



الجامعة اللبنانية  
UNIVERSITE LIBANAISE

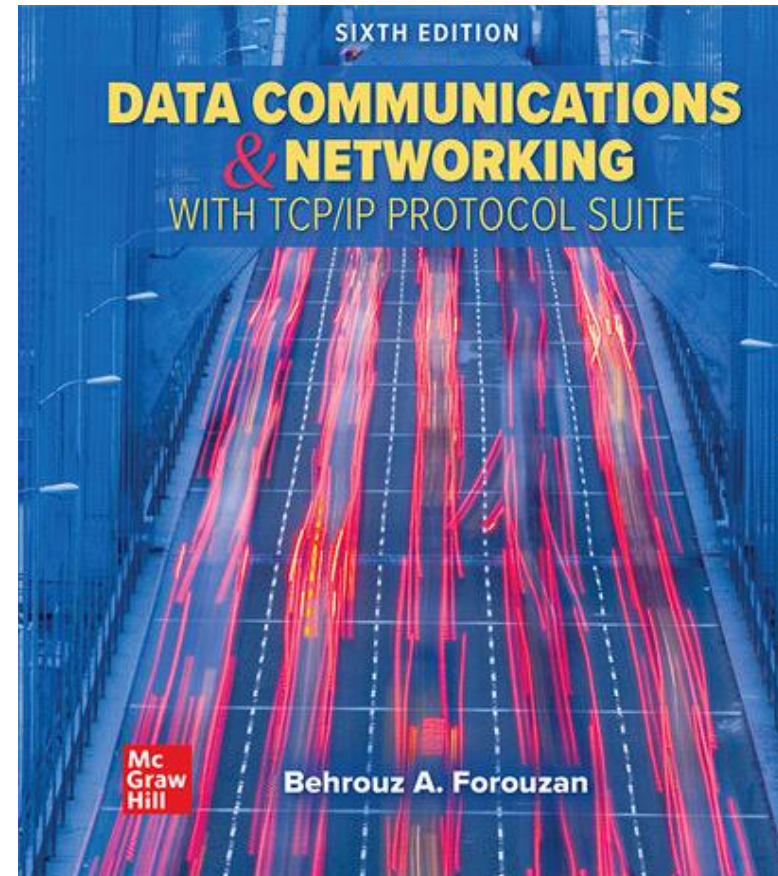
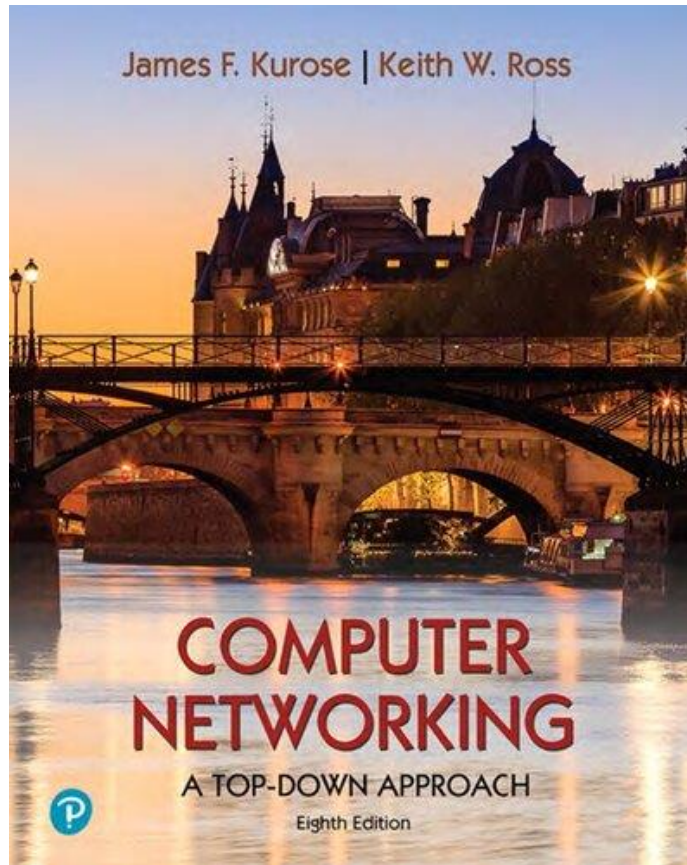


I3304

# Network administration and security

Ahmad Fadlallah

# Reference Textbooks



# Outline



- Introduction
  - ⊙ Introduction to the course
  - ⊙ Recall Network Basics (I2208)
- Network Layer
  - ⊙ Static Routing
  - ⊙ **Dynamic Routing**
    - Dynamic Routing Algorithm
    - Dynamic Routing Protocols
  - ⊙ NAT (Network Address Translation)
- Transport Layer
  - ⊙ Function of the transport layer
  - ⊙ UDP Protocol
  - ⊙ TCP Protocol
    - Connection management
    - Flow control
    - Congestion control
- Application Layer
  - HTTP protocol
  - FTP protocol
  - Mail protocols
  - DNS
- Introduction to Security
  - Security services
  - Cryptography
  - Digital Signature
  - Principle of network security protocols

# References



- The slides are based on the:
  - ⦿ Cisco Networking Academy Program, Routing and Switching Essentials v6.0, Chapter 1: Routing Concepts
  - ⦿ Jim Kurose, Keith Ross Slides for the Computer Networking: A Top-Down Approach, 8th edition, Pearson, 2020

