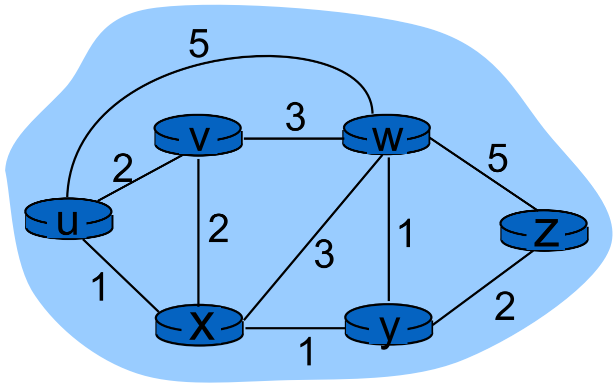
ROUTING PROTOCOLS

Goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- ***path***: sequence of routers packets will traverse in going from given initial source host to given final destination host
- ***Metrics***: "good": "least cost", "fastest", "least congested"

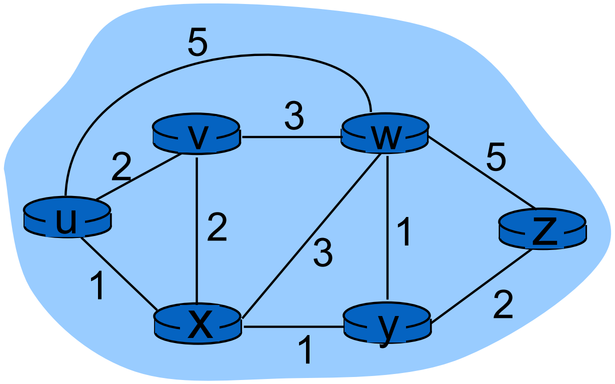


- routing: a "top-10" networking challenge!

GRAPH ABSTRACTION OF THE NETWORK

graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }



E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

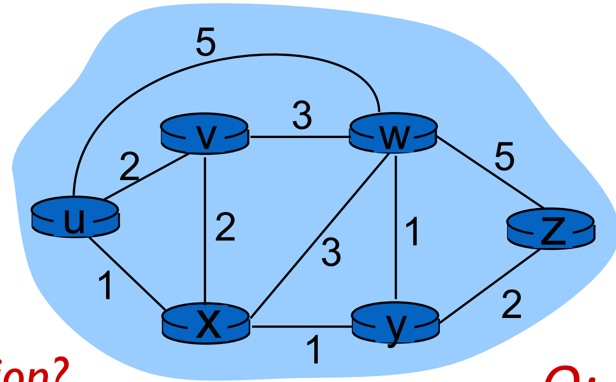
Metric cost of path: C
c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could be directly related to latency,
or inversely related to length
or directly related to bandwidth,
or inversely related to congestion

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$
Note: Cost of non-existing link can be considered Infinity: $C(v,y) = \infty$

key question: what is the ***least-cost*** path between u and z ?
routing algorithm: algorithm that finds that least cost path

ROUTING (ROUTE BUILDING) ALGORITHM CLASSIFICATION



Q: global or decentralized information?

Global (Centralized):

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Link State Routing – Centralized

- Each node is assumed to know the ***state of links*** to its neighbors

- Step 1: Each node ***broadcasts*** its state to ***all*** other nodes
    - Reliable broadcast mechanism
        - flooding
        - *may have sequence number issues*

- Step 2: Each node ***locally computes*** shortest paths (***SSSP***) to all other nodes from its global state
    - Shortest path tree (SPT) algorithm
    - Dijkstra's SPT algorithm

A LINK-STATE ROUTING ALGORITHM

Dijkstra's algorithm (how to decide route)

- 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 from a given source.

Given that we know: (preconditions)

- network topology, link costs of all the nodes.
  - accomplished via "***link state broadcast***"
  - all nodes have *same topology information.*

notation:

- c(x,y): link cost from node x to y; set to ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

