Contents

Introduction

MPLS LSP Ping is a basic tool used to validate the health of the label switched path (LSP) between ingress and egress. This document aims to explain interaction of multipath information between initiator and responder in LSP tree trace. For detailed options available for this tool, it would be useful to refer this document.

Background Information

This implementation of the MPLS EM—MPLS LSP Multipath Tree Trace feature is based on RFC 4379, *Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures*.

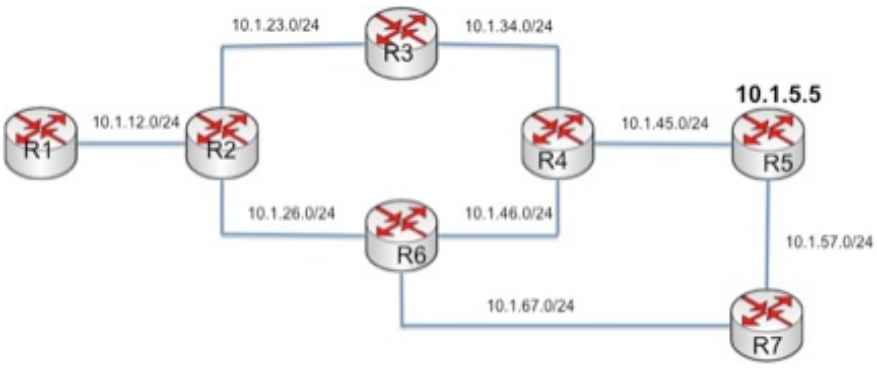
By setting the IP destination address of the probe packet as loopback address (127.x.x.x), LSP tree trace can be used to detect failure in LSP by avoiding the packet get IP routed. So whenever there is end-to-end connectivity issue, it is helpful to use LSP Ping as the first step to eliminate any LSP failure.

In case of multipath scenarios, LSP ping may not always help identify all LSP failures. As it might be noted, any label switch router (LSR) on receiving a labeled packet which can be sent out multiple egress interfaces, uses certain keys from the packet and input to hashing algorithm to decide the egress interface. Depending on vendor, hardware, etc, any of the below options may be considered for hashing:

1. Incoming label stack alone.
2. Incoming label stack and IP header details (If the payload is IP).
3. Incoming label stack, IP header, and transport header details.

Normally, Cisco routers consider a combination of label stack and IP header if the stack is of size less than or equal to 3 (with IP as the payload).

Assume following topology.



R1-R7 are routers. In above topology, there are 3 equal cost multi path (ECMP) routes from R1 to R5 as below,

PATH1: R1-R2-R3-R4-R5

PATH2: R1-R2-R6-R4-R5

PATH3: R1-R2-R6-R7-R5

Assume there is an issue between R6 and R7 (like broken label distribution protocol (LDP) or label misprogramming, etc.) causing traffic from R1 to R5 via PATH3 to drop. If LSP Ping from R1 takes PATH1 or PATH2, you may end up assuming that the path between R1 and R5 is fine.

LSP Ping allows setting the IP destination address as any one from 127.0.0.0/8 range. While one simple option is to manually try sending multiple ping packets with different destination address, there is no guarantee that all possible ECMP paths will be validated. You need a way that queries and validates all possible paths between source and destination. LSP Multipath tree trace leverages the “Multipath Information Encoding” defined in Section 3.3.1 of RFC4379 and helps you to validate all ECMP paths.

