Содержание

Введение

Эхо-запрос LSP MPLS является основным программным средством, используемым для проверки состояния пути коммутации меток (LSP) между входом и выходом. Этот документ стремится объяснять взаимодействие многопутевой информации между инициатором и респондентом в трассировке дерева LSP. Для подробных опций, доступных для этого программного средства, было бы полезно обратиться [этот документ.](#)

Общие сведения

Эта реализация EM MPLS? LSP MPLS Многопутевая Древовидная функция Трассировки основывается на RFC 4379, *Обнаруживая Сбои Плоскости Данных Коммутируемой многопротокольной метки (MPLS)*.

Путем установки IP - адреса назначения (DA) тестового пакета как адрес обратной связи (127. x . x . x), трассировка дерева LSP может использоваться для обнаружения сбоя в LSP путем предотвращения, чтобы пакет маршрутизировал IP. Таким образом каждый раз, когда существует проблема сквозного соединения, полезно использовать Эхо-запрос LSP в качестве первого шага для устранения любого Сбоя LSP.

В случае сценариев с несколькими маршрутами эхо-запрос LSP может не всегда помогать определять все Сбои LSP. Как на это можно было бы обратить внимание, любой маршрутизатор с коммутацией меток (LSR) при получении помеченного пакета, который может быть отослан множественные исходящие интерфейсы, определенные ключи использования от пакета и ввода к алгоритму хеширования для решения исходящего интерфейса. В зависимости от поставщика, аппаратных средств, и т.д., любой из ниже опций можно рассмотреть для хеширования:

1. Один только стек входящей метки.
2. Стек входящей метки и подробные данные IP - заголовка (Если информационное наполнение является IP).
3. Стек входящей метки, IP - заголовок и транспортные подробные данные заголовка.

Обычно, если стек имеет размер, меньше чем или равный 3 (с IP как информационное наполнение), маршрутизаторы Cisco рассматривают комбинацию стека меток и IP - заголовка.

Примите следующую топологию.

R1-R7 являются маршрутизаторами. В вышеупомянутой топологии существует 3 маршрута равноценного много пути (ECMP) от R1 до R5 как ниже,

Путь 1: R1-R2-R3-R4-R5

PATH2: R1-R2-R6-R4-R5

PATH3: R1-R2-R6-R7-R5

Предположите, что существует проблема между R6 и R7 (как сломанный протокол распределения меток (LDP) или метка misprogramming, и т.д.) то, чтобы заставлять трафик от R1 до R5 через PATH3 понизиться. Если Эхо-запрос LSP от R1 берет PATH1 или PATH2, можно закончить тем, что предположили, что путь между R1 и R5 прекрасен.

Эхо-запрос LSP позволяет устанавливать IP - адрес назначения (DA) как любого из диапазона 127.0.0.0/8. В то время как одна простая опция должна вручную попытаться передать множественные ping - пакеты с другим адресом назначения (DA), нет никакой гарантии, что будут проверены все возможные пути ECMP. Вам нужен путь, который делает запрос и проверяет все возможные пути между источником и назначением. LSP Многопутевая древовидная трассировка усиливает? Многопутевое информационное Кодирование? определенный в Разделе 3.3.1 из RFC4379 и помогает вам проверять все пути ECMP.

Трассировка Дерева LSP - Как это работает

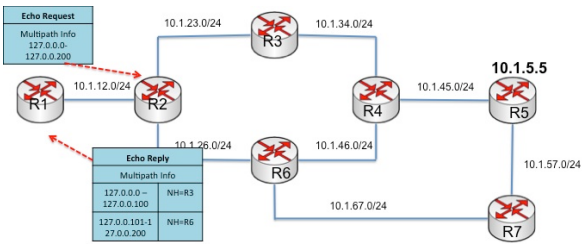
Обычный эхо-запрос MPLS или traceroute могут указать, что нет никакого сбоя в зависимости от того, как load-share транзитных маршрутизаторов пакеты по ECMP, однако трассировка дерева LSP предоставляет лучший метод, чтобы проверить, что фактически работают все пути.

