A Practical Guide to (Correctly) Troubleshooting with Traceroute

Created by Richard A Steenbergen <ras@turkbergen.com>
Presented by Dani Roisman <dani@roisman.com>
      at NANOG on the Road Waterloo 13 September 2016

Introduction

- Troubleshooting problems on the Internet?
    - The number one go-to tool is "traceroute"
        - Every OS comes with a traceroute tool of some kind.
        - There are thousands of websites which can run a traceroute.
        - There are dozens of "visual traceroute" tools available, both commercially and free.
    - And it seems like such a simple tool to use too
        - Type in the target IP address, and it shows me every router hop along the way, along with a latency measurement.
        - And where the traceroute stops, or where the latency jumps up a lot, that's where the problem is, right?
        - How could this possibly go wrong?
    - Unfortunately, it never quite works this way in practice.

Introduction

- So what can go wrong with traceroute?
  - Most modern networks are actually pretty well run
    - Simple issues like congestion or routing loops are becoming a smaller percentage of the total network issues encountered
    - The remaining issues are becoming ever more complex, so that often a naïve traceroute interpretation is practically useless
  - Very few people are skilled at interpreting traceroute
    - Traceroute output *looks* relatively straight-forward to read and use
    - This tends to give people the impression that they are qualified to interpret it and locate the fault, even when they are not
    - In reality most ISP NOCs, and even most mid-level IP engineering staff, are not qualified to correctly interpret a complex traceroute
    - The resulting swarm of bogus complaints makes it very difficult for knowledgeable people to get traceroute issues looked at and fixed

Example Traceroute Output

- Implementations vary, but all traceroute looks a little something like this:

Hop #	Router (DNS)	Router (IP)	Latency Measurements

In this example, each router hop has three latency probes:

```
traceroute to 4.2.2.2 (4.2.2.2), 64 hops max, 52 byte packets
 1  ge0-34.aggrFZ155-2.ord6.us.scnet.net (204.93.176.73)  1.673 ms  1.187 ms  2.847 ms
 2  ge-9-2-9.ar2.ord6.us.scnet.net (75.102.0.69)  0.317 ms  0.163 ms  0.155 ms
 3  72.ae3.cr2.ord6.us.scnet.net (204.93.204.158)  0.203 ms
 4  as3549.xe-0-2-1.cr2.ord6.us.scnet.net (204.93.144.30)  0.929 ms  0.898 ms  0.893 ms
 5  ae9.503.edge3.Chicago.Level3.net (4.68.62.253)  1.005 ms  1.028 ms  1.023 ms
 6  vlan52.ebr2.Chicago2.Level3.net (4.69.138.190)  1.194 ms
 7  4.69.158.237 (4.69.158.237)  1.172 ms
 8  b.resolvers.Level3.net (4.2.2.2)  1.169 ms  1.182 ms  1.178 ms
```

Traceroute at the Packet Level

The traceroute operation is comprised of the following steps:

1. Launch a probe packet towards the DST, with an initial TTL of 1.

2. Each router that forwards the packet decrements the TTL value by 1.

3. When the TTL hits 0, the router returns an ICMP TTL Exceed to SRC.

4. SRC receives the ICMP, calculates a time difference, and displays a "hop".

5. Go back to step 1, but increment the TTL of the probe packet by 1.

6. Rinse and repeat, until DST receives probe and returns ICMP Unreachable.

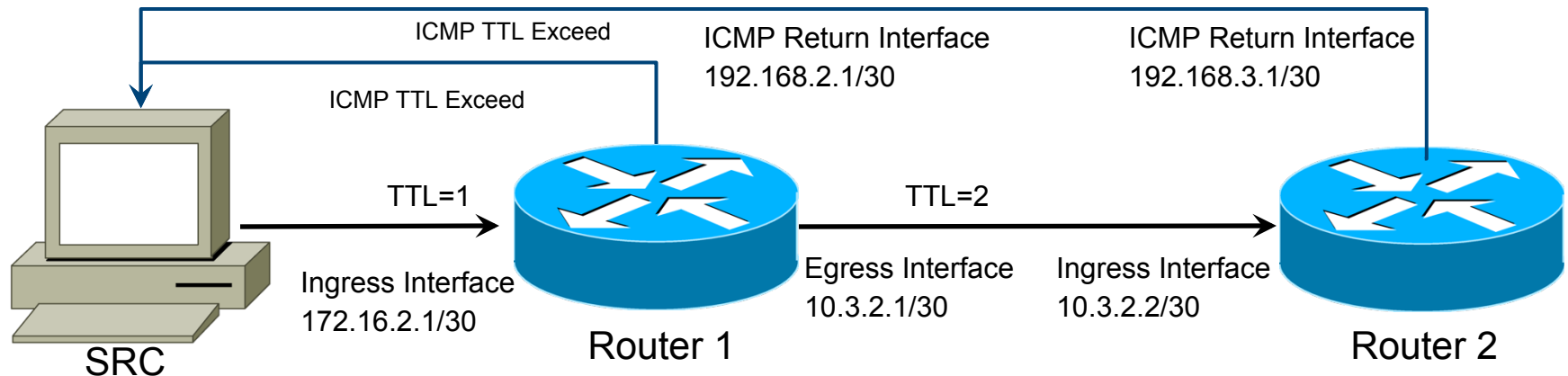7. When SRC receives an ICMP Unreachable, the traceroute ends.
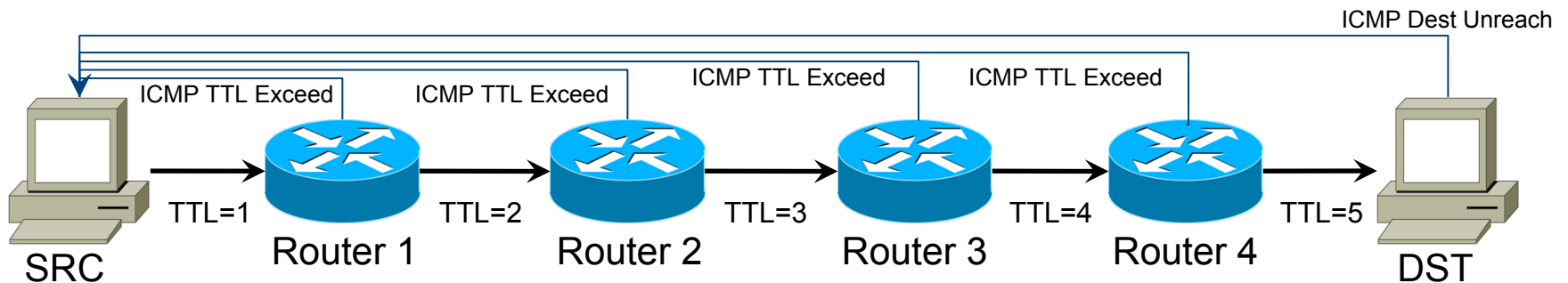
Traceroute and Multiple Probes