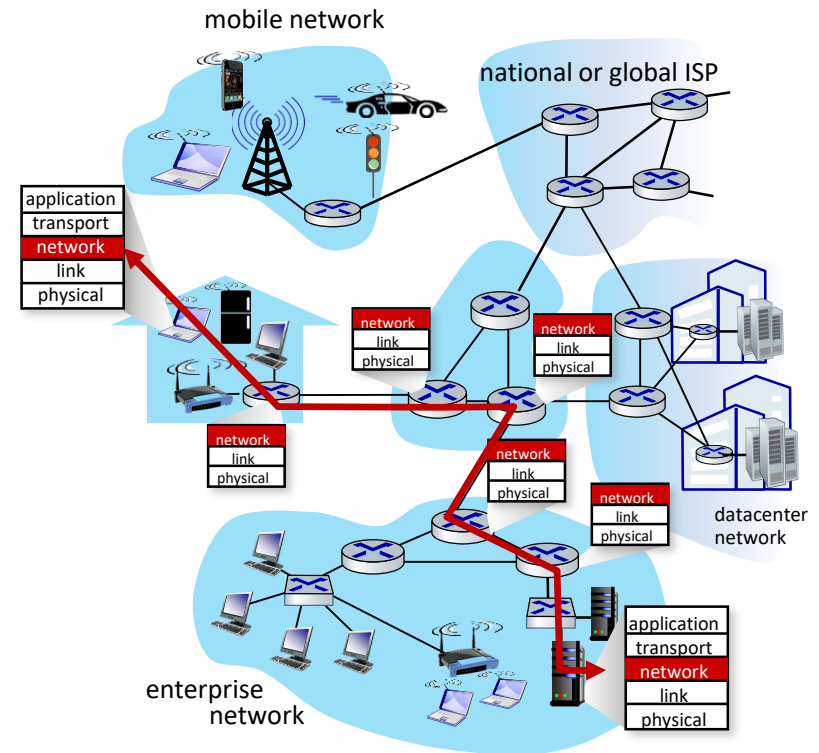


# Network Layer Dynamic Routing

# 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”, ...



# 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*

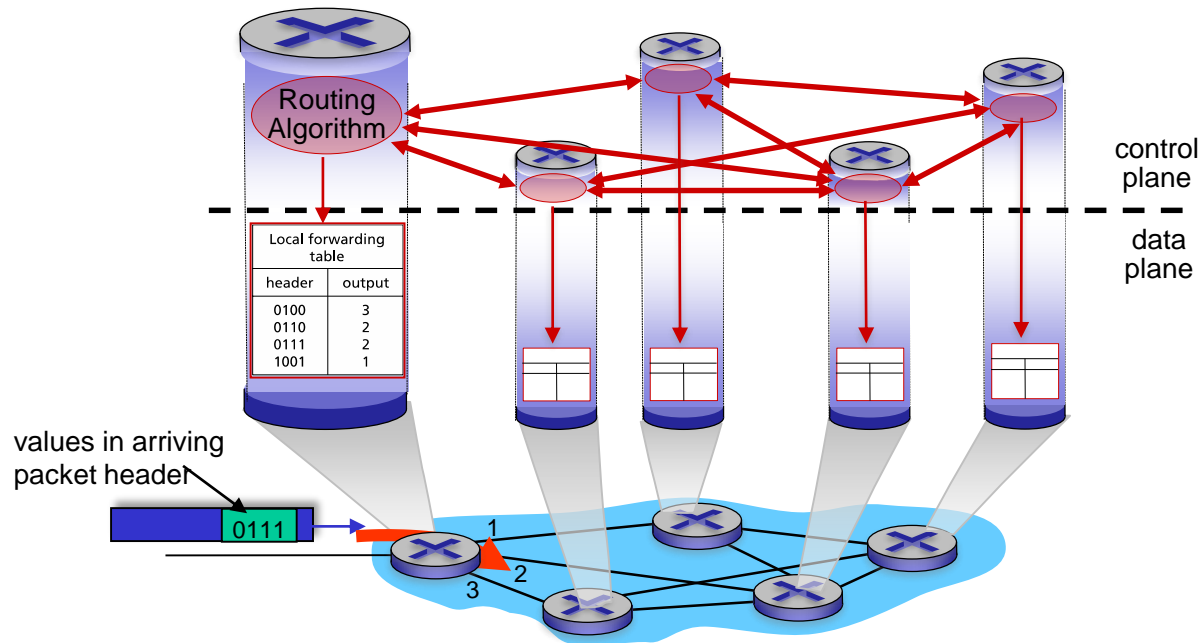
## Two approaches to structuring network control plane:

- 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

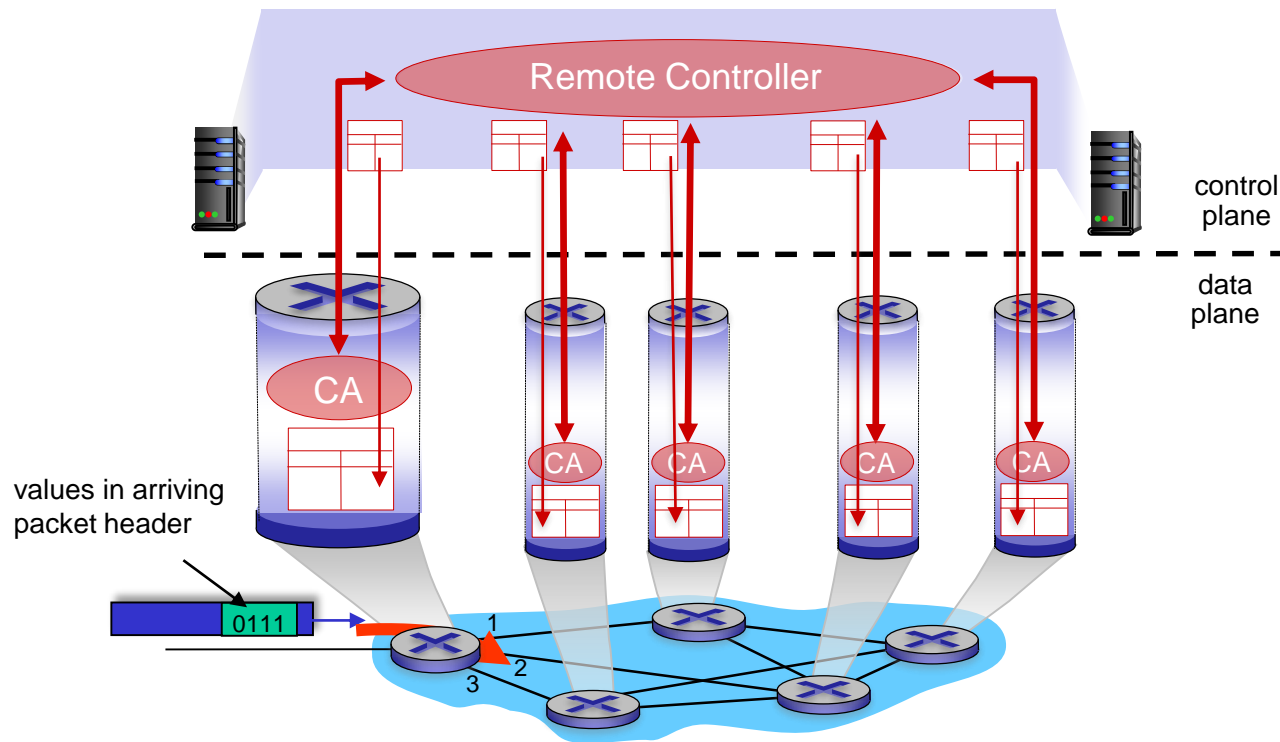




# Software-Defined Networking (SDN)



Remote controller computes, installs forwarding tables in routers





# Purpose of Dynamic Routing Protocols

- **Routing Protocols** are used to facilitate the **exchange** of routing information between routers.
- The purpose of dynamic routing protocols includes:
  - ⊙ **Discovery of remote networks**
  - ⊙ Maintaining **up-to-date routing information**
  - ⊙ Choosing **the best path** to destination networks
  - ⊙ Ability to **find a new best path** if the current path is no longer available



# Static Routing Scorecard

## Static Routing Advantages and Disadvantages

Advantages	Disadvantages
Easy to implement in a small network.	Suitable only for simple topologies or for special purposes such as a default static route. Configuration complexity increases dramatically as network grows.
Very secure. No advertisements are sent as compared to dynamic routing protocols.	
Route to destination is always the same.	Manual intervention required to re-route traffic.
No routing algorithm or update mechanism required; therefore, extra resources (CPU or RAM) are not required.	

# Dynamic Routing Scorecard



## Dynamic Routing Advantages and Disadvantages

Advantages	Disadvantages
Suitable in all topologies where multiple routers are required.	Can be more complex to implement.
Generally independent of the network size.	Less secure. Additional configuration settings are required to secure.
Automatically adapts topology to reroute traffic if possible.	Route depends on the current topology.
	Requires additional CPU, RAM, and link bandwidth.



# Dynamic Routing Protocols Components

## Data structures

- Routing protocols typically use **tables or databases** for its operations.
- This information is kept in RAM.

## Routing protocol messages

- Routing protocols use various types of messages to *discover neighboring routers, exchange routing information*, and other tasks to **learn and maintain accurate information about the network**.

## Algorithm

- Routing protocols use **algorithms** for facilitating routing information for best path determination.

# Dynamic Routing Protocol Operation



The router sends and receives routing messages on its interfaces.

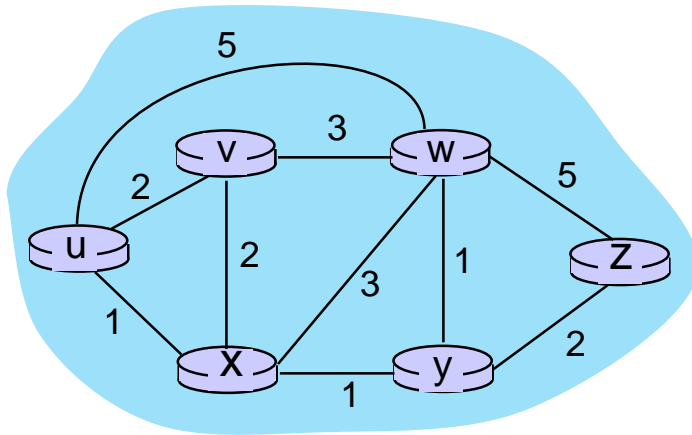
The router shares routing messages and routing information with other routers that are using the same routing protocol.

Routers exchange routing information to learn about remote networks.

When a router detects a topology change the routing protocol can advertise this change to other routers.



