается до или на этапе входа пакета в сеть, вместо использования обычного алгоритма динамической маршрутизации при передаче пакета через сеть. Это можно сделать в форме правил или средствами организации трафика[2]. При традиционной пересылке такой маршрут потребуется поместить в заголовок и передавать вместе с пакетом (source routing). В MPLS вместо задания маршрута может использоваться метка, которая будет представлять маршрут, что позволяет избавиться от явной передачи маршрута в заголовке пакета.

Некоторые маршрутизаторы анализируют в пакетах заголовок сетевого уровня не только для выбора следующего интервала, но и для определения «предпочтительности» или класса обслуживания пакета. После такого определения маршрутизаторы могут использовать разные пороги отбрасывания или дисциплины очередей для различных пакетов. MPLS разрешает (но не требует) влияние уровня предпочтения или класса обслуживания на выбор маршрута, который обычно определяется меткой. В таких случаях можно говорить, что метка определяется комбинацией FEC и уровня предпочтения или класса обслуживания.

MPLS означает многопротокольную коммутацию, поскольку методы MPLS применимы для **любого** протокола сетевого уровня. Однако в этом документе основной упор делается на использование IP в качестве протокола сетевого уровня.

Маршрутизаторы, поддерживающие MPLS, называют маршрутизаторами с коммутацией по меткам или LSR[3].

2.2. Терминология

В этом параграфе приведён общий концептуальный обзор терминов, используемых в документе. Некоторые из терминов более точно определяются в последующих параграфах документа.

DLCI
: Метка, используемая в сетях Frame Relay для идентификации устройств (соединений).

forwarding equivalence class - класс эквивалентной пересылки
: Группа пакетов IP, которые пересылаются единообразно (например, по одному пути с однотипным обслуживанием).

frame merge - слияние кадров
: Слияние меток, применяемое в средах на основе передачи кадров для того, чтобы не возникало проблем «чередования» ячеек.

label - метка
: Короткий, физически непрерывный идентификатор фиксированного размера, используемый для обозначения FEC (обычно с локальной значимостью).

[1] Forwarding Equivalence Class
[2] Traffic Engineering.
[3] Label Switching Router.

label merging - слияние меток
Замена множества входящих меток для определённого FEC одной исходящей меткой.

label swap - переключение меток
Базовая операция пересылки, включающая просмотр входящей метки для определения исходящей метки, инкапсуляции, порта и других параметров обработки.

label swapping - переключение меток
Парадигма пересылки, обеспечивающая возможность «потоковой» пересылки данных с использованием меток для идентификации класса пакетов, которые при пересылке трактуются, как неразличимые.

label switched hop - интервал коммутации по меткам
Интервал между двумя узлами MPLS, на котором пересылка осуществляется с использованием меток.

label switched path - путь с коммутацией по меткам
Путь через один или множество LSR одного уровня иерархии, по которому следуют пакеты определённого FEC.

label switching router - маршрутизатор с коммутацией по меткам
Узел MPLS, способный пересылать обычные пакеты L3.

layer 2 - уровень 2
Уровень протокола, лежащий ниже уровня 3 (который, следовательно, обеспечивает сервис, используемый уровнем 3). Пересылка при её осуществлении путём коммутации коротких меток фиксированного размера, происходит на уровне 2, независимо от того, будут проверяться ATM VPI/VCI, DLCI или метки MPLS.

layer 3 - уровень 3
Уровень протокола, на котором работает IP и связанные с ним протоколы маршрутизации.

link layer
Синоним layer 2.

loop detection - обнаружение петель
Метод работы, при котором возникновение петель и передача данных через петли возможны, но образовавшиеся петли впоследствии детектируются.

loop prevention - предотвращение петель
Метод работы с петлями, при котором данные никогда не передаются через петлю.

label stack - стек меток
Упорядоченный набор меток.

merge point - точка слияния
Узел, на котором происходит слияние меток.

MPLS domain - домен MPLS
Непрерывное множество узлов, на которых работает маршрутизация и пересылка MPLS, относящихся к одному домену маршрутизации или администрирования.

MPLS edge node - краевой узел MPLS
Узел MPLS, который соединяет домен MPLS с узлом, находящимся за пределами домена, по причине того, что на нем не работает MPLS или данный узел относится к другому домену. Отметим, что если LSR имеет соседний хост, на котором не работает MPLS, данный LSR будет краевым узлом MPLS.

MPLS egress node - выходной узел MPLS
Краевой узел MPLS, который обслуживает трафик, выходящий из домена MPLS.

MPLS ingress node - входной узел MPLS
Краевой узел MPLS, который обслуживает трафик, входящий в домен MPLS.

MPLS label - метка MPLS
Метка, которая передаётся в пакете для представления FEC.

MPLS node - узел MPLS
Узел, на котором работает MPLS. Узел MPLS будет поддерживать протоколы управления MPLS, один или множество протоколов маршрутизации L3, а также возможность пересылки пакетов по меткам. Узел MPLS может также пересылать обычные пакеты L3.

MultiProtocol Label Switching
Рабочая группа IETF и связанная с этой группой деятельность.

network layer - сетевой уровень
Синоним уровня 3.

stack - стек
Синоним стека меток.

switched path - путь коммутации
Синоним пути с коммутацией по меткам.

virtual circuit - виртуальное устройство
Устройство (канал) в основанной на прямых соединениях технологии уровня 2 (например, ATM или Frame Relay), требующее поддержки некой информации о состоянии этого устройства в коммутаторах уровня 2.

VC merge - слияние виртуальных устройств
Слияние меток, при котором метка MPLS переносится в поле ATM VCI (или поля VPI/VCI), что позволяет объединять множество VC в одно виртуальное устройство (VC).

VP merge - слияние виртуальных путей
Слияние меток, при котором метка MPLS переносится в поле ATM VPI, что позволяет объединять множество VP в один виртуальный путь (VP). В этом случае две ячейки будут иметь одно значение VCI лишь при их происхождении от одного узла. Это позволяет распределять ячейки из различных источников по разным VCI.

VPI/VCI - идентификатор виртуального пути/виртуального устройства
Метки, используемые в сетях ATM для идентификации устройств.

2.3. Используемые сокращения

ATM Asynchronous Transfer Mode - асинхронный режим передачи.

BGP Border Gateway Protocol - протокол граничного шлюза.

DLCI Data Link Circuit Identifier - идентификатор соединения на канальном уровне.

FEC Forwarding Equivalence Class - класс эквивалентной пересылки.

FTN FEC to NHLFE Map - отображение FEC на NHLFE.

IGP Interior Gateway Protocol - протокол внутреннего шлюза.

ILM Incoming Label Map - отображение входящих меток.

IP Internet Protocol - протокол Internet (IP).

LDP Label Distribution Protocol - протокол распространения меток.

L2 Layer 2 - уровень 2.

L3 Layer 3 - уровень 3.

LSP Label Switched Path - путь с коммутацией по меткам.

LSR Label Switching Router - маршрутизатор с коммутацией по меткам.

MPLS MultiProtocol Label Switching - многопротокольная коммутация по меткам.

NHLFE Next Hop Label Forwarding Entry - запись для следующего интервала пересылки по меткам.

SVC Switched Virtual Circuit - коммутируемое виртуальное устройство (канал).

SVP Switched Virtual Path - коммутируемый виртуальный путь.

TTL Time-To-Live - время жизни.

VC Virtual Circuit - виртуальное устройство (канал).

VCI Virtual Circuit Identifier - идентификатор виртуального устройства.

VP Virtual Path - виртуальный путь.

VPI Virtual Path Identifier - идентификатор виртуального пути.

2.4. Благодарности

Идеи и текст этого документа были собраны из множества источников и полученных от читателей комментариев. Мы благодарим Rick Boivie, Paul Doolan, Nancy Feldman, Yakov Rekhter, Vijay Srinivasan, и George Swallow за их идеи и вклад в подготовку документа.

3. Основы MPLS

В этом разделе вводятся некоторые базовые концепции MPLS и в общем виде описана используемая модель.

3.1. Метки

Метка представляет собой короткий идентификатор фиксированного размера, имеющий локальную значимость и служащий для обозначения FEC. Метка - это то, что помещается в пакет для связывания этого пакета с определенным классом пересылки FEC.

В общем случае пакет связывается с FEC на основе (полностью или частично) адреса получателя. Однако метка никогда не является представлением этого адреса.

Если маршрутизаторы Ru и Rd являются LSR, они могут договориться между собой, что Ru при передаче пакетов Rd будет маркировать их меткой L тогда и только тогда, когда эти пакеты относятся к некому FEC-классу F. Согласуется «связка» метки L с классом F для пакетов, передаваемых от Ru к Rd. В результате такого соглашения L становится исходящей меткой маршрутизатора Ru для представления класса F и входящей меткой для представления этого класса на маршрутизаторе Rd.

Отметим, что L не обязательно представляет класс F для каких-либо пакетов, кроме тех, которые будут передаваться от Ru к Rd. L - произвольное значение, связь которого с классом F является локальной для пары маршрутизаторов Ru и Rd.

Когда выше мы писали «пакет, передаваемый от Ru к Rd», мы не предполагали, что источником пакета является Ru или конечным получателем является Rd. Пакеты, о которых мы говорим, обычно являются транзитными для обоих LSR.

Иногда маршрутизатору Rd трудно или невозможно сказать для принятого пакета с меткой L, что эта метка была помещена в пакет маршрутизатором Ru, а не другим LSR (обычно это происходит в тех случаях, когда Ru и Rd не являются прямыми соседями). В таких случаях Rd должен быть уверен, что отображение классов FEC на метки является взаимно-однозначным, т. е. для Rd **недопустимо** соглашаться с тем, что Ru1 будет отображать метку L на класс F1, если он уже согласился с тем, что Ru2 уже связал метку L с другим классом F2, в том случае, когда Rd не может надёжно отличить пакеты с меткой L, полученные от Ru1, и пакеты с такой же меткой от Ru2.

Каждый маршрутизатор LSR отвечает за уникальную интерпретацию входящих меток.

3.2. Восходящий и нисходящий LSR

Предположим, что маршрутизаторы Ru и Rd согласовали связывание метки L с FEC-классом F для пакетов, передаваемых от Ru к Rd. В этом случае относительно такой связи маршрутизатор Ru будет «восходящим LSR[1]», а Rd - «нисходящим LSR[2]».

Термины «восходящий» и «нисходящий» применительно к данной связке означают лишь, что конкретная метка представляет конкретный класс FEC для пакетов, передаваемых в направлении от восходящего узла к нисходящему. Это **не** значит, что пакеты данного FEC будут на деле маршрутизироваться от восходящего узла к нисходящему.

[1]Upstream LSR.
[2]Downstream LSR.

3.3. Помеченный пакет

Помеченными называют пакеты со включённой меткой. В некоторых случаях метка размещается в заголовке инкапсуляции, используемом специально для этой цели. В остальных случаях метка может размещаться в имеющемся заголовке сетевого или канального уровня, если там имеется доступное поле. Использование конкретного метода представления меток должно согласовываться обеими сторонами (кодирующей и декодирующей).

3.4. Присваивание и распространение меток

В архитектуре MPLS решение о связывании конкретной метки L с определенным FEC-классом F принимается LSR, который является **нисходящим** относительно данной связки. Нисходящий LSR информирует о созданной связи восходящий LSR. Таким образом метки распределяются нисходящими узлами и распространяются в направлении от нисходящего маршрутизатора к восходящему.

Если LSR может видеть метки лишь из определённого диапазона, нужно обеспечить связывание FEC только с метками из этого диапазона.

3.5. Атрибуты привязки меток

Конкретное связывание метки L с классом F, распространяемое маршрутизатором Rd маршрутизатору Ru, может иметь некие «атрибуты». Если Ru, действуя как нисходящий LSR, также распространяет связывание метки с FEC-классом F, при некоторых условиях от него может потребоваться также распространение соответствующего атрибута, полученного от Rd.

3.6. Протокол распространения меток

Протокол распространения меток представляет собой набор процедур, с помощью которых один маршрутизатор LSR информирует другие маршрутизаторы о связывании меток с классами FEC. Два LSR, использующих протокол распространения меток для обмена информацией о связках «метка-FEC», называют партнёрами по распространению меток. Если два LSR являются партнёрами по распространению меток, мы будем говорить о наличии между ними «смежности по распространению меток».

Важно отметить, что два LSR могут быть партнёрами по распространению меток применительно к некоторому множеству связок, не являясь партнёрами для остальных связок.

Протокол распространения меток включает также все согласования, которые требуются двум партнёрам для определения возможностей друг друга в части MPLS.

Для архитектуры недопустимо предположение о существовании единственного протокола распространения меток. Фактически может быть стандартизовано множество таких протоколов. Существующие протоколы могут расширяться путём добавления в них поддержки распространения меток (см., например, [MPLS-BGP], [MPLS-RSVP-TUNNELS]). Могут также разрабатываться новые протоколы именно для распространения меток (см., например, [MPLS-LDP], [MPLS-CR-LDP].

В этом документе аббревиатура LDP будет использоваться преимущественно для обозначения протокола, определённого в [MPLS-LDP].

3.7. Распространение меток по запросам и без запроса

Архитектура MPLS позволяет LSR явно запрашивать от следующего интервала информацию о метке для конкретного класса FEC. Такой механизм называется «нисходящим распространением меток по запросу».

Архитектура MPLS допускает также распространение маршрутизаторами LSR информации о связках без явного запроса такой информации. Такой механизм называется «незапрошенным распространением».

Предполагается, что некоторые реализации MPLS будут поддерживать только распространение по запросу, другие - только незапрошенное распространение, а часть маршрутизаторов будет поддерживать оба механизма. Выбор механизмов может зависеть от характеристик интерфейсов, поддерживаемых конкретной реализацией. Оба упомянутых механизма могут использоваться в сети одновременно. Для любой конкретной пары смежных маршрутизаторов нисходящий и восходящий LSR согласуют между собой используемый механизм.

3.8. Режимы удерживания меток

LSR Ru может получать (или получает) метку для конкретного FEC от LSR Rd, даже в тех случаях, когда Rd не является (или перестал быть) для Ru следующим интервалом применительно к данному FEC.

Ru в таком случае может сохранять полученную информацию о метках или отбрасывает её после использования. Если Ru сохраняет информацию о метках, он может в любой момент возобновить использование сохранённой метки, когда Rd снова становится потенциальным следующим интервалом для рассматриваемого FEC. Если Ru отбрасывает информацию о метках, тогда при восстановлении Rd в качестве следующего интервала метку нужно будет получать заново.