- Most implementations send multiple probes per "hop"
  - The default for the vast majority of traceroute implementations is 3.
  - That is, 3 probe packets are launched before each TTL increment.
  - This results in 3 latency measurements per "hop".
- Each probe packet uses a unique key to distinguish itself.
  - Most implementations use UDP packets with incrementing dest ports.
  - But ICMP or TCP can be used (e.g. Windows tracert.exe uses ICMP).
- Each probe packet is a **completely** independent trial.
  - Each packet MAY be forwarded down a completely different path.
  - This MAY be visible to the end-user, as multiple IP's for each hop.
  - But it can also be completely invisible too.

Traceroute and Latency Calculation

- The traceroute "latency" calculation is very simple:
  - Timestamp when the probe packet is launched.
  - Timestamp when the return ICMP packet is received.
  - The difference, $return - $launch, is the round-trip time.

- Routers along the path do NOT do any time processing.
  - But they do return some of the original probe payload in the ICMP.
  - Some implementations will encode the original launch timestamp (relative to their own clocks) in the probe packet to reduce state.

- Thus the reported latency is actually the sum of:
  - The time taken to forward the packet to the displayed router hop.
  - The time taken for the router to generate an ICMP response packet.
  - The time taken for that ICMP packet to reach the sender.

Traceroute – What Hops Are You Seeing?

ICMP TTL Exceed

ICMP Return Interface
192.168.2.1/30

ICMP Return Interface
192.168.3.1/30

ICMP TTL Exceed

TTL=1

TTL=2

Ingress Interface
172.16.2.1/30

Egress Interface
10.3.2.1/30

Ingress Interface
10.3.2.2/30

SRC

Router 1

Router 2

- Probe packet enters Router 1, TTL hits 0, packet is dropped
  - Router 1 generates an ICMP TTL Exceed message towards SRC.
  - The source address of this ICMP determines the traceroute hop IP.
- By *convention*, the ICMP is sourced from the ingress interface.
  - The above traceroute will read: 172.16.2.1 10.3.2.2
- Random factoid: This behavior is actually non-standard
  - RFC1812 says the ICMP source MUST be from the egress interface.
  - If obeyed, this would prevent traceroute from working properly.

How to Interpret DNS in a Traceroute

Interpreting DNS in a Traceroute

- One does not read a Traceroute by IP alone.
  - Most operators are kind enough to embed some DNS info.
  - Correctly interpreting this DNS is one of the most important aspects of successfully reading a traceroute.
- Important information you can discover includes:
  - Geographic Locations
  - Interface Types and Capacities
  - Router Type and Roles
  - Network Boundaries and Relationships

Interpreting Traceroute - Location

- Knowing the geographical location of the routers is an important first step to understanding an issue.
  - To identify incorrect/suboptimal routing.
  - To help you understand network interconnections.
  - And even to know when there isn't a problem at all, i.e. *knowing when high latency is justified and when it isn't.*
- The most commonly used location identifiers are:
  - IATA or ICAO Airport Codes
  - CLLI Codes
  - Attempts to abbreviate based on the city name.
  - And sometimes you just have to take a guess.

Location Identifiers – IATA Airport Codes

- IATA/ICAO Airport Codes
  - Good International coverage of most large cities.
  - Most common in networks with a few big POPs.
  - Examples:
    - Santo Domingo = SDQ
    - San Jose California = SJC
  - Sometimes represented by pseudo-airport codes
    - Especially where multiple airports serve a region
    - Or where the airport code is non-intuitive
      - New York, NY is served by JFK, LGA, and EWR airports.
        - But is frequently written as NYC.
      - Northern VA is served by IAD, Washington DC by DCA.
        - But both may be written as WDC.