# Graph abstraction: link costs



graph:  $G = (N, E)$

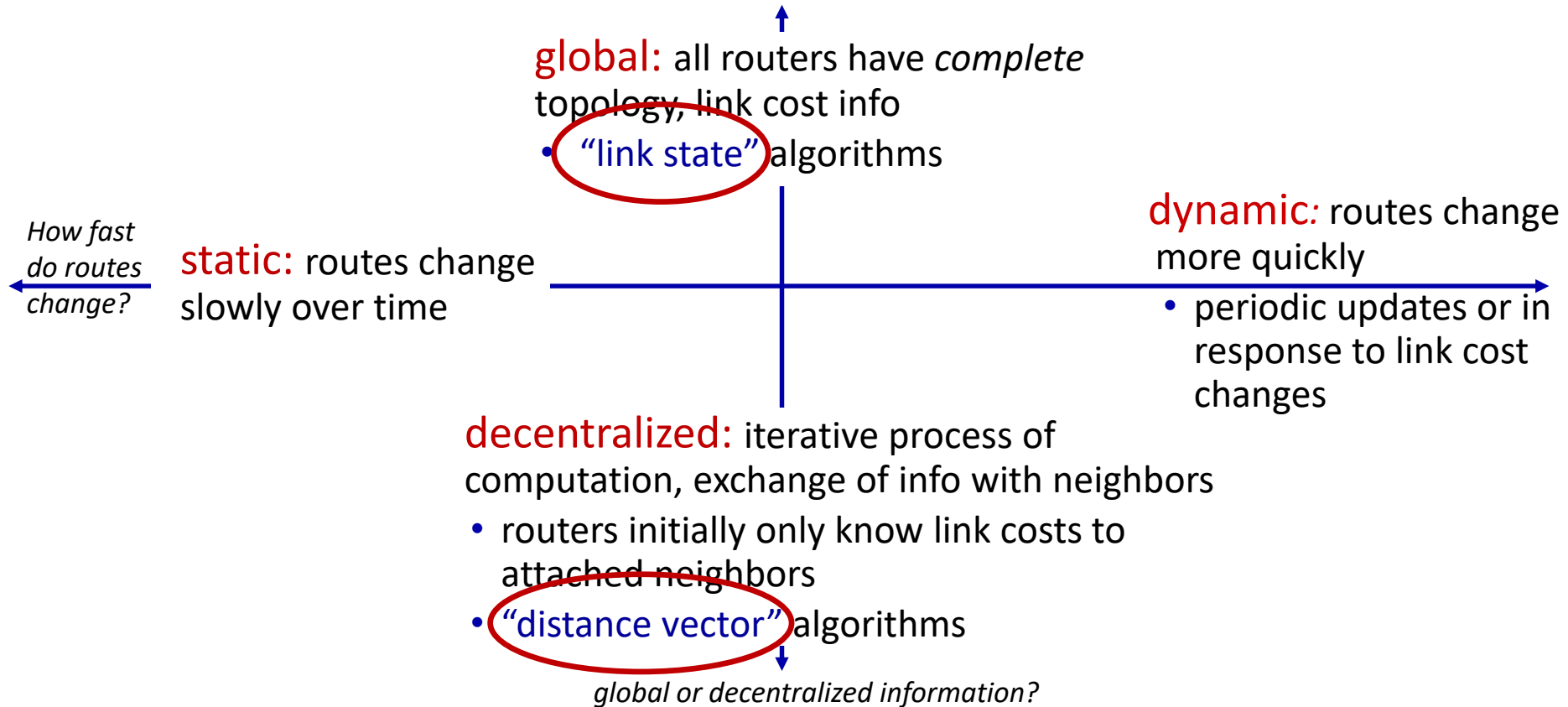
$c_{a,b}$ : cost of *direct* link connecting  $a$  and  $b$   
e.g.,  $c_{w,z} = 5$ ,  $c_{u,z} = \infty$

