Lecture 6

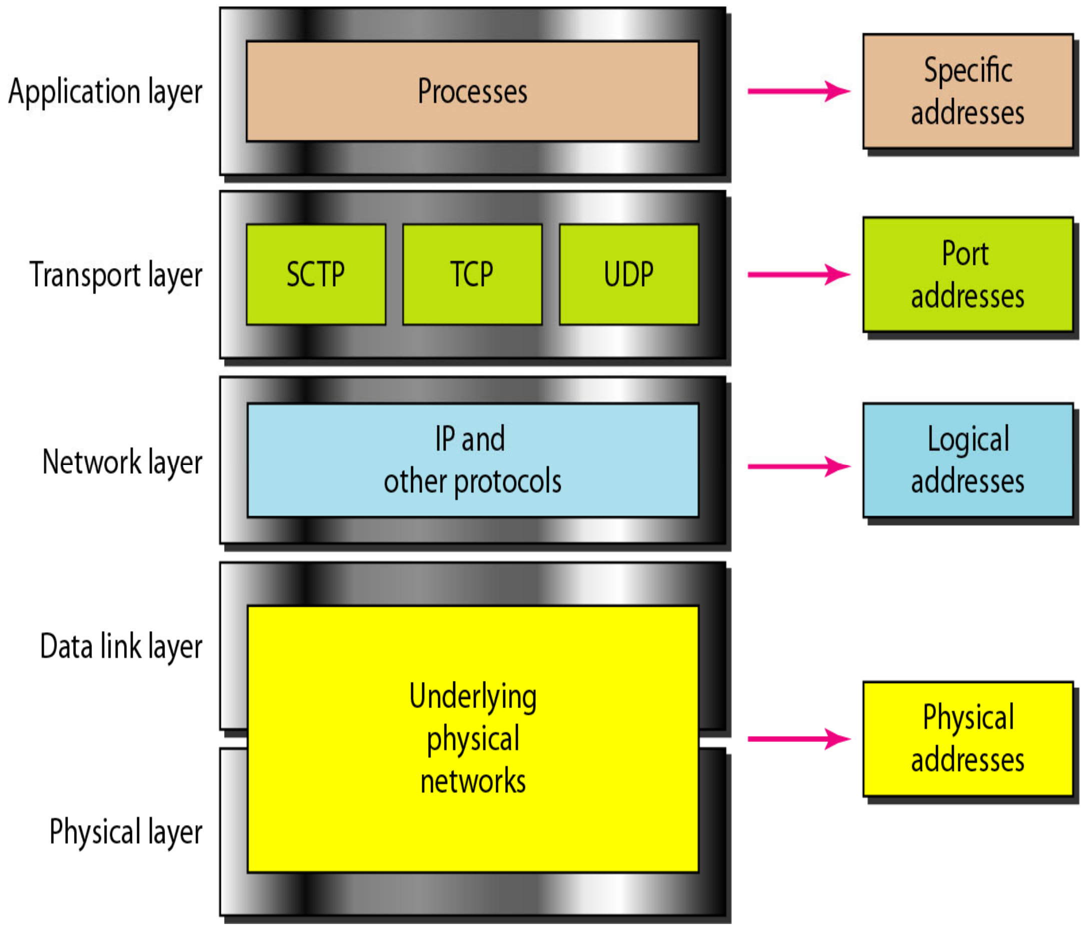
Network Layer – Routing Protocols

ELEC 3506 9506
Communication Networks

Dr Wibowo Hardjawana
School of Electrical and Information Engineering

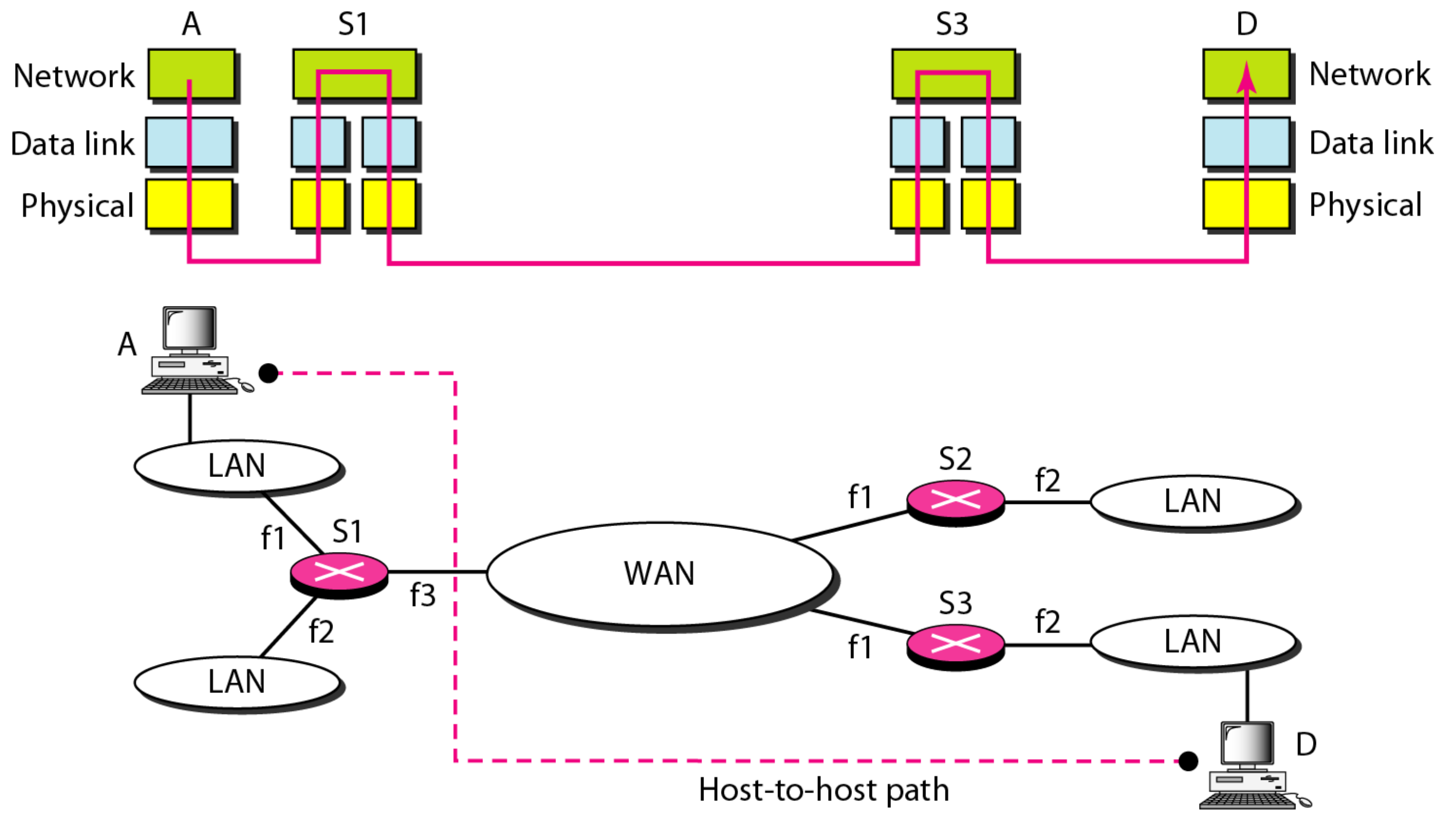
Topics for the Day

- Datagram Networks
- Introduction to Routing
- Routing Protocols
- Control Protocols
- IPv6



Link Layer vs. Network Layer Delivery

- End-to-End delivery at the **Network Layer**



- Hop-to-Hop delivery at the **Data Link Layer**

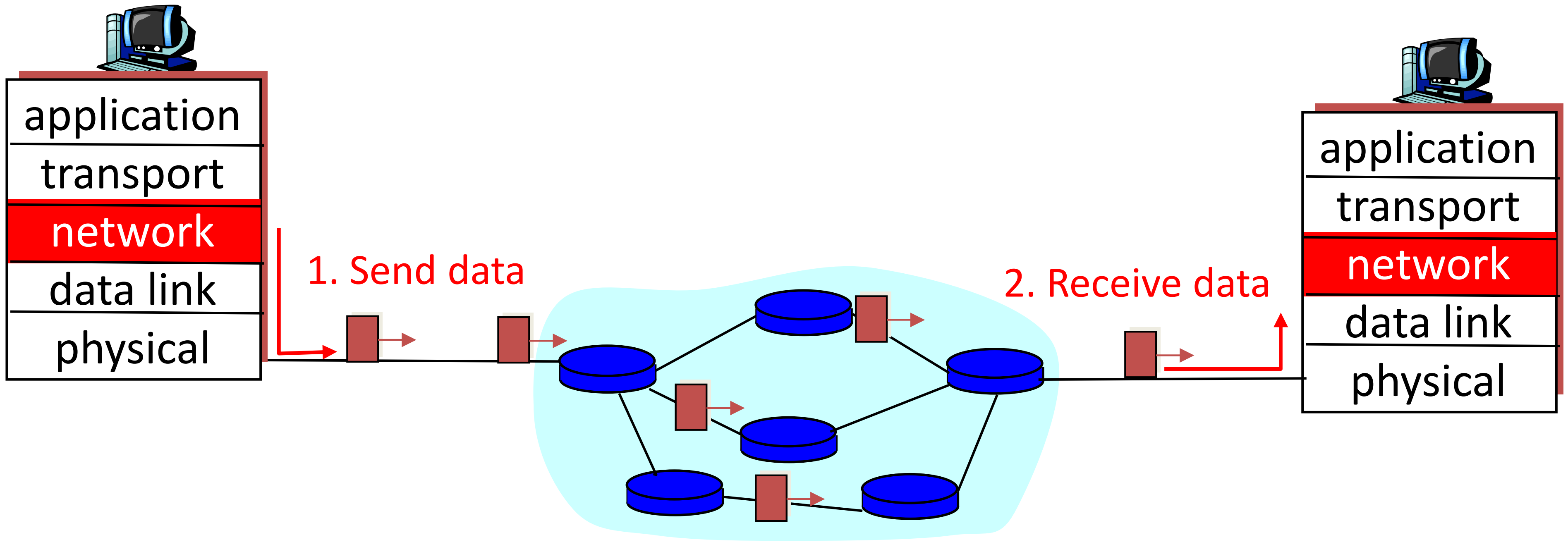
  A Source-to-destination delivery (End-to-End delivery or Host-to-Host delivery) may contain multiple hops **(Recall the need of Network Layer Address)**

Datagram Networks

- Network Layer is designed to solve the problem of
  - host-to-host delivery
  - Routing packets over multiple networks
- A unique addressing scheme:
  - Each device has its own physical address (MAC address)
  - Logical IP address must be unique to identify different hosts in internetwork
- The service is provided by the datagram networks are:
  - Connectionless (no setup and teardown phases)
  - Best effort delivery – no service guarantee
    - No Flow Control
    - No Error Control
  - Unreliable
  - It does not make any attempt to recover from the failure
  - BUT fast

Datagram Networks – Internet Model

- No call setup at network layer
- Packets are typically routed using the destination host ID
  - Packets between same source-destination pair may take different paths



Introduction to Routing

- Routing is the act of moving information across an internetwork from a source to a destination.
- Routing occurs at Layer 3 (the network layer).
- Routing involves two basic activities:
  - Determining optimal routing paths (Complex)
  - Transporting packets through an internetwork (Packet switching – very straightforward)

- Routing is often contrasted with bridging
  - Bridging occurs at Layer 2 (the link layer) of the OSI reference model, whereas **routing occurs at Layer 3 (the network layer)**

THE UNIVERSITY OF SYDNEY

Routing Path Determination

- Routing protocols use various mechanisms to evaluate what path will be the best for a packet to travel.
- To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information.
- Routers communicate with one another and maintain their routing tables.
- The routing update message is one such message that generally consists of all or a portion of a routing table.
- By analysing routing updates from all other routers, a router can build a detailed picture of network topology.

Autonomous System (AS)

- Routers belonging to one organization is said belong to one Autonomous System.
- Routers within the same AS run the same routing algorithm
- Routing algorithm running within an autonomous system is called an intra autonomous system routing protocol.
- Routers connecting different AS are called as gateway routers.
- Routing algorithm used by gateway routers are called as inter-autonomous system routing protocol.

Routed and Routing Protocols

- Routed protocols are protocols that are routed over an internetwork.
  - Internet Protocol (IP), AppleTalk, Novell NetWare
- Routing protocols, on the other hand, are protocols that implement routing algorithms.
  - Intra-Autonomous System Routing Protocols (interior gateway protocol (IGP)):
    - Routing Information Protocol (RIP)
    - Open Shortest Path First (OSPF)
  - Inter-Autonomous System Routing Protocols (exterior gateway protocol (EGP)):
    - Border Gateway Protocol (BGP)

Interior Gateway or Intra-AS Protocols

- There are three types of interior gateway protocols
  - Distance vector based protocol
    - Use number of hops to decide routing path, distance vector algorithm
    - Example: RIP
  - Link state based protocol
    - Recreate the exact topology of the entire network, e.g. each router in the network, its neighbors, its attached links, etc, Djikstra algorithm
    - Example: OSPF
  - Balanced hybrid protocol
    - Exchange more information than distance vector protocol
    - Does not require full topology and not so computationally intensive
    - Example: Enhanced Interior Gateway Routing Protocol (EIGRP)

We will cover RIP and OSPF in this course which is the basic of these protocols

Distance vector algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y.

Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

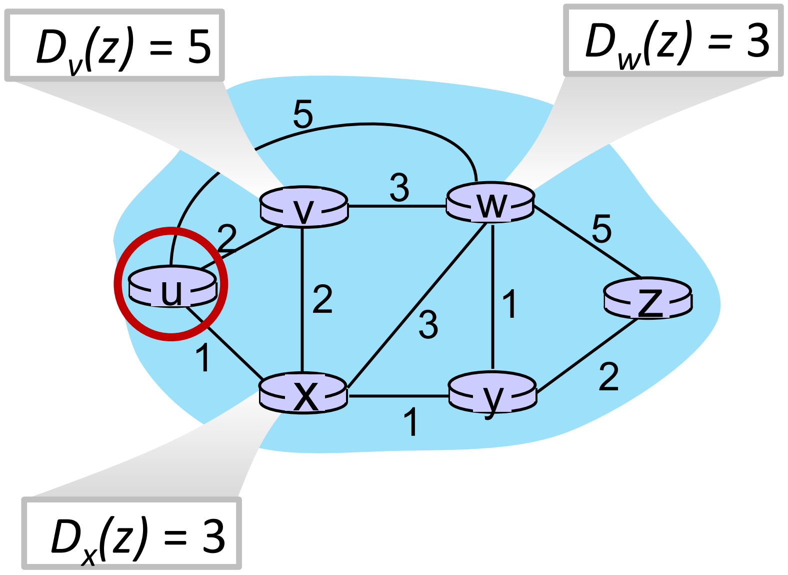
v's estimated least-cost-path cost to y

min taken over all neighbors v of x

direct cost of link from x to v

Bellman-Ford Example

Suppose that *u*'s neighboring nodes, *x,v,w,* know that for destination *z*:



Bellman-Ford equation says:

$$\begin{aligned} D_u(z) = \min \{ & c_{u,v} + D_v(z), \\ & c_{u,x} + D_x(z), \\ & c_{u,w} + D_w(z) \} \\ = \min \{ & 2 + 5, \\ & 1 + 3, \\ & 5 + 3 \} = 4 \end{aligned}$$

node achieving minimum cost if (x) is next hop on estimated least-cost path to destination (z)

Distance vector (DV) algorithm

key idea:

- from time-to-time, each node v sends its own distance vector estimate to neighbors, $D_v(z)$ where v and z are the source and destination
- when *x* receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c_{x,v} + D_v(y)\} \text{ for each node } y \in N$$

- under minor, natural conditions, the estimate *$D_x(y)$ converge to the actual least cost* $d_x(y)$

Distance vector algorithm

each node:

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

recompute DV estimates using DV received from neighbor

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

iterative, asynchronous: each local iteration caused by:

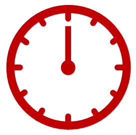
- local link cost change
- DV update message from neighbor
- **DV Table contains cost from other nodes**

