Move Fast, Unbreak Things!

Network debugging at scale

Petr Lapukhov
Network Engineer

People who made this possible

Aijay Adams

Lance Dryden

Angelo Failla

Zaid Hammoudi

James Paussa

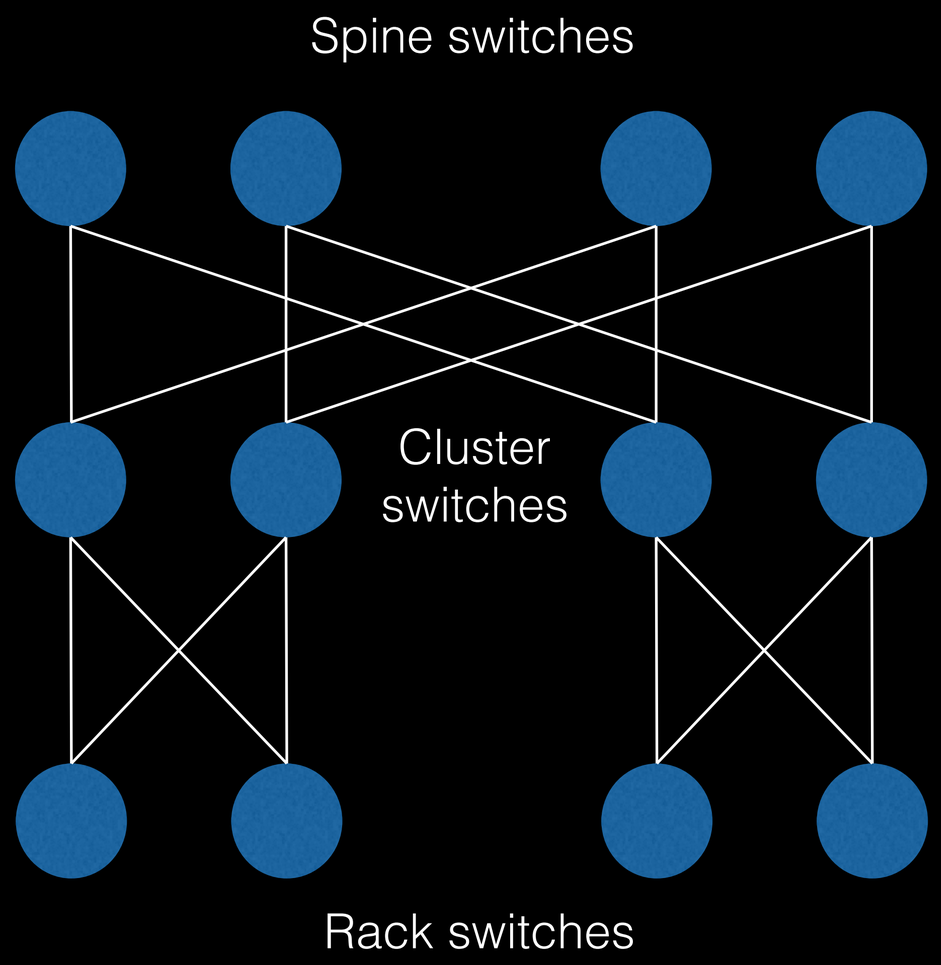
James Zeng

Basics of fault detection

How people fix broken networks

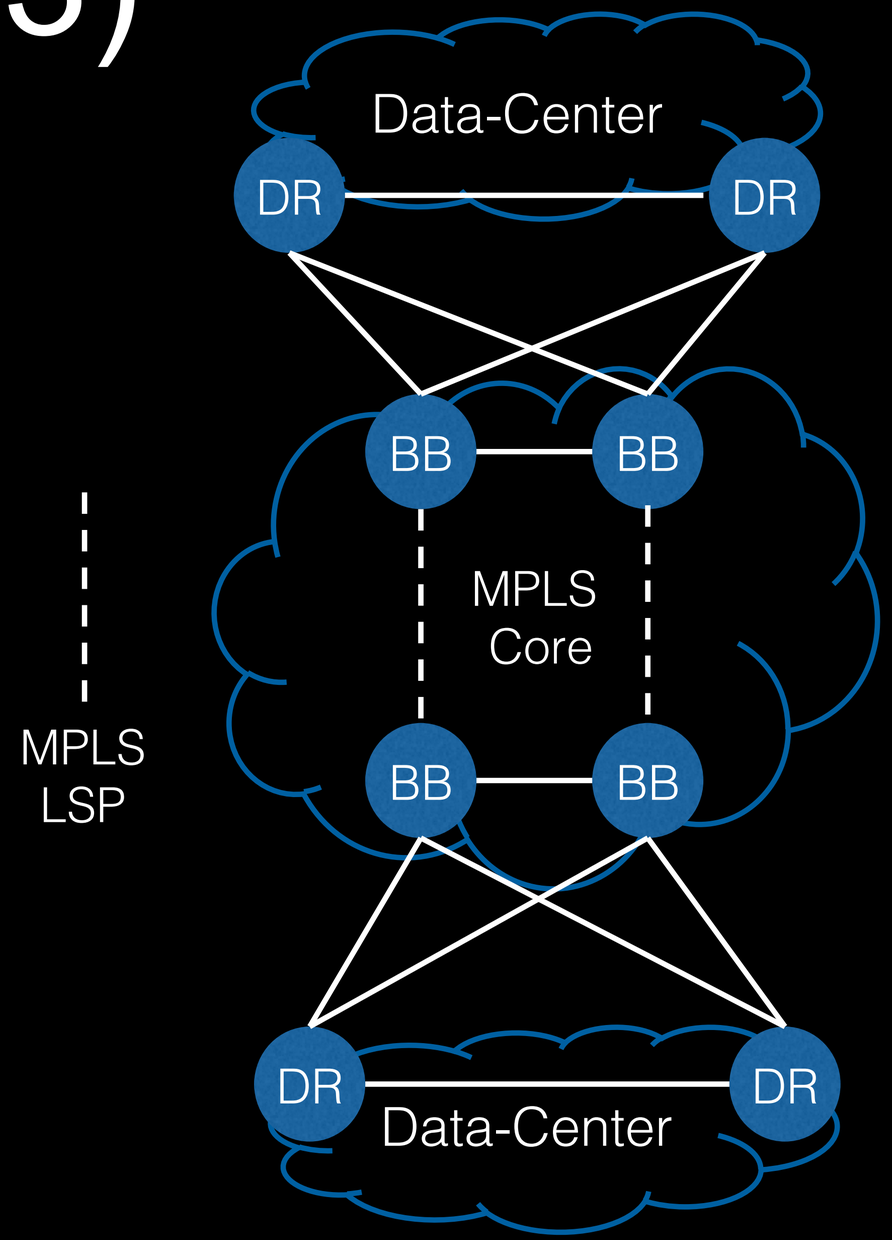
Data-center network (3)

- Multi-stage Clos Topologies
- Lots of devices and links
- BGP Only
- IPv6 >> IPv4
- Large ECMP fan-out
- L2 and L3 ECMP



Backbone network (3)

- MPLS core
  - BB = Backbone Router (LSR)
- Data-center attachment
  - DR = Datacenter Router (LER)
- Auto-bandwidth
- ECMP over MPLS tunnels



Detecting packet loss (4)

Standard counters

Non-Standard counters

```
fsw001.p001.f01.atn1# show platform trident counters
Debug counters
            Description
T2Fabric19/0/1 RX - Non congestion discards
T2Fabric19/0/1 TX - IPV4 L3 unicast aged and dropped pkts
T2Fabric19/0/1 RX - Receive policy discard
T2Fabric19/0/1 TX - L3 multicast drop
T2Fabric19/0/1 RX - Tunnel error packets
T2Fabric19/0/1 TX - Invalid VLAN
T2Fabric19/0/1 RX - Receive VLAN drop
T2Fabric19/0/1 RX - Receive multicast drop
T2Fabric19/0/1 TX - Dropped because TTL counter
T2Fabric19/0/1 RX - Receive uRPF drop
T2Fabric19/0/1 TX - Packet dropped due to any condition
T2Fabric19/0/1 RX - IBP discard and CPB full
T2Fabric19/0/1 TX - Miss in VXLT table counter
```