Если LSR поддерживает «либеральный режим удержания меток», он сохраняет информацию о связях меток с FEC, полученную от LSR, которые не являются следующим интервалом для данного FEC. Если LSR поддерживает «консервативный режим удержания меток», информация о таких метках будет отбрасываться.

Либеральный режим позволяет быстрее адаптироваться при смене картины маршрутизации, а консервативный не требует хранения большого числа меток.

3.9. Стек меток

Выше мы говорили о пакетах, которые содержат единственную метку. Далее мы покажем, что в общем случае будет полезна обобщённая модель, в которой помеченный пакет может содержать множество меток, организованных в стек[1]. Будем называть это стеком меток.

Хотя, как будет показано далее, MPLS поддерживает иерархию меток, обработка помеченных пакетов полностью независима от уровня иерархии. Обработка всегда осуществляется по верхней[2] метке, несмотря на то, что ранее в стеке над этой меткой могли находиться другие метки и некоторое количество меток может располагаться в стеке под верхней меткой.

Пакеты без меток можно рассматривать, как пакеты с пустым стеком меток (т. е., стеком глубиной 0).

Если стек меток имеет глубину m, нижнюю метку будем называть меткой уровня 1, расположенную непосредственно над ней - меткой уровня 2 (если такая метка присутствует), а верхняя метка будет иметь уровень m.

Эффективность использования стека меток станет понятной при обсуждении туннелей LSP и иерархии MPLS (см. параграф 3.27. Туннели и иерархия).

3.10. NHLFE

При пересылке помеченных пакетов используется элемент NHLFE, содержащий:

1. следующий интервал для пересылки пакета;
2. операцию, выполняемую над стеком меток, в качестве каковой может использоваться:
   a) замена верхней метки в стеке заданной новой меткой;
   b) выталкивание метки из стека;
   c) замена верхней метки в стеке заданной новой меткой и включение в стек одной или множества новых меток.

NHLFE может также включать:

d) инкапсуляцию канального уровня для использования при передаче пакета;

e) способ представления стека меток при передаче пакета;

f) другую информацию, которая может потребоваться для корректной обработки пакета.

Отметим, что для данного LSR следующим интервалом пересылки пакетов может быть он сам. В этом случае LSR должен будет вытолкнуть из стека верхнюю метку и «переслать» после этого пакет самому себе. Далее маршрутизатор будет принимать новое решение о пересылке на основе оставшихся в стеке меток или заголовка IP, если стек пуст.

Это означает, что в некоторых случаях маршрутизатору LSR для пересылки пакета может потребоваться работа с заголовком IP.

Если следующим интервалом пересылки пакета является текущий LSR, в качестве операции над стеком **должно** быть указано выталкивание метки.

3.11. ILM

ILM отображают входящие метки на NHLFE. Это отображение используется при пересылке помеченных пакетов.

Если ILM отображает конкретную метку на множество NHLFE, включающее более одного элемента, для пересылки пакета нужно выбрать единственный элемент из такого множества. Процедуры выбора элемента из множества выходят за рамки документа. Наличие ILM-отображения метки на множество, содержащее более одного NHLFE, может быть полезно, например, в тех случаях, когда желательно распределить нагрузку между равноценными путями.

3.12. Отображение FEC на NHLFE (FTN)

FTN отображает каждый класс FEC на множество NHLFE. Такие отображения используются при пересылке пакетов, которые были получены без меток, но должны быть помечены до пересылки.

Если FTN отображает метку на множество NHLFE, содержащее более одного элемента, до пересылки пакета должен быть выбран единственный элемент из такого множества. Процедуры выбора элемента из множества выходят за рамки документа. Наличие FTN-отображения FEC[3] на множество, содержащее более одного NHLFE может быть полезно, например, в тех случаях, когда желательно распределить нагрузку между множеством равноценных путей.

3.13. Замена меток

Замена меток (Label swapping) представляет собой использование перечисленных ниже процедур при пересылке пакета.

- Для пересылки помеченного пакета LSR смотрит метку в стеке и с помощью ILM отображает эту метку на NHLFE. Используя данные NHLFE, маршрутизатор определяет, куда следует пересылать пакет, и выполняет требуемые операции над стеком меток пакета. Новый стек меток помещается в пакет и после этого выполняется пересылка.
- При пересылке пакета без метки LSR анализирует заголовок сетевого уровня для отнесения пакета к тому или иному классу FEC. С помощью FTN маршрутизатор отображает пакет на NHLFE. Используя данные NHLFE,

[1] Метка, помещённая в стек первой, извлекается из него последней и наоборот.
[2] В стеке. *Прим. перев.*
[3] В оригинале ошибочно указано «label» вместо «FEC». См. http://www.rfc-editor.org/errata_search.php?eid=2992

маршрутизатор определяет, куда следует пересылать пакет, и выполняет требуемые операции над стеком меток пакета (выталкивание метки из стека в этом случае будет некорректной операцией). Новый стек меток помещается в пакет и после этого выполняется пересылка.

Важно отметить, что при использовании замены меток следующий интервал пересылки всегда берётся из NHLFE. В некоторых случаях полученный таким путём следующий интервал пересылки будет отличаться от того интервала, который будет найден, если MPLS не используется.

3.14. Зона действия и актуальность меток

LSR-маршрутизатор Rd может связать метку L1 с FEC-классом F и сообщить об этой связке своему партнёру по распространению меток Ru1. Маршрутизатор Rd может также связать метку L2 с FEC-классом F и распространить информацию об этой связке другому партнёру Ru2. Выполнение для таких случаев условия L1 == L2 не задаётся архитектурой и определяется локально.

Данный LSR-маршрутизатор Rd может связать метку L с FEC-классом F1 и распространить информацию об этой связке своему партнёру Ru1. Маршрутизатор Rd может также связать метку L с классом F2 и распространить информацию об этой связке другому партнёру Ru2. **Если (и только в этом случае) RD может определить при получении пакета с верхней меткой L, отправлен этот пакет маршрутизатором RU1 или маршрутизатором RU2, то архитектура не требует выполнения условия F1 == F2.** В таких случаях мы будем говорить, что Rd распространяет маршрутизаторам Ru1 и Ru2 разные пространства меток.

В общем случае Rd может определить, получен пакет с верхней меткой L от маршрутизатора Ru1 или маршрутизатора Ru2 только при выполнении обоих приведённых ниже условий.

- Ru1 и Ru2 исчерпывают число партнёров, которым Rd распространил информацию о связывании метки L.
- Каждый из маршрутизаторов Ru1 и Ru2 подключён к Rd через интерфейс «точка-точка».

При выполнении этих условий LSR может использовать метки с зоной действия в пределах одного интерфейса (т. е. уникальность меток требуется лишь в рамках отдельного интерфейса). Мы можем в таком случае говорить, что LSR использует «пространство меток для интерфейса». Когда приведённые выше условия не выполняются, метки должны обеспечивать уникальность в масштабе LSR, который выделяет их. В таких случаях можно говорить, что LSR использует «пространство меток для платформы».

Если некий LSR Rd подключён к другому LSR Ru через два интерфейса «точка-точка», Rd может распространять маршрутизатору Ru информацию о привязке метки L к FEC-классу F1, а также о привязке этой метки к классу F2 (F1 != F2) тогда и только тогда, когда каждая из привязок относится лишь к пакетам, которые Ru передаёт Rd только через один из интерфейсов. В остальных случаях маршрутизатору Rd **недопустимо** распространять Ru связывание одной метки с двумя разными классами FEC.

Этот запрет сохраняется и в тех случаях, когда привязки осуществляются на разных «уровнях иерархии». В MPLS не вводится нотации, обеспечивающей разные пространства меток для разных уровней иерархии, - при интерпретации метки она рассматривается независимо от уровня иерархии.

Возникает вопрос о возможности для LSR использовать множество пространств меток для платформы или множество пространств для интерфейса на одном интерфейсе. Архитектура не запрещает такого использования. Однако в таких случаях LSR должен обеспечивать некий (зависящий от архитектуры) способ определения принадлежности входящих меток к тому или иному пространству. Например, [MPLS-SHIM] задаёт использование разных пространств меток для индивидуальной (unicast) и групповой (multicast) адресации, а для определения принадлежности метки к тому или иному пространству служат коды канального уровня.

3.15. LSP, LSP Ingress, LSP Egress

Путь с коммутацией по меткам (LSP) уровня m для пакета P представляет последовательность маршрутизаторов

<R1, ..., Rn>,

которая обладает перечисленными ниже свойствами.

1. R1 - LSP Ingress[1] - маршрутизатор LSR, который помещает метку в стек P, увеличивая глубину стека до m.
2. Для всех i от 1 до n (1<i<n) пакет P имеет стек меток глубиной m при получении этого пакета маршрутизатором LSR Ri.
3. Ни в какой момент передачи P от R1 до R[n-1] глубина стека меток не становится меньше m.
4. Для всех i от 1 до n (1<i<n) маршрутизатор Ri передаёт P маршрутизатору R[i+1] с помощью MPLS, т. е. используя верхнюю метку стека (метку уровня m) в качестве индекса ILM.
5. Если при любом i от 1 до n (1<i<n) система S получает и пересылает P после того, как пакет P был передан Ri, но до того, как P был принят R[i+1] (например, Ri и R[i+1] могут быть соединены через коммутируемую на канальном уровне подсеть, а S является одним из коммутаторов), решение системы S о пересылке пакета принимается без использования метки уровня m или информации из заголовка сетевого уровня. Это можно осуществить двумя путями:
   a) решение о пересылке принимается без использования стека меток или заголовков сетевого уровня;
   b) решение принимается на основе стека меток, в который были помещены дополнительные метки (например, на основе метки уровня m+k, где k>0).

Иными словами, мы можем говорить о LSP уровня m для пакета P, как о последовательности маршрутизаторов, которая:

1. начинается с маршрутизатора LSR, помещающего метку на уровень m (LSP Ingress);

[1] Вход пути

2. состоит из промежуточных маршрутизаторов LSR, принимающих решение о пересылке по метке уровня m;
3. заканчивается маршрутизатором (LSP Egress[1]), принимающим решение о пересылке по метке уровня m-k (где k>0) или традиционным способом без использования процедур MPLS.

Следствием этого (или, возможно, допущением) является то, что при присвоении маршрутизатором LSR метки ранее помеченному пакету, нужно быть уверенным в том, что новая метка соответствует классу FEC, для которого в качестве LSP Egress служит LSR, присвоивший метку, ставшую второй в стеке.

Будем называть последовательность LSR «LSP для FEC-класса F», если эта цепочка представляет собой LSP уровня m для некого пакета P, когда метка уровня m в пакете P является меткой, соответствующей FEC-классу F.

Рассмотрим набор узлов, которые могут быть входами LSP для FEC-класса F. Тогда существует LSP для FEC-класса F, который начинается на каждом из таких узлов. Если эти LSP имеют общий выход, можно рассматривать набор таких LSP, как дерево, корнем которого является LSP Egress[2]. Мы можем, таким образом, говорить о дереве LSP для конкретного FEC-класса F.

3.16. «Выталкивание» на предпоследнем интервале

Отметим, что в соответствии с определением в параграфе 3.15 пакет может передаваться от R[n-1] к Rn со стеком меток глубиной m-1, если <R1, ..., Rn> представляет собой LSP уровня m для пакета P. Т. е. выталкивание метки из стека может происходить на предпоследнем LSR пути LSP, а не на узле LSP Egress.

С точки зрения архитектуры это совершенно нормально. Метка уровня m служит для доставки пакета маршрутизатору Rn. Как только R[n-1] принимает решение о передаче пакета маршрутизатору Rn, метка становится ненужной для выполнения каких-либо функций и её не требуется передавать дальше.

Выталкивание метки на предпоследнем этапе даёт и практические преимущества. Если такого выталкивания не происходит, узел LSP Egress при получении пакета сначала смотрит метку верхнего уровня и определяет в результате, что метка предназначена LSP Egress. Эту метку нужно вытолкнуть и просмотреть оставшуюся часть стека. Если в стеке есть другая метка, выходной узел будет пересылать пакет на основе этой метки. В этом случае выходной узел для пакетов LSP уровня m служит также промежуточным узлом для LSP уровня m-1. Если в стеке больше нет меток, пакет пересылается по адресу в заголовке сетевого уровня. Отметим, что выходному узлу требуется выполнять просмотр **дважды** (просмотр двух меток или метки и заголовка сетевого уровня).

Если используется выталкивание метки из стека на предпоследнем этапе, узел этого этапа просматривает метку и определяет:

- что данный узел является предпоследним интервалом пути;
- а также адрес следующего интервала.

После этого предпоследний узел выталкивает метку из стека и пересылает пакет на основе информации, полученной при просмотре метки, которая была верхней в стеке. Когда узел LSP Egress получает такой пакет, верхней меткой в стеке будет та, которая требуется данному узлу для принятия решения о пересылке. Для случая, когда перед выталкиванием на предпоследнем этапе в стеке была единственная метка, узел LSP Egress будет обращаться для принятия решения о пересылке к заголовку сетевого уровня в пакете.

Этот механизм позволяет на выходном узле выполнять единственный просмотр. На предпоследнем узле также достаточно одного просмотра.

Создание «быстрого пути пересылки» в системах с коммутацией по меткам может быть существенно упрощено, если требуется только один просмотр:

- код может быть существенно упрощён, если не требуется многократного просмотра;
- код может базироваться на «временном бюджете», предполагающем единственный просмотр.

Фактически, при выталкивании метки из стека узел LSP Egress может даже не быть LSR.

Однако часть коммутационного оборудования не способна выталкивать метки из стека, поэтому описанный механизм не может быть универсальным. Кроме того, могут возникать ситуации, когда выталкивание метки из стека на предпоследнем этапе нежелательно. В результате предпоследние узлы выталкивают метку из стека только в тех случаях, когда это явно запрашивает выходной узел **или** следующий узел LSP не поддерживает MPLS[3].

LSR, способный выталкивать метку из стека, во всех случаях **должен** выполнять такое выталкивание только по запросу своего нисходящего партнёра по распространению меток.

Начальное согласование протокола распространения **должно** обеспечивать каждому LSR возможность определения способности выталкивания меток из стека у смежных LSR. Для LSR **недопустимо** запрашивать у партнёра выталкивание метки из стека, если тот не объявил о поддержке такого выталкивания.

