

(excerpt from chapter 3)

Debugging techniques in “TraceRoute in Detail” example

In this next scenario traceroute is first explained and then demonstrated.

”Traceroute” is a network debugging utility that attempts to trace the path a packet takes through the network - its route. A key word here is “attempts” - by no means does traceroute work in all cases.

As part of the IP packet options (record route and its variants), methods exist to trace the path of a packet. Traceroute does not depend on any of these facilities.

How Traceroute Works

Traceroute transmits packets with small TTL values. A TTL value (Time To Live) is an IP header field that is designed to prevent packets from being forwarded in an infinite loop. Every router that forwards a packet subtracts one from the packet's TTL. If the TTL reaches zero, the packet has expired and is discarded. Traceroute depends on the common router practice of sending an ICMP Time Exceeded message, documented in RFC 792, back to the originating router when this occurs. By using small TTL values which quickly expire, traceroute causes routers along a packet's normal delivery path to generate these ICMP messages which help identify the router's along a given path. A TTL value of one should produce a message from the first router; a TTL value of two generates a message from the second; etc.

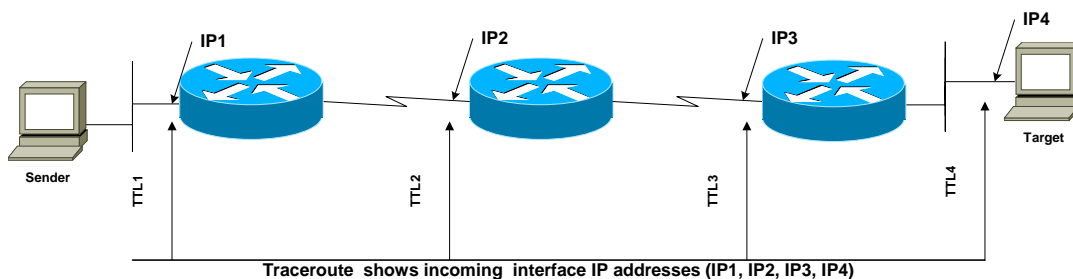


Fig1. How Traceroute Works

In a typical traceroute session, a group of packets with TTL=1 are sent. A single router should respond, using the IP address of the interface it transmits the ICMP Timeout messages on, which should be the same as the interface it received the original packets on. The user is told this IP address, and DNS is used to convert this into a symbolic domain address. Also, round trip times are reported for each packet in the group. Traceroute reports any additional ICMP messages (such as destination unreachable) using a rather cryptic syntax - !N means network unreachable, !H means host unreachable, etc. Once this first group of packets has been processed (this can take 10 seconds or no time at all), the second group (TTL=2) begins transmitting, and the whole process repeats.

Problems you might encounter:

Since TCP/IP is not designed to support traceroute, several kinds of problems might arise.

Changing paths

Always remember - you are not tracing the path of one packet, but of many. Hopefully, all packets will follow the same route, but this is by no means assured. What if a link fails during a traceroute? Your packets may be rerouted, and traceroute's output becomes a confused combination of two separate routes.

No sending addresses

You only see one IP address from each router - the address closest to you. To put it another way, traceroute can't tell you which interfaces routers are sending the packets on. It only shows the interfaces packets are being received on. The sending interfaces can often be deduced by matching each router with the next one in line - typically only one interface could be used between them.

Routing problems

TCP/IP's sinister and ubiquitous routing problems may cause the router not to have a route back to the sender, or to have a route through some interface other than the one it received the packet on. In these cases, you will either receive no reply at all (no route), or a reply showing an IP address that never handled the original packet (it was handled by some other interface on the same router). In short, don't completely trust traceroute.

Traceroute Scenario

Now that you possess an understanding of how traceroute works, let's examine some debug output generated by traceroute. In the following scenario, traceroute is used to discover the route from router R1(172.16.10.1) to R3 (172.16.20.3). Just as with the PING utility demonstrated in the earlier scenario, each router will be configured with an access list:

```
access-list 100 permit icmp any any
access-list 100 permit udp any any gt 30000
```

These access-lists will restrict the debug output to display only packets involved with traceroute.