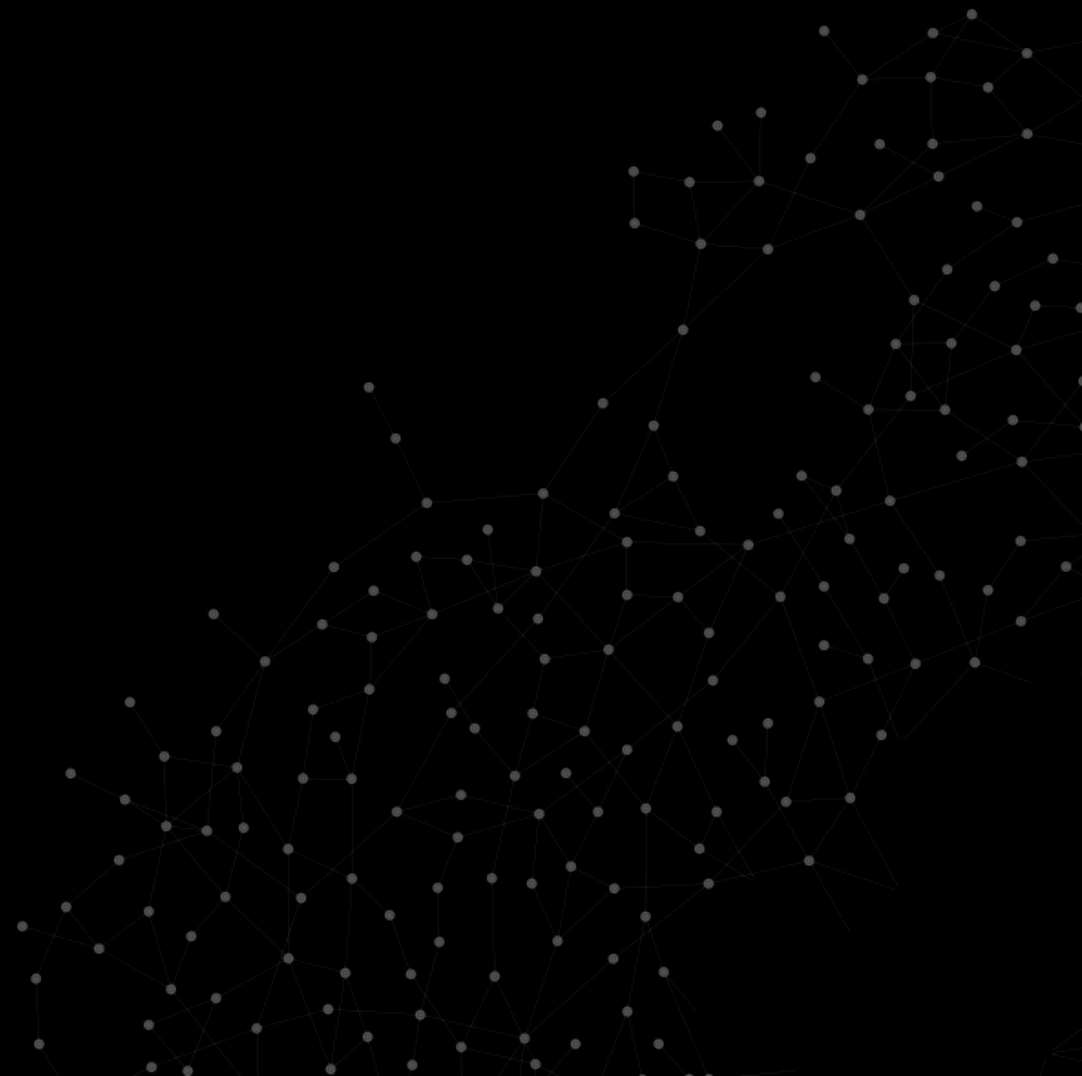
Unreliable…

How human debugs it? (4)

- Ping/hping/nping (TCP/ICMP/UDP probing)
- Change src port to try all ECMP paths
- Find a broken path, then run traceroute over it
- ping && traceroute are still important

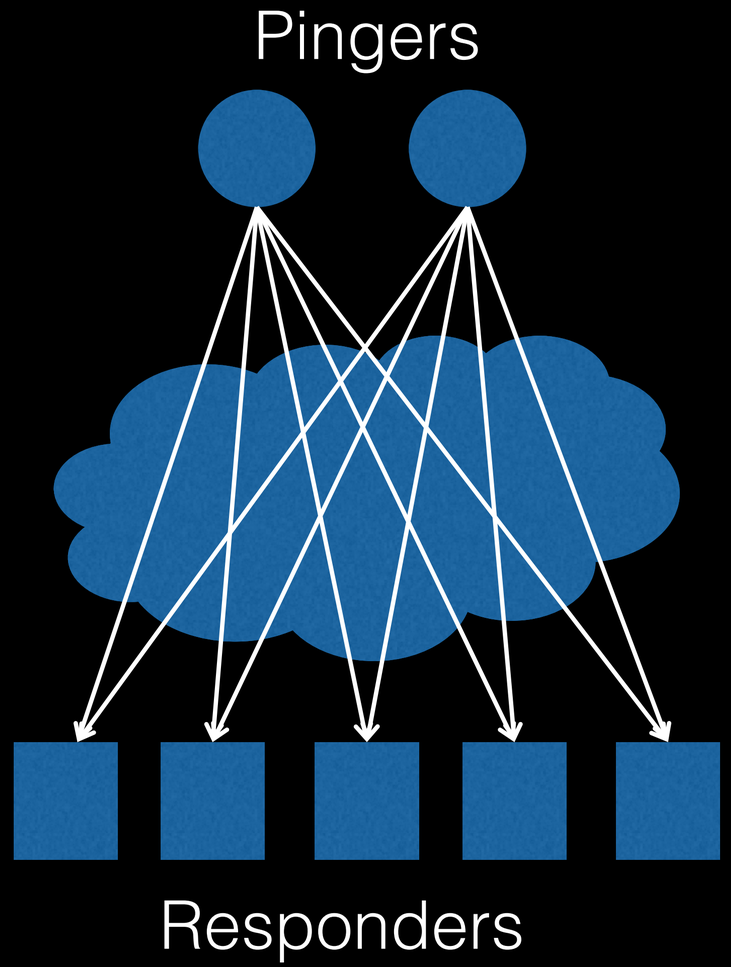
NetNORAD

The network fault detector



Massive pinging FTW

- Run “pingers” on some machines
- Run responders on lots of machines
  - Targets count ~= 100x pingers count
- Collect packet loss and RTT…
- Analyze and report!



NetNORAD evolution (4)

- 1st Run`ping` from python agent
- 2nd Raw sockets, fast TCP probes
- 3rd Raw sockets, fast ICMP probes
- Now: UDP probing + responder agent

Pinger and Responders (5)

Pingers	Responders
Send UDP probes to target list	Receive/Reply to UDP probe
Timestamp & Log results	Timestamp
High ping-rate (up to 1Mpps)	Low load: thousands of pps
Set DSCP marking	Reflect DSCP value back

Open sourced (C++),
https://github.com/facebook/UdpPinger

Allocating pingers and targets (2)

Pingers	Targets
1+ cluster per DC	2+ targets per each rack
10+ racks per cluster	10’s of thousands targets
Two pingers per rack	Consult host alarms