В трассировке дерева LSP маршрутизатор инициатора передает запрос эха MPLS к каждому переходу путем установки TTL в главной метке инкрементным способом (запускающийся от 1). Запрос эха будет нести Многопутевой информационный TLV, который несет диапазон IP-адреса (в диапазоне 127.0.0.0/8) или энтропийном диапазоне метки. В настоящее время устройства Cisco поддерживают опцию IP - адреса назначения и таким образом, наш пример будет детализирован с Диапазоном IP-адресов.

Каждый транзитный LSR при получении пакета запроса ответит со всеми исходящими интерфейсами ECMP и привяжет диапазон IP-адреса (или энтропийная метка) от запроса о каждом интерфейсе.

Трассировка Дерева LSP - Подробный пример

Примите следующую топологию, например, ниже.



Для простоты данный пример использует диапазон адресов 127.0.0.0-127.0.0.200. Вот

подробные данные шагов в трассировку дерева LSP.

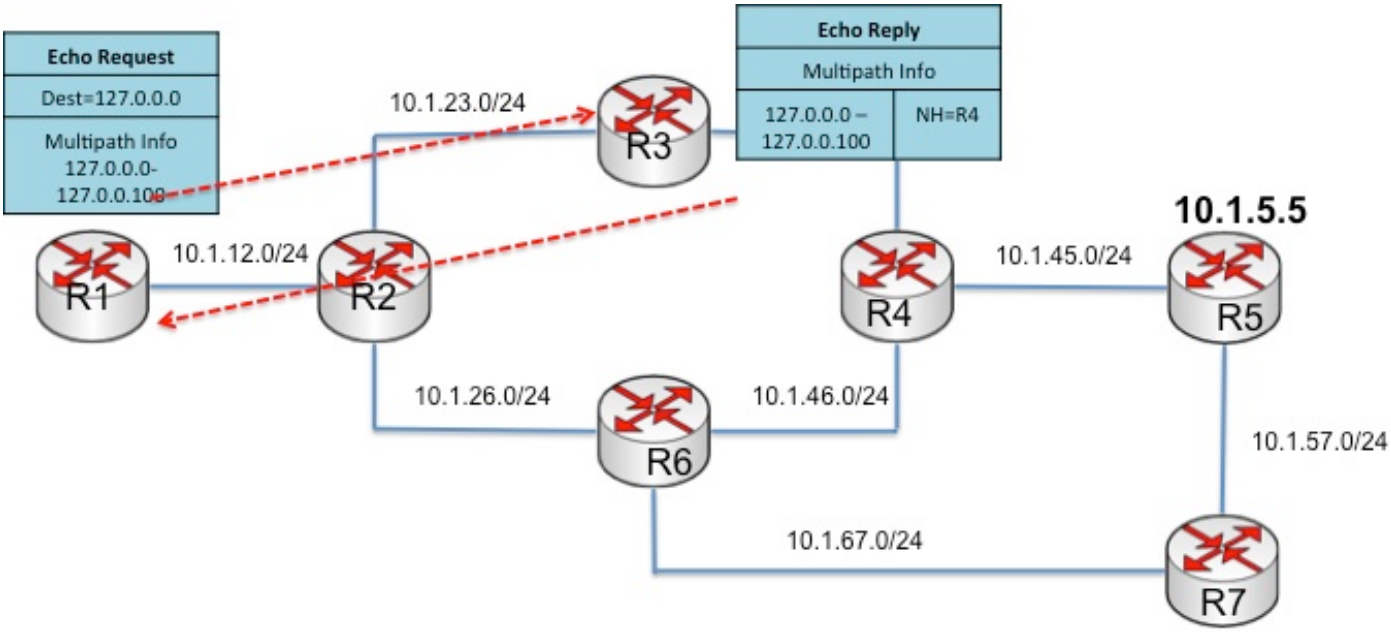
1) Инициатор (R1) передает запрос эха с ниже подробных данных:

- IP - адрес назначения как 127.0.0.0
- Многопутевой информационный TLV, несущий диапазон адресов как 127.0.0.0 к 127.0.0.200.
- TTL главной метки будет установлен в 1.