Location Identifiers – CLLI Codes

- Common Language Location Identifier
  - Full codes maintained (and sold) by Telecordia.
  - Most commonly used by Telephone Companies
    - Example: **HSTNTXMOCG0**
  - In a non-Telco role, may only use the city/state identifiers
    - Examples:
      - HSTNTX = Houston Texas
      - ASBNVA = Ashburn Virginia
  - Well defined standard covering almost all US/CA cities
    - Commonly seen in networks with a larger number of POPs.
    - Not an actual standard outside of North America
      - Some providers fudge these, e.g. AMSTNL = Amsterdam NL

Location Identifiers – Arbitrary Values

- But sometimes, people just make stuff up.
  - Chicago IL
    - Common Airport Code: ORD (O'Hare) or MDW (Midway)
    - CLLI Code: CHCGIL
    - Example Arbitrary Code: CHI
  - Toronto ON
    - Airport Code: YYZ or YTC
    - CLLI Code: TOROON
    - Example Arbitrary Code: TOR
  - Frequently based on the good intentions of making thing readable in plain English, even though these may not follow any known standards.

Common Locations – US Major Cities

Location Name	Airport Codes	CLLI Code	Some Other Codes
Ashburn VA	IAD, DCA	ASBNVA	WDC, WAS
Atlanta GA	ATL	ATLNGA	
Chicago IL	ORD, MDW	CHCGIL	CHI
Dallas TX	DFW, DAL	DLLSTX	DLS
Houston TX	IAH, HOU	HSTNTX	
Los Angeles CA	LAX	LSANCA	LA
Miami FL	MIA	MIAMFL	
Newark NJ	EWR	NWRKNJ	NEW, NWK
New York NY	JFK, LGA	NYCMNY	NYC, NYM
San Jose CA	SJC	SNJSCA	SJO, SV, SF
Palo Alto CA	PAO	PLALCA	PAIX, PA
Seattle CA	SEA	STTLWA	

Common Locations – Non-US Major Cities

Location Name	Airport Codes	CLLI Code (*)	Some Other Codes
Amsterdam NL	AMS	AMSTNL	
Frankfurt GE	FRA	FRNKGE	
Hong Kong HK	HKG	NEWTHK	HK
London UK	LHR	LONDEN	LON
Madrid SP	MAD	MDRDSP	
Montreal CA	YUL	MTRLPQ	MTL
Paris FR	CDG	PARSFR	PAR
Singapore SG	SIN	SNGPSI	SNG
Seoul KR	GMP, ICN	SEOLKO	SEL
Sydney AU	SYD	SYDNAU	
Tokyo JP	NRT	TOKYJP	TYO, TKO
Toronto CA	YYZ, YTC	TOROON	TOR

Interpreting DNS – Interface Types

- Most networks will try to put interface info in DNS
  - Often just to help them troubleshoot their own networks.
  - This may not always be up to date though.
    - Many large networks use automatically generated DNS.
    - Others are surprisingly bad at keeping DNS data updated.
  - Can potentially help you identify the type of interface
    - As well as capacity, and maybe even the make/model of router.
- Examples:
  - xe-11-1-0.edge1.NewYork1.Level3.net
    - XE-#/#/# is Juniper 10GE port. The device has at least 12 slots.
    - It's at least a 40G/slot router since it has a 10GE PIC in slot 1.
    - It must be Juniper MX960, no other device could fit this profile.

Common Interface Naming Conventions

Interface Type	Cisco IOS	Cisco IOS XR	Juniper
Fast Ethernet	Fa#/#		fe-#/#/#
Gigabit Ethernet	Gi#/#	Gi#/#/#/#	ge-#/#/#
10 Gigabit Ethernet	Te#/#	Te#/#/#/#	xe-#/#/#
SONET	Pos#/#	POS#/#/#/#	so-#/#/#
T1	Se#/#		t1-#/#/#
T3			t3-#/#/#
Ethernet Bundle	Po# / Port-channel#	BE####	ae#
SONET Bundle	PosCh#	BS####	as#
Tunnel	Tu#	TT# or TI#	ip-#/#/# or gr-#/#/#
ATM	ATM#/#	AT#/#/#/#	at-#/#/#
Vlan	Vl###	Gi#/#/#/#.###	ge-#-#-#.###

Interpreting DNS – Router Types/Roles

- Knowing the role of a router can be useful
  - But every network is different, and uses different naming conventions.
  - And just to be extra confusion, they don't always follow their own naming rules.