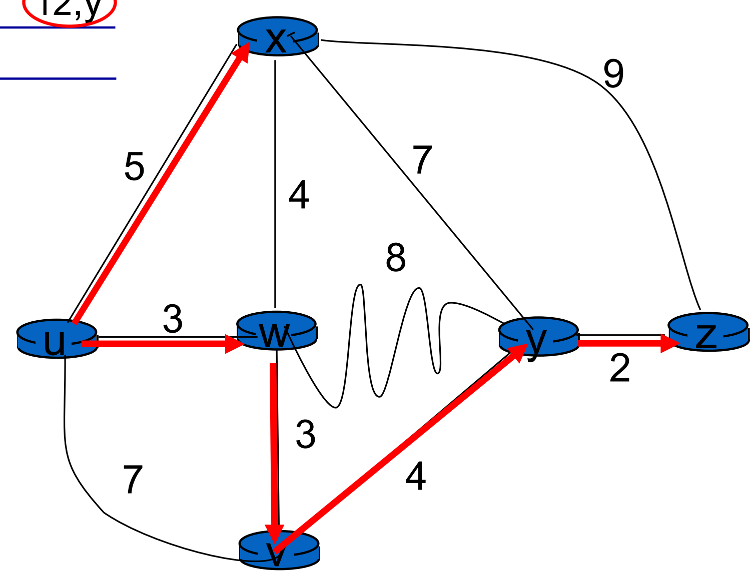
DIJSKTRA'S ALGORITHM

```
Initialization:
   N' = {u}
   for all nodes v
     if v adjacent to u
         then D(v) = c(u,v)
     else D(v) = ∞

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 cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's Algorithm (Example)

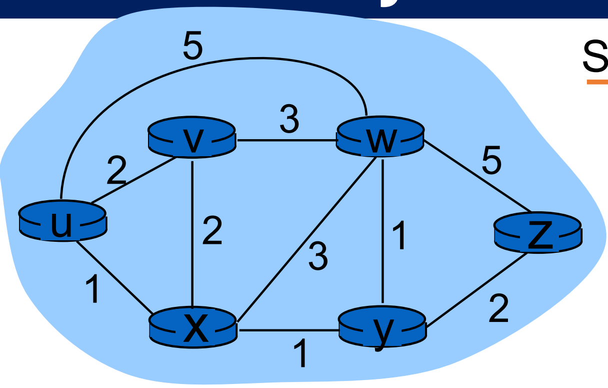
Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



notes:

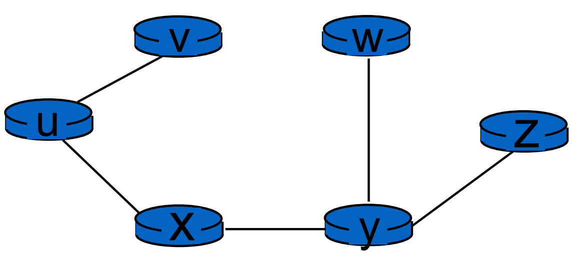
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

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	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					

resulting shortest-path tree from u:



resulting forwarding table in u:

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

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

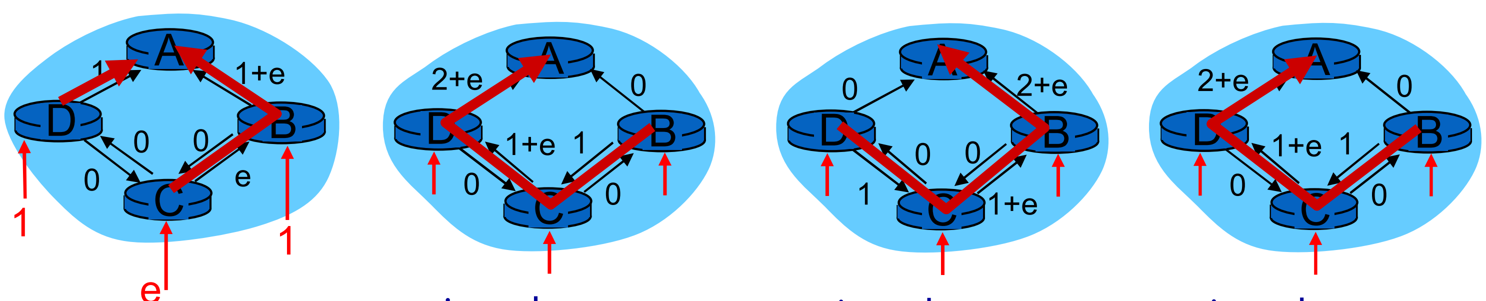
DIJKSTRA'S ALGORITHM, DISCUSSION

algorithm complexity: n nodes

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

oscillations possible:

- e.g., support link cost equals amount of carried traffic:



initially

given these costs, find new routing…. resulting in new costs

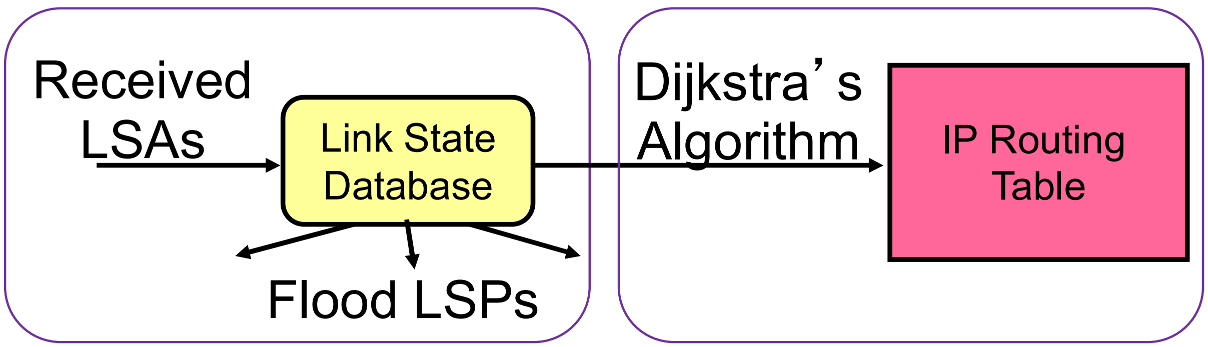
given these costs, find new routing…. resulting in new costs

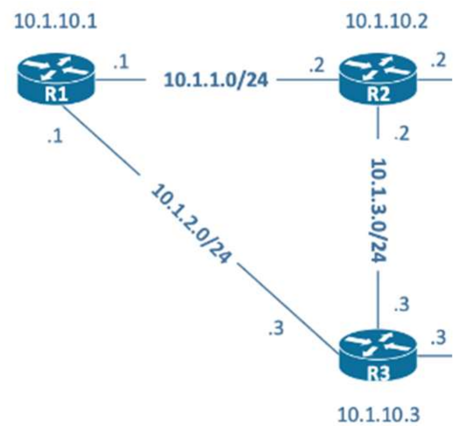
given these costs, find new routing…. resulting in new costs

Link State Algorithm

Stage 1: Flooding:

1) **Periodically distribute link-state advertisement (LSA) to neighbors**
   - **LSA contains cost (e.g., delay) to each neighbor**
   - **Link-state packet (LSP) – packet containing the LSA information.**
2) **Install received LSA in LS database**
3) **Re-distribute LSA to all neighbors**
   - **IF the LSP is the *most-recent* than the one stored (previously seen) THEN:**
     - **Flood LSP: Forward a copy on all links except on the link LSP was received.**
   - **Otherwise, discard LSP.**





Stage 2: Path Computation

1) **Use Dijkstra's shortest path algorithm to compute distances to all destinations**
2) **Install FIB <destination, nexthop> pair in the Forwarding Table**

Link State in Practice - OSPF

- OSPF (Open Shortest Path First Protocol) – *Intra AS Routing*
  - most important and most commonly used routing protocol on the Internet.
  - Also supports ***authentication***, ***additional hierarchy***, ***load balancing***

History and Evolution of OSPF

1989: RFC 1131 OSPFv1

1991: RFC1247 OSPFv2

CIDR based – classes routing

1994: RFC 1583 OSPFv2 (revised)

1997: RFC 2178 OSPFv2 (revised)

1998: RFC 2328 OSPFv2 (current version for IPv4)