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Distance vector: example

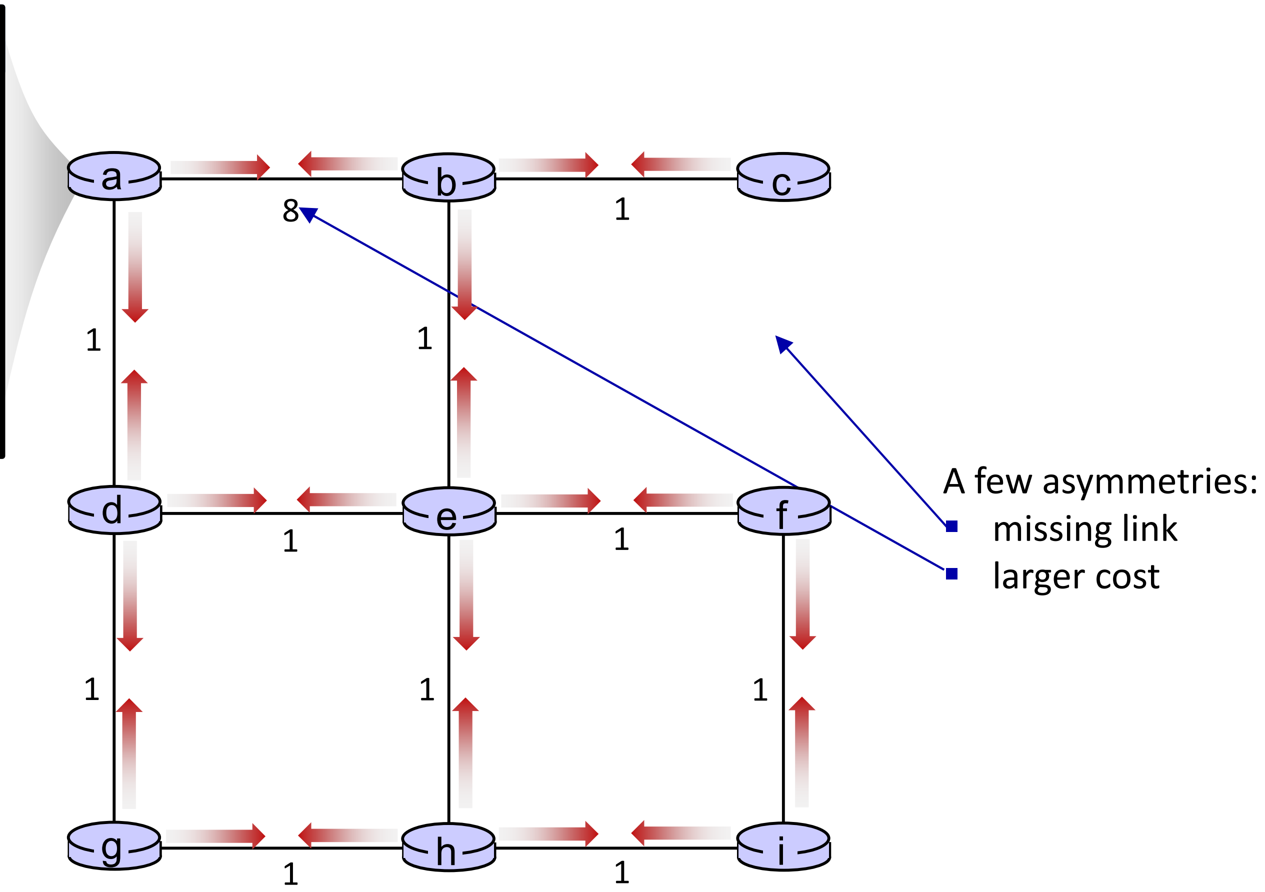
DV Table at a



t=0

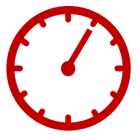
- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



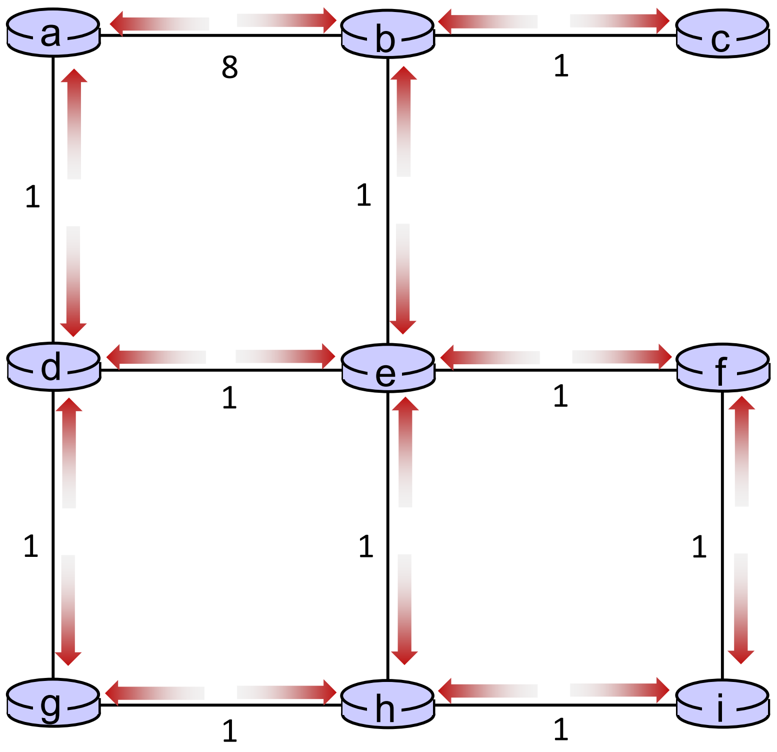
A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

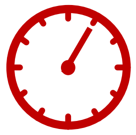


t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

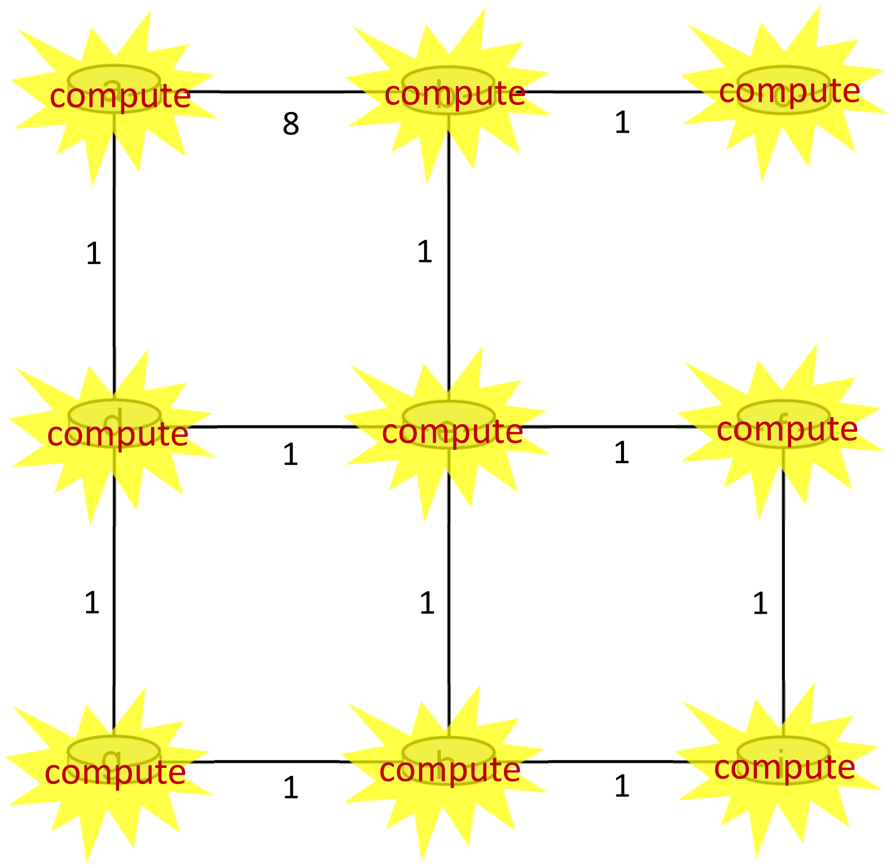


Distance vector example: iteration

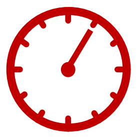


t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



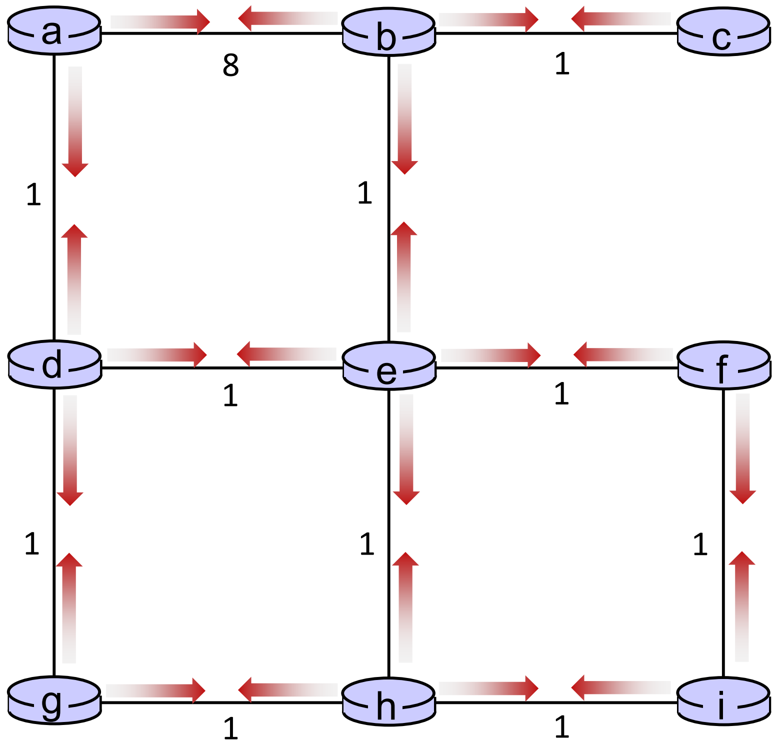
Distance vector example: iteration



t=1

All nodes:

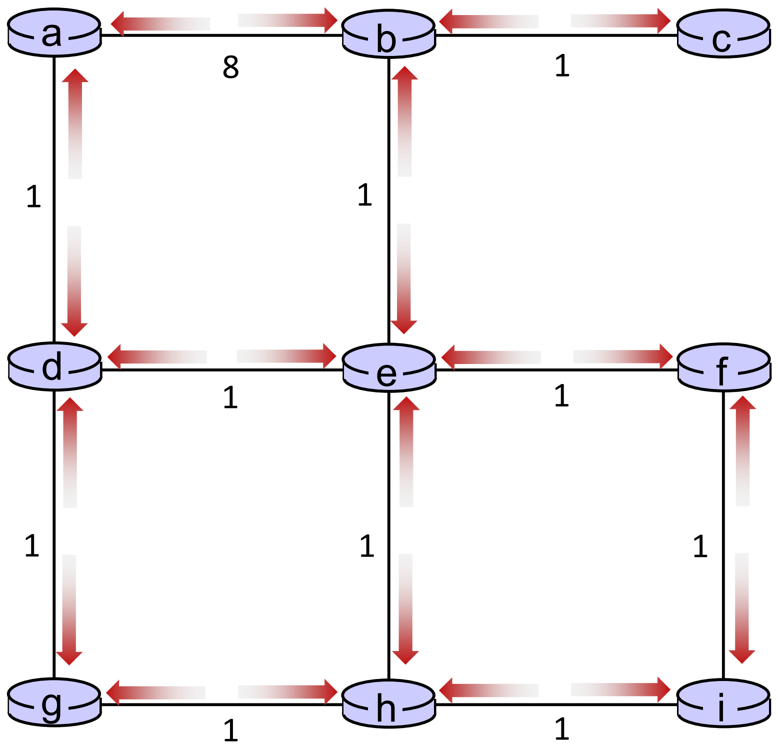
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



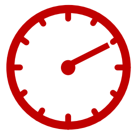
Distance vector example: iteration

t=2

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



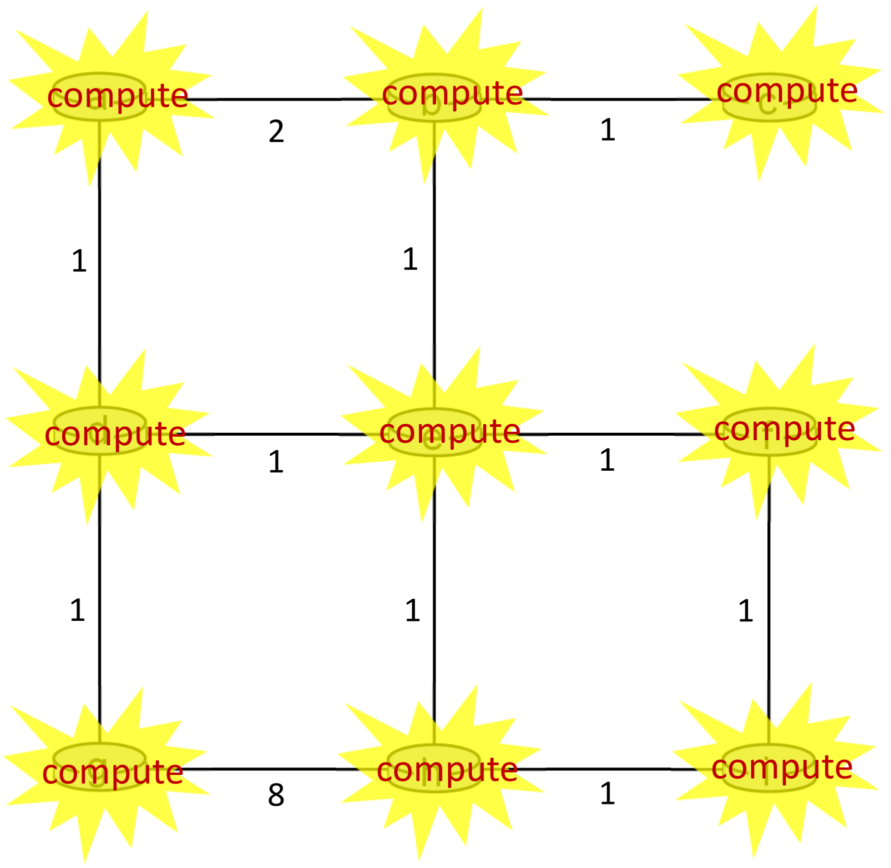
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

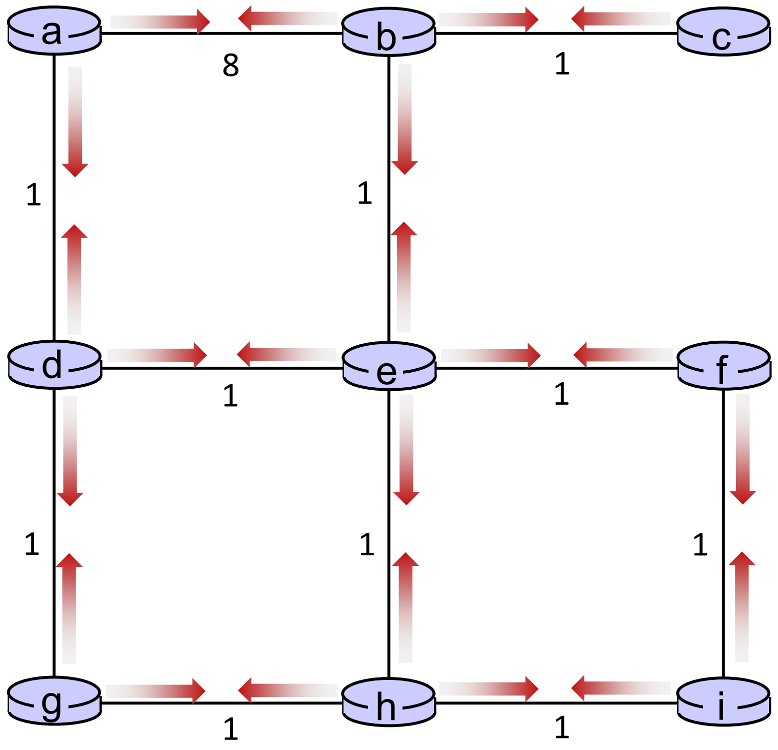


Distance vector example: iteration

t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

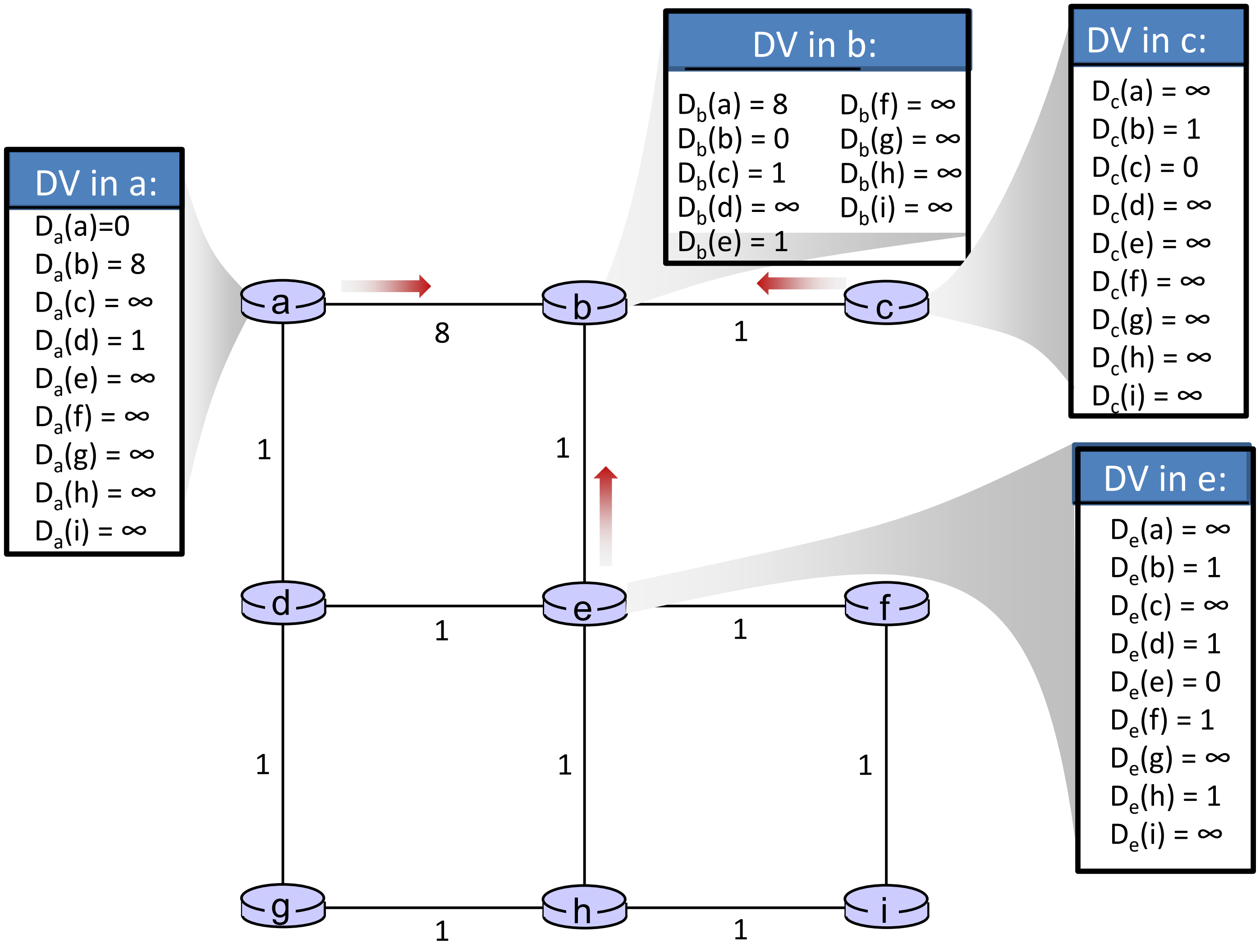
…. and so on

Let’s next take a look at the iterative *computations* at nodes

Distance vector example: computation

t=1

- b receives DVs from a, c, e



Distance vector example: computation

t=1

- b receives DVs from a, c, e, computes:

DV in a:

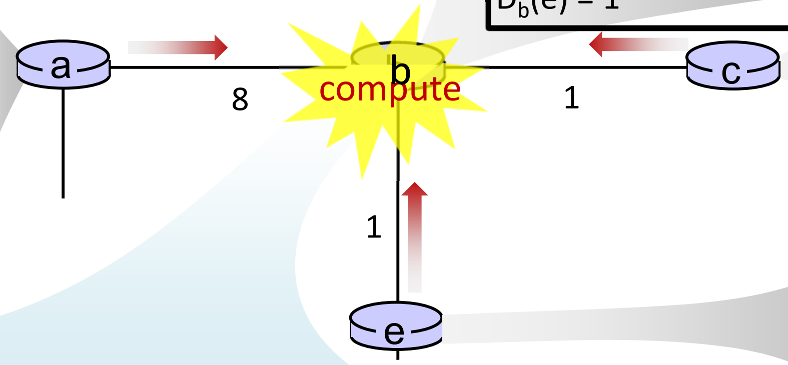
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = \infty$
$D_b(d) = \infty$	$D_b(i) = \infty$
$D_b(e) = 1$	

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$



DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

$D_b(a) = \min\{c_{b,a}+D_a(a),\ c_{b,c}+D_c(a),\ c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c),\ c_{b,c}+D_c(c),\ c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d),\ c_{b,c}+D_c(d),\ c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$

$D_b(e) = \min\{c_{b,a}+D_a(e),\ c_{b,c}+D_c(e),\ c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

$D_b(f) = \min\{c_{b,a}+D_a(f),\ c_{b,c}+D_c(f),\ c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g),\ c_{b,c}+D_c(g),\ c_{b,e}+D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty$

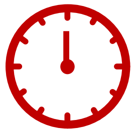
$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2$

$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$

DV in b:

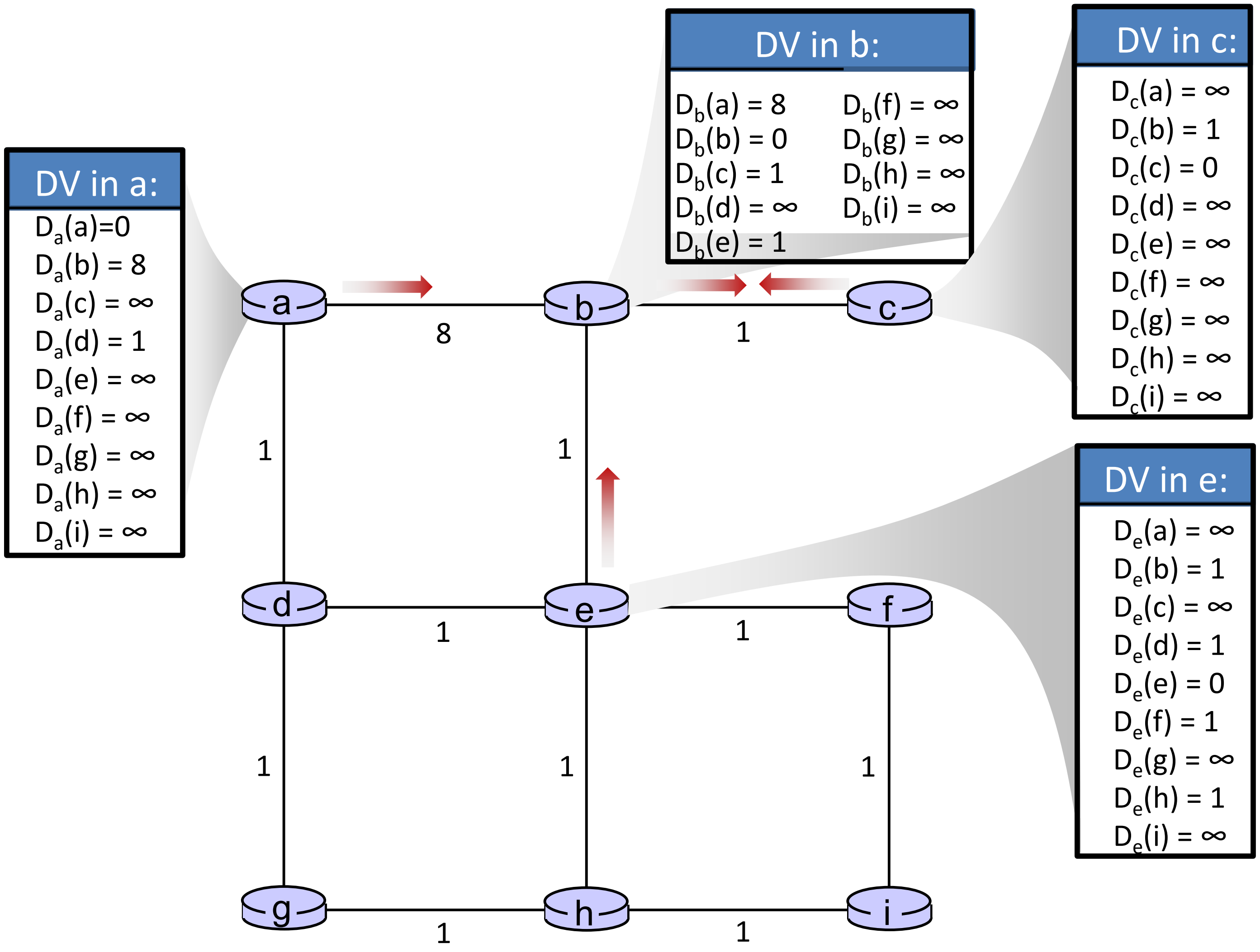
$D_b(a) = 8$	$D_b(f) =2$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = 2$
$D_b(d) = 2$	$D_b(i) = \infty$
$D_b(e) = 1$	

Distance vector example: computation



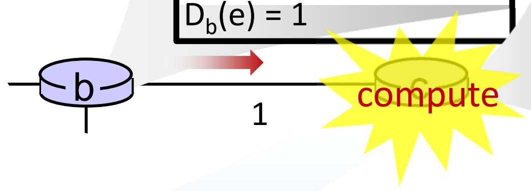
t=1

- c receives DVs from b



Distance vector example: computation

DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = \infty$
$D_b(d) = \infty$	$D_b(i) = \infty$
$D_b(e) = 1$	



DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

t=1

- c receives DVs from b computes:

$D_c(a) = \min\{c_{c,b}+D_b(a)\} = 1 + 8 = 9$

$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$

$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$

$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$

$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$

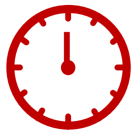
$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty = \infty$

$D_c(h) = \min\{c_{bc,b}+D_b(h)\} = 1+ \infty = \infty$

$D_c(i) = \min\{c_{c,b}+D_b(i)\} = 1+ \infty = \infty$

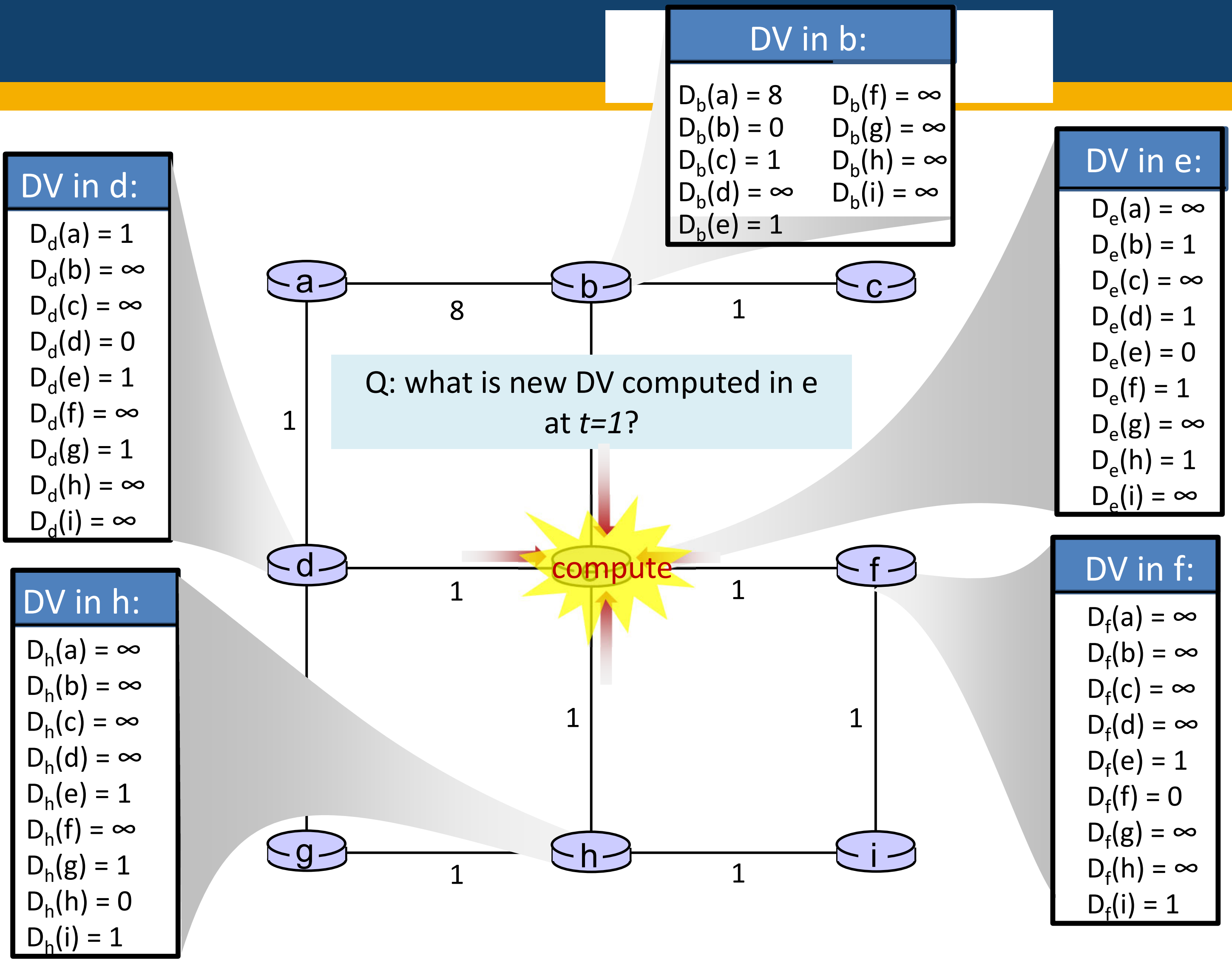
DV in c:
$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = 2$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

Distance vector example: computation



t=1

- e receives DVs from b, d, f, h



Q: what is new DV computed in e at *t=1*?

DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = \infty$
$D_b(d) = \infty$	$D_b(i) = \infty$
$D_b(e) = 1$	

DV in d:
$D_d(a) = 1$
$D_d(b) = \infty$
$D_d(c) = \infty$
$D_d(d) = 0$
$D_d(e) = 1$
$D_d(f) = \infty$
$D_d(g) = 1$
$D_d(h) = \infty$
$D_d(i) = \infty$

DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

DV in h:
$D_h(a) = \infty$
$D_h(b) = \infty$
$D_h(c) = \infty$
$D_h(d) = \infty$
$D_h(e) = 1$
$D_h(f) = \infty$
$D_h(g) = 1$
$D_h(h) = 0$
$D_h(i) = 1$

DV in f:
$D_f(a) = \infty$
$D_f(b) = \infty$
$D_f(c) = \infty$
$D_f(d) = \infty$
$D_f(e) = 1$
$D_f(f) = 0$
$D_f(g) = \infty$
$D_f(h) = \infty$
$D_f(i) = 1$

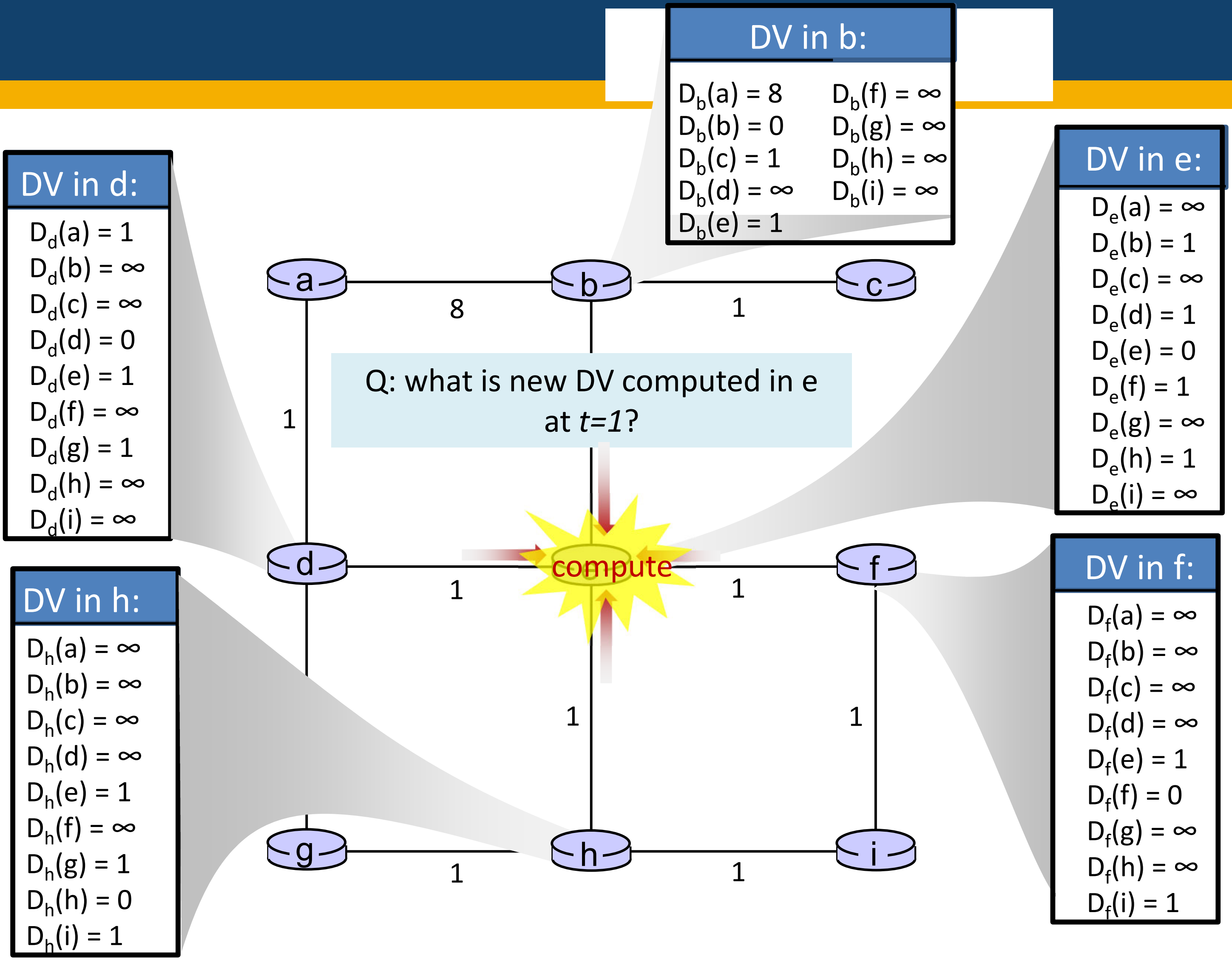
$$D_x(y) \leftarrow \min_v\{c_{x,v} + D_v(y)\}$$

for each node *y* ∊ *N*, at node x, achieved via neighboring nodes v

Distance vector example: computation

t=1

- e receives DVs from b, d, f, h



DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = \infty$
$D_b(d) = \infty$	$D_b(i) = \infty$
$D_b(e) = 1$	

DV in d:

$D_d(a) = 1$
$D_d(b) = \infty$
$D_d(c) = \infty$
$D_d(d) = 0$
$D_d(e) = 1$
$D_d(f) = \infty$
$D_d(g) = 1$
$D_d(h) = \infty$
$D_d(i) = \infty$

DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

DV in h:

$D_h(a) = \infty$
$D_h(b) = \infty$
$D_h(c) = \infty$
$D_h(d) = \infty$
$D_h(e) = 1$
$D_h(f) = \infty$
$D_h(g) = 1$
$D_h(h) = 0$
$D_h(i) = 1$

