Distance Vector Algorithm

Recall Link-State algorithm:

- Each node knows the complete topology graph with link costs
- Each node calculate the shortest path to all other nodes

For Distance-Vector algorithm:

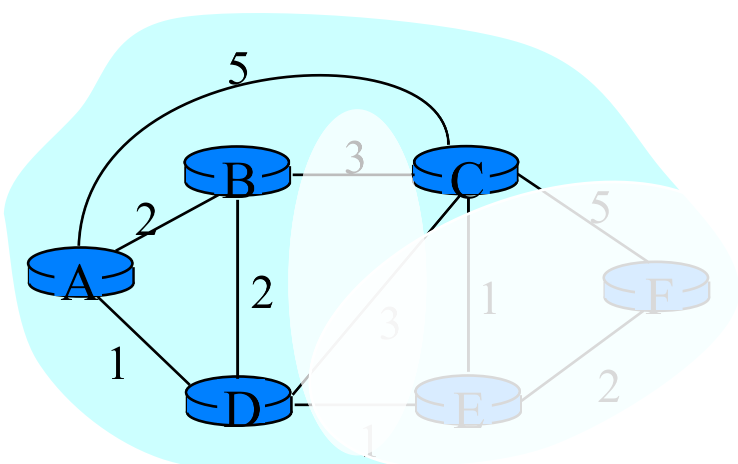
- each node know *only* needs from each direct neighbor its list of distances to all destinations
- Each node computes shortest path based on the input from all its neighbors

Distance Vector Equation

Define: $D_x(y)$:= cost of best path from x to y

Then $\boxed{D_x(y) = \min \{c(x,v) + D_v(y)\}}$

- where min is taken over *all* neighbors v of x



$$D_A(F) = \min \{c(A,B) + D_B(F),$$
$$c(A,D) + D_D(F),$$
$$c(A,C) + D_C(F)\}$$
$$= \min \{2 + 5,$$
$$1 + 3,$$
$$5 + 3\} = 4$$

Node leading to shortest path is next hop ➜ forwarding table

Distance Vector: what a node does

- Node *x* knows link cost to neighbor v: $c(x,v)$
- Node *x* maintains $\mathbf{D}_x = [D_x(y): y \in N]$
  - $D_x(y)$ = estimate of least cost from x to y
- Node *x* sends the distance vector, $\mathbf{D}_x$, to all its neighbors
- Node x receives $\mathbf{D}_v$ from each neighbor v, then calculate $D'_x(y) = \min \{c(x,v) + D_v(y)\}$

  If $D'_x(y) < D_x(y)$:
  - $D_x(y) = D'_x(y)$
  - next hop to y = v
  - Send out the updated $\mathbf{D}_x$

Distance Vector Protocol

Iterative, asynchronous:

- each local iteration caused by:
  - local link cost change
  - DV update message from neighbor
- continues until no nodes exchange info.

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

asynchronous: nodes need *not* exchange info/iterate in lock step

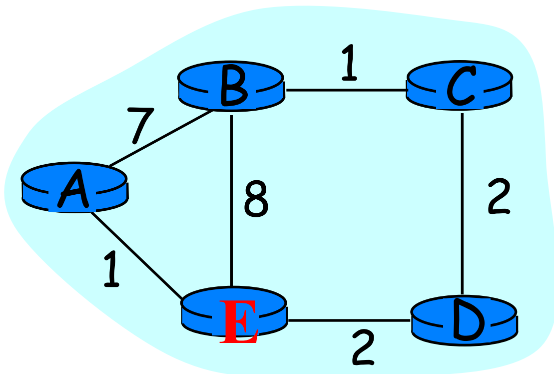
Each node:

wait for (change in local link cost of msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance Table: example



cost to destination via

$D^E()$	A	B	D
A	1	14	5

destination

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\}$$
$$= 8+6 = 14$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\}$$
$$= 2+3 = 5$$

$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\}$$
$$= 2+2 = 4$$

Row: for each possible destination
Column: for each directly-attached neighbor node

Routing table produces forwarding table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

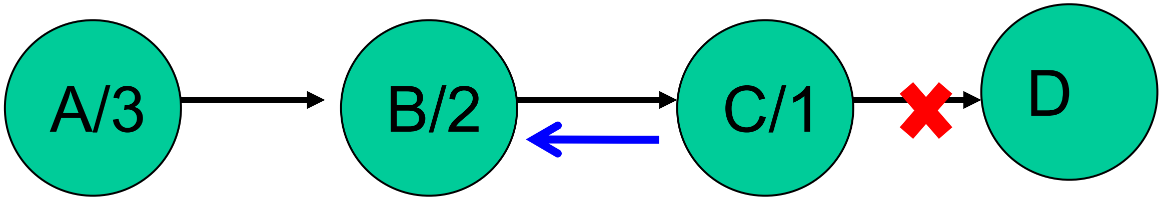
destination

destination	Next hop
A	A
B	D
C	D
D	D

E's routing table → E's forwarding table

Count-To-Infinity Problem

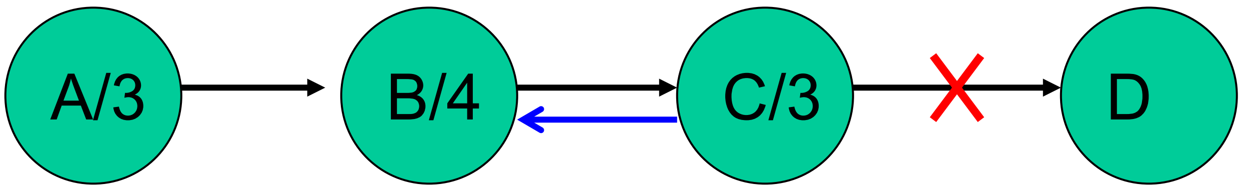
- Assume we use hop count as metric
  - A uses B to reach D with cost 3
  - B uses C to reach D with cost 2
  - C reaches D with cost 1



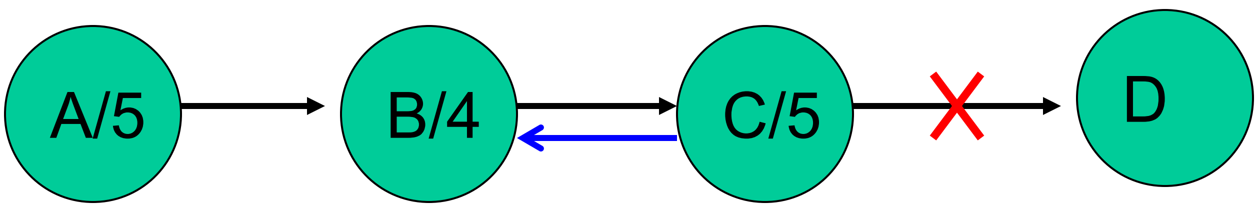
- Suppose link between C and D breaks
  - C switches to B, increase its cost to B’s + 1 = 3

Count-To-Infinity Problem (cont.)

- B's path cost is now 4
  - A has not realized what has happened yet



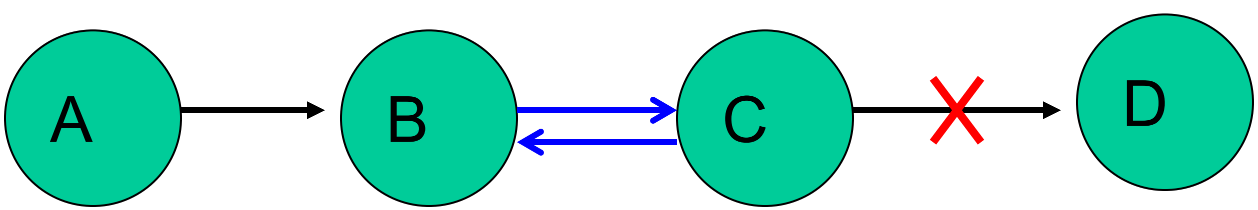
- Then, A's and C's cost are now 5



- B's path cost is changed to 6
  - Cycle repeats while "counting to infinity"