Может возникнуть вопрос о корректности интерпретации выходным узлом верхней метки в стеке для тех случаев, когда предпоследний узел выталкивает метку. Если выполняются требования параграфа 3.14 в части уникальности и зоны действия меток, при интерпретации верхней метки стека неоднозначности не возникает.

3.17. Следующий интервал LSP

Следующим интервалом пути LSP для помеченного пакета в конкретном LSR будет маршрутизатор LSR, который является Next Hop, выбираемым по записи NHLFE для пересылки пакета.

---

[1] Выход пути.

[2] Поскольку данные проходят через это дерево в направлении корня, будем называть это дерево «множество в один» - multipoint-to-point tree.

[3] Если следующий узел LSP поддерживает MPLS, но не делает запроса на выталкивание метки, предпоследний узел не имеет надёжного механизма идентифицировать себя в качестве предпоследнего узла.

LSP Next Hop для конкретного FEC является следующий интервал, выбранный по записи NHLFE, которая индексируется меткой, соответствующей данному FEC.

Отметим, что LSP Next Hop может отличаться от следующего интервала, который будет выбран алгоритмом маршрутизации на основе заголовка сетевого уровня. Для последнего далее будет использоваться термин «следующий интервал L3».

3.18. Некорректные входящие метки

Что следует делать маршрутизатору LSR, если он получает пакет с меткой, для которой у него нет информации о связывании? Заманчиво счесть этот пакет непомеченным и переслать, как обычный пакет IP. Однако в некоторых случаях такой подход может приводить к возникновению петель. Если восходящий маршрутизатор LSR считает, что метка привязана к явному маршруту, а нисходящий LSR не связывает метку ни с чем и последующая поэтапная маршрутизация пакета IP без метки возвращает этот пакет на восходящий LSR, возникает маршрутная петля.

Возможны также ситуации, когда метка предназначена для представления маршрута, который не может быть получен из заголовка IP.

Следовательно, при получении пакета с некорректной меткой такой пакет **должен** быть отброшен, если явно не задано иное поведение для пересылки таких пакетов (рассмотрение этого вопроса выходит за рамки этой спецификации).

3.19. Независимое и упорядоченное управление LSP

Некоторые классы FEC соответствуют адресным префиксам, распространяемым с использованием алгоритма динамической маршрутизации. Организация LSP для таких FEC может быть выполнена двумя способами - Independent LSP Control (независимое управление) или Ordered LSP Control (упорядоченное управление).

В независимом варианте каждый маршрутизатор LSR, принимая решение об идентификации некого класса, независимо принимает решение о связывании метки с данным FEC и передаёт информацию об этой связке своим партнёрам по распространению. Это соответствует механизмам традиционной маршрутизации дейтаграмм IP, когда каждый узел независимо от других принимает решение о трактовке пакета и применяет алгоритм маршрутизации для быстрого построения картины пересылки дейтаграмм.

В упорядоченном варианте LSR связывает метку с определенным классом FEC только в том случае, когда он является выходным маршрутизатором для FEC или связка для этого FEC уже получена от следующего партнёра для FEC.

Если нужно обеспечить передачу трафика некого класса FEC по пути с заданным набором свойств (например, запрет двухкратного прохождения трафика через какой-либо узел, выделяемые для обработки трафика ресурсы, явно заданный маршрут и т. п.), следует применять согласованный режим. При независимом управлении те или иные LSR могут начать коммутацию трафика по меткам для некого FEC до завершения организации LSP и, таким образом, часть трафика для данного FEC может пройти по пути, не обладающему требуемым набором свойств. Упорядоченный режим может потребоваться также в тех случаях, когда идентификация FEC происходит в результате организации соответствующего LSP.

Упорядоченная организации LSP может быть инициирована как входными, так и выходными узлами.

Упорядоченный и независимый режимы могут взаимодействовать. Однако если не все LSR, входящие в LSP, используют упорядоченный режим, интегральное воздействие на поведение сети будет больше, чем при независимом режиме, поскольку нет уверенности в том, что LSP не начнёт использоваться до завершения организации пути.

Архитектура позволяет выбирать упорядоченный или независимый режим локально. Поскольку оба режима могут взаимодействовать между собой, любому LSR достаточно поддерживать один из режимов. В общем случае выбор упорядоченного или независимого режима не оказывает влияния на механизмы распределения меток, которые требуется определить.

3.20. Агрегирование

Одним из способов деления трафика по классам FEC является создание отдельного FEC для каждого адресного префикса, присутствующего в таблице маршрутизации. Однако в рамках отдельного домена MPLS такой подход может приводить к тому, что трафик для всех классов FEC пойдёт по одному маршруту. Например, набор адресных префиксов может иметь общий выходной узел и переключение меток может использоваться только для направления трафика на выходной узел. В этом случае внутри домена MPLS объединение этих классов FEC само является FEC. Возникает дилемма - следует ли разделять метки, связанные с каждой компонентой FEC, или можно использовать метку объединения классов для всего трафика, относящегося к объединённому классу?

Процедура связывания одной метки с объединением классов FEC, которое также является FEC (в неком домене), и использование этой метки для всего трафика объединённого класса называется агрегированием. Архитектура MPLS допускает агрегирование. При агрегировании может снижаться число меток, которые нужны для обслуживания некого множества пакетов, а также объем служебного трафика для распространения меток.

Для данного набора FEC, являющегося агрегируемым в один класс FEC, можно (a) агрегировать множество классов с один FEC, (b) агрегировать их во множество классов FEC или (c) не использовать агрегирования. Таким образом, можно говорить о детализации агрегирования, которая может быть грубой (случай a) и тонкой (случай b[1]).

При использовании упорядоченного режима каждый маршрутизатор LSR следует адаптировать для данного набора FEC с учётом используемой следующим маршрутизатором детализации для этих классов FEC.

При использовании независимого режима возможны ситуации, когда два смежных LSR-маршрутизатора Ru и Rd будут по разному агрегировать один и тот же набор FEC.

Если Ru задаёт более тонкую детализацию, нежели Rd, проблем возникать не будет, поскольку Ru распространяет для данного набора FEC больше меток, чем Rd. Это означает, что для передачи соответствующих пакетов от Ru к Rd,

[1] В оригинале ошибочно указано (c). См. http://www.rfc-editor.org/errata_search.php?eid=3171

будет выполняться отображение n меток на m меток и n > m. Ru может отозвать набор из n распространённых им меток и потом распространить набор из m меток в соответствии с детализацией Rd. Это не требуется для обеспечения корректной работы, но несколько снижает число меток, распространяемых Ru (маршрутизатор Ru не получает каких-либо преимуществ в результате распространения большего числа меток). Решение вопроса о снижении числа распространяемых меток принимается локально.

Если Ru использует более грубую детализацию по сравнению с Rd (т. е., Rd распространяет для набора FEC n меток, а Ru - m и n > m), возникает два варианта:

- Принять более тонкую детализацию Rd. Для этого требуется отозвать m распространённых меток и распространить n. Этот вариант является предпочтительным.
- Можно просто отобразить m меток на подмножество из n меток маршрутизатора Rd, если ясно, что маршрутизация в результате не изменится. Предположим, что Ru использует одну метку для всего трафика, который требуется передать через некий выходной узел LSR, а Rd связывает с таким трафиком множество разных меток в зависимости от индивидуальных получателей пакетов. Если Ru знает адрес выходного маршрутизатора и Rd связывает метку с классом FEC, который идентифицирует этот адрес, Ru может просто использовать соответствующую метку.

В любом случае каждый LSR должен знать (из конфигурации), какой уровень детализации применяется для присваиваемых им меток. При использовании упорядоченного режима каждый узел должен знать только детализацию для FEC, выходящих на данном узле из сети MPLS. Для независимого режима наилучший результат может быть достигнут в тех случаях, когда все LSR настроены согласованно и знают детализацию для каждого FEC. Однако во многих случаях задача может быть решена путём использования одной метки, детализация которой применима для всех FEC (одна метка на префикс IP в таблице маршрутизации или одна метка на выходной узел).

3.21. Выбор маршрута

Выбором маршрута будем называть метод, используемый при определении LSP для конкретного FEC. Рассматриваемая архитектура MPLS поддерживает два варианта выбора маршрута - (1) поэтапная маршрутизация и (2) явное задание маршрута.

Поэтапная маршрутизация позволяет каждому узлу независимо выбирать следующий интервал для каждого FEC. В современных сетях IP это обычная парадигма маршрутизации. Поэтапно маршрутизируемый LSP[1] представляет собой LSP, для которого маршрут выбирается с использованием поэтапной маршрутизации.

В LSP с явной маршрутизацией каждый LSR уже не может независимо выбирать следующий интервал. Вместо этого один маршрутизатор LSR (обычно на входе или выходе LSP) задаёт несколько (или все) маршрутизаторы LSR на пути LSP. Если один LSR задаёт LSP целиком, такой LSP «строго» является явно маршрутизируемым. Если один LSR задаёт только часть LSP, такой путь является «нестрого» явно маршрутизируемым.

Последовательность LSR, по которой следует LSP с явной маршрутизацией, может быть задана в конфигурации или динамически выбрана одним узлом (например, выходной узел может использовать топологические данные, полученные от базы данных о состоянии каналов для расчёта пути через все дерево, заканчивающееся на этом узле).

Явная маршрутизация может быть полезна во многих случаях для решения таких задач, как маршрутизация на базе правил или организация трафика. В MPLS явная маршрутизация должна задаваться к моменту выделения метки, но явный маршрут не задаётся для каждого пакета IP. Это делает явную маршрутизацию в MPLS более эффективной, нежели традиционный вариант IP source routing.

Процедуры использования заданных явно (строго или нестрого) маршрутов выходят за рамки этого документа.

3.22. Отсутствие исходящей метки

Когда помеченный пакет проходит вдоль LSP, может случиться так, что пакет придёт на LSR, где ILM не отобразит входящую метку пакета на NHLFE, несмотря на корректность самой метки. Это может быть вызвано неустойчивостью в сети или ошибкой в LSR, который должен быть следующим интервалом для пакета.

В таких случаях возникает желание отказаться от стека меток и попытаться переслать пакет традиционным способом по заголовку сетевого уровня. Однако в общем случае такое решение небезопасно по двум причинам:

- если пакет следовал по LSP с явной маршрутизацией, может возникнуть маршрутная петля;
- в заголовке может оказаться недостаточно информации для того, чтобы LSR смог корректно переслать пакет.

При отсутствии убеждённости (процедуры такой проверки выходят за рамки этого документа) в том, что ни одна из перечисленных ситуаций не возникает, единственной безопасной процедурой является отбрасывание пакета.

3.23. Время жизни (TTL)

При традиционной пересылке IP используется поле TTL[2] в заголовке каждого пакета. При прохождении пакета через маршрутизатор тот уменьшает значение поля TTL на 1. Если в процессе доставки пакета поле TTL достигло нулевого значения, пакет отбрасывается.

Ограничение времени жизни пакетов обеспечивает некоторую защиту от маршрутных петель, возникающих при некорректной конфигурации или в результате медленного схождения алгоритма динамической маршрутизации. Значение TTL иногда используется для других функций, таких как просмотр области групповой адресации или трассировка пути. В связи с этим в MPLS возникает два аспекта, относящихся к TTL - (i) предотвращение маршрутных петель и (ii) выполнение других функций, например, ограничения области видимости пакетов.

[1] Hop by hop routed LSP.
[2] Time To Live.

При прохождении пакетов через LSP, значение TTL **следует** изменять так же, как оно изменилось бы при прохождении через такую же последовательность маршрутизаторов без коммутации меток. Если пакет проходит через иерархию LSP, в значении TTL **следует** отражать общее число LSR на пути через иерархию LSP.

Способ обслуживания TTL может существенно зависеть от того, передаются значения меток MPLS в специфическом для MPLS shim-заголовке [MPLS-SHIM] или в заголовке канального уровня (как в ATM [MPLS-ATM] или Frame Relay [MPLS-FRMRLY]).

Если метка помещается в shim-заголовок между заголовками канального и сетевого уровня, этот дополнительный заголовок **должен** иметь поле TTL, в которое изначально **следует** помещать значение поля TTL из заголовка сетевого уровня, а в дальнейшем это значение **следует** уменьшать на 1 на каждом интервале LSR, копируя изменённое значение в заголовок сетевого уровня на выходе LSP.

Если метка размещается в заголовке канального уровня (например, поле VPI/VCI в заголовке ATM AAL5) и помеченные пакеты пересылаются коммутатором L2 (например, ATM), а сам канальный уровень (как в случае ATM) не имеет поля TTL, значение времени жизни невозможно изменить на каждом интервале LSR. Сегмент LSP, состоящий из последовательности LSR, не способных декрементировать значение TTL, будем называть «сегментом LSP без поддержки TTL».

При прохождении пакетов через сегменты без поддержки TTL **следует**, тем не менее, отражать в значении TTL число пройденных интервалов LSR. При использовании индивидуальной (unicast) адресации это можно сделать путём распространения информации о протяжённости LSP входным узлам пути, которые в этом случае смогут уменьшить соответственно значение TTL до пересылки пакета в сегмент без поддержки TTL.

Иногда на входе в сегмент LSP без поддержки TTL можно определить, что время жизни пакета истечёт до того, как он покинет данный сегмент. В таких случаях для маршрутизатора LSR на входе сегмента без поддержки TTL недопустимо коммутировать пакет. Это означает, что требуется разработать специальные процедуры для поддержки трассировки (например, пакеты трассировки могут пересылаться с использованием традиционной парадигмы поэтапной пересылки).

3.24. Контроль петель

В сегментах без поддержки TTL значение TTL по определению не может использоваться для предотвращения петель. Важность контроля петель может зависеть от оборудования, используемого для обеспечения функциональности LSR в сегменте без поддержки TTL.

В качестве примера предположим, что для коммутации по меткам будет использоваться коммутационное оборудование ATM и метки будут помещаться в поле VPI/VCI. Поскольку коммутаторы ATM не могут декрементировать поле TTL, в этом случае предотвращение петель не поддерживается. Если оборудование ATM обеспечивает беспристрастную буферизацию для входящих ячеек с различными значениями VPI/VCI, петли не будут оказывать негативного влияния на остальной трафик. Если оборудование ATM не может обеспечить такой буферизации, даже временное существование петель может приводить к существенному снижению производительности LSR.

