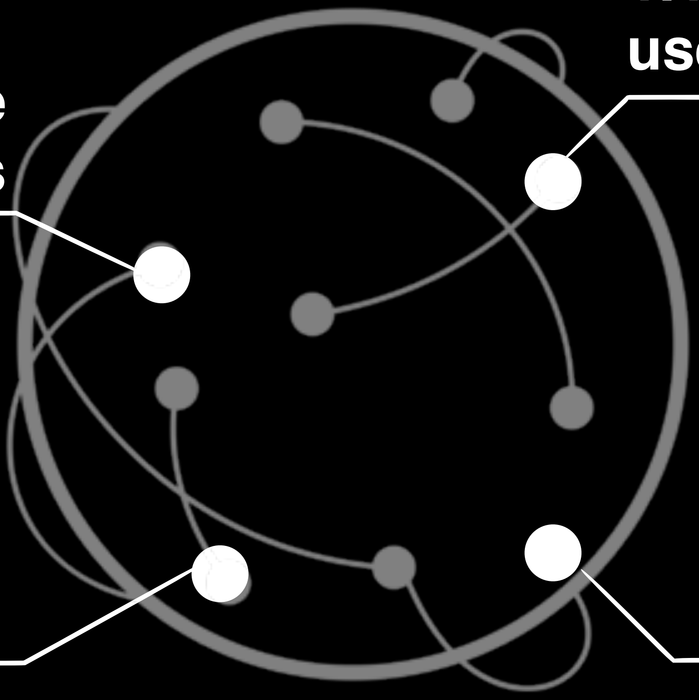
Untrusting the network

Aijay Adams
Jose Leitao
Production Network Engineers

facebook

facebook

as of June 2016



1.13 billion daily active users on average

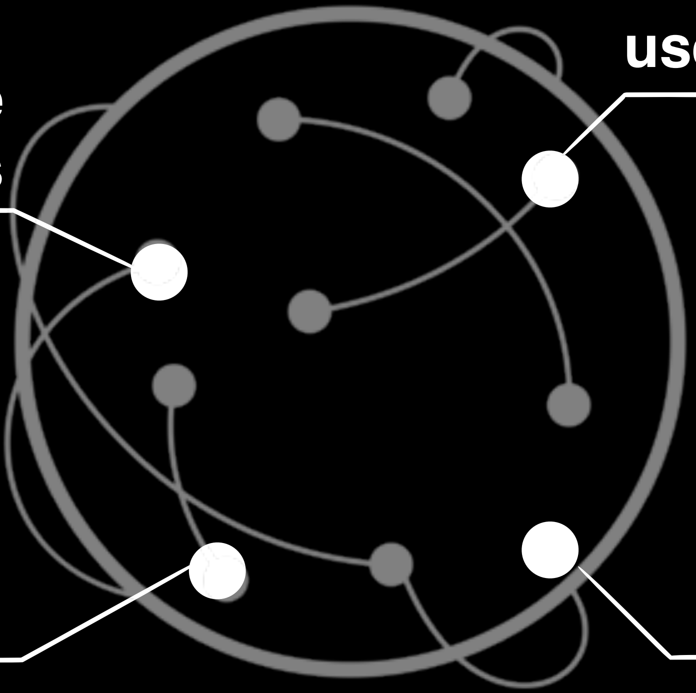
1.57 billion mobile monthly active users

1.71 billion monthly active users

1.03 billion mobile daily active users on average

facebook

as of June 2016



1.57 billion mobile monthly active users

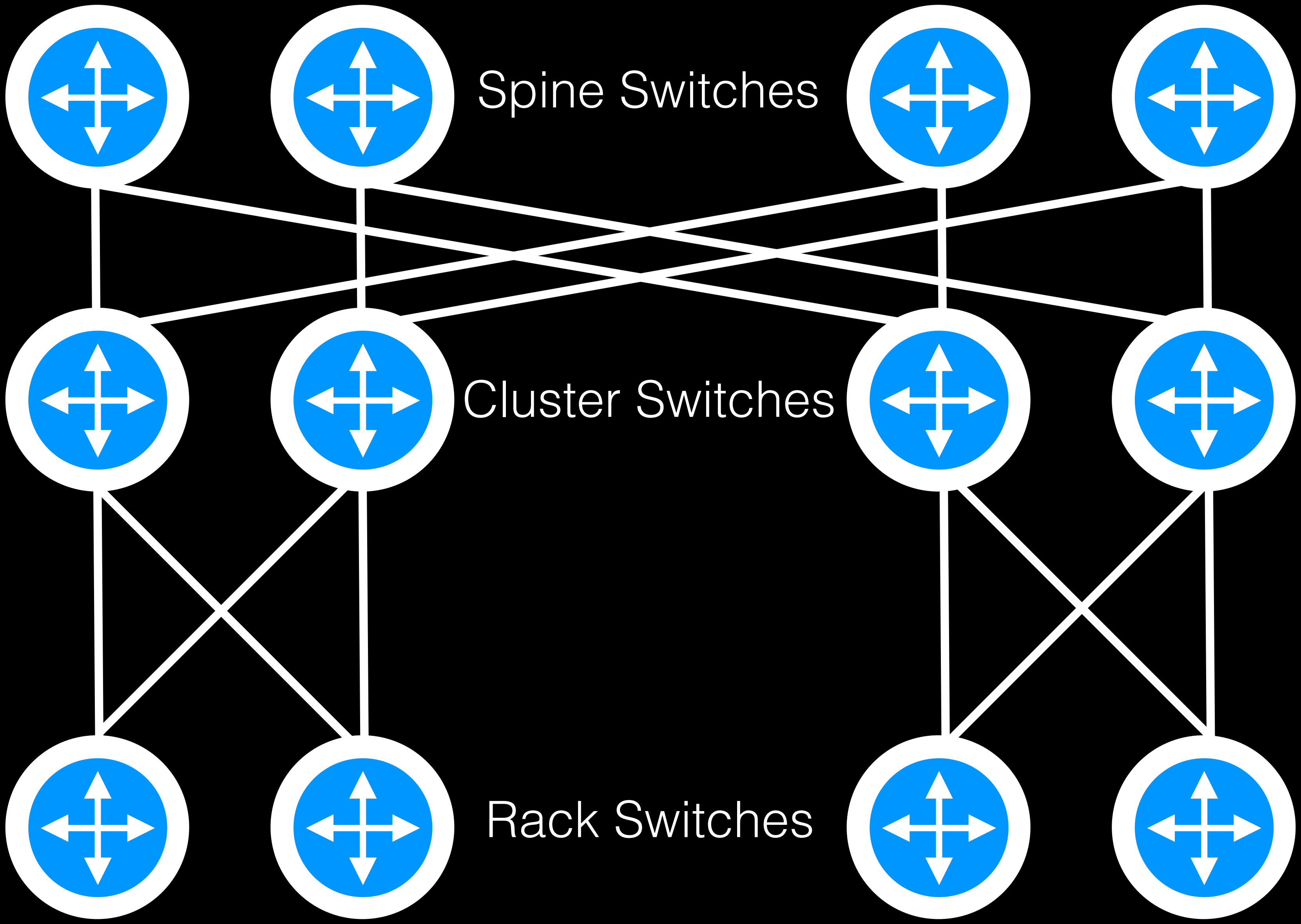
1.13 billion daily active users on average

1.71 billion monthly active users

1.03 billion mobile daily active users on average

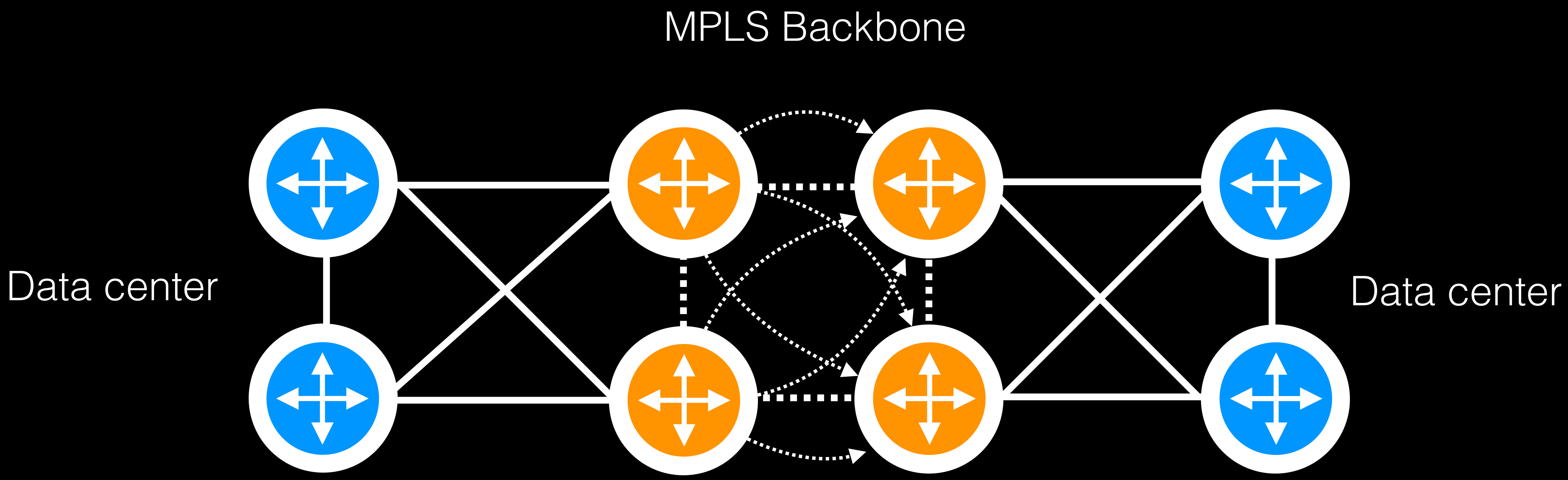
Approximately 84.5% of our daily active users are outside the US and Canada.

Data Center Network



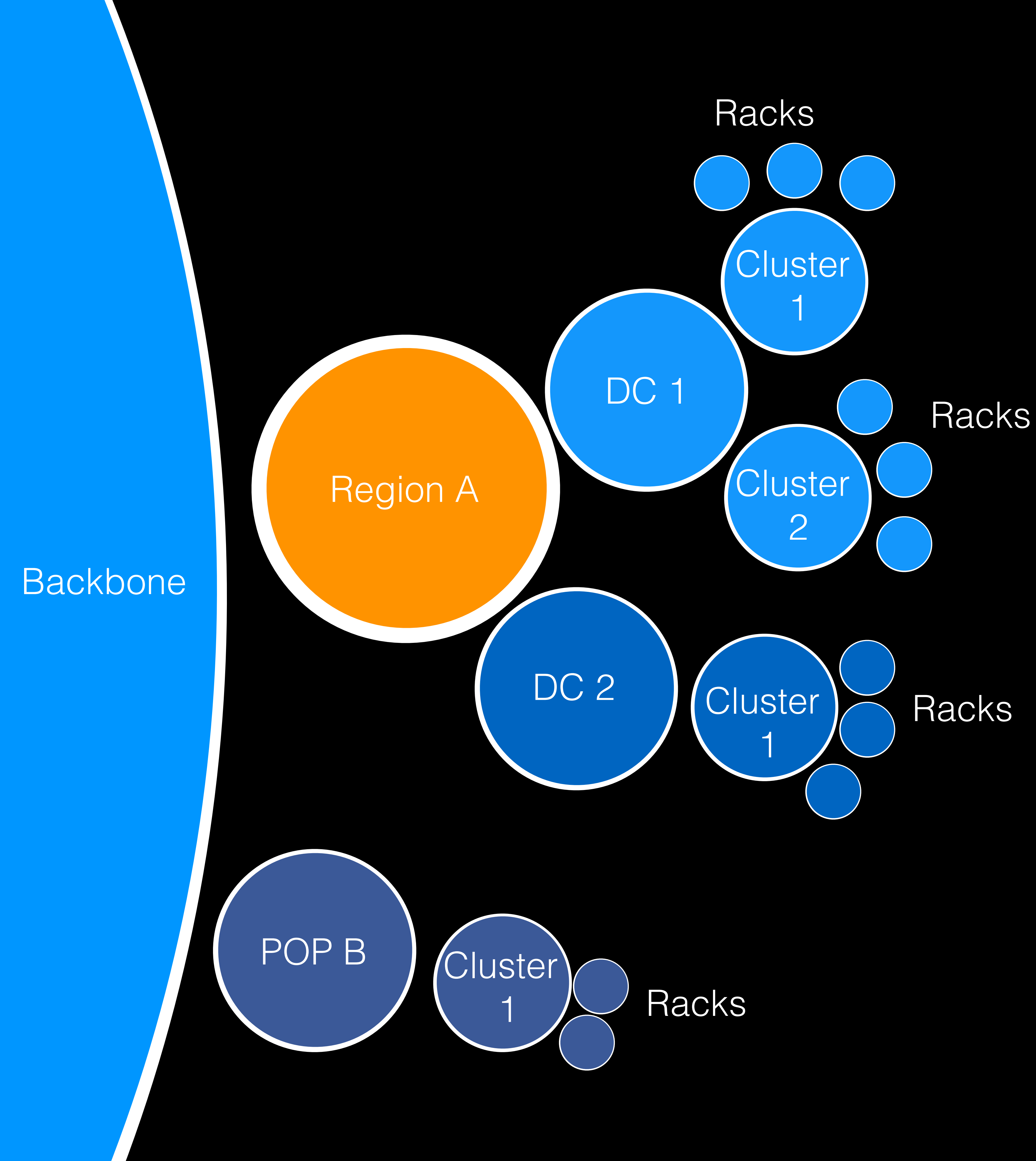
Wide ECMP, many paths!

Backbone Network



Auto Bandwidth

ECMP over MPLS Tunnels


Racks
Cluster 1
DC 1
Racks
Region A
Cluster 2
Backbone
DC 2
Cluster 1
Racks
POP B
Cluster 1
Racks

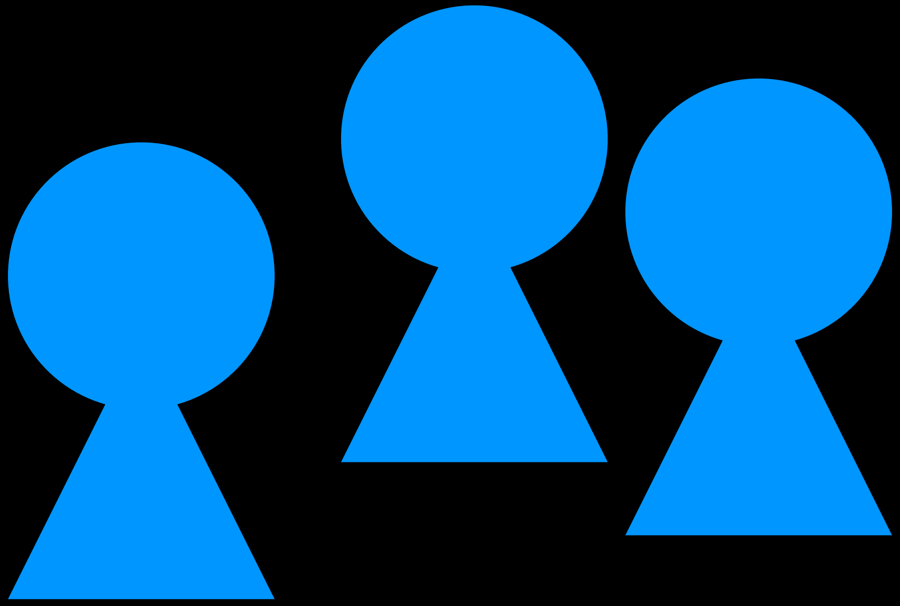
Monitoring the Network

????

