Credits

• Developers (Harvey Mudd College)
  - Ishaan Gandhi
  - Andreas Roeseler
  - Spencer Lang
  - Nick Ludwig

• Mentors
  - Ron Bonica (Juniper Network)
  - Zach Dodds (Harvey Mudd College)
THE IP OAM TOOLKIT

• PING and TRACEROUTE are the most commonly use tools in the IP OAM Toolkit

• What do they do?
• How do they work?
• What can’t they do?

• How can they be enhanced?
PING
WHAT DOES IT DO?

• Test the liveliness of a *reachable* interface
• Test the liveliness of a *reachable* node
  – By testing the liveliness of one of its *reachable* interfaces
HOW DOES IT WORK?

• A probing node sends an ICMP Echo to a probed interface
• The probed interface sends an ICMP Echo Reply to the probing node

• Nerd Notes
  – The ICMP Echo may enter the probed node through any of its interfaces
  – The ICMP Echo reply message may leave the probed node through any of its interfaces
  – There is no guarantee that that either ICMP message traverses the probed interface
WHAT CAN’T IT DO?

• Test the liveness of *less-than-reachable* interface

• Examples of *less-than-reachable* interfaces
  – IPv4 unnumbered
  – IPv6 unicast with narrowly scoped address (link-local, ULA)
  – Any interface to which the probing node lacks a route
HOW CAN IT BE ENHANCED? [RFC 8335]

• A probing node sends an ICMP Extended Echo Request to a proxy interface
  – ICMP Extended Echo Request identifies the probed interface

• The proxy interface sends an Extended ICMP Echo Reply to the probing node
  – The extended ICMP Echo Reply reports the liveliness of the probed interface
How is interface 3?

Interface 3 is up.
RFC 8335 NERD NOTES

• The proxy interface can be different from the probed interface
  – And is different in most cases
• The proxy interface must be **reachable** from the probing node
• The proxy interface must reside on one of the following
  – The same node as the probed interface
  – A node that is directly connected to the probed interface
Example Usage

```
[aroeseler@aroeseler-ly545 ~]$ git/iputils/builddir/pcap ping
PING 10.0.1.28 (10.0.1.28) 56(84) bytes of data.
64 bytes from 10.0.1.28: icmp_seq=1 ttl=64 time=60.5 ms
^C
10.0.1.28 ping statistics:
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 60.456/60.456/60.456/0.000 ms
```

```
[aroeseler@aroeseler-ly545 ~]$ git/iputils/builddir/pcap
```
SECURITY CONSIDERATIONS

• Not enabled by default
• Accessible from specified source addresses only
IMPLEMENTATIONS

• JUNOS 20.3R1
• LINUX Kernel(5.13)
• LINUX IP Utils Ping (in progress)
• Wireshark (3.5)
• TCPDUMP (in progress)
TRACEROUTE
WHAT DOES IT DO?

• Elicit feedback from each **node** on the delivery path between a probing interface and a destination interface

• Identify **nodes** along the delivery path
HOW DOES IT WORK?

• A probing node sends a series of UDP packets to a destination interface
  – Sets the TTL to 1 on the first packet, so that it expires on the first node along the delivery path
  – Increments the TTL on each subsequent, so that it expires on the next node along the delivery path
• When a packet expires on a node along that delivery path, that node sends a ICMP Time Expired message to the probing node
NERD NOTES

• Regarding the UDP probe messages
  – By default, on the first packet, the probing node sets the UDP destination port to 33434
  – Increments UDP destination port on each subsequent packet

• Regarding the ICMP Time Expired message
  – The source address may not identify the interface upon which the UDP probe message arrived
  • IPv4: Identifies the interface through which the ICMP message left the reporting node
  • IPv6: It’s complicated. See RFC 6724.
WHAT CAN’T IT DO?

• Identify *interfaces* along the delivery path
HOW CAN IT BE ENHANCED? [RFC 5837]

- UDP Probe message is unchanged
- ICMP Time Expired message can contain extensions that identify
  - Interface upon which the UDP probe message arrived
  - Interface through which the message would have been routed had the TTL not expired
  - Attributes of those interfaces (name, IP address, MTU)
Example Usage

[ishaangandi@Ishaans-MacBook-Pro-5.local ~]$ traceroute -v cs.hmc.edu
traceroute to cs.hmc.edu (134.173.42.100), 64 hops max, 52 byte packets
1  192.168.0.1 (192.168.0.1)  4.885 ms  3.098 ms  3.378 ms
2  10.79.240.1 (10.79.240.1)  16.694 ms  9.498 ms  15.203 ms
3  100.127.5.90 (100.127.5.90)  17.656 ms  12.379 ms  15.827 ms
4  100.120.102.34 (100.120.102.34)  13.698 ms  15.453 ms  10.932 ms
5  68.1.4.252 (68.1.4.252)  15.853 ms  19.949 ms  28.248 ms

Arrival interface:
  name: en2
  address: 38.1.5.49
  MTU: 1500

6  100ge16-2.core1.lax1.he.net (216.218.224.117)  17.620 ms  18.293 ms  17.679 ms
7  100ge14-1.core1.lax2.he.net (72.52.92.122)  17.727 ms  18.212 ms  22.573 ms
8  65.19.156.114 (65.19.156.114)  19.951 ms
  216.218.223.26 (216.218.223.26)  21.678 ms  20.780 ms
9  130.152.184.99 (130.152.184.99)  23.593 ms  17.517 ms
  130.152.184.162 (130.152.184.162)  19.027 ms
10  hmc-a.router.claremont.edu (134.173.253.23)  24.651 ms
130.152.184.162 (130.152.184.162)  22.136 ms  19.224 ms
11  *  hmc-a.router.claremont.edu (134.173.253.23)  23.770 ms
12  *  knuth.cs.hmc.edu (134.173.42.100)  18.032 ms  19.201 ms

Arrival interface:
  if index: 2
  address: 130.122.104.62
  MTU: 1500
SECURITY CONSIDERATIONS

• Not enabled by default

• Accessible from specified source addresses only
IMPLEMENTATIONS

• JUNOS (in progress, EX first)
• LINUX Kernel (in progress)
• LINUX IP Utils Traceroute (in progress)
• Wireshark (3.5)
THANK YOU

Ron Bonica, Ishaan Gandhi, Spencer Lang, Nick Ludwig, Andreas Roeseler