2008: RFC 5340 OSPF Version 3

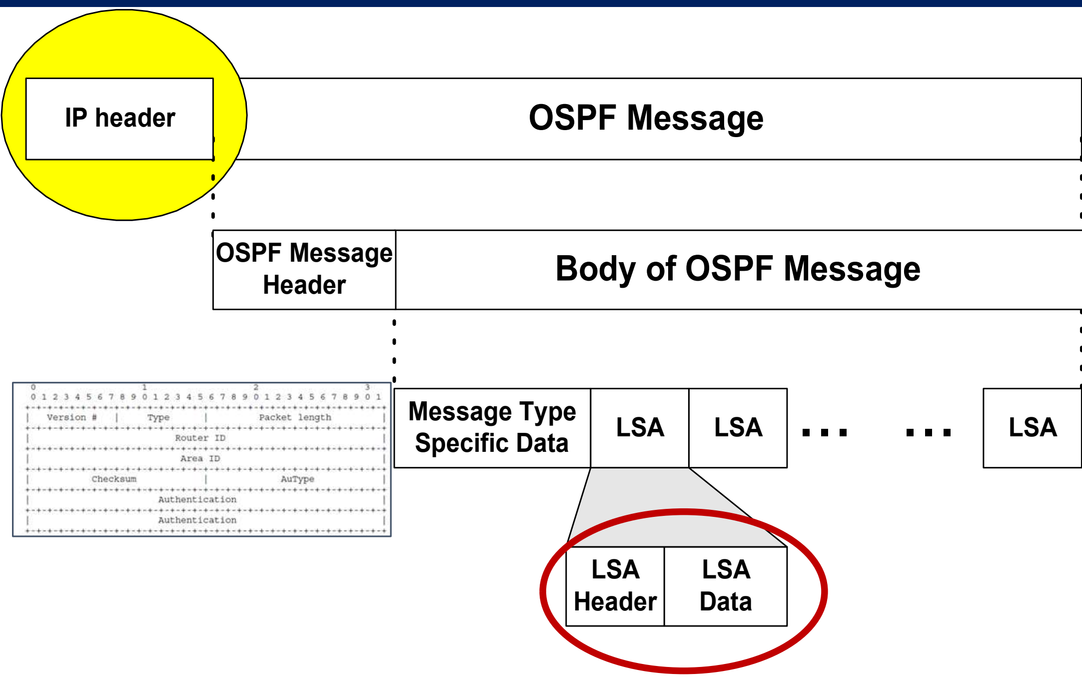
support for IPv6

- *OSPF Packet Structure:*

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Version #   |     Type      |         Packet length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Area ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |             AuType            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

AuType	Description
0	Null authentication
1	Simple password
2	Cryptographic authentication
All others	Reserved for assignment by the IANA (iana@ISI.EDU)

OSPF MESSAGE STRUCTURE



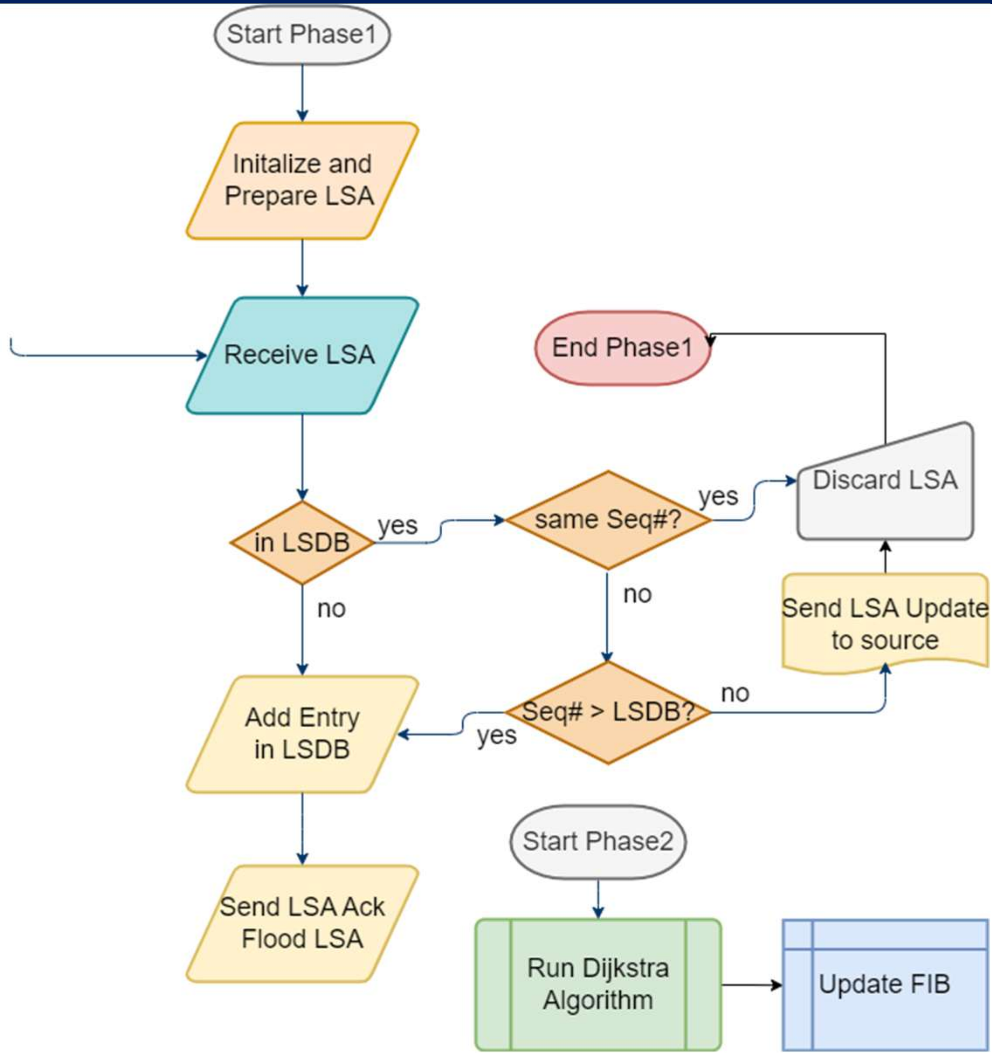
32-bit LS sequence number field, does not wrap
On startup, router need not wait: can start with a lowest sequence number
LSP's compared on basis of sequence number
LSP's purged after about an hour
Synchronized expiration of LSPs
expired LSP reflooded with age zero

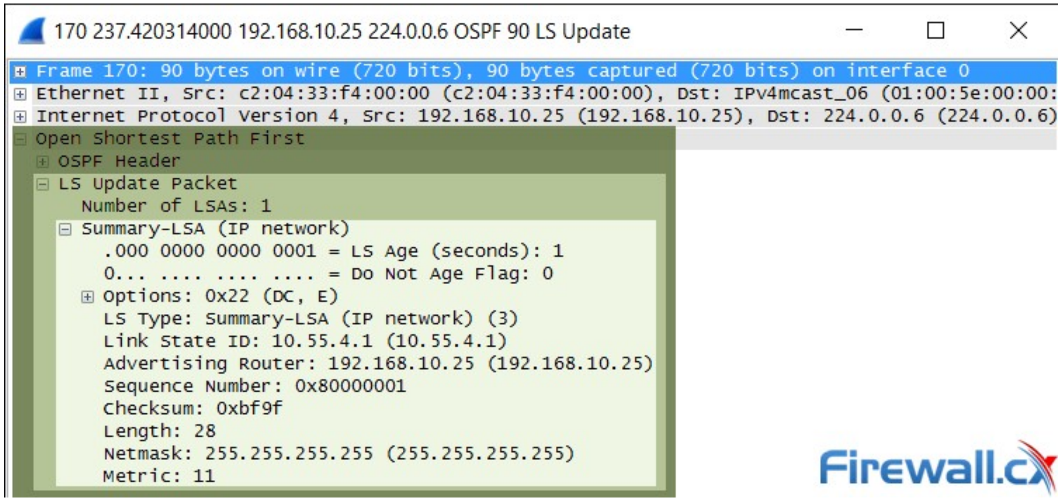
LS Age	Options	Type=1
Link state ID		
Advertising router		
LS sequence number		
LS checksum	Length	

OSPF packets are not carried as UDP or TCP payload!
OSPF has its own IP protocol number: **89**
TTL: set to 1 (in most cases)

Destination IP: neighbor's IP address or multicast address for a broadcast environment:
224.0.0.5 (ALLSPFRouters) or 224.0.0.6 (AllDRouters: (designated and backup designated only)

Link State Algorithm –typical 2 stage Workflow at a Router





Problem: Router Failure

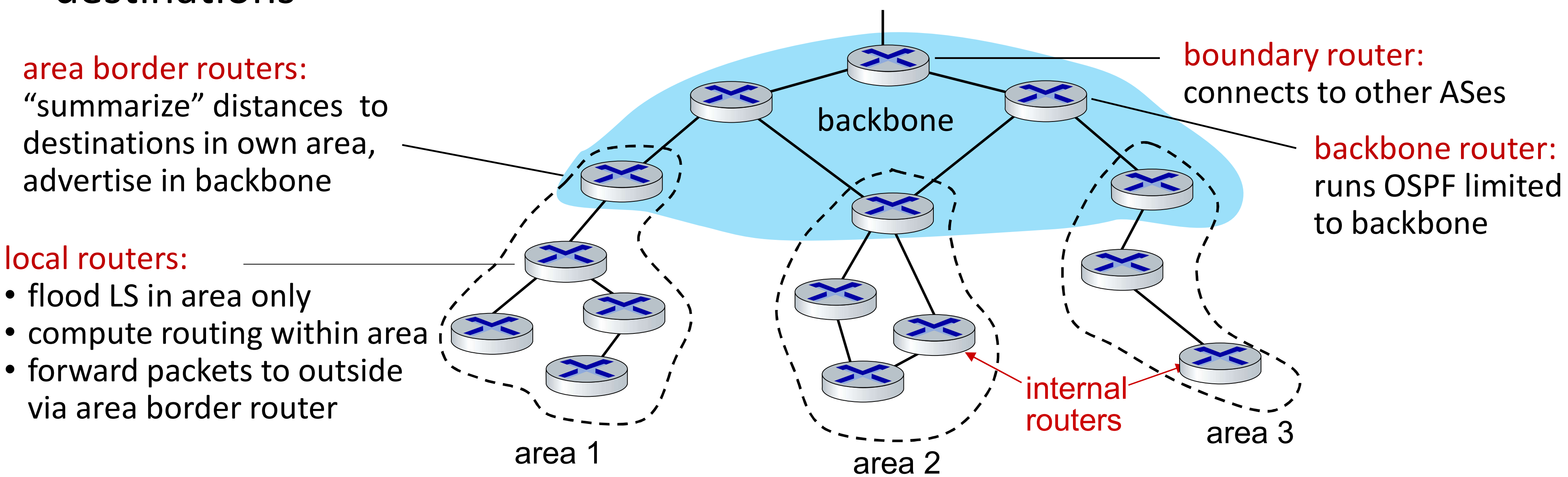
- A failed router then comes up but does not remember the last sequence number it used before it crashed.
- New LSPs by this router may be ignored by others if they have sent with a lower sequence number

One solution: LSP Aging

- Nodes periodically decrement age (TTL) of stored LSPs
- LSPs expire when TTL reaches 0
  - LSP is re-flooded once TTL = 0
- Alternative for a rebooted router is to wait until all LSPs have expired.
- Trade-off between frequency of LSPs and router wait after reboot

HIERARCHICAL OSPF

- two-level hierarchy: local area, backbone.
  - link-state advertisements flooded only in area, or backbone
  - each node has detailed area topology; only knows direction to reach other destinations



Another/~Archaic Approach to Routing

Distance Vector –

I can know the Distance(s) From My Neighbor

Once I know the distances, I can know the network

Vs.

Link State –

I can know the Network Topology From My Neighbor

Once I know the Network, I can compute the Distances

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

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