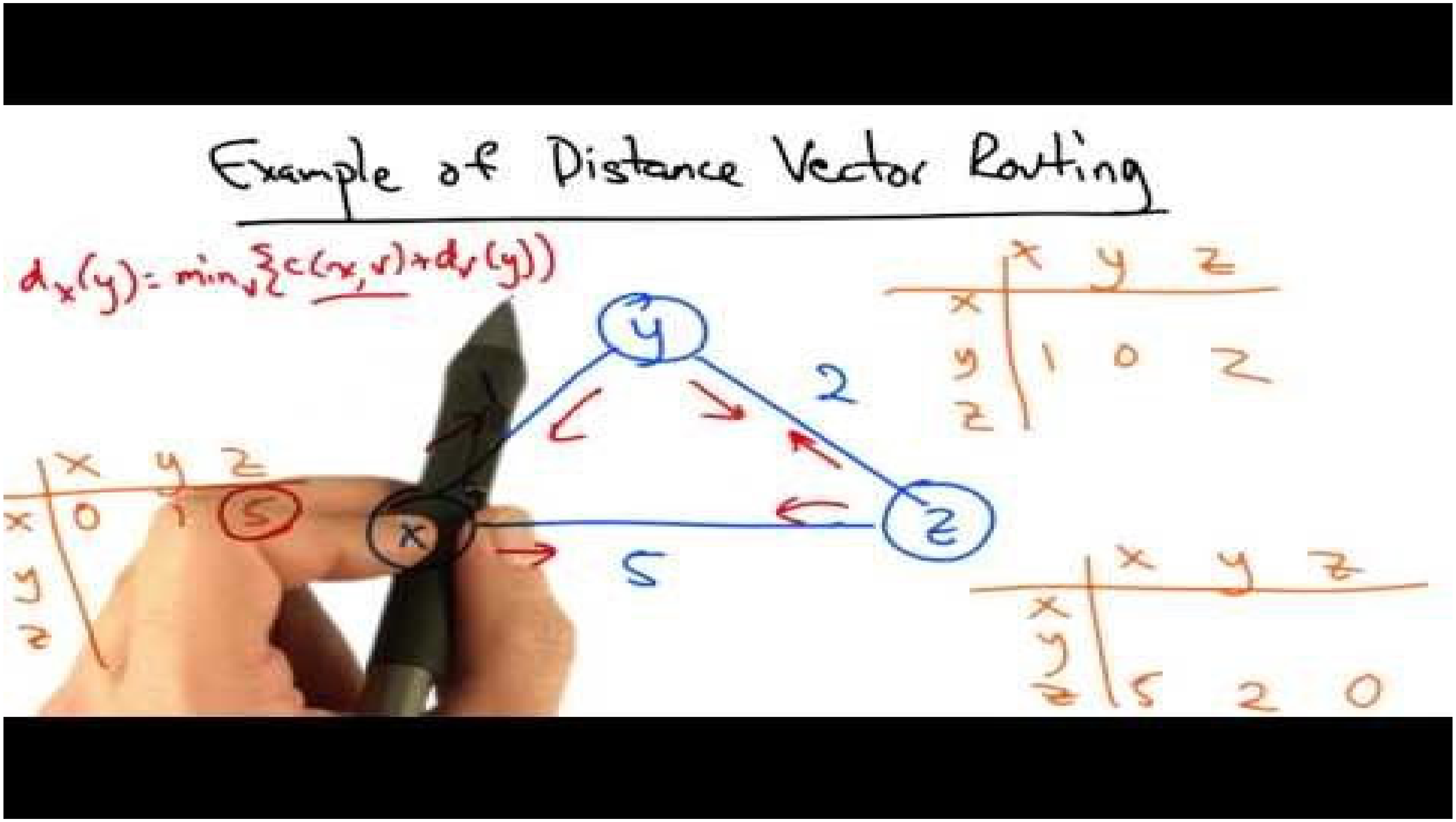
DV in f:

$D_f(a) = \infty$
$D_f(b) = \infty$
$D_f(c) = \infty$
$D_f(d) = \infty$
$D_f(e) = 1$
$D_f(f) = 0$
$D_f(g) = \infty$
$D_f(h) = \infty$
$D_f(i) = 1$

$$D_x(y) \leftarrow \min_v\{c_{x,v} + D_v(y)\}$$

for each node *y* ∊ *N*, at node x, achieved via neighboring nodes v

Distance Vector Routing


Example of Distance Vector Routing
$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$
x y z
x 0 1 5
y
z
x y z
x
y 1 0 2
z
x y z
x
y
z 5 2 0
1
2
5

https://www.youtube.com/watch?v=x9WIQbaVPzY&t=7s

Distance Vector Routing

- Goals of a distance vector protocol and measures to achieve the goals
  - Learning routing information
    - Directly connected subnets are already known by the router; those routes are advertised to neighboring routers
    - Routing updates are broadcasted to all neighboring routers
    - Routing updates from neighboring routers are listened so that new routes can be learned
    - A route learned from a neighboring router is assumed to be through that particular router
    - A metric describing each route is in the update. The metric describes the quality of the route
    - Routing update includes at a minimum, subnet and metric information
    - Only info from immediate neighbor routing tables is used for route updating

Distance Vector Routing

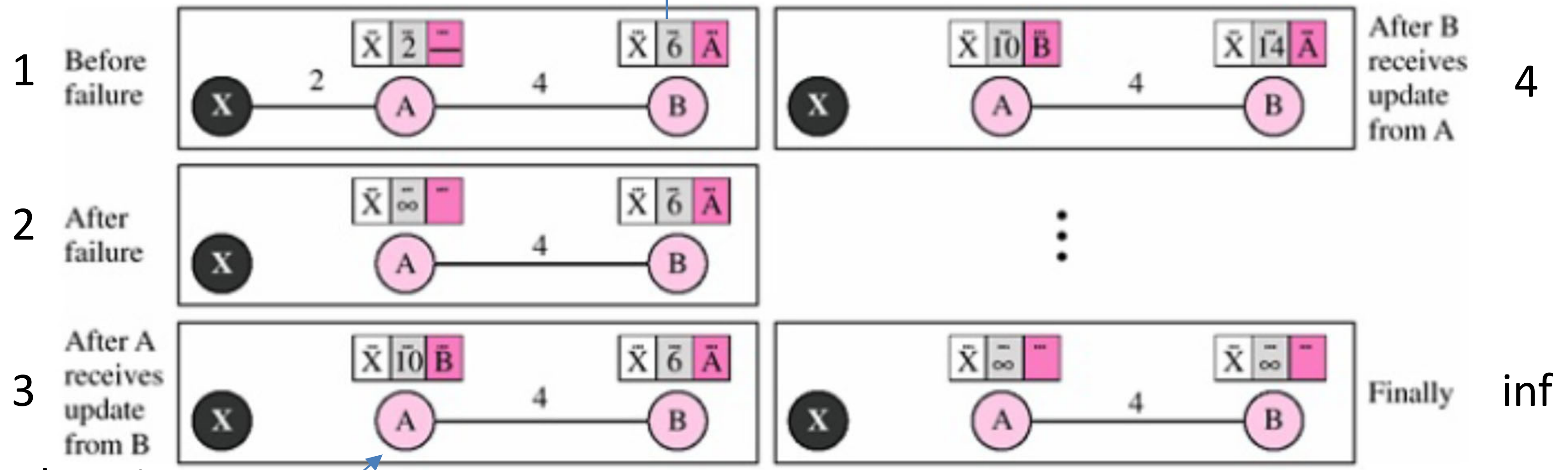
- Noticing failed routes
  - Periodic updates are expected to be received from neighboring routers at specific intervals
  - Triggered updates are sent any time there is a change in a nodes routing table
- Adding the current best route after one has failed
  - A failed route is advertised for a time, with a metric that implies that the network is "infinite" distance
- Low overhead and computational complexity as each router does not need to know the whole network information
- Preventing routing loops
  - Split horizon strategy

Routing Loop Example

Figure 22.17 Two-node instability

B thought can reach X via A so AB update based Table from A

Routing info from A



A thought can reach X via B so A update based Table from B

Bad news travels slow

Split Horizon: instead of flooding the table to all nodes, a node send only table content that did not come from that node.

In step 2, B is not sending that routing portion of the table to A (as B can reach X via A)

A then informs B that X is not reachable from A in Step 3

Back to Previous Example: Split-Horizon

The optimum route in yellow is NOT sent to e by b as the optimum route is via e → b to {e, f,h,d}

t=1

- b receives DVs from a, c, e, computes:

DV in a:

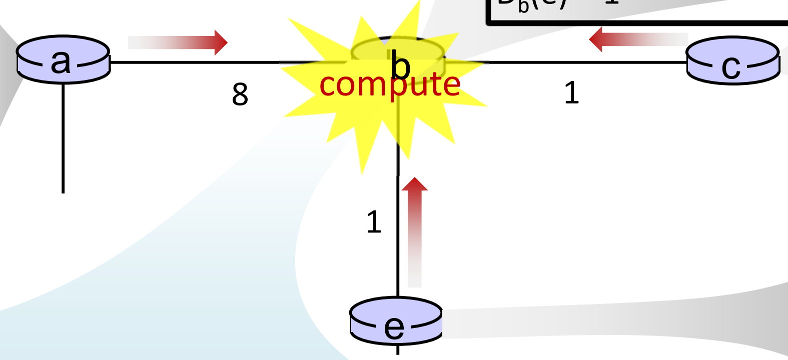
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = \infty$
$D_b(d) = \infty$	$D_b(i) = \infty$
$D_b(e) = 1$	

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$



DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

$D_b(a) = \min\{c_{b,a}+D_a(a),\ c_{b,c}+D_c(a),\ c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c),\ c_{b,c}+D_c(c),\ c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d),\ c_{b,c}+D_c(d),\ c_{b,e}+D_e(d)\} = \min\{9,\infty,2\} = 2$

$D_b(e) = \min\{c_{b,a}+D_a(e),\ c_{b,c}+D_c(e),\ c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

$D_b(f) = \min\{c_{b,a}+D_a(f),\ c_{b,c}+D_c(f),\ c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g),\ c_{b,c}+D_c(g),\ c_{b,e}+D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty$

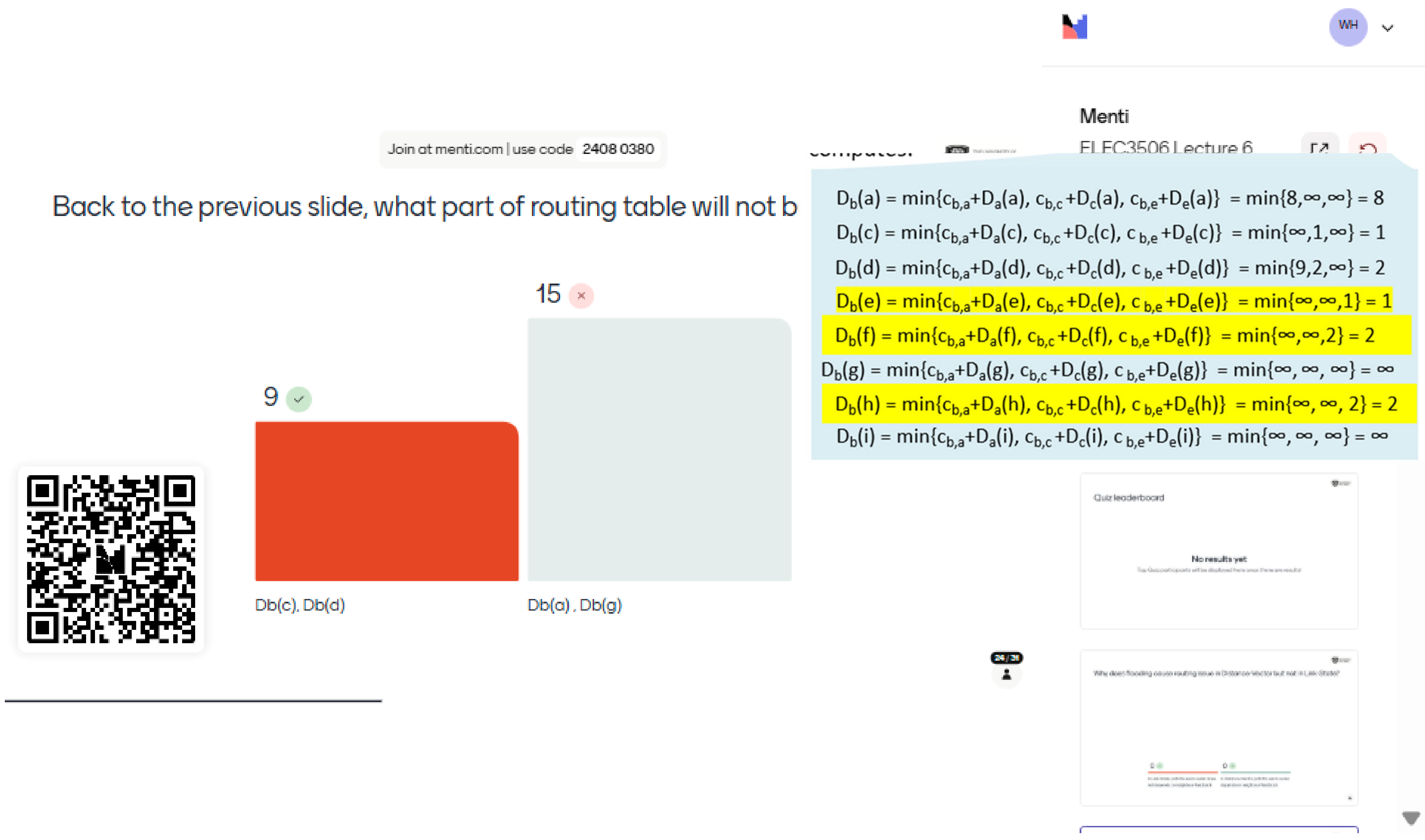
$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2$

$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) =2$
$D_b(b) = 0$	$D_b(g) = \infty$
$D_b(c) = 1$	$D_b(h) = 2$
$D_b(d) = 2$	$D_b(i) = \infty$
$D_b(e) = 1$	

Knowledge Test



Join at menti.com | use code 2408 0380

Back to the previous slide, what part of routing table will not b

$D_b(a) = \min\{c_{b,a}+D_a(a),\ c_{b,c}+D_c(a),\ c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c),\ c_{b,c}+D_c(c),\ c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d),\ c_{b,c}+D_c(d),\ c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$

$D_b(e) = \min\{c_{b,a}+D_a(e),\ c_{b,c}+D_c(e),\ c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

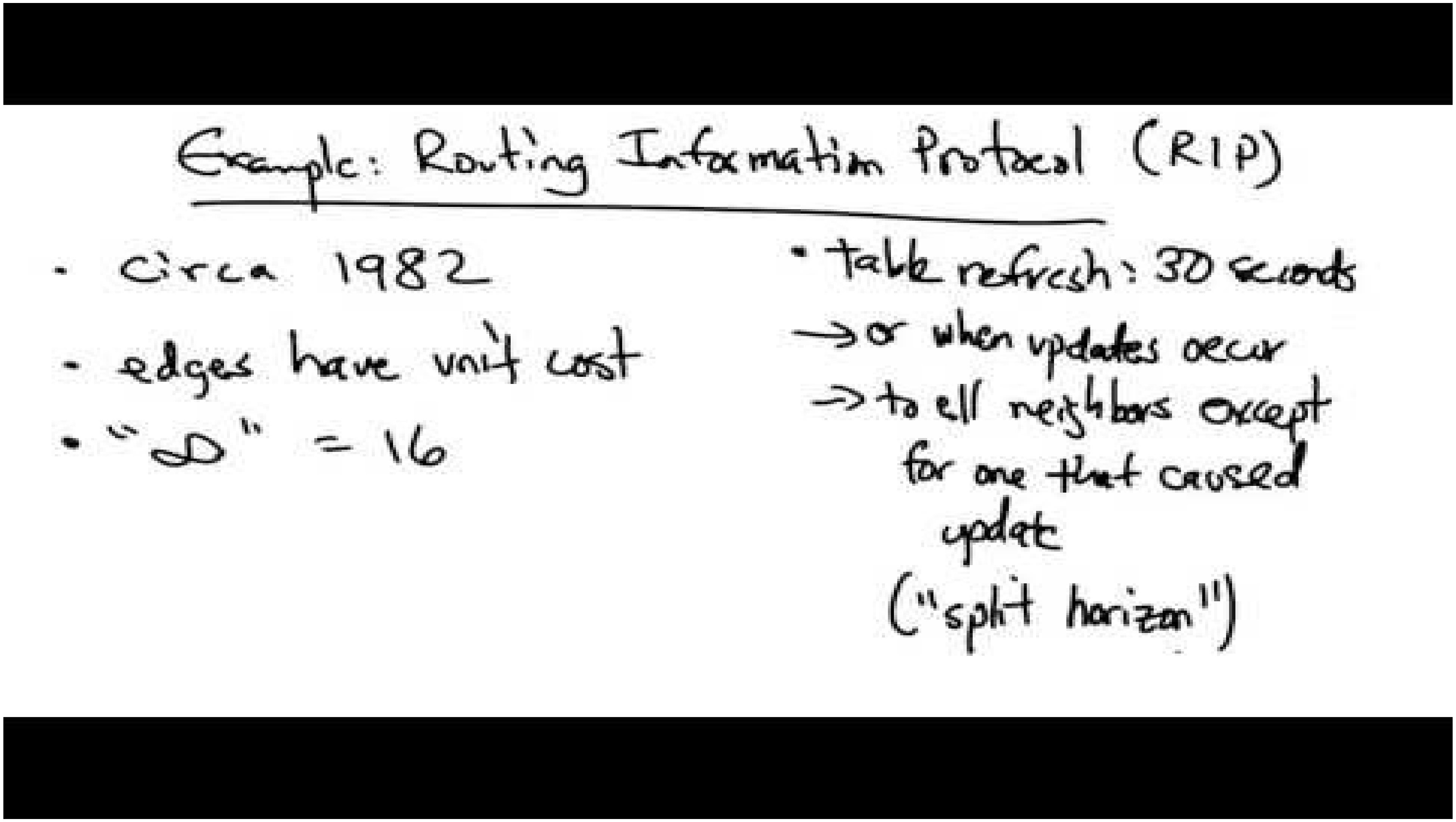
$D_b(f) = \min\{c_{b,a}+D_a(f),\ c_{b,c}+D_c(f),\ c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g),\ c_{b,c}+D_c(g),\ c_{b,e}+D_e(g)\} = \min\{\infty,\infty,\infty\} = \infty$

$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty,\infty,2\} = 2$

$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$

Intra-AS: Routing Information Protocol (RIP)



Example: Routing Information Protocol (RIP)

- circa 1982
- edges have unit cost
- "∞" = 16

- table refresh: 30 seconds
  - → or when updates occur
  - → to all neighbors except for one that caused update ("split horizon")

https://www.youtube.com/watch?v=KqkenpIsJzk

Intra-AS: RIP

- RIP sends routing-update messages at regular intervals when the network topology changes.
- Send to all except the one that caused the update (recall Split Horizon)
- When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route, based on the distance vector algorithm
- RIP routers maintain only the best route to a destination.
- After updating its routing table, the router immediately begins transmitting routing updates to inform other network routers of the change.
- These updates are sent independently of RIP routers' regularly scheduled updates.