LSP Tree trace - How it works

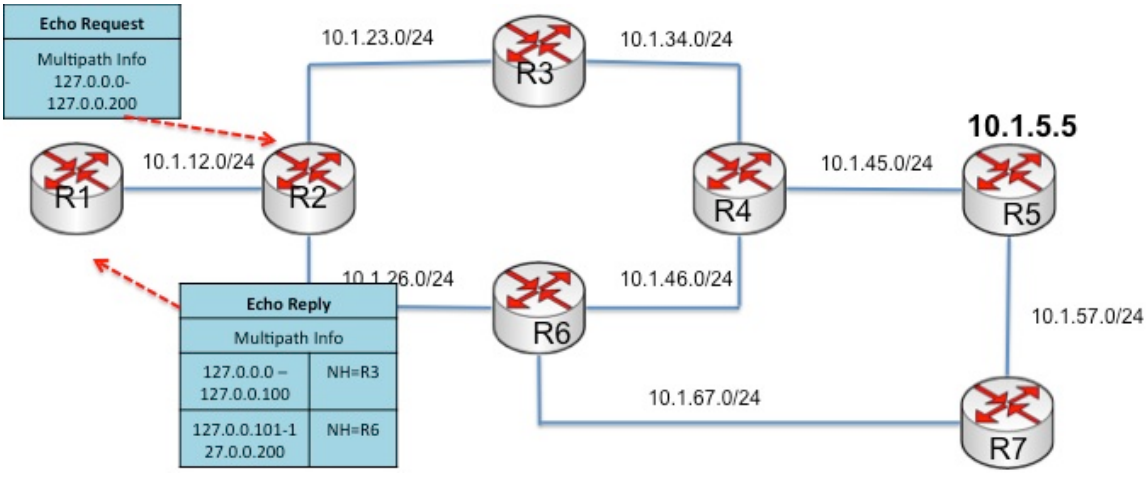
A regular MPLS ping or traceroute may indicate that there is no failure depending on how the transit routers load-share the packets over ECMP, however LSP tree trace provides a better method to validate that all paths are actually working.

In LSP tree trace, initiator router sends MPLS echo request to each hop by setting the TTL in the top label in an incremental manner (starting from 1). The echo request will carry Multipath Information TLV that carries a range of IP address (within 127.0.0.0/8 range) or entropy label range. Currently Cisco devices support the IP destination option and so our example will be detailed with IP address range.

Each transit LSR on receiving the request packet will reply with all ECMP outgoing interfaces and associate a range of IP address (or entropy label) from the request for each interface.

LSP Tree trace - Detailed Example

Assume following topology for example below.



For simplicity, this example uses address range of 127.0.0.0-127.0.0.200. Here are the details of

steps in a LSP tree trace.

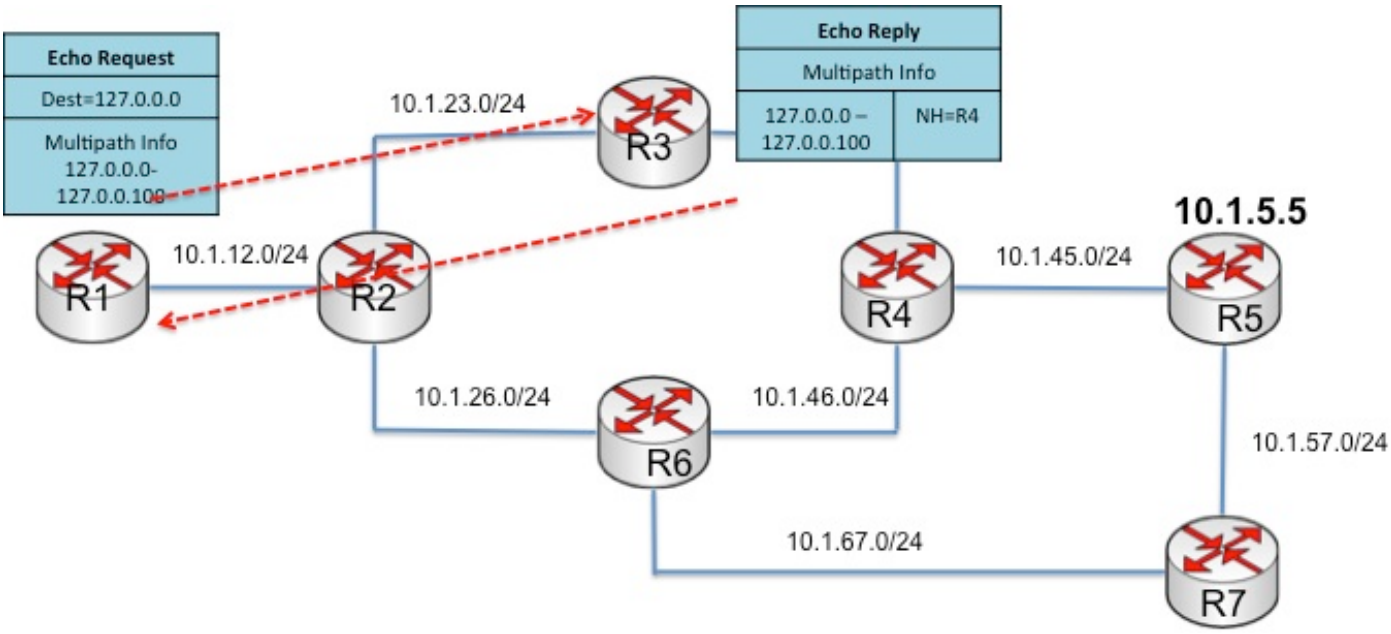
1) Initiator (R1) sends the echo request with below details:

- IP destination as 127.0.0.0
- Multipath Information TLV carrying the address range as 127.0.0.0 to 127.0.0.200.
- The TTL of the top label will be set to 1.