2) R2 при получении того же ответит назад с Многопутевой информацией для каждого исходящего интерфейса. В данном примере это ответит как указано ниже:

- Если IP - адрес назначения будет в 127.0.0.0 к 127.0.0.100, то пакет будет передан к R3.
- Если IP - адрес назначения будет в 127.0.0.101 к 127.0.0.200, то пакет будет передан к R6.

3) R1 понимает, что существует 2 возможных пути ECMP и таким образом, он должен передать 2 Запроса эха с набором TTL к 2. От различных тестов было замечено, что Инициатор всегда разрушает с 1 путем прежде, чем перейти затем. (Но это могло бы быть истинно для определенной реализации).

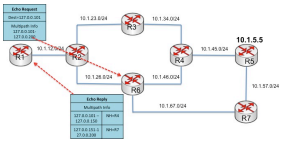


4) R1 теперь передает запрос эха с ниже подробных данных:

- IP - адрес назначения как 127.0.0.0
- Многопутевой информационный TLV, несущий диапазон адресов как 127.0.0.0 к 127.0.0.100.
- TTL главной метки будет установлен в 2.

5) R2 передаст пакет к R3 (как адрес назначения (DA) 127.0.0.0). R3 при получении того же ответит назад с той же Многопутевой информацией, поскольку существует только один исходящий интерфейс.

То же сохраняется, пока оно не достигает R5.



6) Как только трассировка PATH1 завершена (после того, как получение ответа от выхода), Инициатор теперь сделает запрос PATH2. Это выполнено путем передачи запроса эха с ниже подробных данных:

- IP - адрес назначения как 127.0.0.101
- Многопутевой информационный TLV, несущий диапазон адресов как 127.0.0.101 к 127.0.0.200
- TTL главного набора метки к 2.

7) R2 передаст пакет к R6 (как адрес назначения (DA) 127.0.0.101). R6 при получении того же ответит назад с Многопутевой информацией как указано ниже:

- Если IP - адрес назначения будет в 127.0.0.101 к 127.0.0.150, то пакет будет передан к R4.
- Если IP - адрес назначения будет в 127.0.0.151 к 127.0.0.200, то пакет будет передан к R7.

8) R1 понимает, что существует еще один путь ECMP, делающий общие возможные пути как 3. R1 продолжает сделать запрос PATH2 путем передачи следующего запроса эха с ниже подробных данных:

- IP - адрес назначения как 127.0.0.101
- Многопутевой информационный TLV, несущий диапазон адресов как 127.0.0.101 к 127.0.0.150
- TTL главного набора метки к 3.

9) R2 передаст пакет к R6 (как назначение 127.0.0.101), и R6 передаст его R4 (как назначение 127.0.0.101). R4 doesn? t имеют любой путь ECMP и так ответят назад с той же Многопутевой информацией. Следующий пакет достигнет выходного R5.

10) Так как трассировка PATH2 завершена, R1 продолжит запрос для PATH3. Это выполнено путем передачи запроса эха с ниже подробных данных:

- IP - адрес назначения как 127.0.0.151
- Многопутевой информационный TLV, несущий диапазон адресов как 127.0.0.151 к 127.0.0.200
- TTL главного набора метки к 3.

11) R2 будет передача пакетов к R6, который в свою очередь передаст его R7. R7 ответит назад с тем же Многопутевым информационным TLV. Следующий пакет достигает исходящего маршрутизатора R5.

После того, как эти шаги завершены, R1 будет иметь ниже подробных данных:

Multipath Information		
	Address Range	Path
PATH1	127.0.0.0 to 127.0.0.100	R1-R2-R3-R4-R5
PATH2	127.0.0.101 to 127.0.0.150	R1-R2-R6-R4-R5
PATH3	127.0.0.151 to 127.0.0.200	R1-R2-R6-R7-R8

В то время как использование адреса из других диапазонов будет влиять на передачу пакета по соответствующим путям, при помощи адреса назначения (DA) в 127.0.0.0 и 127.0.0.100, на пересылку пакетов будут влиять по PATH1.