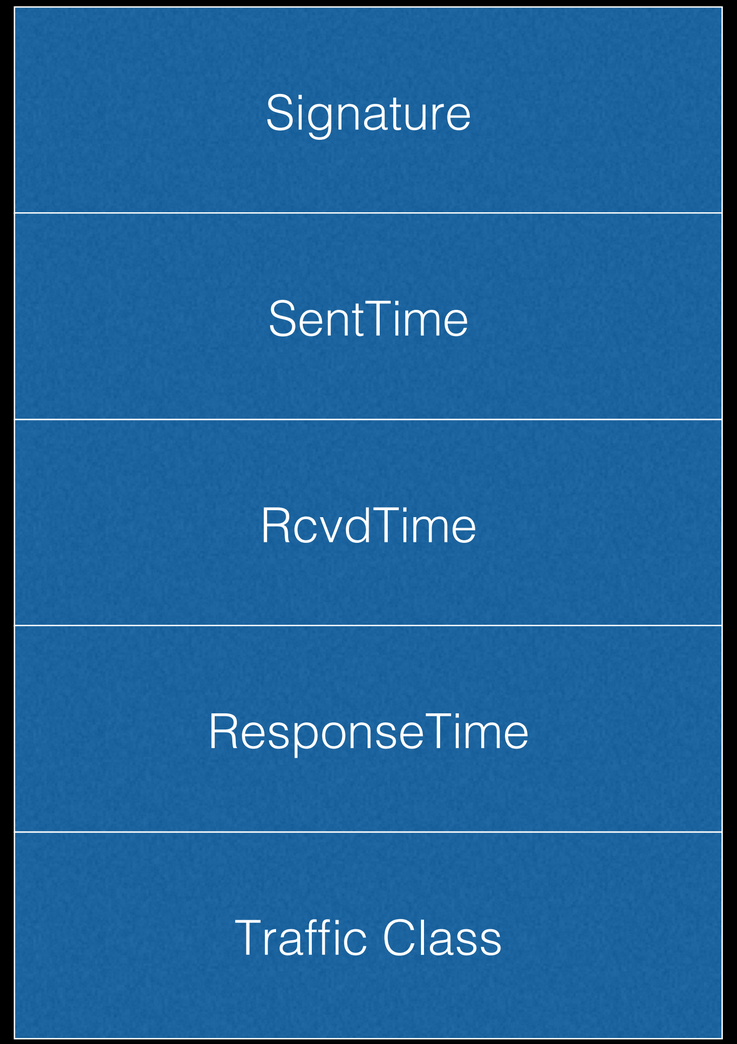
Probe timestamping

- Path changes / congestion
- Kernel time-stamps
- Application timeout tuning

Why UDP probing?

- No TCP RST packets
- Efficient ECMP
- RSS friendly
- Extensible

Probe Format



Deployment caveats (4)

Caveat	Solution
Polarization with ICMP	Use UDP
Slow IPv6 FIB lookups	4.X kernels
High-CPU boxes	Multi-threaded responder/RSS
Checksum offloading	Disable offloading ☺

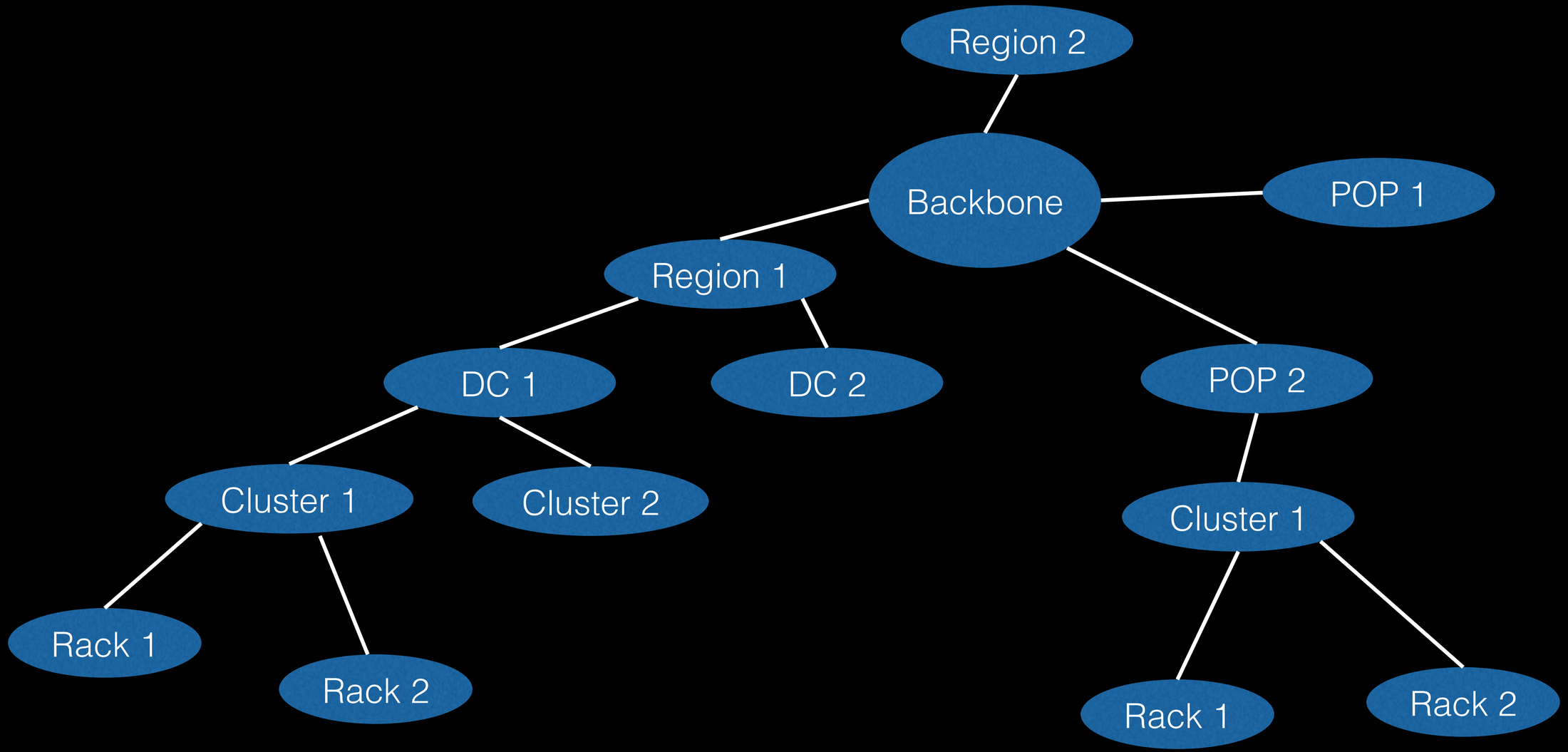
NetNORAD

How to ping and process data?

Challenges (4)

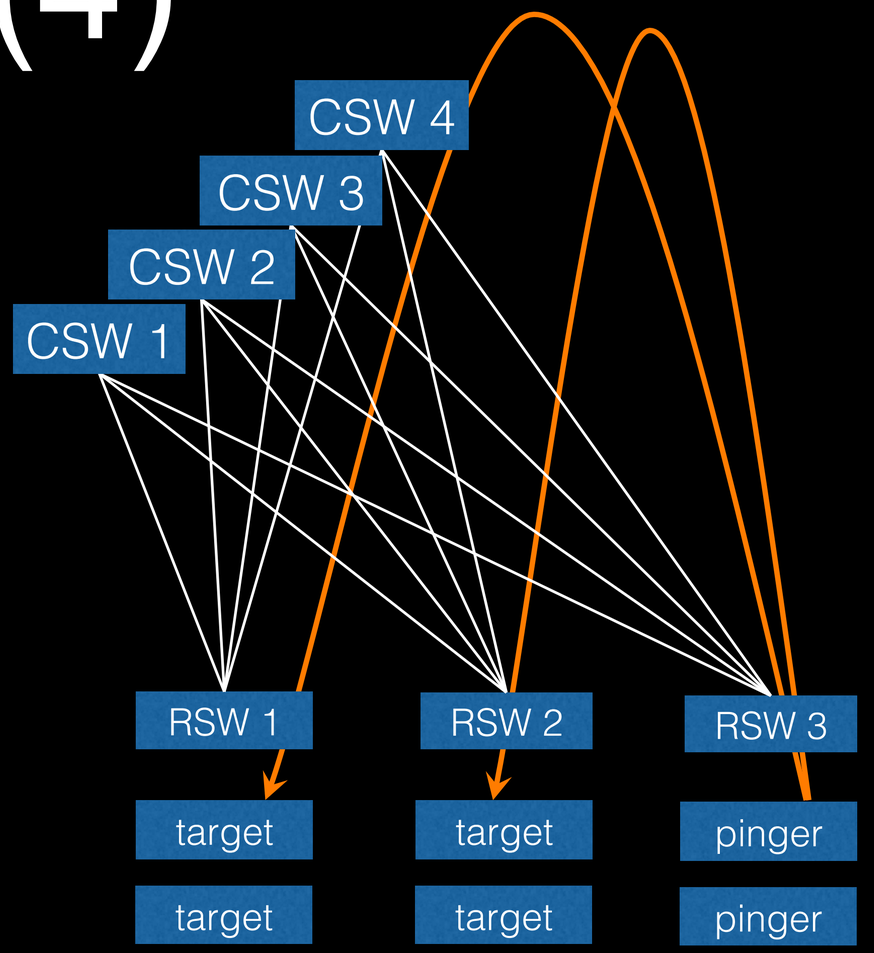
- Nx 100Gbps of ping traffic
- Tens thousands of targets
- Hundreds of pingers
- Lots of data to process
- We really do not care about each host…
- …The unit of interest is “cluster” health