The access-lists are applied to the following two debug commands:

```
debug ip packet 100 detail
deb ip error 100 detail
```

The traceroute is initiated from router R1:

```
1.  r1#trace
2.  Protocol [ip]:
3.  Target IP address: 172.16.20.3
4.  Source address:
5.  Numeric display [n]:
6.  Timeout in seconds [3]:
7.  Probe count [3]:
8.  Minimum Time to Live [1]:
9.  Maximum Time to Live [30]:
10. Port Number [33434]:
11. Loose, Strict, Record, Timestamp, Verbose[none]:
12. Type escape sequence to abort.
13. Tracing the route to 172.16.20.3
14.  1 172.16.10.2 12 msec 8 msec 12 msec
```

15. 2 172.16.20.3 16 msec * 16 msec

```

1. Aug 30 12:57:08.250: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
2. Aug 30 12:57:08.250:   UDP src=41042, dst=33434
3. Aug 30 12:57:08.262: IP: s=172.16.10.2 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
4. Aug 30 12:57:08.262:   ICMP type=11, code=0
5. Aug 30 12:57:08.262: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
6. Aug 30 12:57:08.262:   UDP src=38330, dst=33435
7. Aug 30 12:57:08.270: IP: s=172.16.10.2 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
8. Aug 30 12:57:08.270:   ICMP type=11, code=0
9. Aug 30 12:57:08.270: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
10. Aug 30 12:57:08.270:   UDP src=42074, dst=33436
11. Aug 30 12:57:08.278: IP: s=172.16.10.2 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
12. Aug 30 12:57:08.282:   ICMP type=11, code=0
13. Aug 30 12:57:08.282: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
14. Aug 30 12:57:08.282:   UDP src=34027, dst=33437
15. Aug 30 12:57:08.298: IP: s=172.16.20.3 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
16. Aug 30 12:57:08.298:   ICMP type=3, code=3
17. Aug 30 12:57:08.302: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
18. Aug 30 12:57:08.302:   UDP src=42765, dst=33438
19. Aug 30 12:57:11.298: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
20. Aug 30 12:57:11.298:   UDP src=36551, dst=33439
21. Aug 30 12:57:11.314: IP: s=172.16.20.3 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
22. Aug 30 12:57:11.314:   ICMP type=3, code=3

```

In the debug output generated on router R1 above, the first traceroute UDP packet was sent at 12:57:08.250 (see Line 1 of R1 debug output). The TTL was set to 1. Therefore, when a packet was received on router R2, R2 decremented the received packet's TTL and discarded the packet, generating a "dispose ip.hopcount" error message (line 3 of R2 debug output). Also, router R2 sent an ICMP packet with type=11, code=0 which is "time to live exceeded in transit" message stamped with the time 12:57:08.262. See Appendix for all ICMP message type/code values.

When comparing the timestamp of the packets generated by router R1 with those received by R1 from the first hop router R2, you will derive the roundtrip time displayed in the traceroute. See line 14 of the "extended traceroute command" display above:

14 172.16.10.2 12 msec 8 msec 12 msec

The first time listed: 12 msec (the difference between lines 1 and 3 in the debug output displayed above)
The second time listed: 8 msec (the difference between lines 5 and 7 in the debug output displayed above)
The first time listed: 12 msec (the difference between lines 9 and 12 in the debug output displayed above)

The following debug output is generated by the intermediate router R2:

```

1. Aug 30 12:57:08.257: IP: s=172.16.10.2 (local), d=172.16.10.1 (Serial0/0), len 56, sending
2. Aug 30 12:57:08.257:   ICMP type=11, code=0
3. Aug 30 12:57:08.257: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose ip.hopcount
4. Aug 30 12:57:08.257:   UDP src=41042, dst=33434
5. Aug 30 12:57:08.265: IP: s=172.16.10.2 (local), d=172.16.10.1 (Serial0/0), len 56, sending
6. Aug 30 12:57:08.265:   ICMP type=11, code=0
7. Aug 30 12:57:08.265: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose ip.hopcount
8. Aug 30 12:57:08.265:   UDP src=38330, dst=33435
9. Aug 30 12:57:08.277: IP: s=172.16.10.2 (local), d=172.16.10.1 (Serial0/0), len 56, sending
10. Aug 30 12:57:08.277:   ICMP type=11, code=0
11. Aug 30 12:57:08.277: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose ip.hopcount