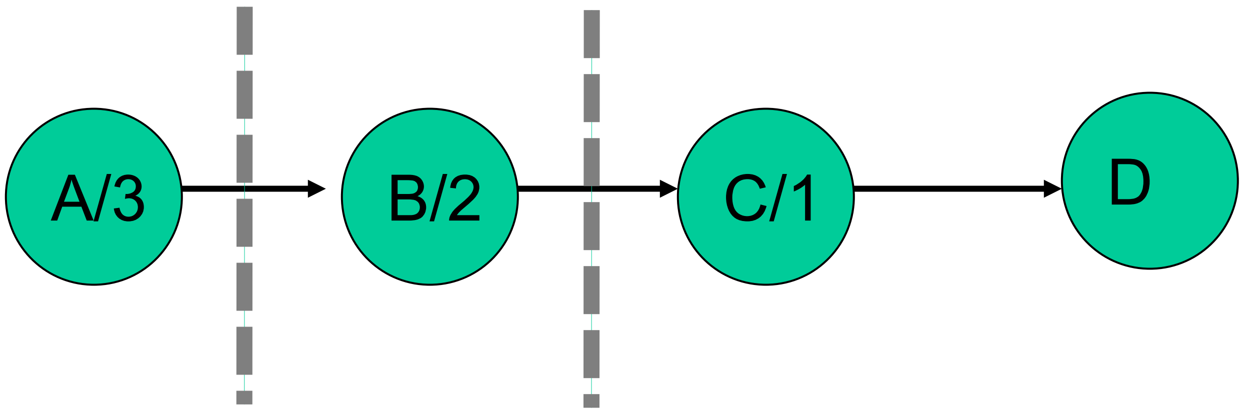
Routing Loops

- In this cases, the packets with destination of D at router A:
  - Go to router B
  - Then go to router C
  - Then go back to router B



Split Horizon

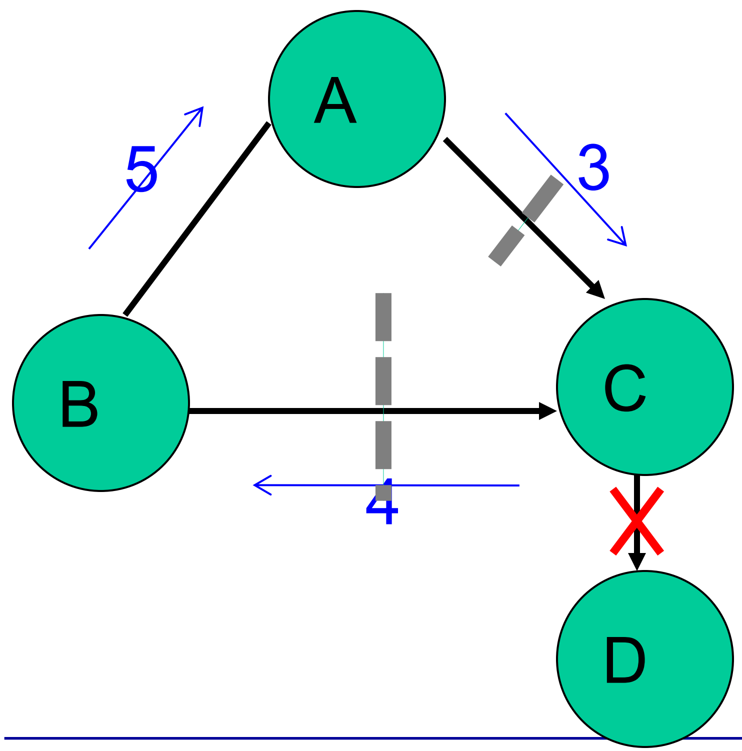
- In Split Horizon: B does not tell C that B can reach D
  - So C does not know that B can reach D



- Once C-D link breaks: C would not switch to go through B to reach D

Split Horizon --- Might Not Work

- Split Horizon doen't eliminate loops in all cases
- Suppose the link between C and D breaks

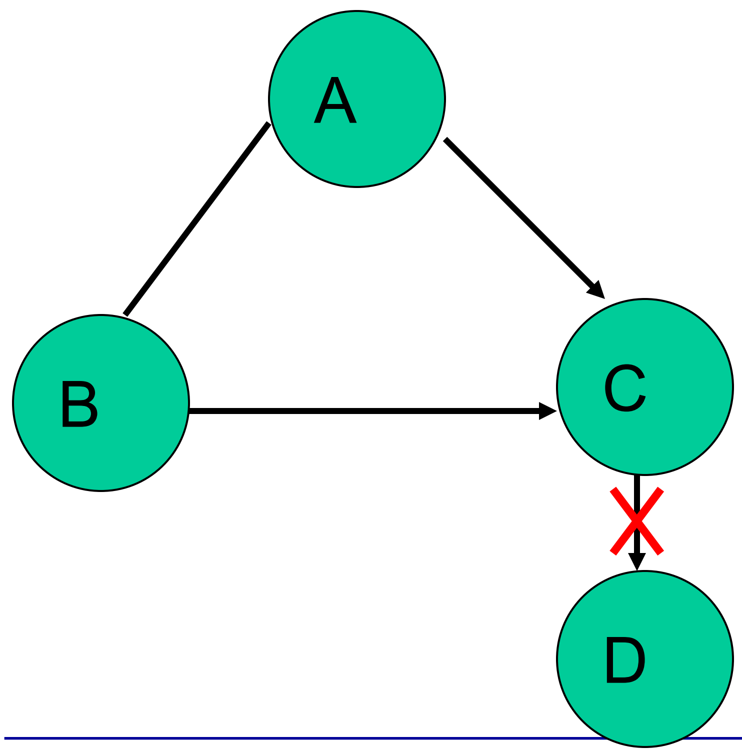


1. A and B do not tell C their distance to D
2. After C–D failure, A learns that B can reach D, so sends new route to C
3. C sends route learned from A to B
4. B sends route learned from C to A
5. A sends route learned from B to C

Routing loop still exists

Split Horizon with poison reverse

- If B goes through C to reach D :
  - B tells C that its (B's) distance to D is *infinite* (so C never attempts to reach D via B)



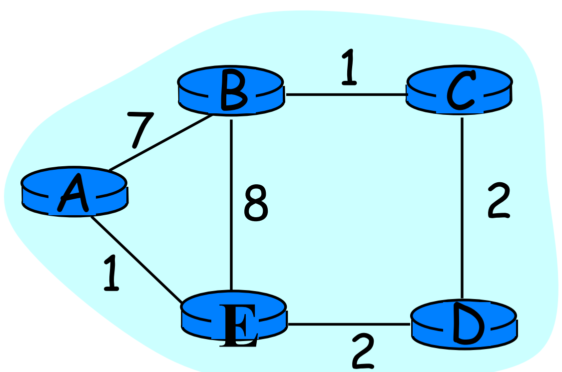
1. A and B tell C that their distance to D is infinite
2. When link C-D fails, C realized that it lost reachability to D
3. C sends to A and B: D_D = infinite

Comparison of LS and DV algorithms

- Performance measure: Message overhead, time to convergence
- distance vector:
  - distribute to neighbors the distances to all destinations
    - Each update msg can be large in size, but travels over one link
  - each node only knows *distances* to other destinations
- link state
  - Broadcast to entire net one’s distance to all neighbors
    - Each update msg is small in size, but travels over all links in the network
  - each node knows entire topology

what happens if a router malfunctions?

- Link-state
  - A node can advertise incorrect *link* cost
  - each node computes its *own* table
- Distance vector
  - A node can advertise incorrect *path* cost
  - one node's distance-list is used by its neighbors for their own routing selection



Node-D: "I have 0 cost to all other nodes"

Link-State:

- updates from A & B: not connected to D
- Updates from C & E: cost not 0

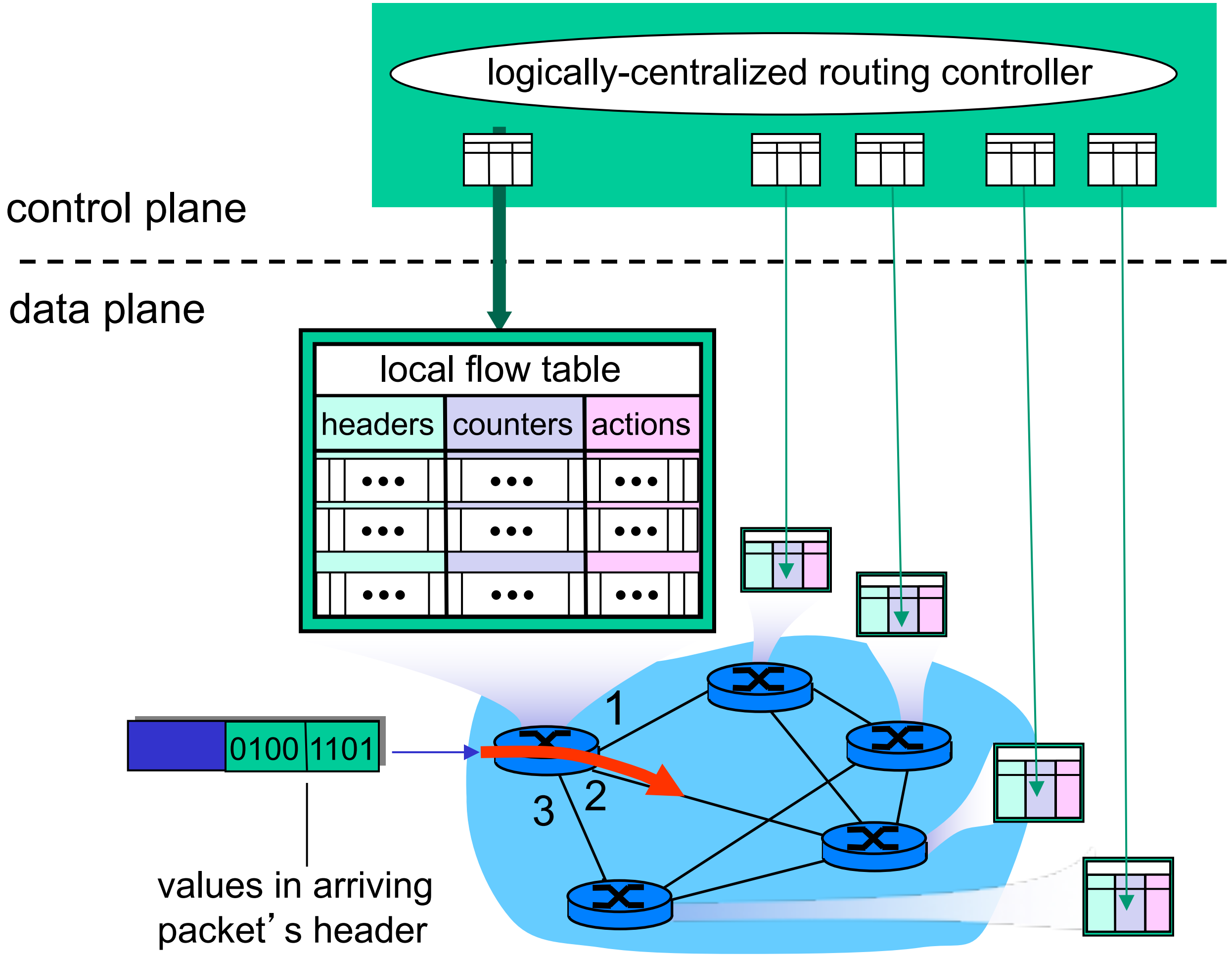
Distance-Vector:

- other nodes do not have info to verify

Software Defined Networking (SDN)

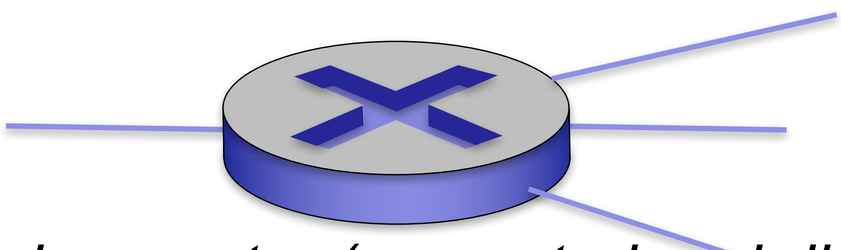
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow data plane abstraction

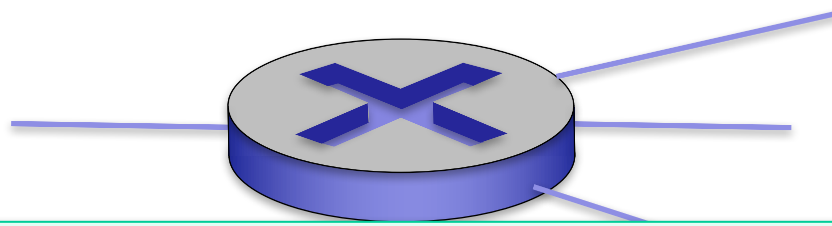
- *flow*: defined by header fields (may across multiple protocol layers)
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



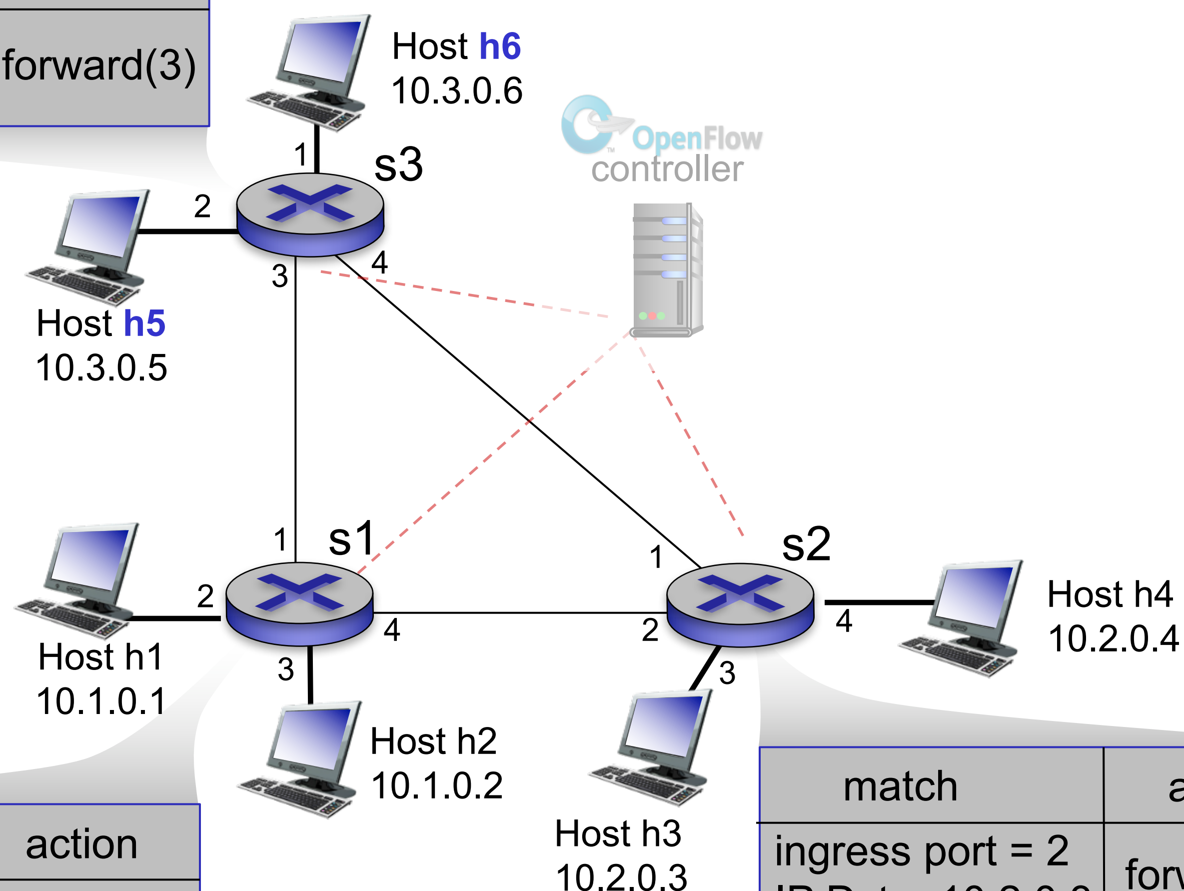
* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow example

Example: datagrams from hosts **h5** and **h6** should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Routing: What we have learned so far

- Link-state routing (Dijkstra) algorithm: **each node** computes the shortest paths to all the other nodes based on the complete *topology map*
- Distance Vector (Bellman-Ford) routing algorithm: **each node** computes the shortest paths to all the other nodes based on its *neighbors distance to all destinations*
- Today: routing protocols to implement them
  - Distance vector: $D_x(y) = \min \{c(x,v) + D_v(y) \}$
    the protocol: a node must send its distance to all destinations to all its neighbors
  - Link-state $\Rightarrow$the protocol must deliver the topology map to all the nodes in the network

