Localizing packet loss

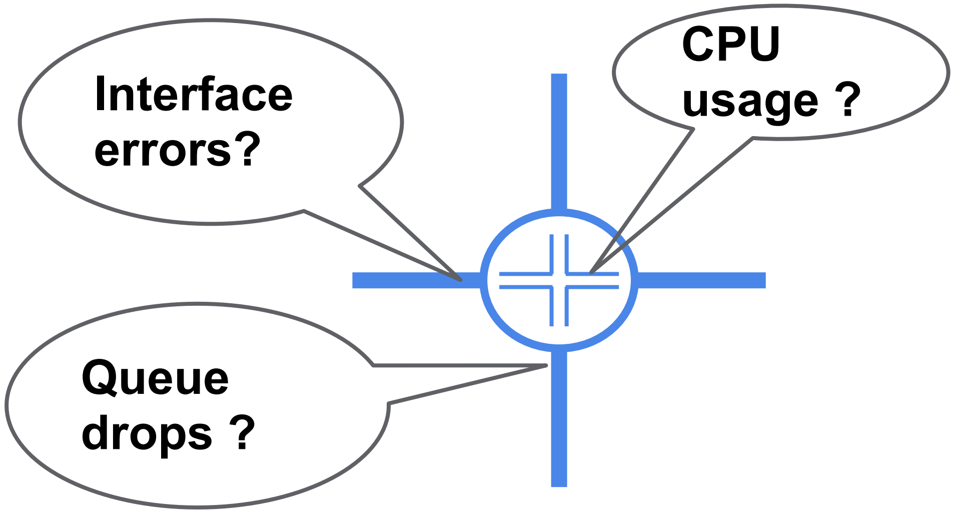
In a large complex network

Traditional network monitoring: White box

White box monitoring it's basically asking the device and monitor its vital parameters.

Unfortunately this is far from being good enough, too many times the device either 'lie' or not tell you the whole picture.

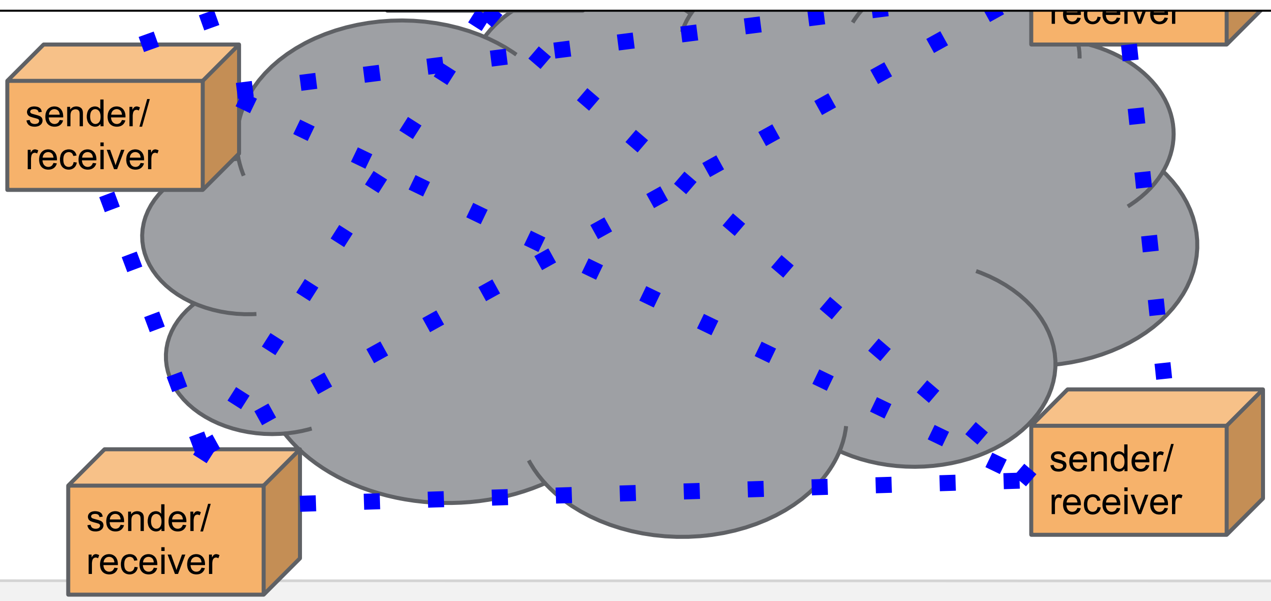
If we can't trust it, we need to test it.