Monitoring the Network

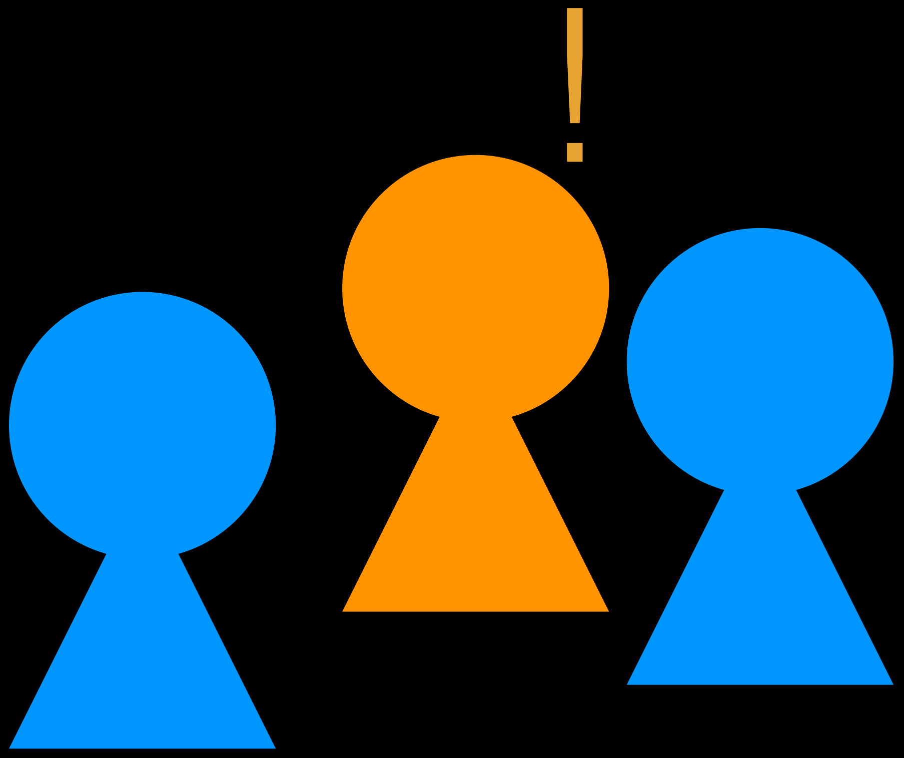
Counters and Logs

Monitoring the Network



Coworkers

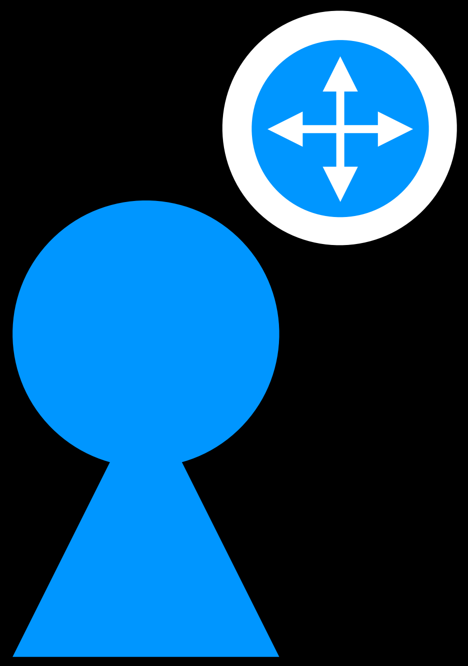
Monitoring the Network



Coworkers

Investigate

#network_engineer

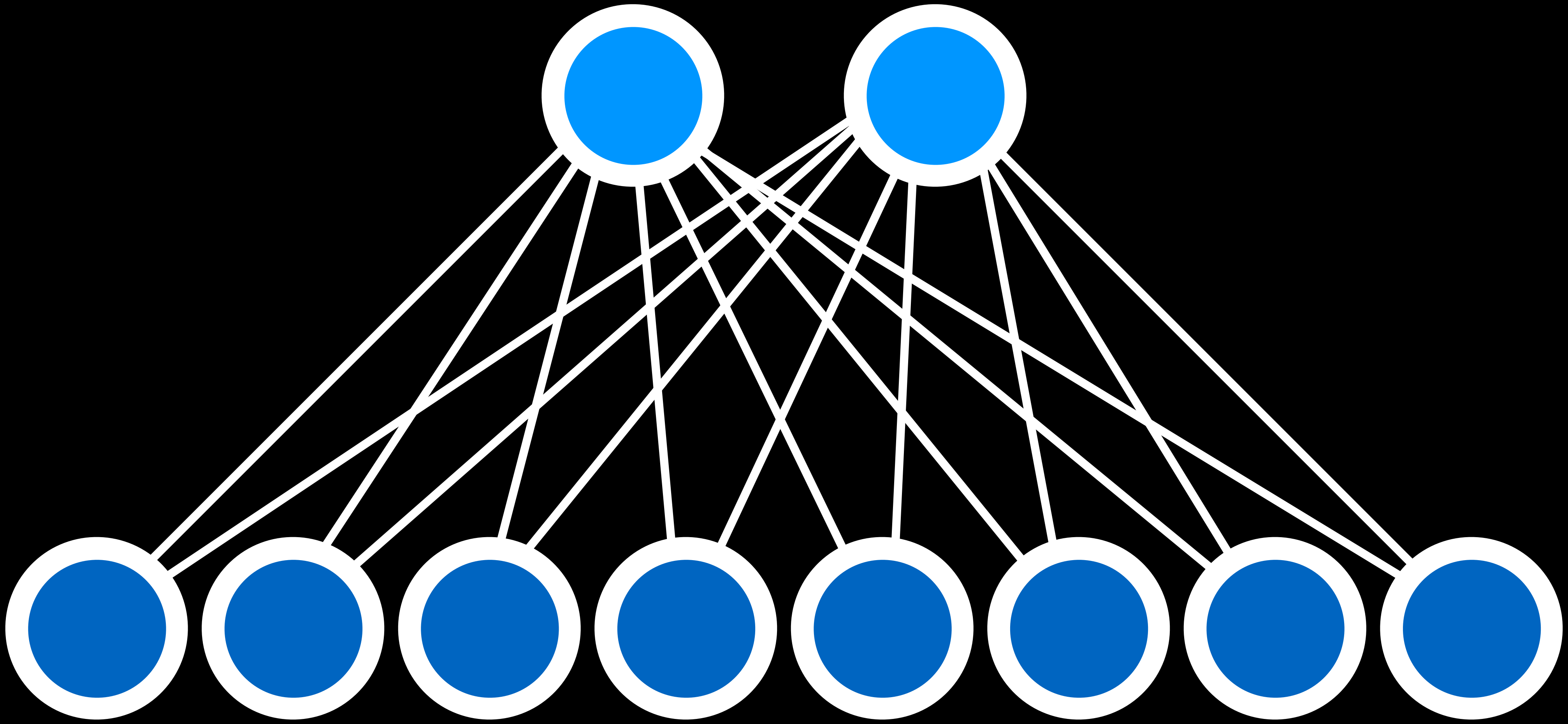


NetNORAD

The network fault detector

github.com/facebook/UdpPinger

Ping all the things!

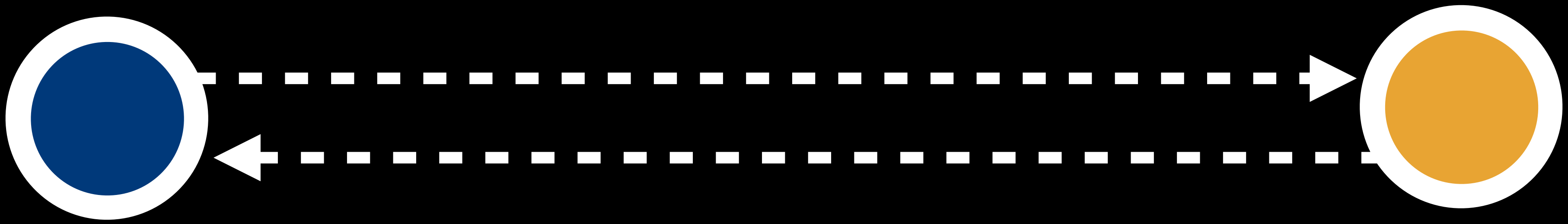


Run pingers on some machines
Run responders on all machines
Collect and analyse data

Evolution

- Run /bin/ping from a python agent
- Raw Sockets, Fast TCP Probes
- Raw Sockets, Fast ICMP Probes
- UDP Probes and Responder
- UDP Probes and Responder + Fast ICMP Probes

Ping Pong



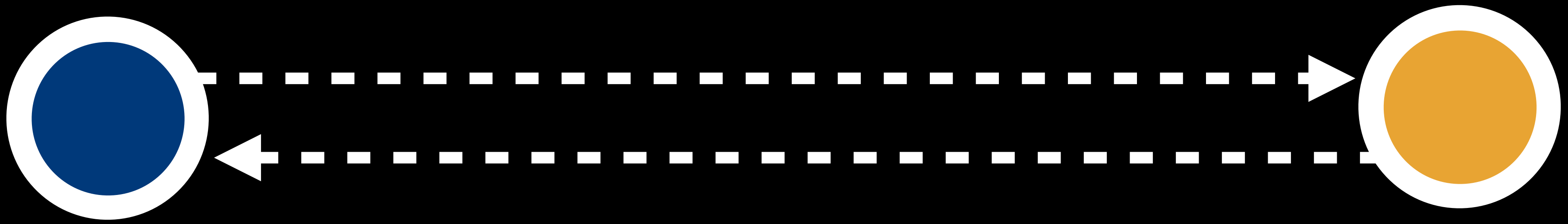
Pingers

- Send UDP and ICMP probes to target list
- Timestamp & Log results
- High ping-rate (up to 1Mpps)
- Set DSCP marking

Responders

- Receive/Reply to probe
- Timestamp
- Low load: thousands of pps
- Reflect DSCP value back

Why UDP?



- No TCP RST packets
- Efficient ECMP coverage
- Extensible

Probe Structure

Signature

Send Time

Receive Time

Response Time

Traffic Class

NetNORAD

Ping and Process Data

github.com/facebook/UdpPinger

© 2016 Facebook | Dublin | Credits: Photo by Jose Leitao https://creativecommons.org/licenses/by/3.0/

