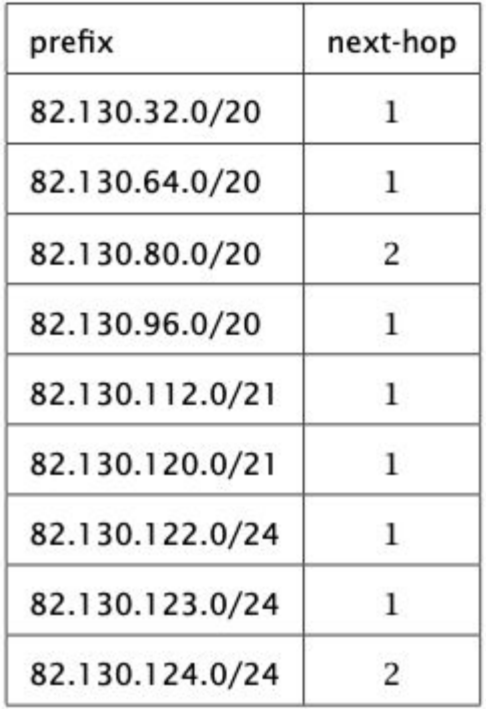
# Pruning IP Forwarding Tables

Consider an IP router with a forwarding table composed of the 9 entries depicted on the right.

Write down an equivalent forwarding table by combining entries together into shorter ones such that the resulting table has the least number of entries. Your reduced forwarding table should be such that the forwarding decision made by the router for any IP packet is equivalent to the initial one.

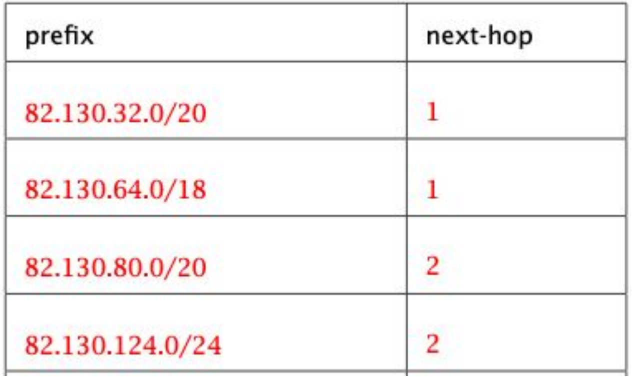


| prefix | next-hop |
|---|---|
| 82.130.32.0/20 | 1 |
| 82.130.64.0/20 | 1 |
| 82.130.80.0/20 | 2 |
| 82.130.96.0/20 | 1 |
| 82.130.112.0/21 | 1 |
| 82.130.120.0/21 | 1 |
| 82.130.122.0/24 | 1 |
| 82.130.123.0/24 | 1 |
| 82.130.124.0/24 | 2 |

# Pruning IP Forwarding Tables

Consider an IP router with a forwarding table composed of the 9 entries depicted on the right.

Write down an equivalent forwarding table by combining entries together into shorter ones such that the resulting table has the least number of entries. Your reduced forwarding table should be such that the forwarding decision made by the router for any IP packet is equivalent to the initial one.

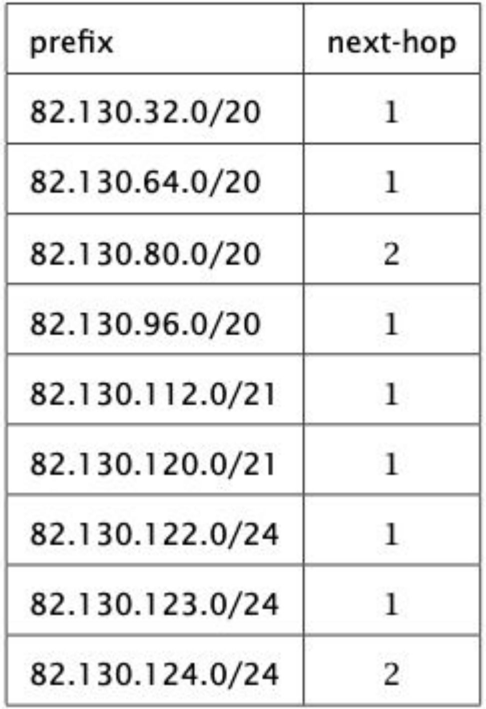


| prefix | next-hop |
|---|---|
| 82.130.32.0/20 | 1 |
| 82.130.64.0/18 | 1 |
| 82.130.80.0/20 | 2 |
| 82.130.124.0/24 | 2 |

# Pruning IP Forwarding Tables

Network admins also can install “default routes” that are catch-alls for traffic that doesn’t fit into a specific rule or for those they want to aggregate. The default route is typically 0.0.0.0/0

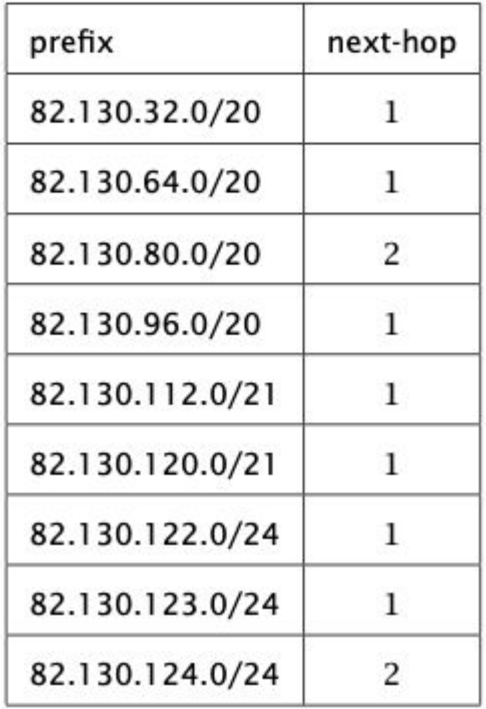
How would the table change with a default route?



| prefix | next-hop |
|---|---|
| 82.130.32.0/20 | 1 |
| 82.130.64.0/20 | 1 |
| 82.130.80.0/20 | 2 |
| 82.130.96.0/20 | 1 |
| 82.130.112.0/21 | 1 |
| 82.130.120.0/21 | 1 |
| 82.130.122.0/24 | 1 |
| 82.130.123.0/24 | 1 |
| 82.130.124.0/24 | 2 |

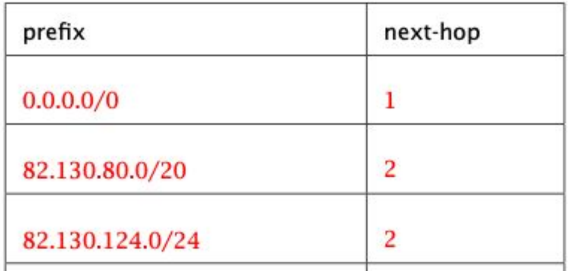
# Pruning IP Forwarding Tables

Network admins also can install “default routes” that are catch-alls for traffic that doesn’t fit into a specific rule or for those they want to aggregate. The default route is typically 0.0.0.0/0



| prefix | next-hop |
|---|---|
| 82.130.32.0/20 | 1 |
| 82.130.64.0/20 | 1 |
| 82.130.80.0/20 | 2 |
| 82.130.96.0/20 | 1 |
| 82.130.112.0/21 | 1 |
| 82.130.120.0/21 | 1 |
| 82.130.122.0/24 | 1 |
| 82.130.123.0/24 | 1 |
| 82.130.124.0/24 | 2 |

How would the table change with a default route?