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

$N$ : 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







# Dynamic Routing Algorithms

## Link-State Algorithms

# 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*

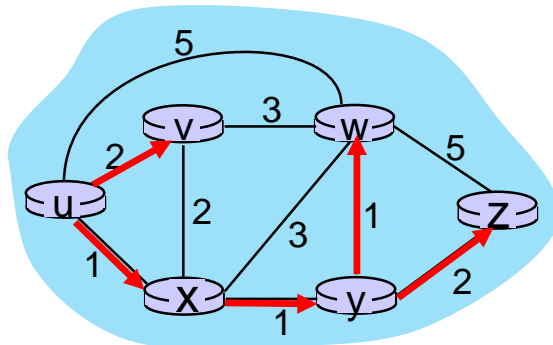
```
9  find  $w$  not in  $N'$  such that  $D(w)$  is a minimum
10 add  $w$  to  $N'$ 
11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :
12    $D(v) = \min ( D(v), D(w) + c_{w,v} )$ 
13   /* new least-path-cost to  $v$  is either old least-cost-path to  $v$  or known
14   least-cost-path 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	u, x	2, u	4, x		2, x	$\infty$
2	u, x, y	2, u	3, y			4, y
3	u, x, y, v		3, y			4, y
4	u, x, y, v, w					4, y
5	u, x, y, v, w, z					



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

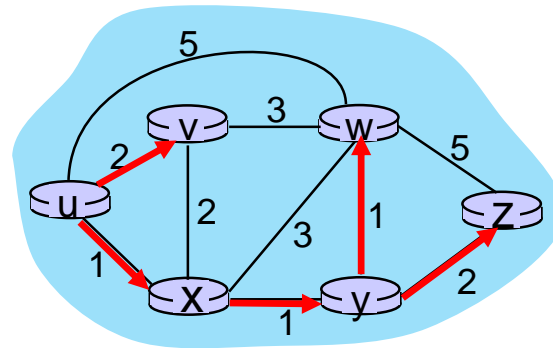
find  $a$  not in  $N'$  such that  $D(a)$  is a minimum

add  $a$  to  $N'$

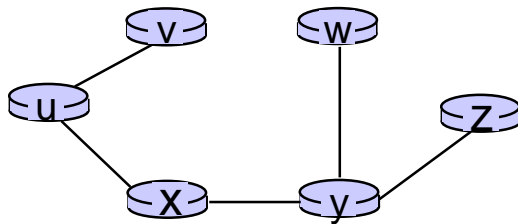
update  $D(b)$  for all  $b$  adjacent to  $a$  and not in  $N'$ :

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

# 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)
x	(u,x)

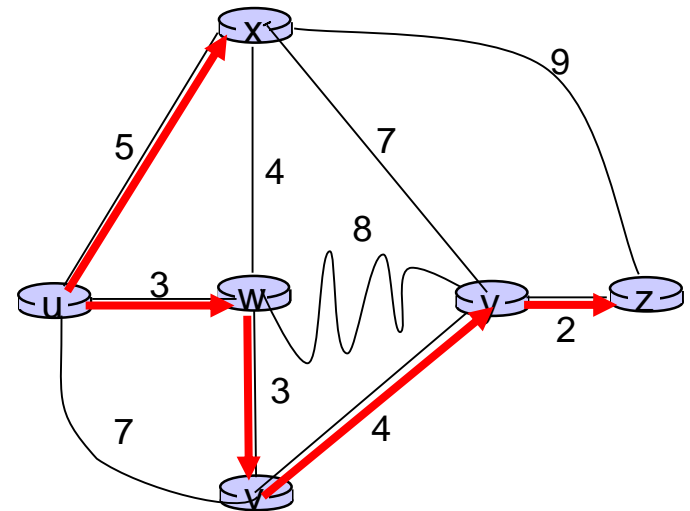