Challenges

Tens of thousands of targets

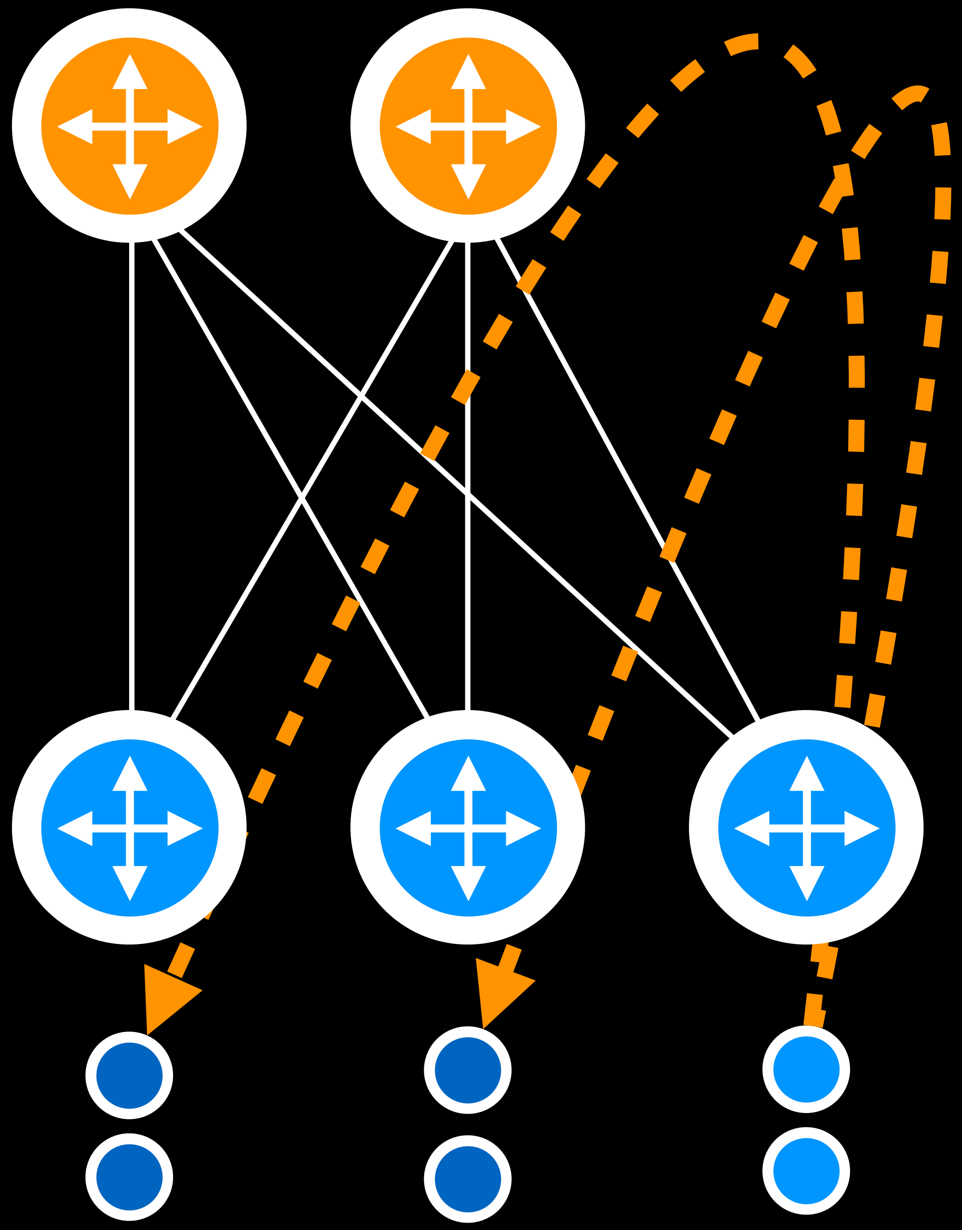
Hundreds of pingers

Lots of data to process

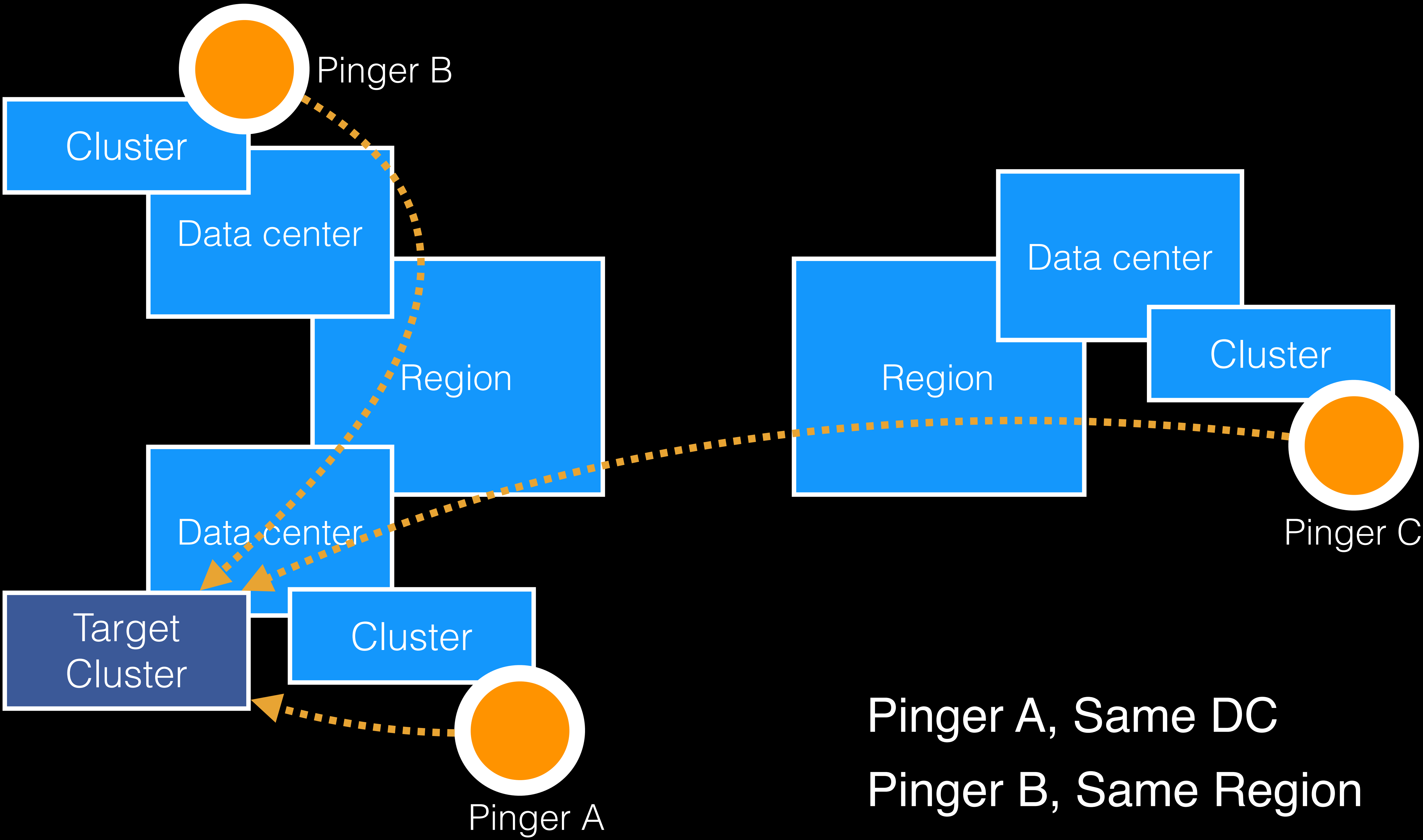
We really do not care about each host…
…The unit of interest is cluster health

Pinging inside clusters

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



Pinging the clusters



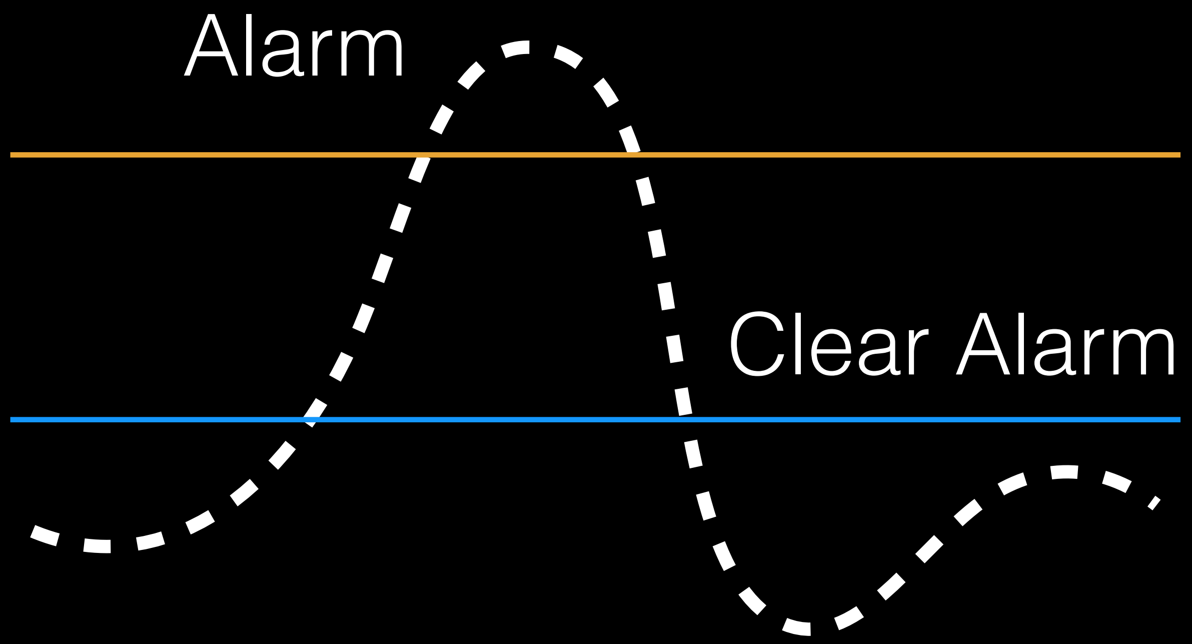
Pinger A, Same DC

Pinger B, Same Region

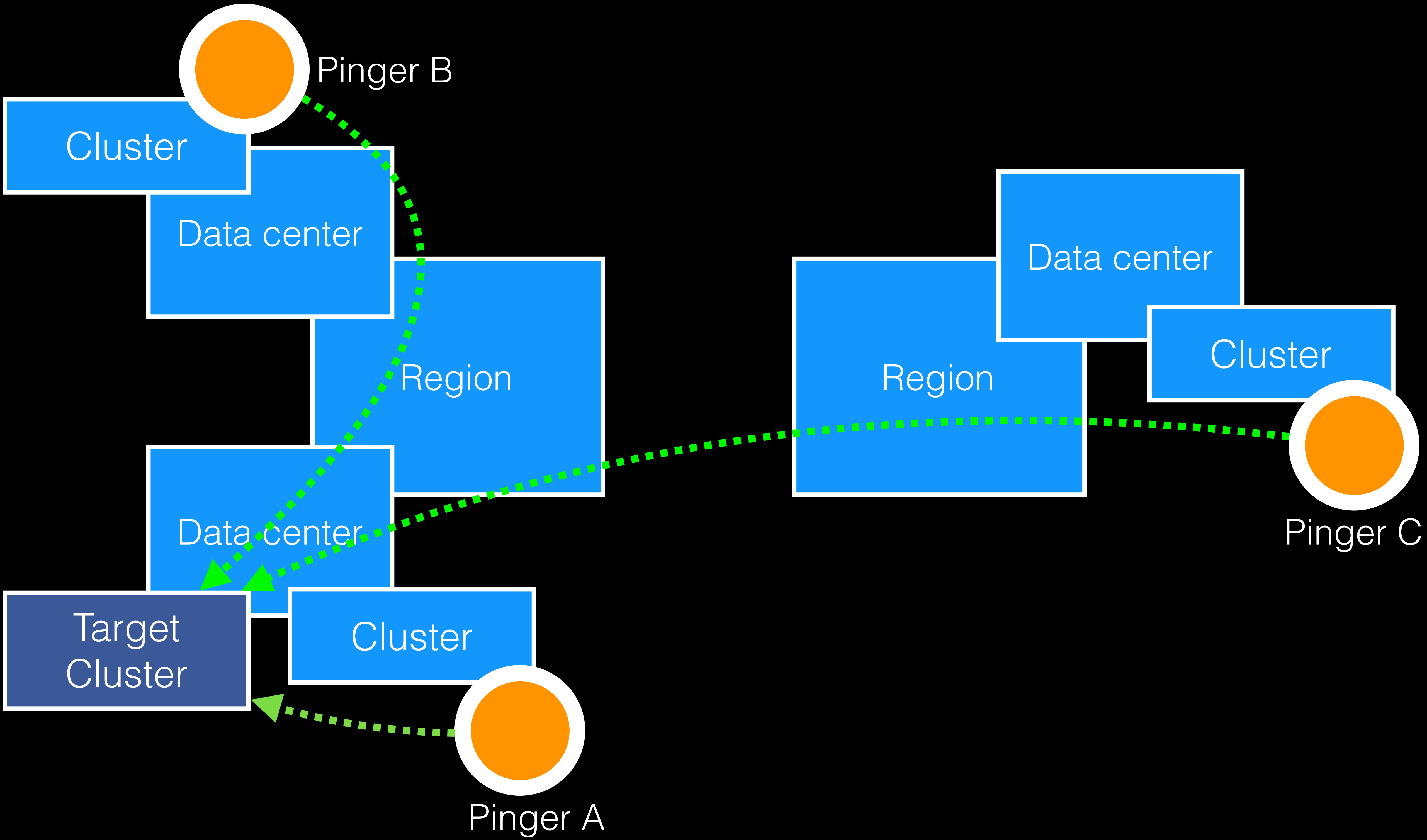
Pinger C, Outside of region

Alarming on Loss

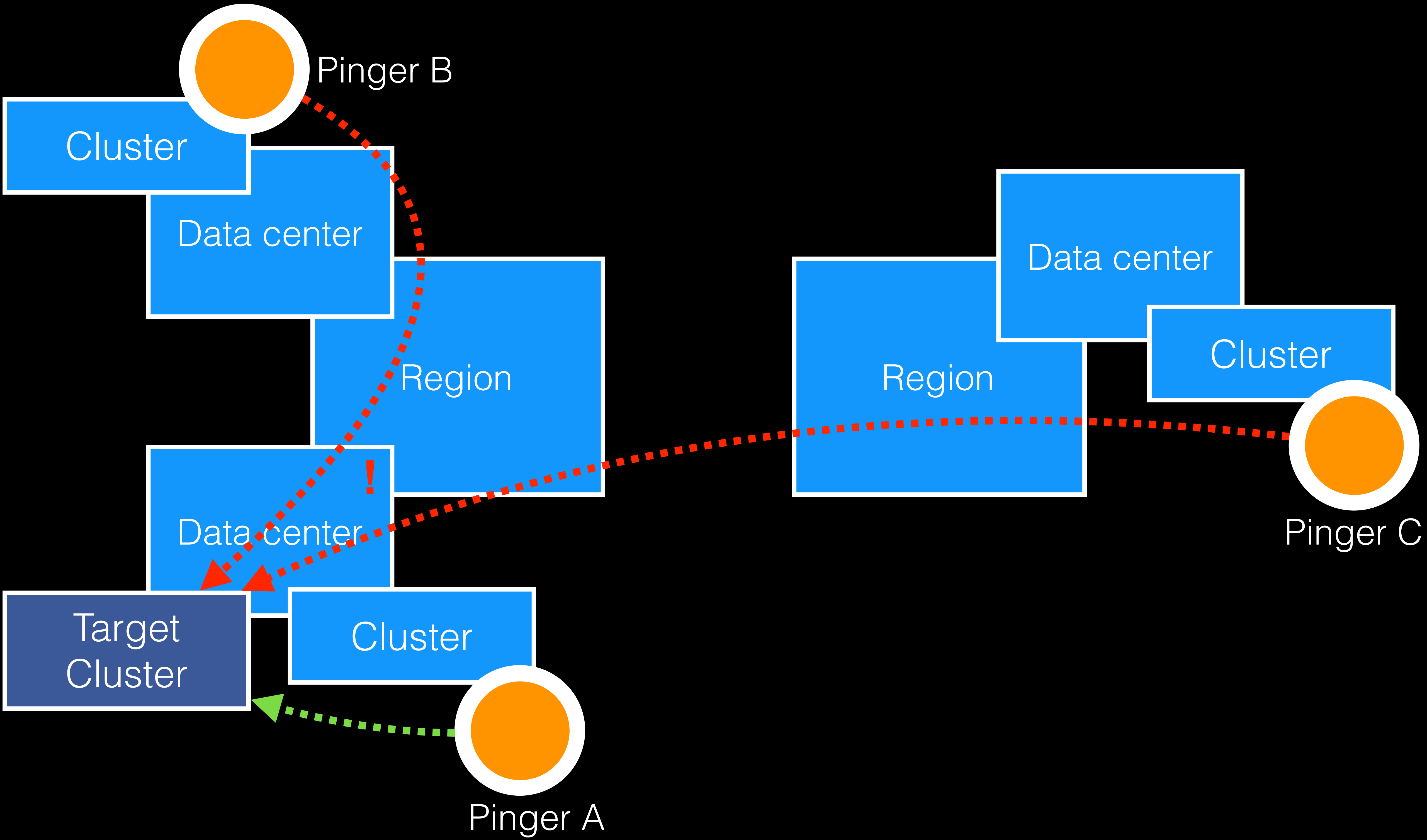
- Build packet loss time-series
- Track percentiles
- Time to detect loss? 20 Seconds



