

# Table of Contents

<b><u>The Traceroute Command in MPLS</u></b> .....	<b>1</b>
<u>Document ID: 26585</u> .....	1
<u>Introduction</u> .....	1
<u>Prerequisites</u> .....	1
<u>Requirements</u> .....	1
<u>Components Used</u> .....	1
<u>Conventions</u> .....	1
<u>Normal traceroute Command</u> .....	1
<u>MPLS traceroute Command</u> .....	2
<u>no mpls ip propagate-ttl Command</u> .....	7
<u>NetPro Discussion Forums – Featured Conversations</u> .....	9
<u>Related Information</u> .....	9

# The Traceroute Command in MPLS

Document ID: 26585

---

## Introduction

### Prerequisites

Requirements

Components Used

Conventions

### Normal traceroute Command

### MPLS traceroute Command

### no mpls ip propagate-ttl Command

### NetPro Discussion Forums – Featured Conversations

### Related Information

---

## Introduction

This document explains the way the **traceroute** command works in a Multiprotocol Label Switching (MPLS) environment.

## Prerequisites

### Requirements

Readers of this document should have:

- Basic MPLS knowledge

For information, refer to MPLS FAQ For Beginners.

### Components Used

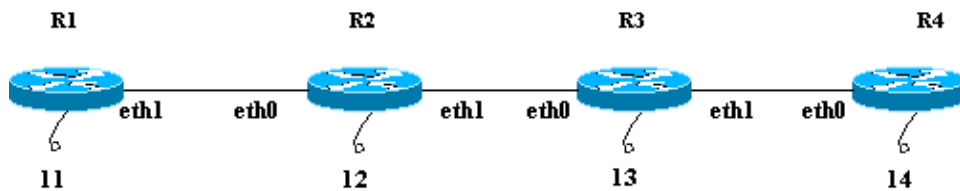
This document is not restricted to specific software and hardware versions.

### Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

## Normal traceroute Command

This section describes how a traditional **traceroute** command works. A service provider setup is shown below where Router 1 (R1) and Router 4 (R4) are Provider Edge (PE) routers and Router 2 (R2) and Router 3 (R3) are the Provider (P) routers.



This example conducts a **traceroute** to the R4 loopback 14 from R1. R1 uses a User Datagram Protocol (UDP) datagram with an arbitrary destination port value greater than 32000. Selecting such a high value for the port number ensures that such a port does not exist on the intended recipient. It puts this datagram in an IP packet.

**Note:** Throughout this document, whenever an IP packet is mentioned, it is an IP packet that contains the UDP datagram.

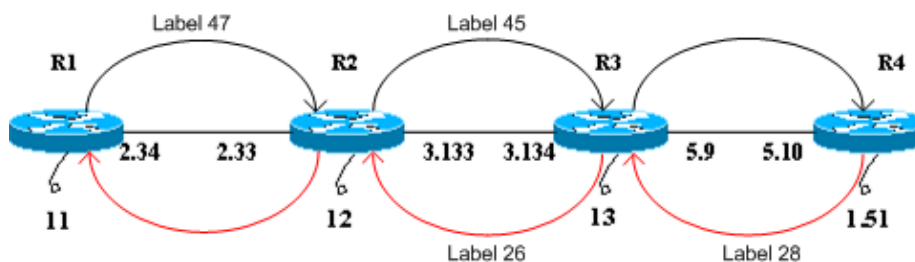
This is a sequence of events for a normal **traceroute** command:

1. R1 sends the IP packet with a destination address of 14 and a Time to Live (TTL) of 1 through its eth1 interface.
2. R2 receives the packet and notes that it is not the intended recipient and the TTL of the packet is 1. It drops the packet and sends a TTL-expired Internet Control Message Protocol (ICMP) message to R1. The source address of this ICMP message is the IP address of the R2 eth0 (the address of the interface that received the original packet).
3. On receipt of the ICMP message, R1 sends another IP packet destined for 14 with a TTL of 2 through its eth1 interface.
4. R2 receives the packet, notes that it is not the intended recipient and the intended recipient can be reached through R3. It decrements the TTL (from 2 to 1) and forwards the packet to R3. R3 receives the packet and sees that it is not the intended recipient. The TTL is 1. It drops the packet and sends a TTL-expired ICMP message to R1 with its eth0 address as the source address.
5. R1 receives the ICMP message and sends another IP packet to 14 through its eth1 interface with a TTL value of 3. On the way, R2 and R3 decrement the TTL and pass it on to R4. R4 gets the packet, sees that it is the intended recipient, and tries to connect to the port value in the UDP datagram. R4 finds this port does not exist and sends an ICMP `port unreachable` error message to R1.