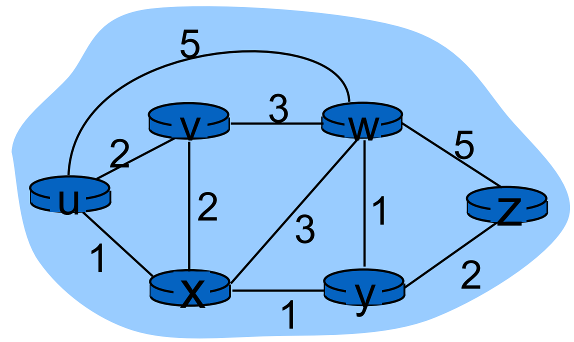
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Bellman-Ford equation (dynamic programming)

Let $d_x(y)$:= cost of *least-cost* path from x to y

then

$$d_x(y) = \underbrace{\min_v}_{\text{min taken over all neighbors v of x}} \{\overbrace{c(x,v)}^{\text{cost to neighbor v}} + \underbrace{d_v(y)}_{\text{cost from neighbor v to destination y}} \}$$



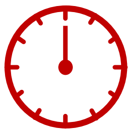
clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

- node x:
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors.
  - For each neighbor v, x maintains $\mathbf{D}_v=[D_v(y): y \in N$

B-F equation says:

$$\begin{aligned} d_u(z) = \min \{ & c(u,v) + d_v(z), \\ & c(u,x) + d_x(z), \\ & c(u,w) + d_w(z) \} \\ = \min \{ & 2 + 5, \\ & 1 + 3, \\ & 5 + 3\} = 4 \end{aligned}$$

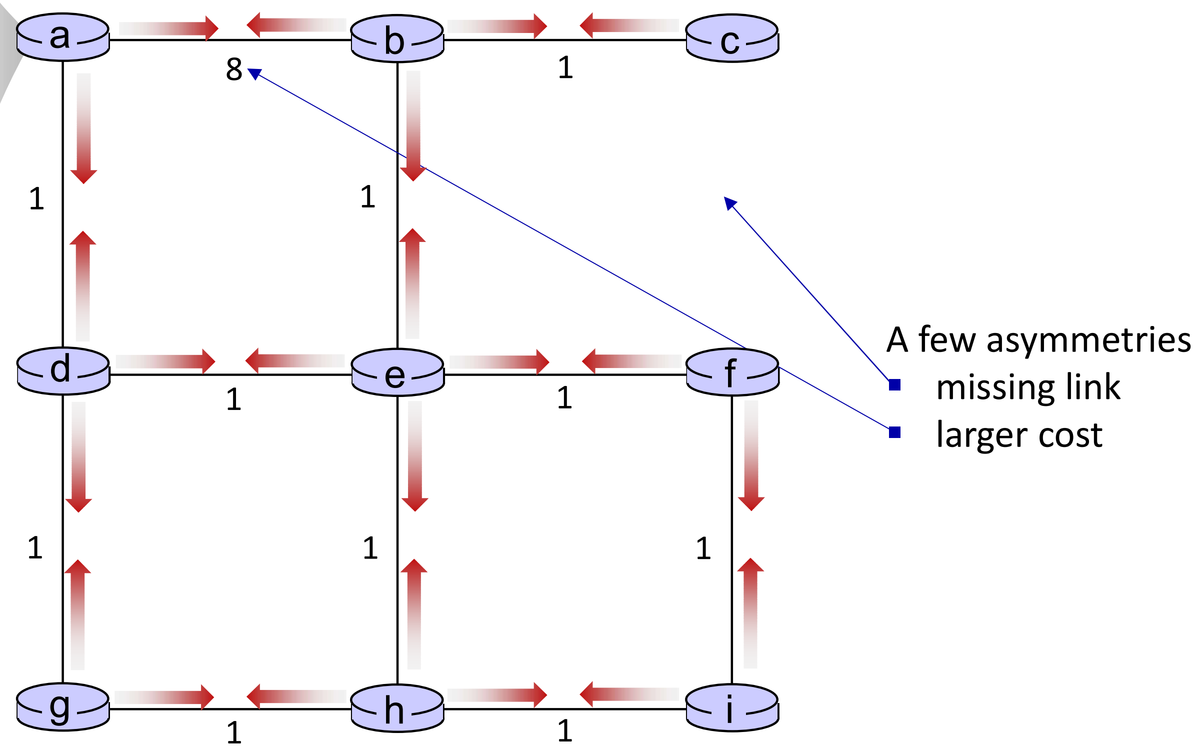
DISTANCE VECTOR: EXAMPLE



t=0

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

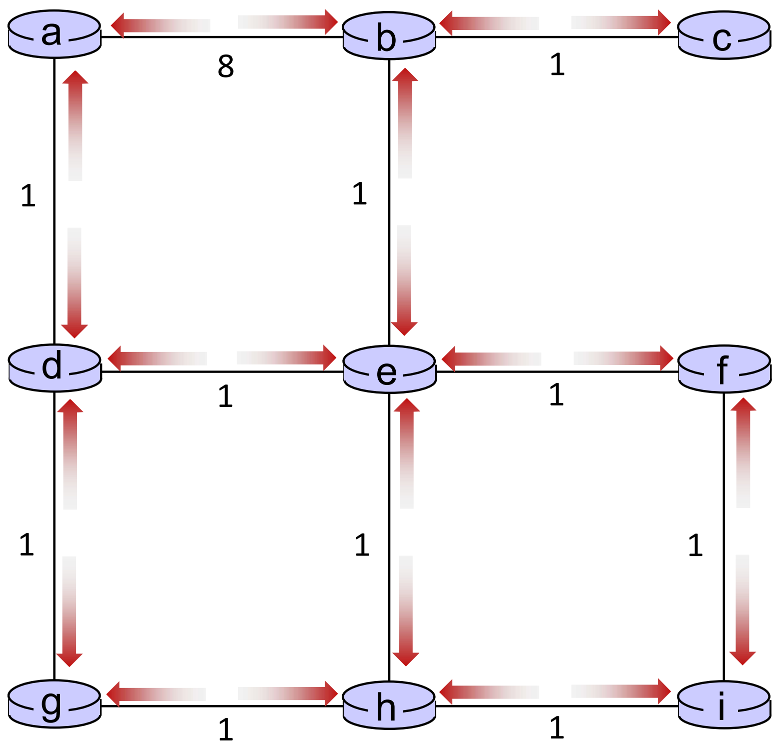
DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



DISTANCE VECTOR EXAMPLE: ITERATION

t=1

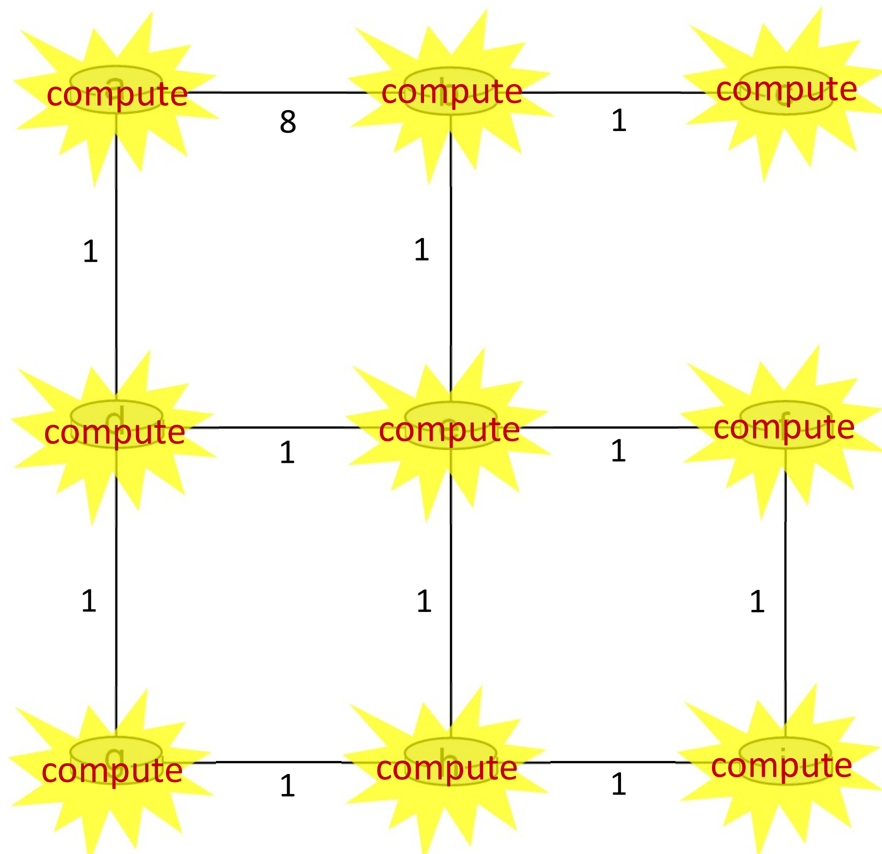
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



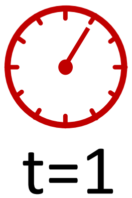
Distance vector example: iteration

t=1

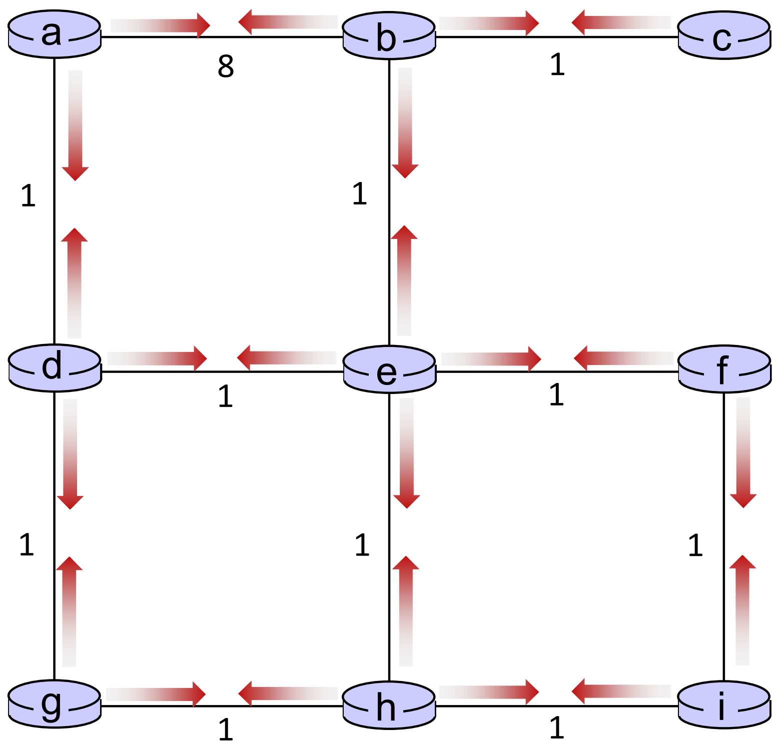
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



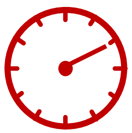
DISTANCE VECTOR EXAMPLE: ITERATION



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



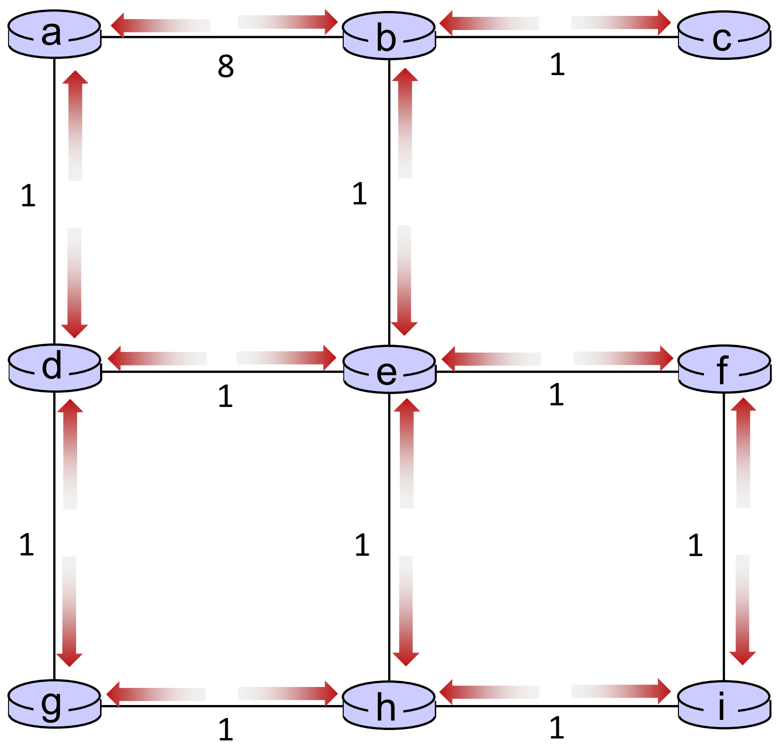
Distance vector example: iteration



t=2

All nodes:

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

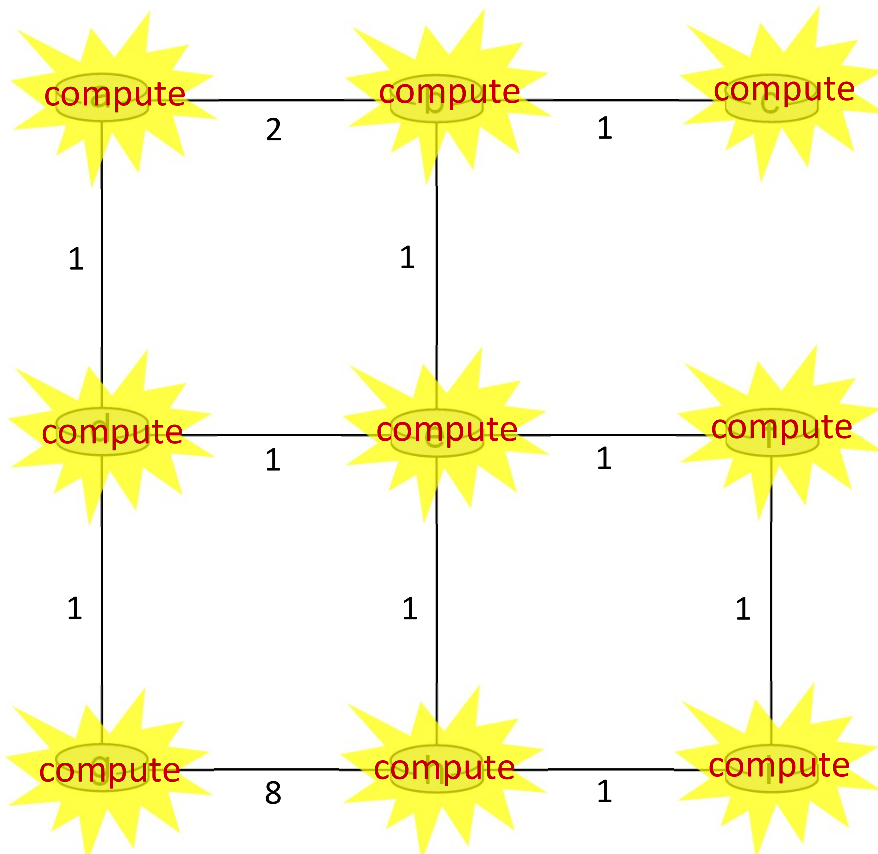


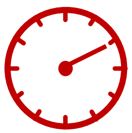
DISTANCE VECTOR EXAMPLE: ITERATION

t=2

All nodes:

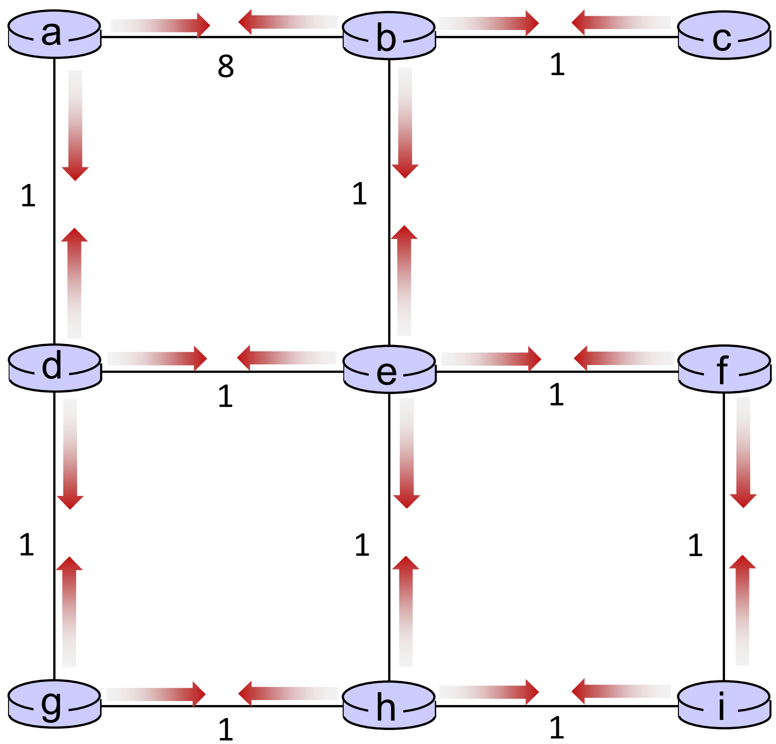
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors





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

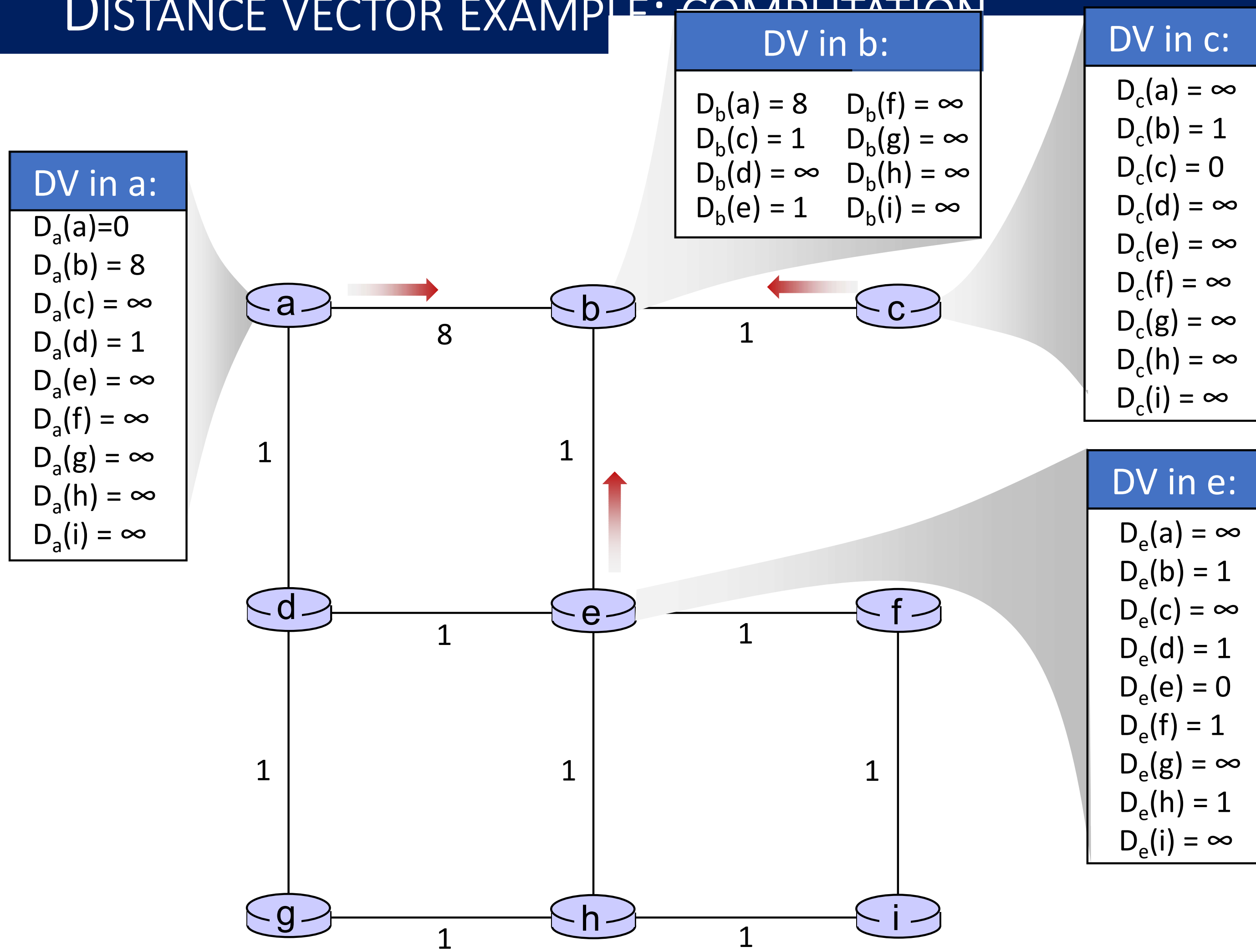
…. and so on

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

DV in a:

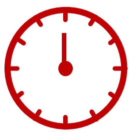
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

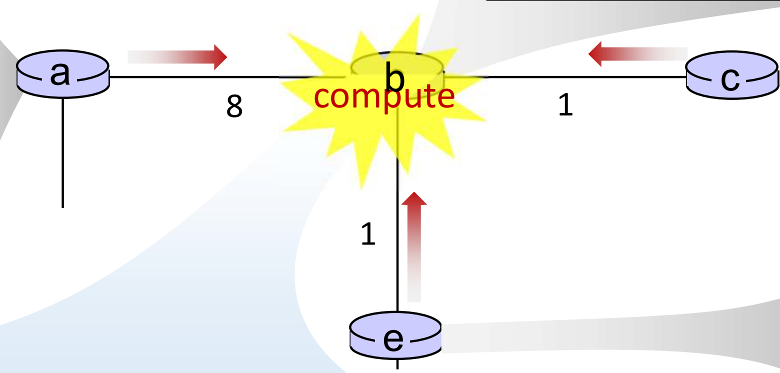
DV in b:

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

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

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

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

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

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