route from *u* to *v* directly

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		u	11, w	14, x
3	uwxv				10, v	14, x
4	uwxvy					12, y
5	uwxvyz					



## notes:

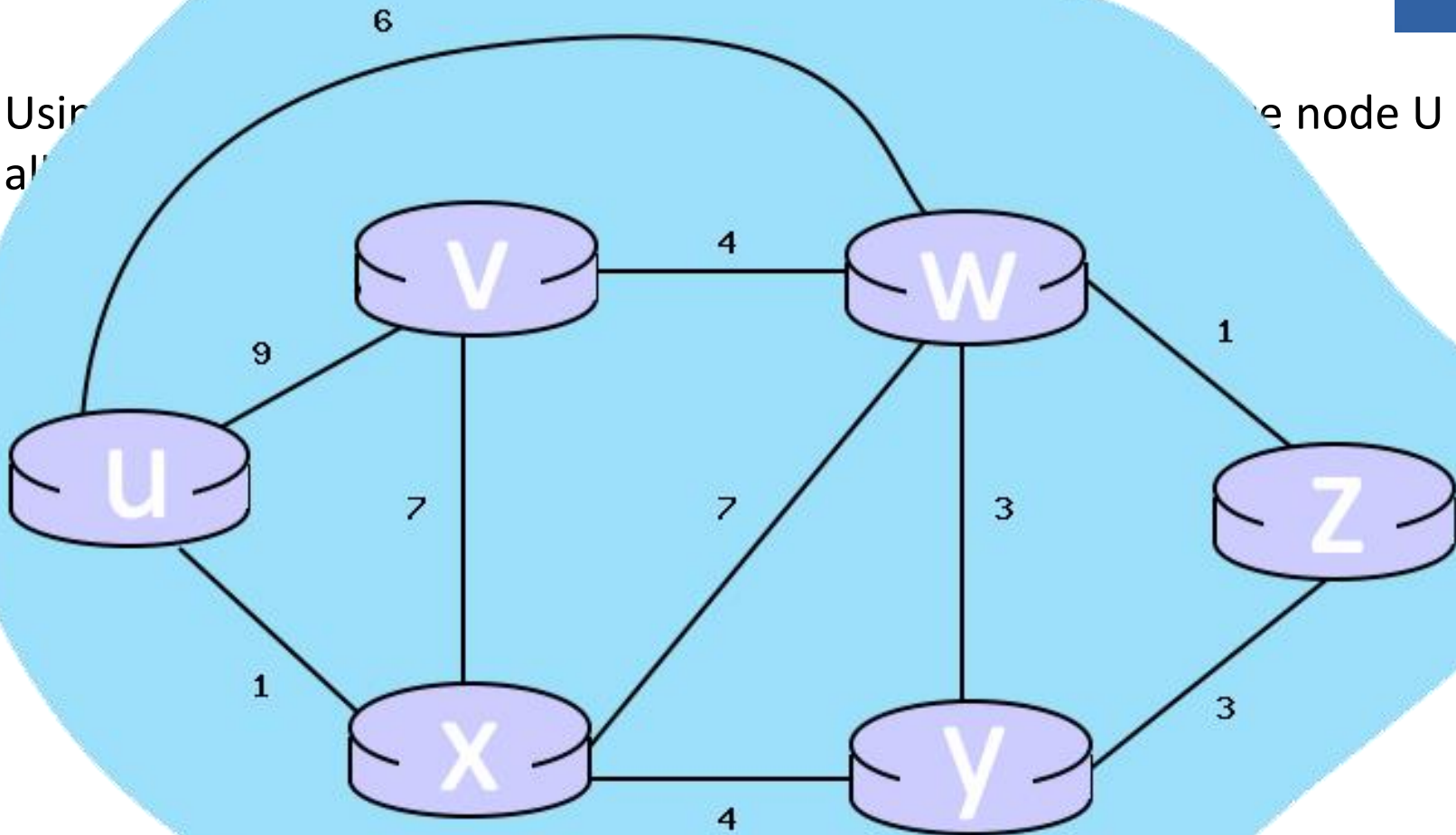
- construct least-cost-path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



# Exercise

- Using all

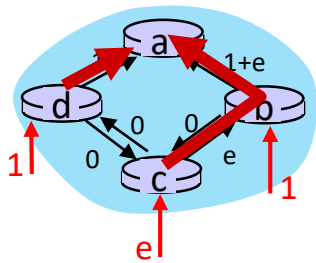
the node U to



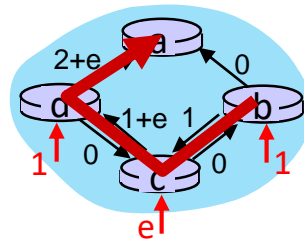
# Dijkstra's algorithm: oscillations possible



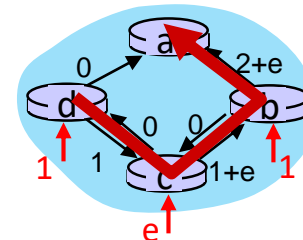
- When link costs depend on traffic volume, **route oscillations** possible
- Sample scenario:
  - Routing to destination a, traffic entering at d, c, e with rates 1,  $e$  ( $<1$ ), 1
  - Link costs are directional, and volume-dependent



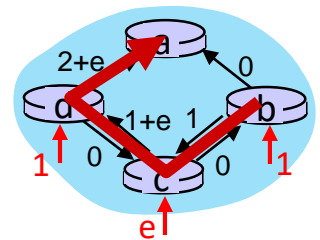
initially



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





# Dynamic Routing Algorithms

## Distance-Vector Algorithms

# Distance vector algorithm



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

Bellman-Ford equation

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

Then:

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

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