As before, the source address of this ICMP message is eth0 of R4. The **traceroute** program now has all the ICMP error messages with the corresponding source addresses and has the complete route to the destination.

## MPLS traceroute Command

Consider the same scenario detailed above in the Normal traceroute Command section, except all routers, R1 through R4, now do label switching instead of IP forwarding. The test bed setup is shown below. All the interfaces shown in the test bed are in the 10.13.0.0 network.



For the purpose of this document, assume that:

- R1 uses a label of 47 to reach R4 and forwards packets to R2.
- R2 uses a label of 45 to reach R4 and forwards packets to R3.
- R3 pops the label and forwards the packet to R4.
- R4 uses a label of 28 to reach R1 and forwards packets to R3.
- R3 uses a label of 26 to reach R1 and forwards packets to R2.
- R2 pops the label and forwards the packet to R1.

The following steps show the sequence of events for conducting a **traceroute** from R1 to the R4 loopback 10.13.1.51.

1. R1 sends a label-switched packet with a label of 47 and a TTL of 1 to R2. The TTL field of the IP packet is copied onto the TTL field of the label header.
2. R2 sees that it is not the intended recipient and the TTL is 1. It drops the packet and creates a TTL-expired ICMP message, as it would for a regular IP packet. In this case, the ICMP message packet is generated per ICMP extensions for MPLS.
3. R2 appends the label 47 (the incoming label that expired) to the ICMP message. It does not send the packet to R1 directly. Instead, it consults its label forwarding information base (LFIB) and finds it should use a label of 45 for packets received with a label of 47. It puts a label of 45 on the packet and sends the TTL-expired ICMP message to R3.
4. R3 pops the label and sends it to R4. R4 sees that the destination is R1, gives a label of 28 to the message, and sends it through R3 and R2 to R1.
5. The ICMP error message travels all the way to the other end before it is sent back to R1. This example provides an illustration:



The sniffed packets on the Ethernet interface at R4 confirm Steps 1 , above. In the following sniffer output, **Frame 1** is the incoming packet and **Frame 2** is the outgoing packet from R4. The output is formatted to reflect this discussion, and points to be noted are in boldface.

```
Frame 1 (182 on wire, 182 captured)
Ethernet II
Destination: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Source: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Type: IP (0x0800)
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 254
Protocol: ICMP (0x01)
Header checksum: 0x1b8e (correct)
Source: 10.13.2.33 (10.13.2.33)
Destination: 10.13.2.34 (10.13.2.34)
Internet Control Message Protocol
Type: 11 (Time-to-live exceeded)
```

```

Code: 0 (TTL equals 0 during transit)
Checksum: 0x0c88 (correct)
Data (140 bytes)
04500 001c 9e19 0000 0111 044a 0a0d 0222E.....J..."
100a0d 0133 989d 829a 0008 cd37 0000 0000...3.....7....
200000 0000 0000 0000 0000 0000 0000 0000.....
300000 0000 0000 0000 0000 0000 0000 0000.....
400000 0000 0000 0000 0000 0000 0000 0000.....
500000 0000 0000 0000 0000 0000 0000 0000.....
600000 0000 0000 0000 0000 0000 0000 0000.....
700000 0000 0000 0000 0000 0000 0000 0000.....
802000 edf2 0008 0101 0002 f101.....

