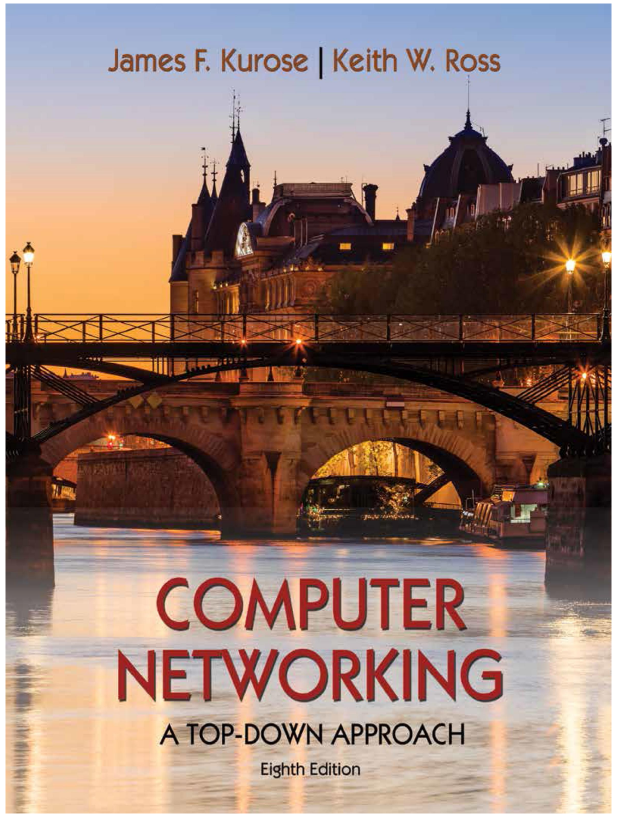
Network Layer: Control Plane

Computer Networks

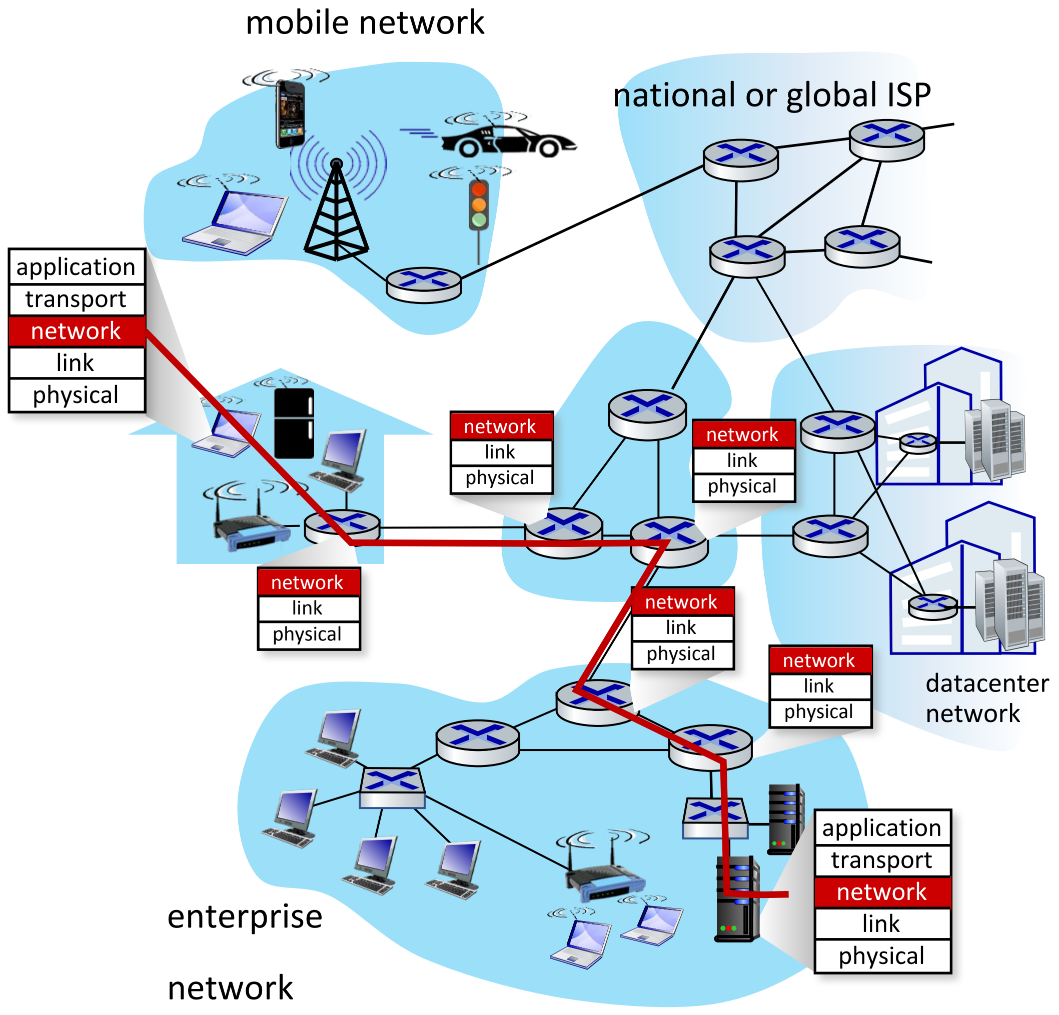
- introduction
- routing algorithms
    - link state
    - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- Network management, configuration