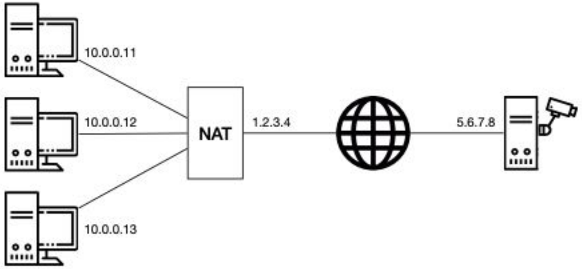


| prefix | next-hop |
|---|---|
| 0.0.0.0/0 | 1 |
| 82.130.80.0/20 | 2 |
| 82.130.124.0/24 | 2 |

# NAT

Consider the network topology shown. Alice has multiple PCs at home (10.0.0.11–13) which share a single public IP address (1.2.3.4) via a NAT device. Further, she operates a surveillance camera server which is directly connected to the Internet with a public IP address (5.6.7.8). The camera transmits the live video signal as a stream of UDP packets with source port 1000 to a configurable destination IP address and port.

Alice wants to receive the live video stream on one of her PCs and thus configures the camera to send the video signal to IP 10.0.0.11 and port 1234. However, she does not receive it on her PC. Why? Where is this traffic sent to?



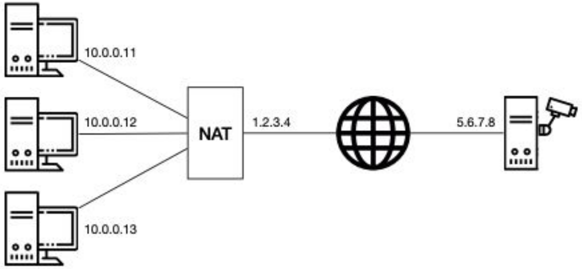
Alice operates three PCs and one camera server

# NAT

Consider the network topology shown. Alice has multiple PCs at home (10.0.0.11–13) which share a single public IP address (1.2.3.4) via a NAT device. Further, she operates a surveillance camera server which is directly connected to the Internet with a public IP address (5.6.7.8). The camera transmits the live video signal as a stream of UDP packets with source port 1000 to a configurable destination IP address and port.

Alice wants to receive the live video stream on one of her PCs and thus configures the camera to send the video signal to IP 10.0.0.11 and port 1234. However, she does not receive it on her PC. Why? Where is this traffic sent to?

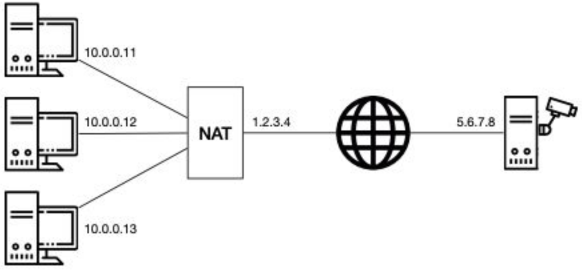
Solution: All IP addresses in the 10.0.0.0/8 prefix are private and not routed in the Internet. As 10.0.0.11 is one of these internal IPs, the camera has no route to this address. Consequently, that traffic is dropped.



Alice operates three PCs and one camera server

# NAT

Now Alice configures the camera to send the video signal to IP 1.2.3.4 and port 1234. But she still does not receive it on any of her PCs. Why? Where is this traffic sent to?

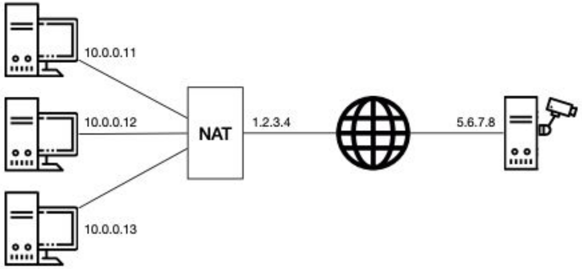


Alice operates three PCs and one camera server

# NAT

Now Alice configures the camera to send the video signal to IP 1.2.3.4 and port 1234. But she still does not receive it on any of her PCs. Why? Where is this traffic sent to?

Solution: The IP address 1.2.3.4 is a globally routed address and therefore, the traffic arrives at the NAT box. However, as there is no corresponding address translation rule in the NAT for that specific destination port, the NAT does not know how to rewrite the packet and where to forward the traffic to. The traffic is dropped at the NAT box.

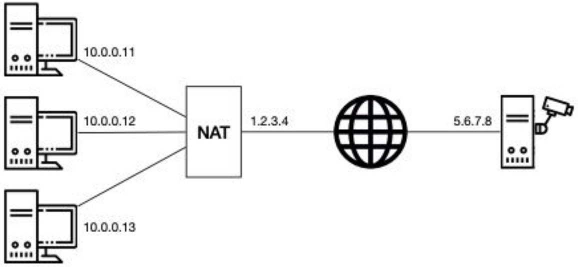


Alice operates three PCs and one camera server

# NAT

What can Alice do such that she receives the video signal at her PC with IP address 10.0.0.11 and at port 1234 assuming that she cannot modify the configuration of the NAT? Describe step-by-step what she can do if she has the following possibilities:

- Send one single UDP packet with arbitrary source and destination addresses and ports from each of her PCs;
- observe the received packets at each of her PCs and the camera server;
- specify the destination IP address and port for the video signal.

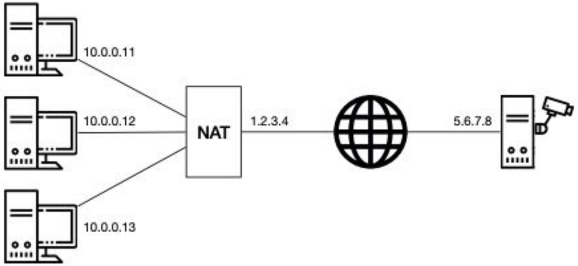


Alice operates three PCs and one camera server

# NAT

What can Alice do such that she receives the video signal at her PC with IP address 10.0.0.11 and at port 1234 assuming that she cannot modify the configuration of the NAT? Describe step-by-step what she can do if she has the following possibilities:

- Send one single UDP packet with arbitrary source and destination addresses and ports from each of her PCs;
- observe the received packets at each of her PCs and the camera server;
- specify the destination IP address and port for the video signal.



Alice operates three PCs and one camera server

Solution: First, you want to “punch a hole” in the NAT box for the video stream to enter your network. To do this, you send a packet from an internal host, for example 10.0.0.11:1234, to the camera 5.6.7.8:1000 (or to a rendezvous server). This leads the NAT box to install an address translation rule.

