"Where the !?*! are the packets going?"

ITNOG 2024

Luca Sani

Senior R&D Software Engineer @Catchpoint

catchpoint

ITNOG

THE ITALIAN NETWORK OPERATORS GROUP

Traceroute

Traceroute is one of the most famous and long-lasting diagnostic tools in networking environment

First implementation by Van Jacobson in late 80s to answer the question:

"where the !?*! are the packets going" ?

Posted-Date: Tue, 20 Dec 88 05:13:28 PST
Received-Date: Tue, 20 Dec 88 05:14:46 PST
Received: from helios.ee.lbl.gov by venera.isi.edu (5.54/5.51)
id AA25560; Tue, 20 Dec 88 05:14:46 PST
Received: by helios.ee.lbl.gov (5.59/s2.2)
id AA03127; Tue, 20 Dec 88 05:13:30 PST
Message-Id: <8812201313.AA03127@helios.ee.lbl.gov>
To: ietf@venera.isi.edu, end2end-interest@venera.isi.edu
Subject: 4BSD routing diagnostic tool available for ftp
Date: Tue, 20 Dec 88 05:13:28 PST
From: Van Jacobson <van@helios.ee.lbl.gov>
Content-Length: 2373
X-Lines: 46
Status: RO

After a frustrating week of trying to figure out "where the !?*! are the packets going?", I cobbled up a program to trace out the route to a host. It works by sending a udp packet with a ttl of one & listening for an icmp "time exceeded" message. If it gets one, it prints the source address from the icmp message, then bumps the ttl by one & etc. (As usual, I didn't come up with this clever idea -- I heard Steve Deering mention it at an end-to-end task force meeting.)

Traceroute implementations

- Many traceroute implementations have been created on different OSes
- Over the years it became one of the most used tools in the Internet measurement and topology discovery fields (multipath, de-aliasing, NAT traversal, …)
    - Paris, Dublin, Pamplona traceroute…

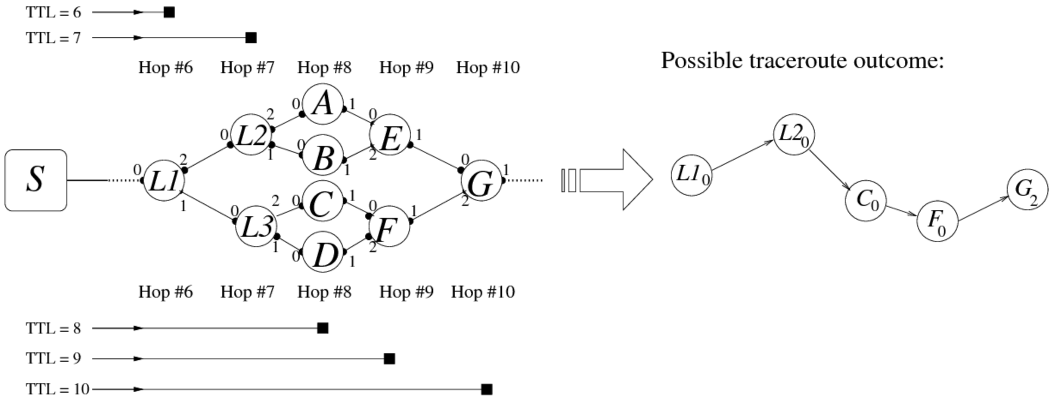


Fig. 1. Traceroute under load balancing

Augustin, Brice, Timur Friedman, and Renata Teixeira. "Multipath tracing with Paris traceroute." Workshop on End-to-End Monitoring Techniques and Services. IEEE, 2007.

Linux traceroute

- We leverage Dmitry Butskoy's "Linux traceroute"
  - Very fast
  - Open source
  - Easily extendible

Project Page Download Mail List

This is a new modern implementation of traceroute(8) utility for Linux systems.

It has replaced the old one in the majority of distributions now, including Fedora, RHEL, Debian, Mandriva, Gentoo, Ubuntu.