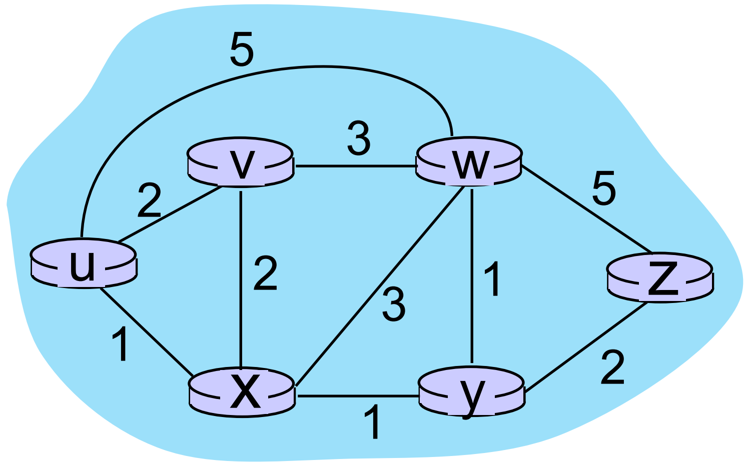
Routing algorithm

Routing algorithm 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., $c_{w,z} = 5$, $c_{u,z} = \infty$

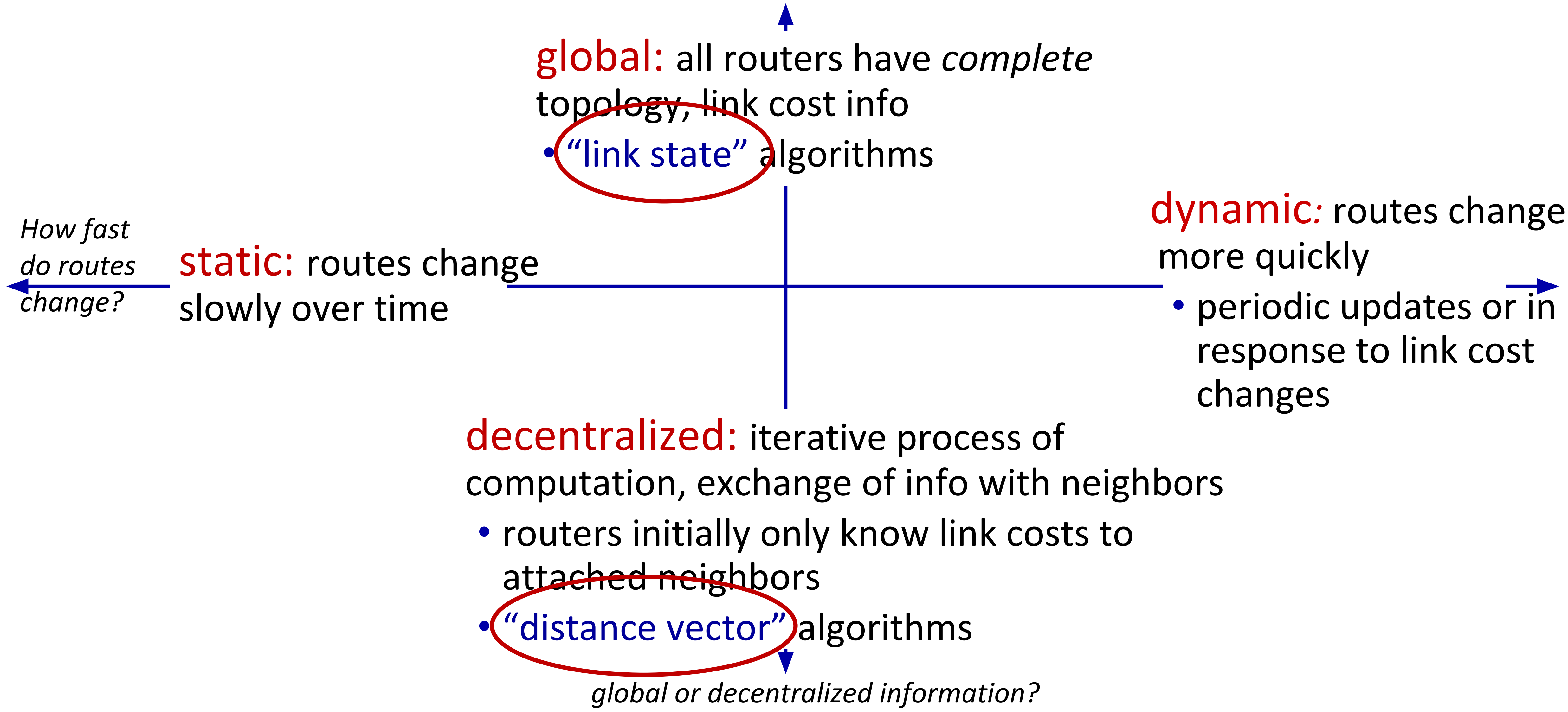
cost defined by network operator: could always be 1, or inversely related to bandwidth, or related to congestion