Traditional network monitoring: Black box

Two major drawbacks:

- Only the best paths between the senders/receivers are monitored.
- It is hard to isolate a faulty element



How do we cover every component ?

not only the best paths

Exhaustive coverage.

We can't just rely on destination based routing, otherwise only the best paths between two locations would get tested.

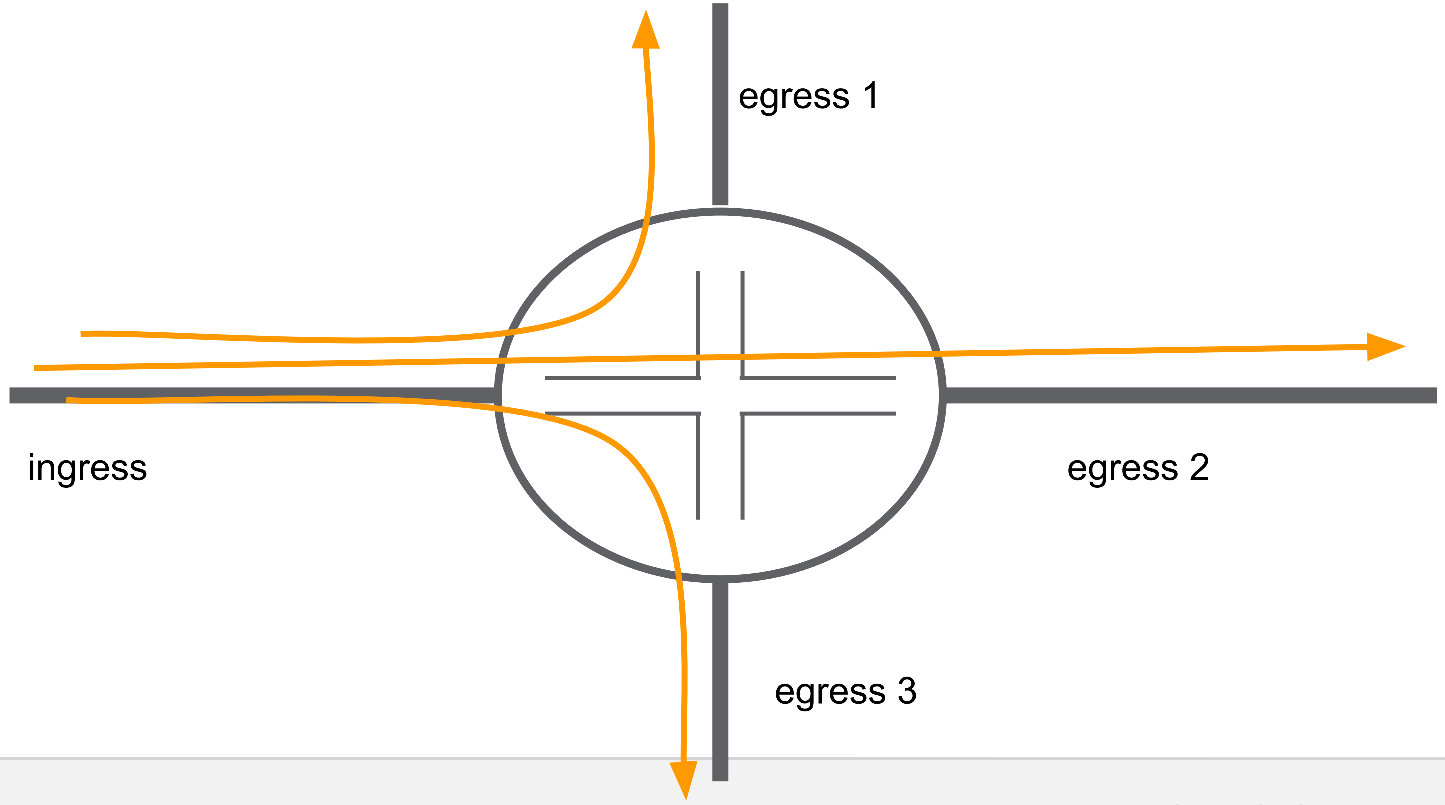
We source route the test packets instead.