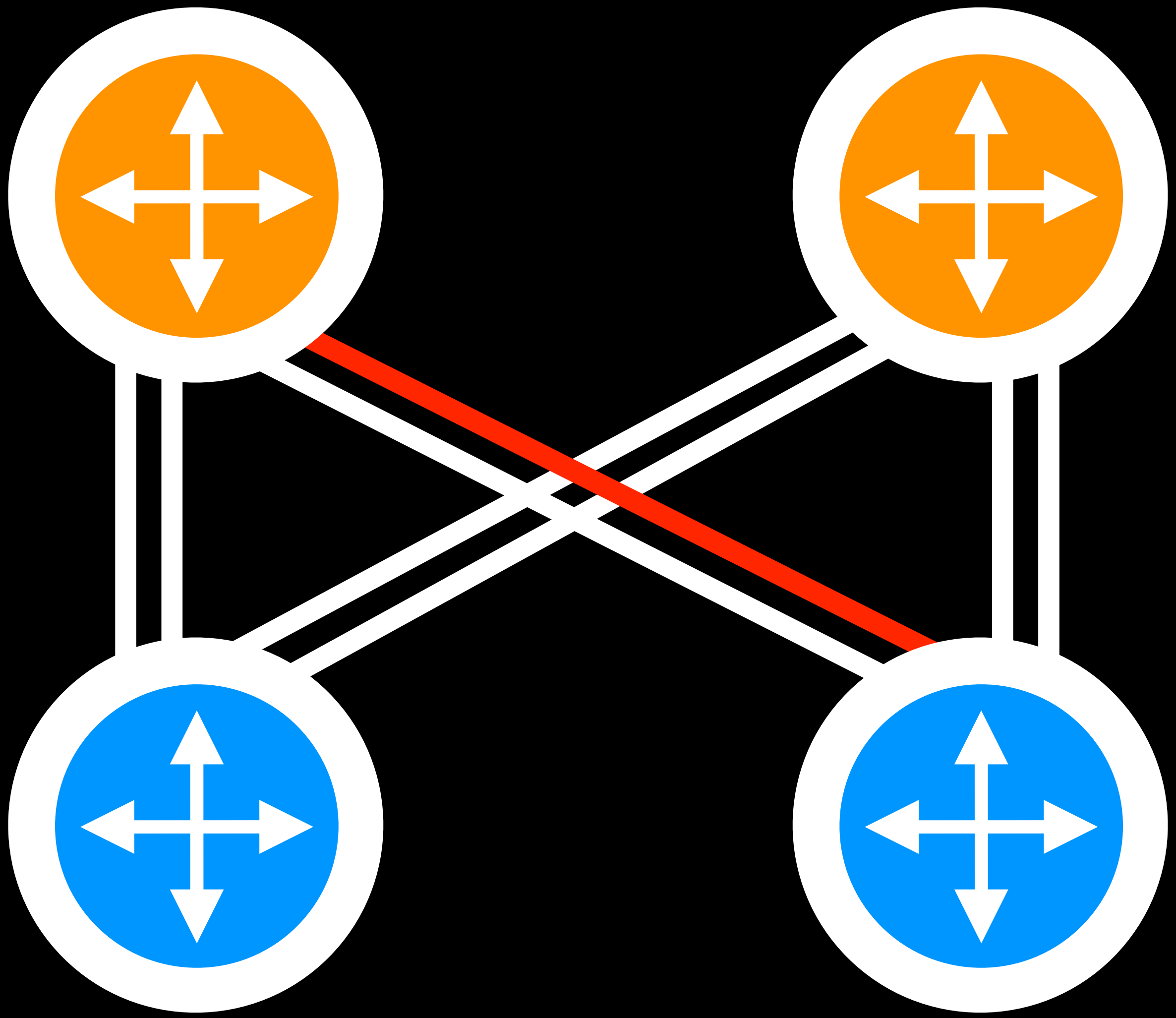
Finding the problem



Finding the problem



Finding the problem



Each layer of the network contains many devices
...and many more links

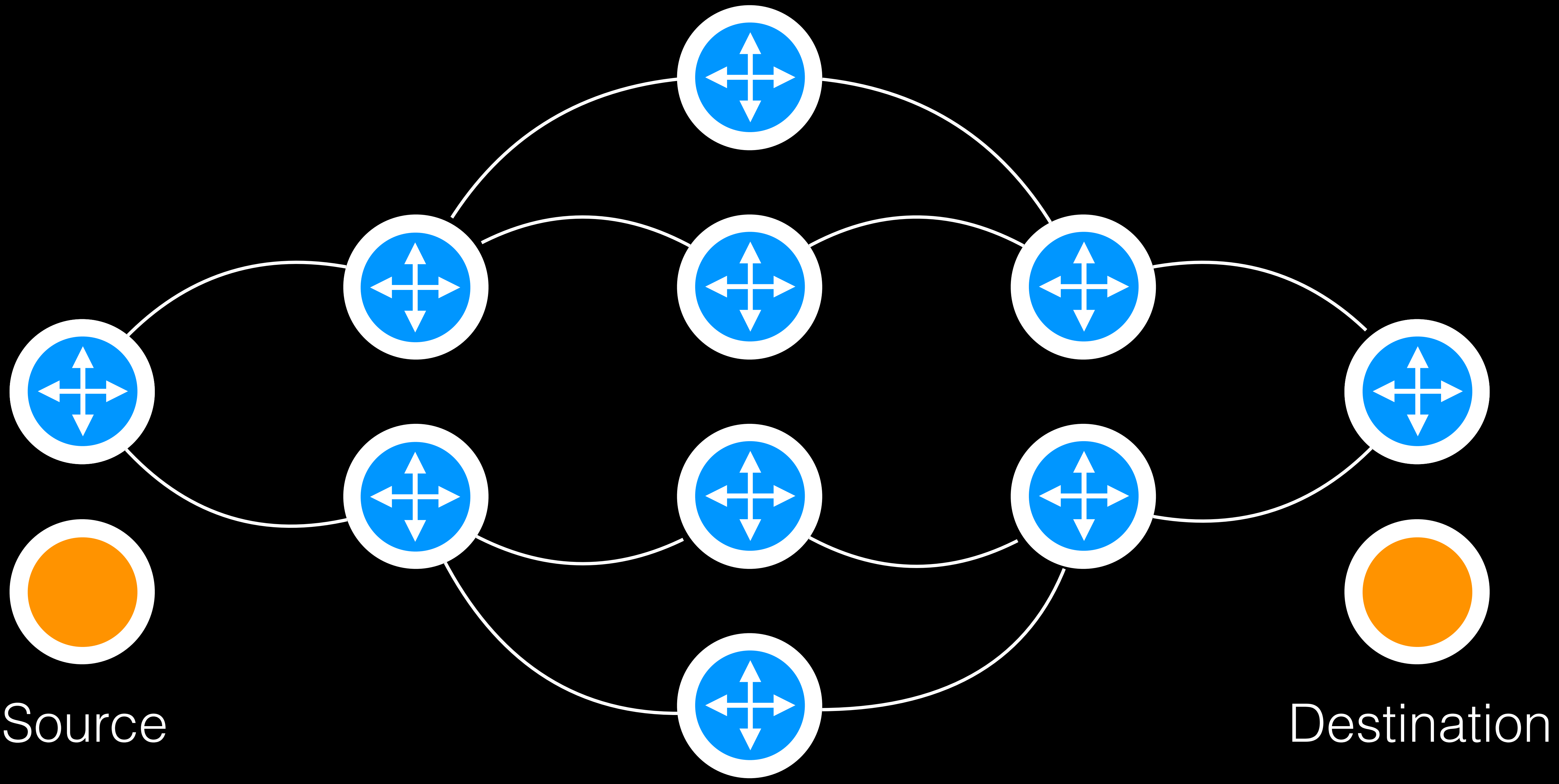
fbtracert

Isolating Network Faults

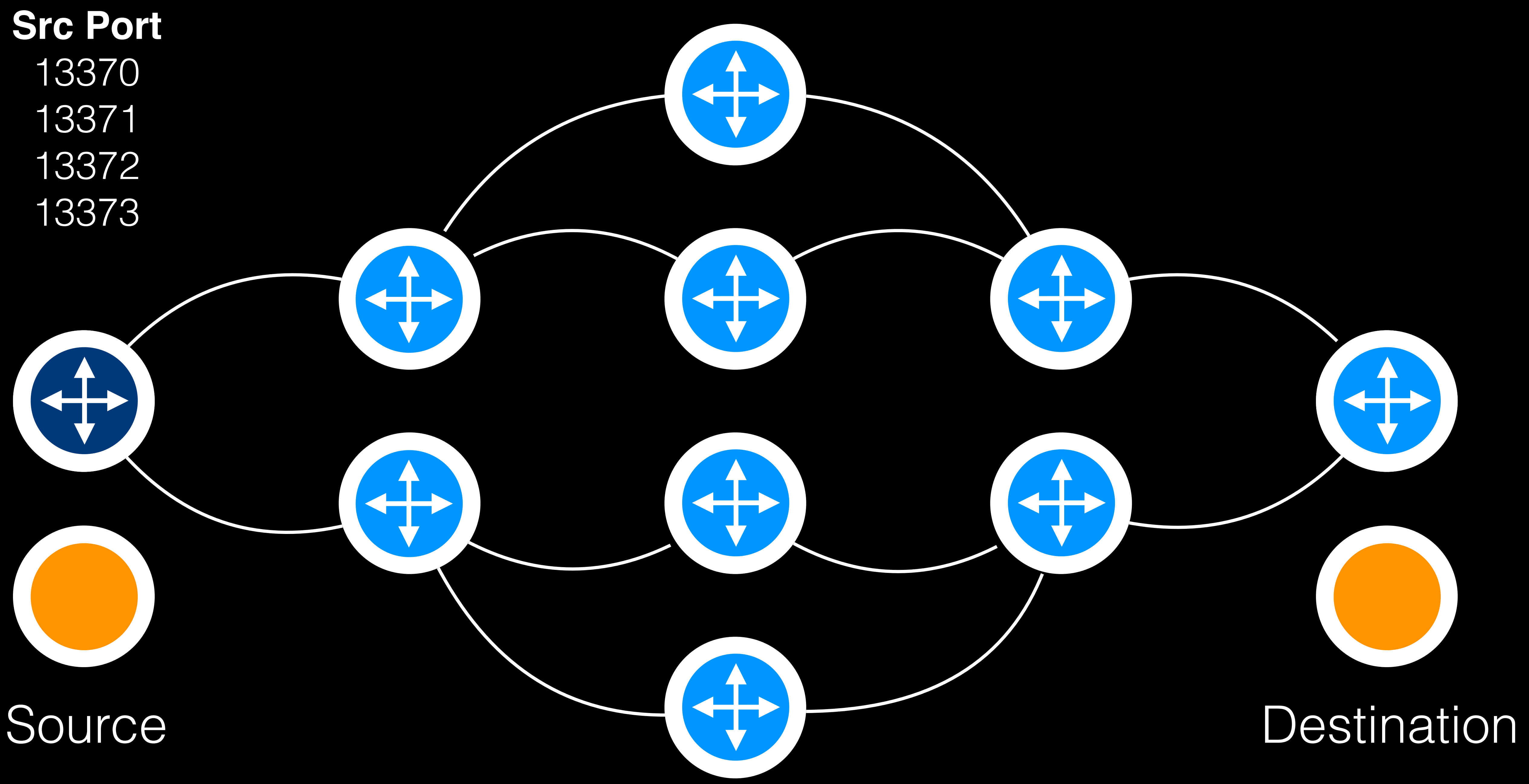
github.com/facebook/fbtracert

© 2016 Facebook | Dublin | Credits: Photo by Jose Leitao https://creativecommons.org/licenses/by/3.0/