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

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

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

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

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

DV in e:

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

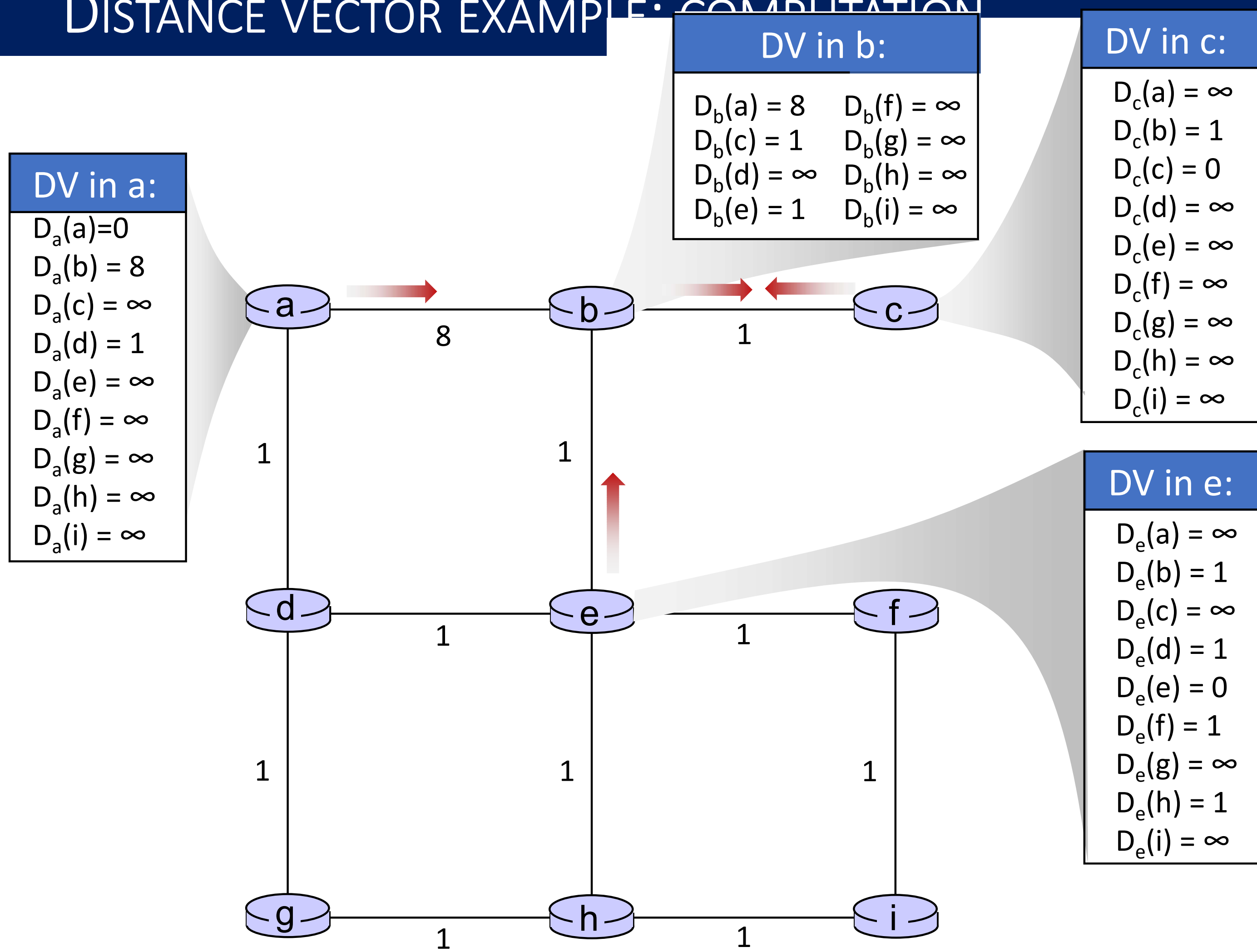
DV in b:

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

Distance vector example: computation

t=1

- c receives DVs from b



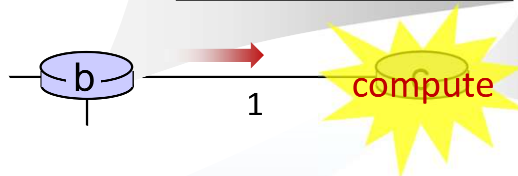
DISTANCE VECTOR EXAMPLE: COMPUTATION

DV in b:

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

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:

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

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

DISTANCE VECTOR EXAMPLE: COMPUTATION

DV in b:

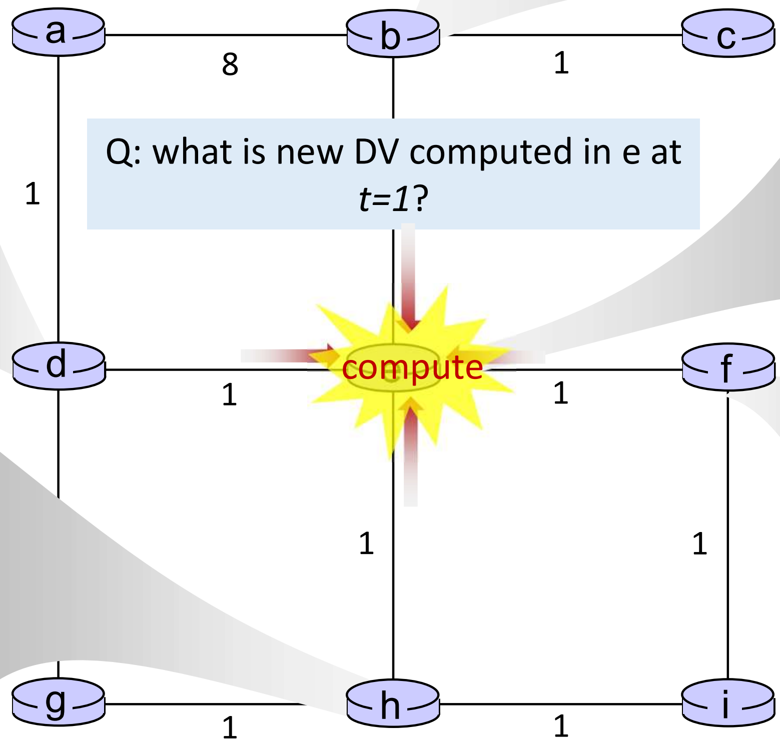
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

t=1

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

DV in d:

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



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

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:

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

DV in f:

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

Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

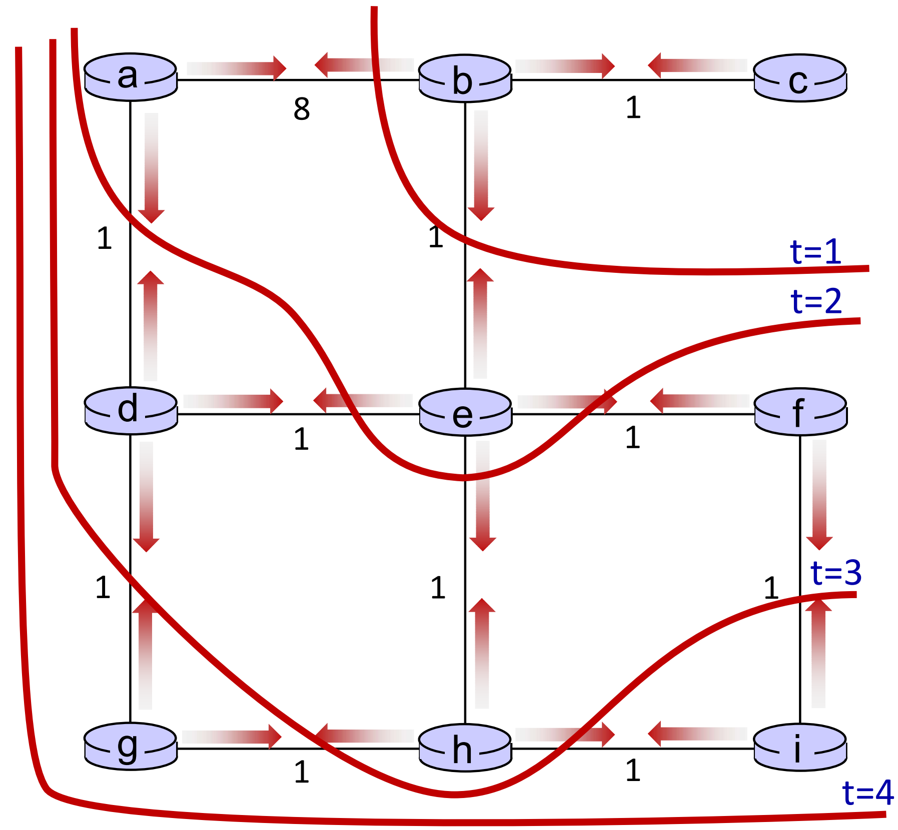
t=0 c's state at t=0 is at c only

t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b

t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well

t=3 c's state at t=0 may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well

t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Distance Vector

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

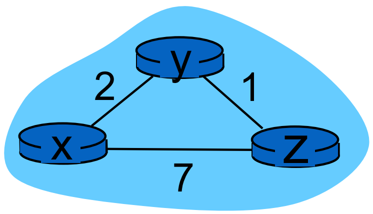
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

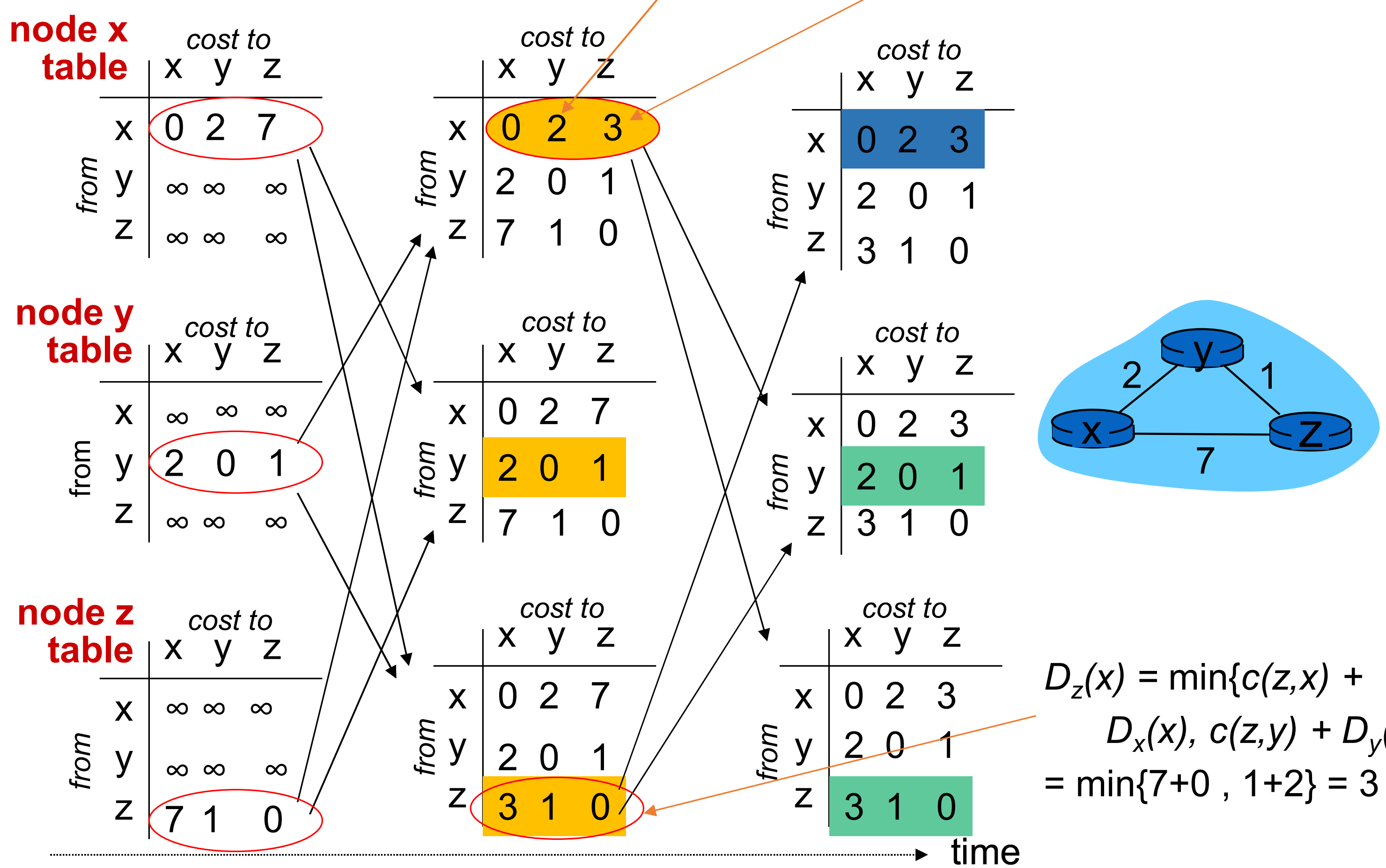
from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) +$
$D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$



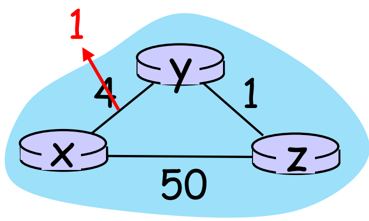
$D_z(x) = \min\{c(z,x) +$
$D_x(x),\ c(z,y) + D_y(x)\}$
$= \min\{7+0\ ,\ 1+2\} = 3$

4-91

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors



"good news travels fast"

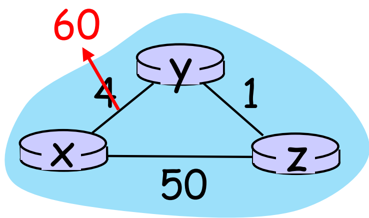
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

link cost changes:

- node detects local link cost change and notify neighbors

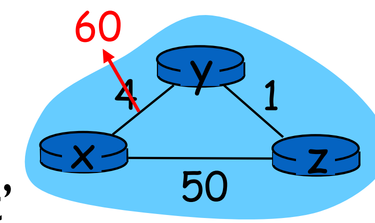


"bad news travels slow" – count-to-infinity problem:

- *y* sees direct link to *x* has new cost 60, but z has said it has a path at cost of 5. So *y* computes "my new cost to x will be 6, via z); notifies *z* of new cost of 6 to *x.*
- *z* learns that path to *x* via *y* has new cost 6, so *z* computes "my new cost to *x* will be 7 via y), notifies *y* of new cost of 7 to *x.*
- *y* learns that path to *x* via *z* has new cost 7, so *y* computes "my new cost to *x* will be 8 via y), notifies *z* of new cost of 8 to *x.*
- *z* learns that path to *x* via *y* has new cost 8, so *z* computes "my new cost to *x* will be 9 via y), notifies *y* of new cost of 9 to *x.*

  …

DISTANCE VECTOR: HOW TO OVERCOME COUNT TO INFINITY!



Different Approaches have been proposed to avoid/prevent Routing Loops:

- **Split Horizon**: (Kind of Avoidance/Passive Prevention)
  - 'Never advertise a route out of the interface through which you learned it'
  - Prevent a router from advertising the route back on the same interface from which it learns the route.
- **Route Poisoning**: (– almost the opposite of split horizon) Upon Failure detection, poison the route by advertising infinity metric to the neighbors.
  - Until the Route is activated – apply split horizon
  - When route fails, route poisoning overrules the split horizon.
- **Holddown Timer**: Upon 'Route Poison' updates disregard any other routing updates until the holddown time expires (typically 180 seconds)
- **Poison Reverse** (Kind of Active Prevention)
  - 'Once you learn a route through an interface advertise it as unreachable back through the same interface'

Split Horizon with poisoned reverse:

Try to Read RFC 1058

Q: will this completely solve count to infinity problem?

COMPARISON OF LS AND DV ALGORITHMS

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

Message Update interval

- ***LS:*** large duration (~30 mins) or on change
- ***DV:*** small time interval(~1 min) or on change.

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - *error propagate through the network*

CPU and Bandwidth tax:

LS:

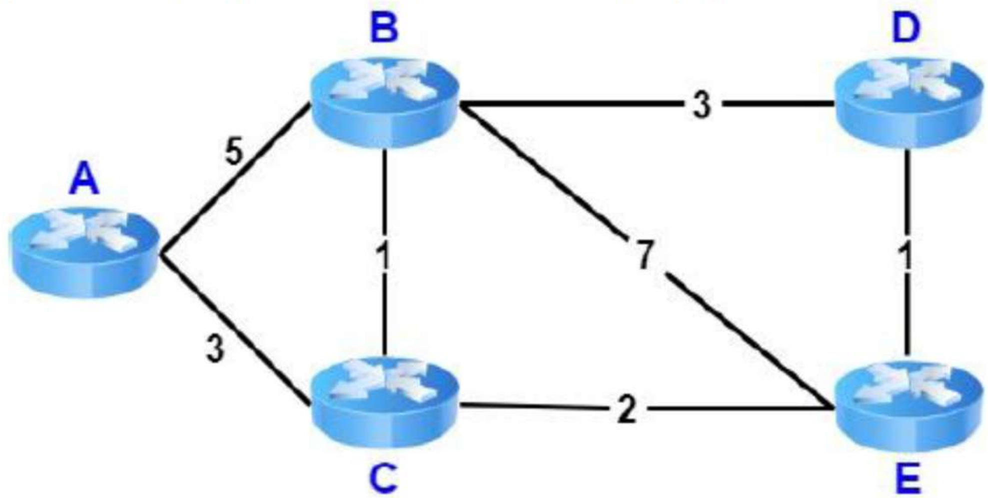
- Flooding small packets
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - *error propagate through the network*

PROBLEM

5. In the following network topology consider that it takes 1ms to transmit a message on each link. Consider running the Djkstra or Bellman-Ford algorithm to find the shortest path for the topology takes negligible time i.e. 0ms.**(10 pts)**



   a. What is the convergence time needed for LSR and DVR respectively? **(2 pts)**