Intra-AS: RIP Routing Metric and Stability

- RIP uses a single routing metric (hop count) to measure the distance between the source and a destination network.
- RIP enforces a limit on the number of hops allowed in a path from the source to a destination.
- The maximum number of hops in a path is 15 (infinity defined at 16).
  - If the number of hops becomes 16 after it is increased by 1, the network destination is considered unreachable
- RIP also implements mechanisms to prevent incorrect routing information from being propagated (e.g., defining infinity, split horizon strategy)

RIP

At the beginning, the routing table in B is empty

Router A sends a message to B and informs B that it can forward packets to subnet 162.11.5.0 and 162.11.9.0 with 1 hop.

Router B updates its own routing table: if the destination is 162.11.5.0 or 162.11.9.0, the next hop should be 162.11.8.1 and the physical interface is S0 (since it receive the routing update from S0)

Router A Advertising Directly Connected Routes

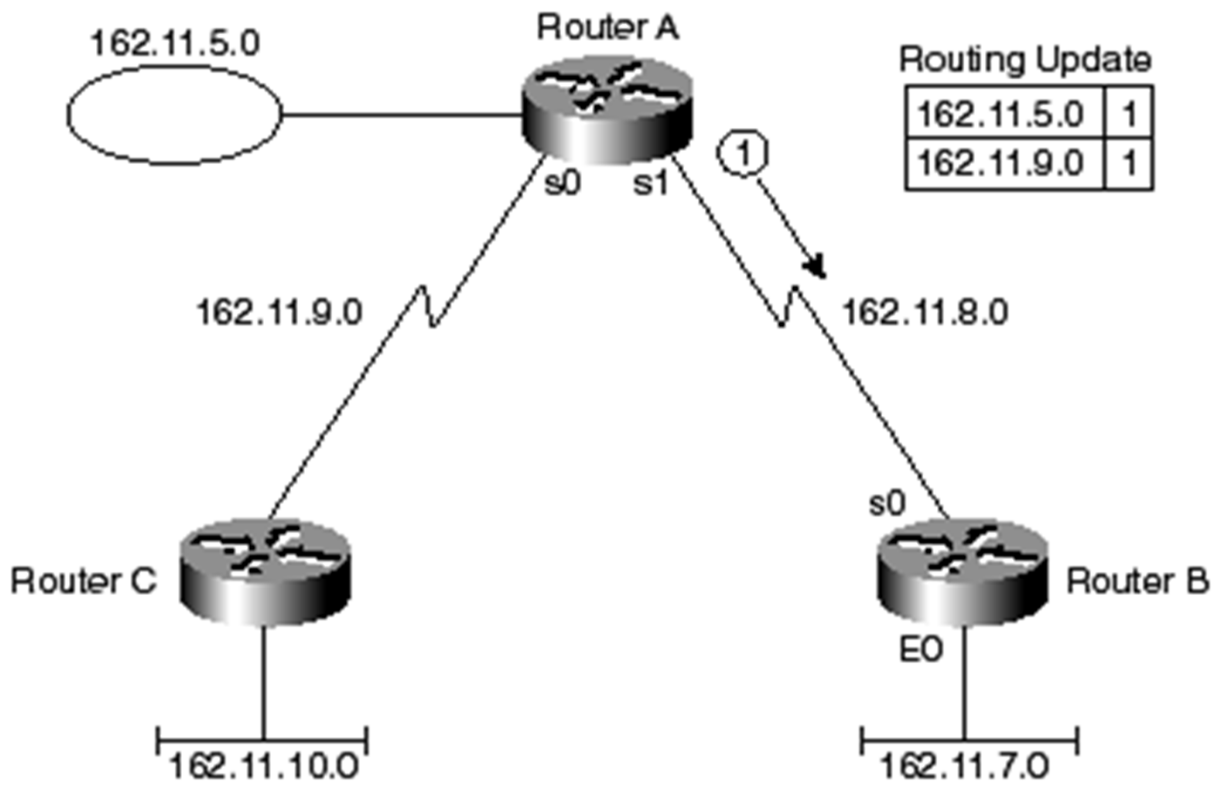


Table 6-3 shows the resulting routing table on Router B.

Router B Routing Table, After Receiving Update in Figure 6-2

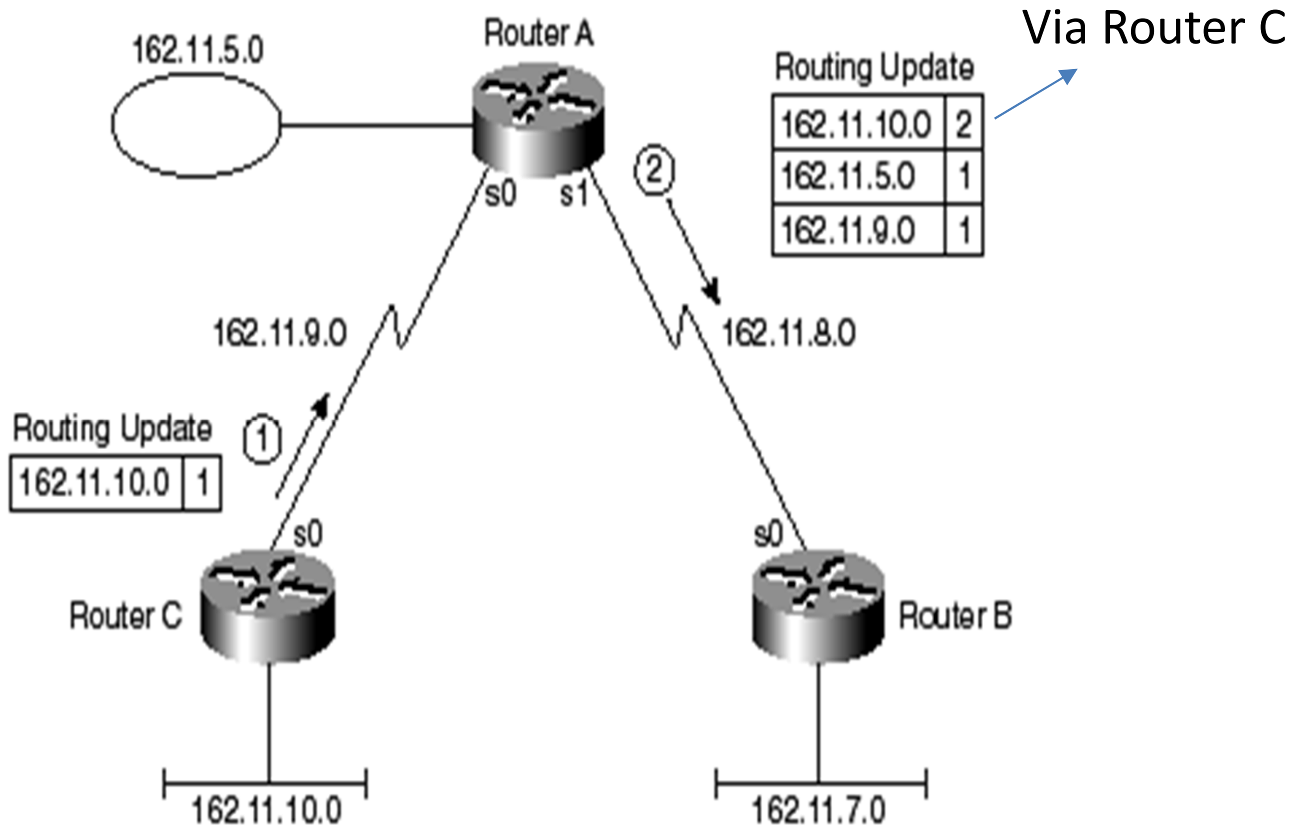
Group (Mask Is 255.255.255.0)	Outgoing Interface	Next Router
162.11.5.0	S0	162.11.8.1
162.11.7.0	E0	
162.11.8.0	S0	
162.11.9.0	S0	162.11.8.1

Router C sends a routing update to router A and informs it can forward packet to subnet 162.11.10.0 with one hop

Router A receives this update and combines it with its own routing table.

Router A sends the combined table to B again. Compared to the first update, now there is a new entry (162.11.10.0). Note that before sending the routing table, A need to increase the number of hops for the entry that it receives from other routers by 1 (the number of hops for 162.11.10.0 is increased from 1 to 2).

Router A Advertising Routes Learned from Router C



Router B Routing Table, After Receiving Update in Figure 6-3

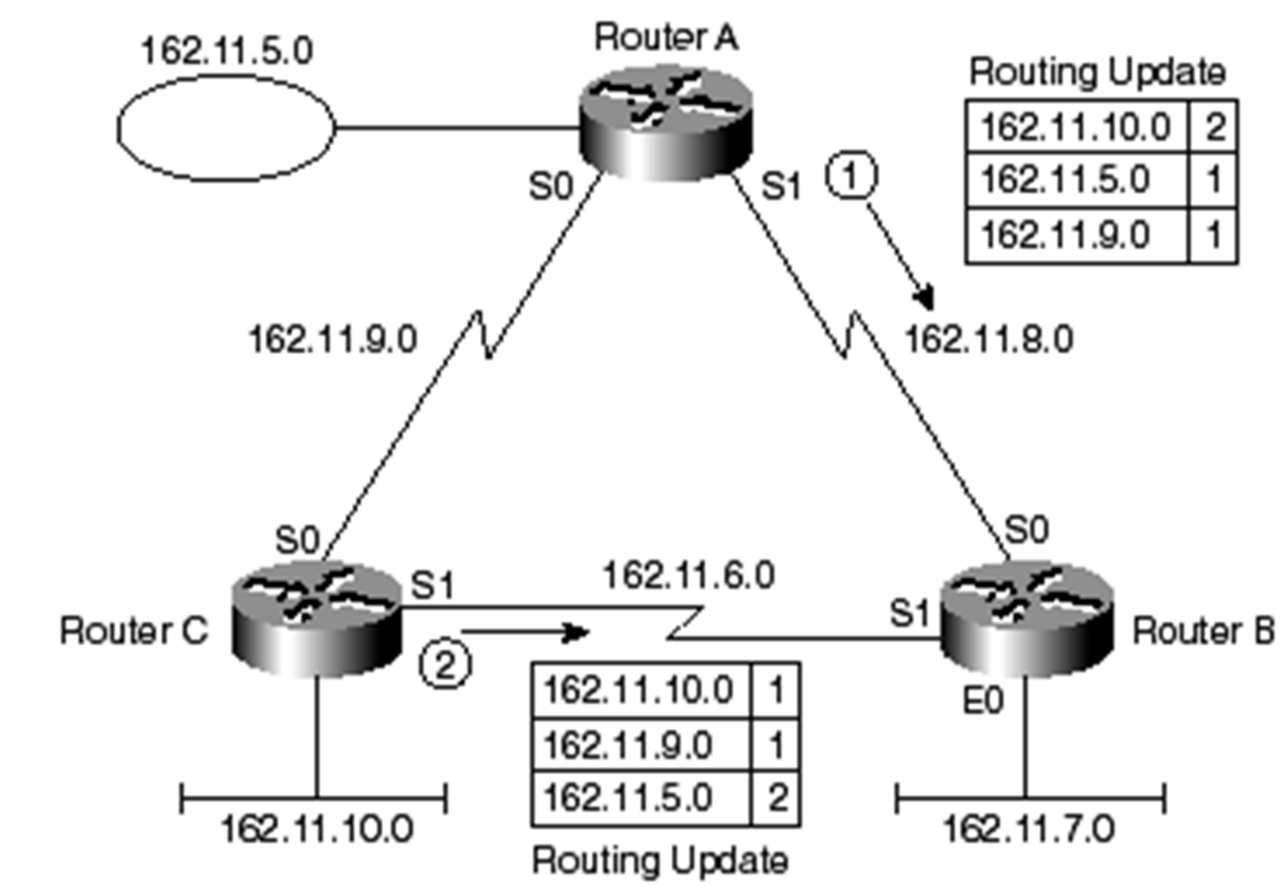
Group	Outgoing Interface	Next Router
162.11.5.0	S0	162.11.8.1
162.11.7.0	E0	
162.11.8.0	S0	
162.11.9.0	S0	162.11.8.1
162.11.10.0	S0	162.11.8.1

When multiple routes to the same subnet exist

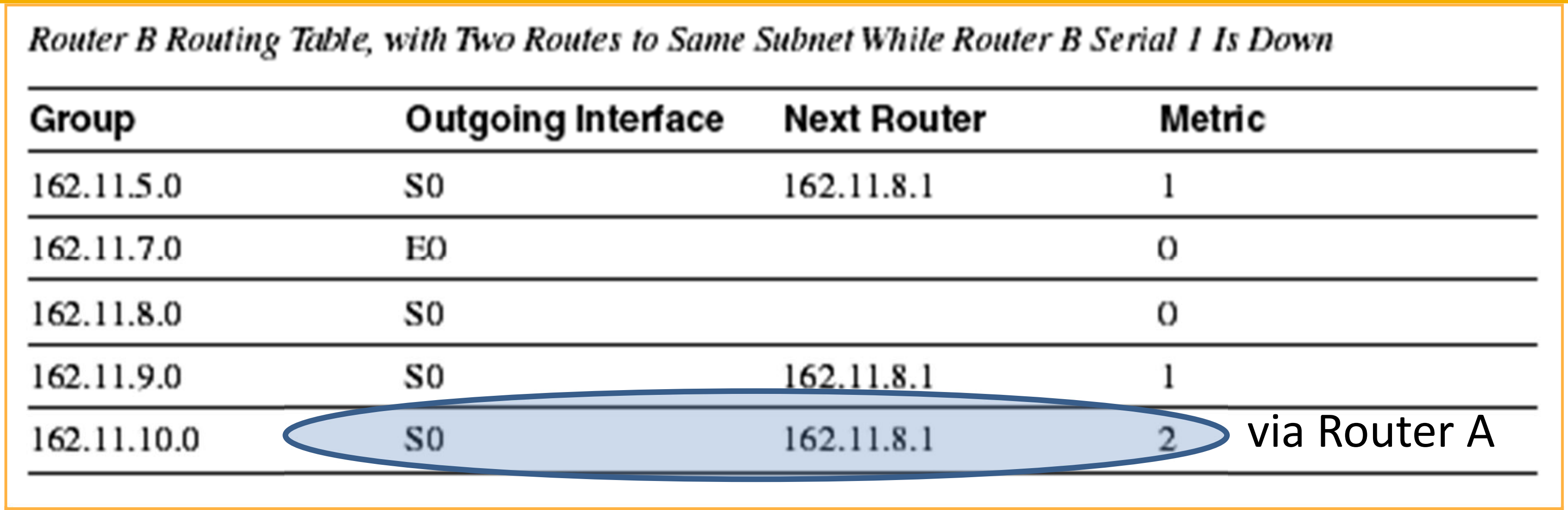
If a new link Router C(S1)- Router B(S1) is established, there are two paths from Router B to subnet 162.11.10.0
(1)S0->Router A->Router C (2 hops)
(2)S1->Router C (1 hops)

According to the update rules of RIP, only the path with the minimum number of hops will be remained. Therefore, the outgoing interface for 162.11.10.0 is changed from S0 to S1

Routers A and C Advertising to Router B



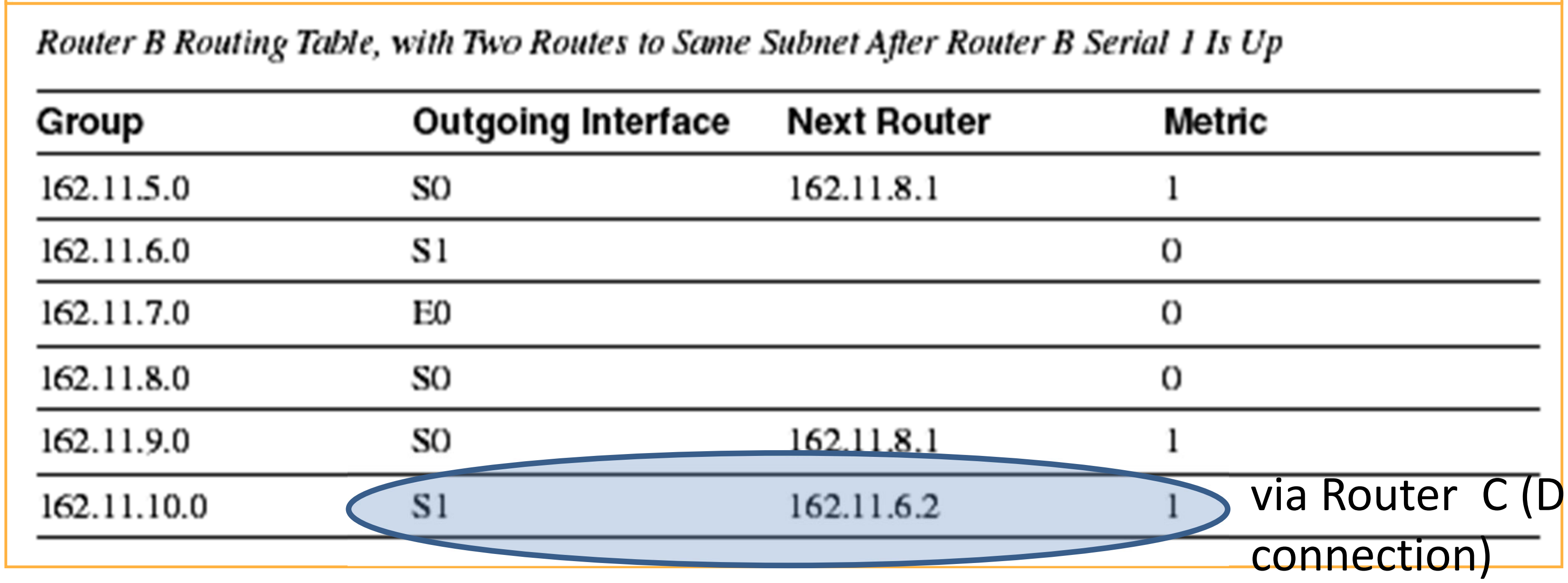
Routing table comparison when S1 link of router B is down and up