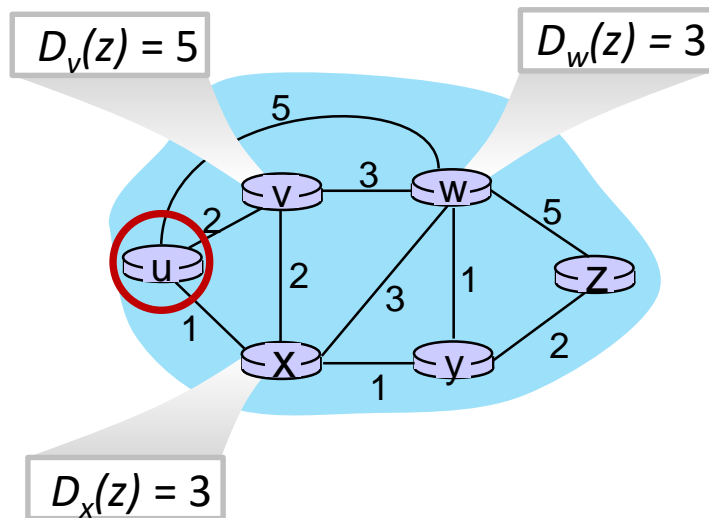
direct cost of link from  $x$  to  $v$

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

# 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:

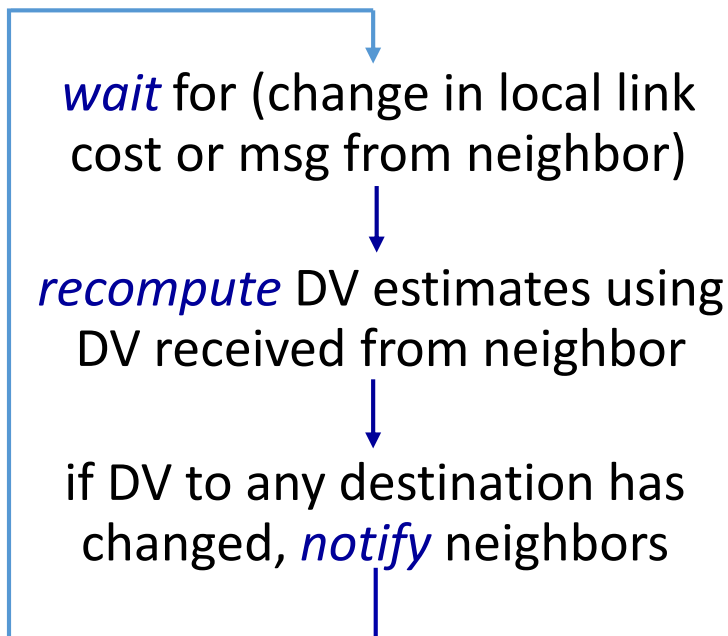
$$D_x(y) \leftarrow \min_v \{c_{x,v} + D_v(y)\} \text{ for each node } y \in N$$

- 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!



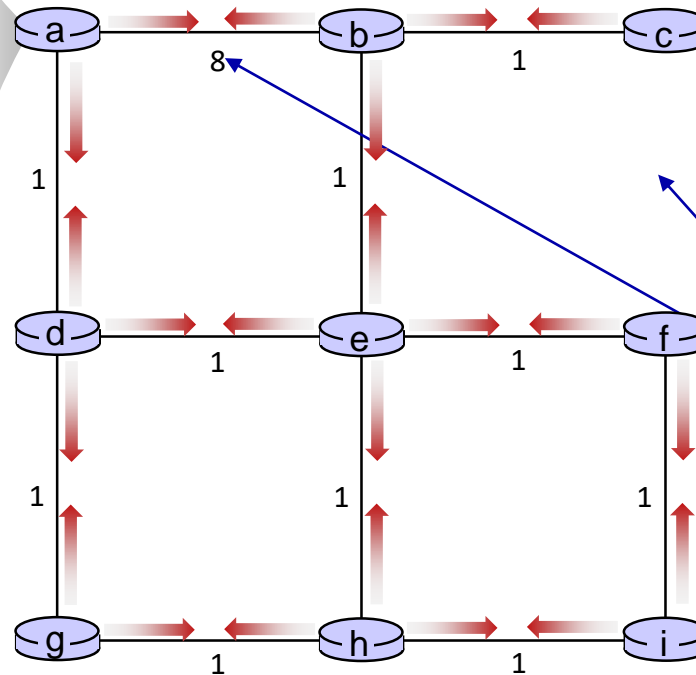
# Distance vector: example



$t=0$

- 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



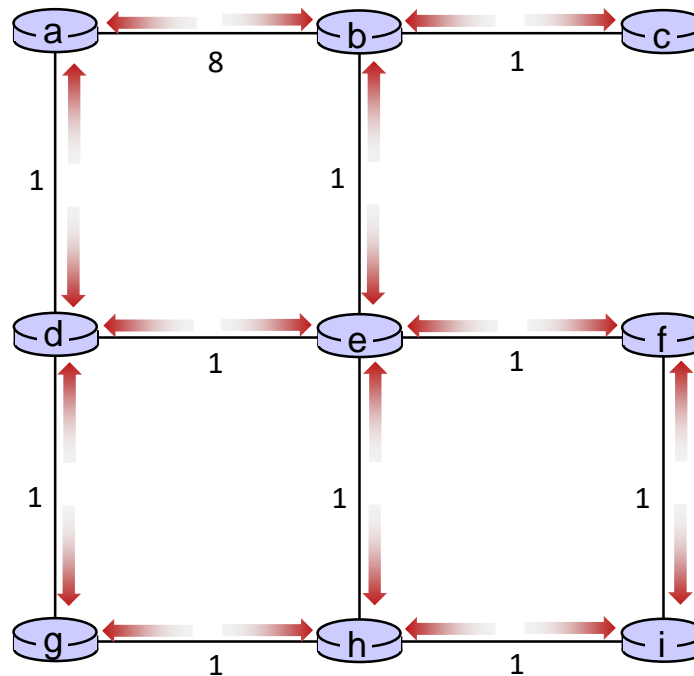
# Distance vector example: iteration



$t=1$

All nodes:

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





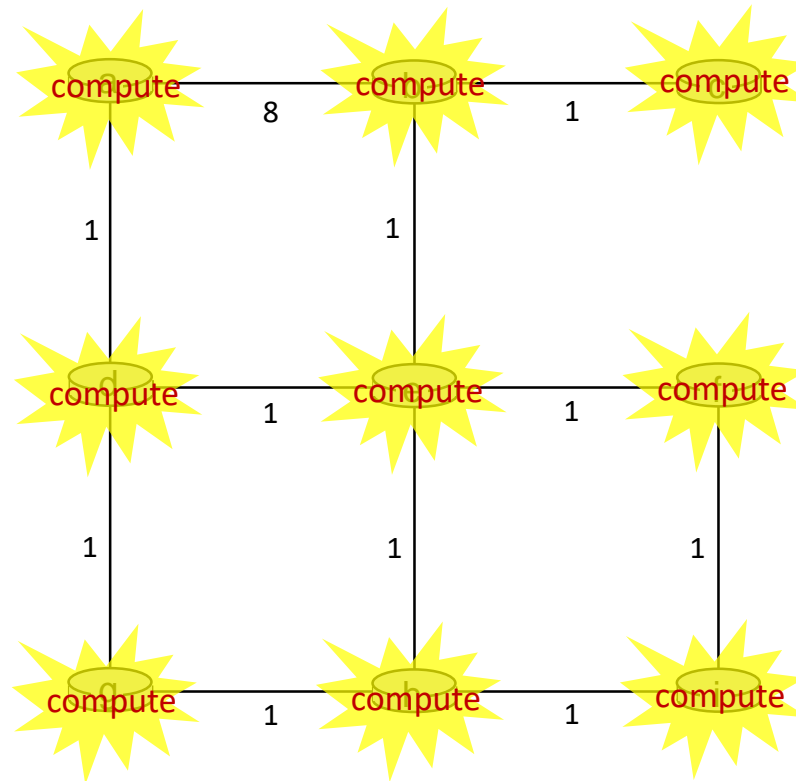
# Distance vector example: iteration



$t=1$

All nodes:

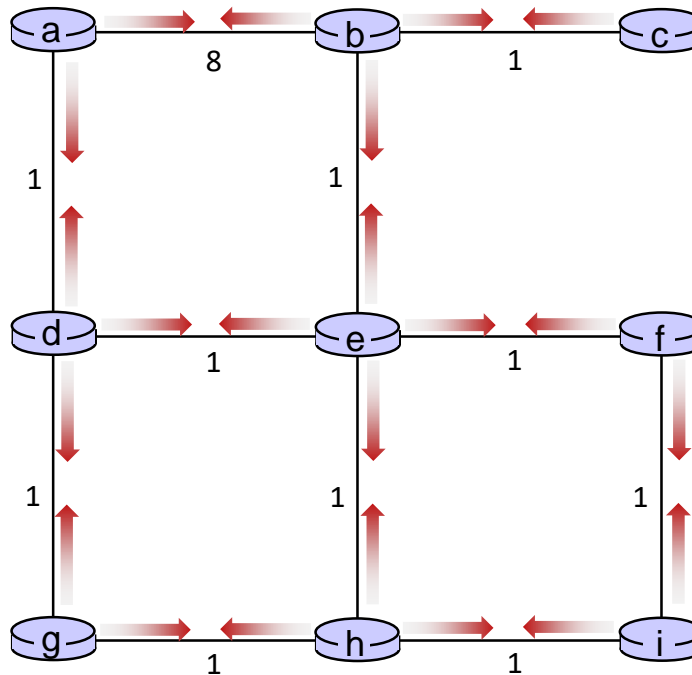
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors







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





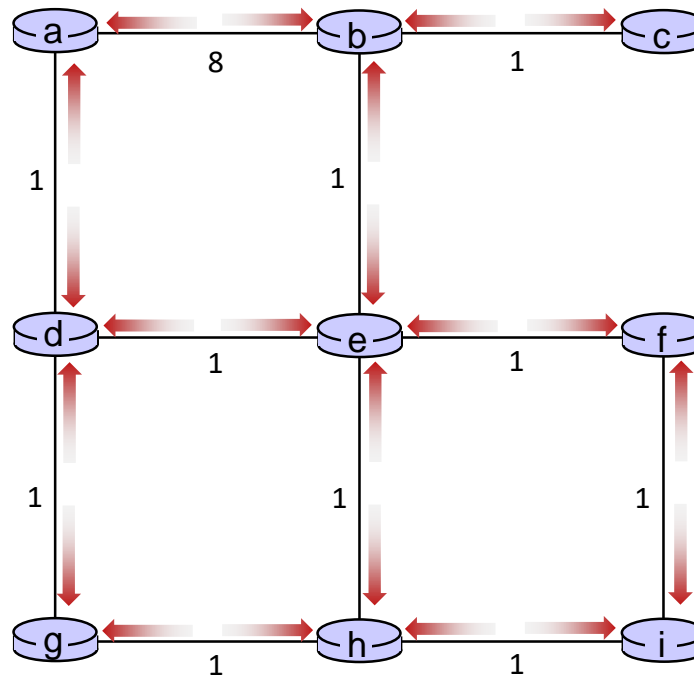
# Distance vector example: iteration



$t=2$

All nodes:

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





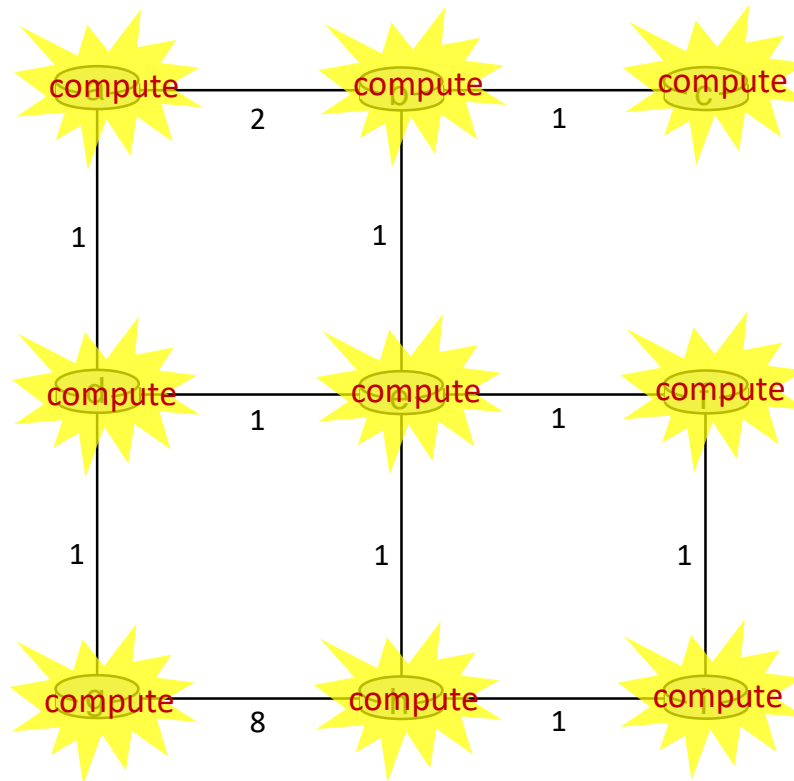
# Distance vector example: iteration



$t=2$

All nodes:

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



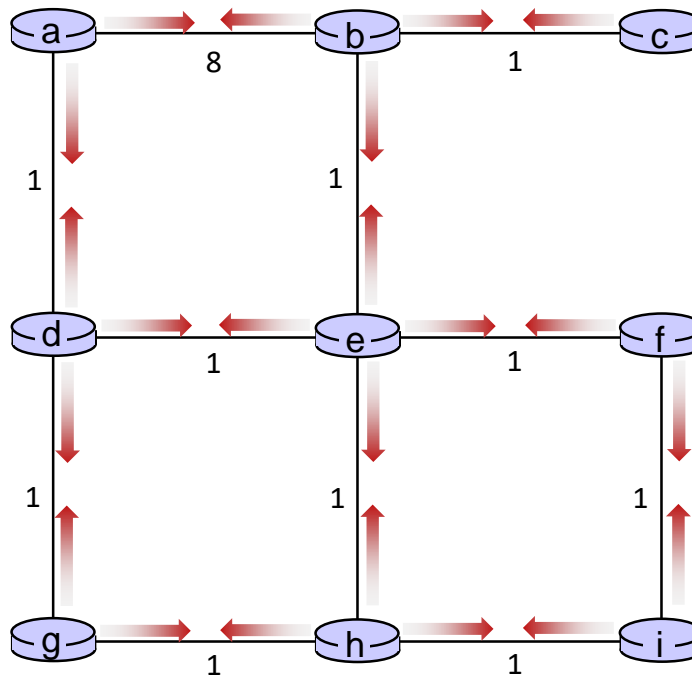
# Distance vector example: iteration



$t=2$

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

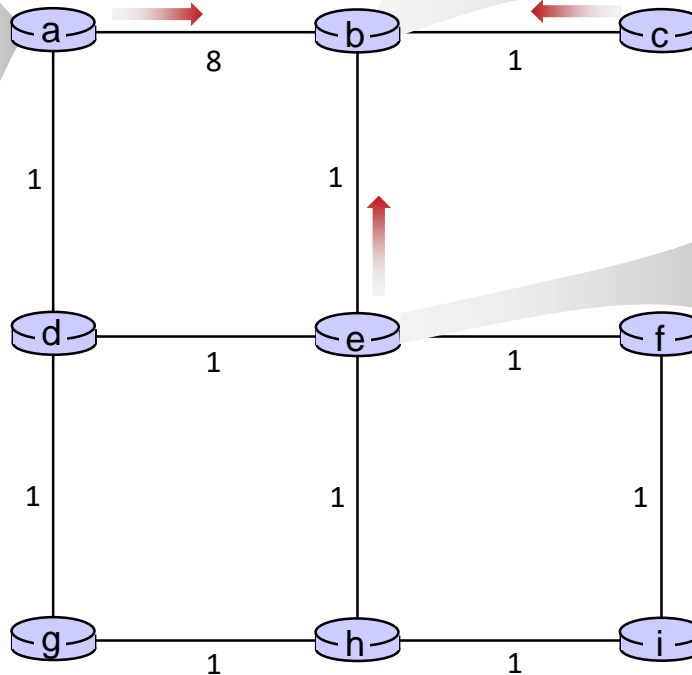
# Distance vector example: computation



**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

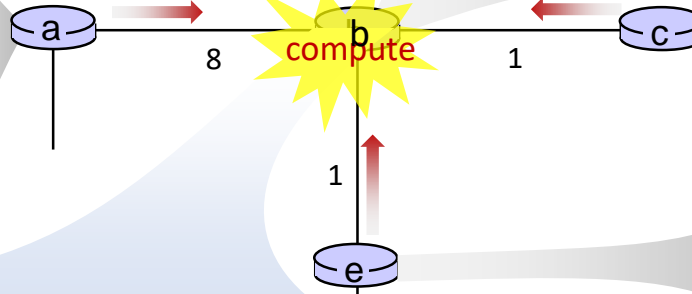


$t=1$

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

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

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

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

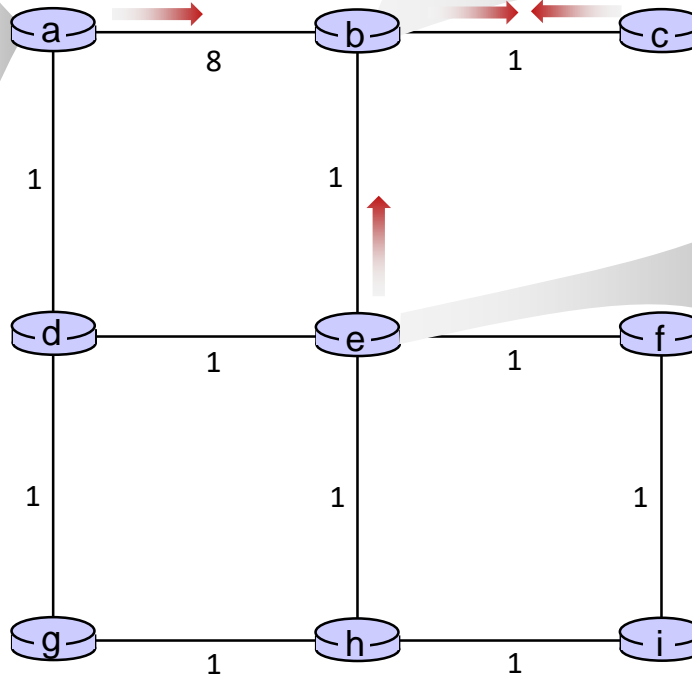
# Distance vector example: computation



$t=1$

- c receives DVs from b

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



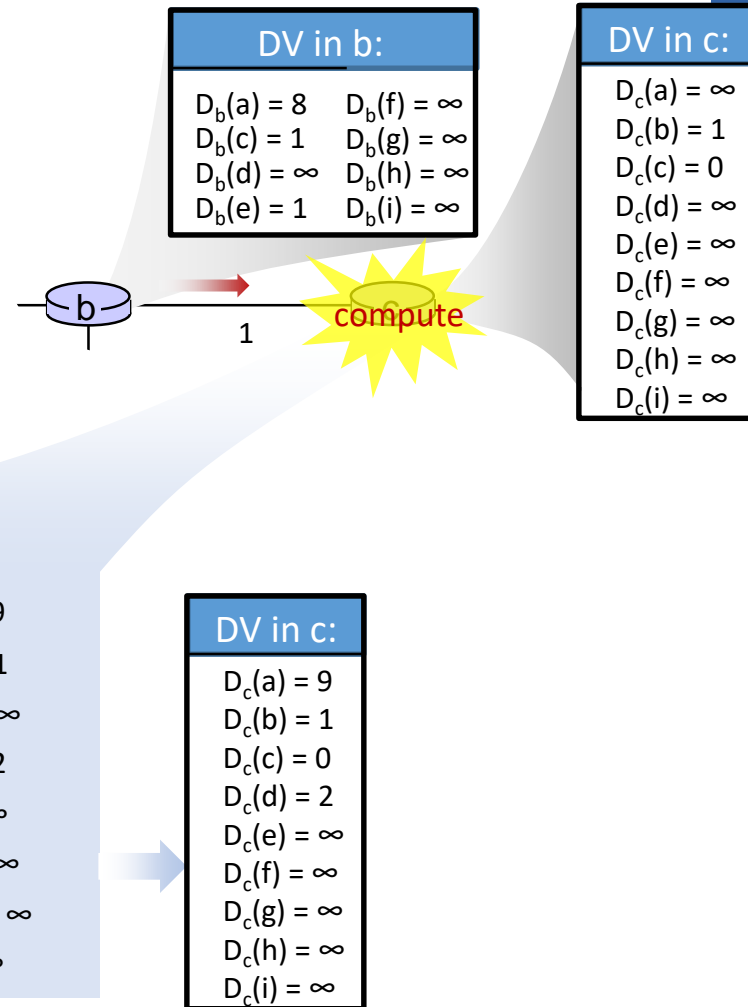
# Distance vector example: computation



$t=1$

- c receives DVs from b computes:

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



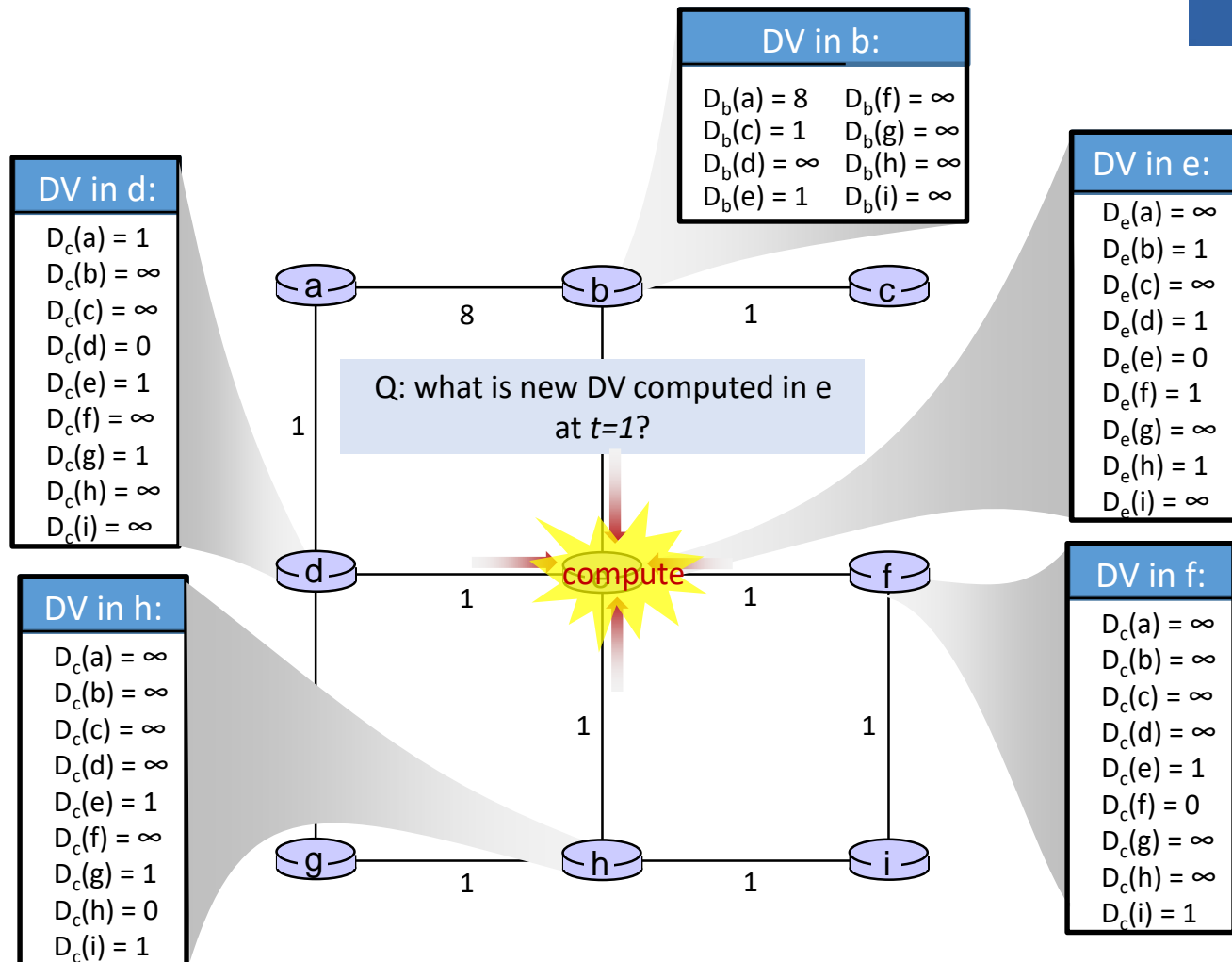


# Distance vector example: computation



$t=1$






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

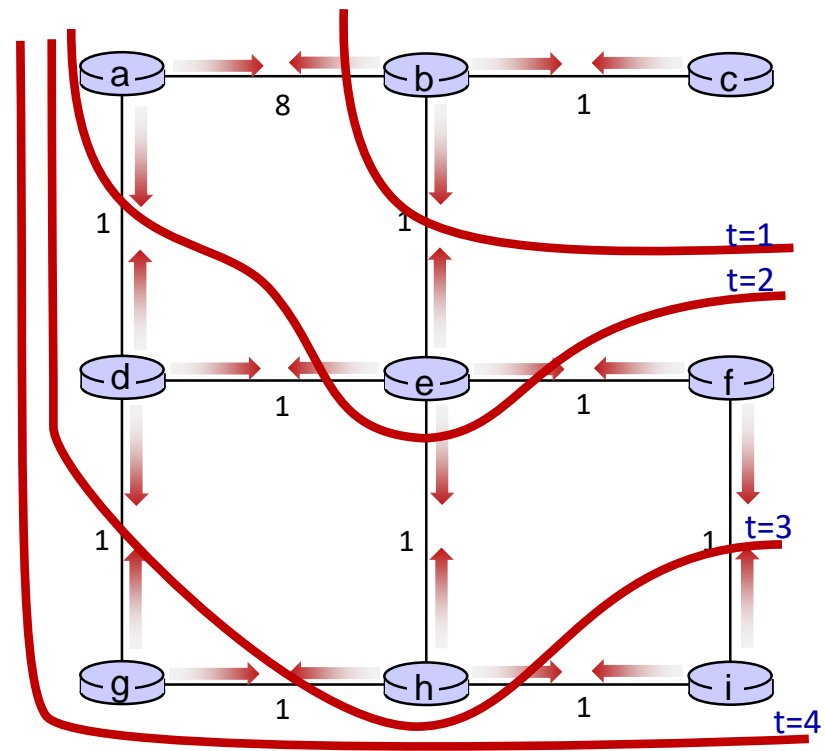