With source routing, we can target what gets monitored and ensure full layer3 coverage.

For layer3 paths composed with link aggregation groups, we cover them by creating *n* distinct flows per individual link (today we use *n*=4). The flows need to match the hashing algorithm configured.

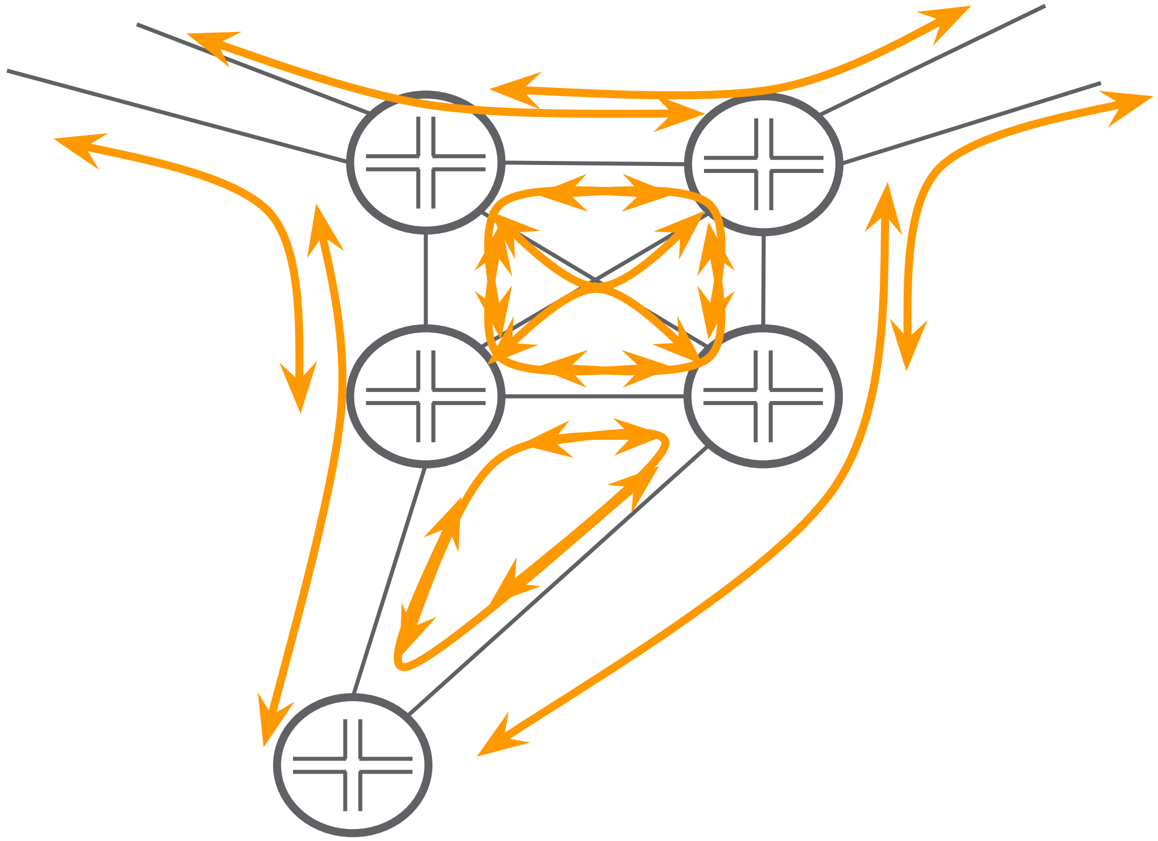
Exhaustive coverage.

More importantly, instead of testing simply interfaces and nodes, we test the ability for a node to forward a packet from every ingress to every egress interfaces.
We test every combination of ingress to egress for each node.