Router B Routing Table, with Two Routes to Same Subnet While Router B Serial 1 Is Down

Group	Outgoing Interface	Next Router	Metric
162.11.5.0	S0	162.11.8.1	1
162.11.7.0	E0		0
162.11.8.0	S0		0
162.11.9.0	S0	162.11.8.1	1
162.11.10.0	S0	162.11.8.1	2

via Router A



Router B Routing Table, with Two Routes to Same Subnet After Router B Serial 1 Is Up

Group	Outgoing Interface	Next Router	Metric
162.11.5.0	S0	162.11.8.1	1
162.11.6.0	S1		0
162.11.7.0	E0		0
162.11.8.0	S0		0
162.11.9.0	S0	162.11.8.1	1
162.11.10.0	S1	162.11.6.2	1

via Router C (Direct connection)

Link State Routing

- Distance Vector Algorithm had two main problems:
  - Metric was just a hop count and line bandwidth was not considered.
  - Exchanging of routing tables was an overhead.
- In Link state algorithm each router must:
  - Discover their neighbors and learn their network addresses
  - Measure the delay or cost to each of the neighbors (Traffic levels, state of the link, etc considered)
  - Construct a packet (Link State Packet – LSP) with this new information containing: node id, list of links, sequence no, age
  - Floods this packet to all other routers
  - Formation of the shortest path tree to each router
  - Compute the routing table based on the shortest path tree for each router

Dijkstra's link-state routing algorithm

- centralized: network topology, direct link costs $c_{x,y}$ known to *all* nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after *k* iterations, know least cost path to *k* destinations

notation

- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v): current* estimate of cost of least-cost-path from source to destination *v*
- *p(v):* predecessor node along path from source to *v*
- *N':* set of nodes whose least-cost-path *definitively* known

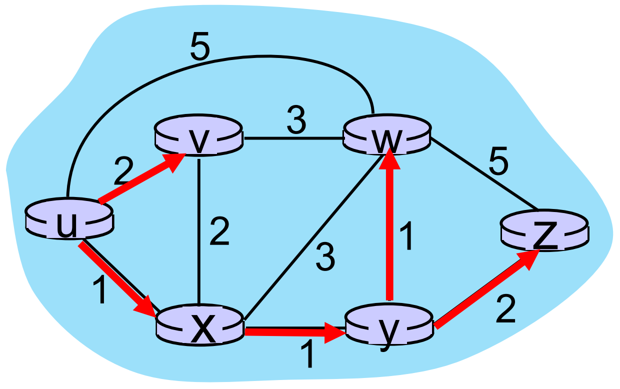
Dijkstra’s link-state routing algorithm

```
Initialization:
  $N' = \{u\}$                        /* compute least cost path from u to all other nodes */
  for all nodes $v$
    if $v$ adjacent to $u$            /* $u$ initially knows direct-path-cost only to  direct neighbors    */
        then $D(v) = c_{u,v}$         /* but may not be minimum cost!                          */
    else $D(v) = \infty$

Loop
   find $w$ not in $N'$ such that $D(w)$ is a minimum
   add $w$ to $N'$
   update $D(v)$ for all $v$ adjacent to $w$ and not in $N'$ :
        $D(v) = \min ( D(v),  D(w) + c_{w,v} )$
   /* new least-path-cost to $v$ is either old least-cost-path to $v$ or known
   least-cost-path to $w$ plus direct-cost from $w$ to $v$ */
until all nodes in $N'$
```