The network hierarchy (4)



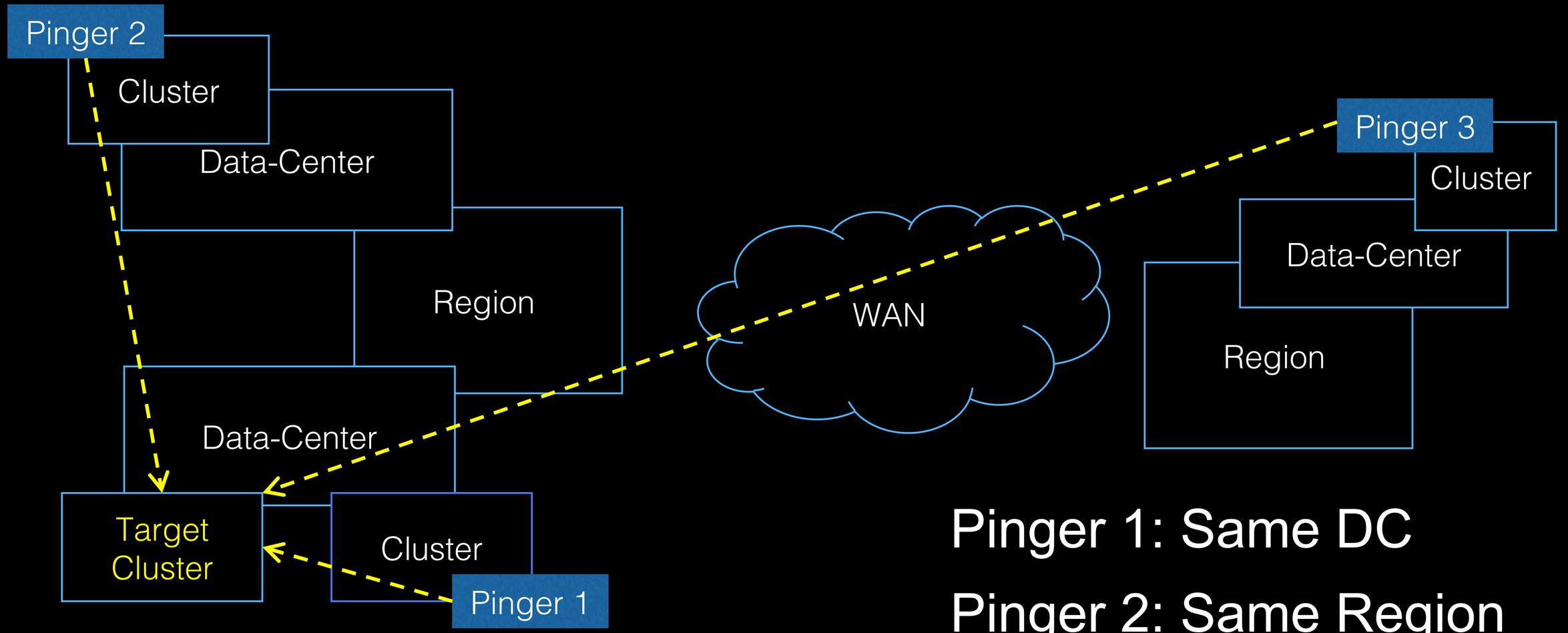
Pinging inside clusters (4)

- Detect issues with rack switches
- Dedicated pingers per cluster
- Probe ALL machines in cluster
- Store time-series per host/rack
  - Think HBase for storage
- Lags real-time by ~2-3 minutes



CSW – cluster switch
RSW – rack switch

Pinging the clusters (4)



Pinger 1: Same DC

Pinger 2: Same Region

Pinger 3: Outside of region

Proximity tagging (3)

Proximity	Scope	Goal

Pinging hierarchy

Proximity	Scope	Goal
Outside of region	Across backbone network	End-to-end issues

Pinging hierarchy

Proximity	Scope	Goal
Outside of region	Across backbone network	End-to-end issues
Same region	Between data-centers in region	Issues inside/between DCs

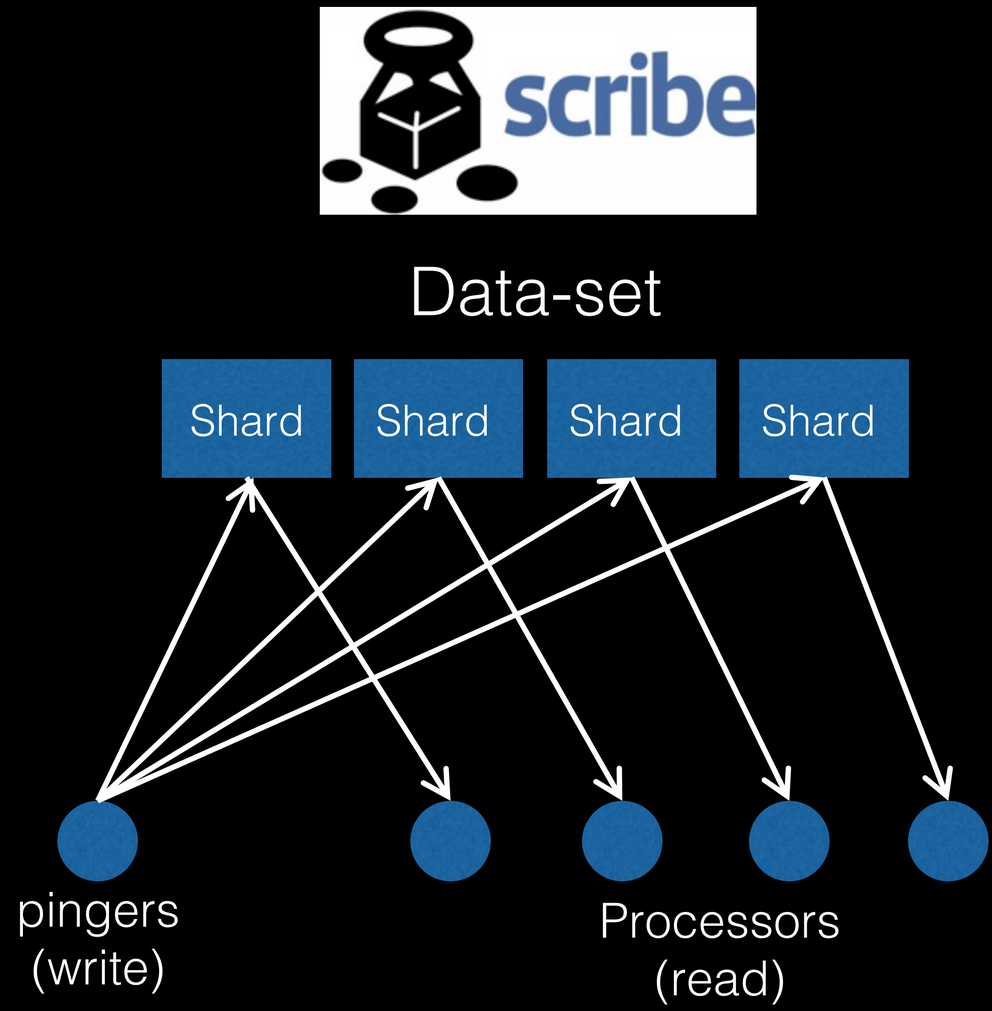
Pinging hierarchy

Proximity	Scope	Goal
Outside of region	Across backbone network	WAN issues
Same region	Between data-centers in region	Issues between DCs
Same DC	Inside one data-center	Issues in cluster switches

