

# Network layer control plane: our goals

- understand principles behind network control plane:
  - traditional routing algorithms
  - SDN controllers
  - network management, configuration
- instantiation, implementation in the Internet:
  - OSPF, BGP
  - OpenFlow, ODL and ONOS controllers
  - Internet Control Message Protocol: ICMP
  - SNMP, YANG/NETCONF

# Network layer: “control plane” roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol



- network management, configuration
  - SNMP
  - NETCONF/YANG

# Network-layer functions

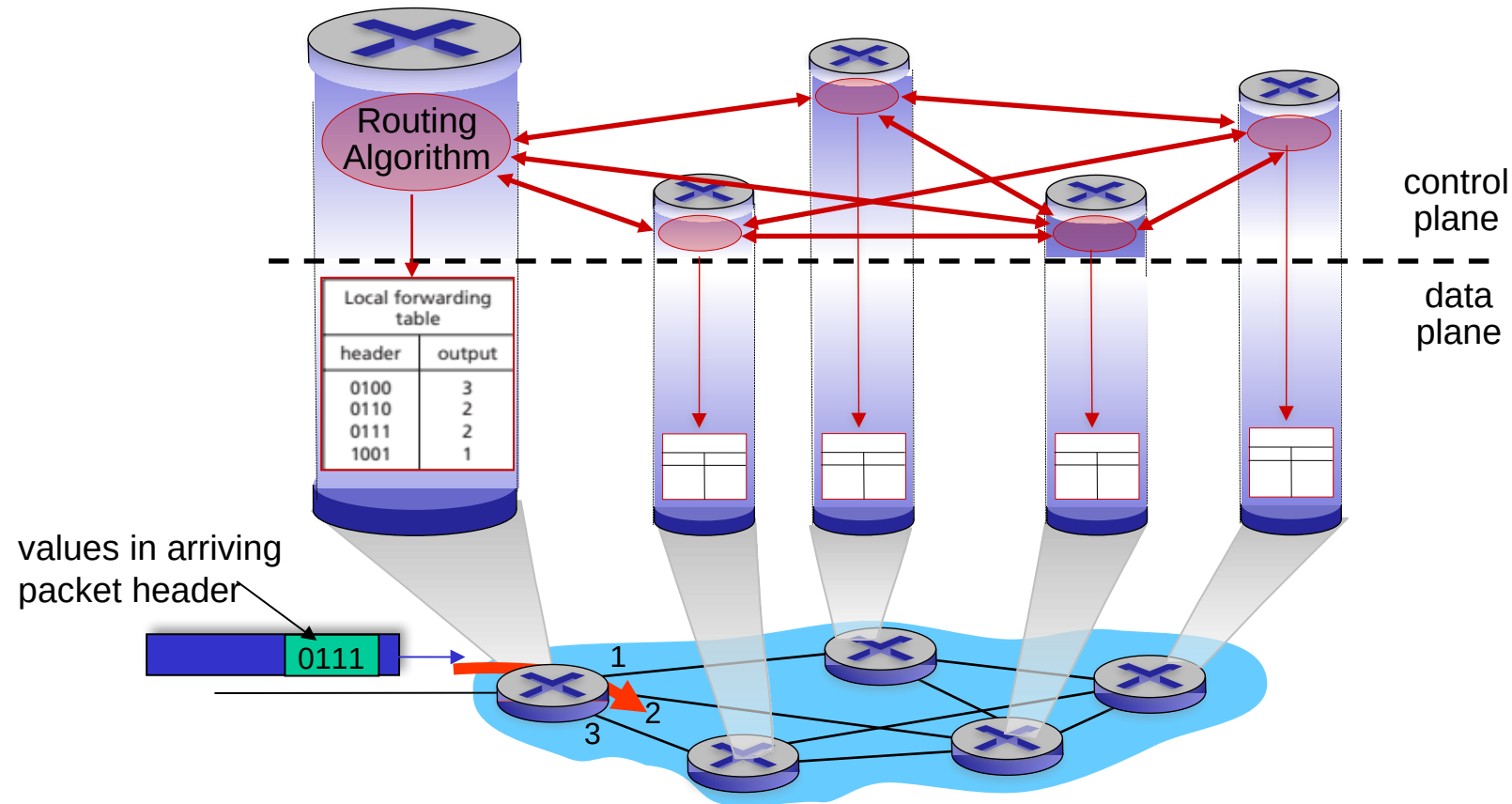
- **forwarding:** move packets from router's input to appropriate router output *data plane*
- **routing:** determine route taken by packets from source to destination *control plane*

## Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

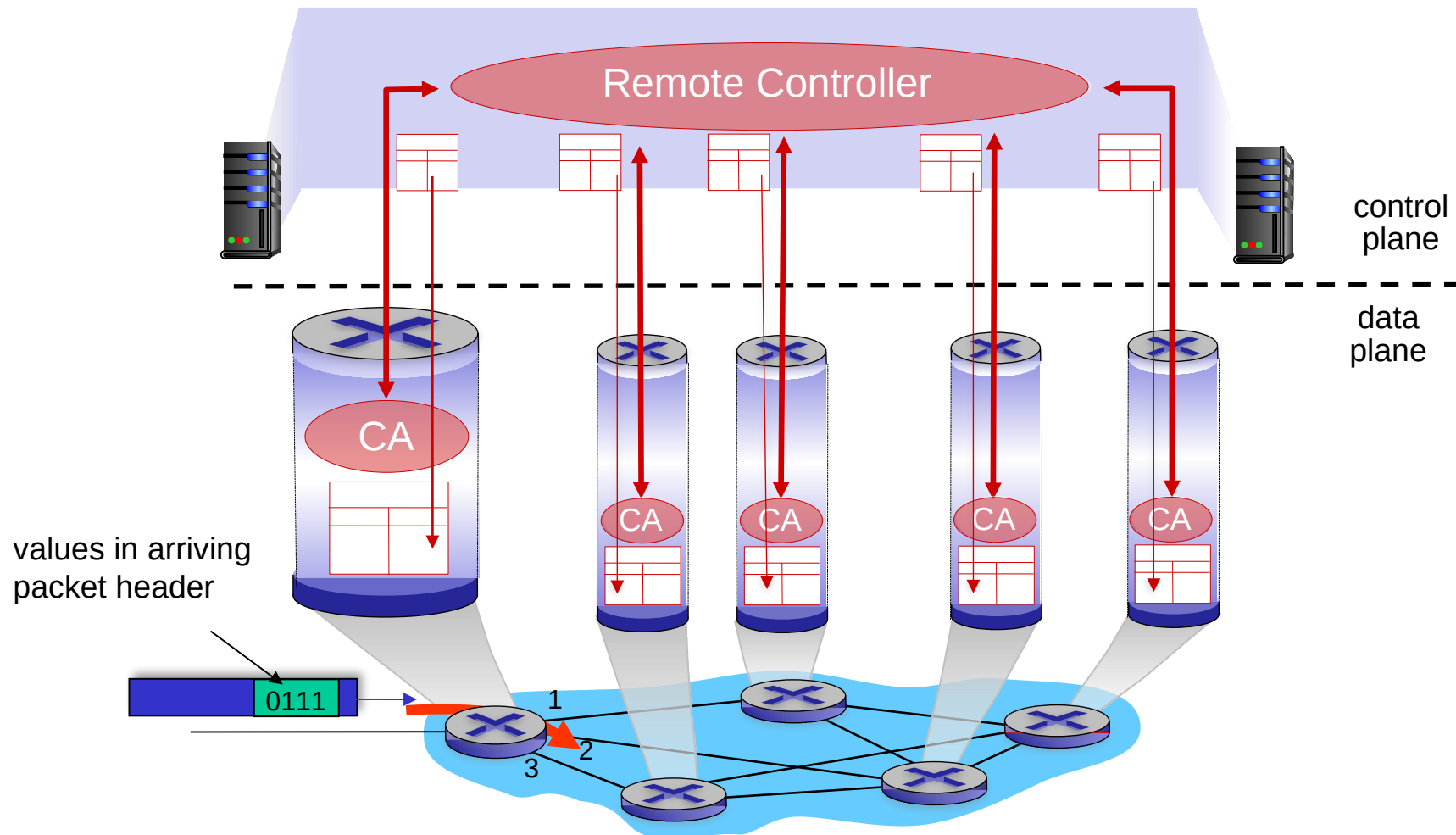
# Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



# Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



# Network layer: “control plane” roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

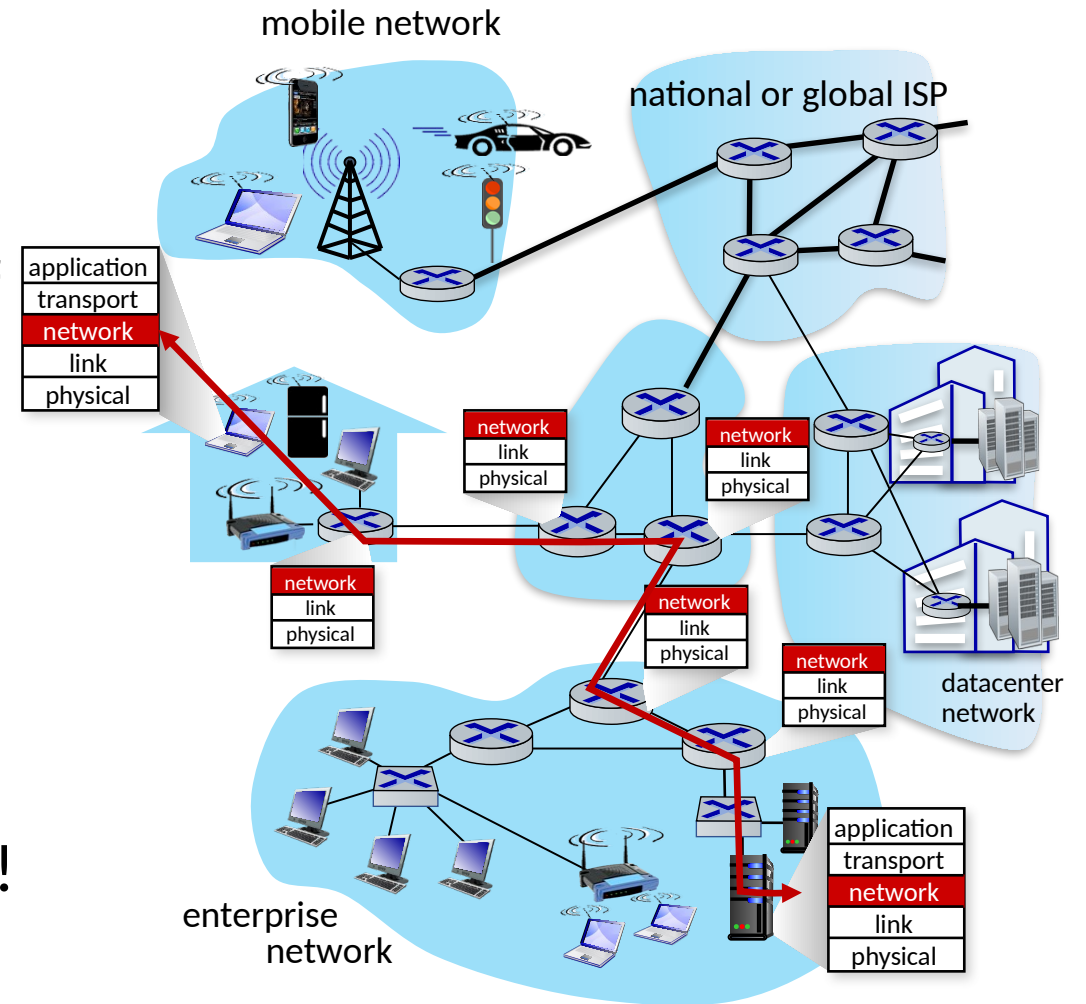


- network management, configuration
  - SNMP
  - NETCONF/YANG

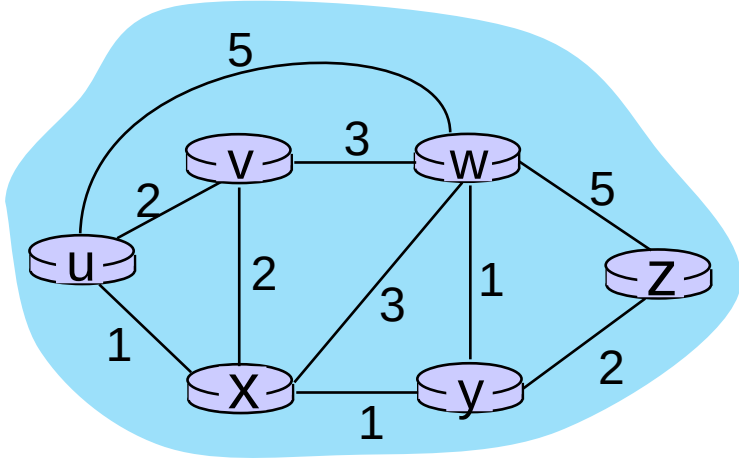
# Routing protocols

**Routing protocol goal:** determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- **path:** sequence of routers packets traverse from given initial source host to final destination host
- **“good”:** least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!



# Graph abstraction: link costs



$c_{a,b}$ : cost of *direct* link connecting  $a$  and  $b$

e.g.,  $c_{w,z} = 5$ ,  $c_{u,z} = \infty$

cost defined by network operator:  
could always be 1, or inversely related  
to bandwidth, or inversely related to  
congestion

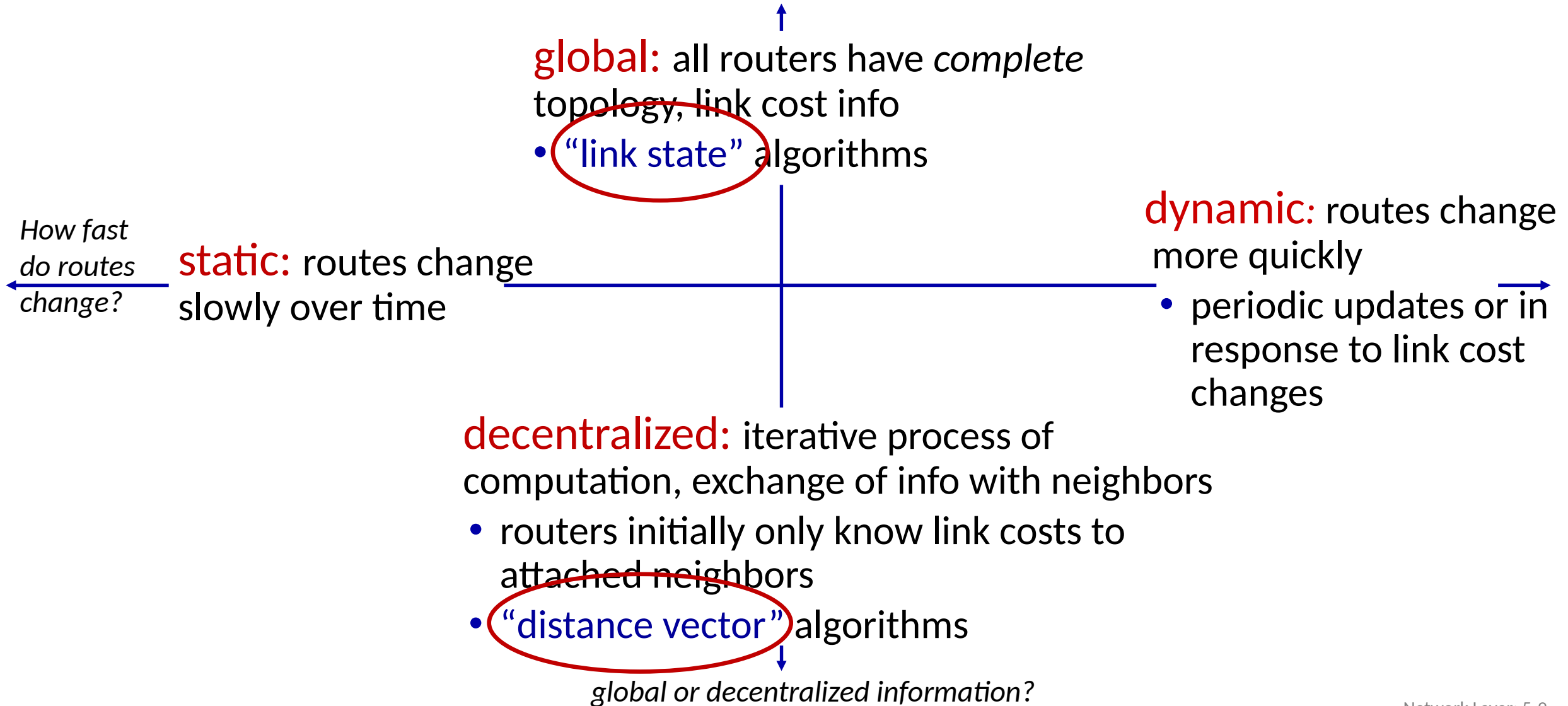
graph:  $G = (N, E)$

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

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



# Routing algorithm classification



# Network layer: “control plane” roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol



- network management, configuration
  - SNMP
  - NETCONF/YANG

# Dijkstra's link-state routing algorithm

- **centralized:** network topology, link costs 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  $y$ ;  $= \infty$  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

1 *Initialization:*

2  $N' = \{u\}$  /\* compute least cost path from u to all other nodes \*/

3 for all nodes v

4 if v adjacent to u /\* u initially knows direct-path-cost only to direct neighbors \*/