```

```
12. Aug 30 12:57:08.277: UDP src=42074, dst=33436
13. Aug 30 12:57:08.285: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3 (Serial0/1), g=172.16.20.3, len 28, forward
14. Aug 30 12:57:08.285: UDP src=34027, dst=33437
15. Aug 30 12:57:08.293: IP: s=172.16.20.3 (Serial0/1), d=172.16.10.1 (Serial0/0), g=172.16.10.1, len 56, forward
16. Aug 30 12:57:08.297: ICMP type=3, code=3
17. Aug 30 12:57:08.305: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3 (Serial0/1), g=172.16.20.3, len 28, forward
18. Aug 30 12:57:08.305: UDP src=42765, dst=33438
19. Aug 30 12:57:11.301: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3 (Serial0/1), g=172.16.20.3, len 28, forward
20. Aug 30 12:57:11.301: UDP src=36551, dst=33439
21. Aug 30 12:57:11.309: IP: s=172.16.20.3 (Serial0/1), d=172.16.10.1 (Serial0/0), g=172.16.10.1, len 56, forward
22. Aug 30 12:57:11.309: ICMP type=3, code=3
```

On lines, 1, 5 and 9, router R2 is sending back to router R1 an ICMP Type 11 Code 0 “time to live exceeded” message. When comparing the timestamps of these packets generated on router R2 with the corresponding time stamps of the packets generated on R1, it can be determined that all of these packets are related to the first set of packets generated by router R1. Also, the “dispose.ip.hopcount” error messages on line 3, 7 and 11 are also associated with this same set of packets.

Packets 13-22 are associated with the second set of packets generated by router R1. These packets have their TTL set to 2. They are forwarded by router R2.

The following debug output is generated by the target of the traceroute, router R3.

```
23. Aug 30 12:57:08.289: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, rcvd 0
24. Aug 30 12:57:08.289: UDP src=34027, dst=33437
25. Aug 30 12:57:08.289: IP: s=172.16.20.3 (local), d=172.16.10.1 (Serial0/0), len 56, sending
26. Aug 30 12:57:08.289: ICMP type=3, code=3
27. Aug 30 12:57:08.289: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose udp.noport
28. Aug 30 12:57:08.289: UDP src=34027, dst=33437
29. Aug 30 12:57:08.309: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, rcvd 0
30. Aug 30 12:57:08.309: UDP src=42765, dst=33438
31. Aug 30 12:57:08.309: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose udp.noport
32. Aug 30 12:57:08.309: UDP src=42765, dst=33438
33. Aug 30 12:57:11.305: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, rcvd 0
34. Aug 30 12:57:11.305: UDP src=36551, dst=33439
35. Aug 30 12:57:11.305: IP: s=172.16.20.3 (local), d=172.16.10.1 (Serial0/0), len 56, sending
36. Aug 30 12:57:11.305: ICMP type=3, code=3
37. Aug 30 12:57:11.305: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose udp.noport
38. Aug 30 12:57:11.305: UDP src=36551, dst=33439
```

Since R3 is the final target of the original traceroute, it generates a unique set of messages that are related to an attempt to access an invalid UDP port number. This is indicated by lines 28, 32 and 38 “dispose.udp.noport”. Due to this error condition router R3 sends R1 an ICMP message using the type code combination of 3 and 3 which is translated to “destination unreachable/port unreachable”.

Generating a Combined Set of Debug Output on a Single Screen

In the two scenarios above, three different sets of debug output related to the same set of packets was generated on three routers. Even though the timestamps made it possible to match up the debug output from one router with the output of another, it is still a cumbersome activity.