fbtracert



fbtracert

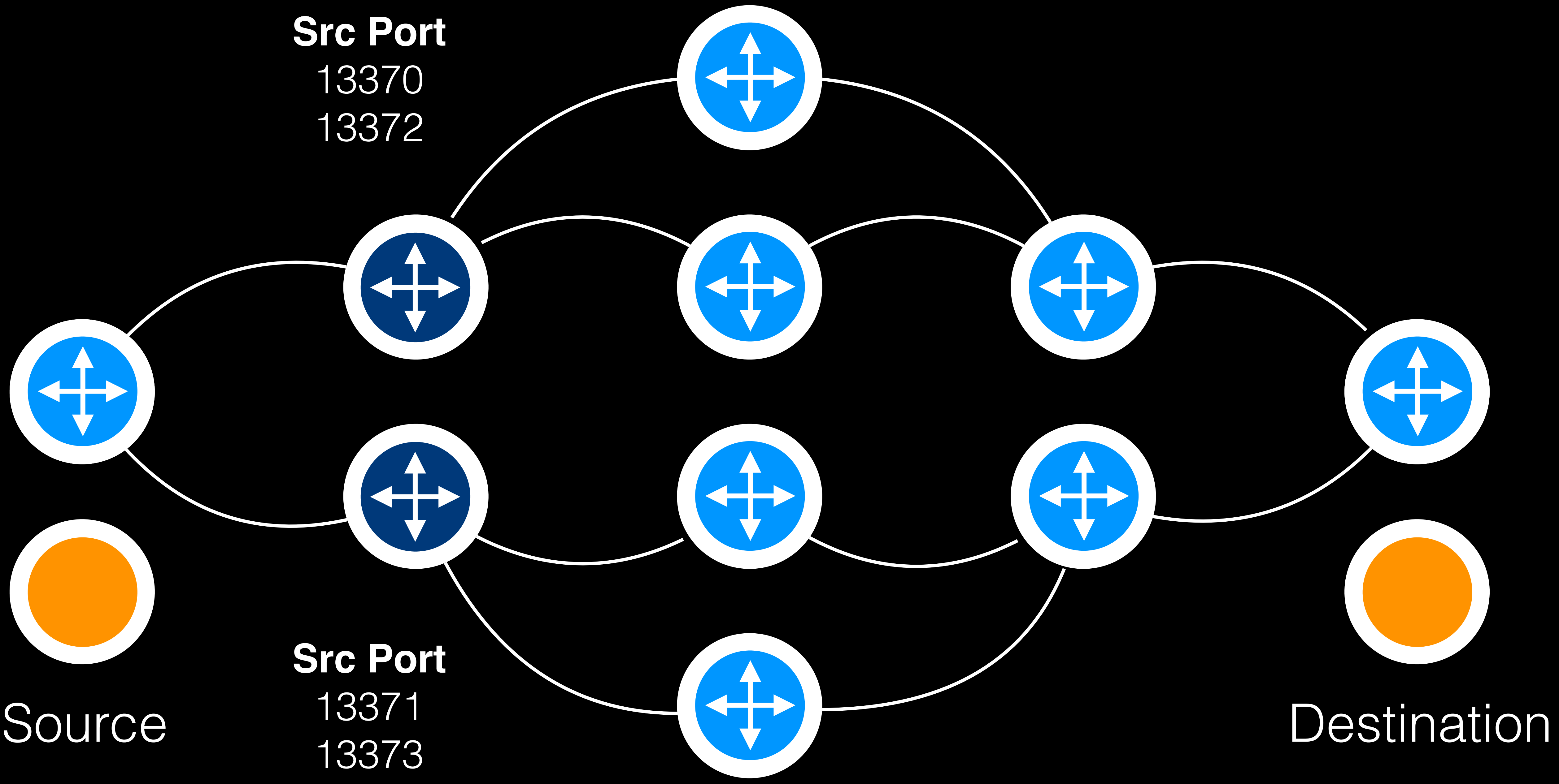


Src Port
13370
13371
13372
13373

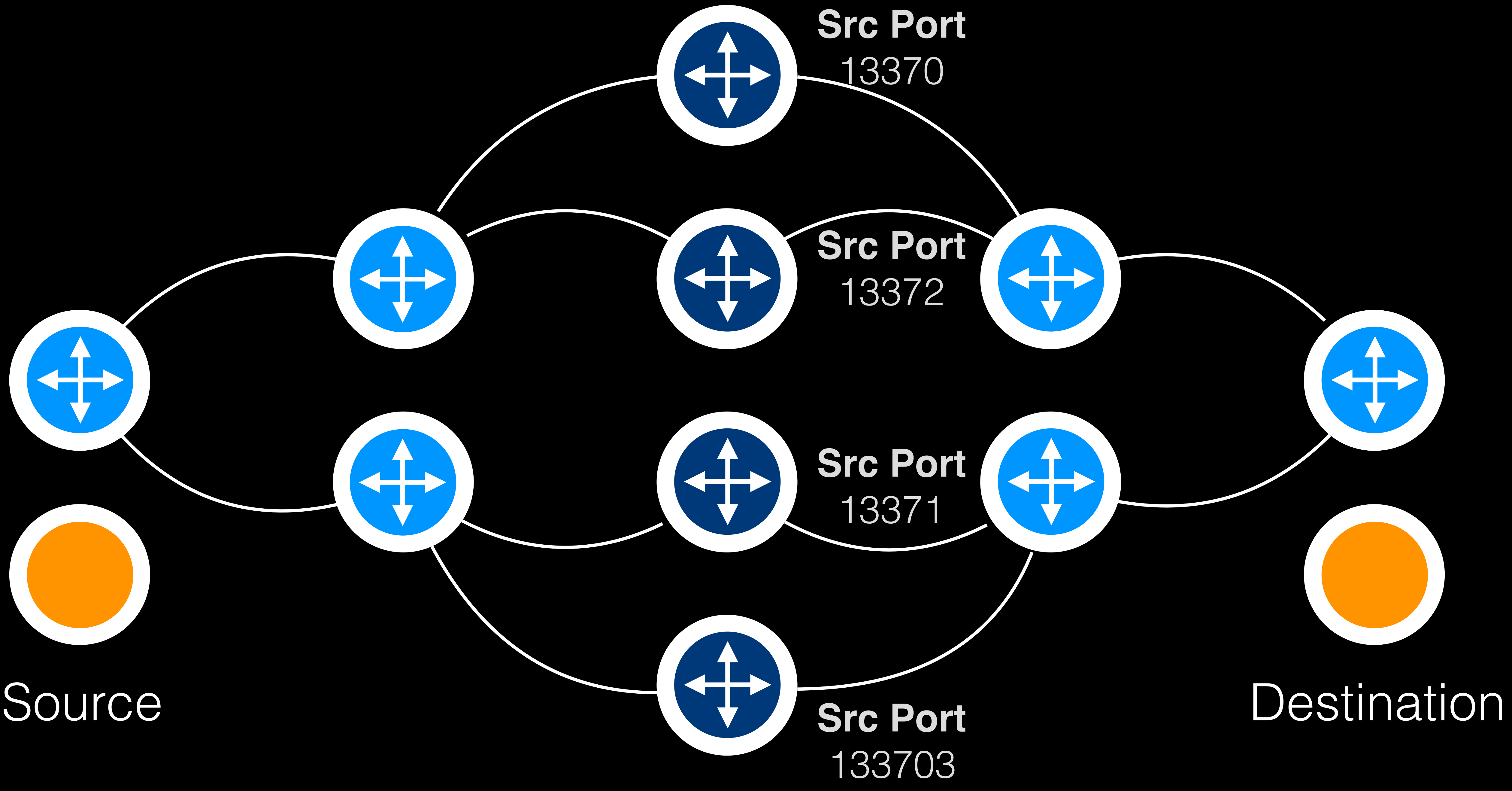
Source

Destination

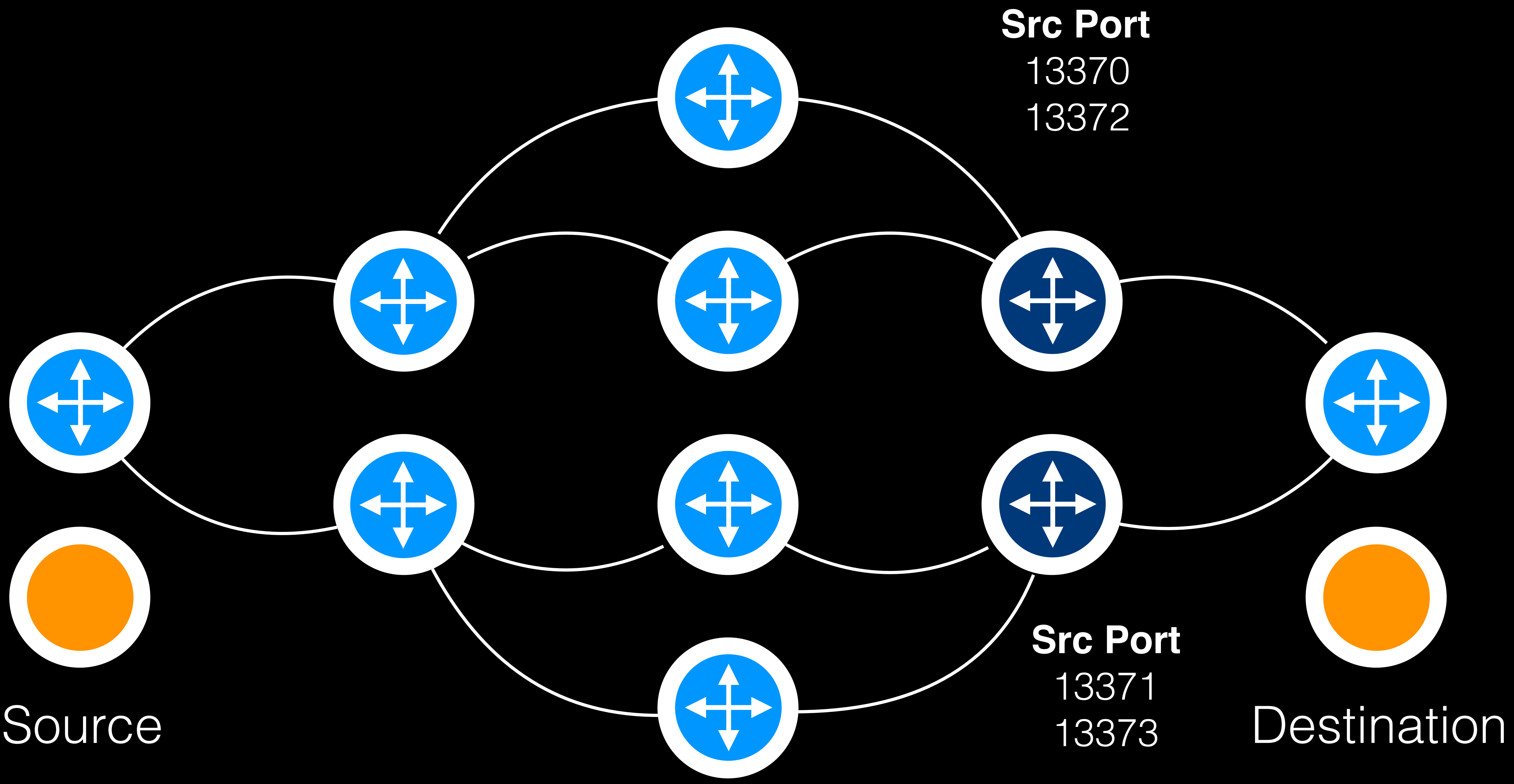
fbtracert


Src Port
13370
13372
Src Port
13371
13373
Source
Destination

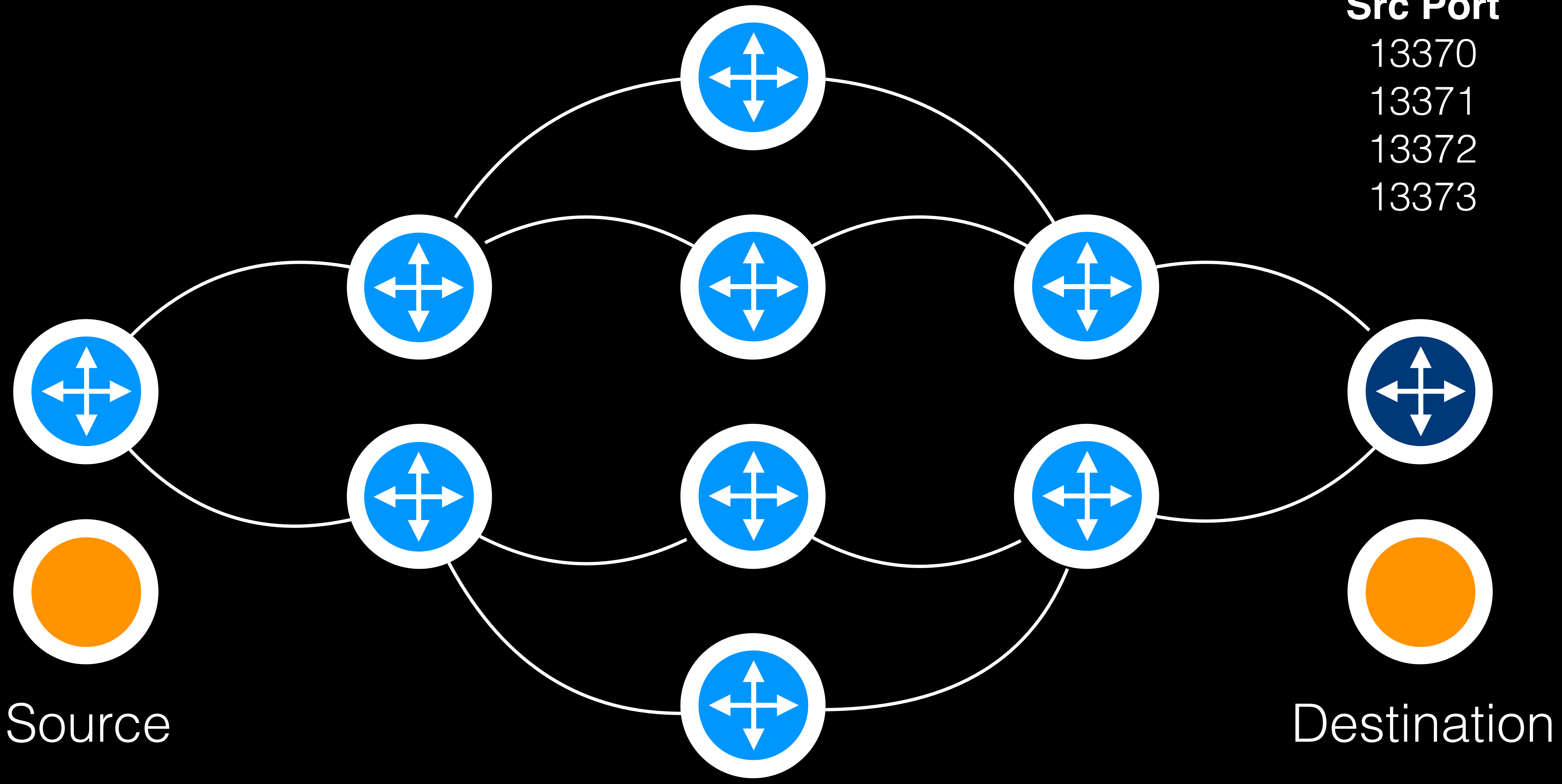
fbtracert


Src Port
13370
Src Port
13372
Src Port
13371
Src Port
133703
Source
Destination

fbtracert


Src Port
13370
13372
Src Port
13371
13373
Source
Destination

fbtracert



Src Port
13370
13371
13372
13373

DC Network Fault Isolation

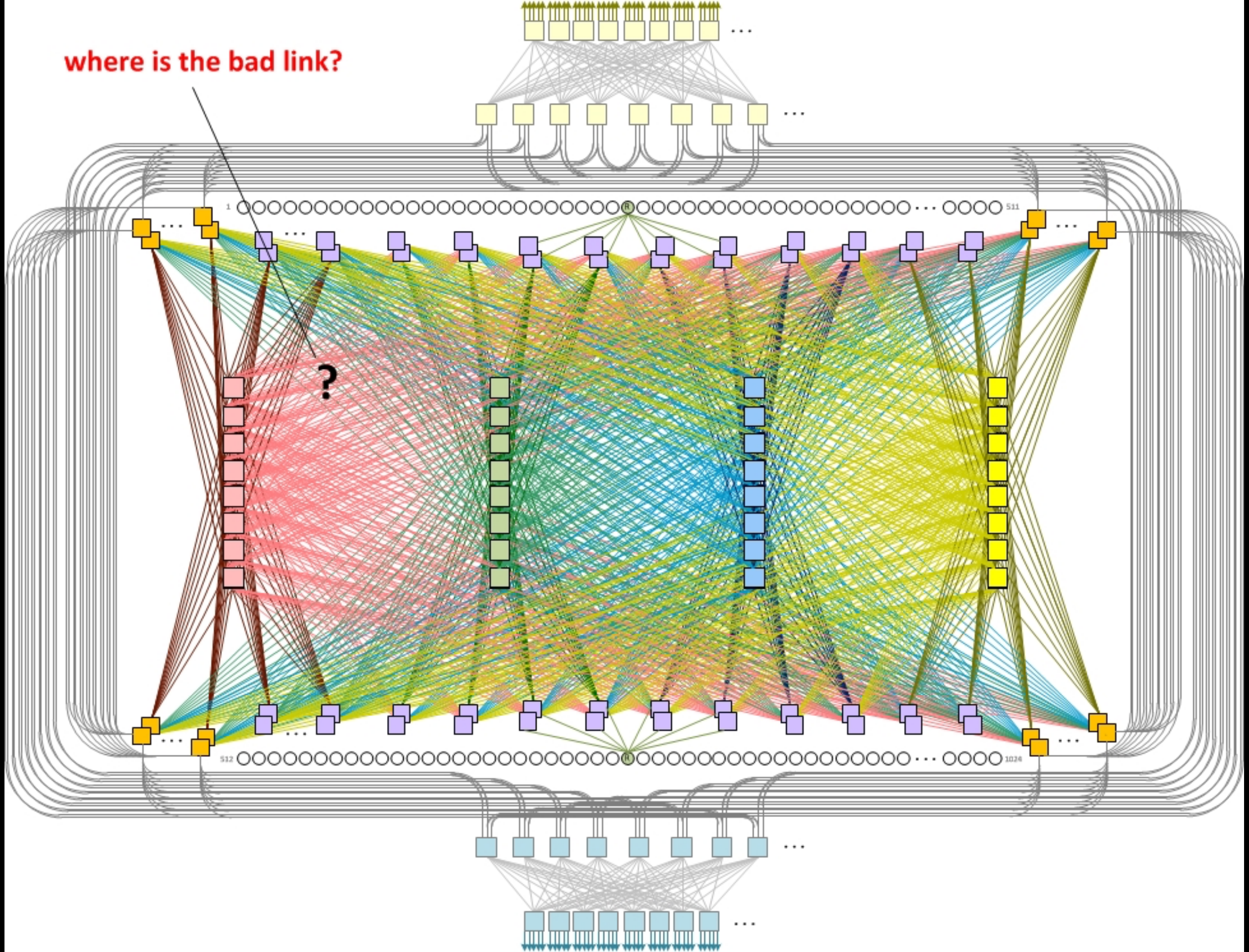
Isolating Network Faults

© 2016 Facebook | Dublin | Credits: Photo by Jose Leitao https://creativecommons.org/licenses/by/3.0/

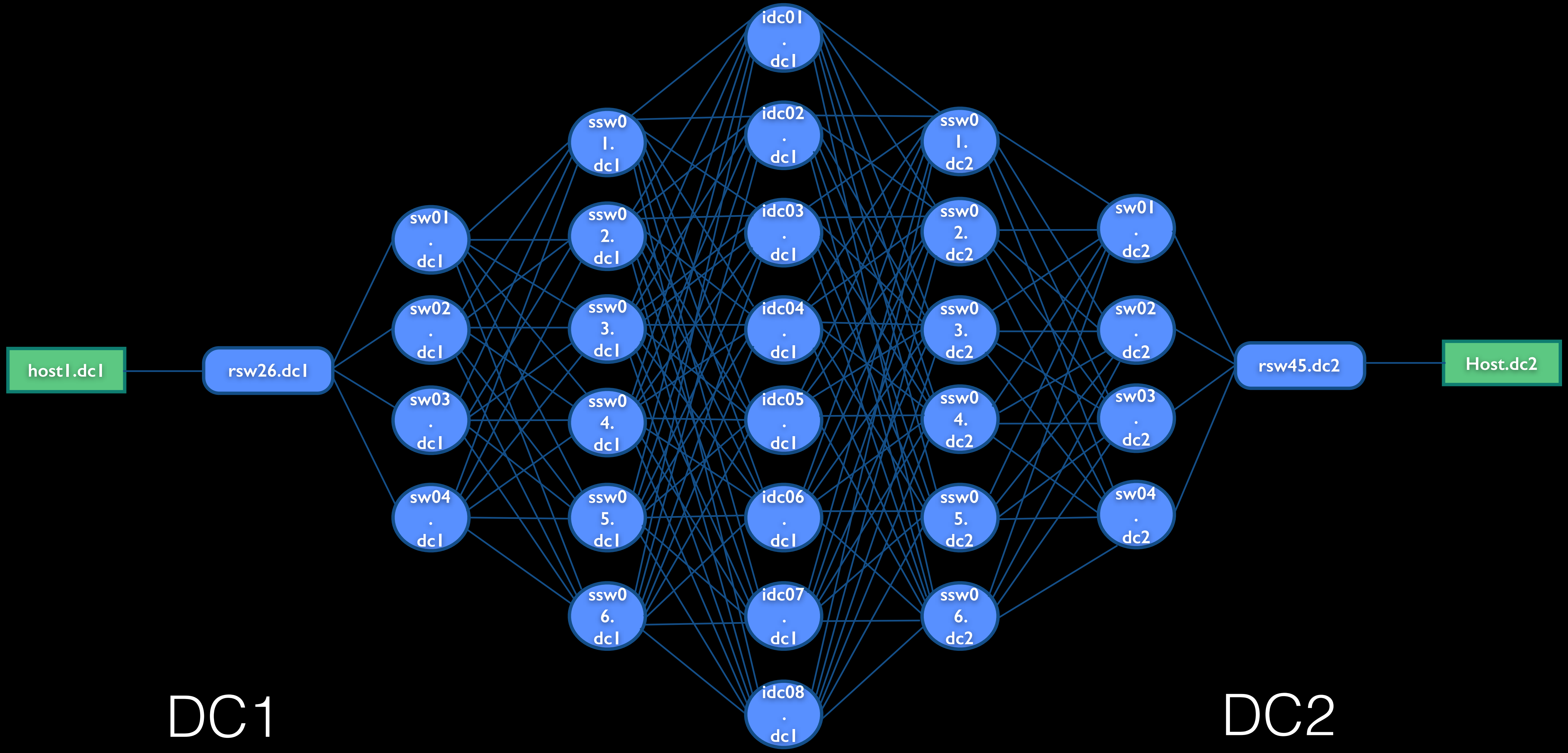
Big Fat Fabrics

- Over 1,000 L3 Links between devices in different DCs in the SAME Region

- We know there is loss between hosts, but where?