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



global: all routers have *complete* topology, link cost info

- "link state" algorithms

How fast do routes change?

static: routes change slowly over time

dynamic: routes change more quickly

- periodic updates or in response to link cost changes

decentralized: iterative process of computation, exchange of info with neighbors

- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

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

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

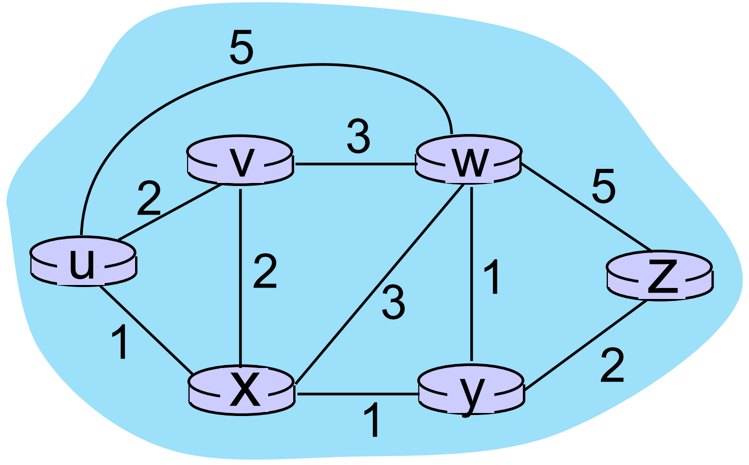
Dijkstra’s link-state routing algorithm

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

Loop
    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 $N'$ :
        $D(b) = \min ( D(b),  D(a) + c_{a,b} )$
    /* new least-path-cost to $b$ is either old least-cost-path to $b$ or known
    least-cost-path to $a$ plus direct-cost from $a$ to $b$ */
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						
2						
3						
4						
5						

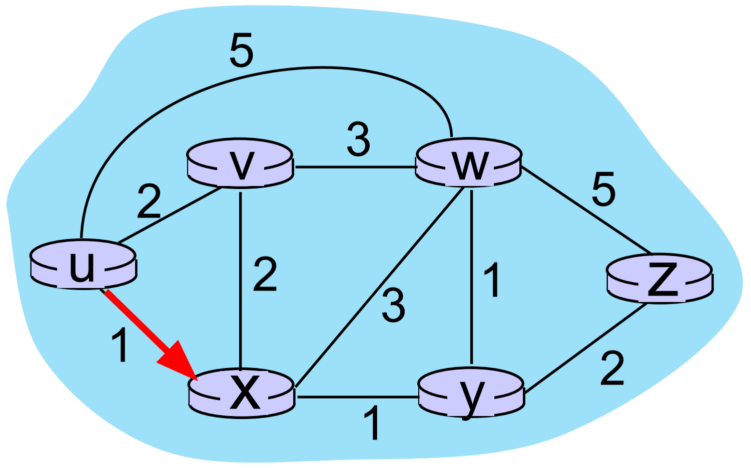


Initialization (step 0):