- But generally speaking, you can guess the context and get a basic understanding of the roles.
  - Core routers – CR, Core, GBR, BB, CCR, EBR
  - Peering routers – BR, Border, Edge, IR, IGR, Peer
  - Customer routers – AR, Aggr, Cust, CAR, HSA, GW

Network Boundaries and Relationships

- Identifying Network Boundaries is important.
  - These tend to be where routing policy changes occur.
    - For example, different return paths based on Local Preference.
  - These also tend to be areas where capacity and routing are the most difficult, and thus likely to be problem spots.
- Identifying the relationship can be helpful too
  - Typically: a) Transit Provider, b) Peer, or c) Customer.
  - Many networks will try to indicate demarcs in their DNS
    - Examples:
      - Clear names like networkname.customer.alter.net
      - Or always landing customers on routers named "gw"

Network Boundaries and Relationships

- It's easy to spot where the DNS changes
  - 4 te1-2-10g.ar3.DCA3.gblx.net (67.17.108.146)
  - 5 sl-st21-ash-8-0-0.sprintlink.net (144.232.18.65)
- Or, look for "remote party" name in the DNS
  - 4 po2-20G.ar5.DCA3.gblx.net (67.16.133.90)
  - 5 cogent-1.ar5.DCA3.gblx.net (64.212.107.90)
  - Common where one side controls the /30 DNS, and the other side doesn't provide interface information.
- For more info, look at the other side of the /30
  - > nslookup 64.212.107.89
  - Result: te2-3-10GE.ar5.DCA3.gblx.net

Understanding Network Latency

Understanding Network Latency

There are 3 primary causes of network induced latency:

1. Serialization Delay
   - The delay caused by the encoding of a data as packets move across the network.

2. Queuing Delay
   - The delay caused by the router/switch buffering the packet while waiting for an opportunity to transmit it.

3. Propagation Delay
   - The delay caused while the electronic/photonic signal is "on the wire", propagating from source to destination.

Latency - Serialization Delay

- In networking, "serialization" is the process of encoding chunks of data (packets) onto the wire for transmission across the network.
  - And the "faster" an interface is, the quicker this process will occur.
    - The math is simple, just take the packet size and divide by the link speed.
    - For example, to send a 1500 byte packet over a 1 Mbps link:
      - First convert the units: 1 Mb/s = 1000 Kb/s = 125 KB/s = 125000 bytes/sec
      - 1500 bytes / 125000 bytes/sec = 0.012 seconds = *12 ms of delay*.
- So how does this cause a "Serialization Delay"?
  - Remember that packets move through the network as an atomic unit.
    - You can't transmit "half a packet" and expect to finish the other half later.
    - Nor can you start transmitting the packet until you've finished receiving it.
      - Some low-latency ("cut-through") switches do so, but this is extremely rare in routers.
- So every time a packet-aware device (a router or switch) touches a packet, a new serialization delay occurs.

Latency - Serialization Delay

- Fortunately this is less of an issue on modern high speed networks.
  - Interface speeds have increased by many orders of magnitude over the years, while packet sizes have stayed mostly the same.
  - But this can still be a noticeable cause of latency on low speed links.

- *Side note: This also makes an interesting counter-argument to the widespread deployment of Ethernet "jumbo frames".*

Interface Speed	Packet Size	Serialization Delay
1 Mbps	1500 bytes	12ms
10 Mbps	1500 bytes	1.2ms
100 Mbps	1500 bytes	0.12ms
1 Gbps	1500 bytes	0.012ms
10 Gbps	1500 bytes	0.0012ms
100 Gbps	1500 bytes	0.00012ms

Latency – Queuing Delay

- ## What is a "queuing delay"?
  - "Queuing" is when a router/switch holds a packet in its memory, while waiting for an opportunity to transmit the packet via a desired interface.
  - Every moment that the router spends holding onto the packet without transmitting it, the overall latency of the packet delivery is increasing.

- ## So why do we need queuing in the first place?
  - First, a quick word about the concept of "utilization".
    - A 10 Gbps port doing 5 Gbps is commonly said to be "50% utilized".
    - This is actually not true. At any given instant, the interface can only ever be in a state of transmitting (100% utilized) or not transmitting (0% utilized).
    - In the above example, what you actually mean is the "average utilization over some time period" is 50%, for example "over 1 second".
  - ***SOME*** queuing is necessary for a router to function correctly.
    - Just ask anyone who has bought an under-buffered switch.

Latency – Queuing Delay

- So when is queuing a good thing?
  - Queuing is especially important when:
    - You have mismatched interface speeds.
      - Especially higher speed interfaces sending to lower speed interfaces.
      - As discussed under serialization, packets will arrive "faster" on the higher speed interface and then need to be buffered, even at low bandwidths.
    - You have a lot of talkers sending traffic to a few receivers.
      - E.g. 40x GigE ports sending traffic to 4x 10GigE Uplink ports.
  - Technically speaking, queuing *always* increases thruput.
    - The longer you hold on to the packet, the more opportunities you will have to find a slot where you can transmit it rather than drop it.
  - The question becomes, how long should you hold it?
    - An extra 5ms to get from 60% to 85% thruput? Probably worth it.
    - But in reality, it depends on your application.

Latency – Queuing Delay

- ## So when is queuing a BAD thing?
  - As an interface becomes more and more full, the amount of time necessary to find an open slot increases.
    - You may still be able to successfully delivery every packet, but you may have to hold on to them for thousands of milliseconds to do it.
    - This is probably not what your application wants, especially latency or jitter sensitive applications like voice and video.
  - Beyond a certain point, queuing becomes counter-productive.
    - Getting from 98% to 99% utilization may be pointless, if it increases the latency on almost every packet by 5000ms to do it.
    - Typically the point where most users start to notice serious problems with normal "internet mix" traffic is around 95% utilization (over 1 sec).
  - Most routers have "bad" defaults around interface queues too.
    - Search for the term "buffer bloat" for more information.