# Distance vector: state information diffusion



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

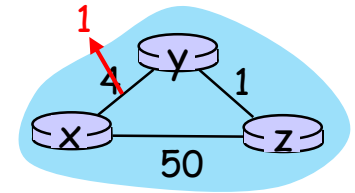


# Distance vector: link cost changes



## 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.

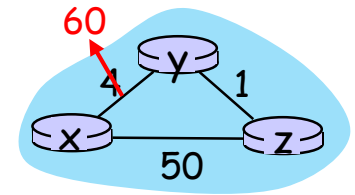
$t_2$ : y receives z's update, updates its distance table. y's least costs do *not* change, so y does *not* send a message to z.



# Distance vector: link cost changes

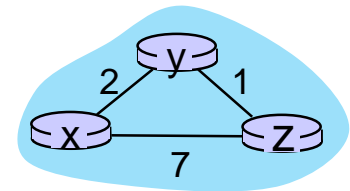
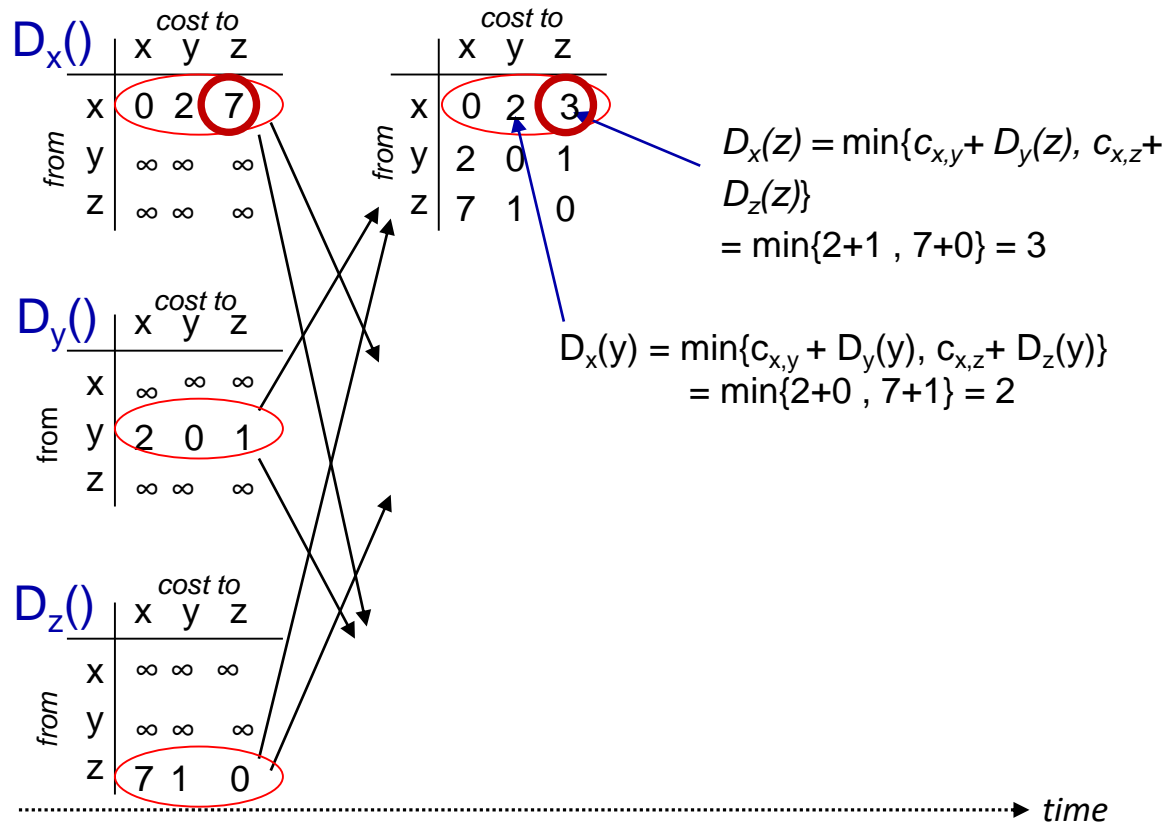
- link cost changes:

- node detects local link cost change
- “bad news travels slow” – count-to-infinity problem:

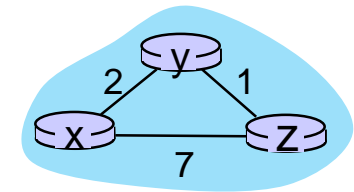
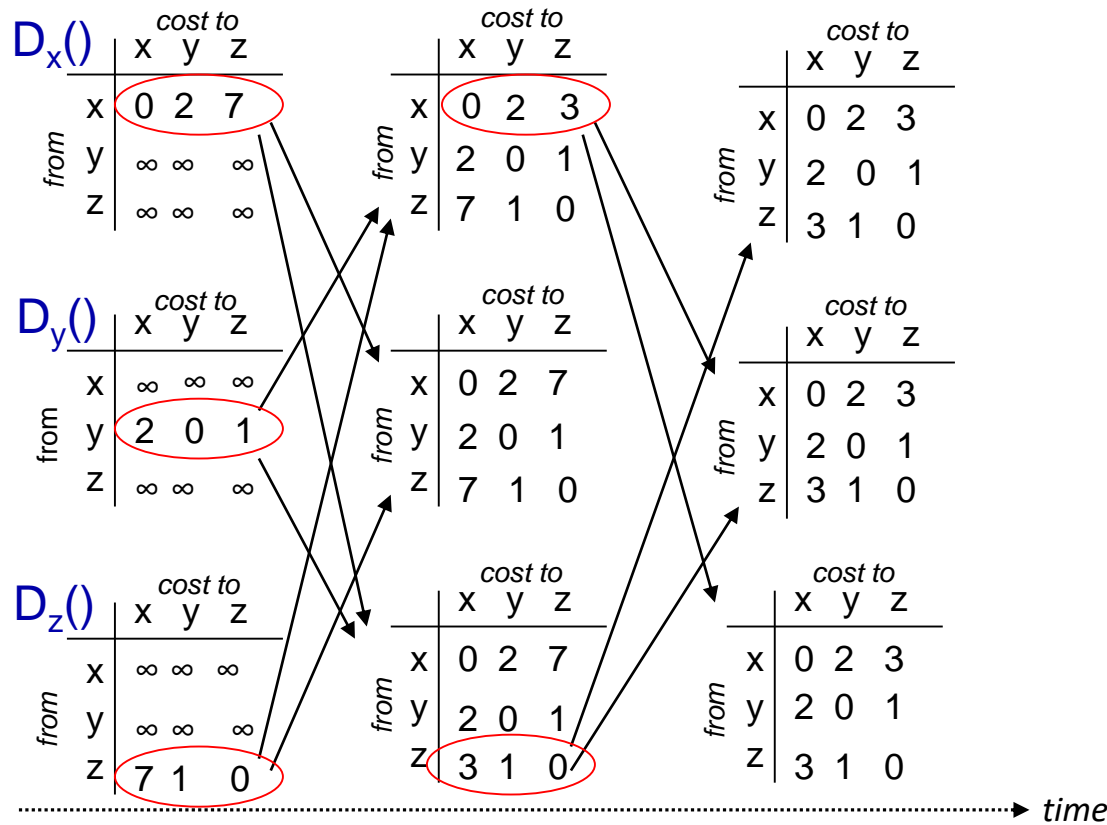


- y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes “my new cost to x will be 6, via z); notifies z of new cost of 6 to x.
- z learns that path to x via y has new cost 6, so z computes “my new cost to x will be 7 via y), notifies y of new cost of 7 to x.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

# Distance vector: another example



# Distance vector: another example





# Exercise

- When the algorithm converges, what are the distance vectors from router 'V' to all routers?