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

Dijkstra's algorithm: an example

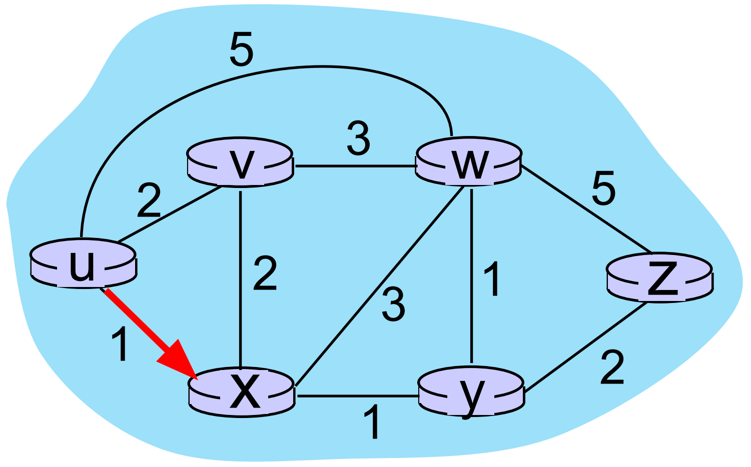
Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux					
2						
3						
4						
5						



```
8   Loop
9       find a not in N' such that D(a) is a minimum
10      add a to 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						
3						
4						
5						



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

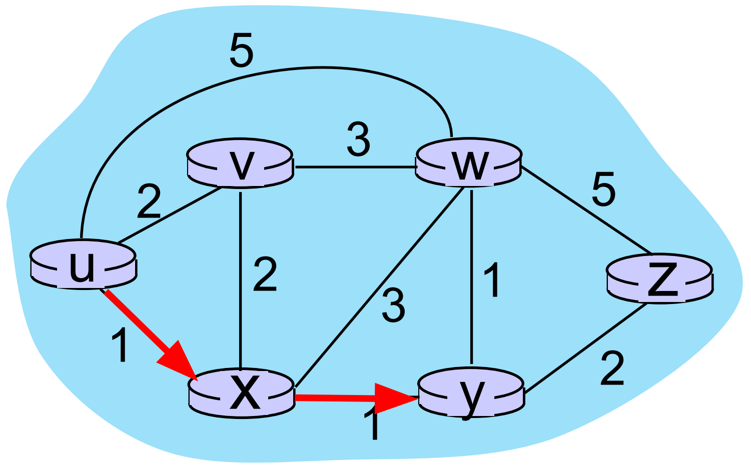
$D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$

$D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$ NEW!

$D(y) = \min (D(y), D(x) + c_{x,y}) = \min(inf,1+1) = 2$ NEW!

Dijkstra’s algorithm: an example

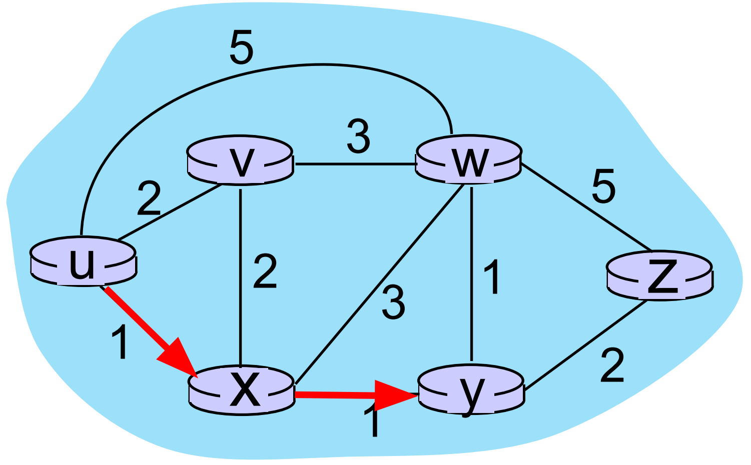
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy					
3						
4						
5						