Даже при наличии доступа к буферу потребуются те или иные способы детектирования петель, которые превышают допустимый размер. В дополнение к этому даже в тех случаях, когда поле TTL и независимая буферизация для каждого VC обеспечивают предотвращение негативного влияния петель, желательно иметь механизм предотвращения организации LSP с петлями. Все LSR, которые могут быть присоединены к сегментам LSP без поддержки TTL, должны, следовательно, поддерживать общий метод обнаружения петель. Однако использование механизмов предотвращения петель является необязательным. Методы детектирования петель описаны в [MPLS-ATM] и [MPLS-LDP].

3.25. Представление меток

Для передачи стека меток вместе с пакетом необходимо определить конкретные способы представления стека меток. Архитектура поддерживает несколько вариантов кодирования меток. Выбор конкретного варианта зависит от типа устройства, служащего для пересылки помеченных пакетов.

3.25.1. Специальное оборудование и программы MPLS

При использовании для пересылки помеченных пакетов специального оборудования и/или программ MPLS наиболее очевидным способом представления стека меток является определение нового протокола, который будет обеспечивать «прослойку» между сетевым и канальным уровнем. В реальности эта прослойка будет просто обеспечивать инкапсуляцию пакетов сетевого уровня. Прослойка будет протокольно-независимой, поскольку она обеспечит возможность инкапсуляции любых протоколов сетевого уровня. Будем называть это «базовой инкапсуляцией MPLS».

После базовой инкапсуляции MPLS будет использоваться обычная инкапсуляция в протокол канального уровня.

Спецификация базовой инкапсуляции MPLS дана в [MPLS-SHIM].

3.25.2. Коммутаторы ATM в качестве LSR

Отмечено, что процедуры пересылки MPLS похожи на те, что используются традиционными системами с «переключением по меткам» типа коммутаторов ATM. Коммутаторы ATM используют идентификатор входного порта и входящие значения VPI/VCI в качестве индекса для таблицы кросс-соединений, из которой определяется номер выходного порта и исходящие значения VPI/VCI. Следовательно, при размещении одной или множества меток непосредственно в полях, доступных традиционным коммутаторам, последние можно путём обновления программных компонент превратить в LSR. Будем называть такие устройства ATM-LSR.

Имеется три очевидных способа кодирования меток в заголовке ячеек ATM (предполагается адаптация AAL5).

1. Кодирование SVC.

   Используется поле VPI/VCI для кодирования метки, находящейся на вершине стека. Этот способ может применяться в любых сетях. При таком кодировании каждый путь LSP реализуется в форме ATM SVC, а

протоколом распространения меток служит протокол сигнализации ATM. При таком кодировании меток устройства ATM-LSR не способны выполнять операций по вталкиванию и выталкиванию меток из стека.

2. Кодирование SVP.

   Используется поле VPI для кодирования верхней метки стека и поле VCI - для второй метки. Этот метод даёт некоторые преимущества в сравнении с предыдущим, поскольку он позволяет использовать принятую в ATM коммутацию VP. В результате LSP реализуются как ATM SVP, а в качестве протокола распространения меток служит сигнальный протокол ATM.

   Однако такой метод в ряде случаев не может использоваться. Если сеть ATM включает виртуальный путь через сегмент без поддержки MPLS, поле VPI может оказаться недоступным для использования в MPLS.

   При использовании этого метода представления меток ATM-LSR на выходе VP может выполнять операцию выталкивания метки из стека.

3. Кодирование SVP Multipoint.

   Используется поле VPI для кодирования верхней метки стека, часть поля VCI для кодирования второй метки (если та имеется), а оставшаяся часть VCI идентифицирует вход LSP. При использовании этого метода традиционные возможности коммутации виртуальных путей ATM могут применяться для организации VP «точка-многоточка». Ячейки из разных пакетов будут содержать разные значения VCI. Как будет показано в параграфе 3.26, это позволяет выполнять слияние меток без возникновения проблемы чередования ячеек в коммутаторах ATM с поддержкой VP «точка-многоточка», но без поддержки слияния VC.

   Этот метод зависит от возможности присваивания 16-битовых значений VCI на каждом коммутаторе, чтобы одно значение VCI не назначалось двумя разными коммутаторами ATM. Если каждый коммутатор может выделить адекватное количество таких значений, можно применять поле VCI для второй метки в стеке.

Если число меток в стеке превышает возможности размещения в заголовке ATM, следует дополнительно применять базовую инкапсуляцию.

3.25.3. Взаимодействие методов кодирования

Если <R1, R2, R3> представляет собой сегмент LSP, возможны случаи, когда R1 будет использовать при передаче пакета P маршрутизатору R2 некий метод кодирования, но R2 будет передавать пакет P маршрутизатору R3 с использованием иного варианта. В общем случае архитектура MPLS поддерживает LSP с разными вариантами кодирования меток на разных интервалах пути. Следовательно, обсуждение процедур обработки пакетов происходит в абстрактных терминах работы со стеком меток пакета. При получении помеченного пакета LSR должен декодировать пакет для определения текущего стека меток, после чего маршрутизатор должен обработать стек для определения нового стека и закодировать этот новый стек с использованием подходящего метода до передачи пакета на следующий интервал.

К сожалению, коммутаторы ATM не имеют возможности простого перевода от одного способа кодирования меток к другому. Поэтому архитектура MPLS требует, чтобы пара смежных коммутаторов ATM, функционирующих в качестве LSR на пути LSP для некого пакета, использовала одинаковое кодирование меток.

Естественно, сеть MPLS может содержать набор коммутаторов ATM, работающих в качестве LSR, а другие LSR могут работать с shim-заголовками MPLS. В таких сетях могут присутствовать LSR, имеющие одновременно интерфейсы ATM и MPLS Shim. Это является одним из примеров LSR с различными методами кодирования меток на разных интервалах. Такие LSR могут принимать закодированные в ATM метки на входном интерфейсе и преобразовывать их с использованием базовой инкапсуляции на выходном интерфейсе.

3.26. Слияние меток

Предположим, что LSR связывает множество входящих меток с неким классом FEC. При пересылке пакетов этого класса маршрутизатор будет использовать одну исходящую метку для всех пакетов. То, что пакеты могли прийти с разными метками, значения не имеет - они будут пересылаться с одинаковыми исходящими метками. В этом случае говорят о слиянии меток.

Будем говорить, что LSR поддерживает слияние меток, если этот маршрутизатор может получить два пакета от разных входных интерфейсов и/или с разными входящими метками и передать оба пакета через один выходной интерфейс с одинаковым метками. После отправки пакетов информация о том, что они были приняты с разными метками, теряется.

Будем говорить, что LSR не поддерживает слияния меток, если два любых пакета, полученные от разных интерфейсов или с различными метками, должны передаваться через разные интерфейсы или с разными метками. ATM-LSR, использующие кодирование SVC или SVP, не могут выполнять слияния меток (этот вопрос более подробно рассматривается в следующем параграфе).

Если некий LSR не поддерживает слияния меток, то при получении двух пакетов одного класса FEC с разными входными метками маршрутизатор должен будет передать эти пакеты также с разными метками. При поддержке слияния меток для каждого класса FEC требуется только одна выходная метка, а если слияние меток не поддерживается, число исходящих меток для одного класса FEC может достигать числа узлов в сети.

При использовании слияния меток число входящих меток для отдельного FEC, которое требуется LSR, в любом случае должно быть не больше числа соседей по распространению меток. Без слияния меток число входящих меток для отдельного FEC, требуемое маршрутизатору LSR, совпадает с числом восходящих узлов, которые пересылают трафик данного класса FEC рассматриваемому LSR. На практике для LSR трудно даже определить число входящих меток, которые маршрутизатор должен поддерживать для определённого класса FEC.

Архитектура MPLS допускает работу маршрутизаторов как со слиянием меток, так и без слияния, Это требует организации корректного взаимодействия между LSR с поддержкой слияния меток и без таковой. Решение этого вопроса различается для сред, основанных на передаче дейтаграмм, и среды ATM. Этот вопрос требует дополнительного рассмотрения.

3.26.1. LSR, не поддерживающие слияния меток

Процедуры пересылки в MPLS очень похожи на процедуры пересылки, используемые в ATM и Frame Relay, - элемент данных поступает на узел, его метка (VPI/VCI или DLCI) отыскивается в таблице кросс-соединений, по результатам просмотра выбирается выходной порт и меняется значение метки. На практике такая же технология может использоваться для пересылки MPLS - протокол распространения меток может служить в качестве протокола сигнализации для создания таблиц кросс-соединений.

К сожалению, упомянутые технологии не всегда поддерживают слияние меток. В ATM слияние меток может приводить к перемешиванию ячеек, относящихся к разным пакетам. Если такое перемешивание происходит, процедура сборки пакета становится невозможной. Некоторые коммутаторы Frame Relay используют коммутацию ячеек на системной шине (backplane). Такие коммутаторы не поддерживают слияния меток по той же причине - ячейки, относящиеся к разным пакетам, могут перемешиваться, делая сборку пакетов невозможной.

Для решения этой проблемы предлагается два метода. Во-первых, включение в MPLS процедур, позволяющих использовать LSR, которые не поддерживают слияния меток. Во-вторых, в MPLS включаются процедуры, которые позволят некоторым коммутаторам ATM работать в качестве LSR с поддержкой слияния меток.

Поскольку MPLS поддерживает LSR со слиянием меток и без такового, MPLS включает также процедуры корректного взаимодействия между такими маршрутизаторами.

3.26.2. Метки для LSR с поддержкой и без поддержки слияния меток

Восходящий LSR, поддерживающий слияние меток, должен передавать только одну метку для каждого класса FEC. Его нисходящий сосед, который не поддерживает слияния меток, должен передавать множество меток для каждого класса FEC. Однако не существует способа заранее определить количество требуемых меток. Это число зависит от количества восходящих LSR для рассматриваемого класса FEC.

В архитектуре MPLS восходящий сосед, не поддерживающий слияния меток, не должен передавать каких-либо меток для того или иного класса FEC, пока он явно не запросил метку для данного FEC. Восходящий сосед может делать множество таких запросов и получать новую метку в ответ на каждый из них. Когда нисходящий сосед, не поддерживающий слияния меток, получает запрос от восходящего, он должен запросить метку для данного класса FEC у своего нисходящего соседа.

Возможно наличие узлов, которые поддерживают слияние меток в ограниченных масштабах (не более определённого числа меток). Предположим в качестве примера, что в силу неких аппаратных ограничений узел может сливать в одну исходящую метку не более четырёх входящих. Если такой узел получит шесть входящих меток для некого класса FEC, он будет объединять их в две исходящие метки.

Применимость слияния меток к LSP с явно заданной маршрутизацией требует дальнейшего изучения.

3.26.3. Слияние меток в коммутаторах ATM

3.26.3.1. Методы предотвращения перемешивания ячеек

Существует несколько методов решения проблемы перемешивания ячеек ATM, тем или иным способом позволяющих коммутаторам ATM выполнять слияние потоков.

1. Слияние VP с использованием кодирования SVP Multipoint.

   При использовании слияния VP множество виртуальных путей объединяются в один VP, но пакеты из разных источников в едином пути разделяются путём использования разных значений VCI.

2. Слияние VC.

   При использовании слияния VC коммутатор должен буферизовать ячейки, соответствующие одному пакету, пока пакет не будет принят целиком (это можно определить путём поиска индикатора завершения кадра AAL5).

Слияние VP обеспечивает преимущество за счёт совместимости с большинством реализаций коммутаторов ATM. Это делает использование слияния VP наиболее подходящим методом для развёртывания в существующих сетях. В отличие от слияния VC, слияние VP не вносит какой-либо задержки в точках слияния и не предъявляет дополнительных требований к буферам. Недостатком этого метода является необходимость поддержки пространства VCI для каждого VP. Реализация такой поддержки возможна множеством способов и для выбора конкретных путей требуется дополнительное исследование.

Противоречие между совместимостью с существующими сетями и сложностью/расширяемостью протоколов приводит к желательности поддержки в MPLS слияния VP и VC. Для реализации такой поддержки каждому коммутатору ATM, участвующему в MPLS, нужно знать, выполняют ли его непосредственные соседи слияние и какой метод (слияние VP или VC) они используют.

3.26.3.2. Взаимодействие - слияние VC, слияние VP, без слияния

Взаимодействие различных форм слияния меток в ATM проще описать, начав со взаимодействия для случая слияния VC и устройств без слияния меток.

В ситуации взаимодействия устройств со слиянием VC и устройств без слияния меток пересылка осуществляется на основе значений VC (т. е. конкатенации полей VPI и VCI). Для каждого узла, если восходящий сосед выполняет слияние VC, нисходящему требуется только один набор VPI/VCI для каждого потока (это идентично требованию наличия одной метки при работе в средах на основе кадров). Если восходящий сосед не поддерживает слияния меток, ему потребуется один набор VPI/VCI для самого потока и достаточное количество VPI/VCI для передачи потока восходящим соседям. Требуемое число будет определяться путём разрешения восходящим узлам запрашивать VPI/VCI от их нисходящих соседей (аналогично методу, используемому в сетях на основе кадров).

Похожий метод возможен и для узлов, выполняющих слияние VP. В этом случае узел со слиянием VP вместо запроса одного или множества VPI/VCI от своего нисходящего соседа может запросить одно значение VP (идентифицируется VPI) и множество значений VCI для этого VP. Предположим, что узел без поддержки слияния меток является нисходящим для двух узлов со слиянием VP. Этому узлу может потребоваться запрос одного набора VPI/VCI (для трафика, исходящего непосредственно с данного узла) и двух VP (по одному для каждого восходящего узла) с заданными наборами значений VCI (в соответствии с запросом восходящего узла).

Поэтому для поддержки всех вариантов (VP, VC и без слияния) требуется разрешить восходящим узлам запрашивать комбинацию (возможно пустую) идентификаторов VC (состоящих из VPI/VCI), а множество (возможно пустое) VP (идентифицируемых VPI), каждый из которых содержит заданное число VC (идентифицируемых набором VCI, значимых для данного VP). Узлы со слиянием VP будут запрашивать один VP с содержащимися в нем VCI для порождаемого данным узлом трафика (если таковой имеется), а также VCI для каждого VC, запрашиваемого в соответствии с приведённым выше описанием (независимо от того, является ли VC частью запрошенного VP). Узел со слиянием VC будет запрашивать только один набор VPI/VCI (поскольку он может объединять весь восходящий трафик в один VC). Узлы без поддержки слияния меток будут передавать все запросы, как описано выше, а также запрашивать один набор VPI/VCI для трафика, генерируемого данным узлом (при наличии такого трафика).