   b. A total of how many packets are exchanged for route convergence in LSR and DVR respectively? Note: A total includes all the packets that are broadcast/flood in the network by different routers; (I.e. A packet is considered per hop) **(4 pts)**
   c. If the cost of link B-C changes from 1 to 5. How quickly can the LS and DVR converge again? **(4 pts)**

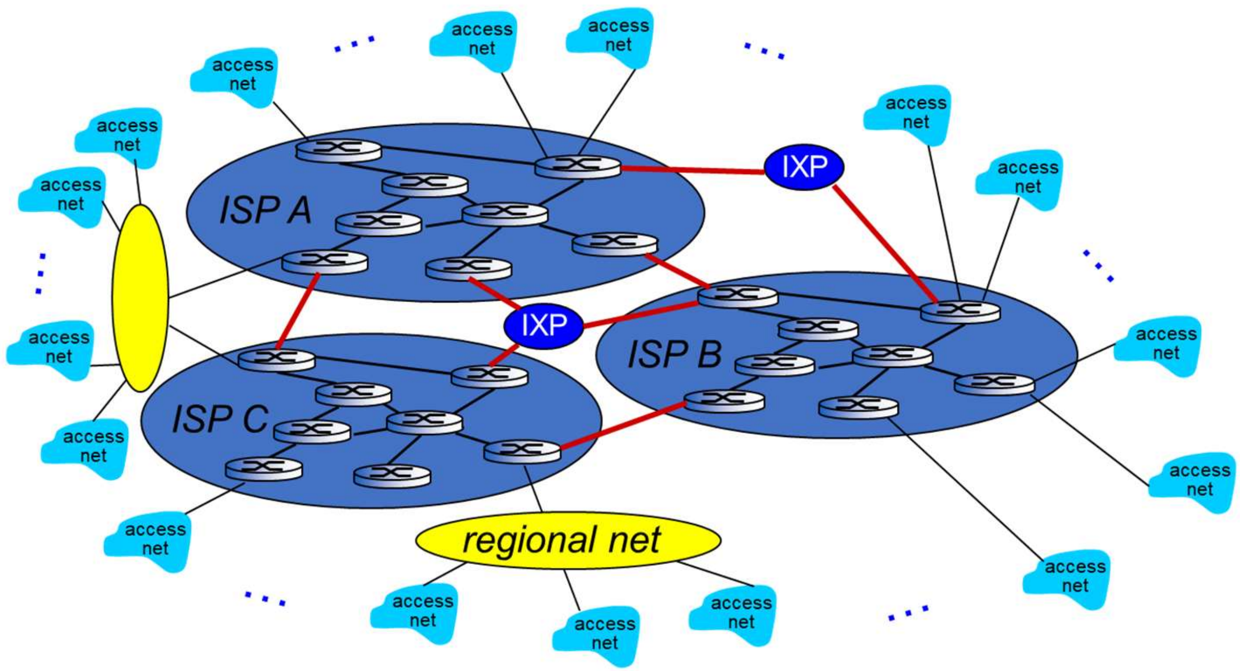
LSR

Node	Round T0 (# new msg x # links to send)	Round T1 (1 hop away)	Round T2 (2 hop away)
A	1 x 2 = 2 [(B,C)]	2 x 1 = 2 [B->C, C->B]	2 x 1 = 2 (From D and E
B	1 x 4 = 4 (A,C,D,E)	4 x 3 = 12 (From A,C,D,E to every other links)	---
C	1 x 3 = 3 (A,B, E)	3 x 2 = 6	----
D	1 x 2 = 2 (B,E)	2 x 1 = 2	1 x 1 = 1 (From A)
E	1 x 3 = 3 (C,D,E)	3 x 2 = 6	1 x 2 = 2 (From A
Total Msgs	14	28	5

DVR

Node	Round T0 (# new msg x # links to send)	Round T1 (All nodes have updates due to T0)	Round T2 (Only A,D will have updates
A	1 x 2 = 2	1 x 2 = 2	1 x 2 = 2
B	1 x 4 = 4	1 x 4 = 4	----
C	1 x 3 = 3	1 x 3 = 3	----
D	1 x 2 = 2	1 x 2 = 2	1 x 2 = 2
E	1 x 3 = 3	1 x 3 = 3	----
Total Msgs	14	14	4

MAKING ROUTING SCALABLE – MEET INTERNET DEMANDS



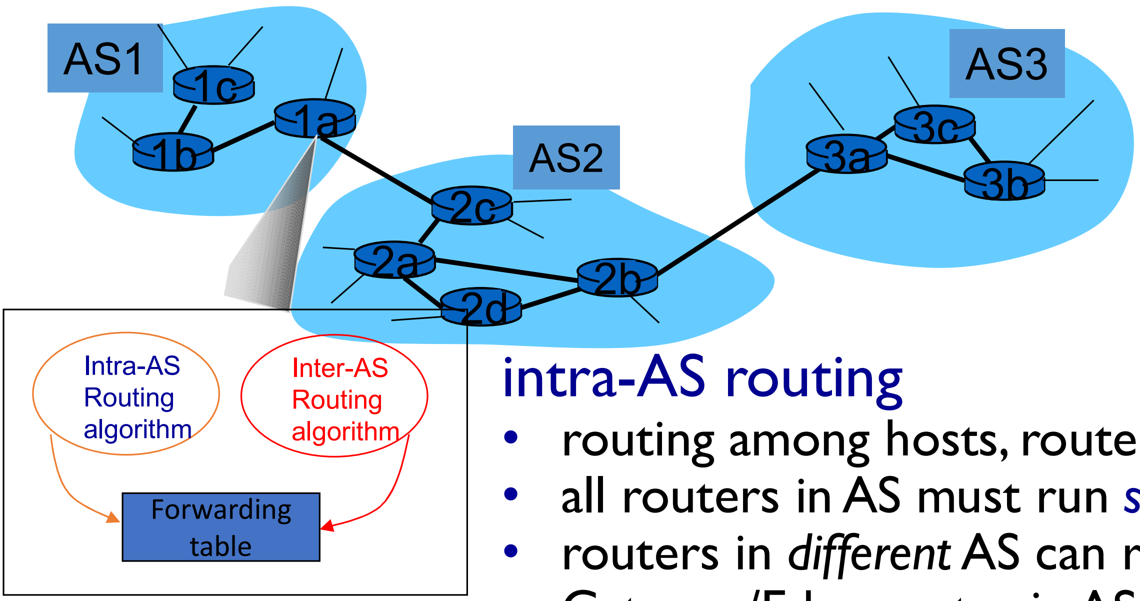
scale: with billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- **internet = network of networks**
- each network admin may want to control routing in its own network.

aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

intra-AS routing

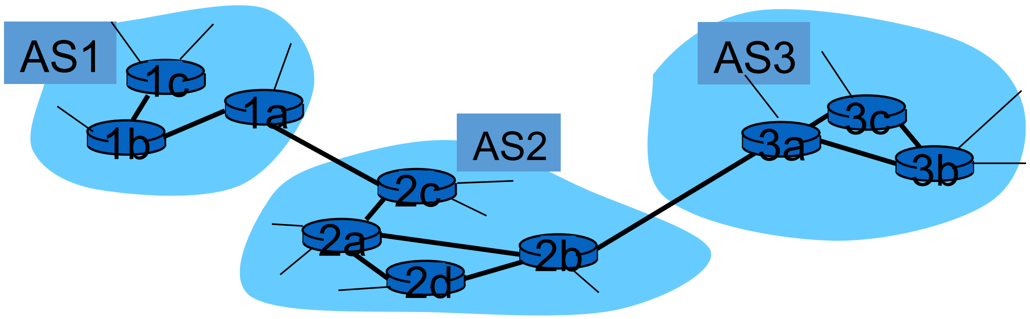
- routing among hosts, routers within the same AS ("network")
- all routers in AS must run *same* intra-domain routing protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- Gateway/Edge router in AS: has link(s) to router(s) in other AS'es

inter-AS routing

- routing among the AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

INTER-AS TASKS

- How should any router in AS2 forward datagram destined outside of AS2?



Inter-AS routing:

1. learn about destinations reachable through other As'es (AS1 & AS3)
2. propagate this information to all routers within AS2.

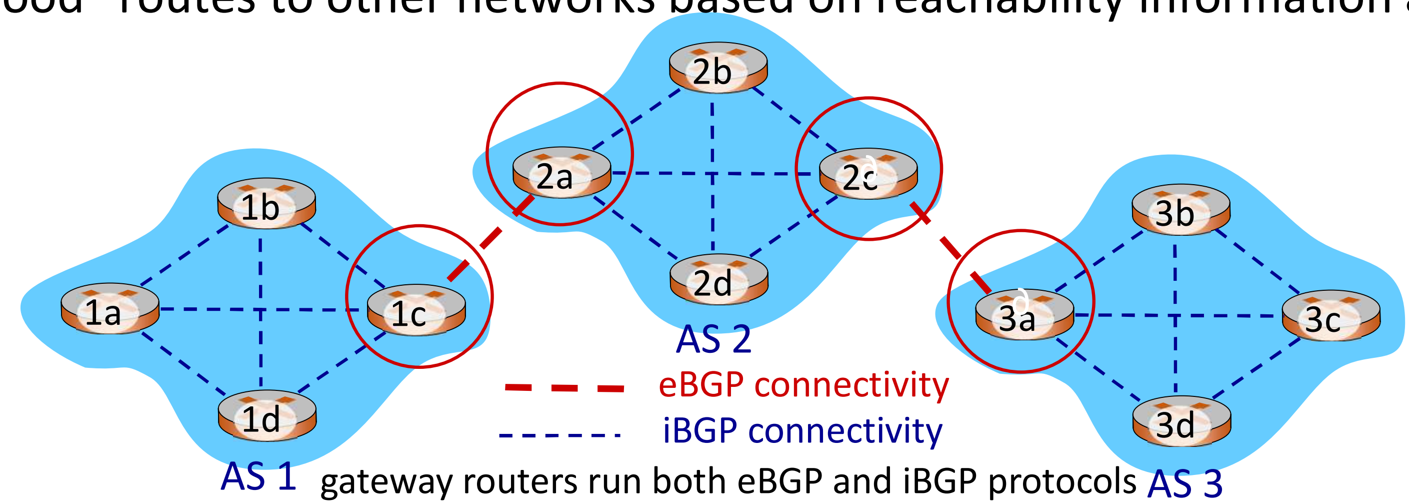
- *Intra-AS routing* *a.k.a* *interior gateway protocols (IGP):* most common protocols:
  - RIP: Routing Information Protocol (now mostly RIPv2) – *DVR*
  - OSPF: Open Shortest Path First -- *LSR*
  - IS-IS: Intermediate System routing protocol essentially same as OSPF -- *LSR*
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary *until 2016*) – *DVR*
- *Inter-AS routing* *a.k.a* *exterior gateway protocols:* most common protocols:
  - BGP: Border Gateway Protocol – *PVR*
  - EGP: Exterior Gateway Protocol (way back in 1982, now obsolete) – *??*