Now, you have to configure the camera with the correct destination IP address and port. The destination IP address is clear: it is the one of the NAT box (1.2.3.4). The port, however, you do not know yet.

Therefore, you start observing the packets arriving at the camera while sending packets from the internal host to the camera, from 10.0.0.11:1234 to 5.6.7.8:1000. At the camera, you will see to what port the NAT changed the source port of the packet.

Finally, configure the camera to send the video stream to the IP address of the NAT box and set the destination port to the port observed previously.

# Routing

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing

inter-domain
routing

Find paths between networks

intra-domain
routing

Find paths within a network

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing

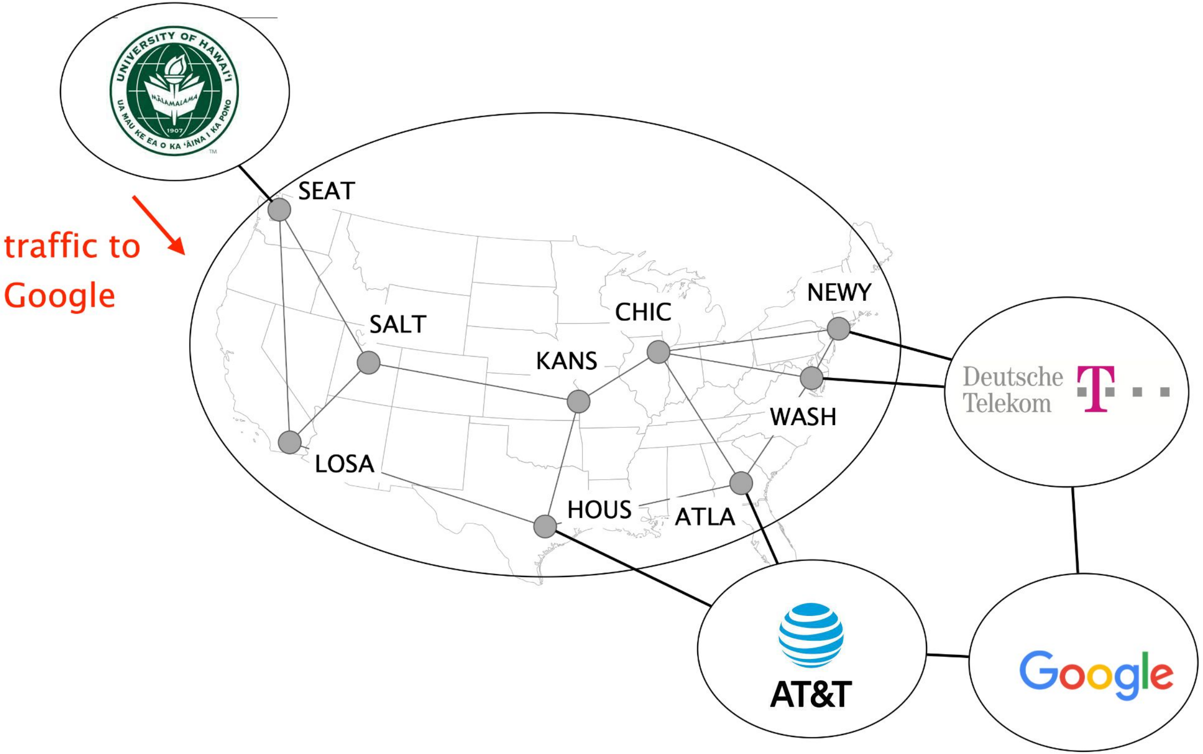
inter-domain routing

intra-domain routing

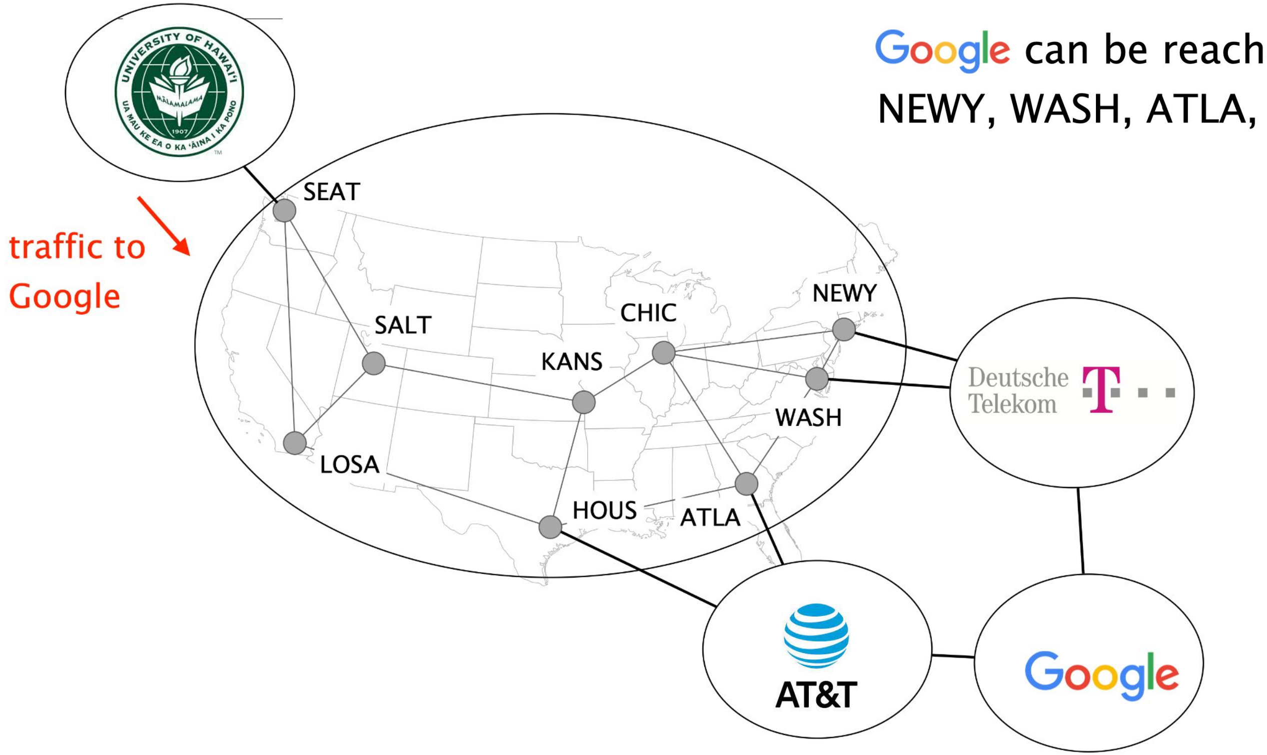
Find paths between networks

Find paths within a network

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing



# Routing Comes in Two Flavors: *intra* and *inter*-domain routing

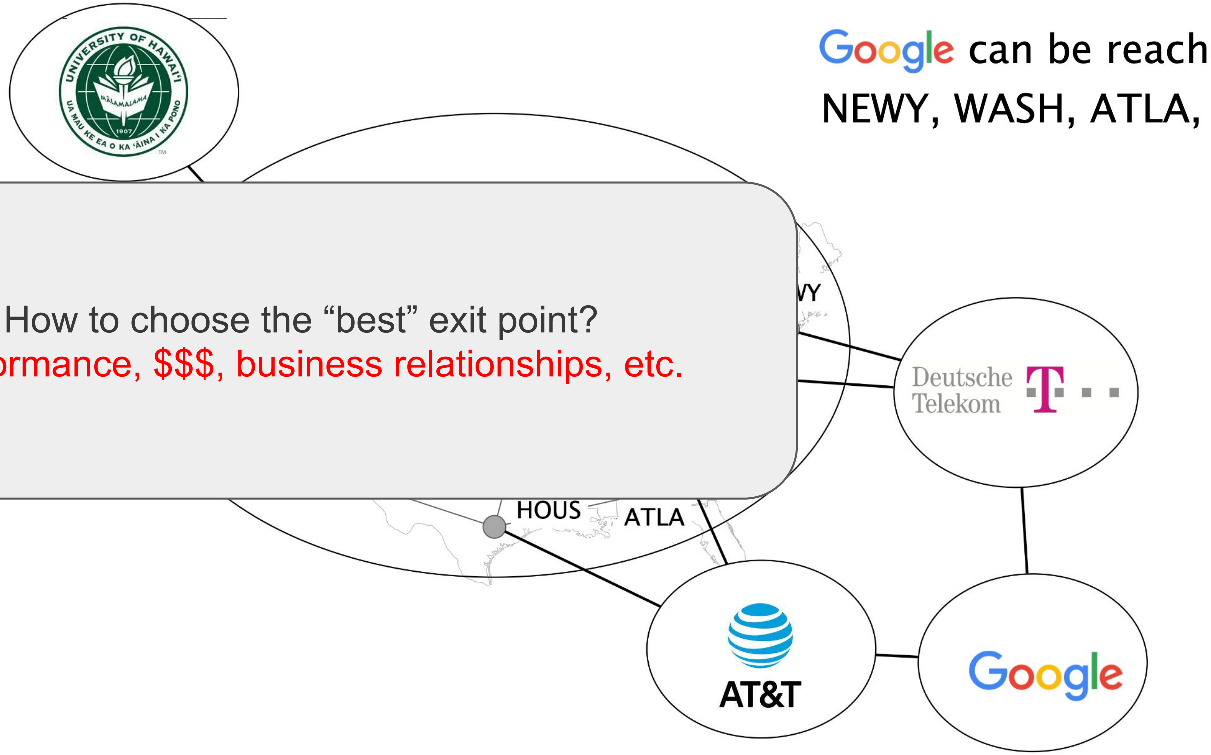


traffic to Google

Google can be reached via NEWY, WASH, ATLA, HOUS

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing

Google can be reached via
NEWY, WASH, ATLA, HOUS



How to choose the “best” exit point?
Performance, $$$, business relationships, etc.

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing

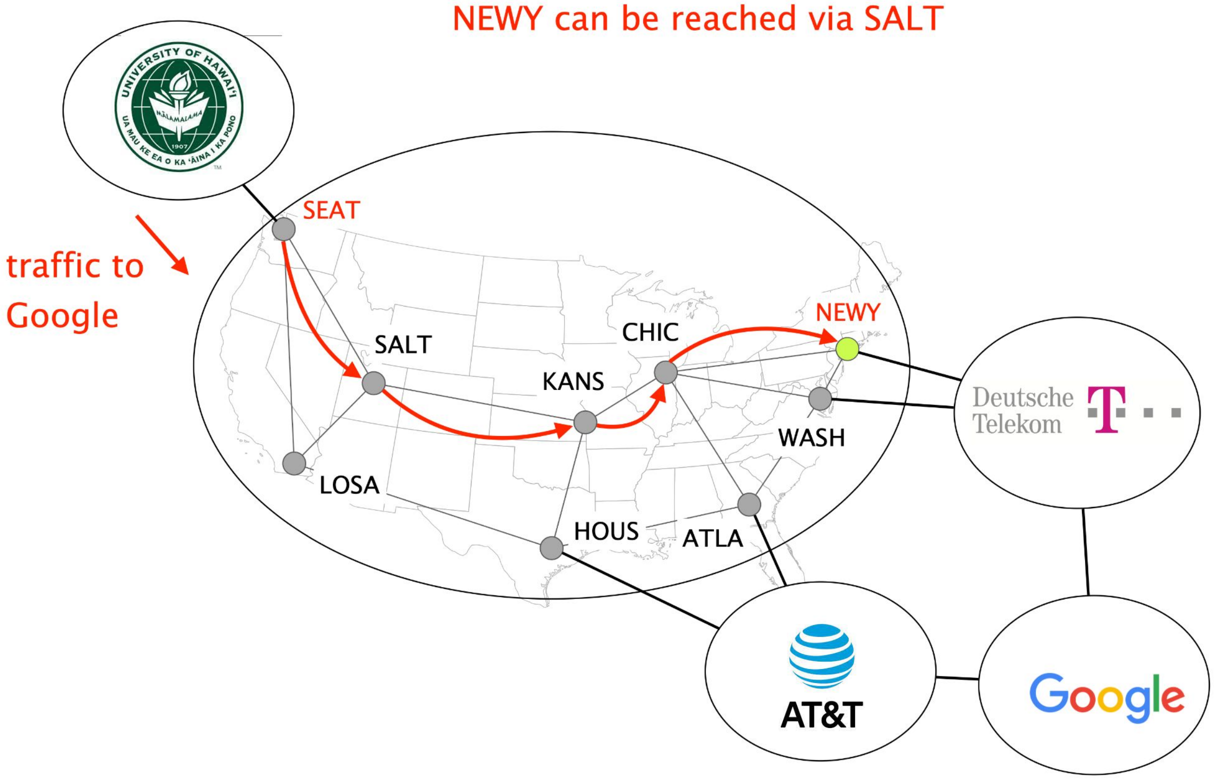
inter-domain routing

intra-domain routing

Find paths between networks

Find paths within a network

# Routing Comes in Two Flavors: *intra* and *inter*-domain routing



```
traceroute to orange.fr (193.252.133.20), 64 hops max, 52 byte packets
```

Intra-domain routing

```
168.105.224.2 (168.105.224.2)  4.483 ms  3.005 ms  4.315 ms
vl-3223-manoa7050-2.uhnet.net (128.171.186.190)  3.182 ms
vl-3222-manoa7050-1.uhnet.net (128.171.186.188)  3.523 ms  2.602 ms
```

Intra-domain routing

```
xe-0-0-0-667-coconut-re0.uhnet.net (128.171.64.182)  8.254 ms
xe-1-0-0-669-coconut-re0.uhnet.net (128.171.213.13)  5.655 ms  5.989 ms
xe-0-0-6-73-ohelo-re0.uhnet.net (205.166.205.46)  5.414 ms  4.974 ms  6.153 ms
```