Processing the data

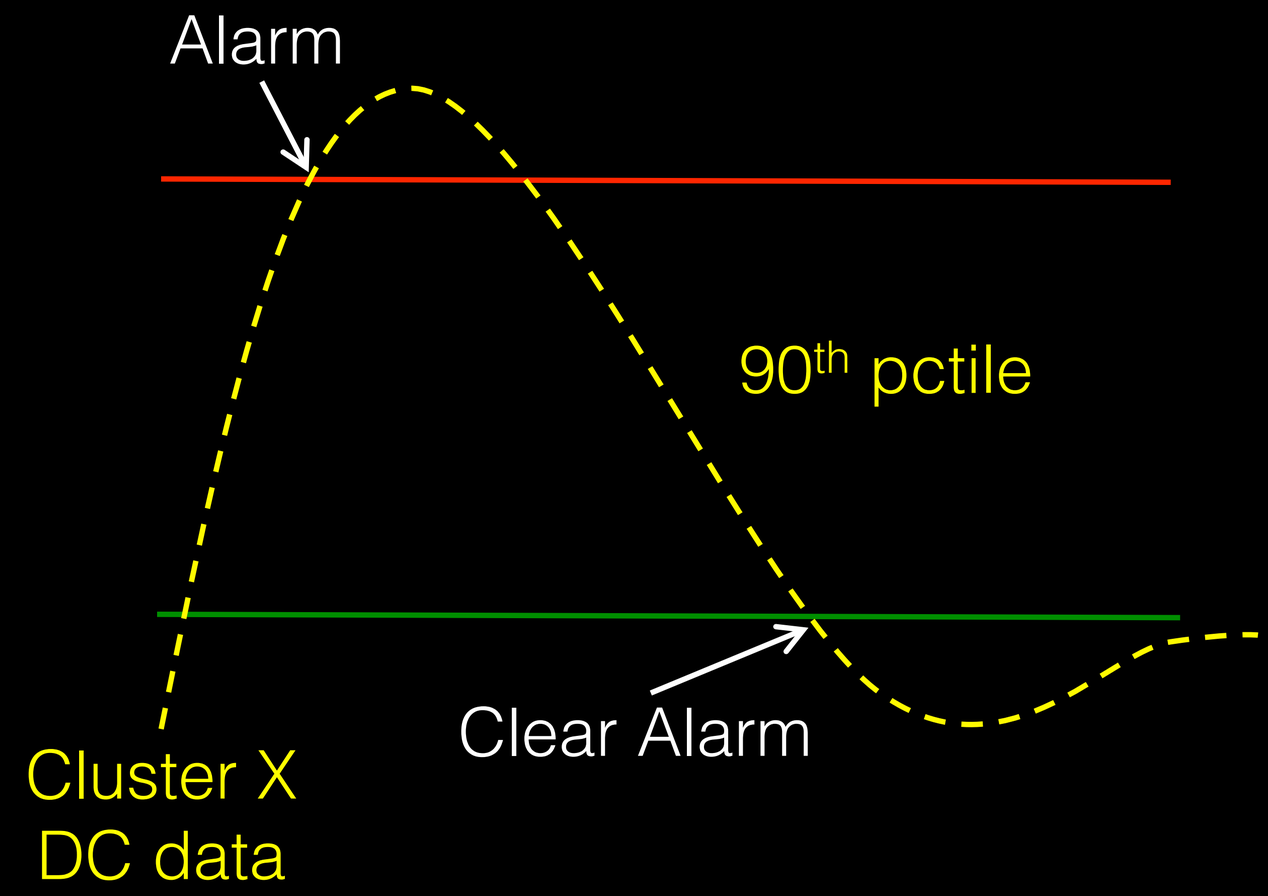
Processing pipeline: Scribe (4)

- Scribe: distributed logging system
- Similar OSS project: Kafka
- Pingers write results
- Processors consume them
- Propagation delay ~1-20 seconds



Alarming on packet loss (4)

- Build packet-loss time-series
- Track percentiles
- Alarm on rising threshold
- Clear on falling threshold
- Time to detect loss: 20 seconds



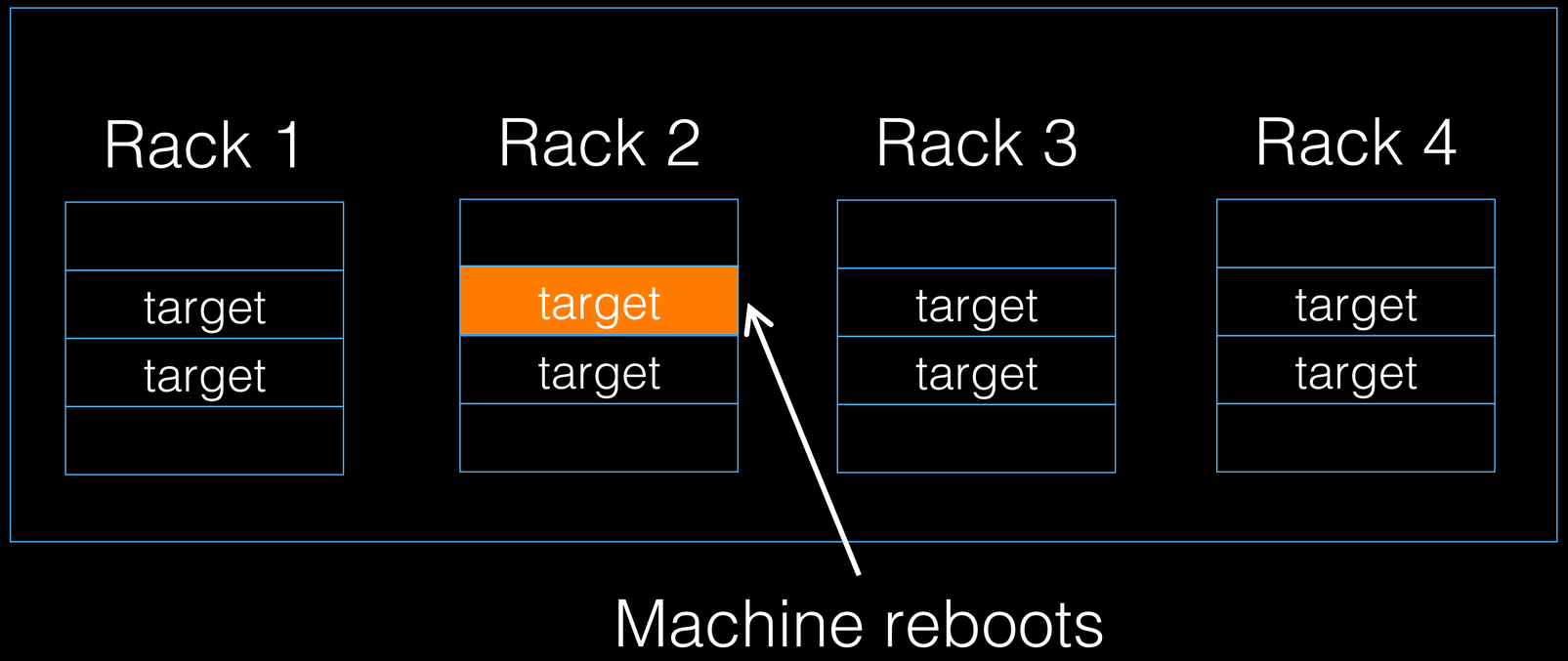
Visual analysis: Scuba

- In-memory row-oriented storage
- “Scuba: Diving into Data at Facebook”
- Similar OSS project: InfluxDB