What else a routing protocol must do

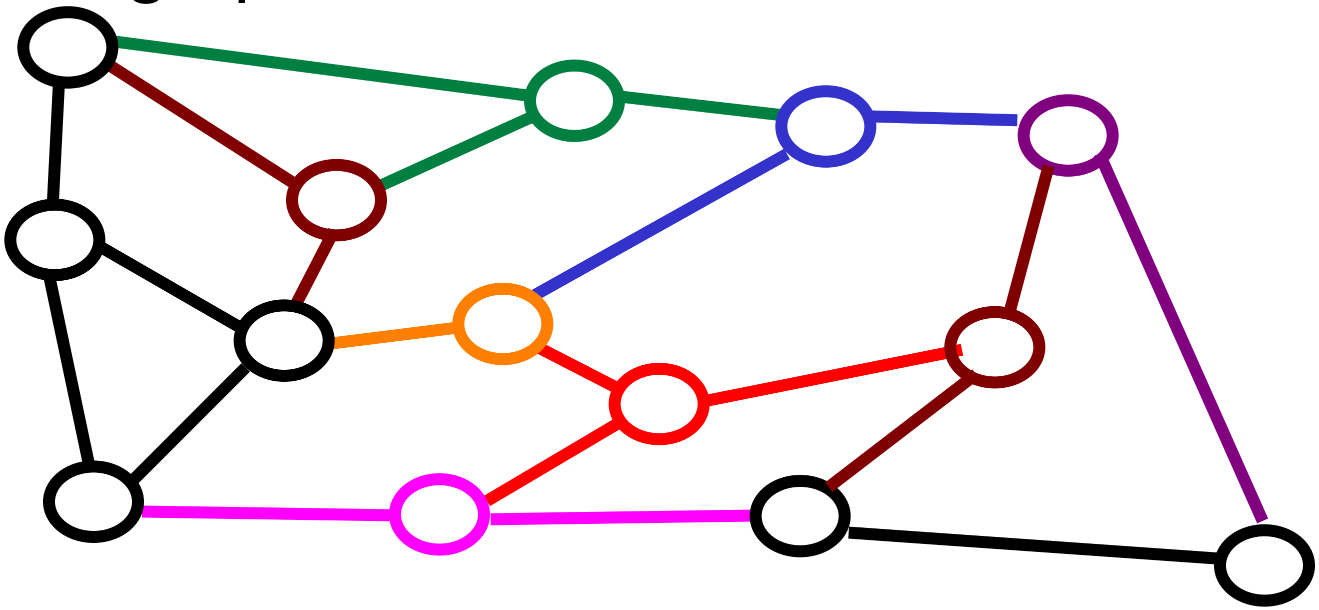
- Recover from packet losses in routing data delivery
- Monitor link and neighbor nodes status
  - Once a failure is detected, needs to inform the rest of the network quickly (directly or indirectly)
- Flush obsolete information out of the system

OSPF: Open Short Path First

https://tools.ietf.org/html/rfc2328

OSPF: Building a complete network graph using Link State

- Every node broadcasts a piece of the topology graph
- assemble all the pieces together, you get the complete graph



Then each node carries out its <u>own</u> routing calculation *independently*

OSPF (Open Shortest Path First)

- Given: each node knows its directly connected neighbors & the link distance to each neighbor
- Each node periodically broadcasts its link-state to the *entire* network
  - **Delivered by raw IP packet (protocol ID = 89)**

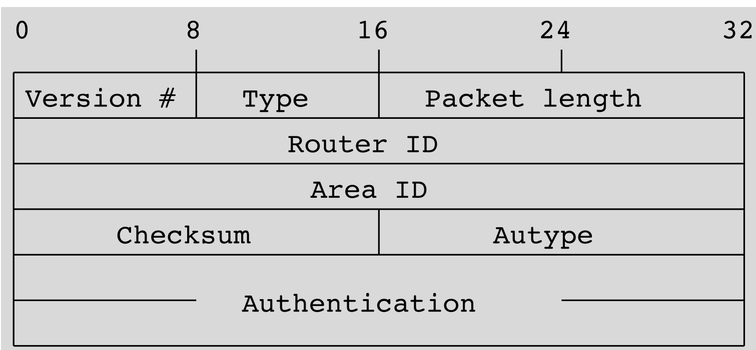
No.		Time	Source	Destination
2		1.663948	10.0.0.6	224.0.0.5
3		3.584090	10.0.0.10	224.0.0.5
4		4.894103	10.0.0.1	224.0.0.5
5		4.894132	10.0.0.5	224.0.0.5

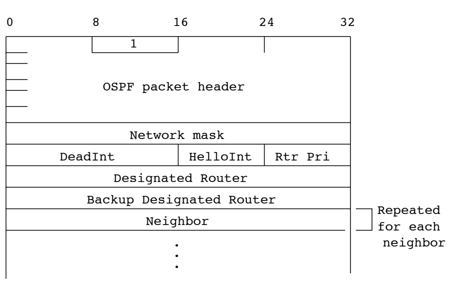
```
▷ Frame 3: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
▷ Frame Relay
▷ Internet Protocol Version 4, Src: 10.0.0.10 (10.0.0.10), Dst: 224.0.0.5 (224.0.0.5)
▽ Open Shortest Path First
    ▷ OSPF Header
    ▷ OSPF Hello Packet
    ▷ OSPF LLS Data Block
```

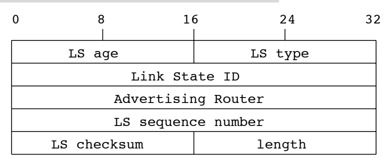
https://www.cloudshark.org

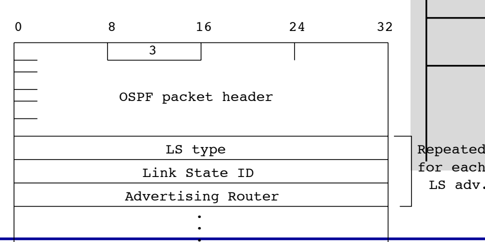
OSPF (Open Shortest Path First)

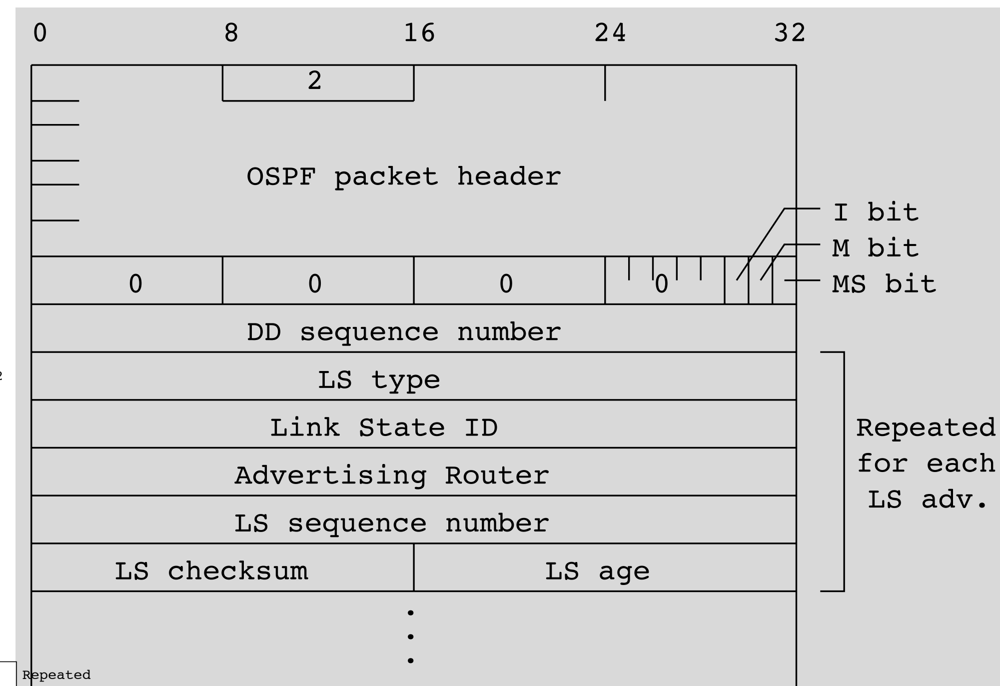
- Link-State Packet (LSP): one entry per neighbor router
  - ID of the node that created the LSP
  - a list of direct neighbors, with link cost to each
  - sequence number (SEQ) for this LSP message
  - time-to-live (TTL) for information carried in this LSP











How OSPF Works

- When neighboring routers discover each other for the first time: Exchange their link-state databases
- Link failure detection
  - Neighbor nodes send HELLO msg to each other periodically
  - Not receiving HELLO message for long enough time → **failure** → Trigger new Link State Update to neighbors
- In the absence of failure: send out update every **30 minutes**

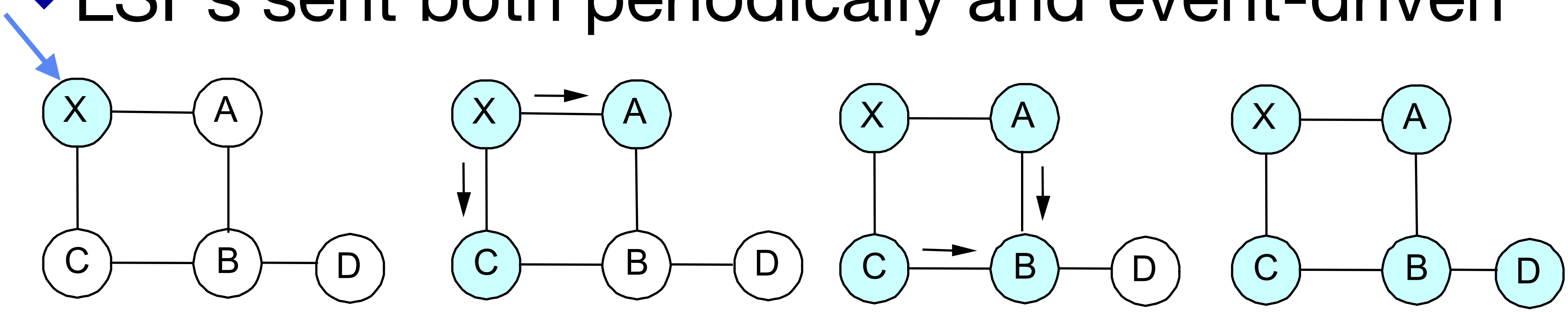
Link-State Routing Protocol

The routing daemon running at each node:

- Generates its own LSP periodically with increasing sequence #
- Stores most recent LSP from all other nodes
  - decrement TTL of stored LSP; discard a LSP when its TTL=0
- Process received updates to build & maintain topology graph
  - Route computation using Link-State algorithm
- Forward most recent LSPs

Reliable Flooding of LSP

- forward each received new LSP to all neighbor nodes but the one that sent it
  - each LSP is reliably delivered over each link
  - use the sender-ID and SEQ in a LSP to detect duplicates
- LSPs sent both periodically and event-driven



Q: How many LSP msgs traverse each link in the absence of failures in an hour?