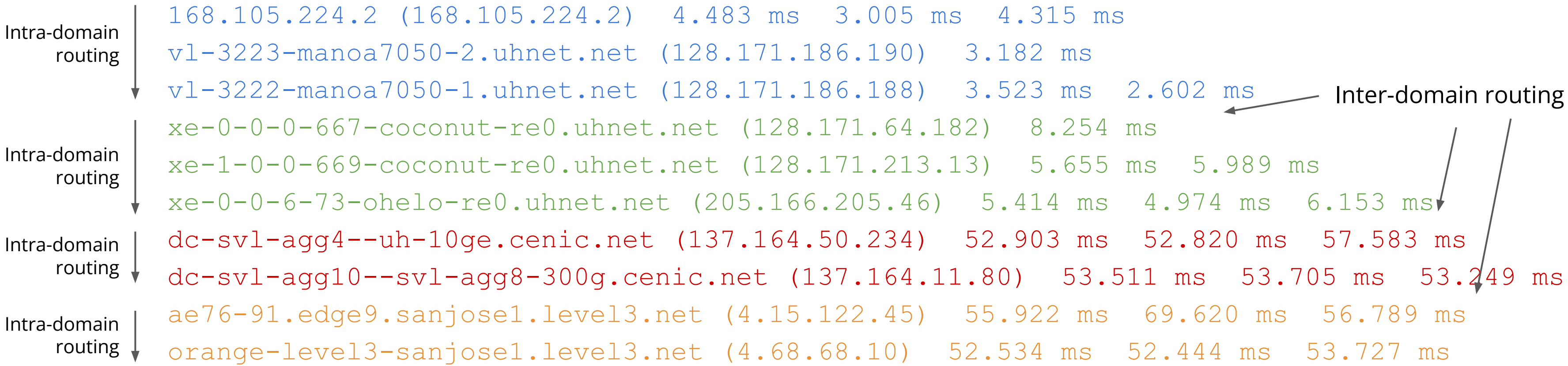
Intra-domain routing

```
dc-svl-agg4--uh-10ge.cenic.net (137.164.50.234)  52.903 ms  52.820 ms  57.583 ms
dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80)  53.511 ms  53.705 ms  53.249 ms
```

Intra-domain routing

```
ae76-91.edge9.sanjose1.level3.net (4.15.122.45)  55.922 ms  69.620 ms  56.789 ms
orange-level3-sanjose1.level3.net (4.68.68.10)  52.534 ms  52.444 ms  53.727 ms
```

```
traceroute to orange.fr (193.252.133.20), 64 hops max, 52 byte packets
```



```
168.105.224.2 (168.105.224.2)  4.483 ms  3.005 ms  4.315 ms
vl-3223-manoa7050-2.uhnet.net (128.171.186.190)  3.182 ms
vl-3222-manoa7050-1.uhnet.net (128.171.186.188)  3.523 ms  2.602 ms
xe-0-0-0-667-coconut-re0.uhnet.net (128.171.64.182)  8.254 ms
xe-1-0-0-669-coconut-re0.uhnet.net (128.171.213.13)  5.655 ms  5.989 ms
xe-0-0-6-73-ohelo-re0.uhnet.net (205.166.205.46)  5.414 ms  4.974 ms  6.153 ms
dc-svl-agg4--uh-10ge.cenic.net (137.164.50.234)  52.903 ms  52.820 ms  57.583 ms
dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80)  53.511 ms  53.705 ms  53.249 ms
ae76-91.edge9.sanjose1.level3.net (4.15.122.45)  55.922 ms  69.620 ms  56.789 ms
orange-level3-sanjose1.level3.net (4.68.68.10)  52.534 ms  52.444 ms  53.727 ms
```

# Internet Routing

1. Intra-domain routing
    - Link-state protocols
    - Distance-vector protocols

2. Inter-domain routing
    - Path-vector protocols

# Internet Routing

1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols

Intra-domain routing enables routers to compute forwarding paths to any internal subnet

# Internet Routing

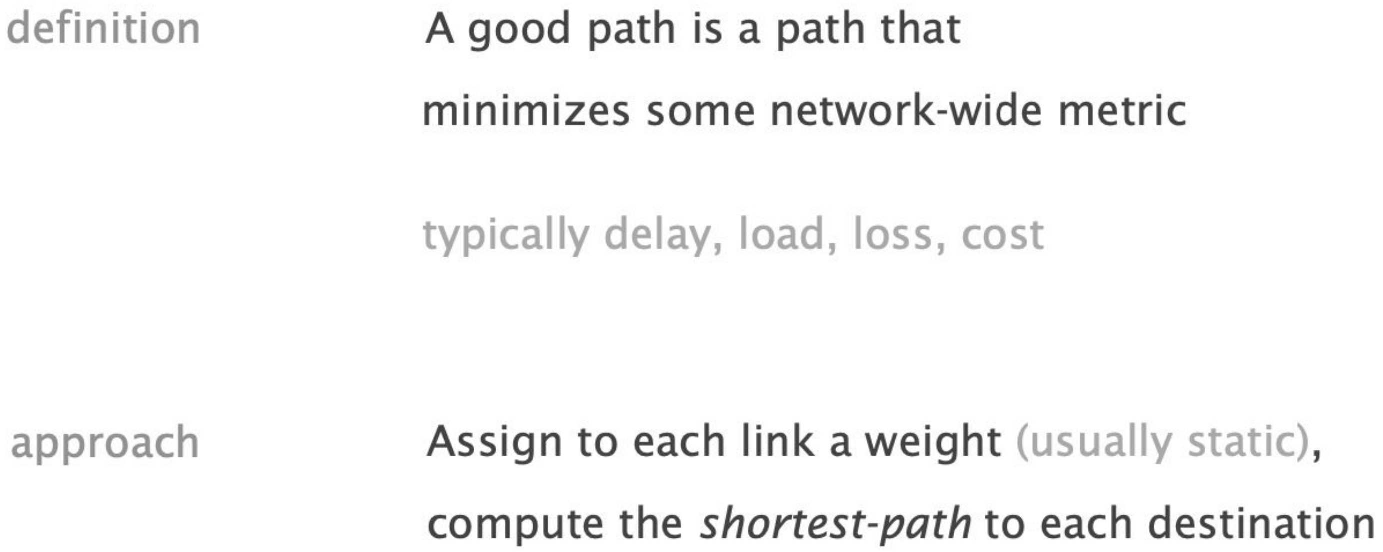
1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols

Intra-domain routing enables routers to compute forwarding paths to any internal subnet

What kind of paths?