Dijkstra’s algorithm: an example

		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

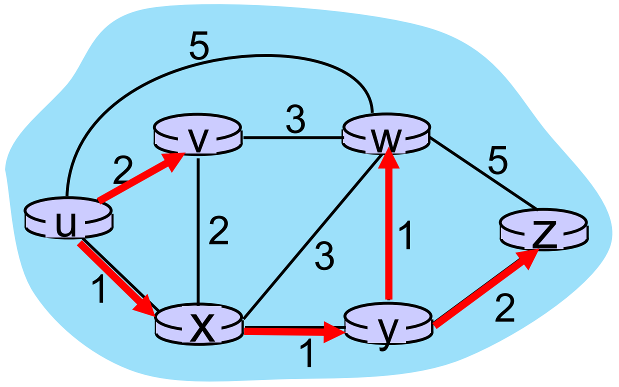


Initialization (step 0): For all a: if *a* adjacent to then $D(a) = c_{u,a}$

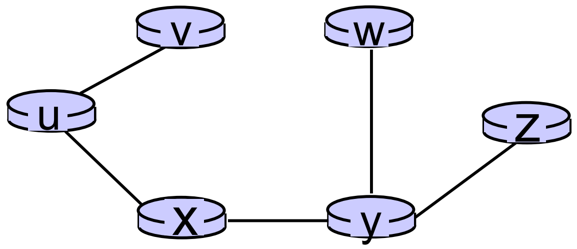
```
find a not in N′ such that D(a) is a minimum
add a to N′
update D(b) for all b adjacent to a and not in N′ :
    $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

Link costs of all nodes $c_{a,b}$ is needed at u

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

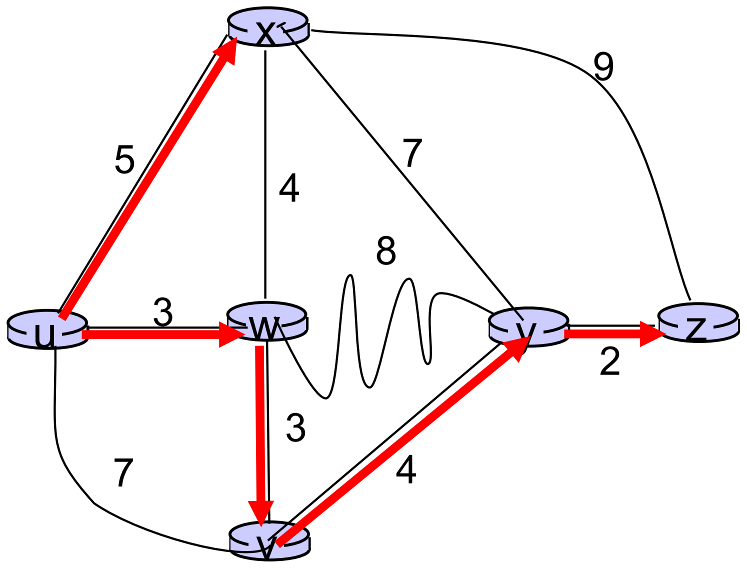
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

Dijkstra's algorithm: another example

Step	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	7,u	3,u	5,u	∞	∞
1						
2			?			
3						
4						
5						



resulting forwarding table in u?

resulting least-cost-path tree from u?

Dijkstra’s algorithm: discussion

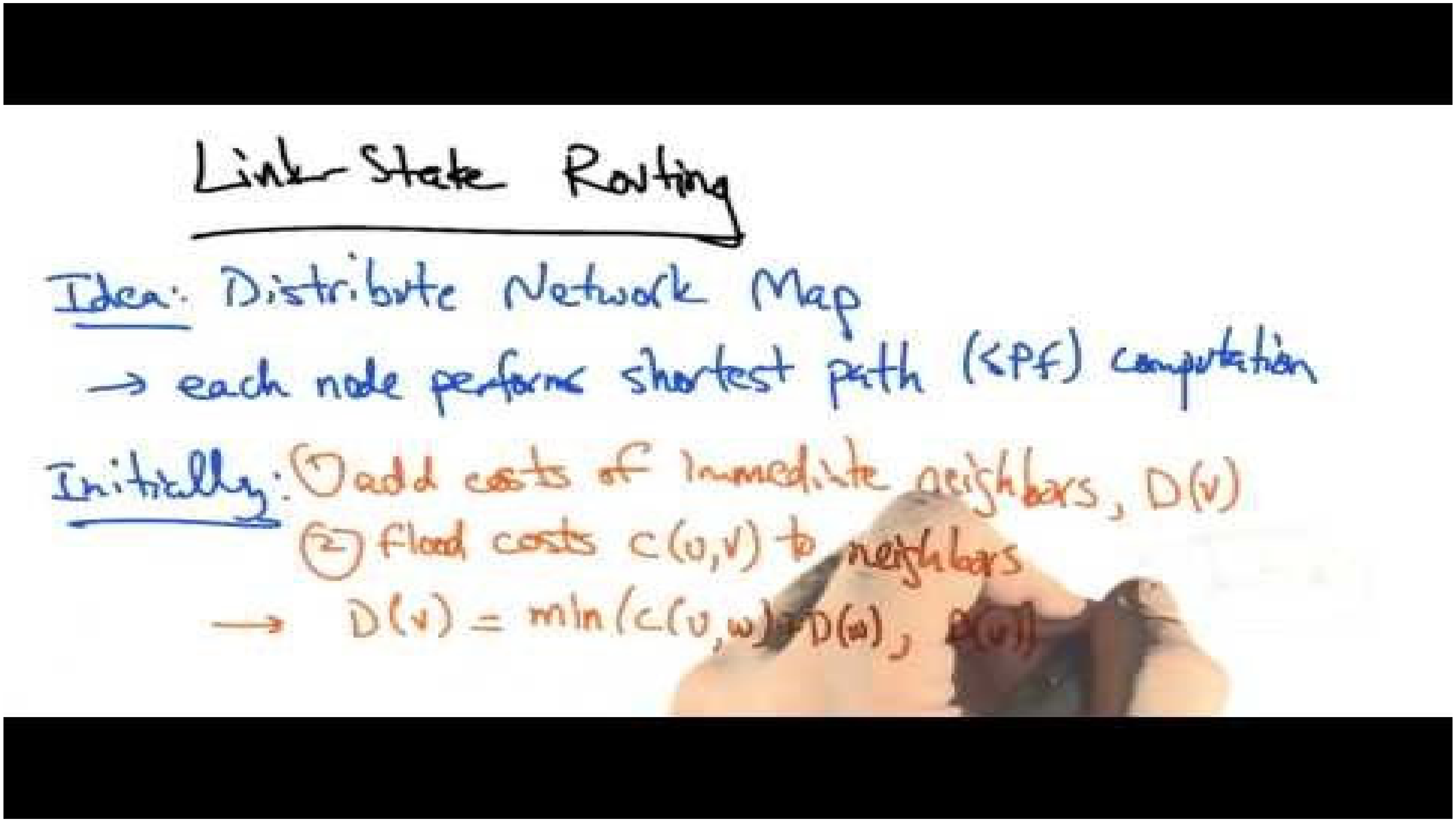
algorithm complexity: *n* nodes

- each of *n* iteration: need to check all nodes, *w*, not in *N*
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

message complexity:

- each router must *broadcast* its link state information to other *n* routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router’s message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Link State Routing



Link-State Routing

Idea: Distribute Network Map

→ each node performs shortest path (SPF) computation

Initially: ① add costs of immediate neighbors, $D(v)$

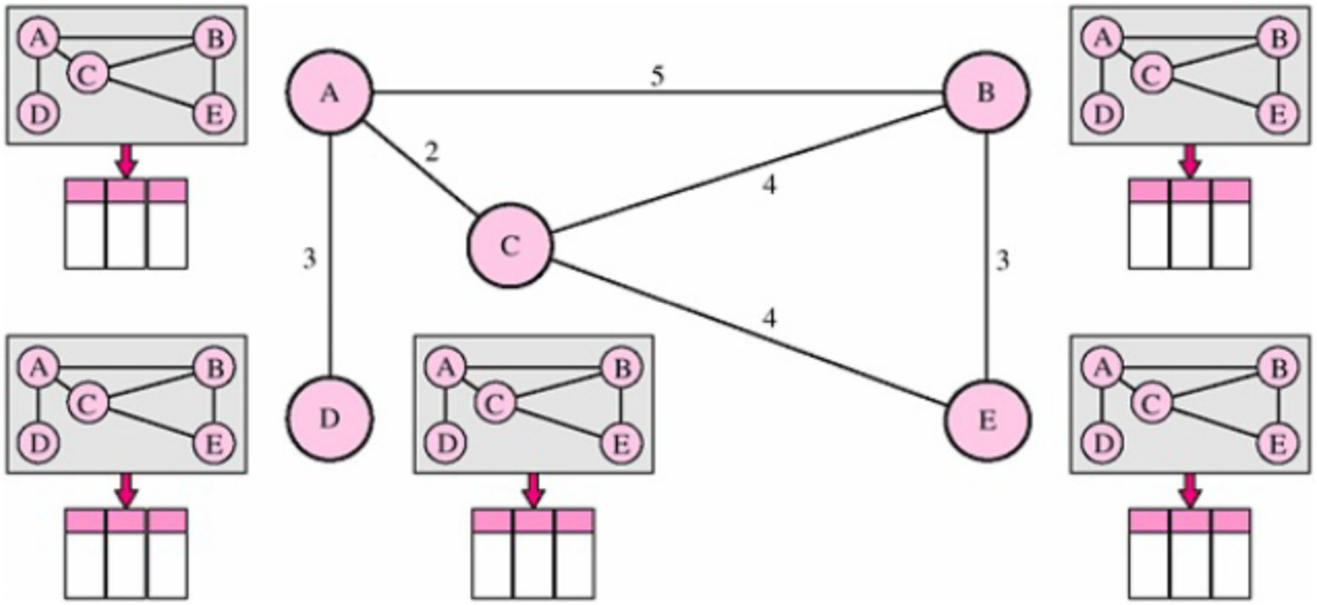
② Flood costs $c(u,v)$ to neighbors

→ $D(v) = \min(c(u,w) + D(w), D(v))$

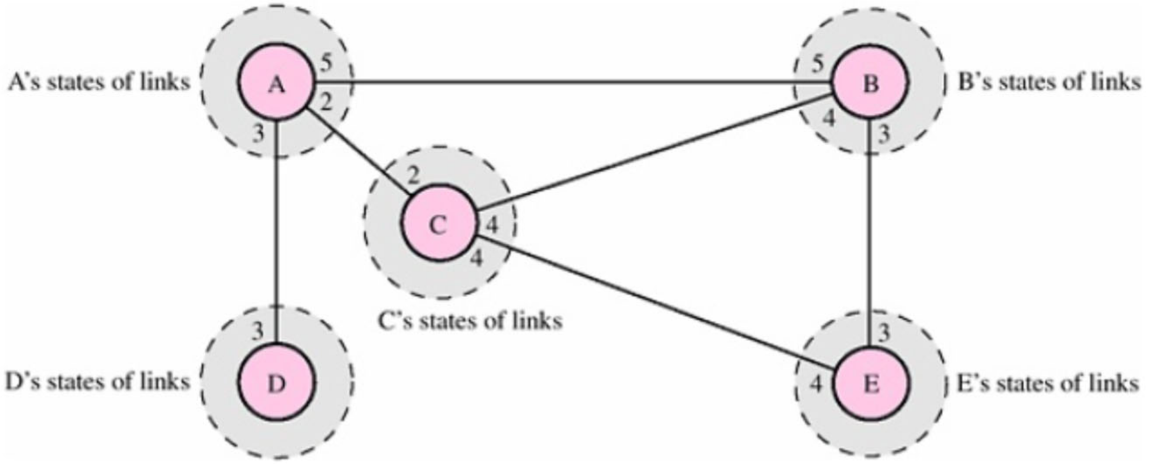
https://www.youtube.com/watch?v=sYDcnN6_clg&t=1s

Link State Routing

- Every node has the complete routing info (computed by using Djikstra's algo)



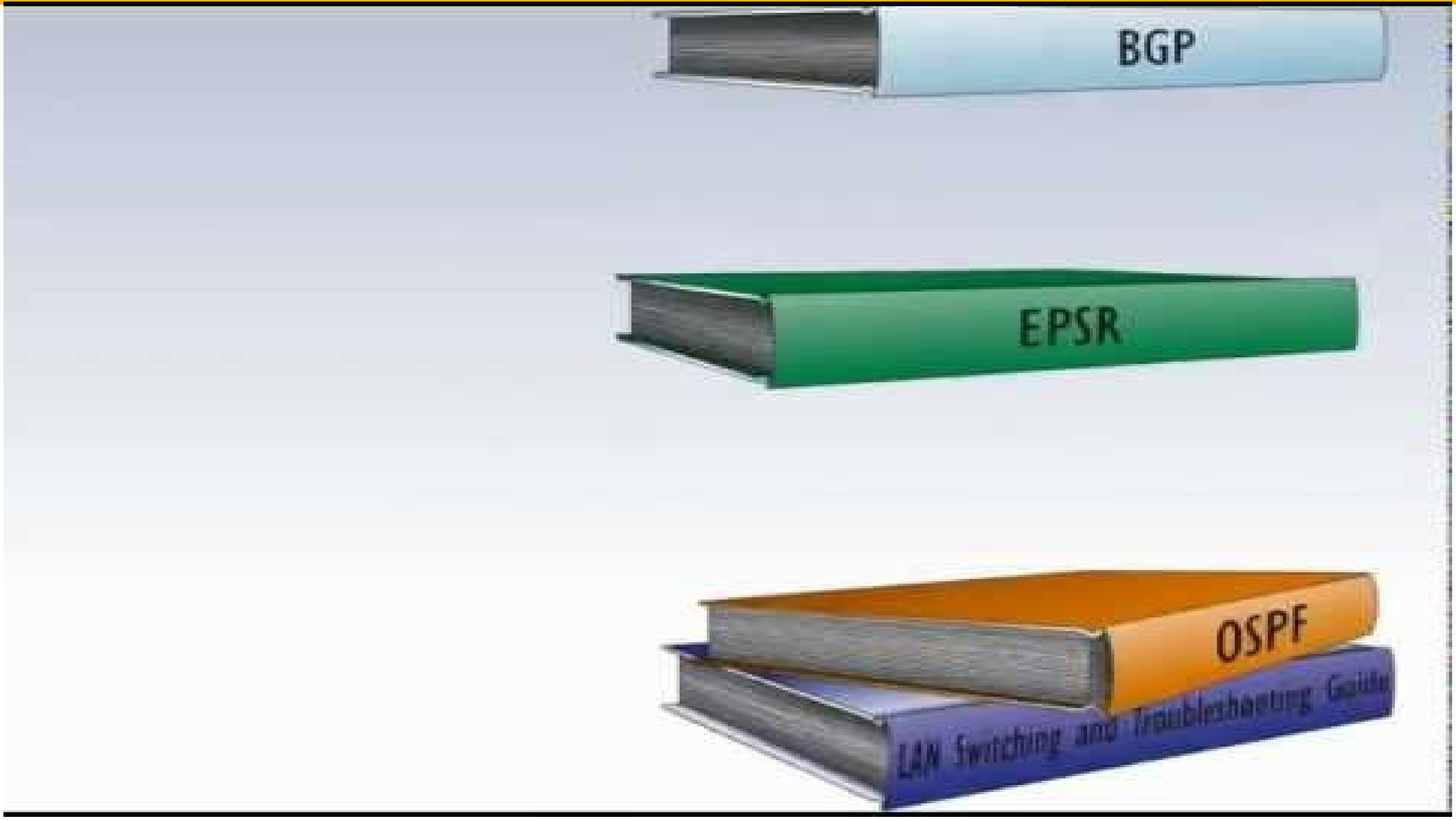
- Flooding local info



Intra-AS: Open Shortest Path First (OSPF)

- Each router sends only the portion of the routing table that describes the following state of its own links to the internetwork.
  - Information on attached interfaces
  - metrics used (min delay, max throughput, etc.)
  - other variables
- Each router builds a picture of the entire network in its routing table
- Next a Shortest Path First (SPF) algorithm (Dijkstra’s or Link State algorithm) is used to calculate the shortest path to each node

Open Shortest Path First (OSPF)



https://www.youtube.com/watch?v=_Iktarf8RXM&t=3s

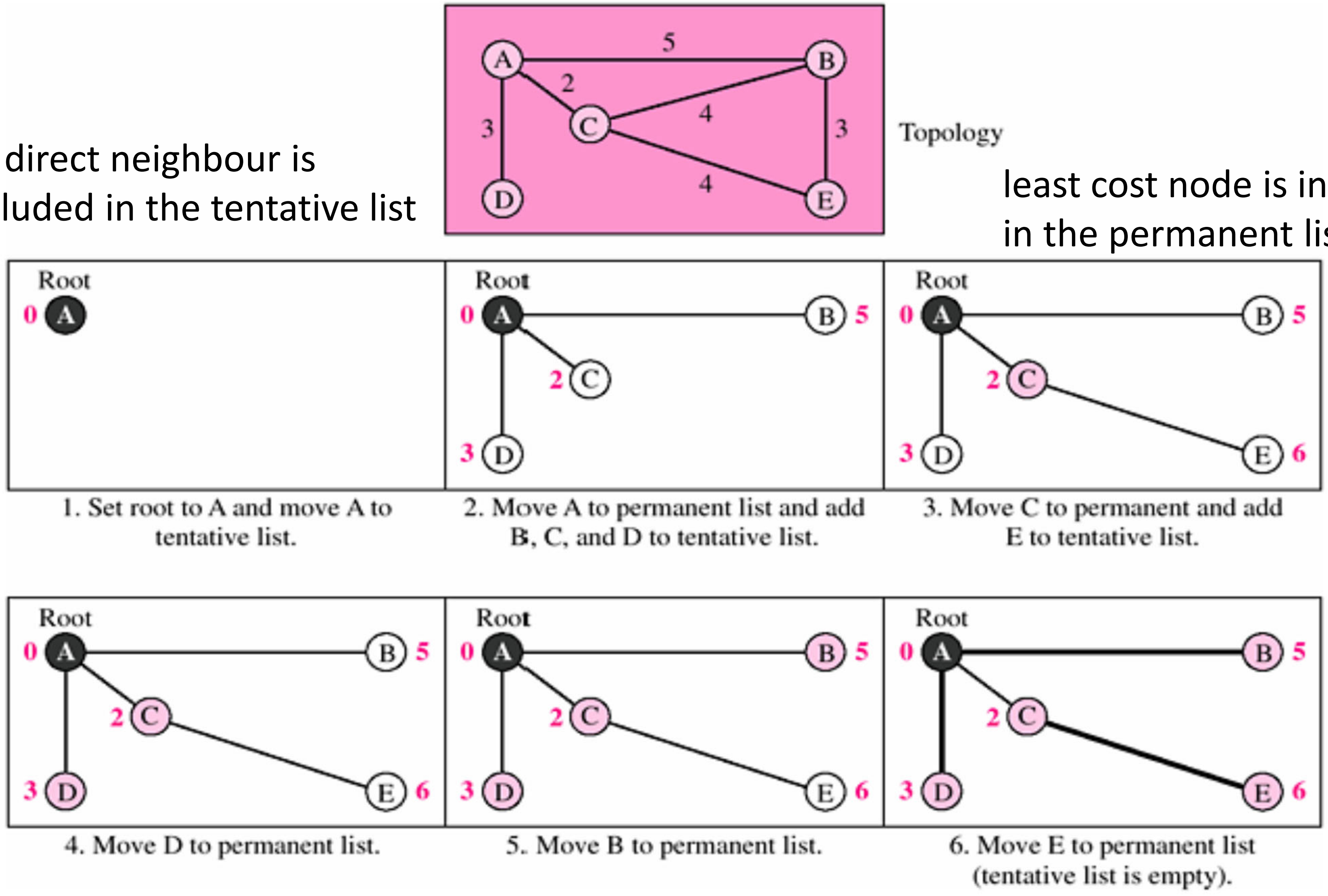
Shortest Path First Algorithm

- The OSPF protocol sends hello packets to its neighbors and receives their hello packets.
- Hello packets also act as a keep alive to let routers know that other routers are still functional.
- Failed routers can be detected quickly, and the network's topology can be altered appropriately.
- Each router calculates a shortest-path tree, with itself as the root.
- The shortest-path tree, in turn, yields a routing table.

Shortest Path First Algorithm

All direct neighbour is included in the tentative list

least cost node is included in the permanent list



1. Set root to A and move A to tentative list.

2. Move A to permanent list and add B, C, and D to tentative list.

3. Move C to permanent and add E to tentative list.

4. Move D to permanent list.

5. Move B to permanent list.

6. Move E to permanent list (tentative list is empty).

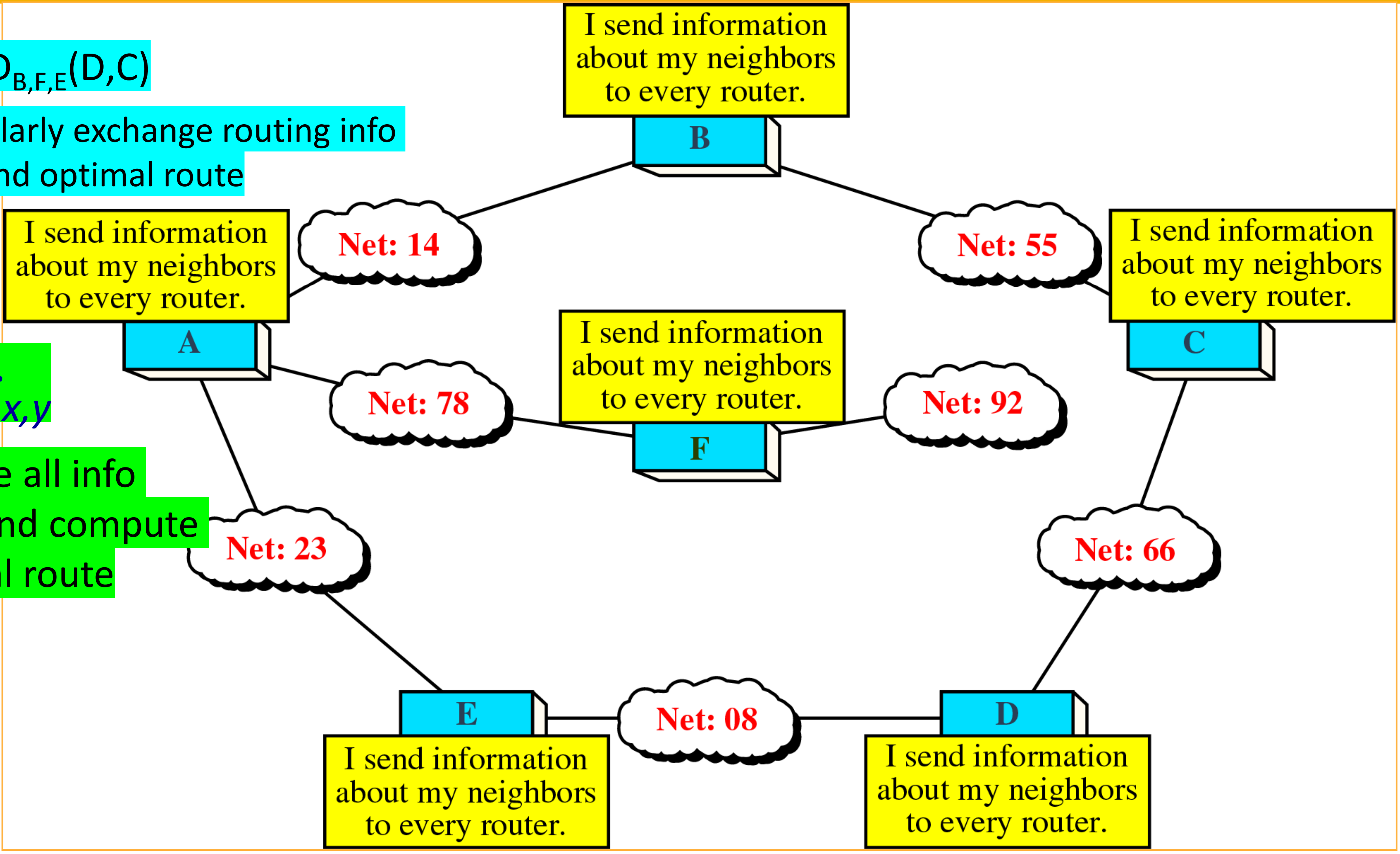
Link-State vs. Distance-Vector Routing

$D_{B,F,E}(D,C)$

regularly exchange routing info to find optimal route

$c_{x,y}$

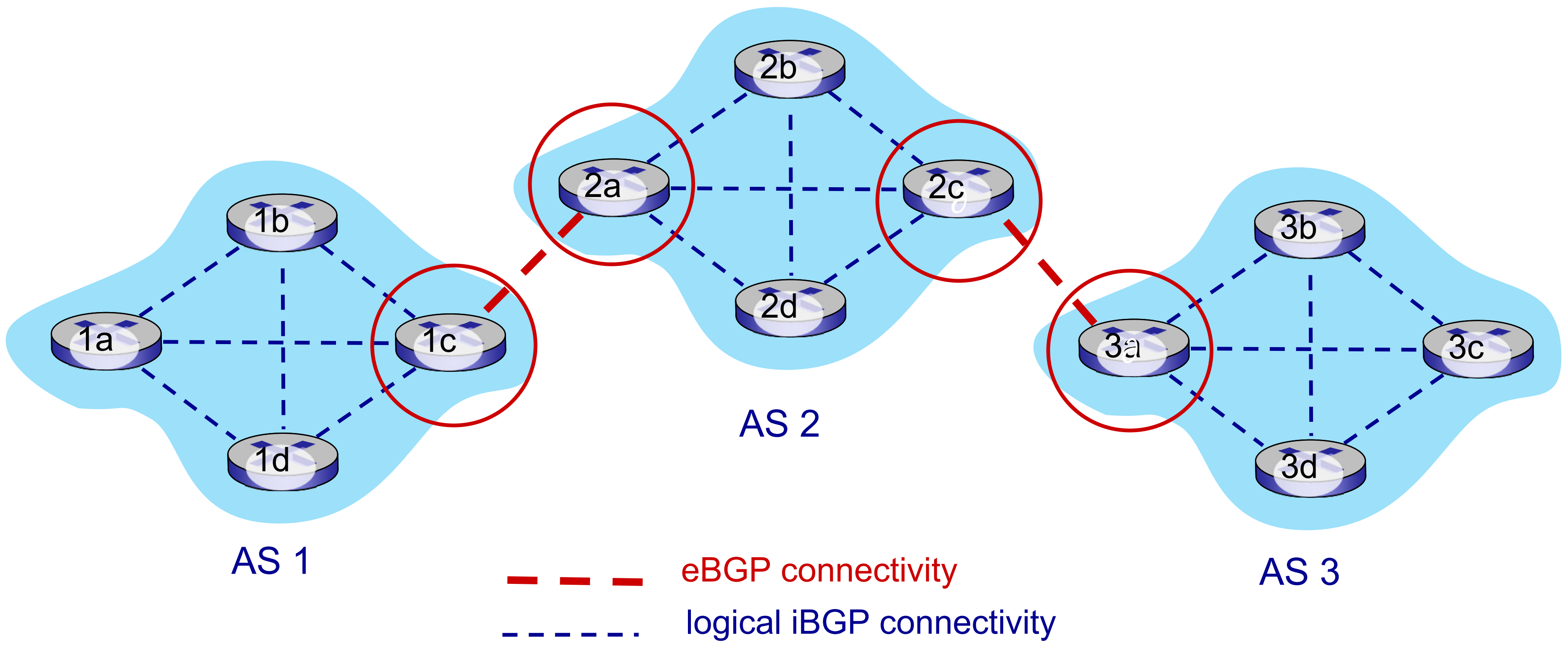
Receive all info once and compute optimal route

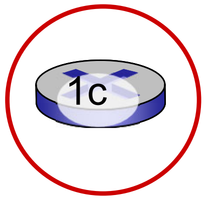

I send information about my neighbors to every router.
B
Net: 14
Net: 55
I send information about my neighbors to every router.
I send information about my neighbors to every router.
A
C
Net: 78
I send information about my neighbors to every router.
Net: 92
F
Net: 23
Net: 66
E
Net: 08
D
I send information about my neighbors to every router.
I send information about my neighbors to every router.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet: *"I am here, here is who I can reach, and how"*