```
8   Loop
9       find a not in N′ such that D(a) is a minimum
10      add a to N′
```

Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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						
4						
5						



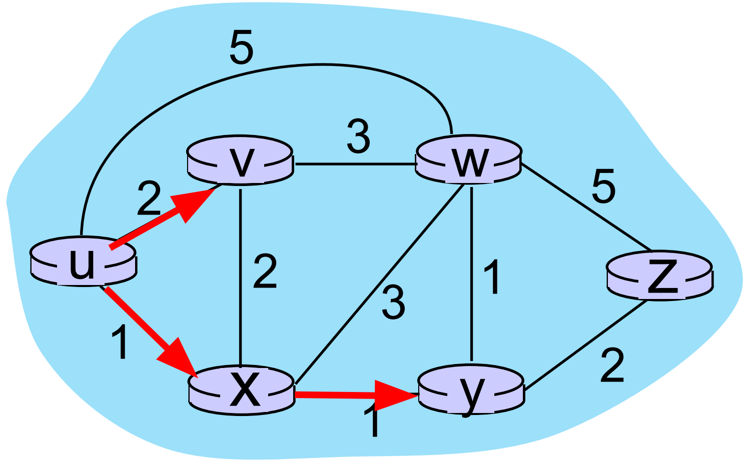
```
8   Loop
9       find a not in N' such that D(a) is a minimum
10      add a to N'
11      update D(b) for all b adjacent to a and not in N' :
            D(b) = min ( D(b), D(a) + c_{a,b} )
```

$D(w) = \min (D(w), D(y) + c_{y,w}) = \min (4, 2+1) = 3$ NEW!

$D(z) = \min (D(z), D(y) + c_{y,x}) = \min(\inf, 2+2) = 4$ NEW!

Dijkstra's algorithm: an example

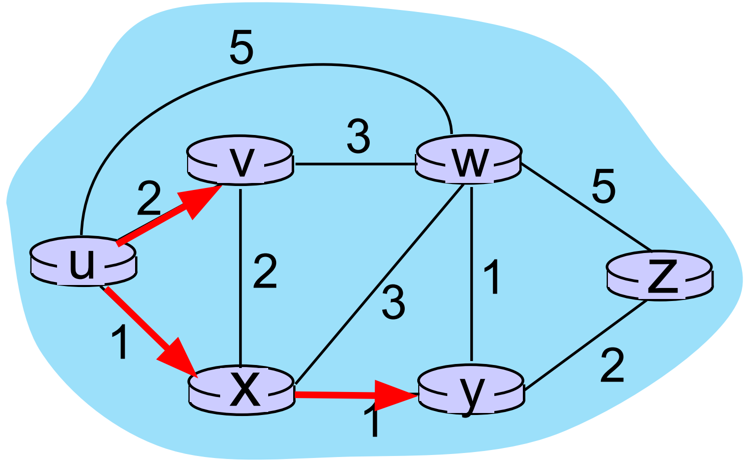
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						



```
8   Loop
9       find a not in N' such that D(a) is a minimum
10      add a to 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						
5						

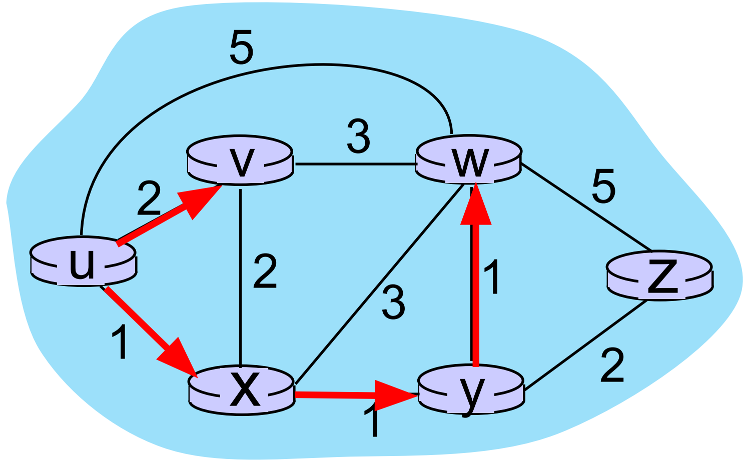


```
8  Loop
9      find a not in N' such that D(a) is a minimum
10     add a to N'
11     update D(b) for all b adjacent to a and not in N' :
           D(b) = min ( D(b), D(a) + c_{a,b} )
```