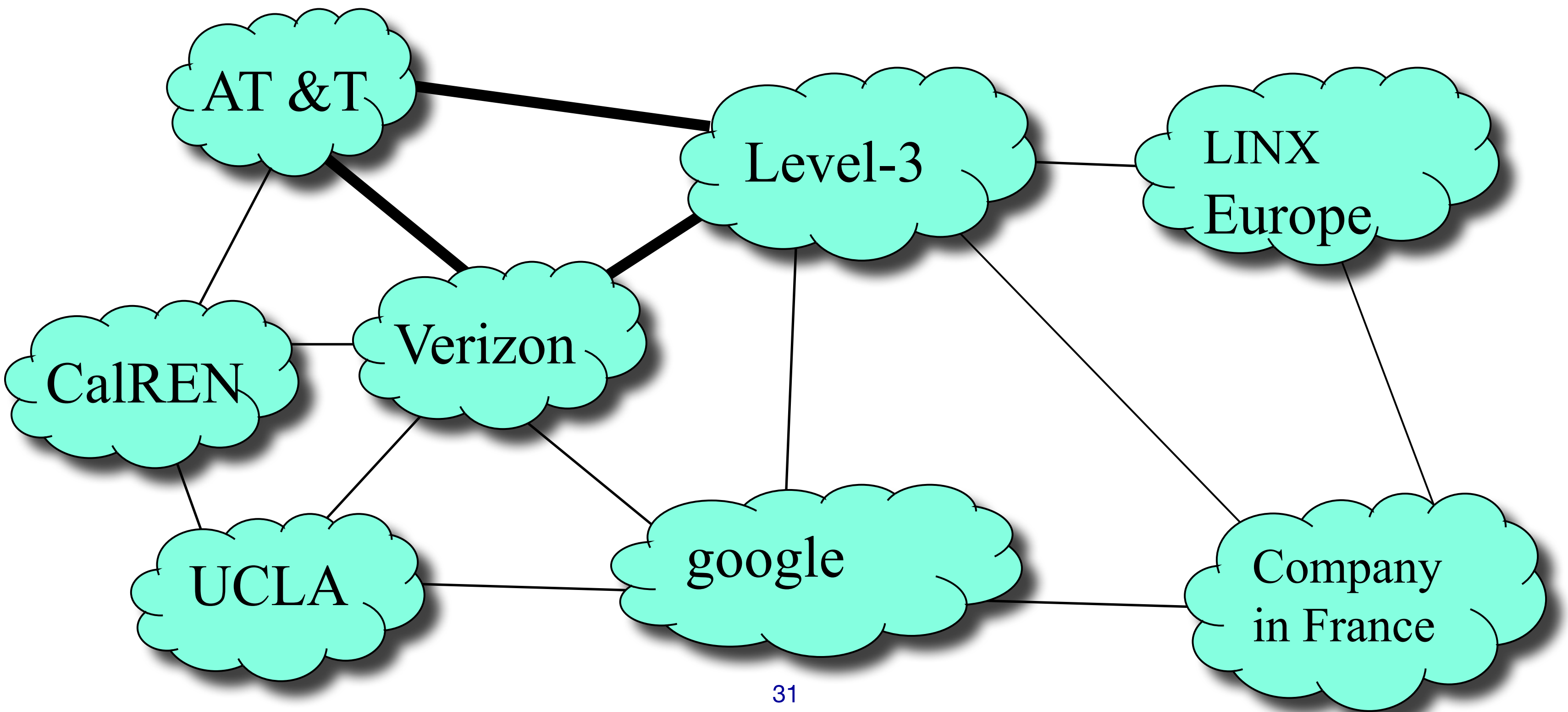
ROUTING ON THE INTERNET

Routing in the Internet

- So far: all routers faithfully execute the same routing protocol
  - Goal: Find best path (sequence of routers) through network from source to destination
    - Based on delay, loss, bandwidth, or other measures
- The Global Internet: interconnection of a large number of Autonomous Systems (AS)
  - Stub AS: end user networks (corporations, campuses)
    - Multihomed AS: stub ASes that are connected to multiple service providers
  - Transit AS: Internet service provider

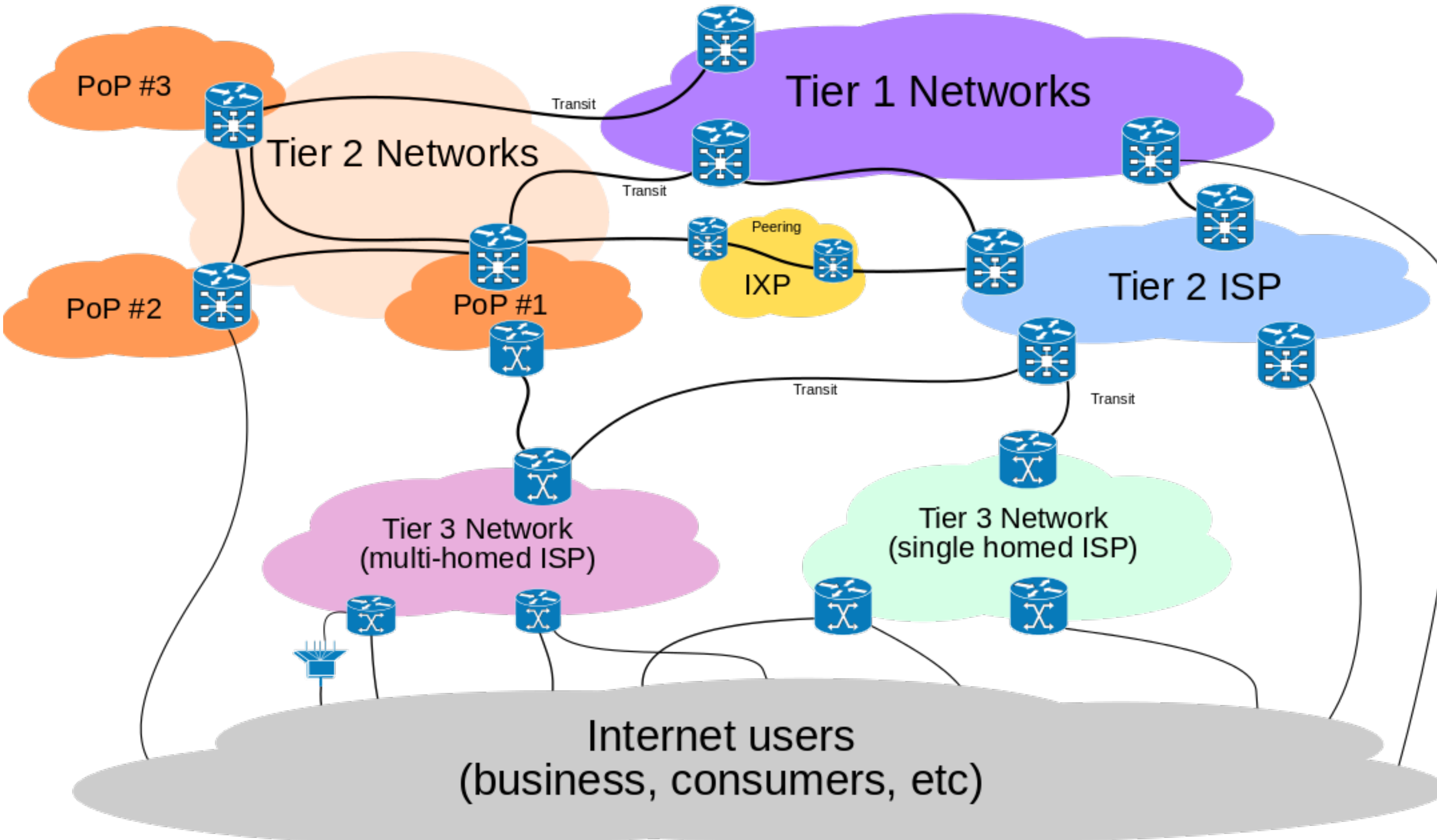
Internet routing structure: 2-level hierarchy