Exhaustive coverage: Testing every forwarding path

Quick illustration of what the coverage of a typical Core, Distribution, Access topology would look like.



How do we localize faulty components ?

Where did my packet go ?

```
$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1
Request timeout for icmp_seq 0
Request timeout for
Request timeout for
Request timeout for icmp_seq 3
^C
--- 1.1.1.1 ping st
5 packets transmitte
100.0% packet loss
```

Dropped on the way there ?

Dropped on the way back ?

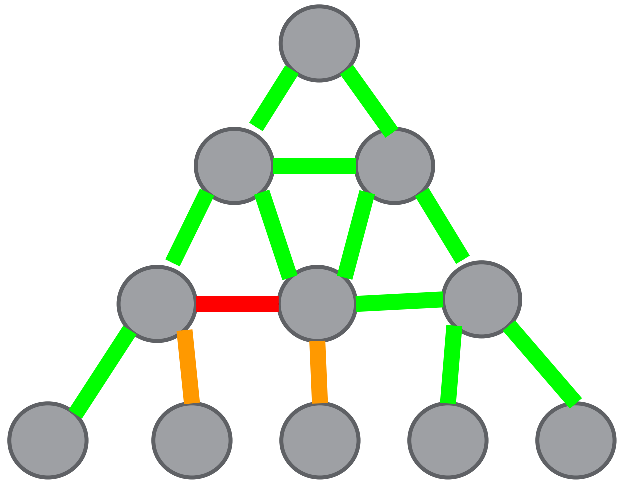
Fiber ? Link ? Node ?
Switching fabric ?

What was the state of the network at the moment of the failure?

Isolate a fault in a complex network

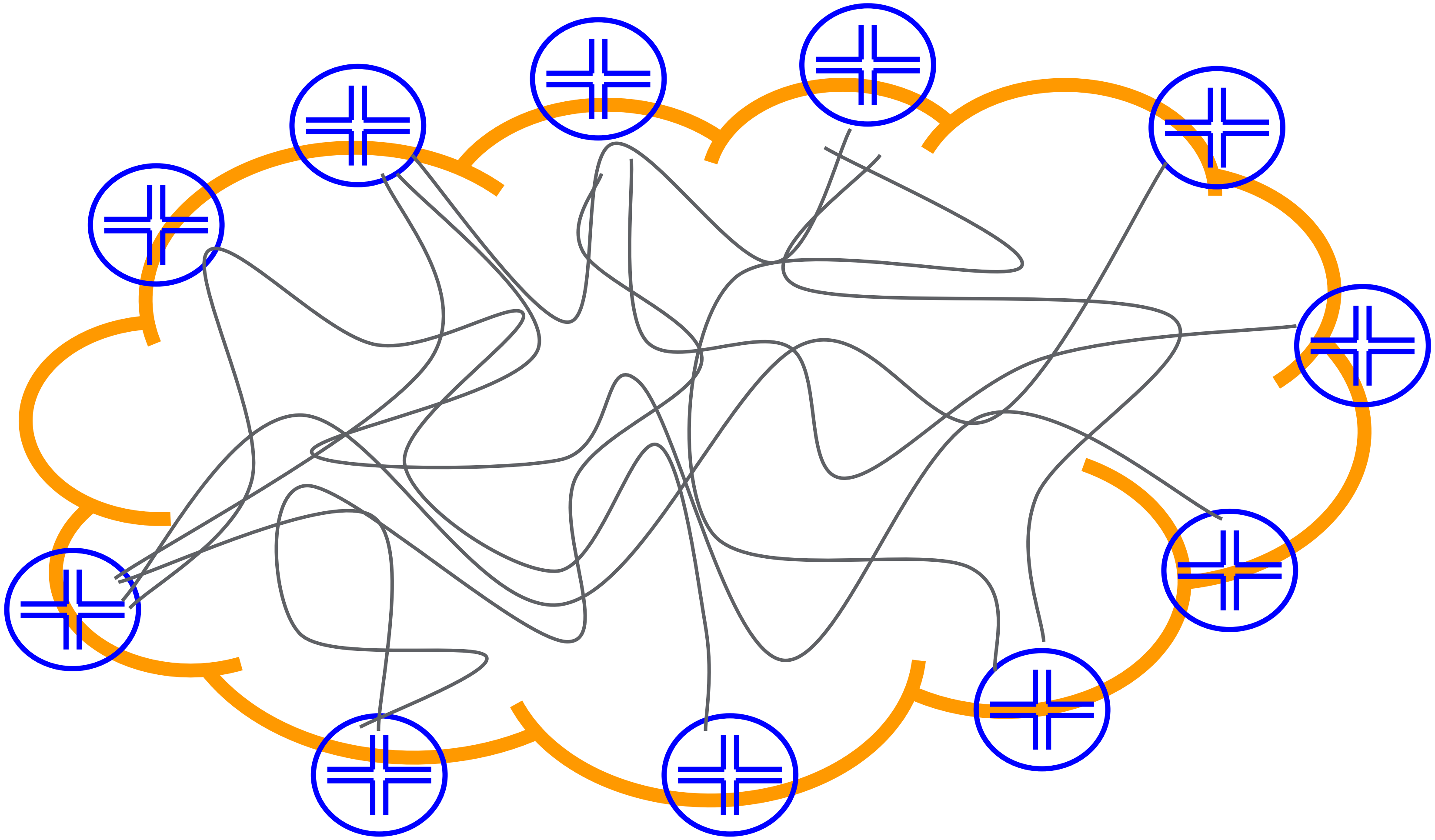
With traditional blackbox monitoring this is similar to finding a needle in a haystack.