12) Теперь Инициатор передаст 3 пакета эхо-запроса с набором TTL к 255 и выберет адрес из каждого диапазона так, чтобы все пути были проверены от начала до конца.

Команда, которая будет использоваться для трассировки ECMP, является *ipv4 traceroute mpls multipath <префикс> <mask>.* Придерживающееся является примером выходных данных.

```
R1#traceroute mpls multipath ipv4 10.1.5.5 255.255.255.255

Starting LSP Multipath Traceroute for 10.1.5.5/32
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
'L' - labeled output interface, 'B' - unlabeled output interface,
'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
'P' - no rx intf label prot, 'p' - premature termination of LSP,
'R' - transit router, 'I' - unknown upstream index,
'l' - Label switched with FEC change, 'd' - see DDMAP for return code,
'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
LLL!
Path 0 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.4
LL!
Path 1 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.2
L!
Path 2 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.0
Paths (found/broken/unexplored) (3/0/0)
Echo Request (sent/fail) (9/0)
Echo Reply (received/timeout) (9/0)
Total Time Elapsed 27 ms
```

Обратите внимание на то, что выше выходных данных показывает, что существует 3 пути, и все пути хорошо работают. Использование многословной кнопки в вышеупомянутой команде перечислит все переходы как указано ниже:

```
R1#traceroute mpls multipath ipv4 10.1.5.5 255.255.255.255 verbose

Starting LSP Multipath Traceroute for 10.1.5.5/32
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
'L' - labeled output interface, 'B' - unlabeled output interface,
'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
'P' - no rx intf label prot, 'p' - premature termination of LSP,
'R' - transit router, 'I' - unknown upstream index,
'l' - Label switched with FEC change, 'd' - see DDMAP for return code,
'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
LLL!
Path 0 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.4
0 10.1.12.1 10.1.12.2 MRU 1500 [Labels: 22 Exp: 0] multipaths 0
L 1 10.1.12.2 10.1.23.3 MRU 1500 [Labels: 23 Exp: 0] ret code 8 multipaths 2
```

```
L 2 10.1.23.3 10.1.34.4 MRU 1500 [Labels: 22 Exp: 0] ret code 8 multipaths 1
L 3 10.1.34.4 10.1.45.5 MRU 1500 [Labels: implicit-null Exp: 0] ret code 8 multipaths 1
! 4 10.1.45.5, ret code 3 multipaths 0
LL!
Path 1 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.2
0 10.1.12.1 10.1.12.2 MRU 1500 [Labels: 22 Exp: 0] multipaths 0
L 1 10.1.12.2 10.1.26.6 MRU 1500 [Labels: 16 Exp: 0] ret code 8 multipaths 2
L 2 10.1.26.6 10.1.46.4 MRU 1500 [Labels: 22 Exp: 0] ret code 8 multipaths 2
L 3 10.1.46.4 10.1.45.5 MRU 1500 [Labels: implicit-null Exp: 0] ret code 8 multipaths 1
! 4 10.1.45.5, ret code 3 multipaths 0
L!
Path 2 found,
output interface Et0/0.12 nexthop 10.1.12.2
source 10.1.12.1 destination 127.0.0.0
0 10.1.12.1 10.1.12.2 MRU 1500 [Labels: 22 Exp: 0] multipaths 0
L 1 10.1.12.2 10.1.26.6 MRU 1500 [Labels: 16 Exp: 0] ret code 8 multipaths 2
L 2 10.1.26.6 10.1.67.7 MRU 1500 [Labels: 17 Exp: 0] ret code 8 multipaths 2
L 3 10.1.67.7 10.1.57.5 MRU 1500 [Labels: implicit-null Exp: 0] ret code 8 multipaths 1
! 4 10.1.57.5, ret code 3 multipaths 0
Paths (found/broken/unexplored) (3/0/0)
Echo Request (sent/fail) (9/0)
Echo Reply (received/timeout) (9/0)
Total Time Elapsed 29 ms
```