3.27. Туннели и иерархия

Иногда маршрутизатор Ru предпринимает явные действия по доставке некого пакета другому маршрутизатору Rd, хотя Ru и Rd не являются последовательными маршрутизаторами на поэтапном (hop-by-hop) пути для данного пакета и Rd не является конечным получателем. Это может осуществляться, например, путём инкапсуляции данного пакета в другой пакет сетевого уровня, в котором адресом получателя является адрес Rd. Такая инкапсуляция создаёт туннель от Ru к Rd. Будем называть все пакеты, которые проходят через такие туннели, туннелируемыми пакетами.

3.27.1. Туннель с поэтапной маршрутизацией

Если туннелируемый пакет следует поэтапным путём от Ru к Rd, будем называть это «туннелем с поэтапной маршрутизацией[1]», передающей стороной которого является Ru, а приёмной - Rd.

3.27.2. Явно маршрутизируемый туннель

Если туннелируемый пакет проходит от Ru к Rd по пути, отличному от поэтапного, будем называть это «туннелем с явной маршрутизацией[2]», начальной точкой которого является Ru, а конечной - Rd. Например, можно передавать пакеты через туннель с явной маршрутизацией путём инкапсуляции этих пакетов в пакеты с явной маршрутизацией (source routed).

3.27.3. Туннели LSP

Можно реализовать туннель как LSP и использовать коммутацию по меткам взамен инкапсуляции на сетевом уровне для доставки пакета через такой туннель. Туннель может быть LSP <R1, ..., Rn>, где R1 - передающая сторона туннеля, а Rn - приёмная. Такой туннель будем называть «туннелем LSP».

Множество пакетов, передаваемых через туннель LSP, представляет класс FEC и каждый маршрутизатор LSR в туннеле должен присваивать метку данному FEC (т. е. присваивать метку туннелю). Критерии отнесения конкретных пакетов к туннелю задаются локально на передающей стороне туннеля. Для отправки пакета в туннель передающая сторона помещает метку туннеля в стек меток и передаёт пакет на следующий интервал туннеля.

Если для туннеля не имеет значения возможность определения на приёмной стороне факта получения пакета через туннель (см. ниже), выталкивание меток из стека может выполняться на предпоследнем LSR туннеля.

Туннель LSP с поэтапной маршрутизацией представляет собой туннель, реализованный, как LSP с поэтапной маршрутизацией между передающей и приёмной точками туннеля.

Туннель LSP с явной маршрутизацией представляет собой туннель LSP, на основе LSP с явной маршрутизацией.

3.27.4. Иерархия - туннели LSP внутри LSP

Рассмотрим путь LSP <R1, R2, R3, R4>. Предположим, что R1 получает непомеченный пакет P и помещает в его стек метку для передачи по этому пути, который фактически является путём с поэтапной маршрутизацией. Далее предположим, что R2 и R3 не соединены напрямую, но являются соседями за счёт того, что служат конечными точками туннеля LSP. Таким образом, реальная последовательность LSR при передаче пакета P будет иметь вид <R1, R2, R21, R22, R23, R3, R4>.

При прохождении P от R1 до R2 пакет будет иметь стек меток глубиной 1. R2 при коммутации по метке определяет, что P нужно направить в туннель. Маршрутизатор R2 сначала меняет входную метку на метку, понятную R3. После этого он вталкивает в стек новую метку. Эта метка уровня 2 имеет значение, которое понятно маршрутизатору R21. В маршрутизаторах R21, R22, R23 коммутация осуществляется по меткам уровня 2. Маршрутизатор R23, который является предпоследним интервалом туннеля R2-R3, выталкивает метку из стека до пересылки пакета маршрутизатору R3. Когда R3 видит пакет P, последний имеет только метку уровня 1, показывающую, как выйти из туннеля. Поскольку R3 является предпоследним интервалом для LSP уровня 1, он выталкивает метку из стека и маршрутизатор R4 получает P без метки.

Механизм стека меток позволяет организовать туннелирование LSP с произвольным уровнем вложенности.

3.27.5. Партнерство и иерархия при распространении меток

Предположим, что пакет P проходит по LSP уровня 1 <R1, R2, R3, R4>, а участок пути от R2 до R3 является LSP уровня 2 <R2, R21, R22, R3>. С точки зрения LSP уровня 2 партнёром R2 по распространению меток является R21. С точки зрения LSP уровня 1 партнёрами R2 по распространению меток являются R1 и R3. Партнёры по распространению

[1] Hop-by-Hop Routed Tunnel.
[2] Explicitly Routed Tunnel.

меток могут присутствовать на каждом уровне иерархии. Способы использования этой иерархии будут рассматриваться в параграфах 4.6 и 4.7. Отметим, что в приведённом примере R2 и R21 должны быть IGP-соседями, а от R2 и R3 этого не требуется.

Когда два LSR являются соседями по IGP, будем называть их «локальными партнёрами по распространению меток». Если два LSR могут быть партнёрами по распространению меток, но не являются IGP-соседями, будем называть их «удалёнными партнёрами по распространению меток». В приведённом выше примере R2 и R21 являются локальными партнёрами, а R2 и R3 - удалёнными.

Архитектура MPLS поддерживает два способа распространения меток на разных уровнях иерархии - явное и неявное партнерство.

Можно передавать в сообщениях протокола распространения меток по адресу локального партнёра по распространению, а можно обмениваться метками с удаленным партнёром. Для обмена с удаленным партнёром возможны два варианта.

1. Явное партнерство (Explicit Peering).

   При явном партнёрстве маршрутизатор передаёт метки партнёру путём передачи сообщений протокола распространения меток, как это происходит при обмене с локальным партнёром. Такой вариант хорошо подходит при распространении меток небольшому числу удалённых партнёров, при большом количестве связок меток на верхнем уровне, а также в тех случаях, когда удалённые партнёры по распространению меток находятся в других областях маршрутизации или доменах. Естественно, при этом требуется знать, какие метки распространяются каждому из партнёров (см. параграф 4.1.2).

   Примеры использования явного партнёрства приведены в параграфах 4.2.1 и 4.6.

2. Неявное партнерство (Implicit Peering).

   При неявном партнёрстве маршрутизатор не передаёт сообщения протокола распространения меток, адресованные своему партнёру. Вместо этого для распространения меток верхнего уровня маршрутизатор кодирует такие метки как атрибуты меток более низкого уровня и распространяет эти метки своим локальным партнёрам. Те, в свою очередь, распространяют полученную информацию своим локальным партнёрам и процесс продолжается, пока метка не достигнет удалённого партнёра.

   Этот метод хорошо подходит для случаев, когда число удалённых партнёров велико. Неявное партнерство не требует организации n^2 партнерских связей для распространения меток n удаленным партнёрам, поскольку метки таким партнёрам распространяются опосредованно через локальных партнёров. Однако при использовании неявного партнёрства промежуточные узлы должны хранить у себя информацию, в которой они непосредственно не заинтересованы.

   Примеры использования неявного партнёрства приведены в параграфе 4.3.

3.28. Транспорт LDP

Протокол распространения меток используется между узлами сети MPLS для организации и поддержки связывания меток. Для корректной работы MPLS требуется обеспечить надёжную передачу информации о метках, а протокольные сообщения, относящиеся к определённому классу FEC, должны передаваться по порядку. Желательно также обеспечить управление потоком данных и возможность передачи множества протокольных сообщений в одной дейтаграмме.

Единственным способом достижения указанных целей является использование на транспортном уровне протокола TCP, как это предложено в [MPLS-LDP] и [MPLS-BGP].

3.29. Почему используется множество LDP?

Архитектура не задаёт жёстких правил выбора протокола распространения меток для тех или иных обстоятельств. Однако можно дать некоторые рекомендации по выбору протокола.

3.29.1. BGP и LDP

Во многих случаях желательно связать метки с классами FEC, которые могут идентифицироваться маршрутом к некому адресному префиксу (см. параграф 4.1). Если существует стандартный, широко применяемый алгоритм распространения таких маршрутов, наиболее эффективным вариантом распространения меток будет включение информации о метках в процесс распространения маршрутов.

Например, протокол BGP распространяет такие маршруты и если узлу BGP нужно также распространять метки своим партнёрам по BGP, использование протокола BGP для распространения меток (см. [MPLS-BGP]) обеспечивает множество преимуществ. В частности, это даёт возможность использовать для распространения меток рефлекторы BGP, что обеспечивает существенное повышение уровня расширяемости по сравнению с использованием LDP для распространения меток между узлами BGP.

3.29.2. Метки для RSVP Flowspec

При использовании протокола RSVP для резервирования ресурсов по отдельным потокам, может оказаться желательным помечать пакеты в таких потоках, чтобы избавиться от необходимости использования фильтров RSVP на каждом интервале. Можно показать, что распространение меток с помощью RSVP в процессе организации/резервирования пути является наиболее эффективным методом распространения меток в таких случаях.

3.29.3. Метки для явно маршрутизируемых LSP

В некоторых приложениях MPLS, связанных, в частности, с организацией трафика, желательно создание путей с явной маршрутизацией от входной точки до выходной. Для таких путей также желательно резервировать ресурсы.

Возможны два варианта решения этой задачи:

- начав с существующего протокола организации резервирования ресурсов, расширить этот протокол для поддержки явной маршрутизации и распространения меток;
- начав с существующего протокола распространения меток, расширить его для поддержки явной маршрутизации и резервирования ресурсов.

Первый вариант привёл к разработке протокола [MPLS-RSVP-TUNNELS], второй - [MPLS-CR-LDP].

3.30. Групповая адресация

Это тема для исследования в будущем.

4. Некоторые приложения MPLS

4.1. MPLS и трафик с поэтапной маршрутизацией

Многие приложения MPLS требуют, чтобы пакеты с некой меткой пересылались по тому же пути с поэтапной маршрутизацией, который будет использоваться для пересылки пакетов с заданным адресом в поле получателя заголовка сетевого уровня.

4.1.1. Метки для адресных префиксов

В общем случае маршрутизатор R определяет следующий интервал для пакета P путём поиска в своей таблице маршрутизации адресного префикса X с максимальным соответствием адресу получателя пакета. Т. е. пакеты данного класса FEC являются просто пакетами, которым соответствует данный префикс в таблице маршрутизации R. В этом случае FEC можно идентифицировать по адресному префиксу.

Отметим, что пакет P может быть отнесён к классу FEC F, а FEC может указываться префиксом X даже в тех случаях, когда адрес получателя пакета P не соответствует префиксу X.

4.1.2. Распространение меток для адресных префиксов

4.1.2.1. Партнёры по распространению меток для адресного префикса

LSR-маршрутизаторы R1 и R2 считаются партнёрами по распространению меток для префикса X тогда и только тогда, когда выполняется одно из перечисленных условий.

1. Маршрут R1 к X является маршрутом, полученным через конкретный интерфейс от конкретного экземпляра определённого протокола IGP, и R2 является соседом R1 для данного экземпляра этого IGP.
2. Маршрут R1 к X является маршрутом, полученным неким экземпляром алгоритма маршрутизации A1, этот маршрут распространяется в экземпляре алгоритма маршрутизации A2 и R2 является соседом R1 для данного экземпляра A2.
3. R1 является приёмной стороной туннеля LSP, организованного в другом LSP, R2 является передающей стороной этого туннеля, R1 и R2 участвуют в одном экземпляре IGP и находятся в одной области IGP (если IGP имеет области), а маршрут R1 к X получен от данного экземпляра IGP или распространяется маршрутизатором R1 в данный экземпляр IGP.
4. Маршрут R1 к X является маршрутом, полученным по протоколу BGP, и R2 - BGP-партнер R1.

В общем случае приведённые правила гарантируют, что при распространении маршрута к некому адресному префиксу по протоколу IGP партнёры по распространению меток для этого префикса являются соседями IGP. Если маршрут к некому адресному префиксу передаются по протоколу BGP, партнёры по распространению меток для данного префикса являются партнёрами BGP. В других случаях туннелирования LSP конечные точки туннеля являются партнёрами по распространению меток.

4.1.2.2. Распространение меток

Чтобы использовать MPLS для пересылки пакетов поэтапным маршрутом, соответствующим произвольному адресному префиксу, маршрутизатор LSR **должен**:

1. связать одну или множество меток с каждым адресным префиксом в своей таблице маршрутизации;
2. для каждого префикса X использовать протокол распространения меток для передачи информации о связывании метки с префиксом X каждому из партнёров по распределению меток для префикса X.

Существует также одно обстоятельство, когда LSR должен распространять информацию о связывании меток для адресного префикса, даже если этот LSR не связывал данную метку с этим префиксом:

3. если R1 использует BGP для распространения маршрута к X, называя некий другой LSR R2 следующим интервалом (BGP Next Hop) для X, и если R1 знает, что R2 выделил метку L для префикса X, тогда R1 должен распространять информацию о связывании L и X всем партнёрам BGP, которым он передаёт данный маршрут.

Эти правила гарантируют, что метки, соответствующие адресным префиксам, которые, в свою очередь, соответствуют маршрутам BGP, распространяются соседям IGP тогда и только тогда, когда маршруты BGP распространяются в IGP. В остальных случаях, метки, связанные с маршрутами BGP, распространяются только другим узлам BGP.

Эти правила предназначены лишь для указания информации о связывании меток, которую данный LSR должен распространять другим LSR.

4.1.3. Использование поэтапного пути в качестве LSP

Если поэтапный путь, которому должен следовать пакет P имеет вид <R1, ..., Rn>, то <R1, ..., Rn> может быть LSP при выполнении двух условий.

1. Существует такой префикс X, что для всех i, 1<=i<n, X имеет максимальное соответствие с адресом получателя P в таблице маршрутизатора Ri.
2. Для всех i, 1<i<n, Ri связывает с префиксом X метку и распространяет её маршрутизатору R[i-1].

Отметим, что LSP для пакета может простираться только до тех пор, пока он проходит через маршрутизаторы, в чьих таблицах имеется префикс с максимальным соответствием адресу получателя пакета. На последнем из таких маршрутизаторов LSP должен завершаться, а алгоритм поиска максимального соответствия выполняться снова.