Detecting false-positives

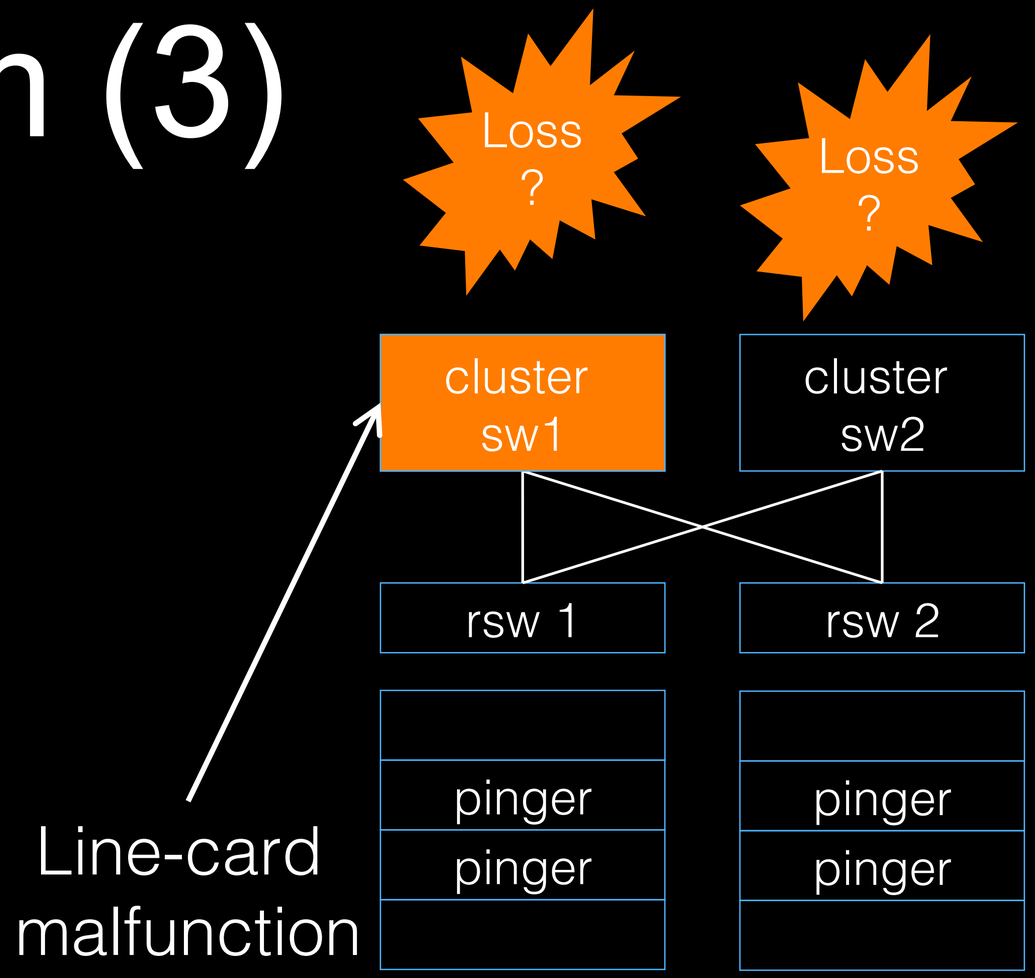
“Bad” target detection (3)

- Baseline loss
- Packet loss spike
- Filter outliers
- Done in pinger

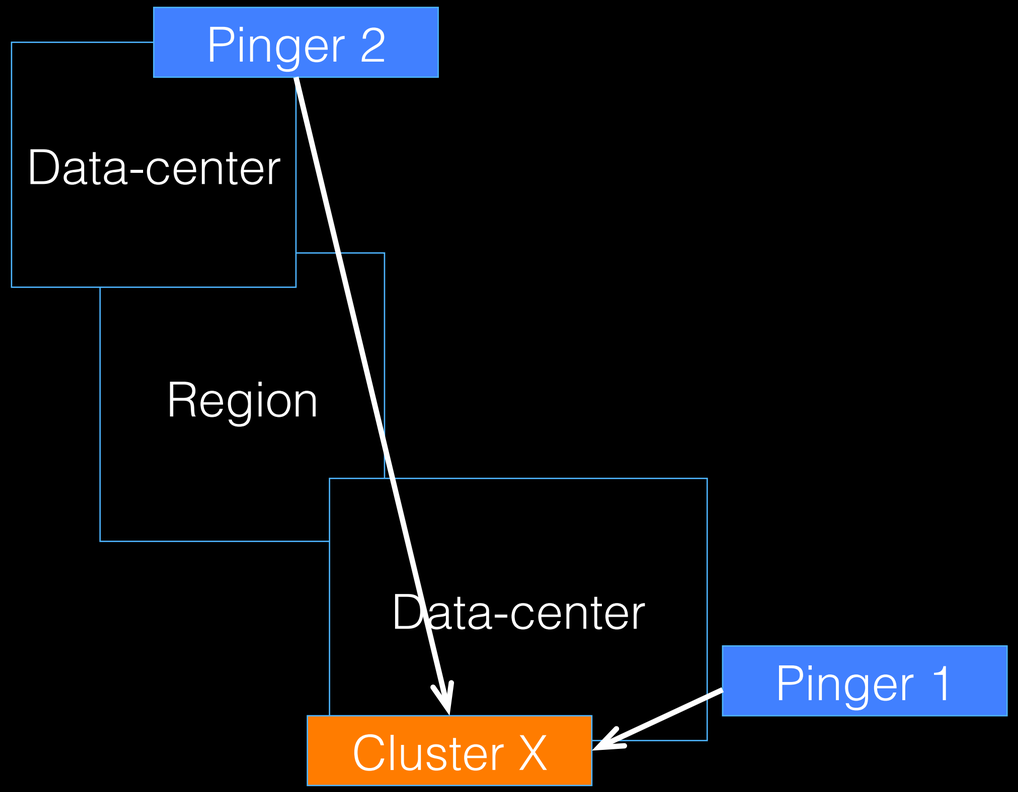


“Bad” Pinger problem (3)

- Bad cluster switch!
- Pingers see loss everywhere
- Population size is small
- Harder to weed outliers



“Bad” Pinger detection (2)



- Need more data…
- Monitor pinger cluster
- Use DC/Region pingers
- Mark “bad” clusters
- Done in processor

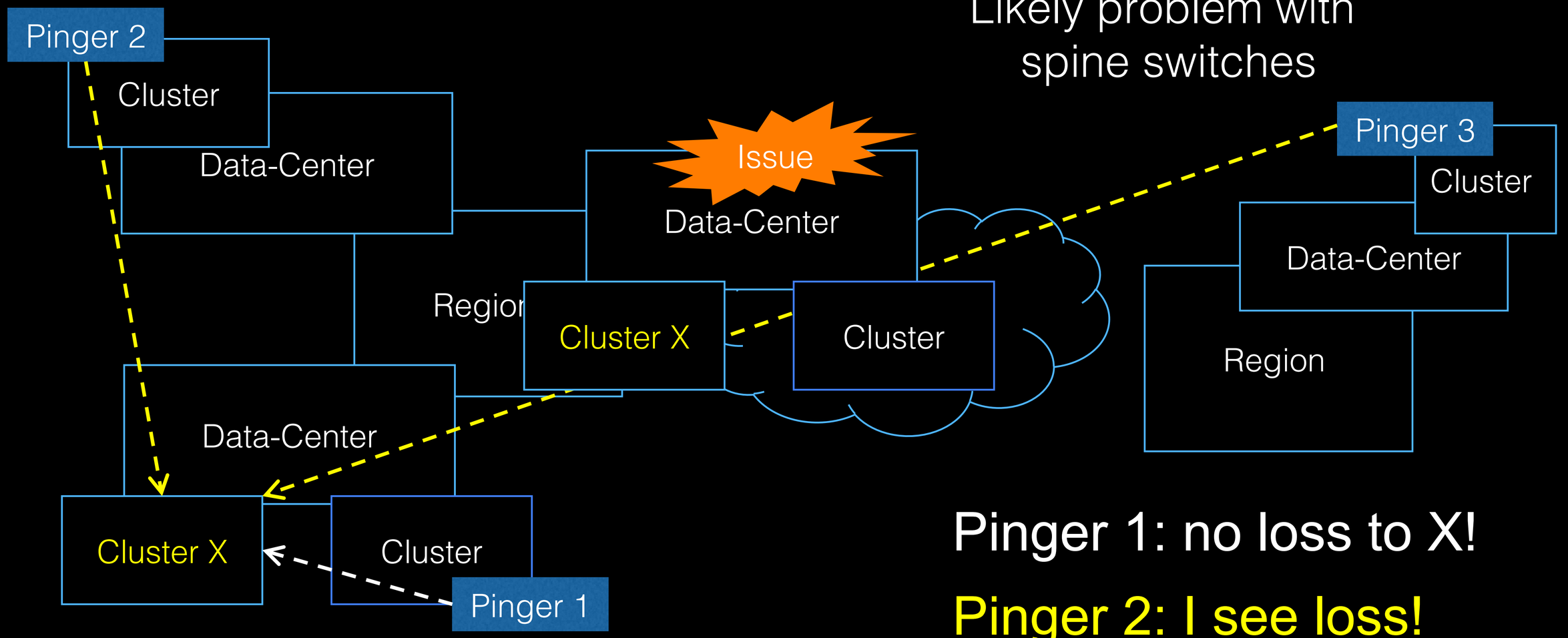
Conclusions

- Pinger/responder asymmetry
- Real-time is key
- Pinging hierarchy
- False positive elimination