5 then  $D(v) = c_{u,v}$  /\* but may not be minimum cost! \*/

6 else  $D(v) = \infty$

7

8 *Loop*

9 find w not in  $N'$  such that  $D(w)$  is a minimum

10 add w to  $N'$

11 update  $D(v)$  for all v adjacent to w and not in  $N'$  :

12  $D(v) = \min ( D(v), D(w) + c_{w,v} )$

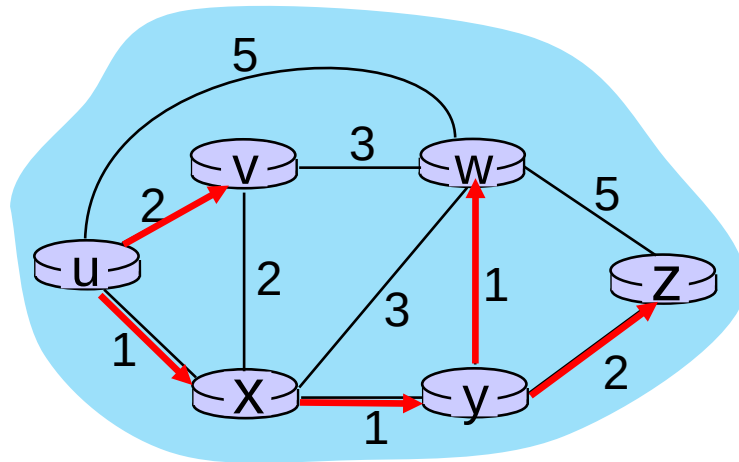
13 /\* new least-path-cost to v is either old least-cost-path to v or known

14 least-cost-path to w plus direct-cost from w to v \*/

15 *until all nodes in  $N'$*

# Dijkstra's algorithm: an 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	$\infty$	$\infty$
1	ux	2, u	4, x		2, x	$\infty$
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  $D(a) = c_{u,a}$

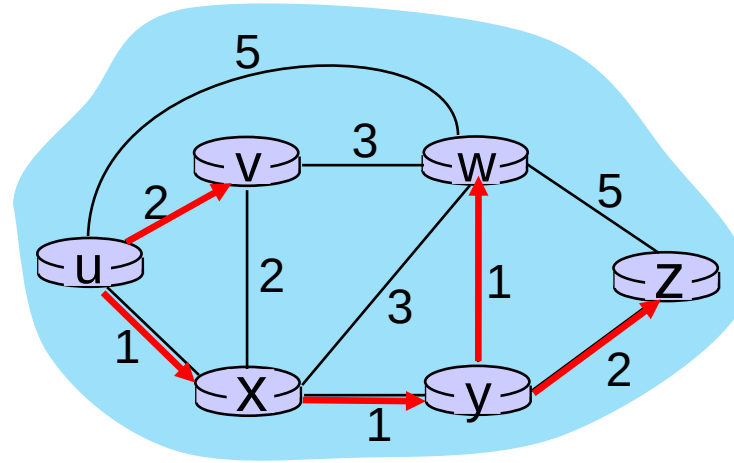
find  $a$  not in  $N'$  such that  $D(a)$  is a minimum

add  $a$  to  $N'$

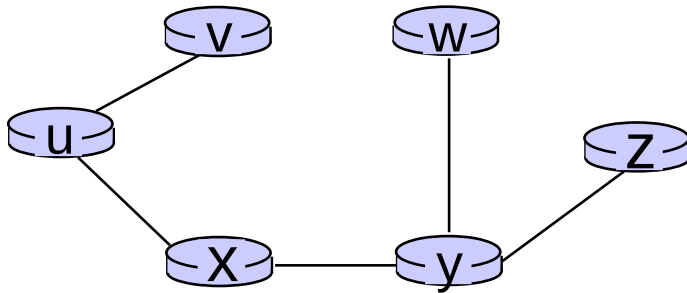
update  $D(b)$  for all  $b$  adjacent to  $a$  and not in  $N'$ :

$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

# Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

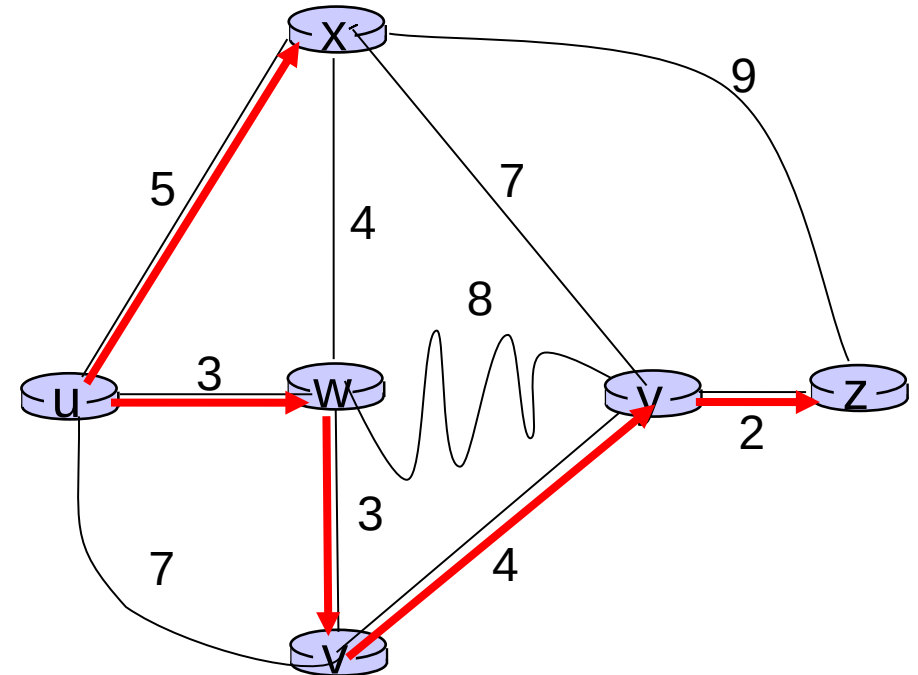
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

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	$\infty$	$\infty$
1	uw	6, w		5, u	11, w	$\infty$
2	uwvx	6, w			11, w	14, x
3	uwvx				10, v	14, x
4	uwxvy					12, y
5	uwxvyz					



## notes:

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

# 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:  $O(n \log n)$

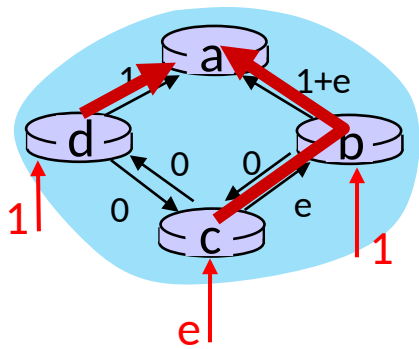
message complexity:

- 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:  $O(n^2)$

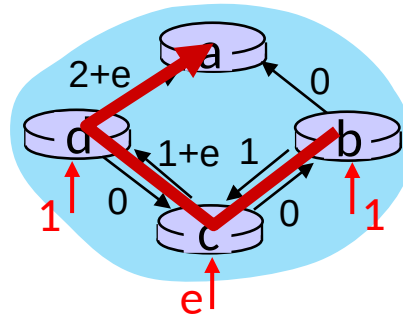


# Dijkstra's algorithm: oscillations possible

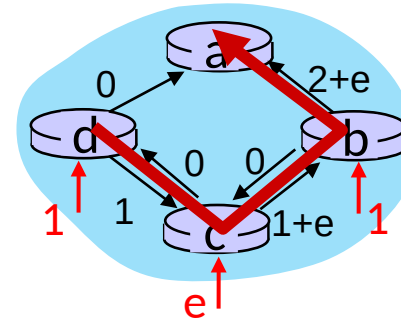
- when link costs depend on traffic volume, **route oscillations** possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, b with rates 1,  $e$  ( $<1$ ), 1
  - link costs are directional, and volume-dependent



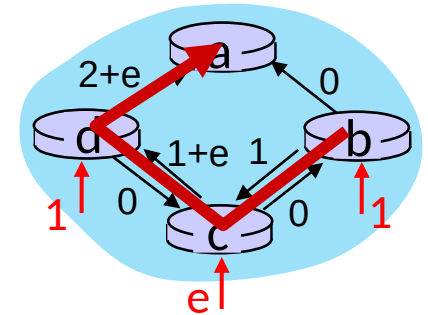
initially



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

# Network layer: “control plane” roadmap

- introduction
- routing protocols
  - link state
  - **distance vector**
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol



- network management, configuration
  - SNMP
  - NETCONF/YANG

# 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:

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

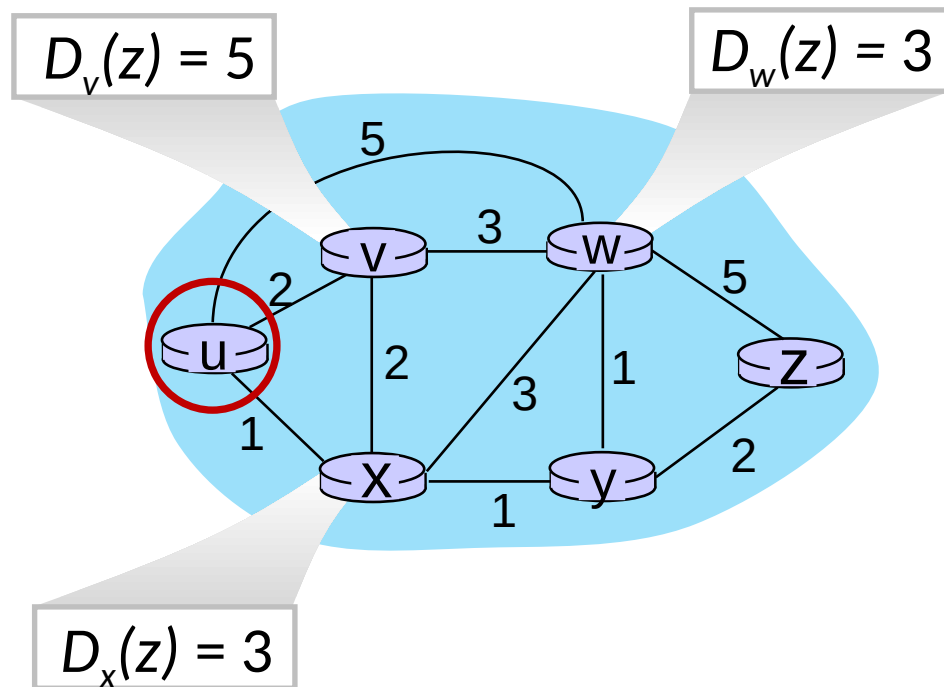
$D_v(y)$ :  $v$ 's estimated least-cost-path cost to  $y$

$\min$  taken over all neighbors  $v$  of  $x$

$c_{x,v}$ : 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:

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

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

# Distance vector algorithm

## key idea:

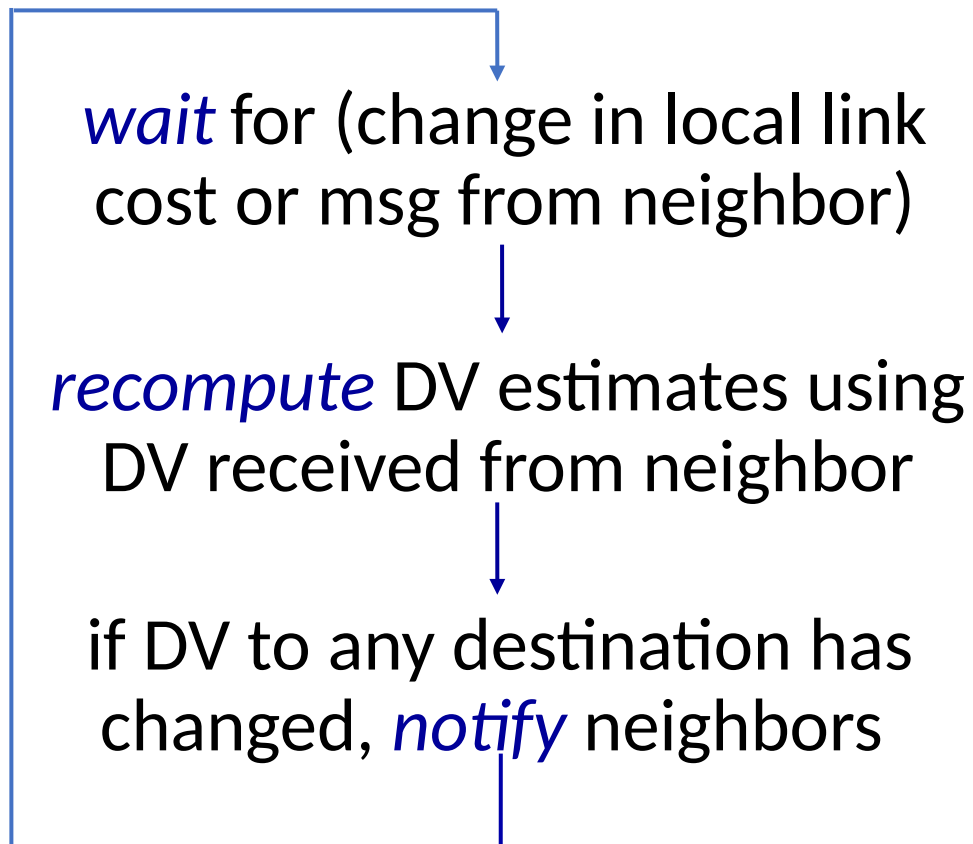
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when  $x$  receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

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

# Distance vector algorithm:

each node:



**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!

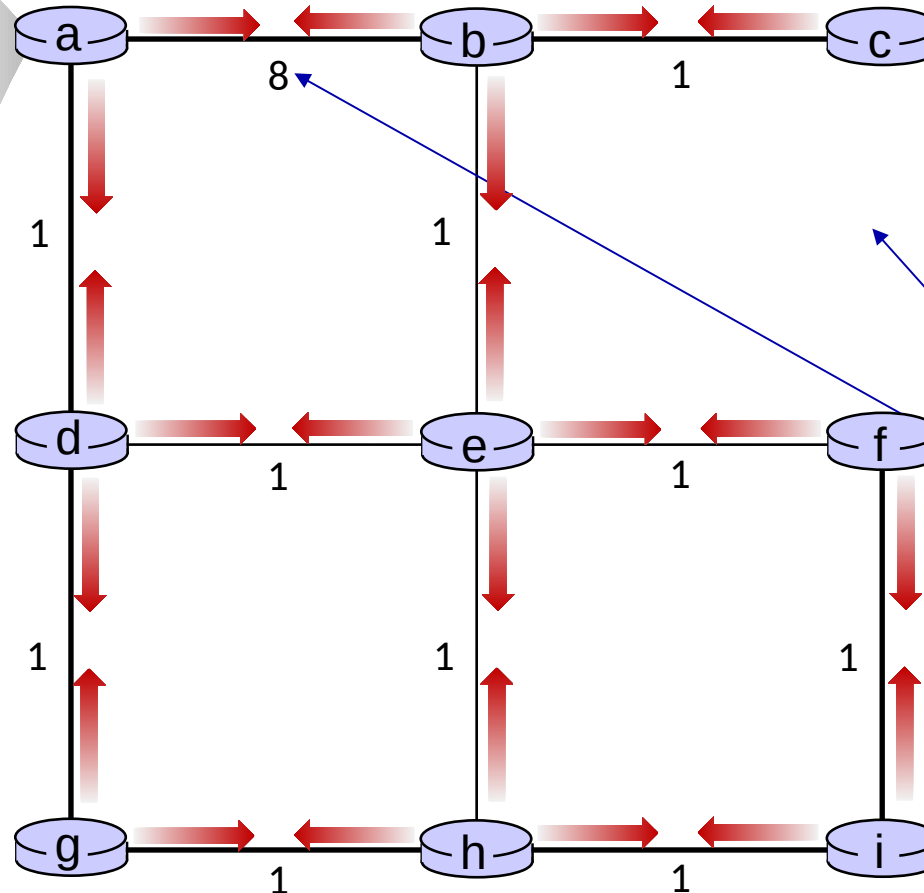
# Distance vector: example



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
$D_a(e)$	$\infty$
$D_a(f)$	$\infty$
$D_a(g)$	$\infty$
$D_a(h)$	$\infty$
$D_a(i)$	$\infty$