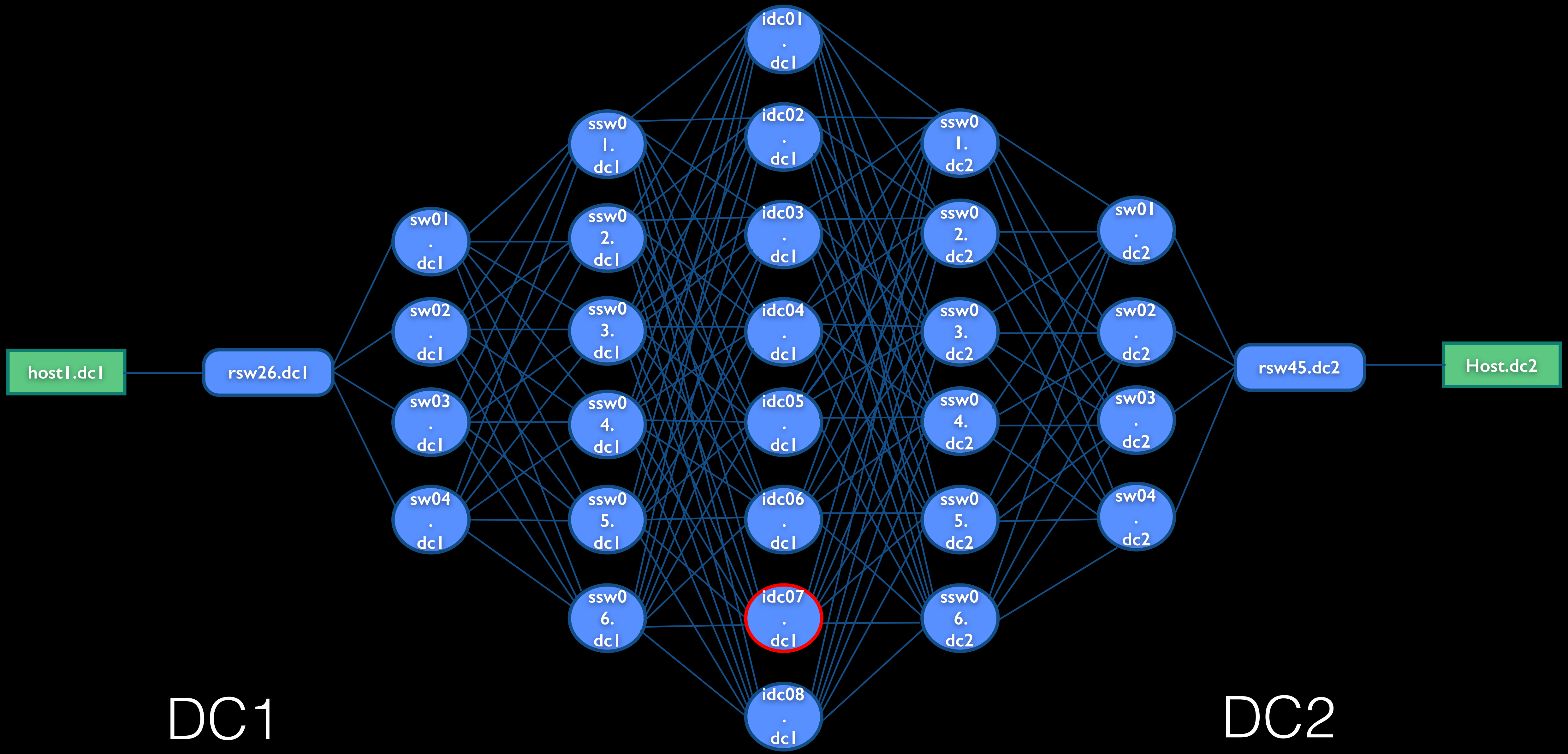
DC Network Fault Isolation



DC1

DC2

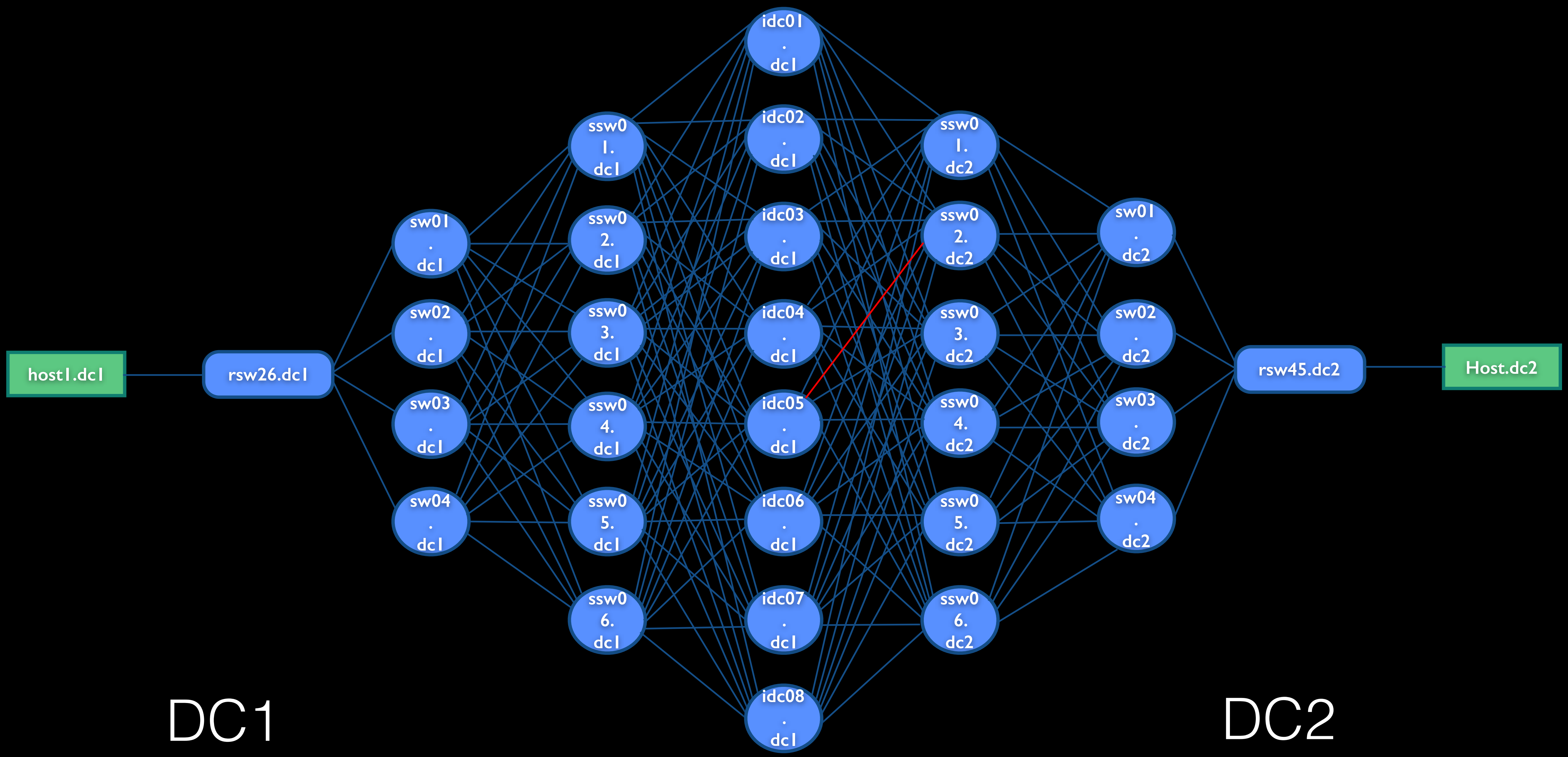
Bad Fabric Card



DC1

DC2

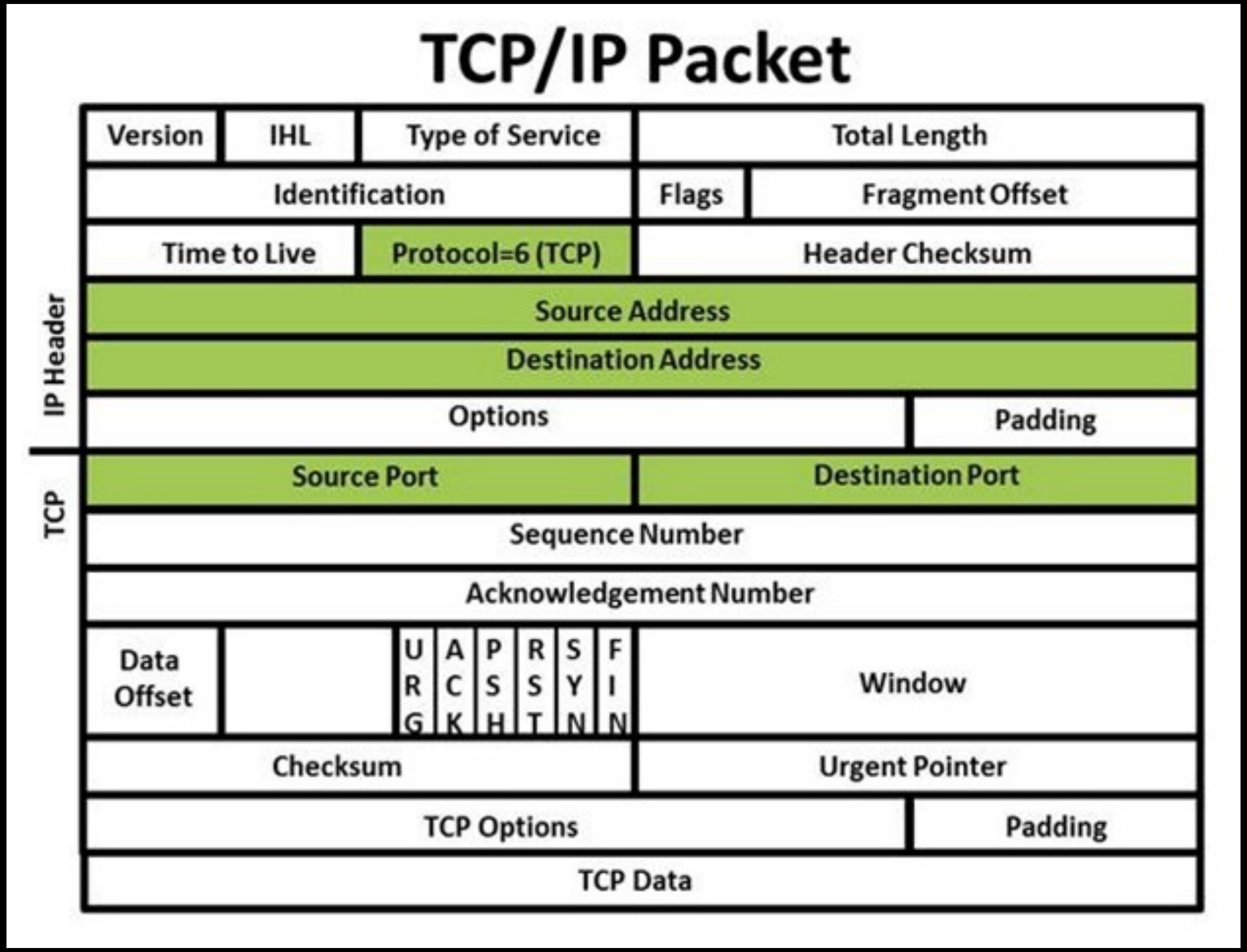
Bad Link



DC1

DC2

ECMP Packet Hashing



TCP/IP Packet

IP Header

Version | IHL | Type of Service | Total Length

Identification | Flags | Fragment Offset

Time to Live | Protocol=6 (TCP) | Header Checksum

Source Address

Destination Address

Options | Padding

TCP

Source Port | Destination Port

Sequence Number

Acknowledgement Number