- Intra-AS (within a campus, or within an ISP)
  - Intra-Domain Routing: RIP, OSPF (and a few others)
- Inter-AS (between ISPs, or between stub and transit ASes)
  - Inter-Domain Routing: BGP (Border Gateway Protocol)



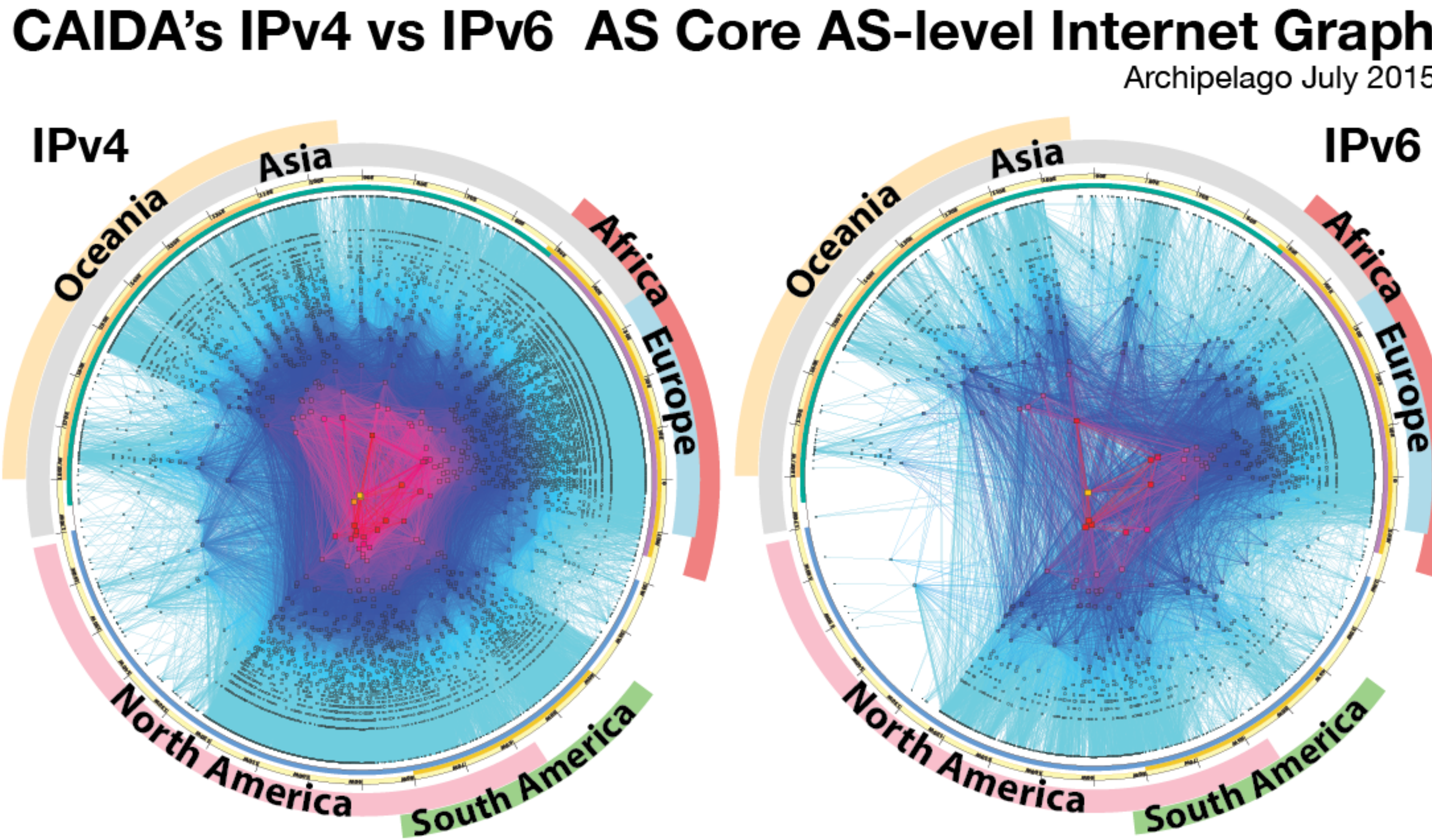
Internet Structure

FYI



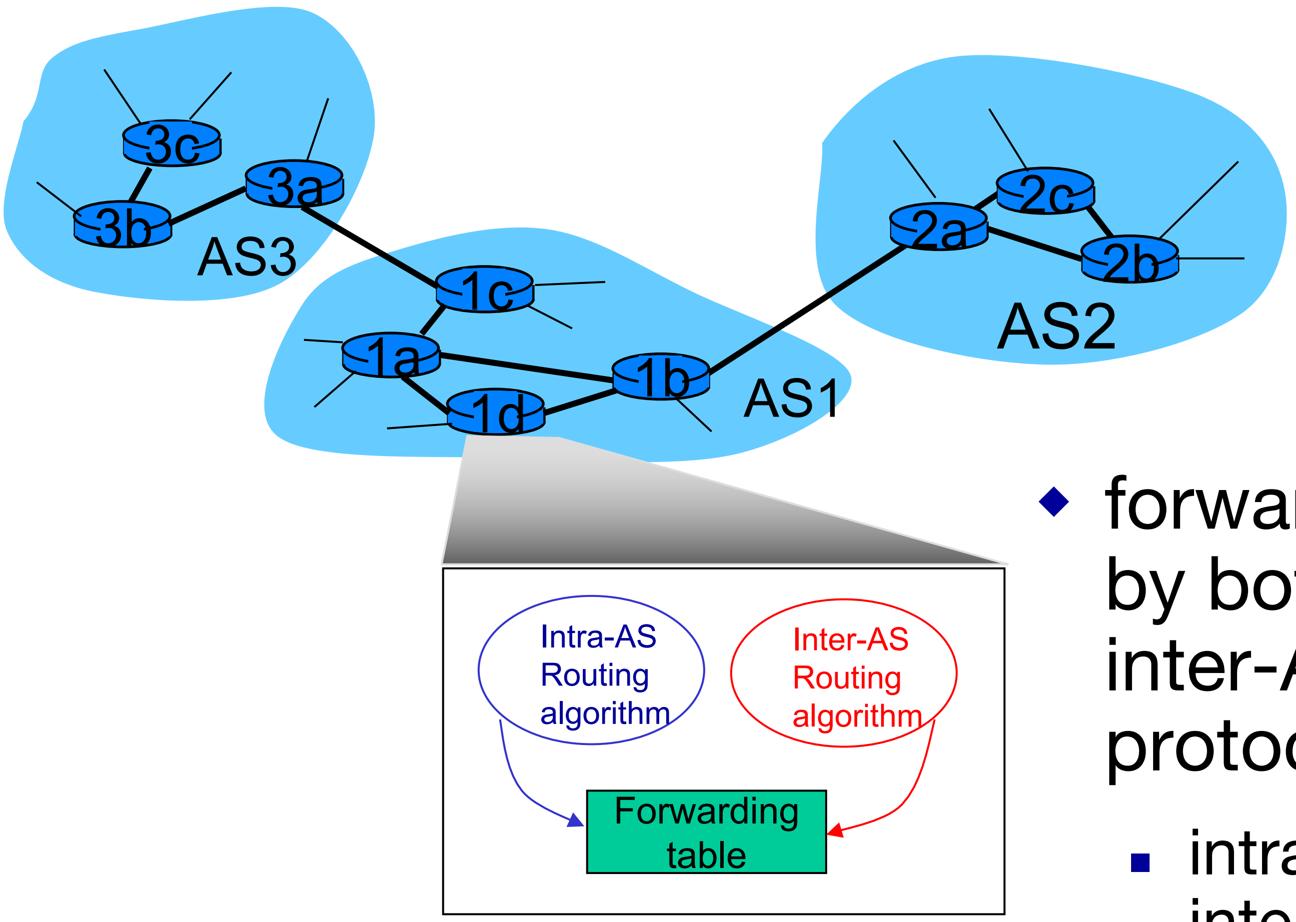
Internet Interconnections as a Graph

FYI



Copyright © 2015 UC Regents. All rights reserved.