But we know **where the test packets go** before they even leave (source routing) and **we test every single path one by one.**



If the red link is the only common component to the failed tests, then it's the root cause.

Our current implementation, entropic paths



What did we learn ?

It works

- It is working pretty well, we found problems that nobody knew about
- Silent drops are not that frequent but it does happen regularly
- The system finds low level packet loss down to 0.01%
- It takes about 30 seconds with our current deployment, from fault to localization
- We can test components (interfaces, links, devices) **not yet in production**. Because we use source based routing (in the form of RSVP-TE LSPs signaled with strict static EROs)
- We found RSVP signaling errors (bugs or shortcuts to improve convergence time)

It creates a lot of state

For a 16 interfaces device it creates at least **240** tests. When using RSVP-TE that results in 240 LSPs. Multiply that by hundreds for a large network and the RSVP state and amount of nexthops can become a problem.

The "mapping" and correlation can become fairly complex to limit the amount of of state, especially on transit nodes.

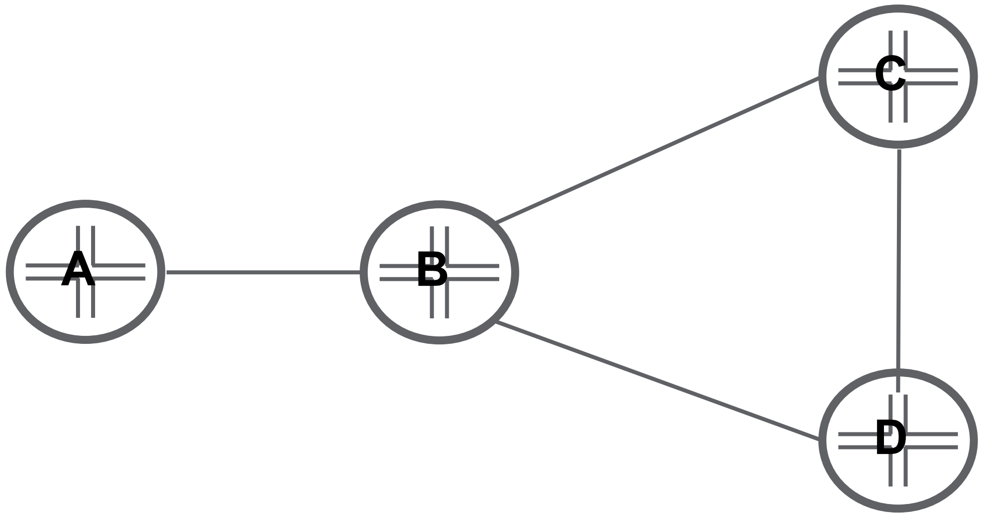
What's next ?