Data Offset | | URG | ACK | PSH | RST | SYN | FIN | Window

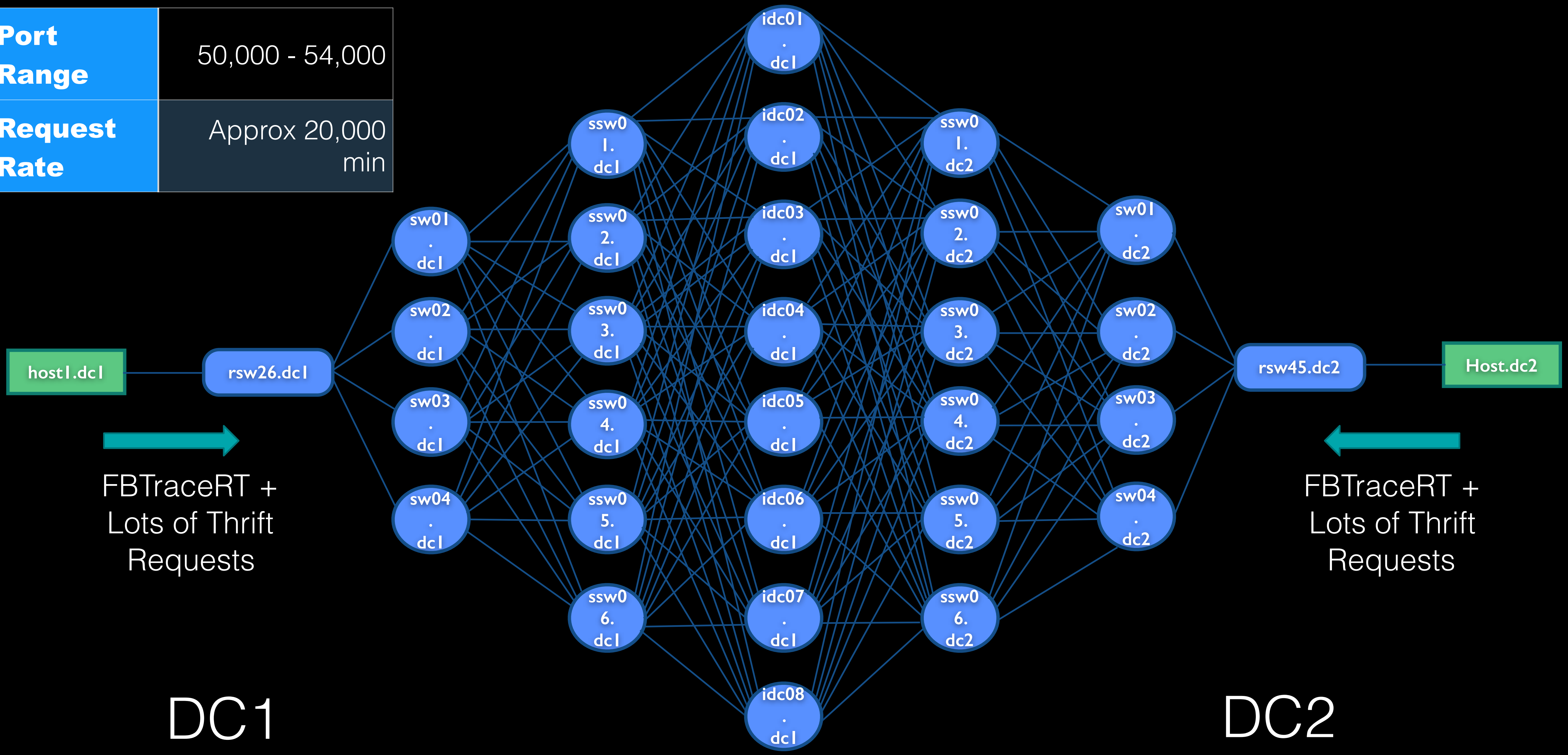
Checksum | Urgent Pointer

TCP Options | Padding

TCP Data

DC Network Fault Isolation

Port Range	50,000 - 54,000
Request Rate	Approx 20,000 min



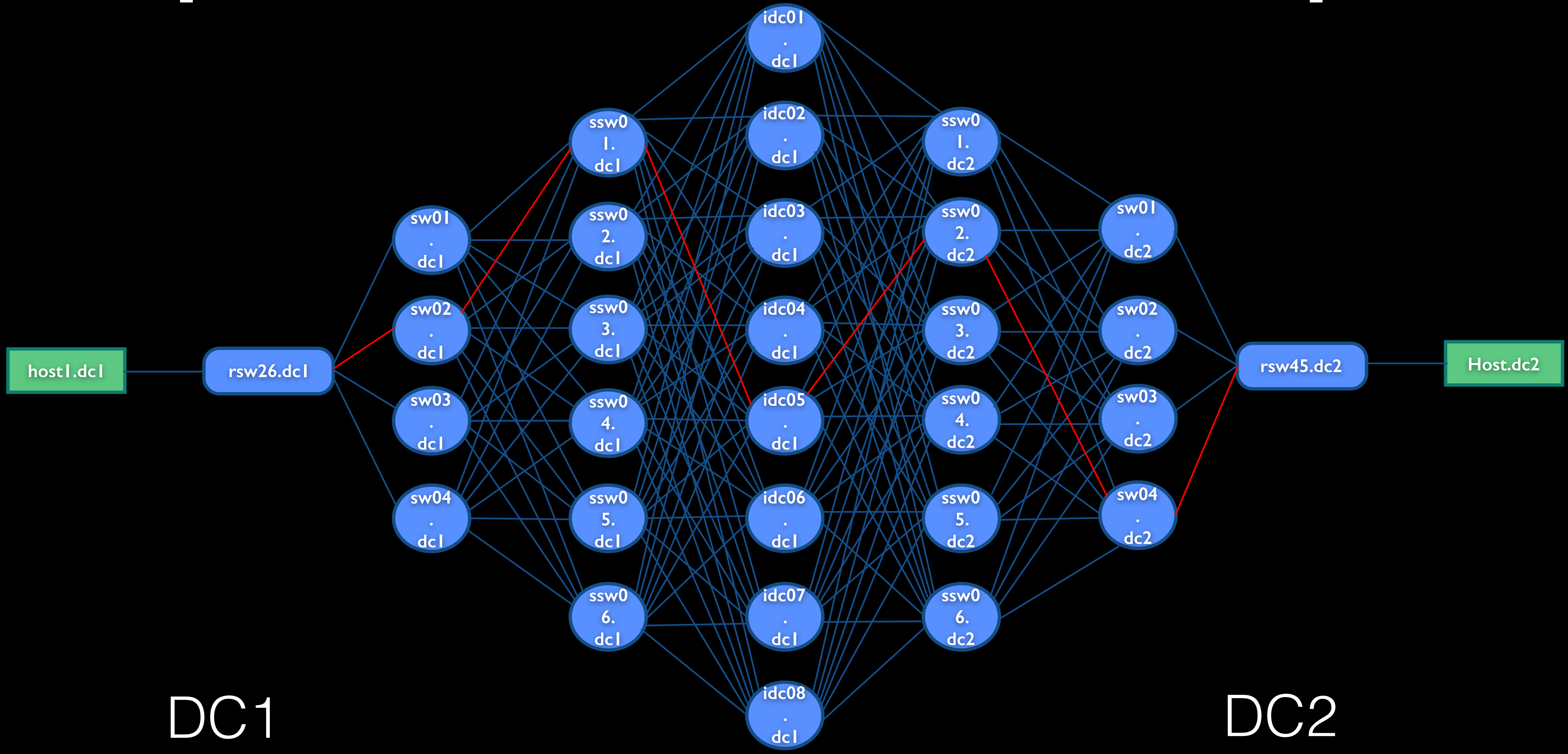
FBTraceRT + Lots of Thrift Requests

FBTraceRT + Lots of Thrift Requests

DC1

DC2

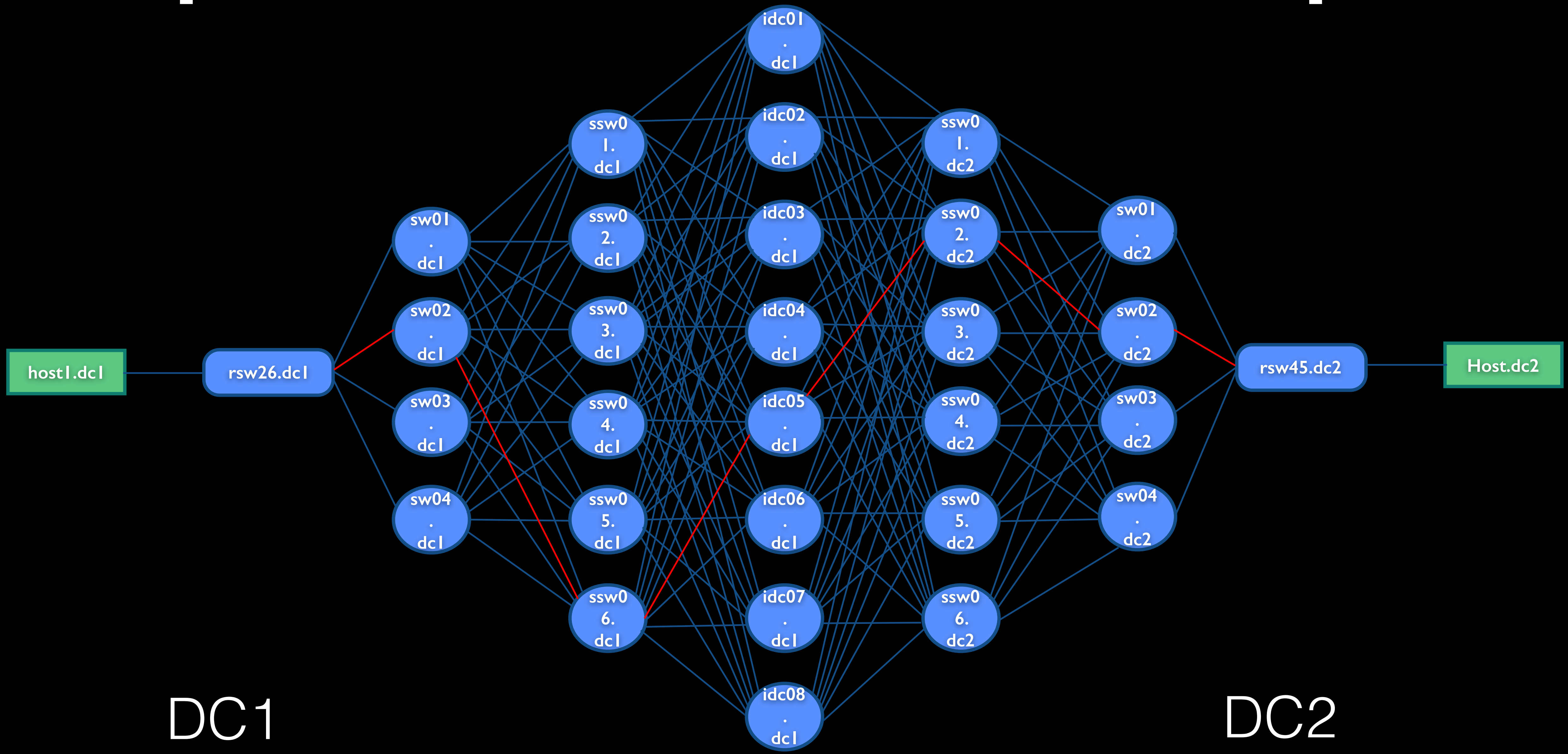
Map The Loss for Each Request



DC1

DC2

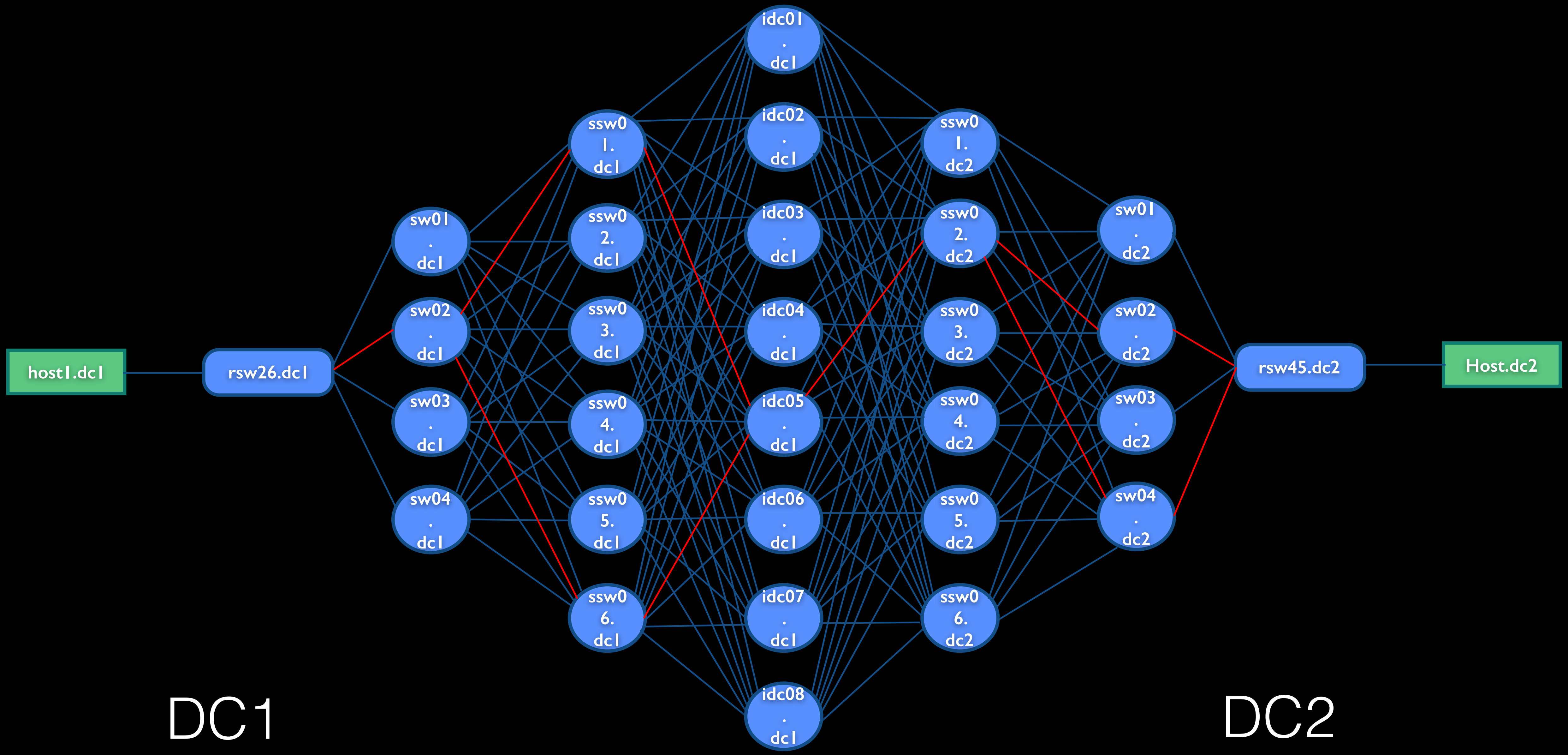
Map The Loss for Each Request



DC1

DC2