- During the years we enhanced this traceroute to include new monitor capabilities
- We hope these enhancements can be useful to the community

Pietrasanta Traceroute

"A noble town since 1841 and a city of art"

(and where our Italian office is located!)

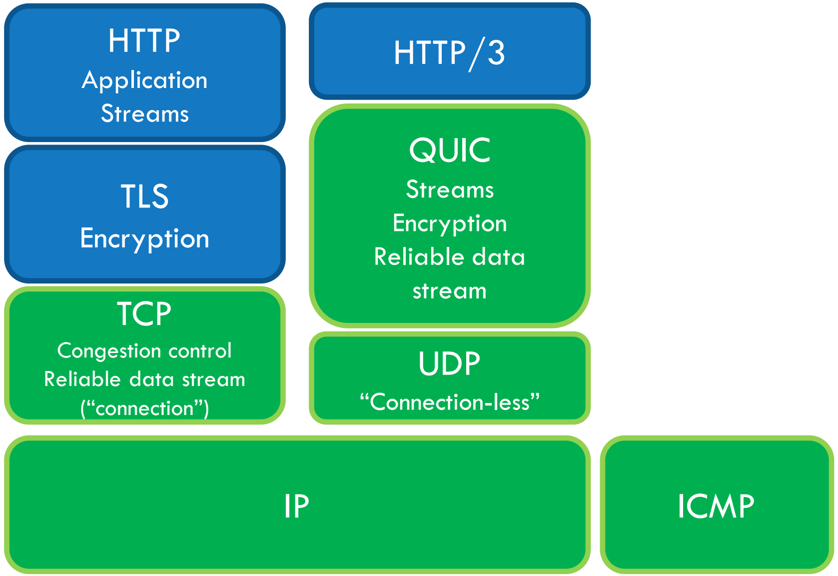
Pietrasanta Traceroute

- QUIC traceroute
- ECN bleaching detection
- Work in Azure environment
- TCP “In Session”
- … and many more

QUIC traceroute

QUIC

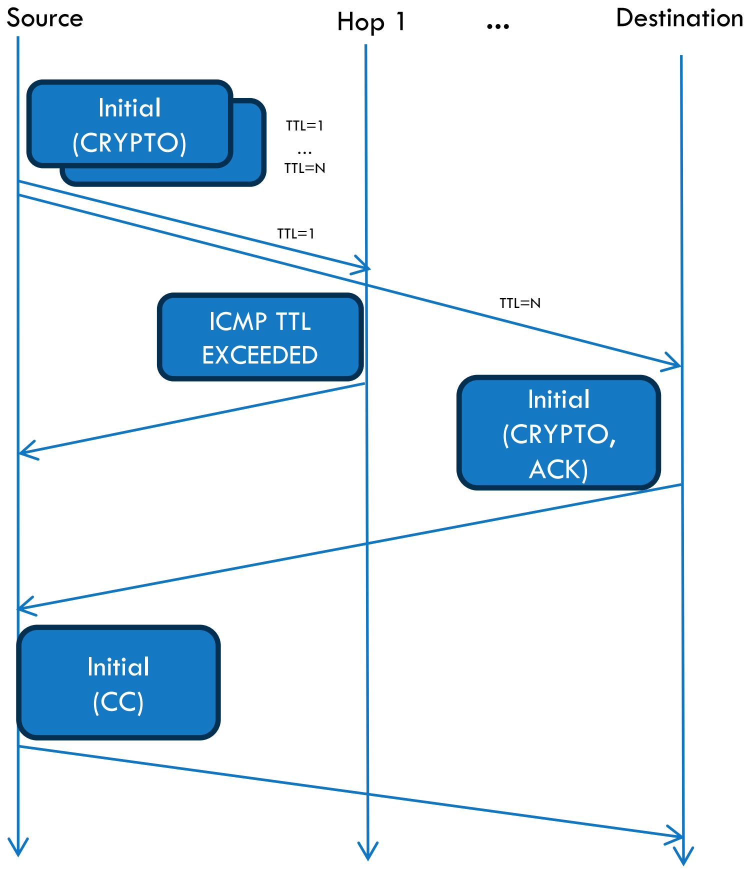
- QUIC is considered a transport layer protocol
  - More than just “UDP”
  - e.g., it is the transport layer of HTTP/3



QUIC assumes responsibility for the confidentiality and integrity protection of packets. For this it uses keys derived from a TLS handshake, but instead of carrying TLS records over QUIC (as with TCP), TLS handshake and alert messages are carried directly over the QUIC transport, which takes over the responsibilities of the TLS record layer.