# Network Operators don’t use Arbitrary Paths, they use Good Paths



definition

A good path is a path that

minimizes some network-wide metric

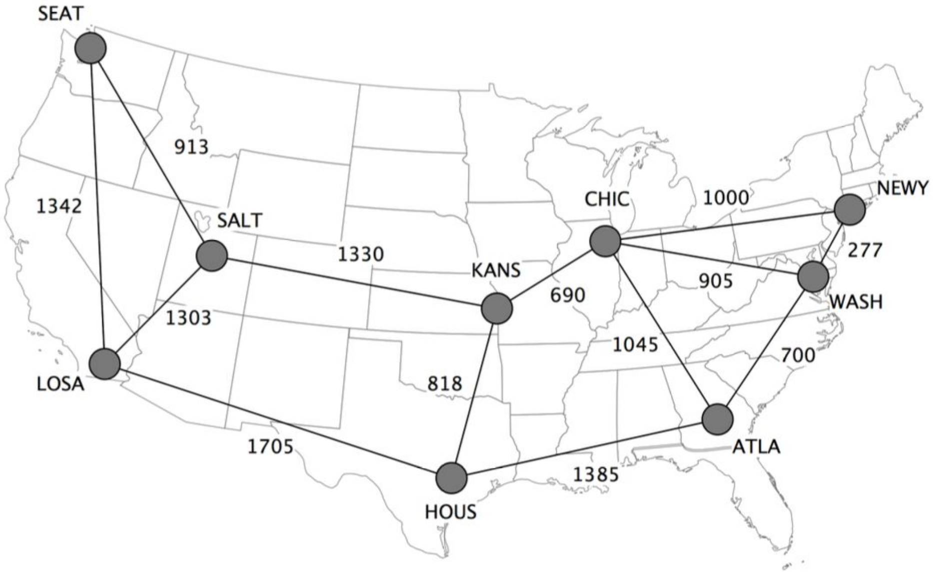
typically delay, load, loss, cost

approach

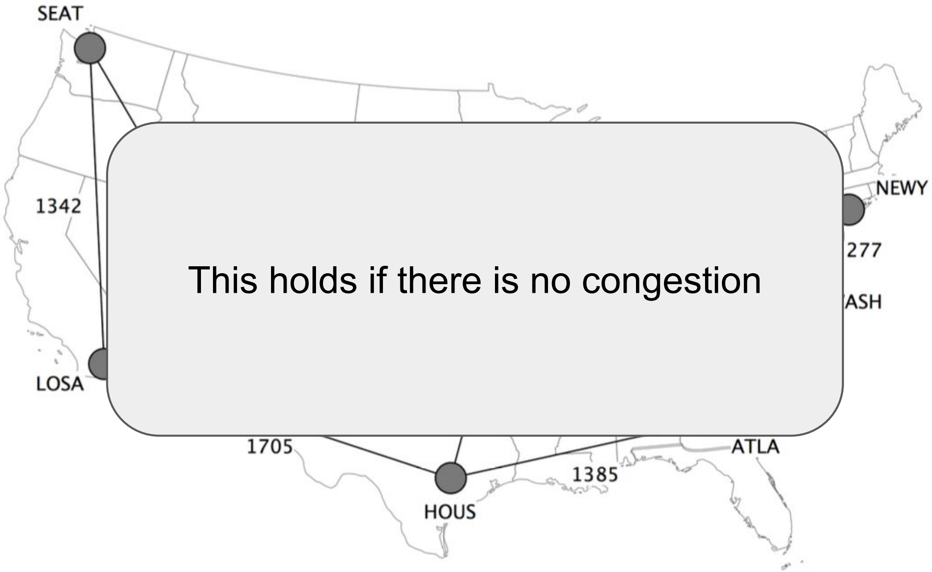
Assign to each link a weight (usually static),

compute the *shortest-path* to each destination

# When Link Weights are Proportional to Distance, Shortest Paths Minimize end-to-end Delay

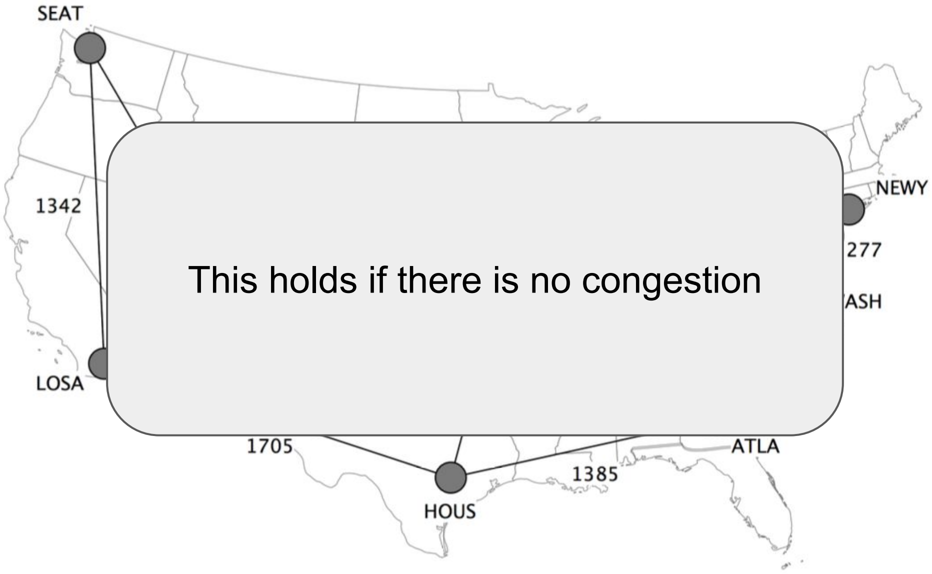


# When Link Weights are Proportional to Distance, Shortest Paths Minimize end-to-end Delay



This holds if there is no congestion

# When Link Weights are Inversely Proportional to Link Capacity, Throughput is Maximized



This holds if there is no congestion

# Link-State Routing - Recall...

Each router keeps track of its incident links and cost as well as whether it is up or down

Each router broadcast its own links state to give every router a complete view of the graph

Routers run Dijkstra on the corresponding graph to compute their shortest-paths and forwarding tables

# Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

# Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

All nodes are ensured to receive the latest version of all link-states

# Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

All nodes are ensured to receive the latest version of all link-states

- challenges
  - packet loss
  - out of order arrival

# Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

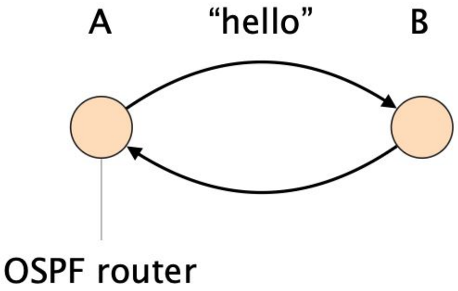
All nodes are ensured to receive the latest version of all link-states

- challenges
  - packet loss
  - out of order arrival
- solutions
  - ACK & retransmissions
  - sequence number
  - time-to-live for each link-state

# When to Initiate Flooding?

| | |
|---|---|
| Topology change | link or node failure/recovery |
| Configuration change | link cost change |
| Periodically | refresh the link-state information<br>every (say) 30 minutes<br>account for possible data corruption |

# How do actual Link-State Protocols Detect Topology Changes?
# Software-based Beaconing

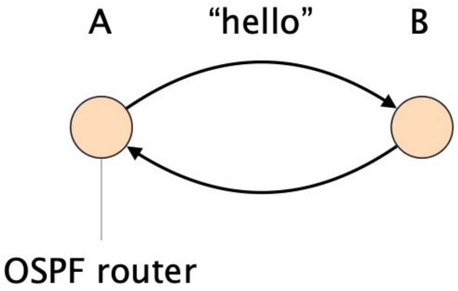


Routers periodically exchange "Hello" in both directions (*e.g.* every 30s)

Trigger a failure after few missed "Hellos" (*e.g.*, after 3 missed ones)

What kind of tradeoffs are present here?