A few asymmetries:  
■ missing link  
■ larger cost

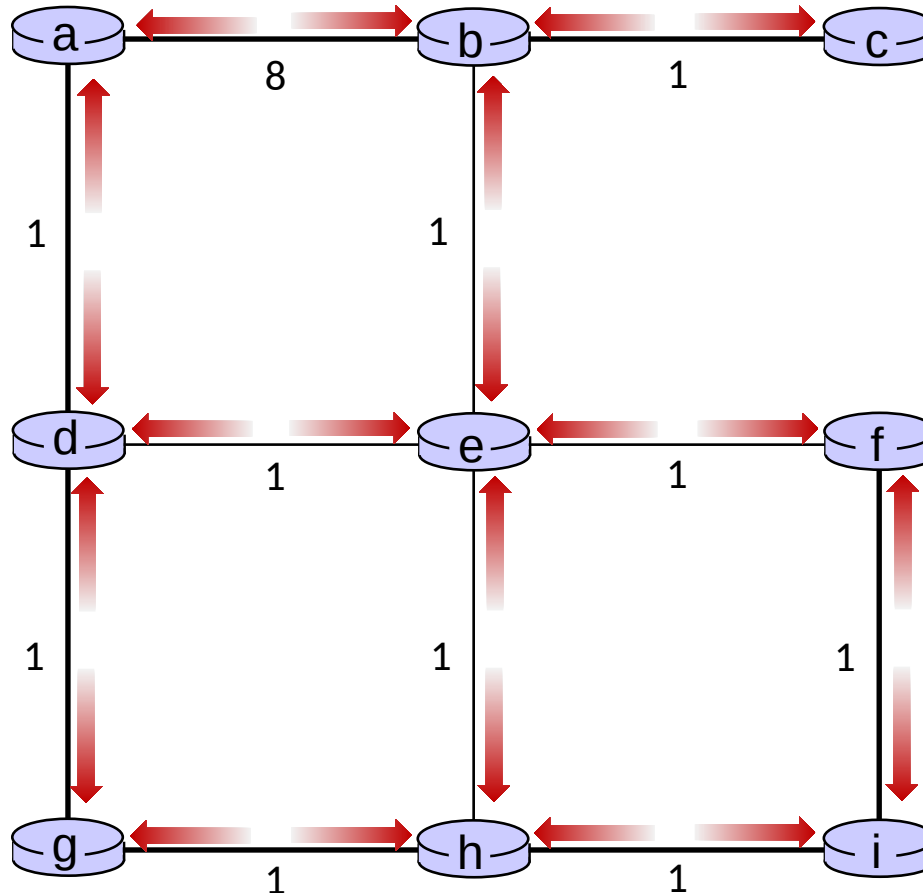
# Distance vector example: iteration



$t=1$

All nodes:

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





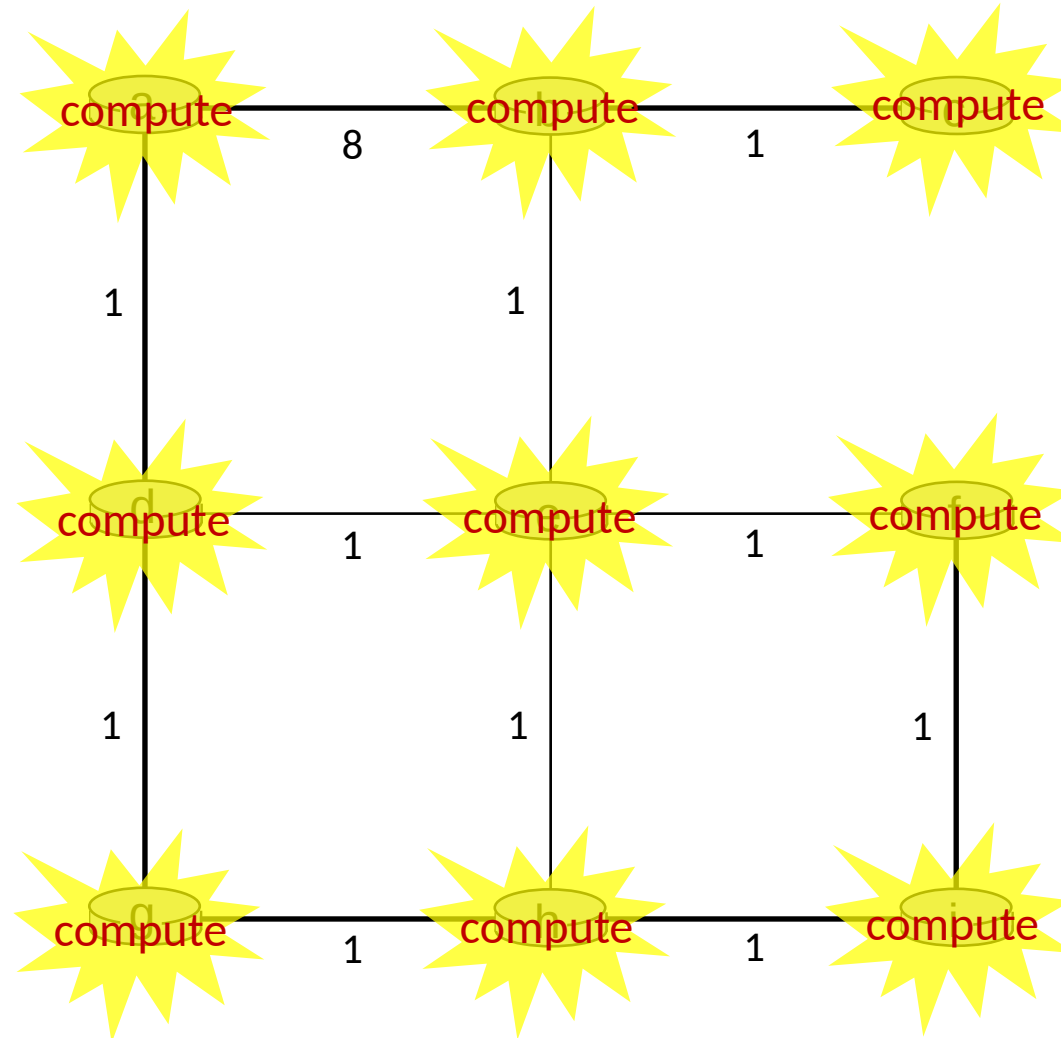
# Distance vector example: iteration



$t=1$

All nodes:

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



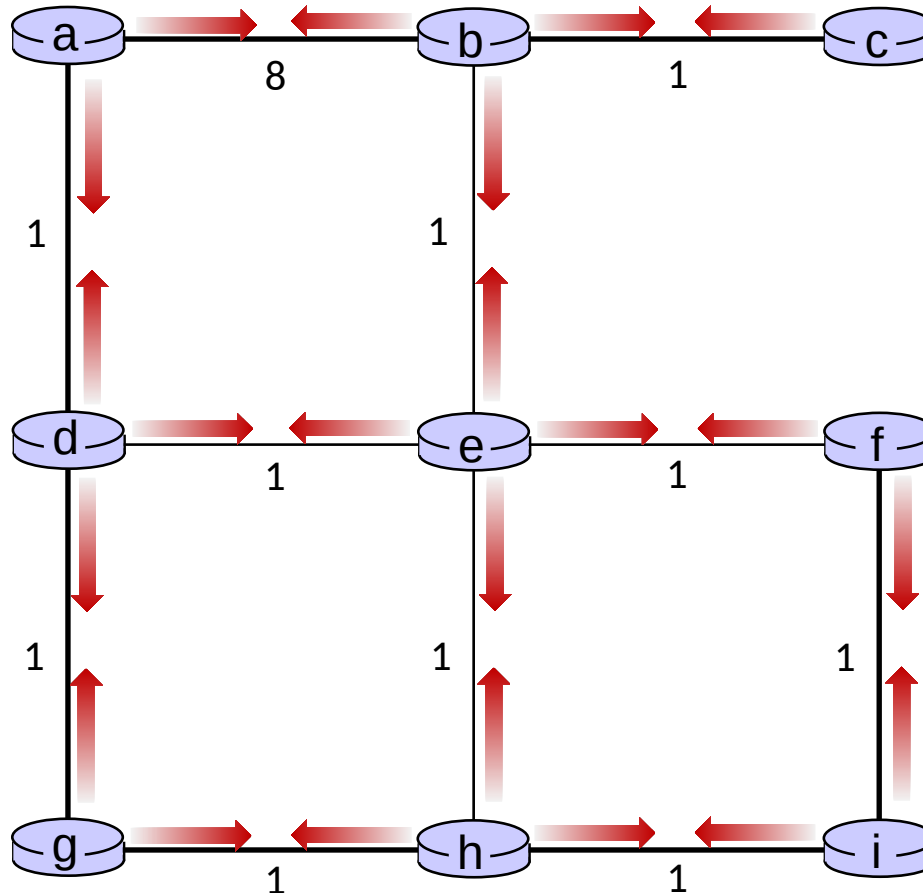
# Distance vector example: iteration



$t=1$

All nodes:

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



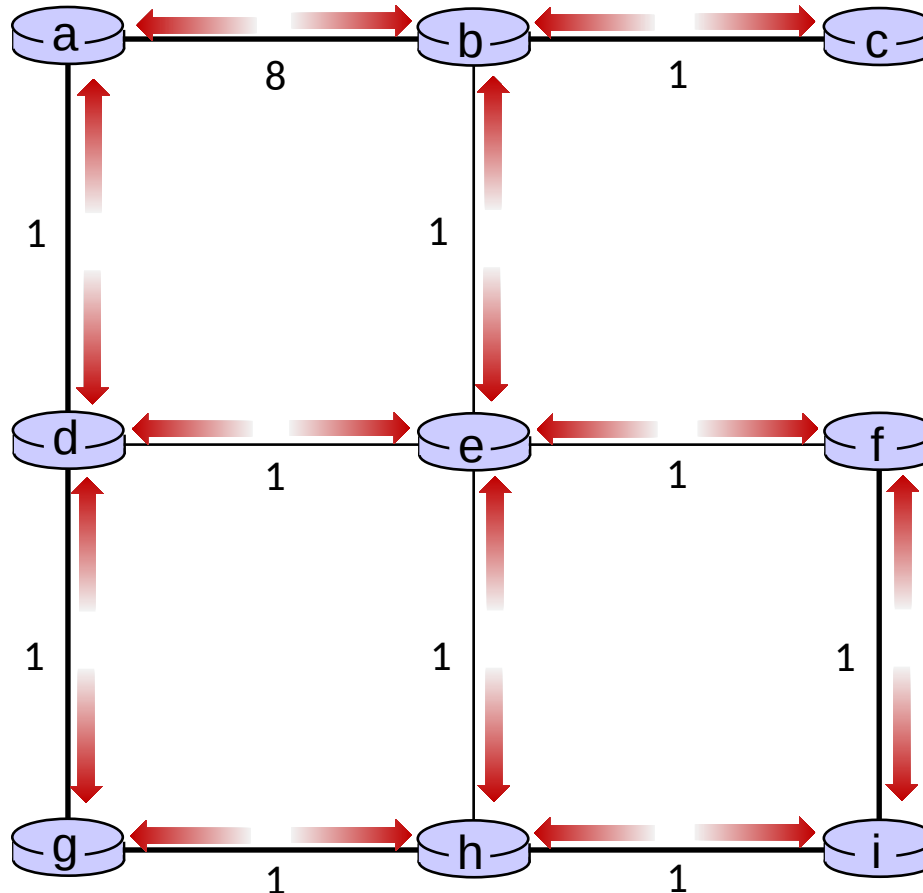
# Distance vector example: iteration



t=2

All nodes:

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



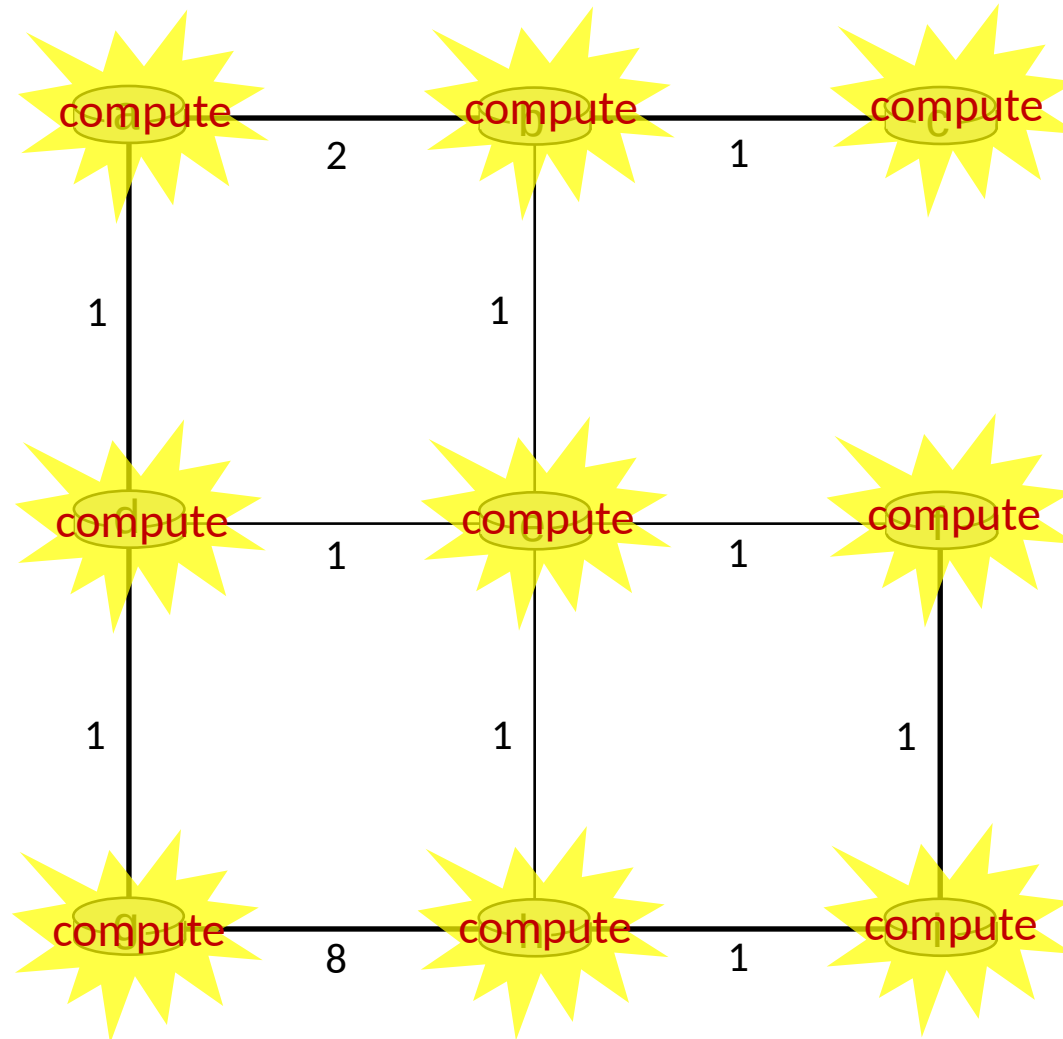
# Distance vector example: iteration



t=2

All nodes:

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



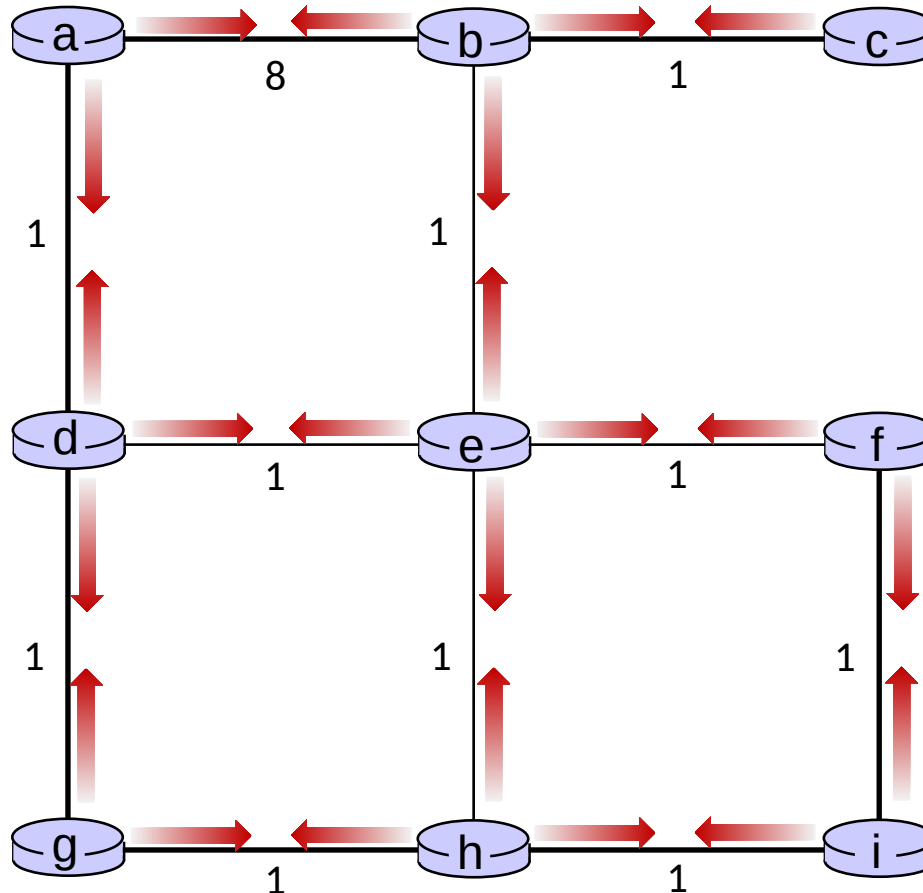
# Distance vector example: iteration



$t=2$

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

## 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:

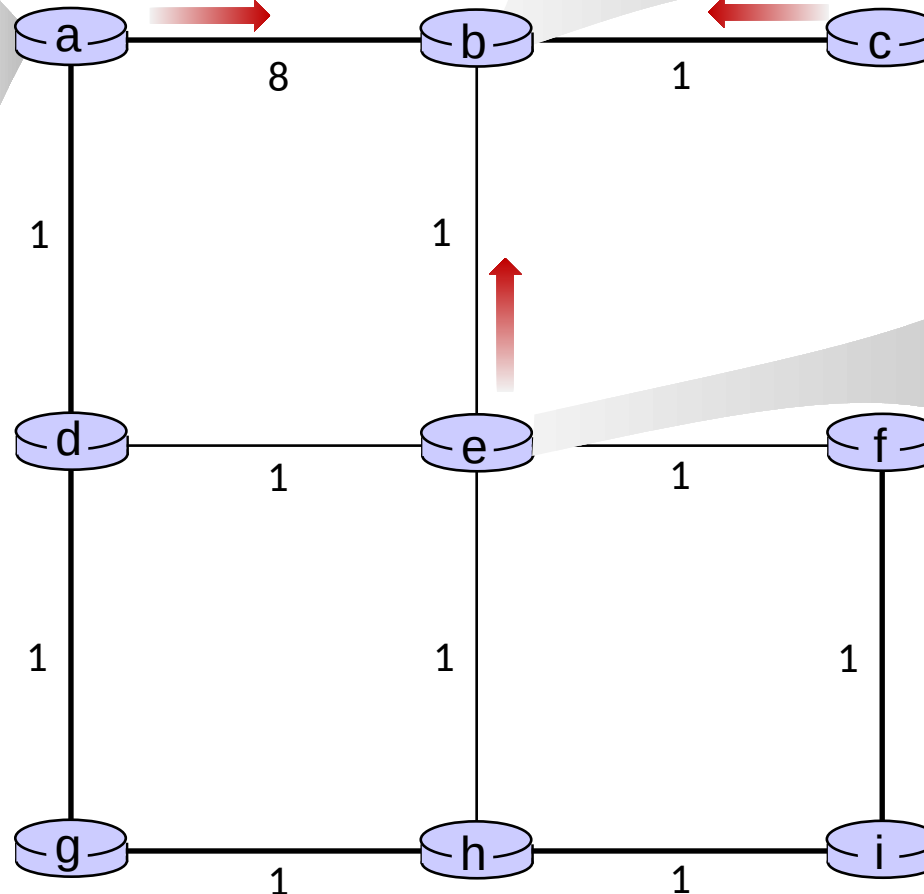
$D_b(a)=8$     $D_b(f)=\infty$   
 $D_b(c)=1$     $D_b(g)=\infty$   
 $D_b(d)=\infty$     $D_b(h)=\infty$   
 $D_b(e)=1$     $D_b(i)=\infty$

## DV in c:

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

## DV in e:

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



# Distance vector example: computation



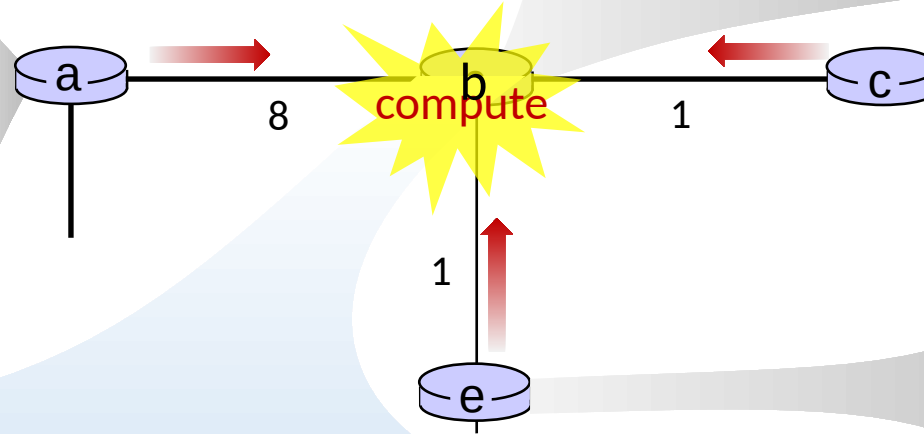
$t=1$

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

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

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



$$\begin{aligned}
 D_b(a) &= \min\{c_{b,a}+D_a(a), c_{b,c}+D_c(a), c_{b,e}+D_e(a)\} = \min\{8, \infty, \infty\} = 8 \\
 D_b(c) &= \min\{c_{b,a}+D_a(c), c_{b,c}+D_c(c), c_{b,e}+D_e(c)\} = \min\{\infty, 1, \infty\} = 1 \\
 D_b(d) &= \min\{c_{b,a}+D_a(d), c_{b,c}+D_c(d), c_{b,e}+D_e(d)\} = \min\{9, 2, \infty\} = 2 \\
 D_b(e) &= \min\{c_{b,a}+D_a(e), c_{b,c}+D_c(e), c_{b,e}+D_e(e)\} = \min\{\infty, \infty, 1\} = 1 \\
 D_b(f) &= \min\{c_{b,a}+D_a(f), c_{b,c}+D_c(f), c_{b,e}+D_e(f)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(g) &= \min\{c_{b,a}+D_a(g), c_{b,c}+D_c(g), c_{b,e}+D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty \\
 D_b(h) &= \min\{c_{b,a}+D_a(h), c_{b,c}+D_c(h), c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(i) &= \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty
 \end{aligned}$$

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

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



# Distance vector example

## computation



$t=1$

- c receives DVs from b

### 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:

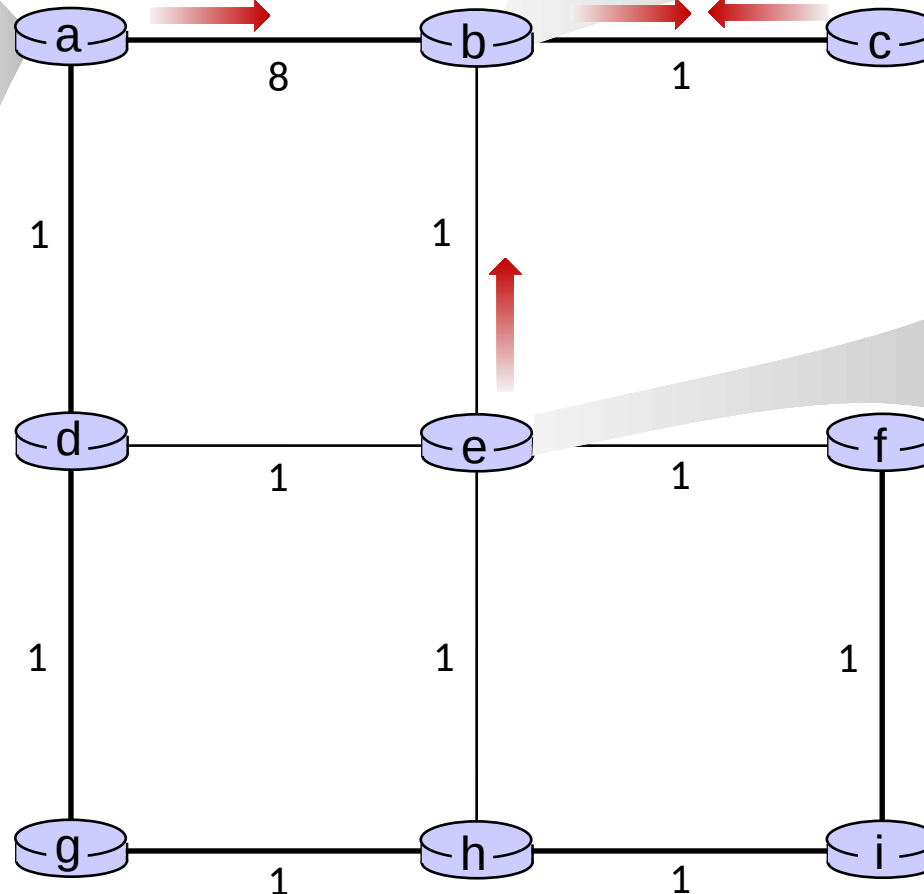
$D_b(a)=8$     $D_b(f)=\infty$   
 $D_b(c)=1$     $D_b(g)=\infty$   
 $D_b(d)=\infty$     $D_b(h)=\infty$   
 $D_b(e)=1$     $D_b(i)=\infty$

### DV in c:

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

### DV in e:

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



# Distance vector example

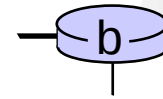
## computation



$t=1$

- c receives DVs from b computes:

$$\begin{aligned} D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\ D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\ D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\ D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\ D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\ D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\ D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\ D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty \end{aligned}$$



1

compute

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:

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

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/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