RFC9001 - Using TLS to Secure QUIC

QUIC support



- Packets sent are QUIC compliant, so the header is protected and the payload (frames) are encrypted
  - We leverage openssl3
- Nice "side effects"
  - Check whether the path filters QUIC
  - Determine if the destination supports QUIC
  - Check whether ECN is supported
    - Set IP-ECN in probes

QUIC traceroute

- Like "TCP half open"
- Do a QUIC handshake then closes the session (if opened)
  - Send QUIC "Initial" packet
    - Include a CRYPTO frame with TLS "Server Hello"
  - Intermediate hops will return ICMP TTL Exceeded
  - Destination may return
    - QUIC packet
    - ICMP port unreachable (still good, dest reached)
    - Nothing (timedout)
  - Close the session if it is the case
    - Send QUIC Initial packet including a CONNECTION_CLOSE frame

ECN bleaching detection

ECN mechanism

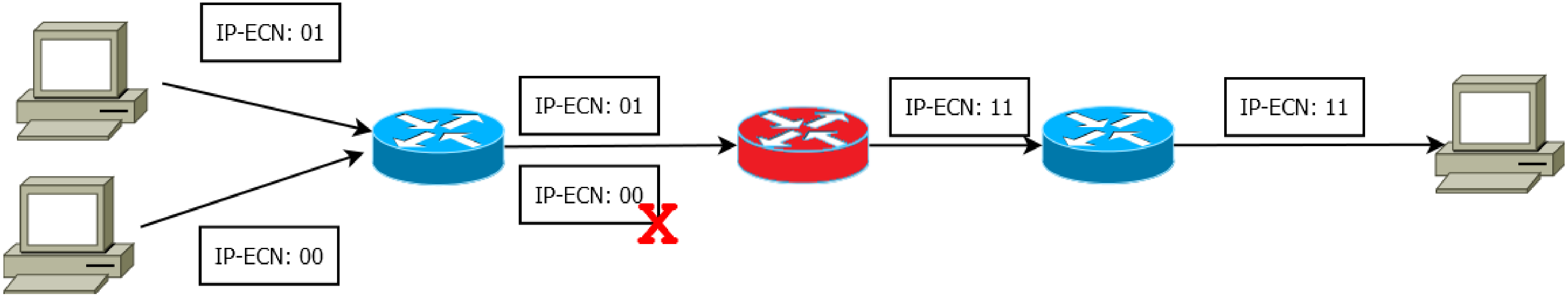
- ECN is a mechanism to signal that a packet experienced congestion (*The Addition of Explicit Congestion Notification to IP* - rfc3168, **2001**)
- When a packet experiences congestion is marked instead of dropped
- The destination signals this event to the source, which in turn adjusts the rate

```
Network Working Group                                  K. Ramakrishnan
Request for Comments: 3168                          TeraOptic Networks
Updates: 2474, 2401, 793                                      S. Floyd
Obsoletes: 2481                                                  ACIRI
Category: Standards Track                                     D. Black
                                                                   EMC
                                                        September 2001


      The Addition of Explicit Congestion Notification (ECN) to IP
```

ECN marking

- Two bits into the IP header
- The source declares that a packet should be treated with ECN by setting the IP-ECN fields either to 01 or 10
- When congestion happens, instead of dropping the packet the router sets the IP-ECN fields to 11 (CE - Congestion Experienced)