# How do actual Link-State Protocols Detect Topology Changes?
# Software-based Beaconing



Routers periodically exchange “Hello”

in both directions (*e.g.* every 30s)

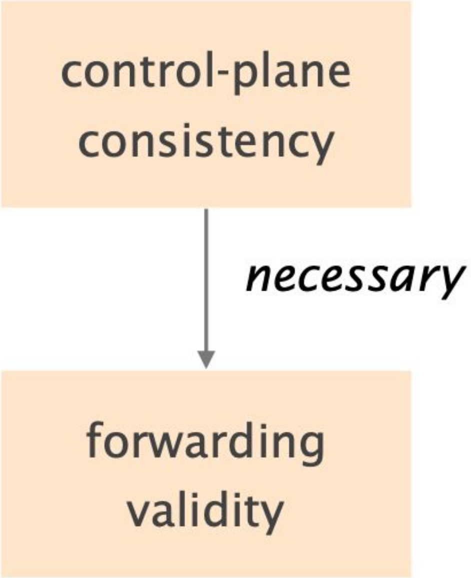
Trigger a failure after few missed “Hellos”

(*e.g.*, after 3 missed ones)

Tradeoffs between:

- detection speed
- bandwidth and CPU overhead
- false positive/negatives

# During Network Changes the Link-State DBs of Each Router May Differ



control-plane consistency

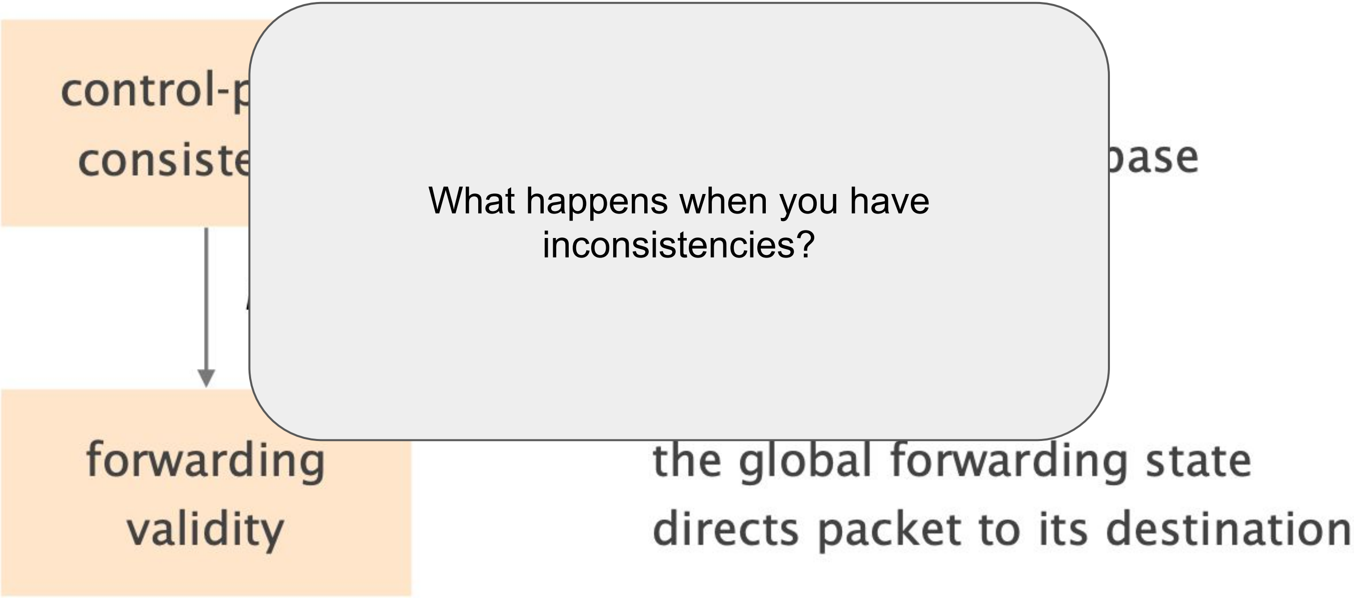
*necessary*

forwarding validity

all nodes have the same link-state database

the global forwarding state directs packet to its destination

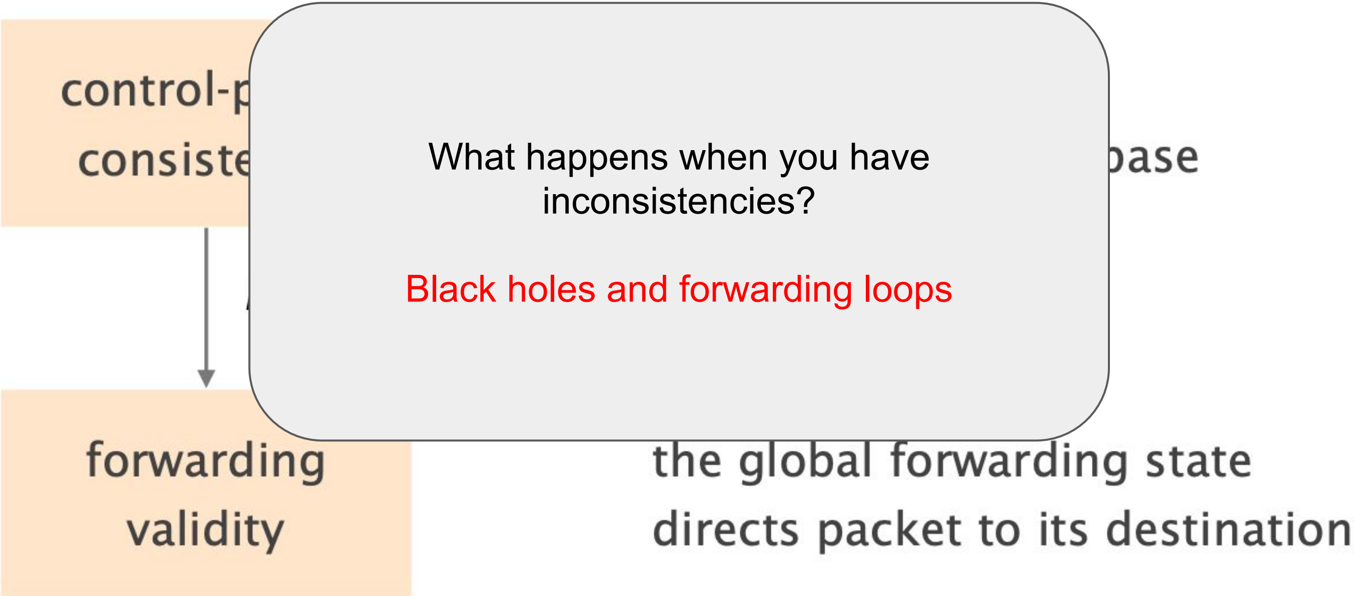
# During Network Changes the Link-State DBs of Each Router May Differ