Reducing state

Reduce state

An alternative is **not to use RSVP** but keep the state in the test packets instead.

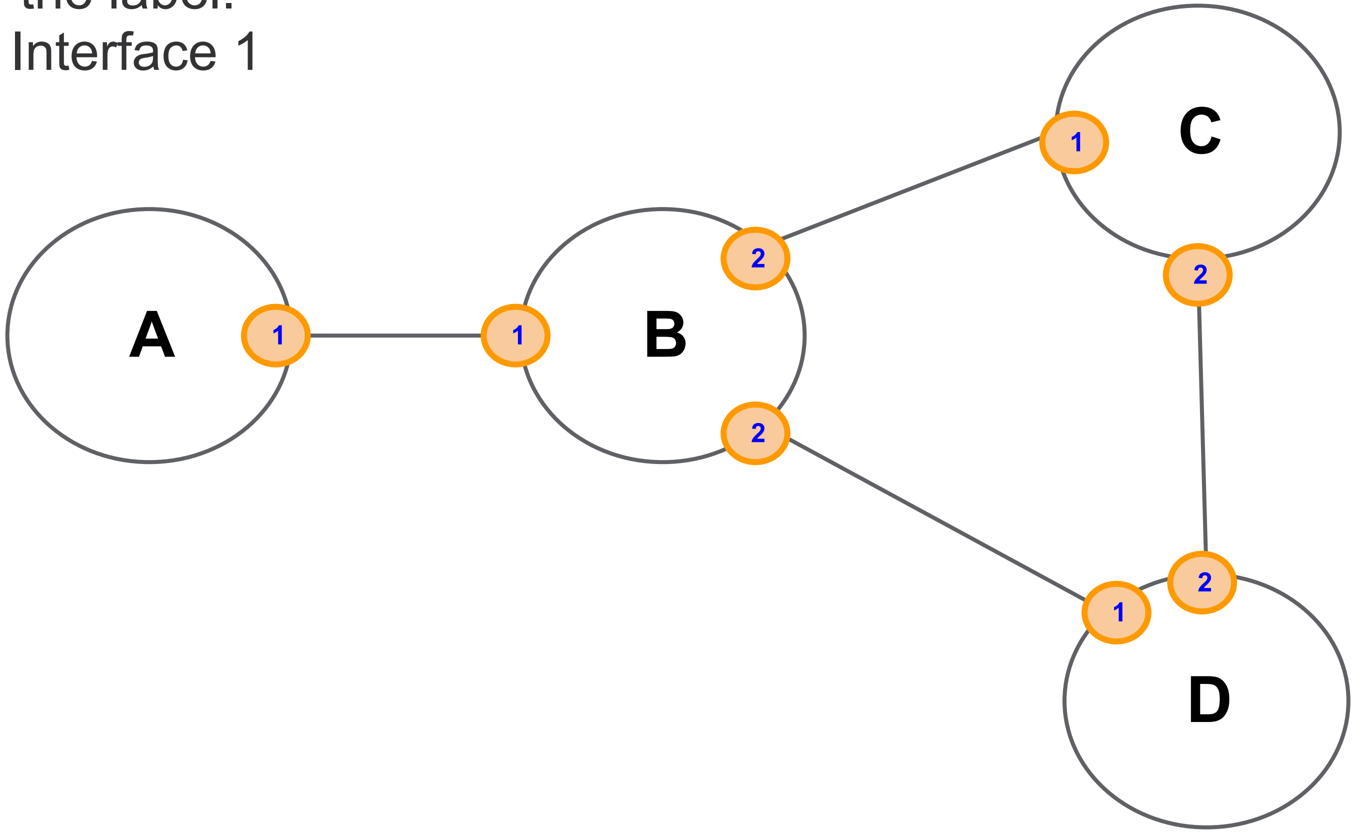
Let say we want to test:
A -> B -> C -> D -> B -> A



Reduce state: One hop static LSP

We can create static LSPs that direct traffic to a specific interface and POP the label.