INTER-AS ROUTING: BORDER GATEWAY PROTOCOL (BGP)

BGP: *the* de facto inter-domain routing protocol

- "glue that holds the ***Internet*** together"
- allows subnet to advertise itself, and the destinations it can reach, to rest of Internet: *"I am here, here is who I can reach, and how".*

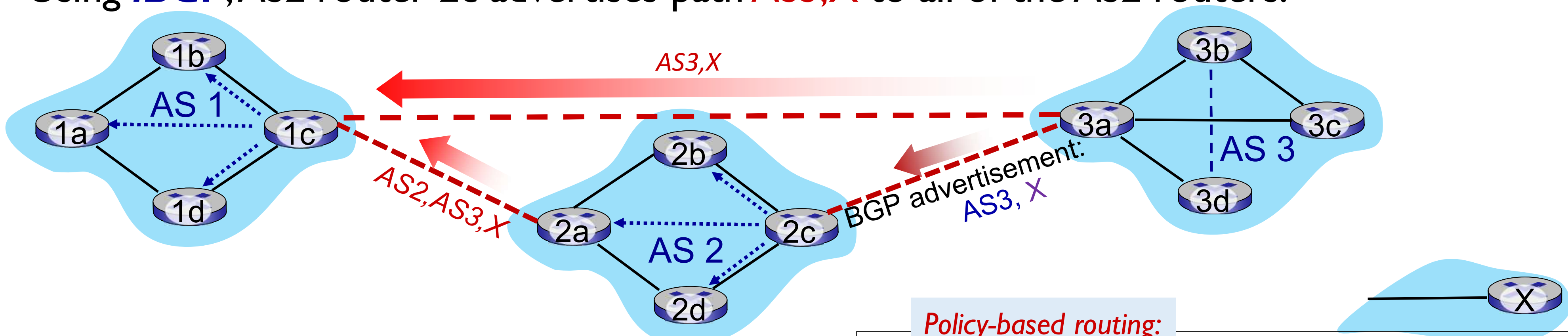
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring AS'es
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and *policy*



gateway routers run both eBGP and iBGP protocols

BGP Outline of Key Operations

- BGP session: BGP routers ("peers") exchange BGP messages over a TCP connection: ***port 179***
  - advertise *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- Using ***eBGP***, AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X
- Using ***iBGP***, AS2 router 2c advertises path AS3,X to all of the AS2 routers.

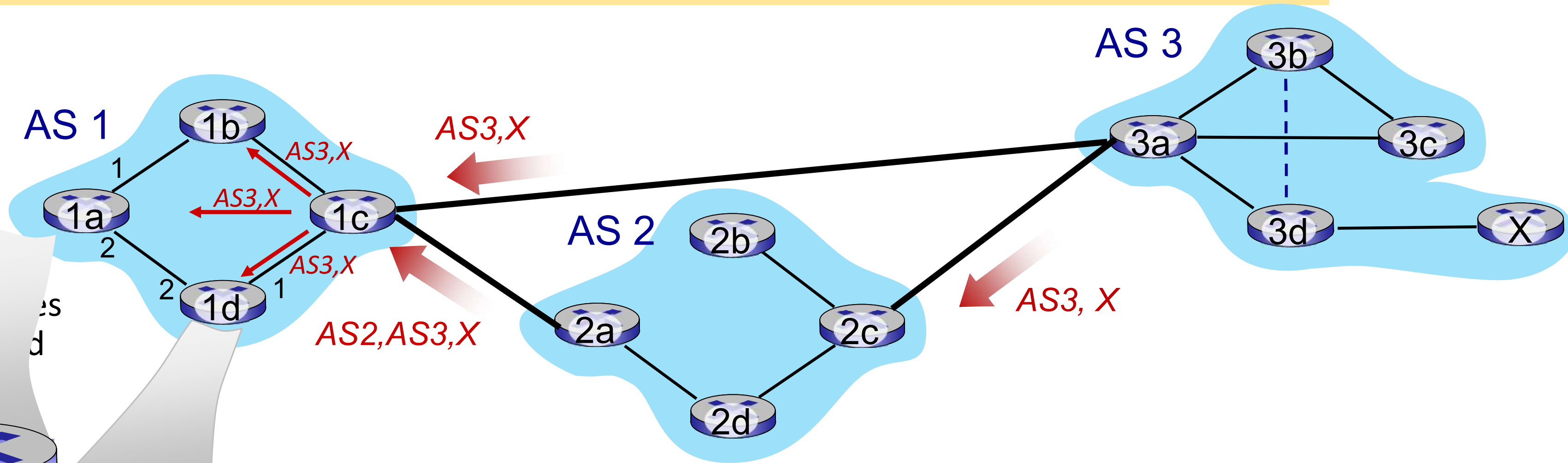


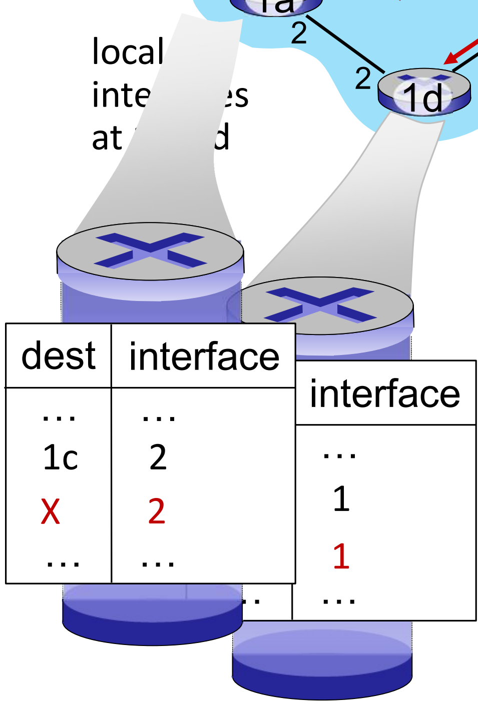
Policy-based routing:
gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS2).
AS policy also determines whether to *advertise* path to other neighboring ASes or not!

- 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 the next-hop AS

BGP, OSPF, FORWARDING TABLE ENTRIES

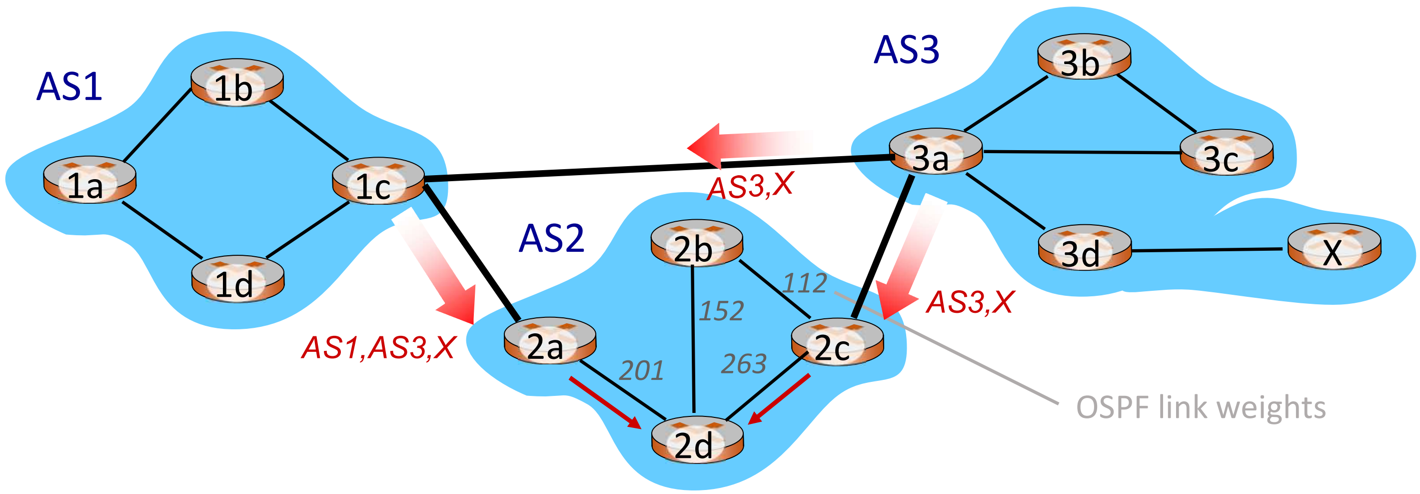
Q: how does router set forwarding table entry to distant prefix?





- 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

HOT POTATO ROUTING

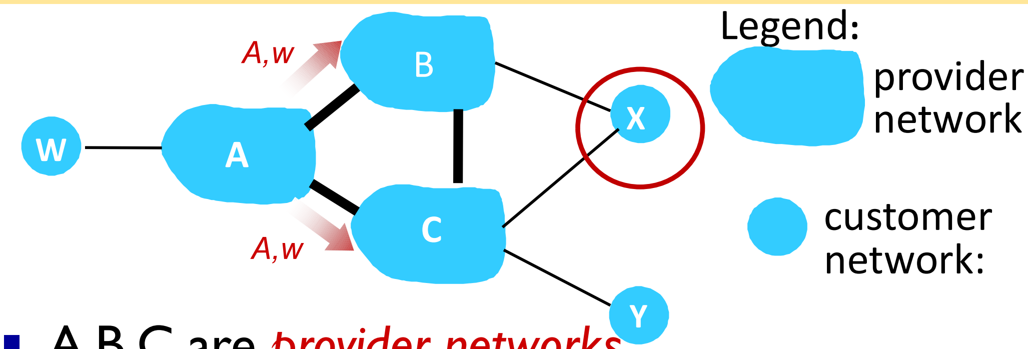


- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato routing:* choose local gateway that has least intra-domain cost

e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

BGP: ACHIEVING POLICY VIA ADVERTISEMENTS

Q: Suppose an ISP does not want to carry transit traffic between other ISPs?



- 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
  - Solution: X will not advertise to B a route to C
- *policy to enforce:* B does not want to route from C to A via B
  - A advertises path Aw to B and to C
- Solution: B *chooses not to advertise* BAw to C:
  - C does not learn about CBAw path
  - C will route CAw (not using B) to get to w

SUMMARY: WHY DIFFERENT INTRA-AS AND INTER-AS ROUTING?

policy:

- inter-AS: admin wants control over how its traffic is routed, who routes through its network, etc.
- 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.

BGP MESSAGES

- BGP messages exchanged between peers over TCP connection; using TCP port 179.
- 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 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