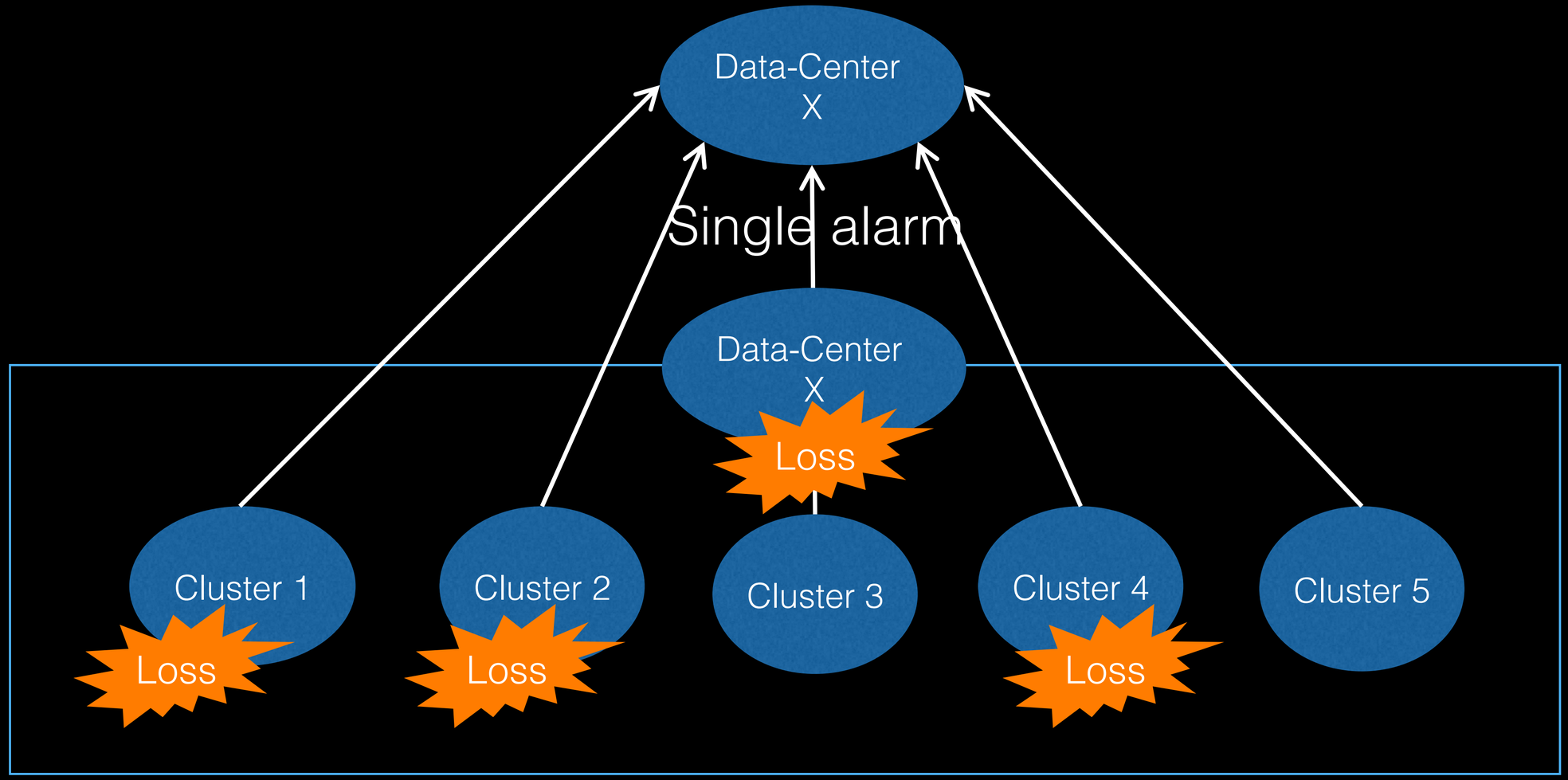
Isolating network faults

Detecting is not everything

Root cause isolation (4)



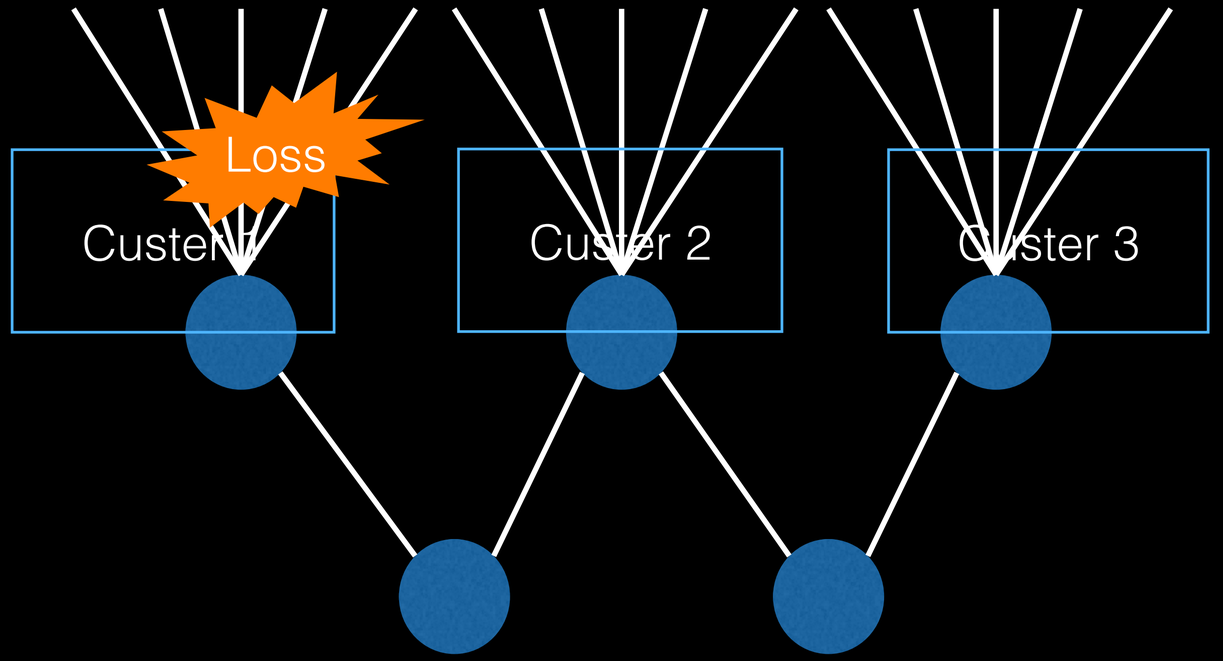
Downstream suppression (3)