```

```

Frame 2 (186 on wire, 186 captured)
Ethernet II
Destination: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Source: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header
MPLS Label: Unknown (28)
MPLS Experimental Bits: 6
MPLS Bottom Of Label Stack: 1
MPLS TTL: 253
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 253
Protocol: ICMP (0x01)
Header checksum: 0x1c8e (correct)
Source: 10.13.2.33 (10.13.2.33)
Destination: 10.13.2.34 (10.13.2.34)
Internet Control Message Protocol
Type: 11 (Time-to-live exceeded)
Code: 0 (TTL equals 0 during transit)
Checksum: 0x0c88 (correct)
Data (140 bytes)
04500 001c 9e19 0000 0111 044a 0a0d 0222E.....J..."
100a0d 0133 989d 829a 0008 cd37 0000 0000...3.....7....
200000 0000 0000 0000 0000 0000 0000 0000.....
300000 0000 0000 0000 0000 0000 0000 0000.....
400000 0000 0000 0000 0000 0000 0000 0000.....
500000 0000 0000 0000 0000 0000 0000 0000.....
600000 0000 0000 0000 0000 0000 0000 0000.....
700000 0000 0000 0000 0000 0000 0000 0000.....
802000 edf2 0008 0101 0002 f101.....

```

In **Frame 1** of the output above, the first packet received by R4 is the TTL–expired ICMP message from R2 (10.13.2.33, the interface on which the original packet was received) to R1 (10.13.2.34). In the data portion of the ICMP message, at bytes 0x89 and the first nibble of 0x8A, the MPLS label (20 bytes) is expired and its value is 0x02F, or 47. This is the incoming label of the packet with a TTL of 1. R2 appends this label in the ICMP error message.

In **Frame 2** of the output above, the type is shown as MPLS label switched packet, which means that it is an MPLS packet. R4 puts a label of 28 to Frame 1 and forwards it to R1 through the label–switching path. The MPLS header in the frame is in bold. Also, if you refer to the TTL portion of the packet, in Frame 1 its value is 254, and in Frame 2 it is 253. R4 has decremented it by 1.

- R1 receives the ICMP message and sends another packet with a label of 47 and a TTL of 2 to R2. R2 swaps labels, decrements TTL (from 2 to 1) and forwards to R3. As in Step 2, R3 sends a TTL–expired ICMP message appended with the incoming label that expired to R4, and R4 then sends it back to R1.

The sniffer output at R4, shown here, confirms Step 6:

```
Frame 3 (182 on wire, 182 captured)
Ethernet II
Destination: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Source: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Type: IP (0x0800)
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 255
Protocol: ICMP (0x01)
Header checksum: 0x146f (correct)
Source: 10.13.3.134 (10.13.3.134)
Destination: 10.13.2.34 (10.13.2.34)
Internet Control Message Protocol
Type: 11 (Time-to-live exceeded)
Code: 0 (TTL equals 0 during transit)
Checksum: 0x0c88 (correct)
Data (140 bytes)
04500 001c 9e1b 0000 0211 0348 0a0d 0222E.....H..."
100a0d 0133 9292 829b 0008 d341 0000 0000...3.....A....
200000 0000 0000 0000 0000 0000 0000 0000.....
300000 0000 0000 0000 0000 0000 0000 0000.....
400000 0000 0000 0000 0000 0000 0000 0000.....
500000 0000 0000 0000 0000 0000 0000 0000.....
600000 0000 0000 0000 0000 0000 0000 0000.....
700000 0000 0000 0000 0000 0000 0000 0000.....
802000 0df3 0008 0101 0002 d101.....
```

```
Frame 4 (186 on wire, 186 captured)
Ethernet II
Destination: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Source: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header
MPLS Label: Unknown (28)
MPLS Experimental Bits: 6
MPLS Bottom Of Label Stack: 1
MPLS TTL: 254
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 254
Protocol: ICMP (0x01)
Header checksum: 0x156f (correct)
Source: 10.13.3.134 (10.13.3.134)
Destination: 10.13.2.34 (10.13.2.34)
Internet Control Message Protocol
Type: 11 (Time-to-live exceeded)
Code: 0 (TTL equals 0 during transit)
Checksum: 0x0c88 (correct)
Data (140 bytes)
04500 001c 9e1b 0000 0211 0348 0a0d 0222E.....H..."
100a0d 0133 9292 829b 0008 d341 0000 0000...3.....A....
200000 0000 0000 0000 0000 0000 0000 0000.....
300000 0000 0000 0000 0000 0000 0000 0000.....
400000 0000 0000 0000 0000 0000 0000 0000.....
500000 0000 0000 0000 0000 0000 0000 0000.....
600000 0000 0000 0000 0000 0000 0000 0000.....
700000 0000 0000 0000 0000 0000 0000 0000.....
802000 0df3 0008 0101 0002 d101.....
```

From the **Frame 3** output above, you can determine that Frame 3 is the ICMP packet from R3 to R1. The source address (10.13.3.134) is the address on which the original packet is received. The ICMP error message contains the expired label information at the end of the data portion. Its value is 0x02d, which is 45. **Frame 4** is the MPLS packet that is sent from R4 to R1.