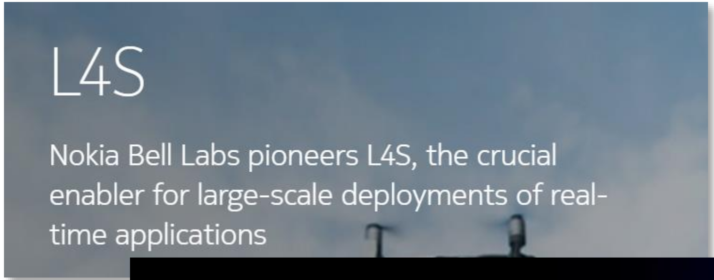
ECN feedback

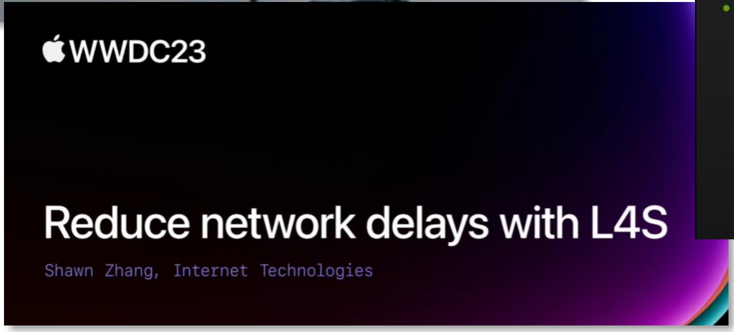
- A destination that receives a packet with IP-ECN = CE should report to the source this event
- The source should then adjust the rate
- The report is done at transport/application layer
  - Example: in TCP, this event can be reported using a dedicated TCP flag (ECE – ECN-Echo)

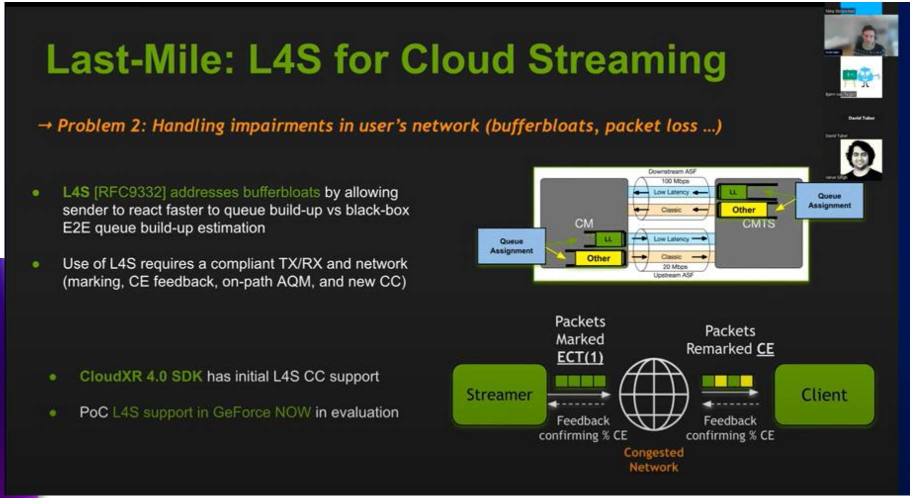
```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |               | C | E | U | A | P | R | S | F |
| Header Length |    Reserved   | W | C | R | C | S | S | Y | I |
|               |               | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

ECN and L4S

- Recently, ECN mechanism got renewed attention due to L4S (Low Latency, Low Loss, and Scalable Throughput – rfc9330, **2023**)
- L4S requires an ECN feedback more accurate wrt the “classic” 2001 version







More accurate ECN feedback

- TCP: More Accurate Explicit Congestion Notification (AccECN) Feedback in TCP (still a draft)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |           | A | C | E | U | A | P | R | S | F |
| Header Length | Reserved  | E | W | C | R | C | S | S | Y | I |
|               |           |   | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

- QUIC: Supported natively via ECN counters into the ACK frame

