The Internet X-Ray: Diagnosing ECMP failures from the edge

Bela Toros, Chris Laffin, Tyler Leeds
Who we are:
Basics of an **Anycast CDN**

**Anycast CDN Basics**
- Same IP prefix announced from multiple places
- User requests are routed by BGP to nearest POP
- Media that is cached is served directly
- Media that isn't cached is proxied by the POP back to an origin datacenter

**Automattic CDN Specifics**
- Origin fetch happens over DFZ transit links
- Each POP and Origin connected to several Tier1 NSP
- No private links between POP and Origin
Basics of **ECMP**

**ECMP Basics**

- Single hop represents multiple physical links
- Links might be aggregated into an LACP LAG
- Links might be several equal cost Layer 3 paths
- Links can congest or drop packets independently
- Layer 1 issues can drop packets without generating errors or discards on routers
The Internet is a **Flow Switched Network**

- Passing through the same routers does not mean you pass through the same physical links.
- Pinging along the same path is insufficient to establish whether path is loss-free.
- Traceroute usually shows router Lo0 IP/hostname or the IP of the interface used to reach the source of the traceroute.
Basics of Link Load Balancing

Load Balancing Basics

- Routers DO NOT use per-packet load balancing
- Routers use their own formula based on the 5tuple or 3tuple to choose the egress port
- Egress port decision is stateless (but stable)
- Hash formula is applied to every packet
- Specific flows (5tuples) will always use same physical ports
- Port hashes are usually recalculated on topology change
The Problem
Our Solution: **PINGO**

Pingo is a custom GoLang daemon that creates a matrix of probes between source-ports at each site. By varying the 5tuple, probes are load-balanced by the internet to different physical links and issues with a single link can be quantified and located. Pingo will be open sourced on Github shortly.

![Diagram]

* Only subset of flows shown for clarity

**Specifications**

- Sends timecode based probe every 500ms.
- Source-ports are assigned to each NSP
- Probe initiated from Origin DC’s
- Each source port pings all ports on all NSPs at target site
- Calculates latency for each probe
- Loss is declared after 2000ms
- Stats are gathered in TSDB (Prometheus)
- Can determine unidirectional loss and provide data to pinpoint where
- Routinely sensitive to < 0.05% loss.
- Accurate to ICMP based latency calculation to within 2ms
- Currently running a 6x6 matrix for every dc/nsp combo. (4x4 shown)
PINGO Dashboard Primer - How to interpret

Current active path

Outbound on 2914

Outbound on 1299

Outbound on 3356

Return on 2914

Return on 3356

Return on 7922
PINGO Dashboard Primer - Drill down

Latency Strata Plot

- Flows self organize into obvious latency strata.
- Each strata represents a distinct path across the internet.
- Number of strata is unfixed.
- Strata will frequently disappear or appear as paths are changed by NSP’s.
Case Study 1: Single Path Congestion
Case Study 1: Single Path Congestion

* Only subset of flows shown for clarity
Case Study 1: Single Path Congestion

What’s happening

- Showing all 36 flows
- Shows several flows are experiencing latency while others are fine
- Root cause was a customer sending lots of traffic via a single flow. The links in that flow congested while most were fine.
- This affected 50% of flows. Normal monitoring would miss this 50% of the time.
Case Study 2: Single Path Congestion + Link dropping
Case Study 2: Single Path Congestion (Tail dropping)

What’s happening

- Showing all 36 flows
- Shows several flows are experiencing latency while others are fine
- Congestion on high-latency circuit is sufficient to induce loss.
- Low latency path is loss-free
Case Study 2: Single Path Congestion + Link dropping
Case Study 2: Link loss

What’s happening

- Not congestion
- Showing all 36 flows
- All paths experiencing loss
- Still shows obvious latency “strata” indicating multipathing
- No jump in latency at all.
- Caused by single “bottleneck” link taking errors.
Case Study 2: Link loss

What’s happening

- Not congestion
- Showing all 36 flows
- All paths experiencing loss
- Still shows obvious latency “strata” indicating multipathing
- No jump in latency at all.
- Caused by single “bottleneck” link taking errors.
Case Study 3: Locating a lossy link
## Case Study 3: Locating a lossy link

```
root@net1.atl.wordpress.com:~# mtr -z -b -u -4 -B253 -L50002 -P50000 192.0.70.252 -z 1.3 -w -c 50 -o LSDNA
Start: 2023-09-21T15:54:24+0000
HOST: net1.atl.wordpress.com
```

<table>
<thead>
<tr>
<th>HOST</th>
<th>Loss%</th>
<th>Snt</th>
<th>Drop</th>
<th>Last</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS2635</td>
<td>0.0%</td>
<td>50</td>
<td>0</td>
<td>0.5</td>
<td>1.3</td>
</tr>
<tr>
<td>AS2914</td>
<td>0.0%</td>
<td>50</td>
<td>0</td>
<td>0.5</td>
<td>3.1</td>
</tr>
<tr>
<td>AS2914</td>
<td>0.0%</td>
<td>50</td>
<td>0</td>
<td>32.1</td>
<td>5.5</td>
</tr>
<tr>
<td>AS2914</td>
<td>0.0%</td>
<td>50</td>
<td>0</td>
<td>17.2</td>
<td>17.6</td>
</tr>
<tr>
<td>AS2914</td>
<td>22.0%</td>
<td>50</td>
<td>11</td>
<td>59.1</td>
<td>62.7</td>
</tr>
<tr>
<td>AS2914</td>
<td>14.0%</td>
<td>50</td>
<td>7</td>
<td>84.7</td>
<td>84.1</td>
</tr>
<tr>
<td>AS2914</td>
<td>16.0%</td>
<td>50</td>
<td>8</td>
<td>80.8</td>
<td>81.6</td>
</tr>
<tr>
<td>AS2914</td>
<td>14.0%</td>
<td>50</td>
<td>7</td>
<td>70.2</td>
<td>71.0</td>
</tr>
<tr>
<td>AS2635</td>
<td>18.0%</td>
<td>50</td>
<td>9</td>
<td>70.3</td>
<td>70.7</td>
</tr>
</tbody>
</table>
Case Study 3: Locating a lossy link

- dfw to sjc via 1299

- dfw to sjc via 2914

- dfw to sjc via 3356
Case Study 3: Locating a lossy link
Case Study 4: WTH?
Case Study 4: WTH?
Case Study 4: WTH?
Case Study 4: WTH?
Case Study 4: WTH?
Case Study 4: WTH?
Case Study 4: WTH?