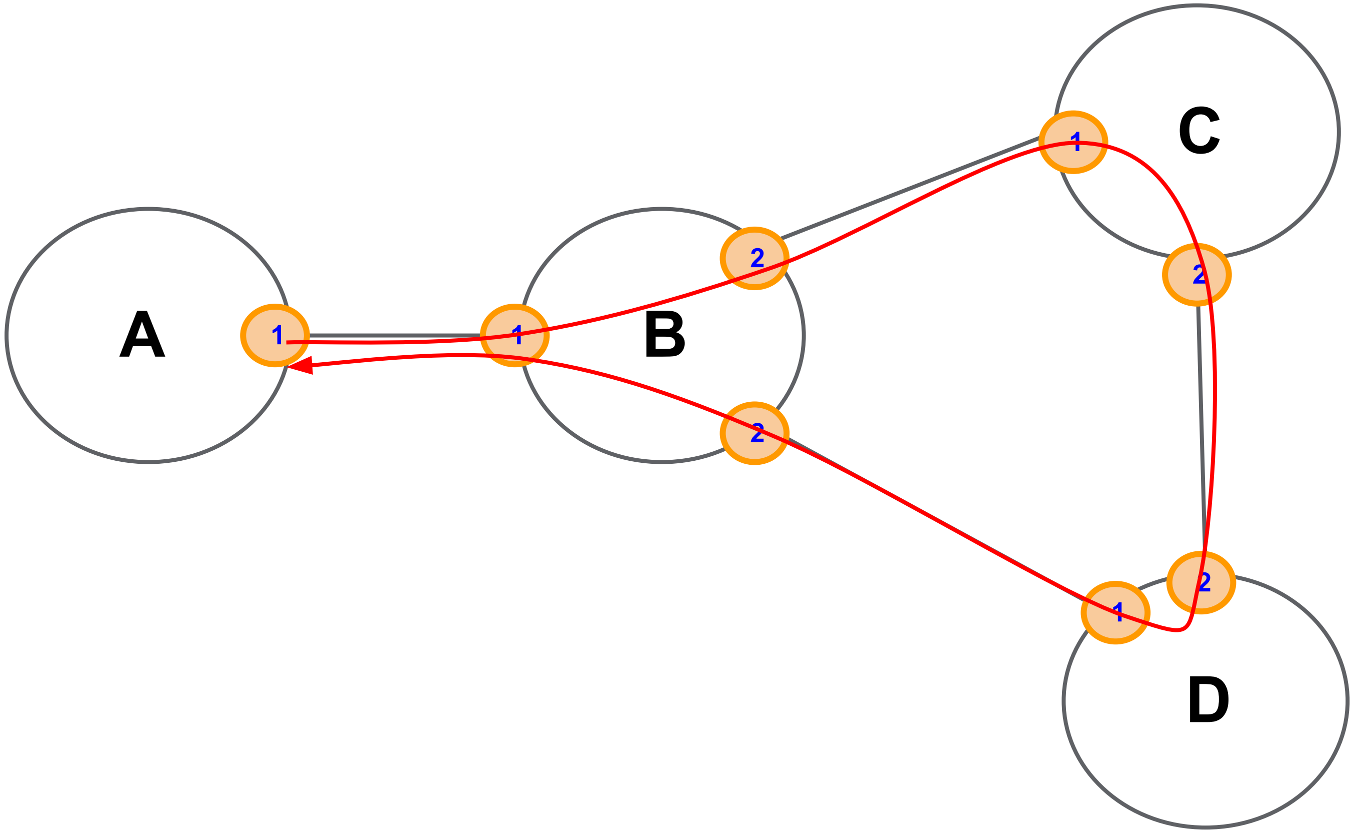
● 1 Interface 1



Google

Reduce state: One hop static LSP

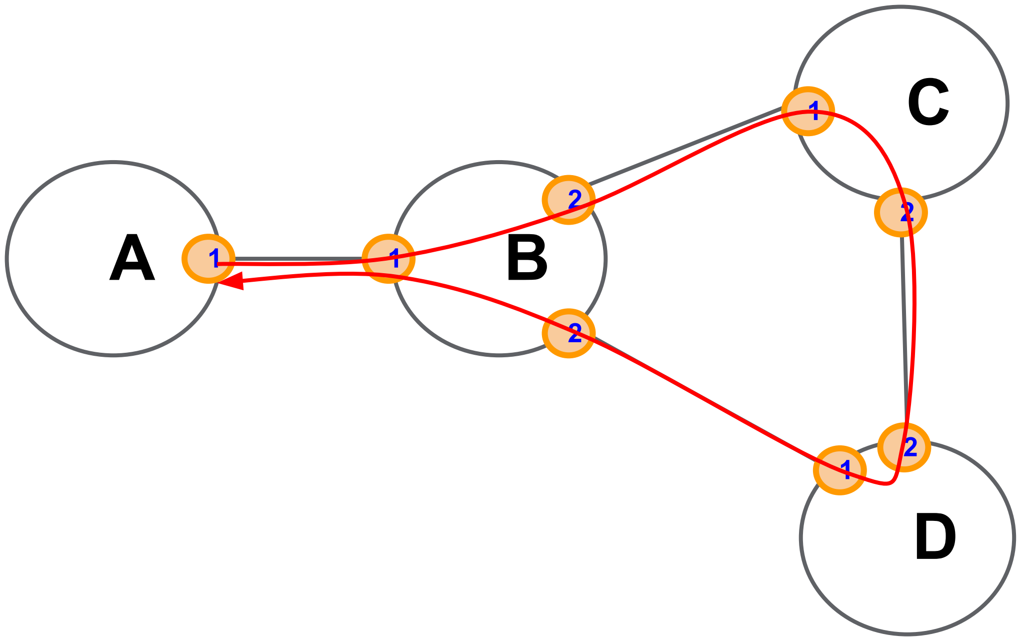
We want to send a packet through the following path:
A:1 ->B:1
B:2 ->C:1
C:2 ->D:2
D:1 ->B:2
B:1 ->A:1



Reduce state: One hop static LSP

We just build a packet with the following stacked labels:
[1, 2, 2, 1, 1(S)]

Router A has a static LSP that says:
For packets with incoming label 1, pop the label and forward to interface 1.