```
ECN Counts {
  ECT0 Count (i),
  ECT1 Count (i),
  ECN-CE Count (i),
}
```

ECN bleaching detection

- Intermediate hops can bleach/alter the value of ECN into the IP header (see for example: *The Benefits of Using Explicit Congestion Notification (ECN)* – rfc8087, 2017

- With Pietrasanta traceroute we can send probes with IP-ECN values different from zero and check hop by hop what was the IP-ECN value of the probe ***when it expired***

- We can also check whether the destination transport layer (either TCP or QUIC) supports more accurate ECN feedbacks
  - TCP stack need to be patched
  - Not all QUIC implementations report ECN counters

ECN detection: Some examples

```
[ bash ]$ sudo ./traceroute -nT -q 1 --ecn=1 -O acc-ecn,info 95.228.44.181
traceroute to 95.228.44.181(95.228.44.181), 30 hops max, 60 byte packets, ov
erall timeout not set
   1  172.21.82.1 <TOS:1,DSCP:0,ECN:1>  0.234 ms
   2  64.79.149.27 <TOS:1,DSCP:0,ECN:1>  1.374 ms
   3  64.79.139.17 <TOS:1,DSCP:0,ECN:1>  1.297 ms
   4  66.209.72.25 <TOS:1,DSCP:0,ECN:1>  1.358 ms
   5  *
   6  *
   7  4.68.39.58 <TOS:1,DSCP:0,ECN:1>  6.609 ms
   8  195.22.195.123 <TOS:1,DSCP:0,ECN:1>  160.604 ms
   9  195.22.205.117 <TOS:1,DSCP:0,ECN:1>  173.535 ms
  10  *
  11  *
  12  *
  13  *
  14  *
  15  95.228.44.181 <TOS:1,DSCP:0,ECN:1>  170.007 ms
  16  95.228.44.181 <syn,ack,ece,cwr>  172.391 ms
   Timedout: false
   Duration: 1713.448 ms
   DestinationReached: true
```

No bleaching, destination supports AccECN over TCP