$D(w) = \min(D(w), D(v) + c_{v,w}) = \min(3, 2+3) = 3$

Dijkstra’s algorithm: an example

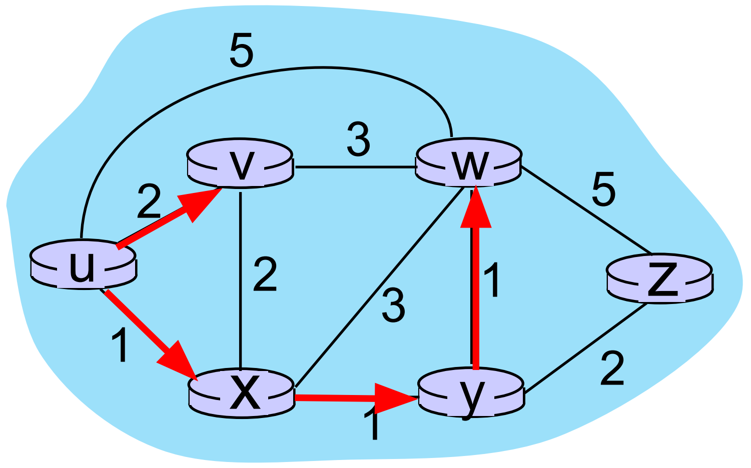
Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5						



```
8   Loop
9       find a not in N' such that D(a) is a minimum
10      add a to 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						

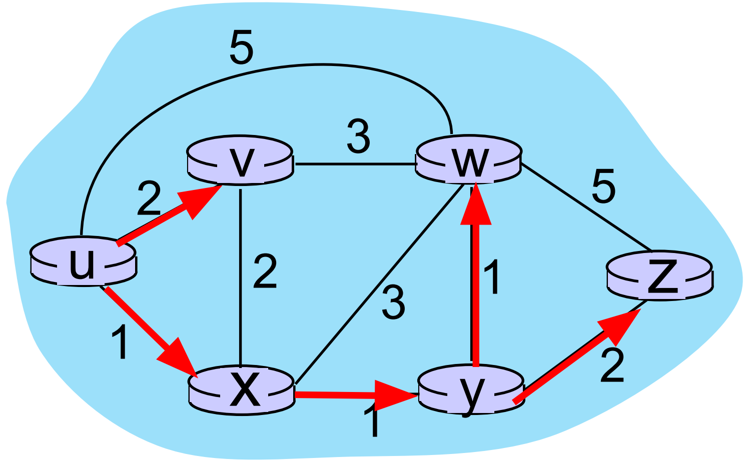


```
8    Loop
9      find a not in N' such that D(a) is a minimum
10     add a to N'
11     update D(b) for all b adjacent to a and not in N' :
           D(b) = min ( D(b), D(a) + c_{a,b} )
```

$D(z) = \min\,(\,D(z),\, D(w) + c_{w,z}\,) = \min\,(4,\, 3+5) = 4$

Dijkstra’s algorithm: an example

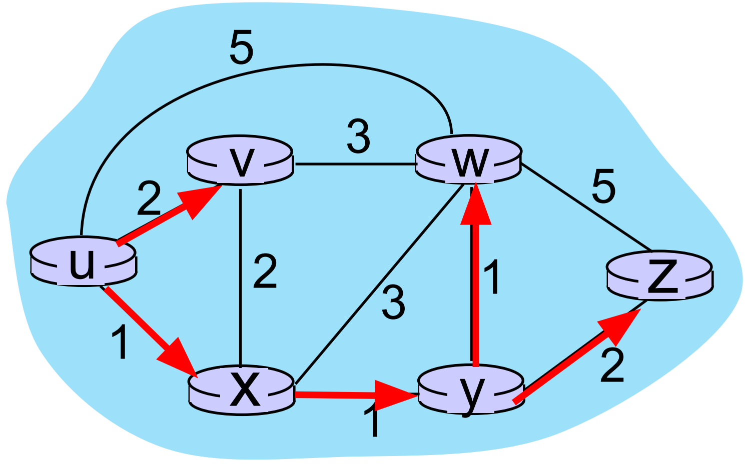
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