Interconnected ASes

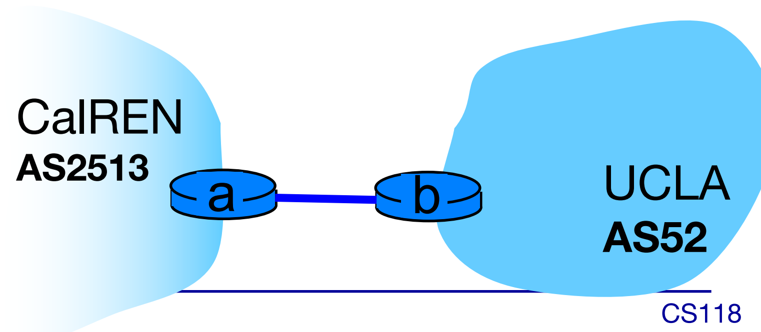


- forwarding table filled by both intra- and inter-AS routing protocols
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

BGP: Border Gateway Protocol

BGP provides each AS a means to:

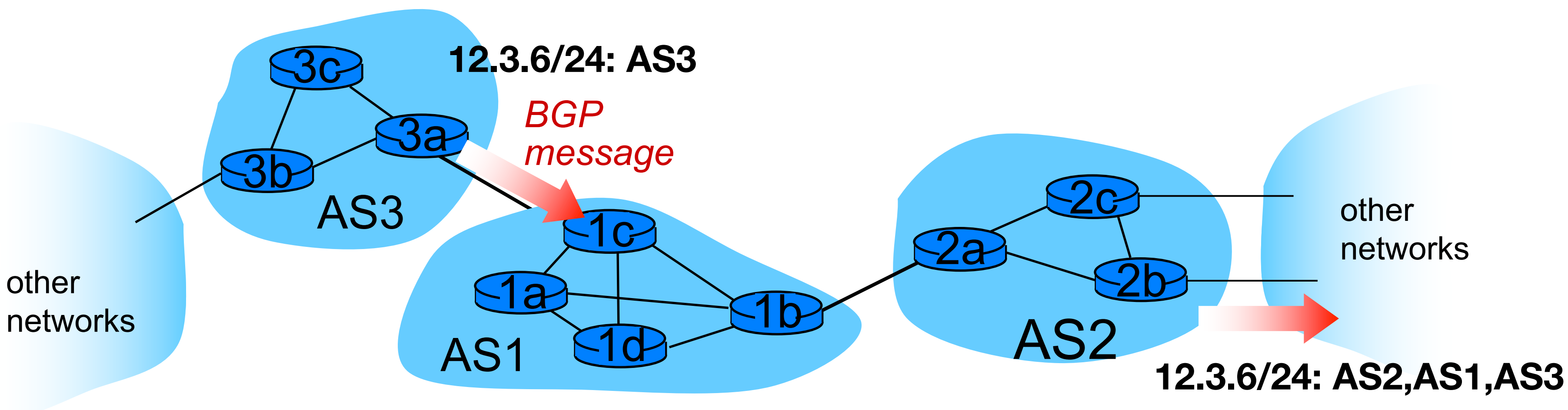
- Obtain each subnet reachability (= an IP address prefix) information from neighboring ASes.
- Propagate the reachability information to all routers internal to the AS.
- Determine "*good*" routes to each destination prefix based on reachability information and policy.
- advertise its own prefixes to the rest of the Internet



BGP basics: distributing path information

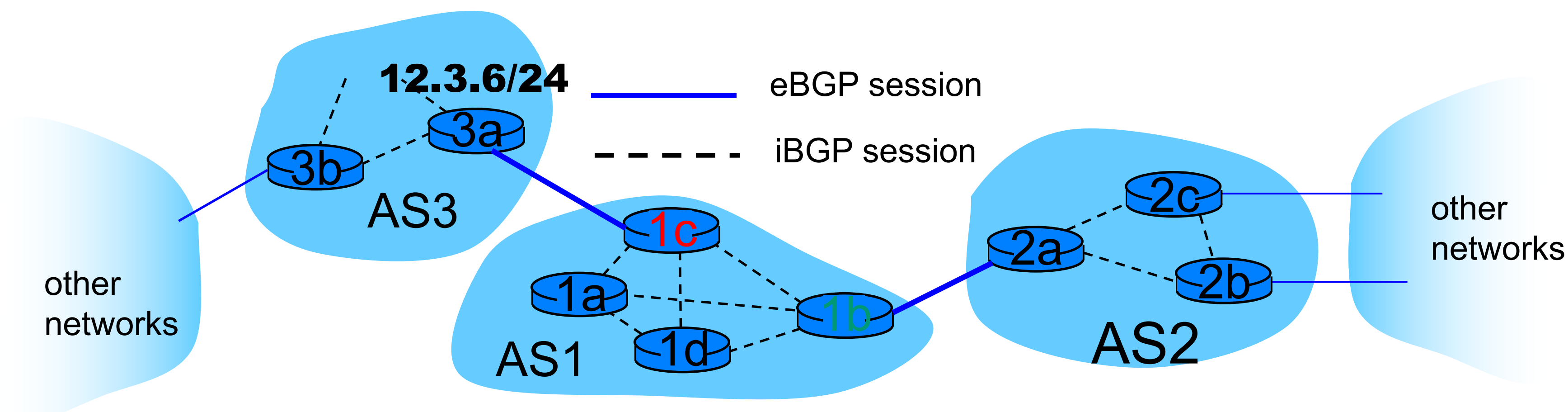
important

- 2 neighbor BGP routers establish a BGP session over a (semi-permanent) **TCP connection** to exchange routing info
  - advertising paths to destination network prefixes ("path vector" protocol)
- when AS3 advertises a prefix to AS1:
  - AS3 promises it will forward packets towards that prefix



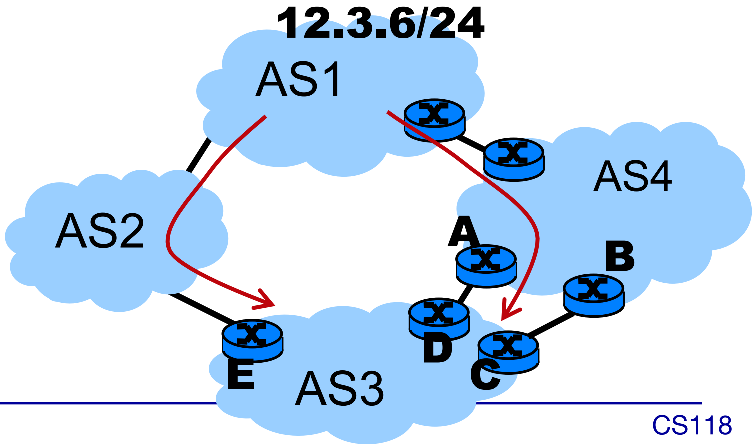
BGP basics: eBGP and iBGP

- eBGP: BGP session between two different ASs
  - e.g. the BGP session between 3a and 1c
- iBGP: BGP session between routers in the same AS
  - Router 1c uses iBGP to distribute new prefix info (e.g. 12.3.6.0/24) to all routers in AS1
  - when router learns of new prefix, it creates entry for prefix in its forwarding table.
  - Router 1b may re-advertise this reachability info to AS2 over 1b-to-2a eBGP session



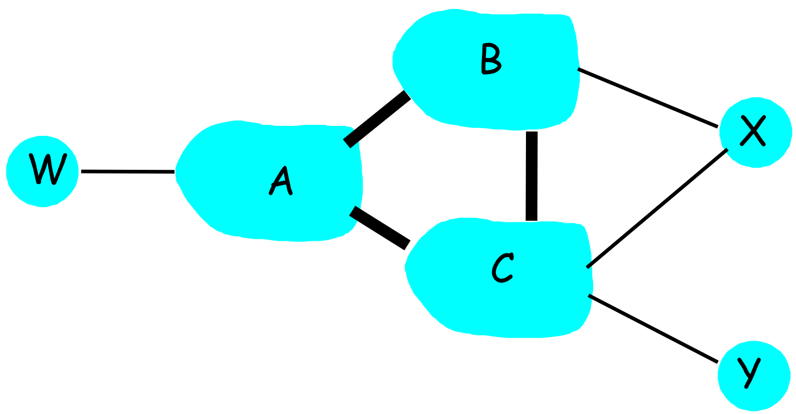
Path attributes and BGP routes

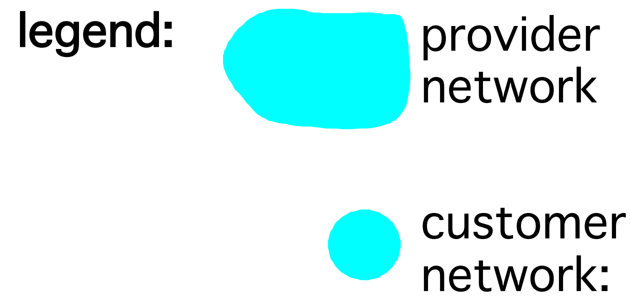
- Each advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- 3 important attributes:
  - AS-PATH: contains a list of ASes through which prefix advertisement has passed
    - When Router-C receives the announcement: 12.3.6/24: AS4, AS1
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
    - may be multiple links from current AS to next-hop-AS
  - Local-Preference: indicates policy preference in path selection
    - Injected into BGP update at border router (by C, D and E)
    - Used by internal routers to decide whether going through AS2 or AS4 to reach 12.3.6/24