```
[ bash ]$ sudo ./traceroute -nT -q 1 --ecn=1 -O acc-ecn,info 81.236.63.162
traceroute to 81.236.63.162(81.236.63.162), 30 hops max, 60 byte packets, ov
erall timeout not set
   1  172.21.82.1 <TOS:1,DSCP:0,ECN:1>  0.233 ms
   2  64.79.149.27 <TOS:1,DSCP:0,ECN:1>  1.270 ms
   3  64.79.139.17 <TOS:1,DSCP:0,ECN:1>  1.254 ms
   4  66.209.72.25 <TOS:1,DSCP:0,ECN:1>  1.271 ms
   5  66.209.64.124 <TOS:1,DSCP:0,ECN:1>  1.115 ms
   6  62.115.32.150 <TOS:1,DSCP:0,ECN:1>  1.052 ms
   7  62.115.132.119 <TOS:1,DSCP:0,ECN:1>  1.875 ms
   8  62.115.135.190 <TOS:1,DSCP:0,ECN:1>  6.789 ms
   9  62.115.137.38 <TOS:1,DSCP:0,ECN:1>  64.044 ms
  10  62.115.136.200 <TOS:1,DSCP:0,ECN:1>  69.195 ms
  11  80.91.254.90 <TOS:1,DSCP:0,ECN:1>  145.761 ms
  12  62.115.139.172 <TOS:1,DSCP:0,ECN:1>  155.524 ms
  13  62.115.140.217 <TOS:0,DSCP:0,ECN:0>  150.248 ms
  14  62.115.35.117 <TOS:0,DSCP:0,ECN:0>  150.434 ms
  15  81.228.89.186 <TOS:0,DSCP:0,ECN:0>  150.790 ms
  16  81.228.83.227 <TOS:0,DSCP:0,ECN:0>  150.816 ms
  17  90.228.166.164 <TOS:0,DSCP:0,ECN:0>  153.555 ms
  18  81.224.167.228 <TOS:0,DSCP:0,ECN:0>  153.135 ms
  19  *
  20  *
  21  81.236.63.162 <syn,ack>  150.907 ms
   Timedout: false
   Duration: 1522.420 ms
   DestinationReached: true
```

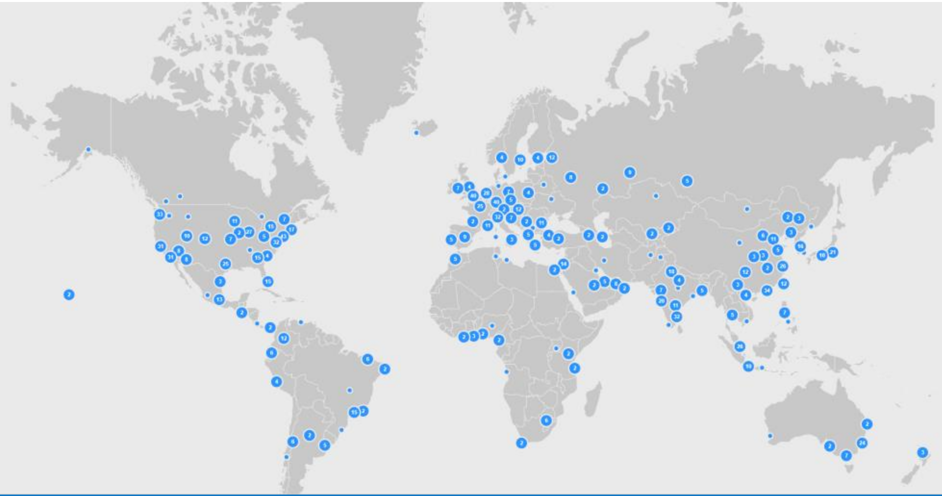
Bleaching happened

ECN bleaching research

- Run Pietrasanta traceroute from our vantage points deployed around the world to understand "how well" the network is ready to accommodate L4S
  - Where is the bleaching is happening?
  - Are there specific countries/ISPs/ASNs where it happens systematically?
- Stay tuned for more information!

City Overview

Source	% Bleach	Avg RTT	Avg Failing hop
Jefferson	100	54	2
Gilroy	100	62	3
San Diego	41	51	2
Las Vegas	27	48	7
Boston	25	50	4
Phoenix	23	52	7
New York	21	49	4
Seattle	19	55	4
Chicago	18	42	7
Washington	11	41	9
Austin	11	63	13
Denver	10	41	10
Honolulu	9	93	7
Kansas	9	35	7
Walla Walla	9	56	10



Work in Azure environment

Azure environment

- (Linux) VM with private IP
- Inbound ICMP packets are allowed

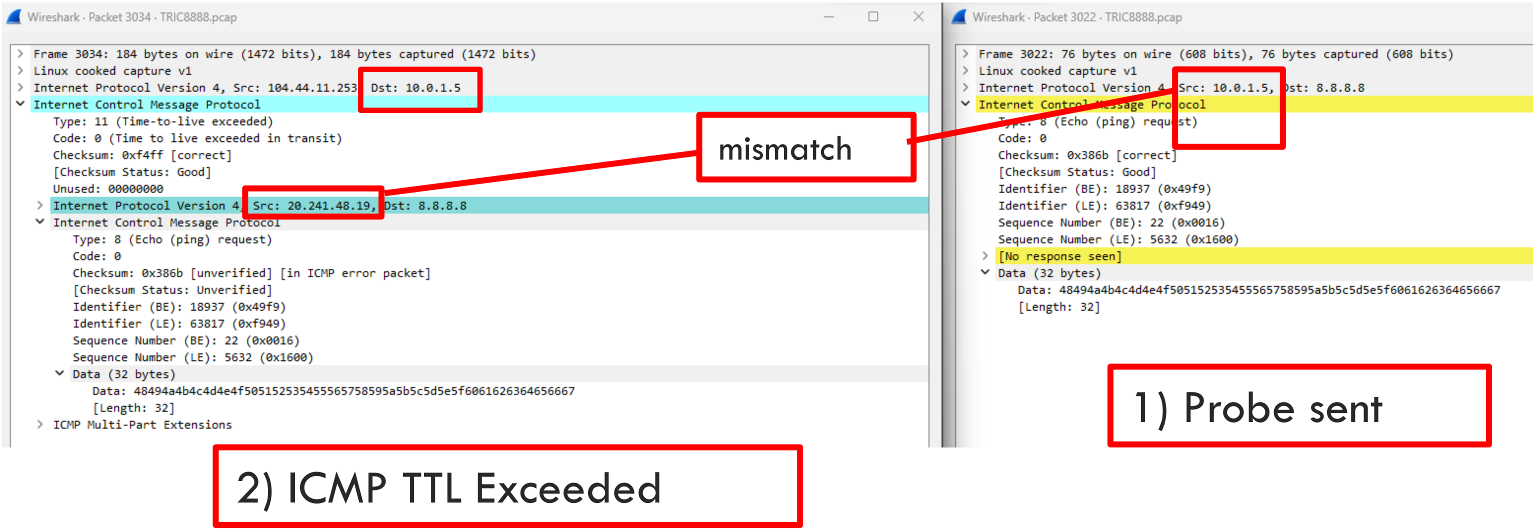
```
[redacted] sudo traceroute -I google.com