Предположим в качестве примера, что пакет P с адресом получателя 10.2.153.178 должен пройти от R1 к R2 и далее к R3. Предположим также, что R2 анонсирует префикс 10.2/16 маршрутизатору R1, а R3 анонсирует префиксы 10.2.153/23, 10.2.154/23 и 10.2/16 маршрутизатору R2, т. е. R2 анонсирует маршрутизатору R1 агрегированный маршрут. В такой ситуации пакет P может помечаться как коммутируемый (Switched), пока он не достигнет маршрутизатора R2. Поскольку R2 выполняет агрегирование маршрутов, он должен выполнить алгоритм поиска лучшего соответствия, чтобы определить класс FEC для пакета P.

4.1.4. LSP Egress и LSP Proxy Egress

LSR R рассматривается, как выходной (LSP Egress) маршрутизатор для префикса X тогда и только тогда, когда выполняется любое из приведённых ниже условий.

1. R имеет адрес Y, такой, что X является в таблице R префиксом с максимальным соответствием адресу Y.
2. R содержит в таблице маршрутизации один или множество адресных префиксов Y, для которых X является начальной подстрокой Y, но в маршрутизаторе R значение «LSP previous hops[1]» для X не содержит префиксов Y (т. е., R является точкой деагрегирования для адресного префикса X).

LSR R1 рассматривается, как «LSP Proxy Egress» для адресного префикса X тогда и только тогда, когда выполняется одно из условий:

1. следующим интервалом на R1 для X будет R2, а маршрутизаторы R1 и R2 не являются партнёрами по распространению меток применительно к X (возможно по той причине, что R2 не поддерживает MPLS);
2. R1 настроен на работу в качестве LSP Proxy Egress для X.

Определение LSP позволяет в качестве LSP Egress использовать маршрутизаторы, не поддерживающие MPLS. В этом случае предпоследним узлом LSP является Proxy Egress.

4.1.5. Неявная NULL-метка

Implicit NULL представляет собой метку специального назначения, которую LSR может связать с адресным префиксом. Если LSR Ru из своего отображения ILM определяет, что помеченный пакет P необходимо переслать маршрутизатору Rd, но Rd распространяет для соответствующего адресного префикса метку Implicit NULL, то вместо замены верхней метки в стеке Ru просто выталкивает метку из стека и пересылает пакет Rd.

LSR Rd распространяет связывание метки Implicit NULL с адресным префиксом X маршрутизатору LSR Ru только при выполнении всех перечисленных ниже условий.

1. Правила параграфа 4.1.2 показывают, что Rd распространяет маршрутизатору Ru привязку метки для X.
2. Rd знает, что Ru может поддерживать метки Implicit NULL (т. е. может выталкивать метку из стека).
3. Rd является LSP Egress (не Proxy Egress) для X.

Это заставляет предпоследний LSR на пути LSP выталкивать метку из стека. Это вполне приемлемо - если LSP Egress является MPLS Egress для X, то в случае невыталкивания предпоследним LSR метки из стека маршрутизатору LSP Egress потребуется найти метку, вытолкнуть метку из стека и только после этого искать метку (или адрес L3, если в стеке не осталось меток) следующего интервала. Заставляя предпоследний LSR выталкивать метку из стека, LSP Egress экономит усилия, поскольку в противном случае ему пришлось бы просматривать две метки для принятия решения о пересылке.

Однако если предпоследний LSR является коммутатором ATM, он может не иметь возможности вытолкнуть метку из стека. Следовательно, связывание метки Implicit NULL может распространяться только в направлении LSR, которые могут поддерживать такую функцию.

Если предпоследний LSR на пути LSP для адресного префикса X представляет собой LSP Proxy Egress, он просто ведёт себя как в случае распространения маршрутизатором LSP Egress метки Implicit NULL для префикса X.

4.1.6. Опция Egress-Targeted Label Assignment

Возможны ситуации, когда LSP Ingress Ri знает, что пакеты нескольких классов FEC должны следовать по одному пути LSP, завершающемуся, скажем, на маршрутизаторе LSP Egress Re. В таких случаях корректная маршрутизация может быть обеспечена при использовании одной метки для всех таких классов FEC и не требуется выделять отдельную метку для каждого FEC. При выполнении обоих приведённых ниже условий:

1. адрес LSR Re присутствует в таблице маршрутизации, как «маршрут к хосту»;
2. Ri имеет возможность определить, что Re является выходом LSP для всех пакетов данного множества FEC

Ri может связать одну метку со всеми FEC из данного множества. Это называется «нацеленным на выход связыванием меток».

LSR Ri может определить, что LSR Re является LSP Egress для всех пакетов конкретного FEC несколькими способами.

– Если в сети используется алгоритм маршрутизации по состоянию каналов и все узлы области поддерживают MPLS, алгоритм маршрутизации предоставляет Ri достаточный объем информации для определения

[1]Предыдущие интервалы LSP.

маршрутизаторов, через которые пакеты данного класса FEC должны покинуть домен или область маршрутизации.

- Если в сети используется протокол BGP, маршрутизатор Ri имеет возможность определить, что пакеты определённого класса FEC должны покидать сеть через некий конкретный маршрутизатор, который является следующим интервалом (BGP Next Hop) для данного FEC.

- Можно использовать протокол распространения меток для передачи информации о том, какие адресные префиксы «подключены» к какому из выходных LSR. Преимущество этого метода заключается в его независимости от наличия маршрутизации по состоянию каналов.

При использовании нацеленного на выход выделения меток, число требуемых для прохождения через сеть меток может быть существенно снижено. Это может быть очень важно в случаях использования унаследованного коммутационного оборудования для работы с MPLS, а также при использовании коммутаторов с ограниченным числом поддерживаемых меток.

Одним из возможных решений будет настройка сети на использование по умолчанию нацеленного на выход выделения меток с одновременной настройкой конкретных LSR **не** использовать нацеленное на выход выделение меток для одного или множества адресных префиксов, по отношению к которым данный LSP является выходом. Предлагается использовать следующее правило.

- Если конкретный маршрутизатор LSR **не** является LSP Egress для некого набора адресных префиксов, этому маршрутизатору следует выделять метки для адресных префиксов так же, как это делает его следующий интервал в LSP для этих префиксов. Пусть Rd является следующим интервалом LSP для Ru по отношению к префиксам X1 и X2. Если Rd выделяет одну метку для X1 и X2, Ru следует поступать так же. Если же Rd выделяет для X1 и X2 разные метки, Ru тоже следует делать так.

Предположим в качестве примера, что нужно по умолчанию выделять нацеленные на выход метки, но для префиксов, имеющих множество возможных выходов LSP (т. е. для тех префиксов, которые являются многодомными), требуются другие метки. Можно настроить все LSR на выделение по умолчанию нацеленных на выход меток, а потом настроить часть LSR на выделение других меток для многодомных адресных префиксов. Для конкретного многодомного префикса X дополнительная настройка потребуется лишь в тех маршрутизаторах, которые являются LSP Egress или LSP Proxy Egress для префикса X.

Важно отметить, что в случае, когда Ru и Rd являются смежными LSR на пути LSP для префиксов X1 и X2, пересылка будет оставаться корректной, если Ru выделяет для X1 и X2 разные метки, а Rd использует одну метку для обоих префиксов. Это лишь означает, что Rd[1] будет отображать разные входные метки на одну выходную метку (обычный случай).

Аналогично, если Rd выделяет разные метки для X1 и X2, а Ru выделяет для обоих одну метку, соответствующую адресу его LSP Egress или Proxy Egress, пересылка будет выполняться корректно. Ru будет просто отображать входящую метку на выделенную Rd метку для адреса LSP Egress.

4.2. MPLS и явно маршрутизируемые LSP

Существует множество причин, по которым может оказаться желательным использование явной маршрутизации взамен поэтапной. Например, явная маршрутизация позволяет организовать все маршруты на базе административных правил, а также создавать LSP с учётом требований организации трафика [MPLS-TRFENG].

4.2.1. Явно маршрутизируемые туннели LSP

В некоторых ситуациях сетевые администраторы могут принимать решение о пересылке неких классов трафика по заданным путям, которые отличаются от путей с поэтапной маршрутизацией, обычно используемых для передачи трафика. Это можно сделать за счёт маршрутизации трафика на основе правил или за счёт организации трафика. Явный маршрут может задаваться жёстко или определяться динамически с учётом неких обстоятельств (например, маршрутизация, основанная на принуждении).

MPLS позволяет с лёгкостью создавать такие маршруты с помощью туннелей LSP с явной маршрутизацией. Для организации такого туннеля нужно:

1. задать механизм отбора пакетов, которые будет направляться в туннель LSP с явной маршрутизацией;
2. организовать туннель LSP с явной маршрутизацией;
3. принять меры против возвращения направленных в туннель пакетов с приёмного конца туннеля на передающий.

Если передающая сторона хочет поместить в туннель помеченный пакет, она должна сначала заменить верхнюю метку стека меткой, которая была получена от приёмной стороны. После этого в стек должна быть помещена метка, соответствующая самому туннелю, которая была получена от следующего интервала по туннелю. Для того, чтобы выполнить это, конечным точкам туннеля следует быть явными партнёрами по распространению меток. Информация о связывании меток, которой эти точки будут обмениваться между собой, не представляет интереса для LSR в туннеле.

4.3. Стеки меток и неявное партнерство

Предположим, что некий LSR Re является LSP Proxy Egress для 10 адресных префиксов и каждый из этих префиксов доступен через свой интерфейс.

Можно выделить одну метку для всех 10 адресных префиксов. Тогда Re будет LSP для всех 10 префиксов. Это гарантирует, что пакеты для всех 10 префиксов будут доставляться Re. Однако Re в этом случае должен будет просматривать адреса сетевого уровня для каждого пакета, чтобы выбрать корректный интерфейс для пересылки этого пакета.

[1] В оригинале ошибочно указано R1. См. http://www.rfc-editor.org/errata_search.php?rfc=3031. *Прим. перев.*

Можно для каждого интерфейса выделить свою метку. Тогда Re для 10 адресных префиксов будет играть роль LSP Proxy Egress. Это избавляет Re от необходимости просмотра адресов сетевого уровня для пересылки пакетов. Однако в результате будет использоваться значительное число меток.

Третий вариант заключается в связывании всех 10 адресных префиксов с одной меткой уровня 1 (которая также связана с адресом самого LSR) и последующем связывании каждого префикса со своей меткой уровня 2. Метки уровня 2 будут трактоваться, как атрибут привязки метки на уровне 1, который мы будем называть атрибутом стека. Мы вводим следующие правила.

- Пусть LSR Ru изначально помечает пакет без метки. Если максимальное соответствие адресу получателя даёт префикс X, следующим интервалом LSP для Ru и префикса X является маршрутизатор Rd и Rd распространяет для Ru привязку метки L1 к префиксу X с атрибутом стека L2, то:

  1. Ru должен втолкнуть сначала L2, а потом L1 в стек меток пакета и после этого переслать пакет Rd;
  2. когда Ru передаёт привязку метки к префиксу X своим партнёрам по распространению меток, он должен включать L2, как атрибут стека;
  3. всякий раз при изменении атрибута стека (возможно в результате смены на Ru следующего интервала LSP для префикса X) Ru должен распространять новый атрибут стека.

Несмотря на то, что значение метки, связанной с X, может различаться на каждом интервале LSP, значение атрибута стека устанавливается LSP Proxy Egress и передаётся неизменным.

Таким образом, LSP Proxy Egress для X становится неявным партнёром с каждым другим LSR в области маршрутизации или домене. В этом случае явное партнерство было бы слишком громоздким, поскольку вело бы к существенному росту числа партнёров.

4.4. MPLS и маршрутизация по множеству путей

Если LSR поддерживает множество маршрутов для некого потока, он может выделить для такого потока множество меток - по одной на маршрут. Таким образом, при получении второй привязки метки от конкретного соседа для некого адресного префикса следует понимать, что каждая из этих меток служит для представления адресного префикса.

Если задано множество привязок меток для конкретного префикса, эти метки могут иметь разные атрибуты.

4.5. Деревья LSP в качестве элементов «точка-многоточка»

Рассмотрим пакеты P1 и P2, каждый из которых имеет адрес получателя, для которого максимальное соответствие, проходящее через некий маршрутный домен, обеспечивает префикс X. Предположим, что поэтапный путь для P1 имеет вид <R1, R2, R3>, а для пакета P2 - <R4, R2, R3>. Далее предположим, что R3 связывает метку L3 с префиксом X и распространяет эту связку маршрутизатору R2. Маршрутизатор R2 привязывает метку L2 к префиксу X и распространяет эту привязку обоим маршрутизаторам R1 и R4. Когда R2 получает пакет P1, входящей меткой является L2. Маршрутизатор R2 будет переписывать L2 на L3 и передавать пакет P1 маршрутизатору R3. Когда R2 получает пакет P2, входящей меткой также будет L2. Маршрутизатор R2 снова заменит L2 на L3 и отправит пакет P2 маршрутизатору R3.

Отметим, что при прохождении пакетов P1 и P2 от R2 к R3, они содержат одинаковые метки и с точки зрения MPLS являются неразличимыми. Таким образом, вместо того, чтобы говорить о двух разных LSP - <R1, R2, R3> и <R4, R2, R3>, мы будем говорить об одном дереве LSP (Multipoint-to-Point LSP Tree), которое обозначим <{R1, R4}, R2, R3>.

Это создаёт сложности при использовании традиционных коммутаторов ATM в качестве LSR. Поскольку такие коммутаторы не поддерживают многоточечных соединений, требуются процедуры, обеспечивающие реализацию каждого LSP, как VC «точка-точка». Однако при использовании коммутаторов ATM, поддерживающих VC «точка-многоточка», LSP более эффективно реализуются как VC «точка-многоточка». Кроме того, при возможности использования SVP Multipoint Encoding (см. параграф 3.25.2) LSP можно реализовать как SVP «точка-многоточка».

4.6. Туннели LSP между граничными маршрутизаторами BGP

Рассмотрим автономную систему A, которая передаёт транзитный трафик между другими автономными системами. AS[1] A будет иметь множество граничных маршрутизаторов BGP, соединённых между собой по протоколу BGP для распространения маршрутов BGP. Во многих таких случаях желательно избежать распространения маршрутов BGP тем устройствам, которые не относятся к числу граничных маршрутизаторов BGP. Если такое распространение удаётся предотвратить, нагрузка по распространению маршрутов на этих маршрутизаторах существенно снизится. Однако требуется принять ряд мер по обеспечению гарантий доставки транзитного трафика между граничными маршрутизаторами через внутренние маршрутизаторы.