Latency – Propagation Delay

- Propagation delay is the "time spent on the wire".
  - The speed of light in a vacuum is *a mere* 300,000 km/sec
  - Over long distances, this can cause significant latency
- Some example math for propagation over fiber:
  - Fiber is made of glass, which has a refractive index of ~1.48
  - 1/1.48 = ~0.67c, so it travels through fiber at ~200,000 km/sec
  - 200,000 km/sec = 200km (or 125 miles) per millisecond
  - Divide by 2 to account for round-trip time (RTT) measurements
- Example:
  - A round-trip around the world at the equator, via a perfectly straight fiber route, would take ~400ms due solely to speed-of-light propagation delays.

Identifying the Latency Affecting You

- ## So, how do you determine if latency is normal?
  - ### Use location identifiers to determine geographical data.
  - ### See if the latency fits with the expected propagation delay.

- ## For example:

      3 xe-3-0-0.cr1.nyc3.us.nlayer.net (69.22.142.74)   6.570ms
      4 xe-0-0-0.cr1.lhr1.uk.nlayer.net (69.22.142.10)   74.144ms
      New York NY to London UK in 67.6ms? 4200 miles? Sounds right.

- ## Another example:

      5 cr2.wswdc.ip.att.net (12.122.3.38) [MPLS: Label 17221 Exp 0] 8 msec 8 msec 8 msec
      6 tbr2.wswdc.ip.att.net (12.122.16.102) [MPLS: Label 32760 Exp 0] 8 msec 8 msec 8 msec
      7 ggr3.wswdc.ip.att.net (12.122.80.69) 8 msec 8 msec 8 msec
      8 192.205.34.106 [AS 7018] 228 msec 228 msec 228 msec
      9 te1-4.mpd01.iad01.atlas.cogentco.com (154.54.3.222) [AS 174] 228 msec 228 msec 228 msec
      Washington DC to Washington DC in 220ms? Nope!

Prioritization and Rate Limiting

Prioritization and Rate-Limiting

- Remember, traceroute latency is the sum of:
    1. The time required for the probe packet to reach the router hop
    2. The time required for the router to drop the probe packet, and generate an ICMP TTL Exceed back to the original SRC
    3. The time required for that ICMP TTL Exceed to reach the SRC

- #1 and #3 at least come from real network characteristics
    - But #2 is an artificially imposed constraint which affects ONLY traceroute packets, not real network traffic

- A wide variety of conditions can cause routers to:
    - Not send the ICMP TTL Exceed packet (artificial loss)
    - Be slow in the generation of the ICMP (artificial latency)

Understanding "To It" vs. "Through It"

- Modern routers have distinct processing paths:
  - Data Plane → Packets forwarded *through* the router
    - Fast Path: hardware based forwarding of ordinary packets
      - Example: Almost every packet in normal Internet traffic.
    - Slow Path: software based handling of "exception" packets
      - Example: IP Options, *ICMP Generation*, logging, etc.
  - Control Plane → Packets being forwarded *to* the router
    - Example: BGP, ISIS/OSPF, SNMP, CLI access (telnet/ssh), ping, or any packets sent directly to a local IP address on the router.

- Router CPUs tend to be relatively underpowered
  - A 320-640+ Gbps router may only have a 600MHz CPU

- ICMP Generation is *NOT* a priority for the router.

The Infamous BGP Scanner

- On some platforms, the slow-path data plane and the control-plane share the same resources.

  - And often don't have the best software for schedulers.
  - As a result, control-plane activity such as BGP churn or heavy CLI use can consume CPU and slow the generation of ICMP TTL Exceeds.
  - This results in random "spikes" in traceroute latency, which is often misinterpreted as a network issue.

- One infamous example of this is the "BGP Scanner" process, which runs every 60 seconds on many Cisco IOS devices.

Rate Limited ICMP Generation

- ICMP generation is a slow-path data plane operation
  - There are **many** other reasons for ICMP generation besides TTL Exceed generation, and the process can be complex.
  - As a result, no commercial router has a "Traceroute ASIC".
  - A general purpose CPU is used to generate the ICMP.

- These CPUs also handle other slow-path functions.
  - If left unchecked, a routing loop or TTL=0 Denial of Service attack could bring down the other functions of the router.
  - As a result, most routers will rate-limit ICMP generation.
  - These rate-limits tend to vary wildly by vendor, platform, and software revision, and are often not configurable or logged!
    - A few users running MTR against a network can easily hit these!

Spotting The Cosmetic Loss/Latency

- If there is an *actual* forwarding issue, the loss or latency will persist across *ALL* future hops as well.