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASes
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and *policy*

eBGP, iBGP connections



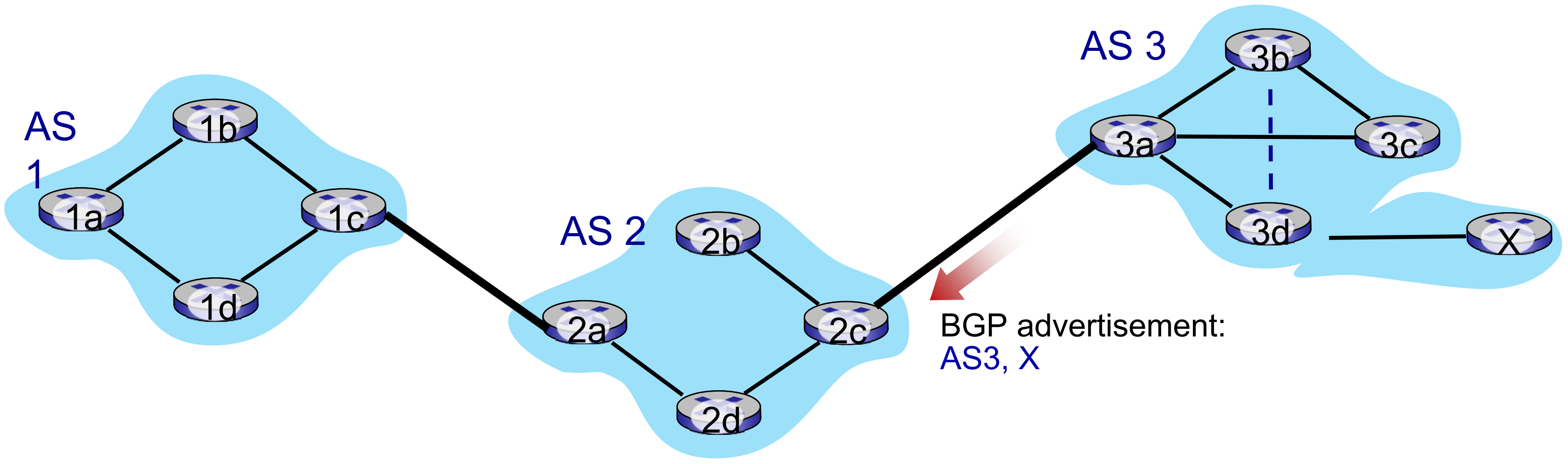


gateway routers run both eBGP and iBGP protocols

BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes

- when AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X



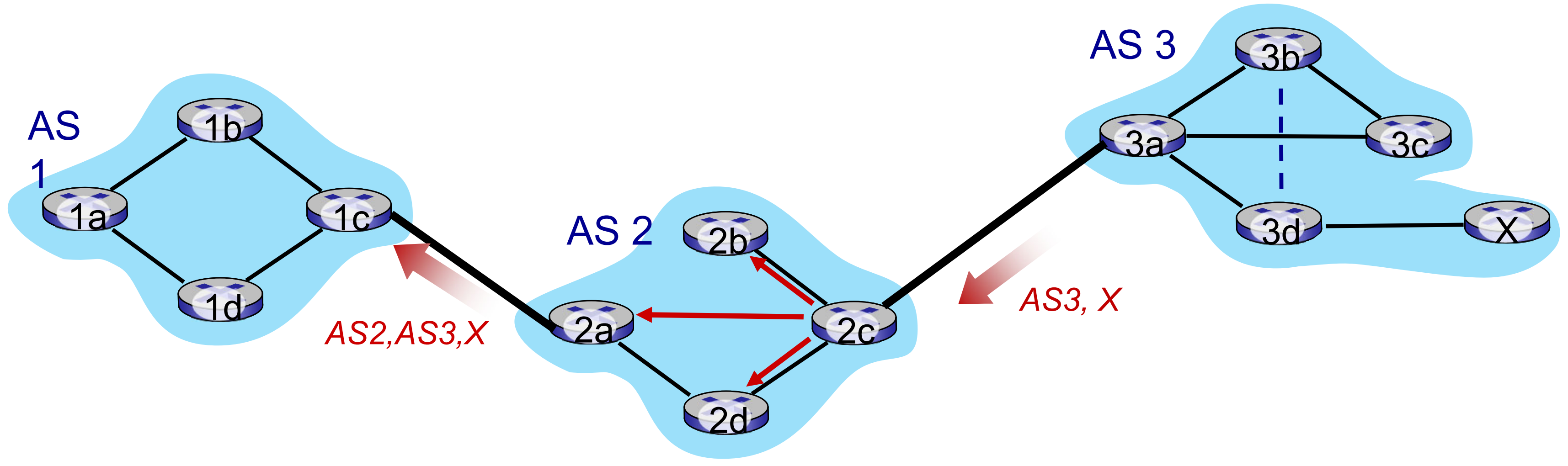
- AS3 has a peering agreement with AS2

Path attributes and BGP routes

- BGP advertised route: prefix + attributes
  - prefix: destination being advertised
  - two important attributes:
    - AS-PATH: list of ASes through which prefix advertisement has passed
    - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- policy-based routing:
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

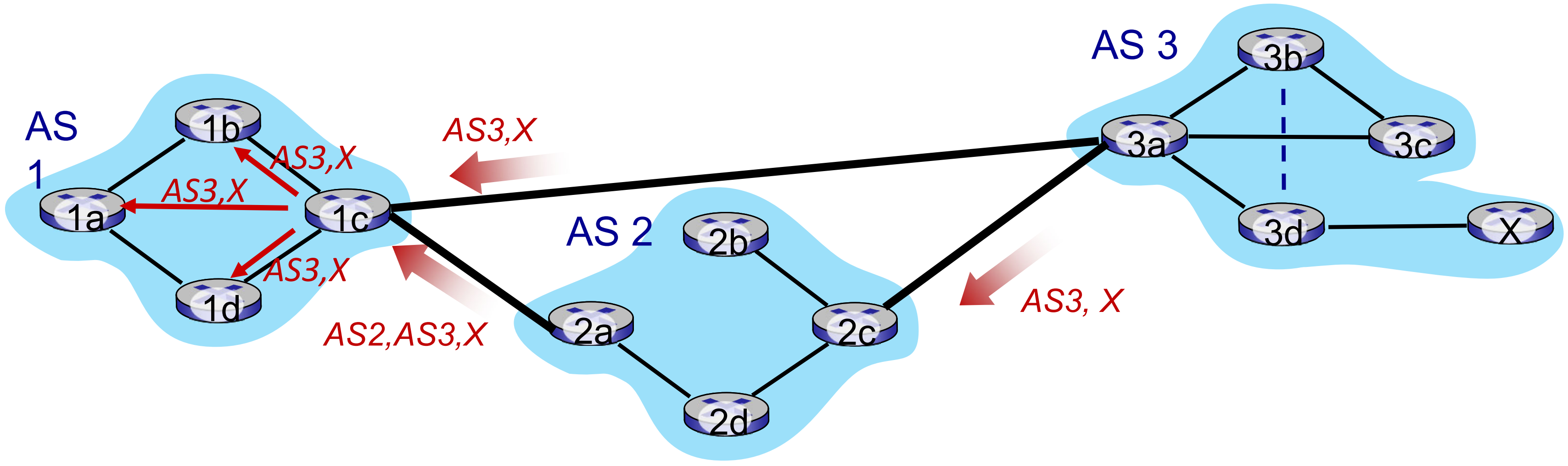
THE UNIVERSITY OF SYDNEY

BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP path advertisement (more)



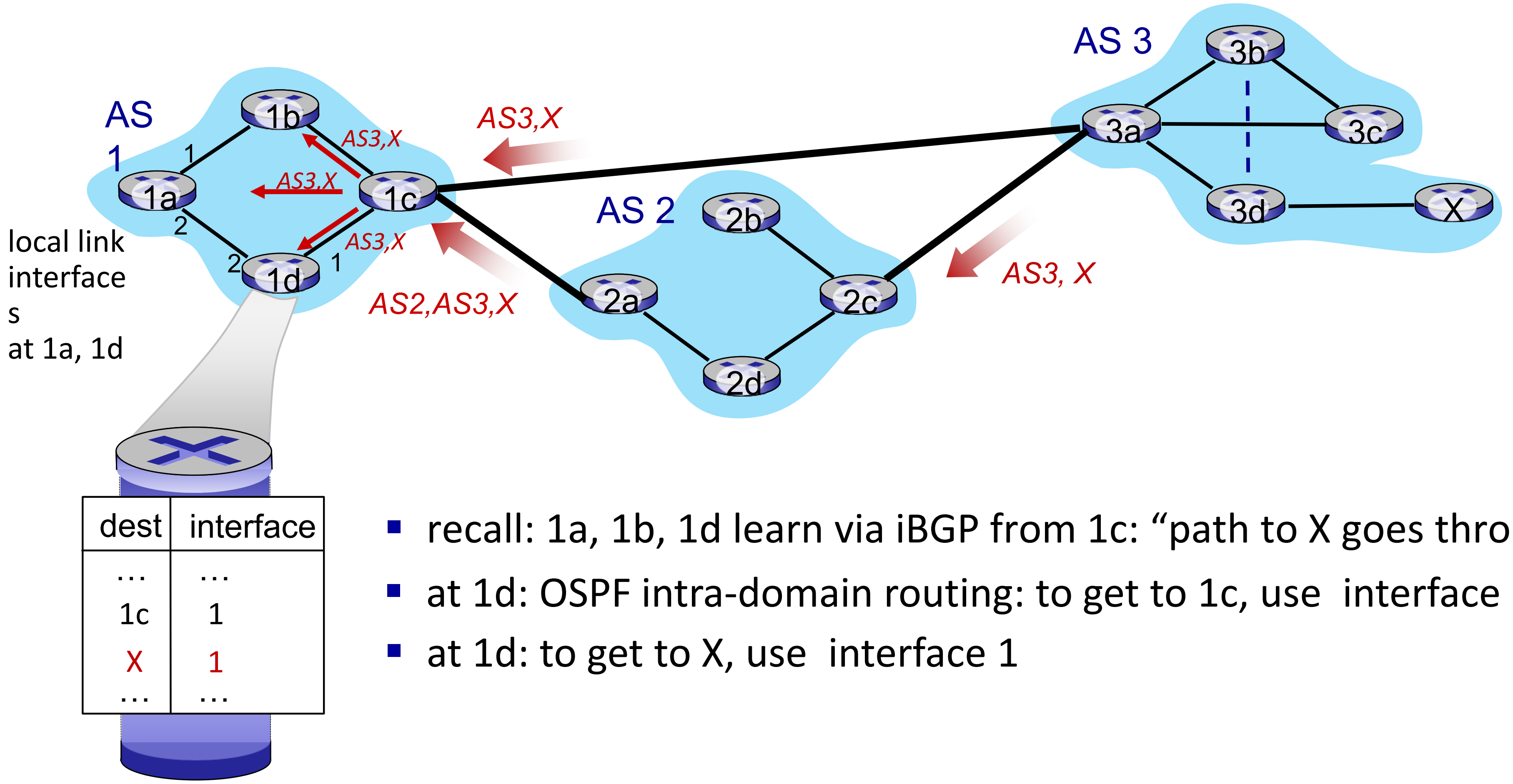
gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- based on *policy,* AS1 gateway router 1c chooses path *AS3,X* and advertises path within AS1 via iBGP

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

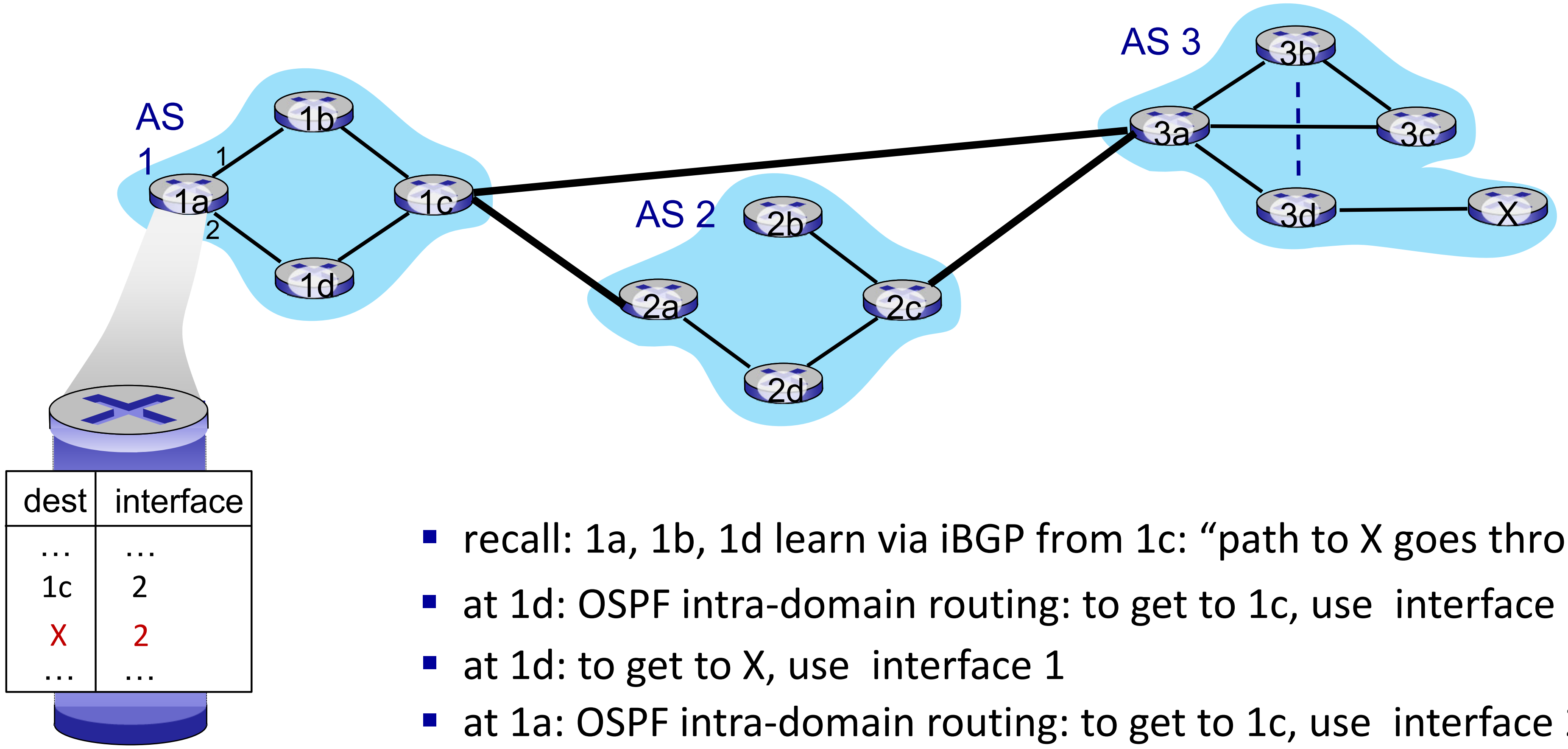
BGP path advertisement



dest	interface
…	…
1c	1
X	1
…	…

- recall: 1a, 1b, 1d learn via iBGP from 1c: "path to X goes through 1c"
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1

BGP path advertisement



dest	interface
…	…
1c	2
X	2
…	…

- recall: 1a, 1b, 1d learn via iBGP from 1c: "path to X goes through 1c"
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1
- at 1a: OSPF intra-domain routing: to get to 1c, use interface 2
- at 1a: to get to X, use interface 2
- Two entries at 1a and 1d, intra-domain and within-domain routing destinations

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its network
- intra-AS: single admin, so policy less of an issue

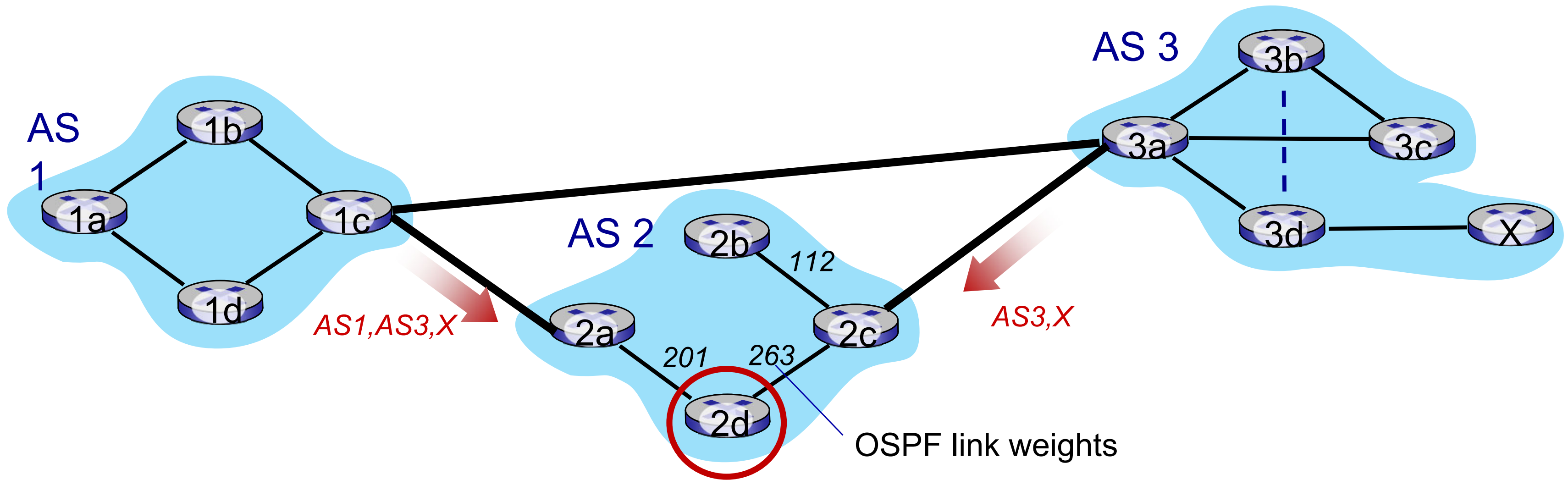
scale:

- hierarchical routing saves table size, reduced update traffic

performance:

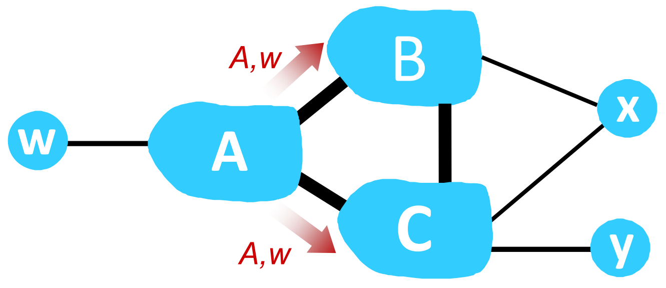
- intra-AS: can focus on performance
- inter-AS: policy dominates over performance

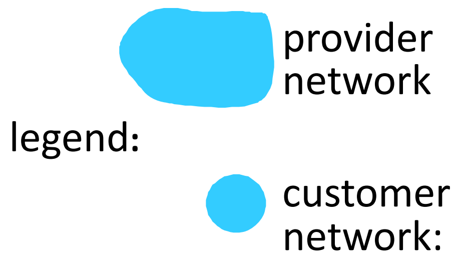
Hot potato routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- hot potato routing: choose local gateway that has least *intra-domain* cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don’t worry about inter-domain cost!

BGP: achieving policy via advertisements

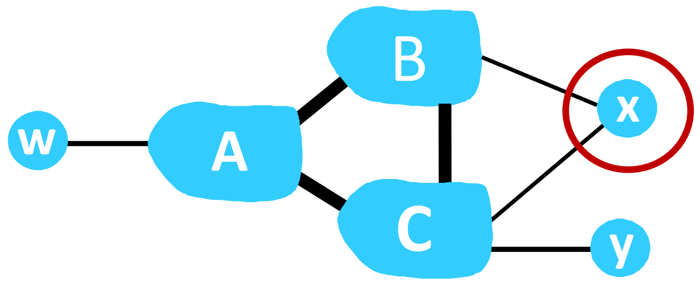


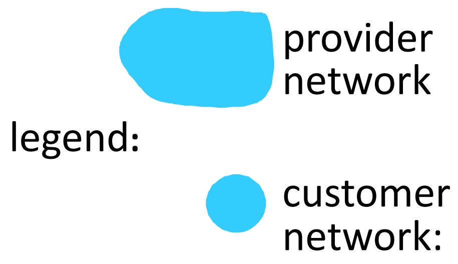


ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C!
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does *not* learn about CBAw path
- C will route CAw (not using B) to get to w

BGP: achieving policy via advertisements (more)





ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

- A,B,C are provider networks
- x,w,y are customer (of provider networks)
- x is dual-homed: attached to two networks
- *policy to enforce:* x does not want to route from B to C via x
  - .. so x will not advertise to B a route to C

BGP route selection

- router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Commonly Used Routing Metrics

- Hop count
  - Number of routers through which a packet will pass
- Ticks
  - Delay on a data link using IBM PC clock ticks (approx. 55ms)
- Cost
  - Arbitrary value, usually based on bandwidth/throughput, dollar expense, or another measurement, that may be assigned by a network administrator
- Bandwidth
  - Data capacity of a link
- Delay
  - Length of time required to move a packet from source to destination

Commonly Used Routing Metrics

- Load
  - Amount of activity of a network resource, such as router or link
- Reliability
  - Usually refers to the bit-error-rate of each network link
- MTU
  - Maximum transmission unit. The maximum frame length in octets that is acceptable to all links on the path

Recommended Reading

- Behrouz A. Forouzan, Data Communications and Networking with TCP/IP Protocol Suite, 6th ed., 2022, Chapters 8

- J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 8th ed., 2022, Chapter 5