BGP routing policy: a provider advertises all prefixes to its customer ASes; a customer does not advertise prefixes between providers

important





no valley policy

A,B,C are provider network ASes

X,W,Y are customer ASes (of provider networks)

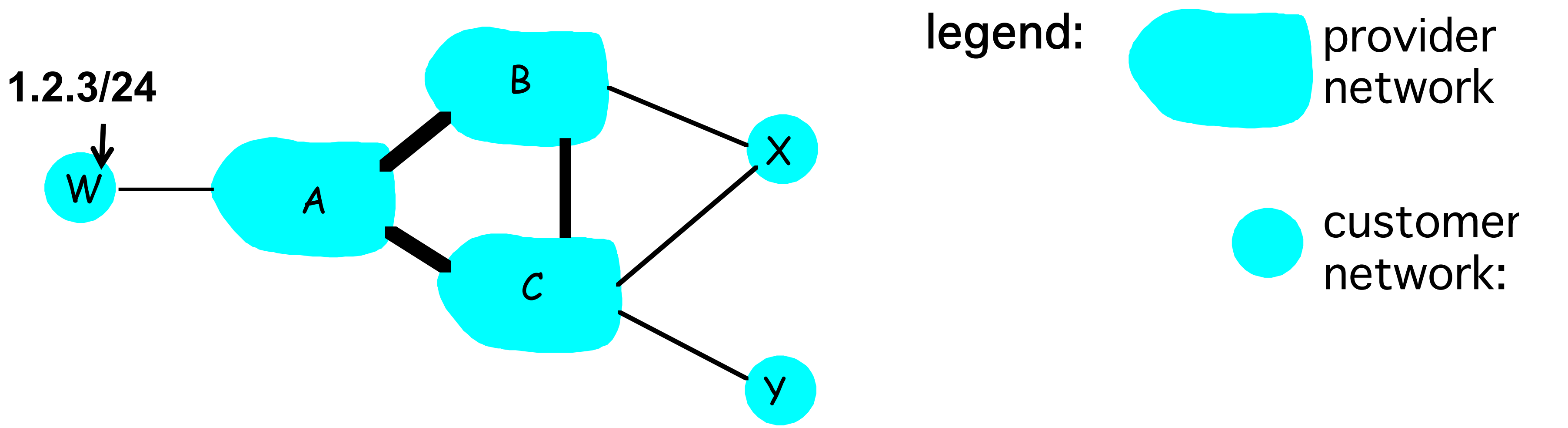
X is dual-homed: attached to two provider networks

- X does not want to forward traffic from B to C
- .. so X will not advertise to B any route it learned from C

(i.e. IP prefixes with AS path)

BGP routing policy: a provider does not pass prefixes that are not its own to another providers

important



A advertises to B the path **[1.2.3/24: AW]**

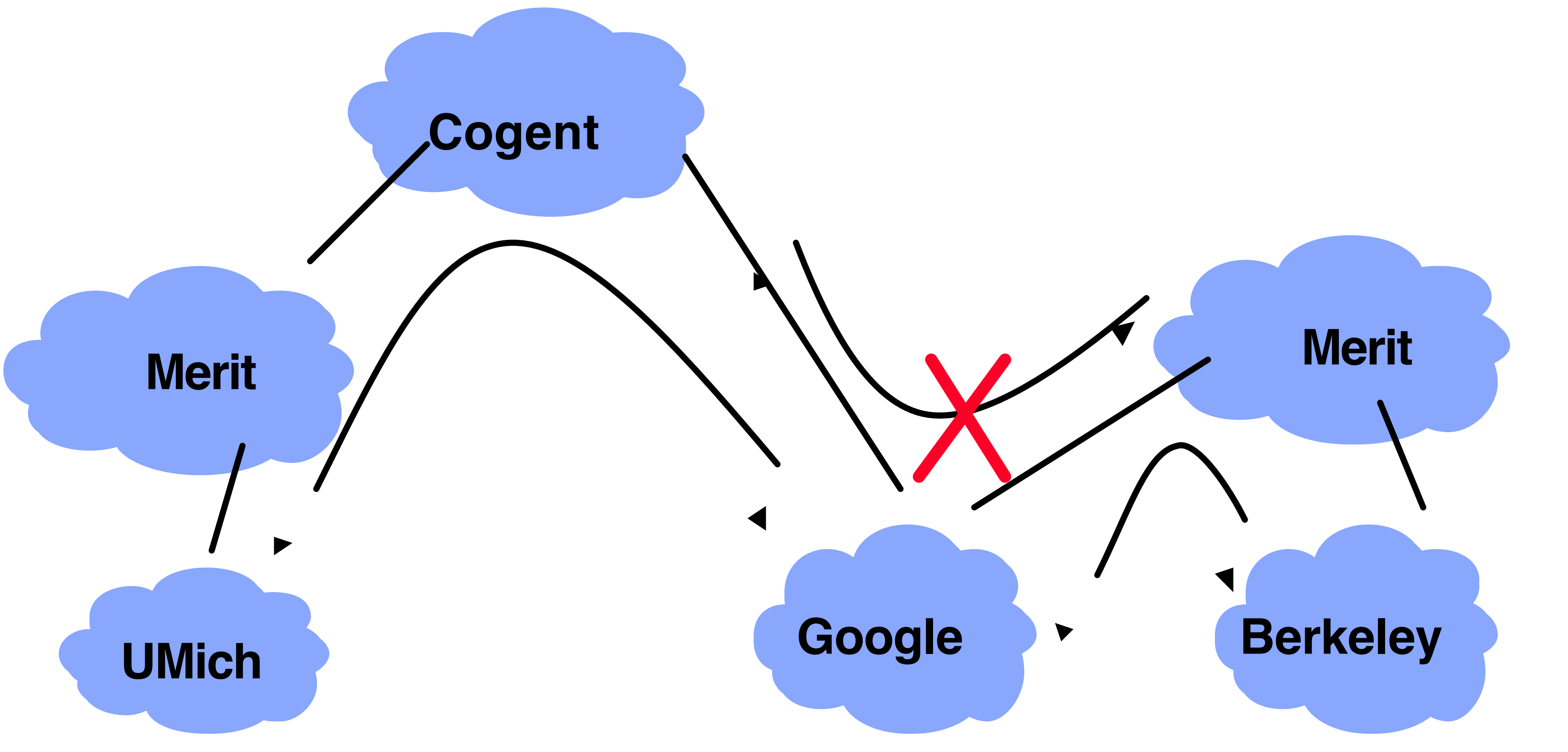
B advertises to X the path **[1.2.3/24: BAW]**

Would B advertise to C the path **[1.2.3/24: CBAW]**?

- No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
- B wants to route *only* to/from its customers!

No Valley Policy

important



Why different Intra- and Inter-AS routing ?

- Policy:
  - Inter-AS: admin wants control over how its traffic routed, who routes through its net.
  - Intra-AS: single admin, so no policy decisions needed
- Scale:
  - hierarchical routing saves table size, reduced update traffic
- Performance:
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

A Quick Summary of Internet Routing

- OSPF: a link-state routing protocol
  - Each router sends Link-State Packet containing
    - ID of the node that created the LSP
    - a list of direct neighbors, with link cost to each
    - sequence number (SEQ) for this LSP message
    - time-to-live (TTL) for information carried in this LSP
  - LSPs are sent periodically, or whenever changes happen
    - flooded everywhere, reliably
  - Neighbor routers use Hello msgs to keep track each other
- BGP: a path-vector (like distance vector by with paths) routing protocol
  - Running over TCP connection
  - Propagate reachable IP prefixes

Routing Security

- Intra-domain routing
  - Controlled by a single party, so could be secured using shared secrets
  - Still there is an issue if routers are getting compromised
- Inter-domain routing
  - Multiple parties
  - Not everyone behaves correctly
    - Configuration errors
    - Malicious activity

Unintended Behavior in BGP

- Route hijacking
  - an AS announcing somebody else's prefix(es) as they own
  - depending on AS-PATH, some part of the Internet will prefer hijacked path
    - Google DNS hijacking in 2014 by Turkish ISPs
    - Youtube blackout in 2008 caused by Pakistan ISPs
    - many more
- Man-in-the-middle (diverting routes)
- Announcing unused prefixes
  - routes disappear after finishing some communications, e.g., sending spam
- Using unallocated AS numbers
  - avoiding legal tracing

http://cyclops.cs.ucla.edu/