What happens when you have inconsistencies?

the global forwarding state directs packet to its destination

# During Network Changes the Link-State DBs of Each Router May Differ

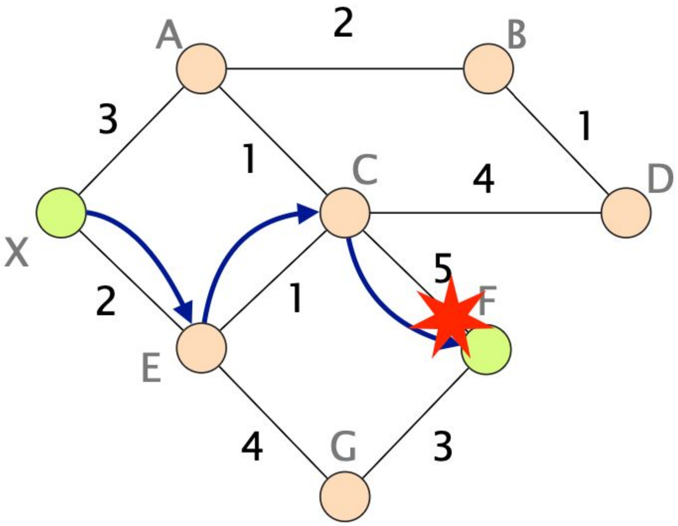


the global forwarding state directs packet to its destination

What happens when you have inconsistencies?

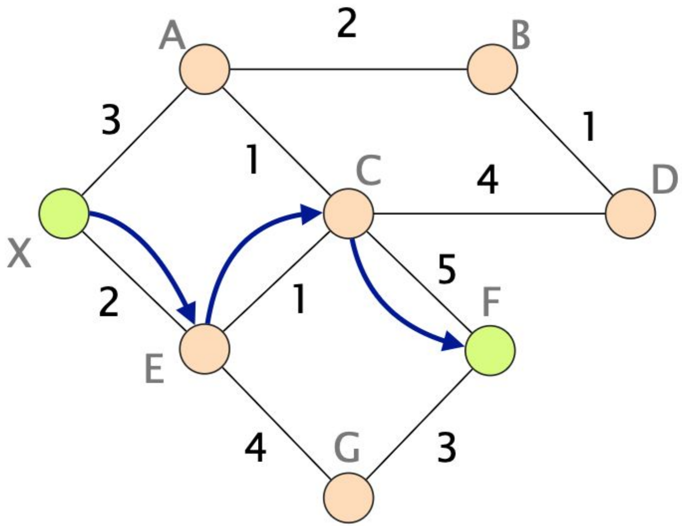
Black holes and forwarding loops

# Black Holes - Due to Detection Delay as Routers Do Not Immediately Detect Failure



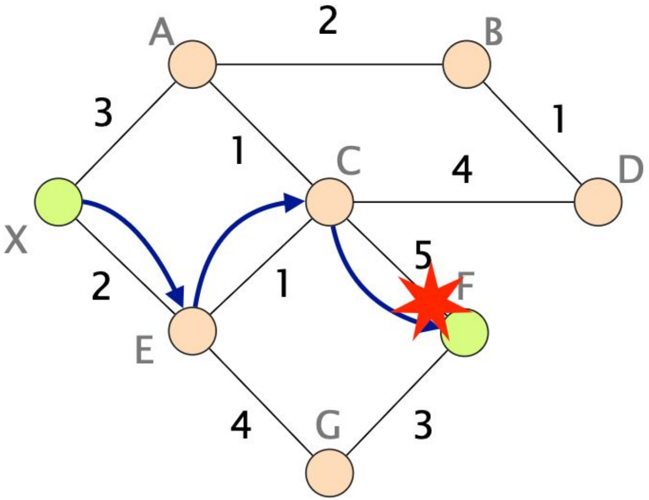
depends on the timeout for detecting lost hellos

# Forwarding Loops - Due to Inconsistent Link-State DBs



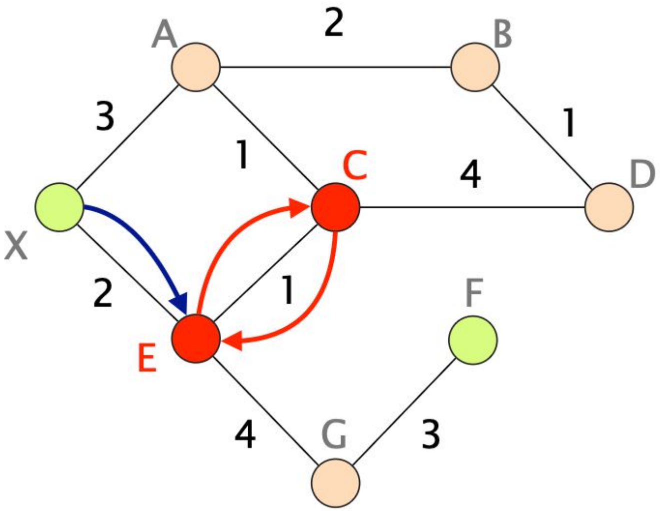
Initial forwarding state

# Forwarding Loops - Due to Inconsistent Link-State DBs



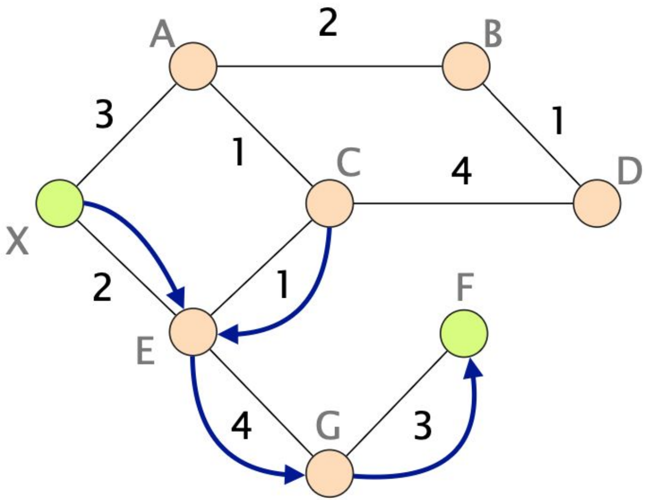
C learns about the failure
and immediately reroute to E

# Forwarding Loops - Due to Inconsistent Link-State DBs



A loop appears as E
isn't yet aware of the failure

# Forwarding Loops - Due to Inconsistent Link-State DBs



The loop disappears as soon as
E updates its forwarding table