7. Upon receipt of the ICMP message, R1 sends another packet with a label of 47 and a TTL of 3. On its way, R2 and R3 decrement the TTL and forward the packet to R4. R4 notes that it is the intended recipient and finds the UDP datagram port unreachable. It sends an ICMP port unreachable message to R1 through R3 and R2.

In this sniffer output, important points to be noted are in boldface:

```
Frame 5 (60 on wire, 60 captured)
Ethernet II
Destination: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Source: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Type: IP (0x0800)
Trailer: 00000000000000000000000000000000...
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 1
Protocol: UDP (0x11)
Header checksum: 0x0446 (correct)
Source: 10.13.2.34 (10.13.2.34)
Destination: 10.13.1.51 (10.13.1.51)
User Datagram Protocol
Source port: 37647 (37647)
Destination port: 33436 (33436)
Length: 8
Checksum: 0xd2c3 (correct)

Frame 6 (74 on wire, 74 captured)
Ethernet II
Destination: 00:03:fd:1c:86:84 (Cisco_1c:86:84)
Source: 00:04:4e:7a:74:00 (Cisco_7a:74:00)
Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header
MPLS Label: Unknown (28)
MPLS Experimental Bits: 6
MPLS Bottom Of Label Stack: 1
MPLS TTL: 255
Internet Protocol
Version: 4
Header length: 20 bytes
Time to live: 255
Protocol: ICMP (0x01)
Header checksum: 0x5694 (correct)
Source: 10.13.5.10 (10.13.5.10)
Destination: 10.13.2.34 (10.13.2.34)
Internet Control Message Protocol
Type: 3 (Destination unreachable)
Code: 3 (Port unreachable)
Checksum: 0x1485 (correct)
Data (28 bytes)
04500 001c 9e1d 0000 0111 0446 0a0d 0222E.....F..."
100a0d 0133 930f 829c 0008 d2c3...3.....
```

**Frame 5** shows that the UDP datagram is sent by R1 to R4. The destination port value in the UDP datagram is 33436 (greater than 32000), as discussed in the Normal traceroute Command section.

In **Frame 6**, R4 sends a destination unreachable ICMP type and a port unreachable code to R1. All the earlier ICMP messages from R2 and R3 had the type field set as time-to-live exceeded. The output of the extended **tracert** command is shown here:

```
R1#tracert
Protocol [ip]:
Target IP address: 10.13.1.51
Source address: 10.13.2.34
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]: 1
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 10.13.1.51
 0 10.13.2.33 [MPLS: Label 47 Exp 0] 0 msec
 1 10.13.3.134 [MPLS: Label 45 Exp 0] 0 msec
 2 10.13.5.10 4 msec
R1#
```

By default, the **tracert** command uses three probes for each TTL value. It sends three packets with a TTL of 1, three packets with a TTL of 2, and so on. This **tracert** command is issued with a single probe, so it is easy to trace and debug. As seen in the output above, the **tracert** command shows the expired label value, also.

## no mpls ip propagate-ttl Command

When configuring MPLS, a label is imposed by the label switch router (LSR) when an IP packet is forwarded into the MPLS domain. This label must have a value in the TTL field. By default, LSR reads the TTL field in the IP header of the incoming packet, decrements it by 1, and copies what is left into the TTL field of the MPLS header. The core LSRs only look at the uppermost label. If the TTL value does not reach 0, the packet is forwarded. The egress edge LSR that pops the label copies what is left in the label TTL field into the TTL field of the IP header and then forwards the IP packet outside the MPLS domain.