```
8   Loop
9       find a not in N' such that D(a) is a minimum
10      add a to N'
```

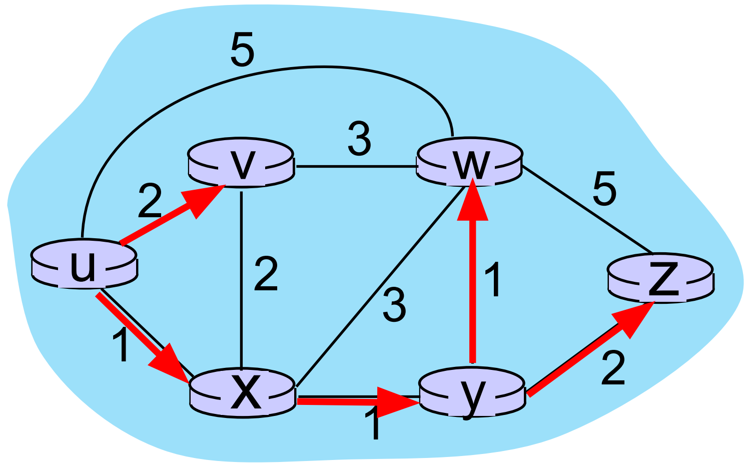
Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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					

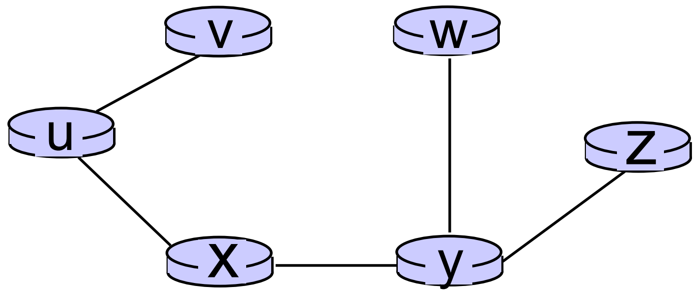


```
8   Loop
9       find $a$ not in $N'$ such that $D(a)$ is a minimum
10      add $a$ to $N'$
11      update $D(b)$ for all $b$ adjacent to $a$ and not in $N'$ :
            $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

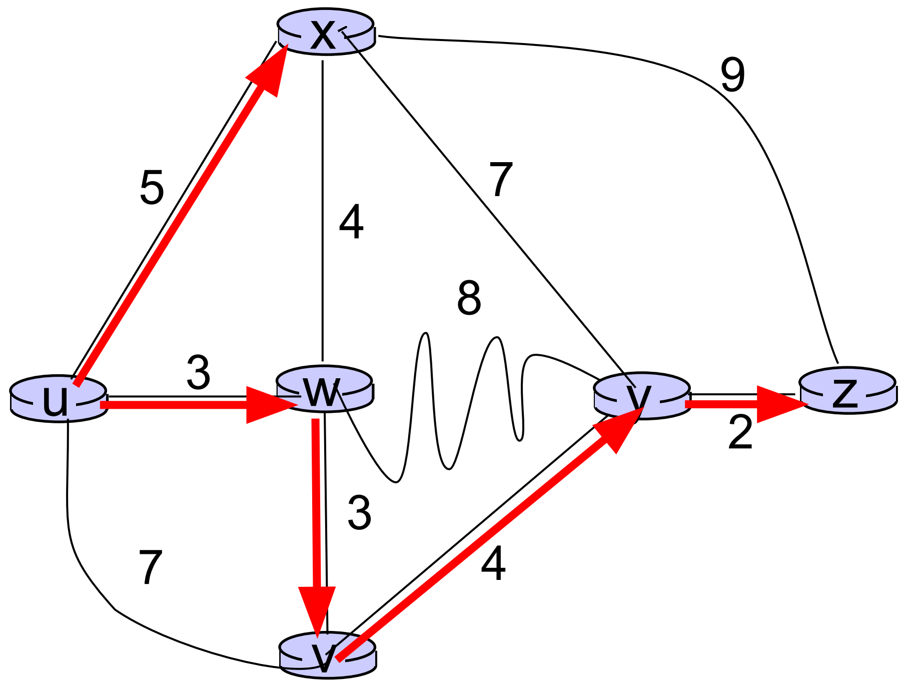
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