The following technique combines the debug output from three different routers onto a single screen.

The technique involves initiating a series of telnet sessions so that the debug output of one session will be combined with the subsequent session.

The following demonstration is applied to the same three router configuration used on the previous scenarios. An assumption is made that all of the debugging tools used described in the previous scenarios are already enabled.

First, a telnet session is opened from router R3 to router R2. Once, the session is opened on router R2, the command “terminal monitor” is performed to allow the debug output to appear during this telnet session.

```
r3#telnet 172.16.20.2
Trying 172.16.20.2 ... Open
```

```
r2#term mon
```

A second telnet session is opened from router R2 to router R1. Once again, the “terminal monitor” command is performed to allow the debug output to appear during this telnet session:

```
r2#telnet 172.16.10.1
Trying 172.16.10.1 ... Open
```

```
r1#term mon
```

Finally, on router R1, the traceroute command is executed. In the example below, only a single traceroute probe packet is generated. See line 7 below.

```
1. r1#trace
2. Protocol [ip]:
3. Target IP address: 172.16.20.3
4. Source address:
5. Numeric display [n]:
6. Timeout in seconds [3]:
7. Probe count [3]: 1
8. Minimum Time to Live [1]:
9. Maximum Time to Live [30]:
10. Port Number [33434]:
11. Loose, Strict, Record, Timestamp, Verbose[none]:
12. Type escape sequence to abort.
13. Tracing the route to 172.16.20.3
14. 1 172.16.10.2 16 msec
15. 2 172.16.20.3 24 msec
```

The resulting output of this nested telnet technique is displayed below.

```
1. Aug 30 13:40:58.379: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, rcvd 0
2. Aug 30 13:40:58.379: UDP src=34748, dst=33435
3. Aug 30 13:40:58.379: IP: s=172.16.20.3 (local), d=172.16.10.1 (Serial0/0), len 56, sending --->r3
4. Aug 30 13:40:58.379: ICMP type=3, code=3
5. Aug 30 13:40:58.379: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose udp.noport
6. Aug 30 13:40:58.379: UDP src=34748, dst=33435
7. Aug 30 13:40:58.360: IP: s=172.16.10.2 (local), d=172.16.10.1 (Serial0/0), len 56, sending
8. Aug 30 13:40:58.360: ICMP type=11, code=0
9. Aug 30 13:40:58.360: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3, len 28, dispose ip.hopcount-----r2
```

```
10. Aug 30 13:40:58.360: UDP src=32813, dst=33434
11. Aug 30 13:40:58.372: IP: s=172.16.10.1 (Serial0/0), d=172.16.20.3 (Serial0/1), g=172.16.20.3, len 28, forward
12. Aug 30 13:40:58.372: UDP src=34748, dst=33435
13. Aug 30 13:40:58.350: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending
14. Aug 30 13:40:58.350: UDP src=32813, dst=33434
15. Aug 30 13:40:58.366: IP: s=172.16.10.1 (local), d=172.16.20.3 (Serial0/0), len 28, sending----->r1
16. Aug 30 13:40:58.366: UDP src=34748, dst=33435
17. Aug 30 13:40:58.390: IP: s=172.16.20.3 (Serial0/0), d=172.16.10.1 (Serial0/0), len 56, rcvd 3
18. Aug 30 13:40:58.390: ICMP type=3, code=3
```

Look at the other articles in the NetMasterClass Technical Library:

<http://www.netmasterclass.net/site/lib.php>

Download NetMasterClass DOiT Sample Scenario at:

<http://www.netmasterclass.net/site/pdf/DOiT-SAMPLE-SCENARIO.pdf>

Read more about NetMasterClass DOiT Workbook at:

<http://www.netmasterclass.com/site/articles/DOiT-Workbook-FAQ.pdf>

<http://www.netmasterclass.net/site/doi1.php>

If you have any question about this article please send them to <http://bbs.netmasterclass.com>