traceroute to google.com (142.251.46.174), 30 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  nuq04s44-in-f14.1e100.net (142.251.46.174)  2.040 ms  2.050 ms  1.784 ms
```

- Intermediate hops are all *
- This happens for all traceroute protocols

Azure environment

- This happens because the source IP of the original probe encapsulated into the ICMP TTL Exceeded is left with the node public IP
- Thus, the ICMP reply is discarded by the kernel (not by traceroute)



Work in Azure environment

- We enhanced traceroute to work in "loose match mode"
- Open an additional raw ICMP socket to receive all ICMP packets and do the "kernel checks" at user level...
  - … but do not check the source address of the encapsulated probe

```
traceroute --loose-match -I google.com

traceroute to google.com (142.251.46.174), 30 hops max, 60 byte packets, overall timeout not set
 1  * * * D=5.003980
 2  * * * D=5.003996
 3  * * * D=5.004010
 4  * * * D=5.004024
 5  * * * D=5.004039
 6  * * * D=5.030275
 7  ae31-0.sjc-96cbe-1b.ntwk.msn.net (104.44.238.247)  1.617 ms  1.617 ms  1.611 ms D=0.001641
 8  google.sjc-96cbe-1b.ntwk.msn.net (207.46.219.195)  1.927 ms  1.924 ms  1.919 ms D=0.001939
 9  142.251.69.83 (142.251.69.83)  4.132 ms  4.128 ms  4.124 ms D=0.004141
10  142.251.224.189 (142.251.224.189)  2.087 ms  2.082 ms  2.081 ms D=0.002101
11  nuq04s44-in-f14.1e100.net (142.251.46.174)  2.050 ms  2.046 ms  1.871 ms D=0.003395
```

TCP InSession

TCP "InSession"

- Classic TCP traceroute sends a different SYN for each hop
  - Different SYNs can take different paths
    - No consistency within a single traceroute
  - Many SYNs are sent per traceroute
    - Trigger firewall rules (SYN flood?)

- TCP InSession firstly opens a TCP session with the destination
- Then tracerouting is performed by sending 1-byte data packets within the session (with incremental TTL)

Checkout our blog for more information: https://www.catchpoint.com/blog/traceroute-insession-catchpoints-effort-towards-a-more-reliable-network-diagnostic-tool

And many more!

- Path MTU performance improvements
- Report ToS/DSCP hop by hop
- Report MSS when running in TCP mode
- Handle print in a separate thread (speed up)
- Overall timeout
- Compile and run on Alpine
- Avoid UDP standard filtering

Thank you!

- Feel free to check/use/ & contribute!
  https://github.com/catchpoint/Networking.traceroute/ (GPL!)
- And come by to meet us!
  - Pietrasanta is a nice town on Tuscany seaside…