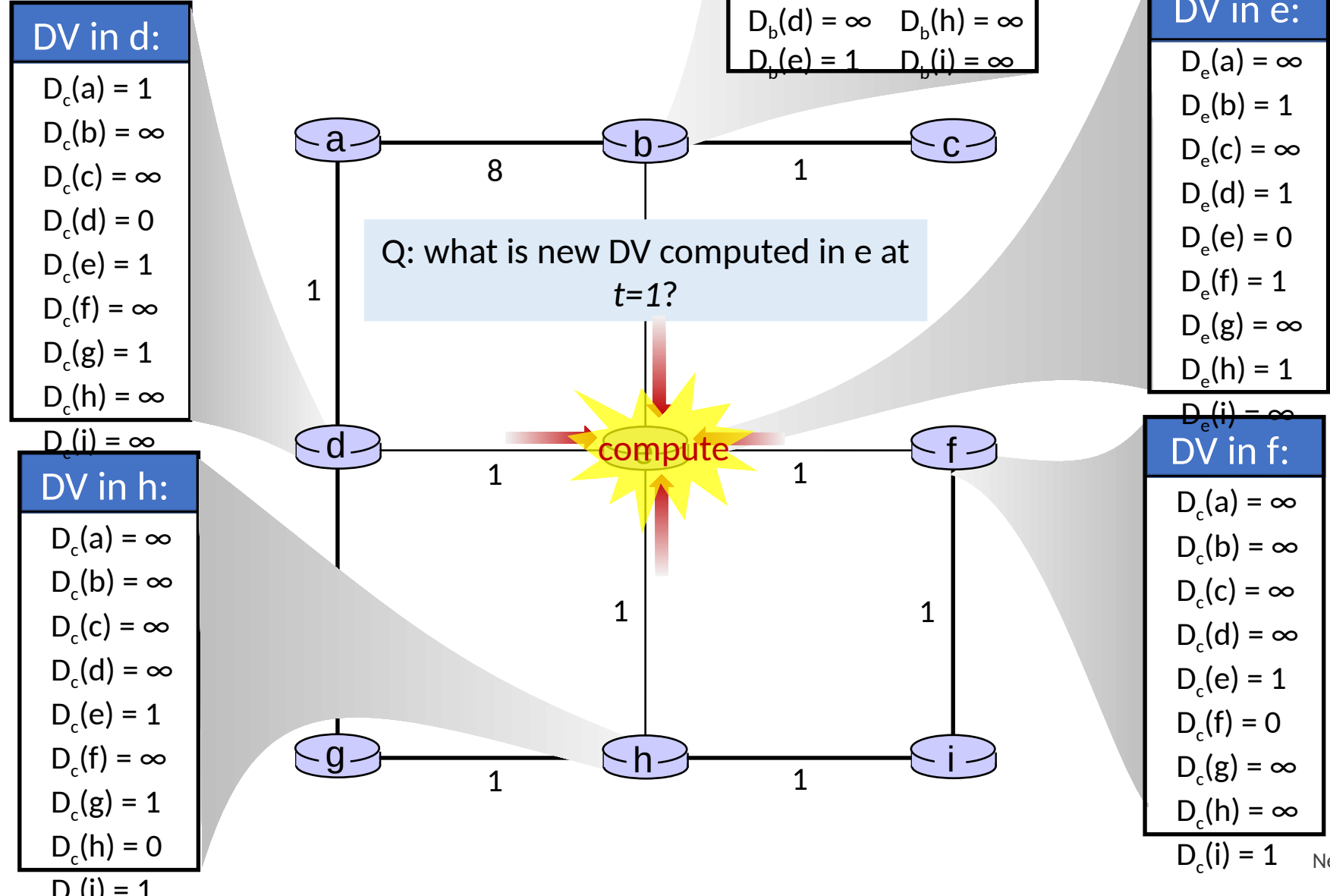
# Distance vector example

## computation








$t=1$

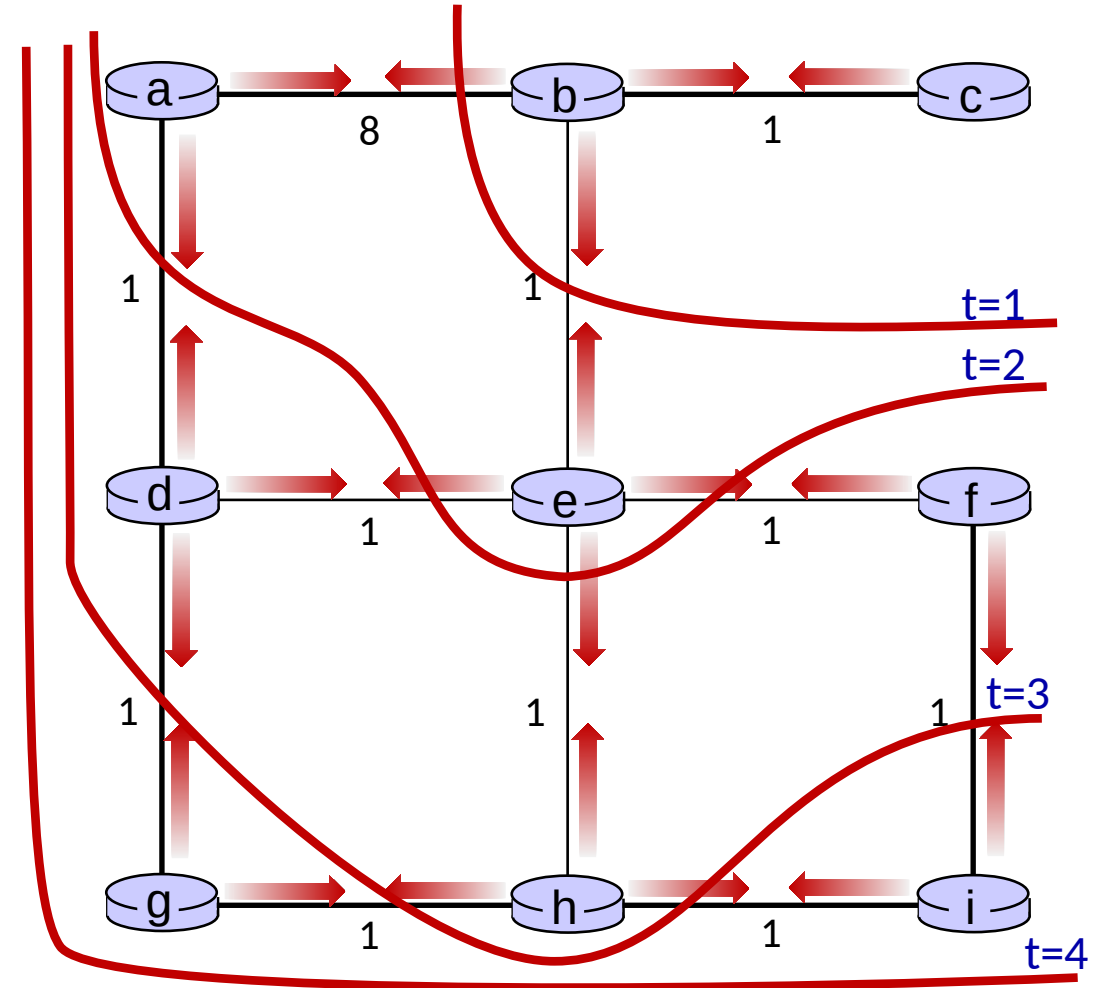
- e receives DVs from b, d, f, h



# Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

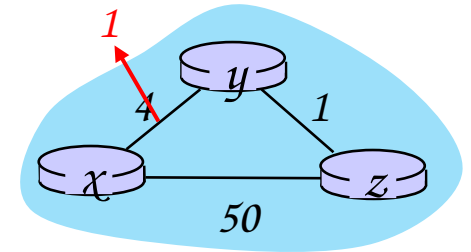
-   $t=0$  c's state at  $t=0$  is at c only
-   $t=1$  c's state at  $t=0$  has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
-   $t=2$  c's state at  $t=0$  may now influence distance vector computations up to **2** hops away, i.e., at b and now at 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: 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.

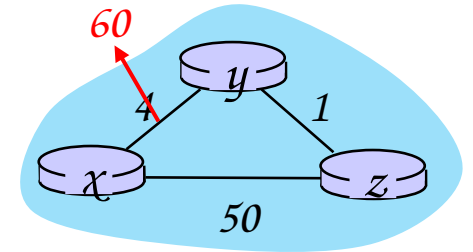
$t_1$ : z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

$t_2$ : 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.

# Distance vector: link cost changes

## link cost changes:

- node detects local link cost change
- “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.
  - ...
- see text for solutions. *Distributed algorithms are tricky!*



# Comparison of LS and DV algorithms

## message complexity

LS:  $n$  routers,  $O(n^2)$  messages sent

DV: exchange between neighbors;  
convergence time varies

## speed of convergence

LS:  $O(n^2)$  algorithm,  $O(n^2)$  messages

- may have oscillations

DV: convergence time varies

- may have routing loops
- count-to-infinity problem

robustness: what happens if router malfunctions, or is compromised?

LS:

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

DV:

- DV router can advertise incorrect *path* cost (“I have a *really* low cost path to everywhere”): black-holing
- each router’s table used by others: error propagate thru network