Reduce state: One hop static LSP

1. Router A: [1, 2, 2, 1, 1]
   =>POP label 1 and fwd to Router B

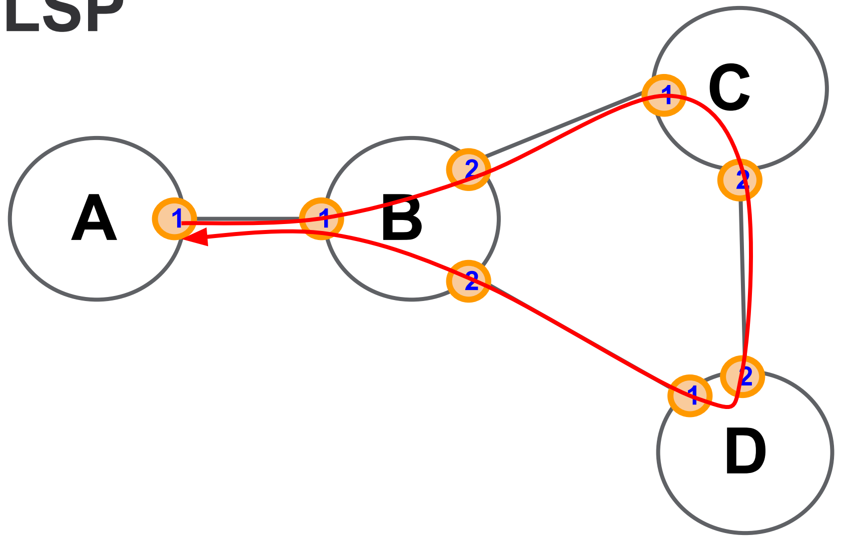
2. Router B: [2, 2, 1, 1]
   =>POP label 2 and fwd to Router C

3. Router C: [2, 1, 1]
   =>POP label 2 and fwd to Router D

4. Router D: [1, 1]
   =>POP label 1 and fwd to Router B

5. Router B: [1]
   =>POP label 1 and fwd to Router A

Router A looks up the IP dest address and sends the packet to its destination.



Questions ?