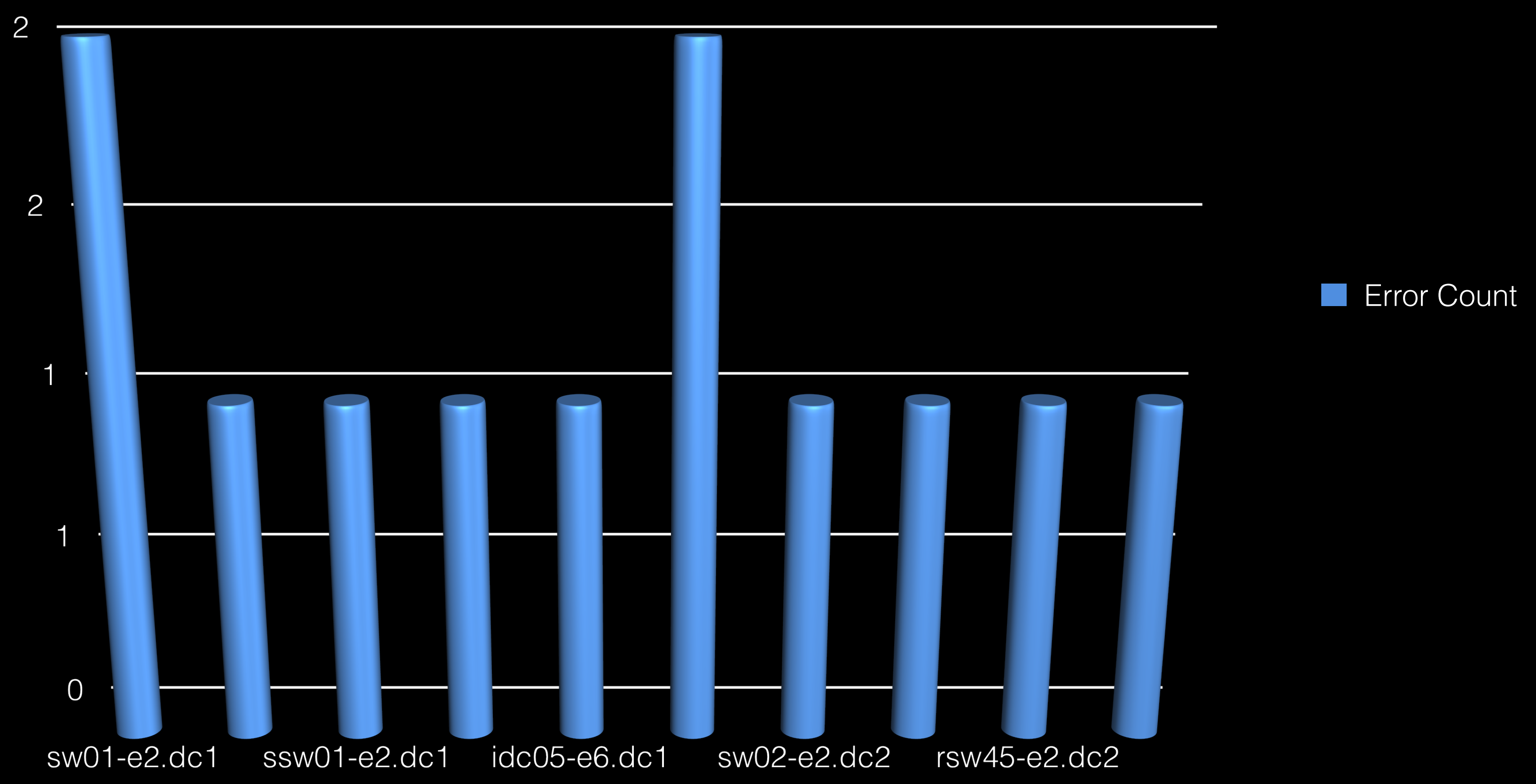
Overlay the Loss on Network Map



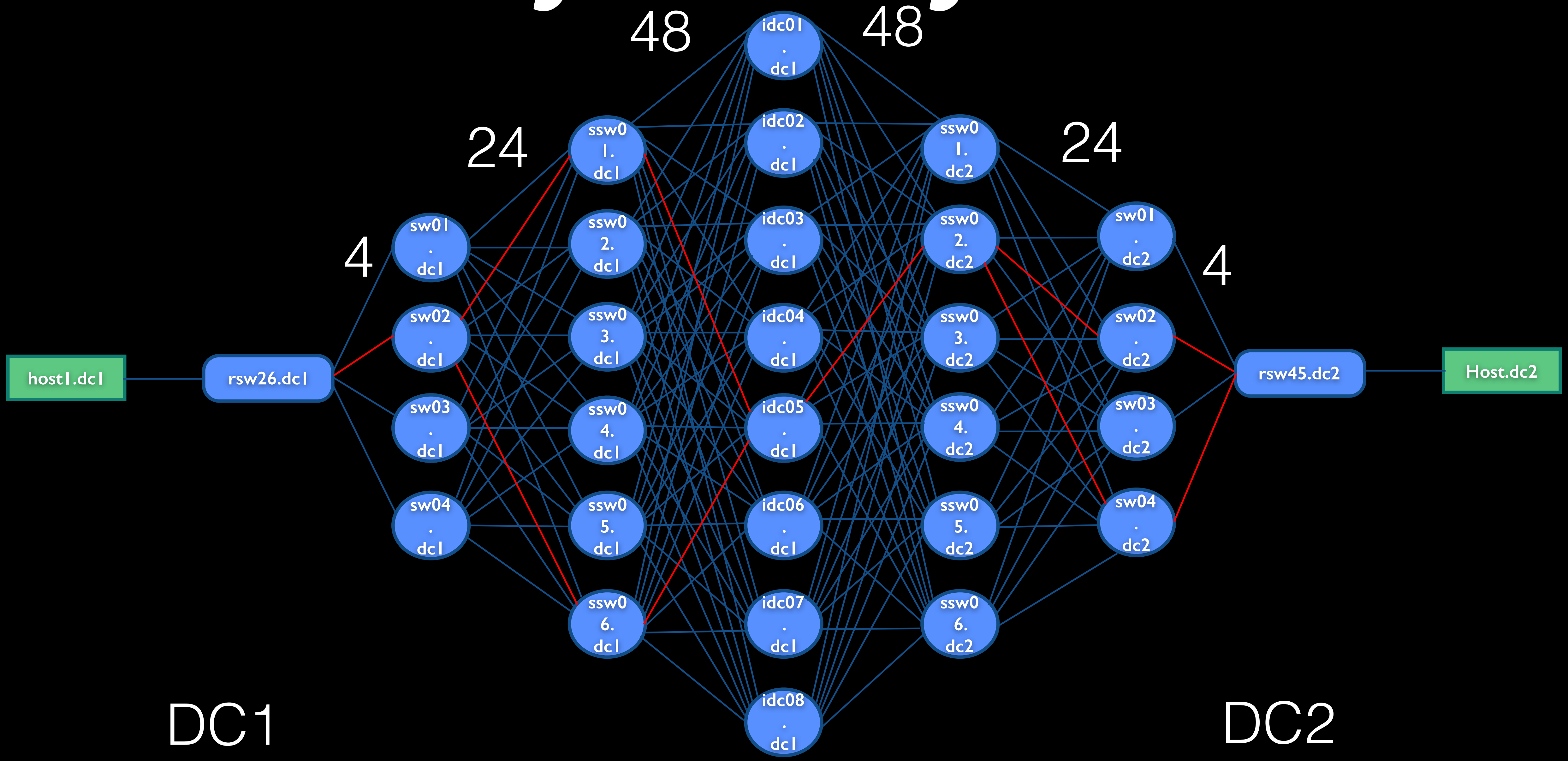
DC1

DC2

Analyze the Loss



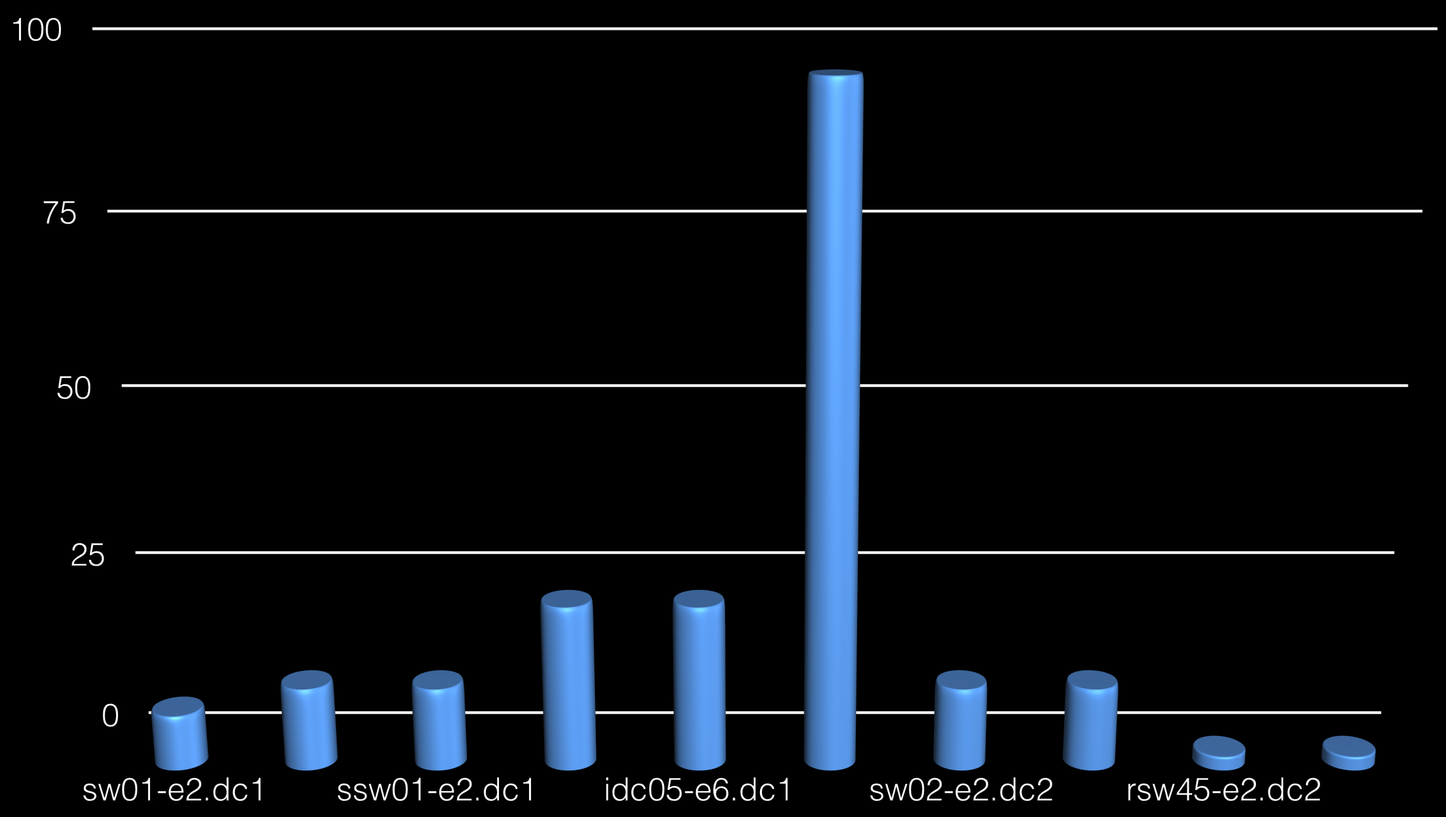
Network Symmetry

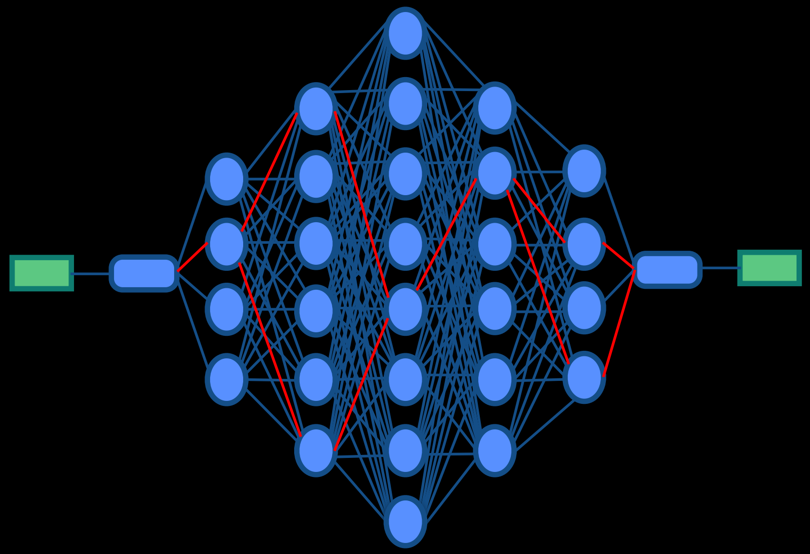


DC1

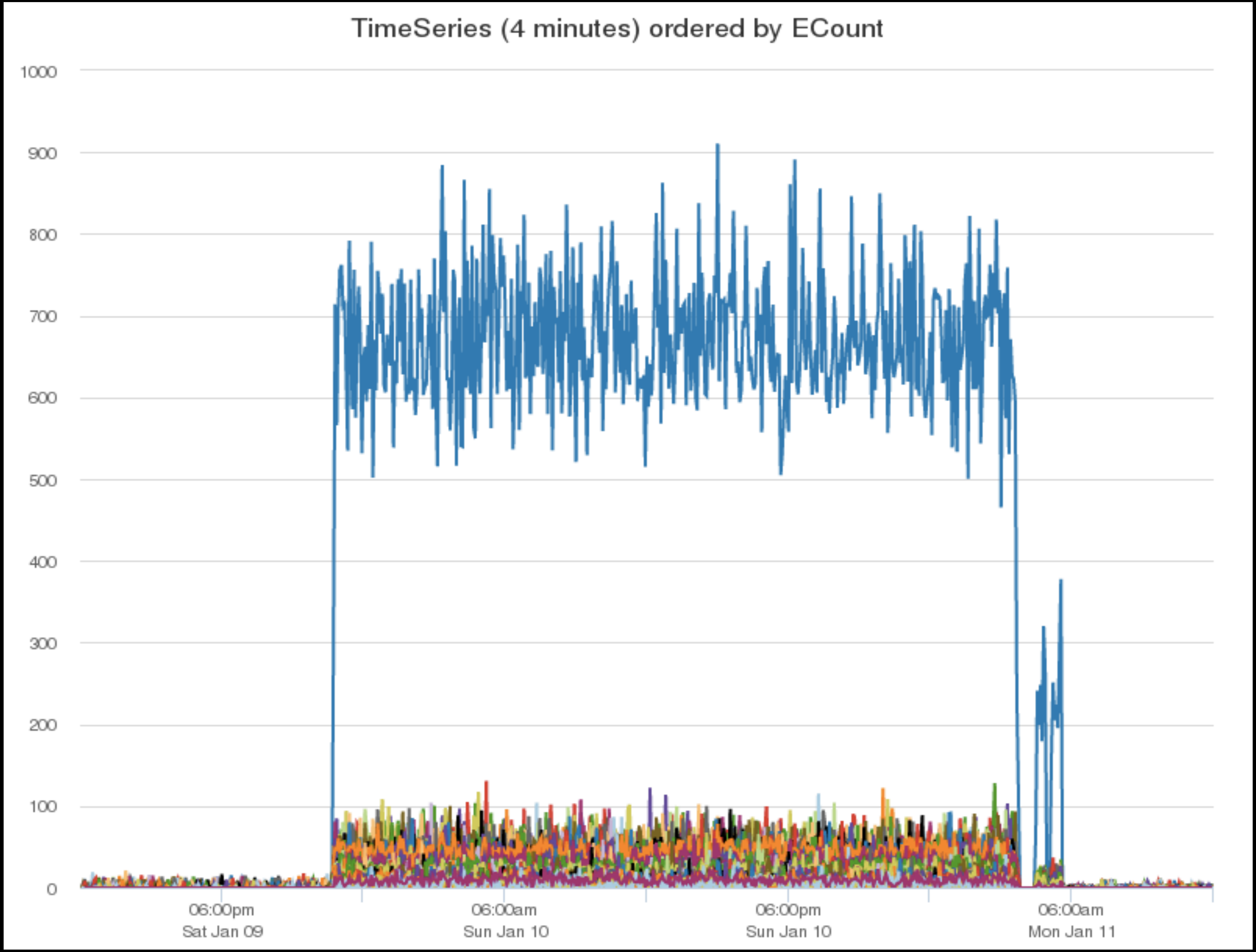
DC2

Clearer Signal





Fabric Grey Failure Detection



Conclusions

- Fault isolation is actively evolving
- Traceroute + probing approach is quite generic
- Limited by current hardware

101
WE ARE
TERRIBLE MAGICIANS
CODE WINS </br>
ARGUMENTS.
JELLO