- Example (not a real issue in hop 2):

```
1 ae3.cr2.iad1.us.nlayer.net  0.275 ms  0.264 ms  0.137 ms
2 xe-1-2-0.cr1.ord1.us.nlayer.net  18.271 ms  128.257 ms  68.001 ms
3 tge2-1.ar1.slc1.us.nlayer.net 53.373 ms  53.213 ms  53.227 ms
```

- Latency spikes in the middle of a traceroute mean absolutely *nothing* if they do not continue forward.
  - At worst it could be the result of an asymmetric path.
  - But more often than not, this is an indication of an artificial rate-limit or prioritization issue.
    - Try a non-TTL expiring method like "ping" to confirm the behavior.

Asymmetric Paths

Asymmetric Paths

- Traceroute *shows* you the forward path *only*
  - But remember, the latency values also include the time it takes for the TTL Exceed reply packet to come back.
  - The reverse path is a significant component of the latency values, but is *completely invisible* to the user.
- Not only is the reverse path hidden, but it can be completely different at every hop in the forward path.
  - The *only* way to confidently analyze a traceroute is to have traceroutes in **BOTH DIRECTIONS!**
  - And even then, it can't catch ALL of the potential asymmetric paths in the middle.

Asymmetric Paths and Network Boundaries

- ## Asymmetric paths often start at network boundaries
  - ### Why? Because that is where administrative policies change.
    te1-1.ar2.DCA3.gblx.net (69.31.31.209) 0.719 ms 0.560 ms 0.428 ms
    te1-2-10g.ar3.DCA3.gblx.net (67.17.108.146) 0.574 ms 0.557 ms 0.576 ms
    sl-st21-ash-8-0-0.sprintlink.net (144.232.18.65) <mark>100.280 ms 100.265 ms 100.282 ms</mark>
    144.232.20.149 (144.232.20.149) 102.037 ms 101.876 ms 101.892 ms
    sl-bb20-dc-15-0-0.sprintlink.net (144.232.15.0) 101.888 ms 101.876 ms 101.890 ms
  - ### What's wrong in the path above?
    - It COULD be congestion between GBLX and Sprint.
    - But it could also be an asymmetric reverse path.
    - At this GBLX/Sprint boundary, the reverse path policy changes.
    - This is most often seen in multi-homed network with multiple paths.
    - In the example above, Sprint's reverse route goes via a circuit that is congested, but that circuit is **NOT** shown in this traceroute.

Using Source Address in your Traceroute

- How can you work around asymmetric paths?
  - The most powerful option is to control your SRC address.
  - In the previous example, assume that:
    - You are multi-homed to Global Crossing and Level3
    - Global Crossing reaches you via Global Crossing
    - Sprint reaches you via Level3
    - There is a problem between Sprint and Level3.
  - How can you prove the issue isn't between GX and Sprint?
    - Run a traceroute using your side of the GBLX /30 as your source.
    - This /30 comes from your provider (GBLX)'s larger aggregate.
    - The reverse path will be guaranteed to go Sprint->GBLX
    - If the latency doesn't persist, you know the issue is on the reverse.

Asymmetric Paths with Multiple Exits

- But remember, asymmetric paths can happen *anywhere*
- Especially where networks connect in multiple locations
  - And use closest-exit (hot potato) routing, as is typically done.
  - Hop 1 (yellow) returns via a Washington DC interconnection.
  - Hop 2 (red) returns via a Chicago interconnection.
  - Hop 3 (green) returns via a San Jose interconnection.

Chicago IL

Destination

San Jose CA

Source

Washington DC
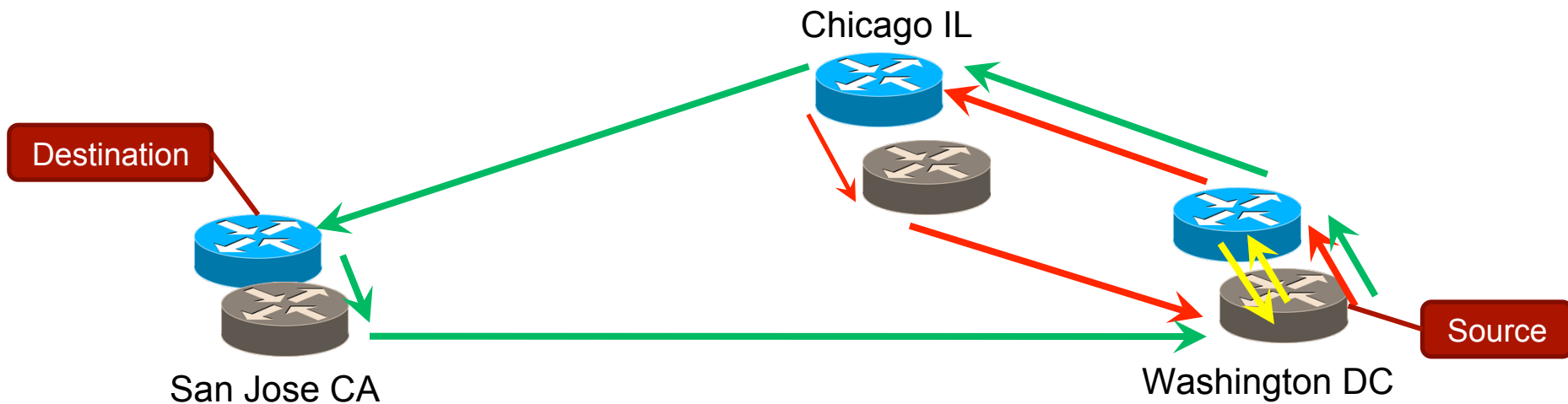
Using Source Address in your Traceroute

- But what if the /30 is numbered out of my space?
  - As in the case of a customer or potentially a peer.