Этого легко добиться с помощью туннелей LSP. Предположим, что маршруты BGP распространяются только граничным маршрутизаторам BGP, а не внутренним маршрутизаторам, расположенным на поэтапном пути между парами граничных маршрутизаторов. Туннели LSP в этом случае можно использовать, как описано ниже.

1. Каждый граничный маршрутизатор BGP распространяет всем остальным граничным маршрутизаторам BGP в своей AS метку для каждого адресного префикса, который он распространяет этому маршрутизатору по BGP.
2. IGP[2] для AS поддерживают хост-маршруты к каждому граничному маршрутизатору BGP. Каждый внутренний маршрутизатор распространяет свои метки для этих хост-маршрутов каждому из своих соседей по IGP.
3. Предположим, что:

   a) граничный маршрутизатор BGP B1 получает непомеченный пакет P;

[1] Autonomous System.
[2] Протокол внутренней маршрутизации.

b) адресный префикс X в таблице маршрутизации B1 обеспечивает максимальное соответствие адресу получателя P;

c) маршрут к X является маршрутом BGP;

d) следующим интервалом BGP для префикса X является B2;

e) B2 связывает метку L1 с префиксом X и распространяет информацию об этом маршрутизатору B1;

f) следующим интервалом IGP для адреса маршрутизатора B2 является маршрутизатор I1;

g) адрес B2 имеется в таблицах внутренней маршрутизации B1 и I1, как маршрут к хосту;

h) I1 связывает метку L2 с адресом B2 и распространяет информацию об этом маршрутизатору B1.

Тогда перед отправкой пакета P маршрутизатору I1 граничный маршрутизатор B1 должен создать стек меток для P, втолкнуть в него метку L1, а затем - метку L2.

4. Предположим, что граничный маршрутизатор BGP B1 получает помеченный пакет P с верхней меткой стека, соответствующей адресному префиксу X, к которому ведёт маршрут BGP, и условия 3b, 3c, 3d и 3e выполнены. Тогда перед отправкой пакета P маршрутизатору I1 граничный маршрутизатор B1 должен заменить верхнюю метку стека на L1, а потом поместить в стек метку L2.

С помощью этих процедур пакет P будет следовать по LSP уровня 1, все члены которого будут граничными маршрутизаторами BGP, а между каждой парой граничных маршрутизаторов BGP на LSP уровня 1 пакет будет следовать по LSP уровня 2.

Эти процедуры по сути дела создают поэтапно маршрутизируемый туннель LSP между граничными устройствами BGP.

Поскольку граничные маршрутизаторы BGP обмениваются привязками меток для адресных префиксов, которые даже не известны протоколу внутренней маршрутизации, узлам BGP следует быть явными партнёрами по распространению меток друг для друга.

Иногда можно создать поэтапно маршрутизируемые туннели LSP между граничными маршрутизаторами BGP, относящимися к разным AS. Предположим, что B1 и B2 находятся в AS1, а B3 является EBGP-соседом B2, расположенным в AS2. Также предположим, что B2 и B3 относятся к некой сети, которая является общей для обеих AS (демилитаризованная зона). В этом случае туннель LSP может быть организован напрямую между B1 и B3 как показано ниже.

- B3 распространяет маршруты B2 (используя EBGP), возможно связывая метки с адресными префиксами.
- B2 распространяет далее эти маршруты B1 (с помощью IBGP), показывая, что следующим интервалом BGP для каждого из таких маршрутов является B3. Если B3 связал с адресными префиксами метки, B2 будет передавать этим метки B1 без изменения.
- IGP автономной системы AS1 имеет хост-маршрут для B3.

4.7. Другие применения туннелей LSP с поэтапной маршрутизацией

Использование туннелей LSP с поэтапной маршрутизацией не ограничено организацией туннелей между BGP Next Hop. В любой ситуации, где применяется инкапсуляция, можно использовать туннель LSP с поэтапной маршрутизацией. Вместо инкапсуляции пакета с созданием нового заголовка, в котором получателем будет приёмная. сторона туннеля, можно поместить в стек метку для адресного префикса с максимальным соответствием адресу приёмной стороны. Пакет, передаваемый в туннель, уже может включать метку или метка может создаваться специально для туннеля.

Если передающая сторона туннеля желает отправить помеченный пакет в туннель, она должна сначала заменить значение верхней метки в стеке на значение метки, распространяемой приёмной стороной туннеля. После этого в стек должна помещаться метка, соответствующая самому туннелю, которая получена от следующего интервала в туннеле. Для выполнения этой операции конечным точкам туннеля следует быть явными партнёрами по распространению меток. Привязки меток, которыми будут обмениваться такие партнёры, не представляют интереса для LSR, образующих туннель.

4.8. MPLS и групповая адресация

Групповая адресация реализуется путём построения multicast-деревьев. Дерево, вдоль которого должен передаваться конкретный пакет с групповым адресом, зависит в общем случае от адресов отправителя и получателя в пакете. Когда тот или иной LSR является узлом некого multicast-дерева, он связывает с этим деревом метку. Информация о такой привязке распространяется родительскому узлу multicast-дерева (если узел подключён к ЛВС и в этой ЛВС есть другие узлы такого типа, информация о привязке распространяется также этим узлам, что позволяет родителю обойтись одной меткой при пересылке пакетов с групповой адресацией всем дочерним объектам в ЛВС).

При получении группового пакета с меткой значение NHLFE, соответствующее метке, показывает набор выходных интерфейсов для этого пакета и задаёт исходящую метку. Если на всех выходных интерфейсах используется одинаковое представление меток, всем потомкам передаются точные копии одного пакета.

5. Процедуры поэтапного распространения меток

В этом разделе рассматриваются только привязки меток, которые используются для трафика, передаваемого на основе коммутации по меткам через путь с поэтапной маршрутизацией. В таких случаях метки соответствуют адресным префиксам в маршрутных таблицах.

5.1. Процедуры анонсирования и использования меток

Существует множество процедур, которые могут быть использованы для распространения информации о привязках меток. Некоторые из этих процедур выполняются нисходящими LSR, другие - восходящими.

Нисходящий LSR должен выполнять процедуры:

- Distribution (распространение);
- Withdrawal (отзыв).

Восходящий LSR должен выполнять процедуры:

- Request (запрос);
- NotAvailable (недоступно);
- Release (освобождение);
- labelUse (использование метки).

Архитектура MPLS поддерживает несколько вариантов каждой процедуры. Однако архитектура не поддерживает всех возможных комбинаций вариантов. Набор поддерживаемых комбинаций будет описан в параграфе 5.2 при обсуждении вопросов взаимодействия.

5.1.1. Нисходящий LSR - процедура Distribution

Процедура Distribution используется нисходящим LSR для определения, когда ему следует сообщать привязку метки к конкретному адресному префиксу своим партнёрам по распространению меток. Архитектура поддерживает 4 различных варианта процедуры распространения.

Независимо от используемой процедуры, если привязка метки к некому адресному префиксу распространяется нисходящим LSR Rd восходящему LSR Ru и в какой-то момент атрибуты этой привязки (см. определение выше) изменяются, Rd должен информировать Ru о новых атрибутах.

Если LSR поддерживает множество маршрутов к некому адресному префиксу, вопрос привязки множества меток к одному префиксу (по одной на маршрут) и, следовательно, распространение множества привязок, определяется локальными установками данного LSR.

5.1.1.1. PushUnconditional - безусловное выталкивание

Предположим, что Rd является LSR и выполняются условия:

1. X является адресным префиксом в таблице маршрутизации Rd;
2. Ru является партнёром Rd по распространению меток применительно к префиксу X.

Всякий раз, когда эти условия выполняются, Rd должен связать метку с X и распространить информацию об этом своему партнёру Ru. Маршрутизатор Rd несёт ответственность за отслеживание привязок, которые он распространяет Ru и должен всегда быть уверен, что Ru имеет информацию об этих привязках.

Эта процедура будет использоваться LSR, которые выполняют незапрошенное выделение меток в режиме независимого управления LSP.

5.1.1.2. PushConditional - условное выталкивание

Предположим, что Rd является LSR и выполняются условия:

1. X является адресным префиксом в таблице маршрутизации Rd;
2. Ru является партнёром Rd по распространению меток применительно к префиксу X;
3. Rd является LSP Egress или LSP Proxy Egress для префикса X или следующим интервалом сетевого уровня для этого префикса в Rd является маршрутизатор Rn, отличный от Ru, связавший метку с префиксом X и распространяющий информацию об этом маршрутизатору Rd.

При выполнении всех условий Rd следует связать метку с X и распространить информацию об этом Ru.

Процедура PushUnconditional вызывает распространение привязок меток для всех адресных префиксов в таблице маршрутизации, а PushConditional вызывает распространение только для тех привязок, информация о которых была получена от следующего интервала LSP, или тех, которые не имеют следующего интервала L3 с поддержкой MPLS.

Эта процедура будет использоваться LSR, которые выполняют незапрошенное нисходящее выделение меток в упорядоченном режиме.

5.1.1.3. PulledUnconditional - безусловное втягивание

Предположим, что Rd является LSR и выполняются условия:

1. X является адресным префиксом в таблице маршрутизации Rd;
2. Ru является партнёром Rd по распространению меток применительно к префиксу X;
3. Ru явно запрашивает у Rd привязку метки к префиксу X и распространение информации об этом Ru.

В этом случае Rd следует связать метку с X и распространить информацию об этом маршрутизатору Ru. Отметим, что в случаях, когда X отсутствует в таблице маршрутизации Rd или Rd не является партнёром Ru по распространению меток для префикса X, то Rd должен информировать Ru о невозможности организовать привязку в данный момент.

Если Rd уже распространил информацию о метке для префикса X маршрутизатору Ru и получает новый запрос от Ru для привязки метки к префиксу X, он будет привязывать вторую метку и распространять информацию об этом Ru. Первая метка остаётся действующей.

Эта процедура используется LSR, выполняющими нисходящее распространение меток по запросам в независимом режиме.

5.1.1.4. PulledConditional - условное втягивание

Предположим, что Rd является LSR и выполняются условия:

1. X является адресным префиксом в таблице маршрутизации Rd;
2. Ru является партнёром Rd по распространению меток применительно к префиксу X;
3. Ru явно запрашивает у Rd привязку метки к префиксу X и распространение информации об этом Ru;
4. Rd является LSP Egress или LSP Proxy Egress для префикса X или следующим интервалом сетевого уровня для этого префикса в Rd является маршрутизатор Rn, отличный от Ru, связавший метку с префиксом X и распространяющий информацию об этом маршрутизатору Rd.

При выполнении всех перечисленных условий Rd следует связать метку с префиксом X и распространить информацию об этом маршрутизатору Ru. Отметим, что для случаев, когда X отсутствует в таблице маршрутизации Rd и привязку для X невозможно получить через следующий интервал Rd для префикса X или Rd не является партнёром Ru по распространению меток для префикса X, маршрутизатор Rd должен информировать Ru о невозможности организовать привязку в данный момент.

Если единственным не выполненным условием является то, что Rn ещё не предоставил метку Rd, маршрутизатор Rd должен задержать ответ Ru до получения нужной информации от Rn.

Если Rd распространяет метку для префикса X маршрутизатору Ru и после этого в какой-то момент меняется любой из атрибутов этой метки, Rd должен заново распространить информацию о привязке метки с новым атрибутом Ru. Это должно происходить даже в тех случаях, когда Ru не делает нового запроса.

Эта процедура используется LSR, выполняющими нисходящее выделение меток по запросу в упорядоченном режиме.

В параграфе 5.2 обсуждается вопрос выбора конкретной процедуры для использования в каждый момент и методы обеспечения взаимодействия LSR, выбравших разные процедуры.

5.1.2. Восходящий LSR - процедура Request

Процедура запроса используется восходящим LSR для адресного префикса, чтобы определить, когда явно запрашивать у нисходящего LSR привязку метки к данному префиксу и распространение информации об этом. Существует три возможных варианта процедуры.

5.1.2.1. RequestNever - никогда не запрашивать

Запросы не выполняются никогда. Это полезно в тех случаях, когда нисходящий LSR использует процедуру PushConditional или PushUnconditional, но бесполезно при использовании нисходящим LSR процедуры PulledUnconditional или PulledConditional.

Эта процедура применяется LSR при использовании режима незапрошенного нисходящего распространения с либеральным удержанием меток.

5.1.2.2. RequestWhenNeeded - запрашивать при необходимости

Запрос делается при изменении следующего интервала L3 для адресного префикса или получении нового префикса, для которого ещё нет метки, полученной от следующего интервала для данного адресного префикса.

Эта процедура применяется LSR в случаях использования консервативного режима удержания меток.

5.1.2.3. RequestOnRequest - запрашивать по запросу

Запрос выдаётся в ответ на получение запроса в дополнение к запросам при возникновении необходимости (как описано в параграфе 5.1.2.2). Если Ru не может быть входом LSP, он может выдавать запрос только при получении запроса от восходящего узла.

Если Rd получает такой запрос от Ru для адресного префикса, информация о привязке для которого уже была распространена Rd маршрутизатору Ru, Rd следует выделить новую (другую) метку, привязать её к X и распространить информацию о привязке (вопрос незамедлительного распространения маршрутизатором Rd этой привязки партнёру Ru решается в зависимости от используемой процедуры Distribution).

Эта процедура используется LSR которые выполняют нисходящее распространение по запросу, но не поддерживают слияния меток (например, ATM-LSR, не поддерживающие слияния VC).

5.1.3. Восходящий LSR - процедура NotAvailable

Если Ru и Rd являются восходящим и нисходящим партнёрами по распространению меток для префикса X, соответственно, Rd является следующим интервалом L3 маршрутизатора Ru для префикса X и Ru запрашивает привязку метки для X у Rd, но Rd отвечает о невозможности привязки в данный момент, поскольку у него нет следующего интервала для X, процедура NotAvailable определяет действия Ru. Существует два возможных варианта поведения Ru.

5.1.3.1. RequestRetry - повторять запрос

Ru следует повторить запрос позднее, т. е. запрашивающая сторона отвечает за повторение попытки получения требуемой привязки. Эта процедура применяется при использовании нисходящего распространения меток по запросу.

5.1.3.2. RequestNoRetry - не повторять запрос

Ru никогда не следует повторять запрос, предполагая взамен этого, что Rd автоматически предоставит нужную информацию, когда она станет доступной. Это полезно в тех случаях, когда Rd использует процедуру PushUnconditional или PushConditional, т. е. при нисходящем распространении меток без запроса.