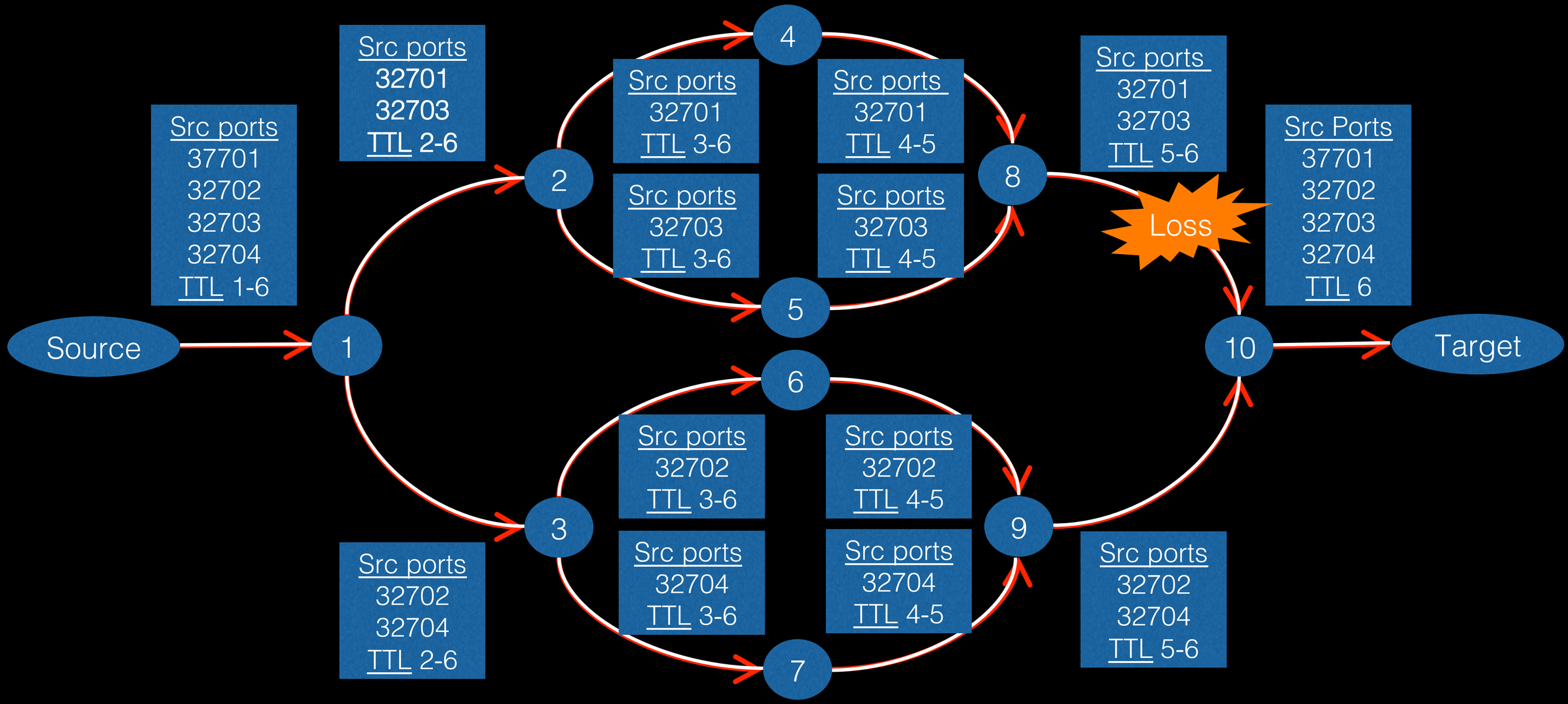
Multiple alarms

Next steps to isolate (4)

- Approximate location
- Still lots of devices/links
- Check device counters
- if that does not help…
- Remember traceroute?



Fbtracert: fast and wide traceroute (6)



Fbtracert: fast and wide traceroute

Port 32701

Path	Sent	Rcvd
1	20	20
2	20	20
4	20	20
8	20	20
10	20	14
TGT	20	15

Port 32702

Path	Sent	Rcvd
1	20	20
3	20	20
6	20	20
9	20	20
10	20	20
TGT	20	20

Port 32703

Path	Sent	Rcvd
1	20	20
2	20	20
5	20	20
8	20	20
10	20	16
TGT	20	17

Port 32704

Path	Sent	Rcvd
1	20	20
3	20	20
7	20	20
9	20	20
10	20	20
TGT	20	20

Fbtracert limitations (5)

- CoPP drops ICMP responses
- Paths may flap (MPLS LSP)
- ICMP gets tunneled with MPLS TE
- ICMP responses from wrong interfaces

Open sourced (Golang),
https://github.com/facebook/fbtracert

Conclusions

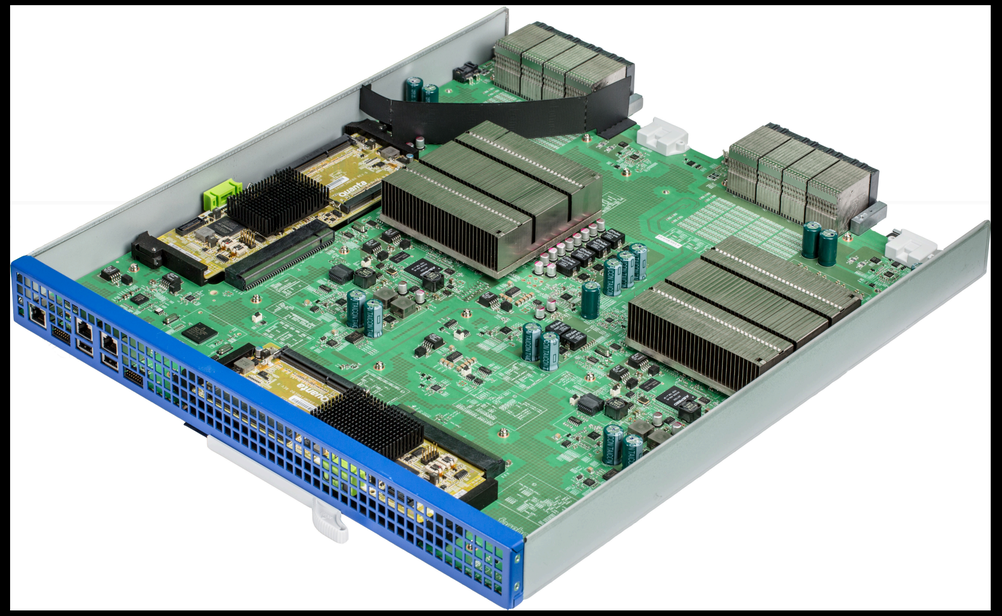
- Fault isolation is actively evolving
- Traceroute approach looks generic
- Limited by current hardware
- Backbone path churn is a serious challenge

Evolving fault detection & isolation

Near and far future

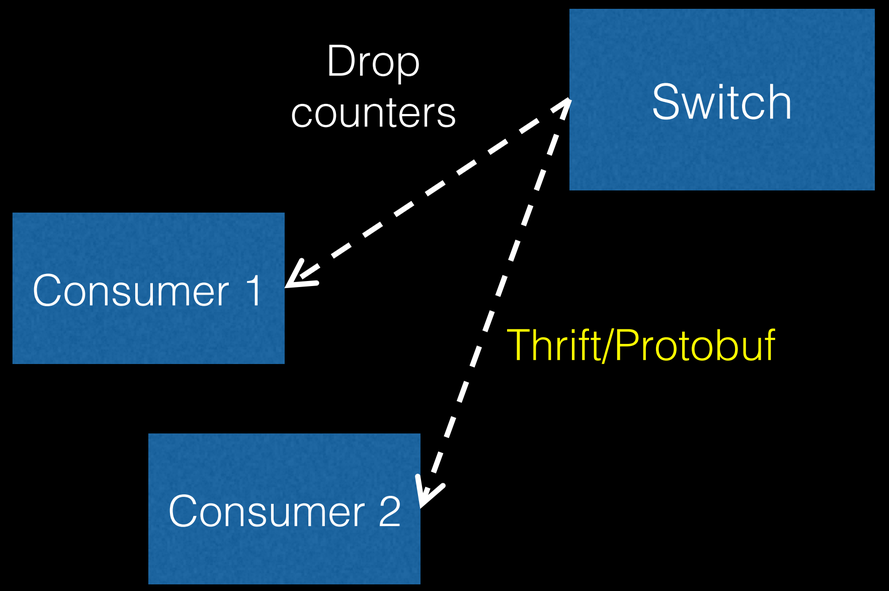
Support for on-box agents (4)

- Run same code on routers
- POSIX API
- Other SDK is welcome
- Some vendors already do that
- Be like FBOSS ☺



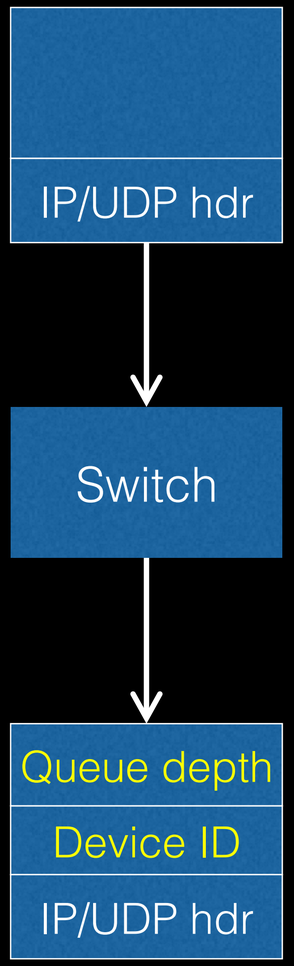
Streaming telemetry (3)

- Publishing device counters
- Faster detection
- Protobuf/Thrift for encoding
- Limited amount of counters
- Platform-specific



In-band telemetry (4)

- Next generation of silicon emerging
- Embed device stats in packets
  - E.g. device ID, or queue depth
- Use extra space in UDP probes
- Allow for real-time path tracing



facebook