- You can still see some benefits from setting SRCs
  - Consider trying to examine the reverse path of a peer who you have multiple interconnection points with.
    - A traceroute sourced from your IP space (such as a loopback) may come back via any of multiple interconnection points.
    - But if the remote network carries the /30s of your interconnection in their IGP (i.e. they redistribute connected into their IGP)…
    - Then the traffic will come back over their backbone, and return to you via the /30 you are testing from.
    - Trying both options can give you different viewpoints.

Default Source Addresses

- When tracerouting from a router…
  - Most routers default to using the source address of the egress interface that the probe leaves from.
  - This may or may not be what you want to see.
  - Some platforms can be configured to default to a loopback address rather than the egress interface.
    - For example, Juniper "system default-address-selection".

Multiple Paths and Load Balancing

Multiple Paths

- Remember, every probe is an independent trial.
  - UDP/TCP traceroute probes typically use a different layer 4 port every time, to identify which probe is which.
  - Equal-Cost Multi-Path (ECMP) may make multiple potential paths show up for each "hop" TTL value.
  - Example:

    ```
    6  ldn-bb2-link.telia.net (80.91.251.14)  74.139 ms  74.126 ms
       ldn-bb1-link.telia.net (80.91.249.77)  74.144 ms
    7  hbg-bb1-link.telia.net (80.91.249.11)  89.773 ms
       hbg-bb2-link.telia.net (80.91.250.150)  88.459 ms  88.456 ms
    8  s-bb2-link.telia.net (80.91.249.13)  105.002 ms
       s-bb2-link.telia.net (80.239.147.169)  102.647 ms  102.501 ms
    ```

  - Of the 3 probes, 2 go over one path, 1 goes over another.

Multiple Paths - Examples

- A slightly more complex example

```
4  p16-1-0-0.r21.asbnva01.us.bb.verio.net (129.250.5.21)  0.571 ms  0.604 ms  0.594 ms
5  p16-1-2-2.r21.nycmny01.us.bb.verio.net (129.250.4.26)  7.279 ms  7.260 ms
   p16-4-0-0.r00.chcgil06.us.bb.verio.net (129.250.5.102)  25.981 ms
6  p16-2-0-0.r21.sttlwa01.us.bb.verio.net (129.250.2.180)  71.027 ms
   p16-1-1-3.r20.sttlwa01.us.bb.verio.net (129.250.2.6)  66.730 ms  66.535 ms
```

- ECMP between two parallel but different paths
  - Ashburn VA – New York NY – Seattle WA
  - Ashburn VA – Chicago IL – Seattle WA

- Also harmless, but potentially confusing.

Multiple Unequal Length Paths

- The most confusing scenario is ECMP across multiple paths that are of *unequal* router hop length.
  - This can make the traceroute appear to go back and forth, and is extremely confusing and difficult to read.

- Consider the following equal-cost paths:
  - A – B – C – D – E
  - A – X – B – C – D – E
  - A Traceroute with 3 probes per hop ends up looking like:
    - 1 A A A
    - 2 B X B
    - 3 C B C
    - 4 D C D
    - 5 E D E

How To Handle Multiple Paths

- When in doubt, try looking at just a single probe.
  - Set your traceroute client to only send 1 probe per hop.
    - For many Unix implementations, the command is "-q 1".
    - JUNOS CLI lacks this, but you can do it in CLI from their unix shell.
  - But be aware that this may not be the path which your actual traffic forwards over.
  - And remember, EVERY PROBE is an independent trial.
    - Even when doing 1 probe per hop, you aren't guaranteed you're going to see a single contiguous path as taken by a single flow.
  - One way to try out different paths which may be available is to increment the dest IP by 1, or try different source IPs.
    - This can come into play when a network is doing ECMP hashing based on only layer 3 information.

MPLS and Traceroute

MPLS ICMP Tunneling

- Many large networks operate an MPLS based core

- Some devices don't even carry an IP routing table

  - This is fine for switching MPLS labeled packets.
  - But presents a problem when ICMPs are generated.
  - How does the MPLS-only router deliver an ICMP?

- One solution is called ICMP Tunneling:

  - If generating an ICMP about a packet inside an LSP
  - Then put the generated ICMP back into the same LSP
  - This works for delivering the message, but…
  - It can make traceroutes look really WEIRD!