Отметим, что для случаев, когда Rd отвечает, что не может предоставить информацию о привязке маршрутизатору Ru по причине возникновения ошибок, а не в результате отсутствия у Rd следующего интервала, поведение Ru будет определяться восстановлением ошибок протокола распространения меток, а не процедурой NotAvailable.

5.1.4. Восходящий LSR - процедура Release

Предположим, что Rd является LSR, связавшим метку с префиксом X и распространяющим эту привязку LSR Ru. Если Rd не является следующим интервалом L3 маршрутизатора Ru для префикса X или перестал быть таковым, Ru не будет использовать эту метку. Процедура Release задаёт действия Ru в таких ситуациях. Возможны два варианта поведения Ru.

5.1.4.1. ReleaseOnChange - освобождать при изменении

Ru следует освободить метку и информировать об этом Rd. Эта процедура используется для реализации консервативного режима удержания меток.

5.1.4.2. NoReleaseOnChange - не освобождать при изменении

Ru следует поддерживать привязку, что позволит незамедлительно начать использование метки, если Rd позднее станет следующим интервалом L3 для префикса X. Эта процедура используется для реализации либерального режима удержания меток.

5.1.5. Восходящий LSR - процедура labelUse

Предположим, что Ru - LSR, который получает привязку метки L для префикса X от LSR Rd и Ru является восходящим по отношению к Rd для префикса X, а фактически Rd является следующим интервалом L3 маршрутизатора Ru для префикса X.

Ru будет использовать привязку метки, если Rd является следующим интервалом L3 маршрутизатора Ru для префикса X. Если на момент получения привязки Ru маршрутизатор Rd **не** является следующим интервалом L3 маршрутизатора Ru для X, Ru не использует полученную привязку. Однако Ru может начать использование привязки позднее, когда Rd станет следующим интервалом L3 маршрутизатора Ru для префикса X.

Процедура labelUse определяет использование маршрутизатором Ru привязки, полученной от Rd.

Существует два варианта поведения Ru.

5.1.5.1. UseImmediate - использовать сразу

Ru может принять привязку для незамедлительного применения. В любой момент, когда у Ru есть привязка для префикса X от Rd и Rd является следующим интервалом L3 маршрутизатора Ru для X, Rd будет для этого префикса также следующим интервалом LSP по отношению к Ru. Эта процедура будет применяться в тех случаях, когда не используется детектирование петель.

5.1.5.2. UseIfLoopNotDetected - использовать при отсутствии петель

Эта процедура совпадает с UseImmediate, пока Ru не видит петли в LSP. При обнаружении петли Ru прекратит использование метки L для пересылки пакетов маршрутизатору Rd.

Эта процедура применяется при использовании детектирования петель.

Использование будет продолжаться до изменения следующего интервала для X или прекращения петли.

5.1.6. Нисходящий LSR - процедура Withdraw

Для этого случая имеется единственная процедура.

Когда LSR Rd решает разорвать связь между меткой L и адресным префиксом X, информация об этом должна быть распространена всем LSR, которым отправлена информация о привязке.

Требуется, чтобы информация о разрыве привязки L к префиксу X распространялась Rd маршрутизатору Ru до того, как Rd распространит Ru новую привязку метки L к любому другому префиксу Y (X != Y). Если Ru получит новую привязку L к Y до того, как узнает о разрыве связи между L и X и от Ru к Rd будут в течение некоторого времени идти пакеты, соответствующие префиксам X и Y, маршрутизатор Ru будет помечать этой меткой как пакеты, соответствующие X, так и пакеты, соответствующие Y.

Распространение и отзыв привязок меток осуществляется с помощью протокола распространения меток. Все протоколы распространения меток требуют, чтобы между парой партнёров по распространению поддерживались отношения смежности (исключением являются неявные партнёры). Если LSR R1 имеет смежность по распространению меток с LSR R2 и получает привязку метки от LSR R2 через эту смежность, тогда при разрыве отношений смежности (в результате отказа или при нормальной работе) все полученные через эту смежность привязки должны рассматриваться, как отозванные.

Пока сохраняются соответствующие отношения смежности, разрываемые привязки меток всегда должны отзываться явно. Если с адресным префиксом связывается вторая метка, это не является неявным отзывом первой метки, поскольку обе метки могут использоваться одновременно - это нужно для поддержки маршрутизации по множеству путей. Если метка связывается со вторым адресным префиксом, это не является неявным отзывом, поскольку метка может использоваться для обоих префиксов одновременно.

5.2. Схемы MPLS - поддерживаемые комбинации процедур

Рассмотрим два LSR - Ru и Rd, которые являются партнёрами по распространению меток применительно к некому набору адресных префиксов. Ru является восходящим партнёром, Rd - нисходящим.

Схема MPLS, управляющая взаимодействием Ru и Rd, может описана в форме квинтета процедур - <Distribution, Request, NotAvailable, Release, labelUse> (поскольку процедура Withdraw имеет единственный вариант, она не включена в квинтет). Символ * в квинтете является заменителем, вместо которого может быть подставлена любая процедура соответствующей категории, обозначение N/A говорит о том, что процедура соответствующей категории не требуется.

Архитектура MPLS поддерживает только схемы, перечисленные ниже. В будущем, при необходимости, набор поддерживаемых схем может быть расширен.

5.2.1. Схемы для LSR с поддержкой слияния меток

Если Ru и Rd являются партнёрами по распространению меток и оба поддерживают слияние меток, должна использоваться одна из приведённых здесь схем.

1. **<PushUnconditional, RequestNever, N/A, NoReleaseOnChange, UseImmediate>**

   Это нисходящее распространение незапрошенных меток с независимым управлением в либеральном режиме удержания без детектирования петель.

2. **<PushUnconditional, RequestNever, N/A, NoReleaseOnChange, UseIfLoopNotDetected>**

   Это нисходящее распространение незапрошенных меток с независимым управлением в либеральном режиме удержания с детектированием петель.

3. **<PushConditional, RequestWhenNeeded, RequestNoRetry, ReleaseOnChange, *>**

   Это нисходящее распространение незапрошенных меток с упорядоченным (со стороны выхода) управлением в консервативном режиме удержания. Детектирование петель является необязательным.

4. **<PushConditional, RequestNever, N/A, NoReleaseOnChange, *>**

   Это нисходящее распространение незапрошенных меток с упорядоченным (со стороны выхода) управлением в либеральном режиме удержания. Детектирование петель является необязательным.

5. **<PulledConditional, RequestWhenNeeded, RequestRetry, ReleaseOnChange, *>**

   Это нисходящее распространение меток по запросам с упорядоченным (инициируется со стороны входа) управлением в консервативном режиме удержания. Детектирование петель является необязательным.

6. **<PulledUnconditional, RequestWhenNeeded, N/A, ReleaseOnChange, UseImmediate>**

   Это нисходящее распространение меток по запросам с независимым управлением в консервативном режиме удержания без детектирования петель.

7. **<PulledUnconditional, RequestWhenNeeded, N/A, ReleaseOnChange, UseIfLoopNotDetected>**

   Это нисходящее распространение меток по запросам с независимым управлением в консервативном режиме удержания с детектированием петель.

5.2.2. Схемы для LSR, не поддерживающих слияния меток

Предположим, что R1, R2, R3 и R4 - коммутаторы ATM, которые не поддерживают слияния меток, но используются в качестве LSR. Предположим также, что поэтапный путь L3 для адресного префикса X имеет вид <R1, R2, R3, R4> и пакеты, адресованные X, могут войти в сеть только через эти LSR. Поскольку здесь нет поддержки режимов «точка-многоточка», LSP должны быть реализованы, как VC «точка-точка», что требует наличия трёх таких VC для префикса X - <R1, R2, R3, R4>, <R2, R3, R4> и <R3, R4>.

Следовательно, если R1 и R2 являются партнёрами MPLS и оба являются LSR, реализованными на стандартном коммутационном оборудовании ATM (т. е., без подавления перемешивания ячеек) или не способны поддерживать слияние меток по иной причине, схема MPLS, используемая при взаимодействии между R1 и R2, должна быть одной из перечисленных ниже.

1. **<PulledConditional, RequestOnRequest, RequestRetry, ReleaseOnChange, *>**

   Это нисходящее распространение меток по запросам с упорядоченным управлением (инициированным со стороны входа) в консервативном режиме удержания с необязательным детектированием петель.

   Использование процедуры RequestOnRequest будет заставлять R4 распространять три метки для префикса X маршрутизатору R3, который будет распространять 2 метки для X маршрутизатору R2, а R2 будет распространять одну метку для X маршрутизатору R1.

2. **<PulledUnconditional, RequestOnRequest, N/A, ReleaseOnChange, UseImmediate>**

   Это нисходящее распространение меток по запросам с независимым управлением в консервативном режиме удержания без детектирования петель.

3. **<PulledUnconditional, RequestOnRequest, N/A, ReleaseOnChange, UseIfLoopNotDetected>**

   Это нисходящее распространение меток по запросам с независимым управлением в консервативном режиме удержания с детектированием петель.

5.2.3. Вопросы взаимодействия

Легко видеть, что некоторые из квинтетов **не** дают жизнеспособных схем MPLS. Например,

– **<PulledUnconditional, RequestNever, *, *, *>**

  <PulledConditional, RequestNever, *, *, *>

  В этих вариантах нисходящий LSR Rd распространяет привязки меток восходящему LSR Ru только по запросам последнего, но Ru никогда не делает таких запросов. Очевидно, что такая схема нежизнеспособна, поскольку она не обеспечивает подобающего распространения привязок меток.

– **<*, RequestNever, *, *, ReleaseOnChange>**

  В этом варианте Rd освобождает привязки, когда он перестаёт их использовать, но никогда не запрашивает их заново. Такая схема не обеспечивает корректного распространения меток.

Приведённые в этом разделе правила предназначены для предотвращения создания партнерских пар с наборами процедур, которые ведут к нежизнеспособным схемам MPLS. Эти правила требуют обмена информацией между партнёрами по распространению меток на этапе организации смежности по распространению или получения такой информации заранее иными путями (обсуждение этих путей выходит за рамки настоящей спецификации).

1. Каждая сторона должна указать, поддерживает ли она слияние меток.
2. Если Rd не поддерживает слияния меток, он должен выбрать процедуру PulledUnconditional или PulledConditional. При выборе Rd процедуры PulledConditional, Ru будет вынужден использовать процедуру RequestRetry.

   В случаях, когда нисходящий LSR не поддерживает слияния меток, его предпочтения имеют более высокий приоритет при выборе схемы MPLS.
3. Если Ru не поддерживает слияния меток, а Rd поддерживает, маршрутизатор Ru должен выбрать процедуру RequestRetry или RequestNoRetry. Это заставляет Rd использовать процедуру PulledConditional или PulledUnConditional, соответственно.

   Т. е. если только один из LSR не поддерживает слияния меток, его предпочтения имеют более высокий приоритет при выборе схемы MPLS.
4. Если Ru и Rd поддерживают слияние меток, выбор между либеральным и консервативным режимом удержания меток остаётся за маршрутизатором Ru, т. е. Ru выбирает использование процедуры RequestWhenNeeded/ReleaseOnChange (консервативный режим) или RequestNever/NoReleaseOnChange (либеральный режим). Однако выбор между push и pull, а также между conditional и unconditional остаётся за Rd. Если Ru выбирает либеральный режим удержания меток, Rd может выбрать процедуру PushUnconditional или PushConditional. Если Ru выбирает консервативный режим удержания меток, Rd может выбрать процедуру PushConditional, PulledConditional или PulledUnconditional.

   Приведённые варианты выбора совместно определяют используемую схему MPLS.

6. Вопросы безопасности

Некоторые маршрутизаторы могут поддерживать процедуры защиты, зависящие от положения заголовка сетевого уровня относительно заголовка канального уровня. Базовая инкапсуляция MPLS помещает дополнительную вставку (shim) между заголовками канального и сетевого уровня, которая может вызывать проблемы при работе упомянутых процедур.

Смысл метки MPLS определяется соглашением между LSR, помещающим метку в стек (label writer), и LSR, который интерпретирует эту метку (label reader). Если помеченный пакет получен из источника, к которому нет доверия, или некая входящая метка принята от LSR, которому метки не распространяются, использование такой метки может приводить к утрате легитимности маршрутизации.

7. Интеллектуальная собственность

IETF уведомлен о правах интеллектуальной собственности, связанных со всеми или некоторыми спецификациями, содержащимися в документе. Дополнительную информацию можно получить в online-списке заявленных прав.

8. Адреса авторов

Eric C. Rosen

Cisco Systems, Inc.

250 Apollo Drive

Chelmsford, MA, 01824

EMail: erosen@cisco.com

Arun Viswanathan

Force10 Networks, Inc.

1440 McCarthy Blvd.

Milpitas, CA 95035-7438

EMail: arun@force10networks.com

Ross Callon

Juniper Networks, Inc.

1194 North Mathilda Avenue

Sunnyvale, CA 94089 USA

EMail: rcallon@juniper.net

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

9. Литература

[MPLS-ATM] Davie, B., Lawrence, J., McCloghrie, K., Rekhter, Y., Rosen, E., Swallow, G. and P. Doolan, "MPLS using LDP and ATM VC Switching", RFC 3035, January 2001.

[MPLS-BGP] "Carrying Label Information in BGP-4", Rekhter, Rosen, Work in Progress[1].

[MPLS-CR-LDP] "Constraint-Based LSP Setup using LDP", Jamoussi, Editor, Work in Progress[2].

[MPLS-FRMRLY] Conta, A., Doolan, P. and A. Malis, "Use of Label Switching on Frame Relay Networks Specification", RFC 3034, January 2001.

[MPLS-LDP] Andersson, L., Doolan, P., Feldman, N., Fredette, A. and B. Thomas, "LDP Specification", RFC 3036, January 2001.

[MPLS-RSVP-TUNNELS] "Extensions to RSVP for LSP Tunnels", Awduche, Berger, Gan, Li, Swallow, Srinvasan, Work in Progress[3].

[MPLS-SHIM] Rosen, E., Rekhter, Y., Tappan, D., Fedorkow, G., Farinacci, D. and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[MPLS-TRFENG] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M. and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

10. Полное заявление авторских прав

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.

[1] Работа опубликована в RFC 3107. *Прим. перев.*
[2] Работа опубликована в RFC 3212. *Прим. перев.*
[3] Работа опубликована в RFC 3209. *Прим. перев.*