2) R2 on receiving the same will reply back with Multipath Information for each egress interface. In this example, it will reply as below:

- If IP destination is within 127.0.0.0 to 127.0.0.100, packet will be sent to R3.
- If IP destination is within 127.0.0.101 to 127.0.0.200, packet will be sent to R6.

3) R1 realizes that there are 2 possible ECMP paths and so it needs to send 2 Echo Request with TTL set to 2. From various tests, it was observed that the Initiator always finishes off with 1 path before going to next. (But this might be true for a specific implementation).

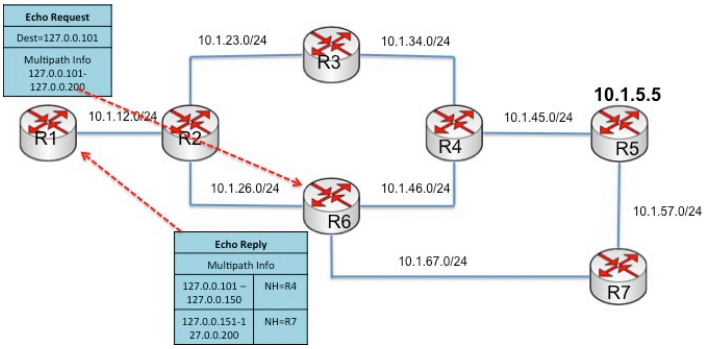


4) R1 now sends the echo request with below details:

- IP destination as 127.0.0.0
- Multipath Information TLV carrying the address range as 127.0.0.0 to 127.0.0.100.
- The TTL of the top label will be set to 2.

5) R2 will forward the packet to R3 (as the destination address is 127.0.0.0). R3 on receiving the same will reply back with same Multipath Information, as there is only one egress interface.

The same holds true till it reaches R5.



6) Once PATH1 trace is complete (after receiving reply from egress), Initiator will now query PATH2. This is performed by sending the echo request with below details:

- IP destination as 127.0.0.101
- Multipath Information TLV carrying the address range as 127.0.0.101 to 127.0.0.200
- The TTL of the top label set to 2.

7) R2 will forward the packet to R6 (as the destination address is 127.0.0.101). R6 on receiving the same will reply back with Multipath Information as below:

- If IP destination is within 127.0.0.101 to 127.0.0.150, packet will be sent to R4.
- If IP destination is within 127.0.0.151 to 127.0.0.200, packet will be sent to R7.

8) R1 realizes that there is one more ECMP path making the total possible paths as 3. R1 continues to query PATH2 by sending the next echo request with below details:

- IP destination as 127.0.0.101
- Multipath Information TLV carrying the address range as 127.0.0.101 to 127.0.0.150
- The TTL of top label set to 3.

9) R2 will forward the packet to R6 (as the destination is 127.0.0.101) and R6 will forward it to R4 (as the destination is 127.0.0.101). R4 doesn't have any ECMP path and so will reply back with same Multipath Information. The next packet will reach egress R5.

10) Since PATH2 trace is complete, R1 will continue the query for PATH3. This is performed by sending the echo request with below details:

- IP destination as 127.0.0.151
- Multipath Information TLV carrying the address range as 127.0.0.151 to 127.0.0.200
- The TTL of top label set to 3.

11) R2 will forward packet to R6, which in turn will forward it to R7. R7 will reply back with same Multipath Information TLV. The next packet reaches egress router R5.

After these steps are complete, R1 will have below details:

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

By using destination address within 127.0.0.0 and 127.0.0.100, the packet forwarding will be influenced over PATH1 while using the address from other ranges will influence forwarding the packet over respective paths.

12) Now Initiator will send 3 echo request packets with TTL set to 255 and select address from each range so that all paths will be validated end-to-end.

The command to be used for ECMP trace is *traceroute mpls multipath ipv4 <prefix> <mask>.* Following is a sample output.

```
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
```

Note that above output shows that there are 3 paths and all paths are working fine. Using verbose knob in above command will list all the hops as below:

```
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
```