MPLS ICMP Tunneling Diagram
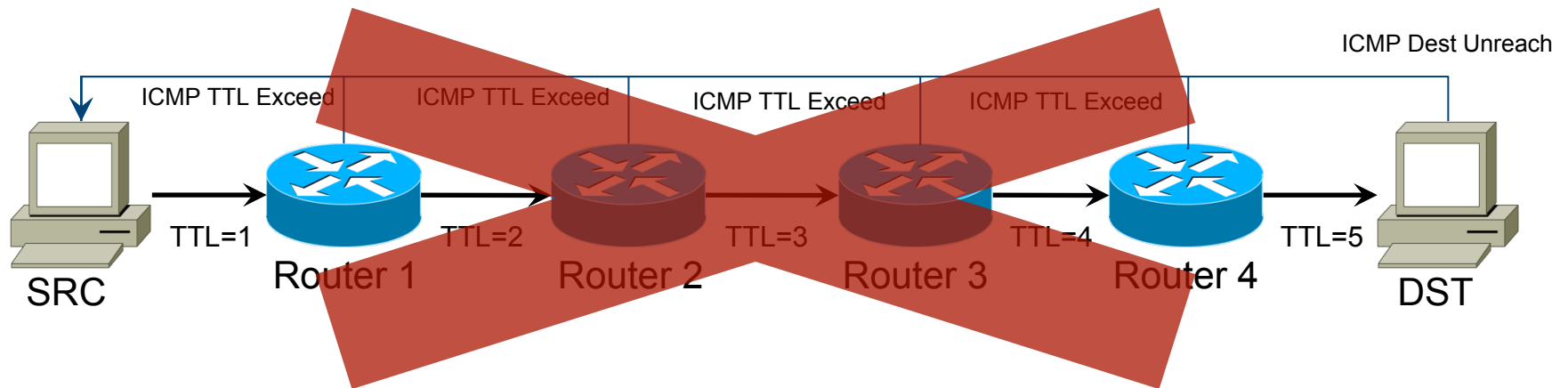
All returned ICMP packets must travel to the end of the LSP before going back to the sender. This makes every hop in the LSP appear to have the same RTT as the final hop.
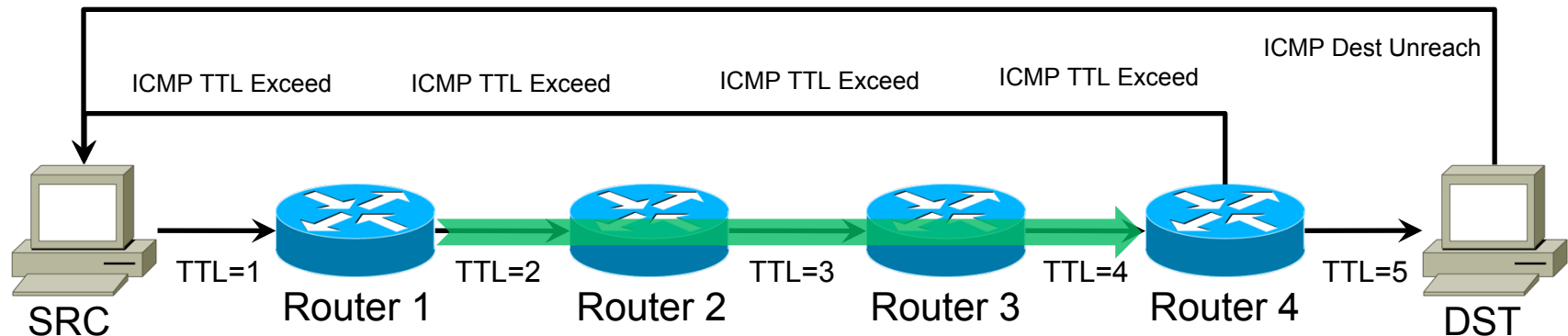
MPLS ICMP Tunneling Example

1. te2-4.ar5.PAO2.gblx.net (69.22.153.209) 1.160 ms 1.060 ms 1.029 ms
2. 192.205.34.245 (192.205.34.245) 3.984 ms 3.810 ms 3.786 ms
3. tbr1.sffca.ip.att.net (12.123.12.25) `74.848 ms  74.859 ms  74.936 ms`
4. cr1.sffca.ip.att.net (12.122.19.1) 74.344 ms 74.612 ms 74.072 ms
5. cr1.cgcil.ip.att.net (12.122.4.122) 74.827 ms 75.061 ms 74.640 ms
6. cr2.cgcil.ip.att.net (12.122.2.54) 75.279 ms 74.839 ms 75.238 ms
7. cr1.n54ny.ip.att.net (12.122.1.1) 74.667 ms 74.501 ms 77.266 ms
8. gbr7.n54ny.ip.att.net (12.122.4.133) 74.443 ms 74.357 ms 75.397 ms
9. ar3.n54ny.ip.att.net (12.123.0.77) 74.648 ms 74.369 ms 74.415 ms
10. 12.126.0.29 (12.126.0.29) 76.104 ms 76.283 ms 76.174 ms
11. route-server.cbbtier3.att.net (12.0.1.28) `74.360 ms  74.303 ms  74.272 ms`

Final Thoughts: The Traceroute Checklist

- Before beginning any serious traceroute analysis, you should **ALWAYS** ask for:
  - Traceroutes in both directions (forward *AND* reverse).
  - The real source and destination IPs for those traceroutes.
- Beware of:
  - Snippits of traceroutes with missing information.
  - Users running a traceroute towards an IP they saw in another traceroute, rather than a real destination host.
    - These can route VERY differently, and aren't the same as the real destination IP for network performance purposes.

Send questions, complaints, to:

Richard A Steenbergen <ras@turkbergen.com>