This behavior can be changed by issuing the **no mpls ip propagate-ttl** configuration command. The ingress edge LSR uses the value of 255 as the TTL value in the label when imposing it. The egress edge LSR does not copy the label TTL value into the IP header when popping the label. The net result is that the IP header TTL does not reflect the hops taken across the MPLS core; so when customers perform a **tracert** from one side of their network to another side, the routers in the MPLS core network do not appear in the **tracert** information. It is important to disable TTL propagation in both the ingress and egress edge LSRs. Otherwise, the IP header may have a higher value when it leaves the MPLS domain than it had when it entered it.

This provides an example:



C1 performs a **traceroute** to C2. With the default IP TTL propagation operation, the **traceroute** in C1 looks like this:

```
C1#traceroute C2.cust.com

Tracing the route to C2.cust.com
 0  1 A.provider.net          44 msec  36 msec  32 msec
 1  2 B.provider.net         164 msec 132 msec 128 msec
 2  3 C.provider.net         148 msec 156 msec 152 msec
 3  4 C2.cust.com            180 msec * 181 msec
```

This output illustrates typical **traceroute** behavior in an MPLS network. As the label header of a labeled packet carries the TTL value from the original IP packet, the routes in the path drop packets for which the TTL is exceeded. Therefore, **traceroute** shows all the routers in the path. The behavior is the following:

1. The first packet is an IP packet with TTL equal to 1. Router A decrements the TTL and drops the packet because it reaches 0. An ICMP TTL-exceeded message is sent to the source.
2. The second packet sent is an IP packet with TTL equal to 2. Router A decrements the TTL, labels the packet, and forwards the packet to Router B.
3. Router B decrements the TTL value in the MPLS header, drops the packet, and sends an ICMP TTL-exceeded message to the source. Since it was an MPLS packet that was dropped, the return address for the ICMP message must be derived from the source address in the IP header inside the MPLS packet. But that IP address actually may not be known to Router B, so Router B forwards the ICMP messages along the same label switched path (LSP) that the dropped packet was traveling (in the direction toward router C). At the end of the LSP, the label is removed and the ICMP messages are forwarded according to the destination address in the IP header (toward Router C1).
4. The third packet (TTL is 3) experiences similar processing as the previous packets, except that Router C is now the one dropping the packet, based on the TTL in the IP header. Router B, because of penultimate hop popping, previously removed the label, and the TTL was copied in the IP header.
5. The fourth packet (TTL is 4) reaches the final destination where the TTL of the IP header is examined.

If IP TTL propagation is disabled with the **no mpls ip propagate-ttl** command in global configuration mode, the TTL value is not copied in the IP header and the **traceroute** in C1 to C2 looks like this:

```
C1#traceroute C2.cust.com

Tracing the route to C2.cust.com
 0  1 A.provider.net          44 msec  36 msec  32 msec
 1  2 C2.cust.com            180 msec * 181 msec
```

When the **traceroute** command is used in this situation, the ICMP replies are only received from those routers that see real TTL stored in the IP header. In this situation, Router C1 is executing a **traceroute** command (as shown above), but the core routers do not copy the TTL to and from the label. This results in the following behavior:

1. The first packet is an IP packet with TTL equal to 1. Router A decrements the TTL, drops the packet, and sends an ICMP TTL-exceeded message to the source.
2. The second packet is an IP packet with TTL equal to 2. Router A decrements the TTL, labels the packet, and sets the TTL in the MPLS header to 255.
3. Router B decrements the TTL in the MPLS header to 254 and forwards a labeled packet with TTL set to 254.
4. Router C removes the label, decrements the IP TTL, and sends the packet to the next hop Router C2. The packet has reached the final destination.

# NetPro Discussion Forums – Featured Conversations

Networking Professionals Connection is a forum for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. The featured links are some of the most recent conversations available in this technology.

NetPro Discussion Forums – Featured Conversations for RP
Service Providers: MPLS
Virtual Private Networks: Services
Virtual Private Networks: Security

## Related Information

- [Understanding the Ping and Traceroute Commands](#)
- [mpls ip propagate-ttl Command](#)
- [MPLS Support Page](#)
- [Technical Support – Cisco Systems](#)

---

All contents are Copyright © 1992–2005 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

---

Updated: Dec 29, 2005

Document ID: 26585

---