Dijkstra’s algorithm: another example

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

Dijkstra's algorithm: discussion

algorithm complexity: *n* nodes

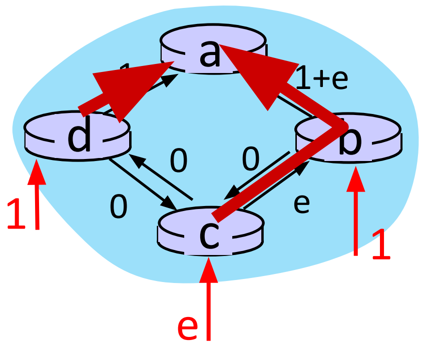
- each of *n* iteration: need to check all nodes, *w*, not in *N*
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

message complexity:

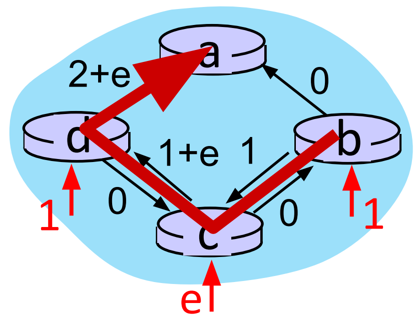
- each router must *broadcast* its link state information to other *n* routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Dijkstra's algorithm: oscillations possible

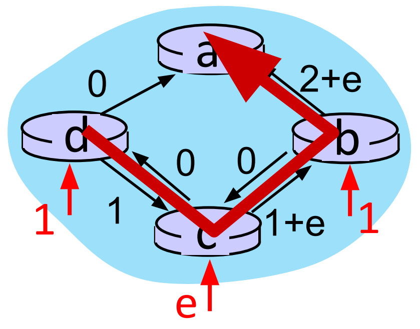
- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - link costs are directional, and volume-dependent



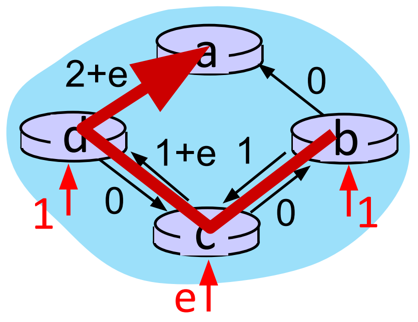
initially



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



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

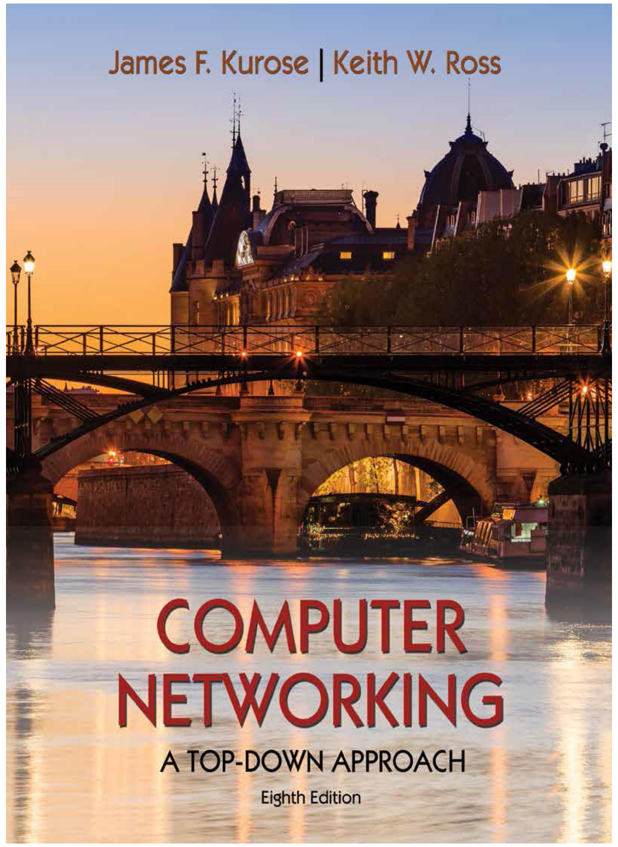


given these costs, find new routing…. resulting in new costs

Network Layer: Control Plane

Computer Networks

- introduction
- routing algorithms
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- Network management, configuration



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:

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

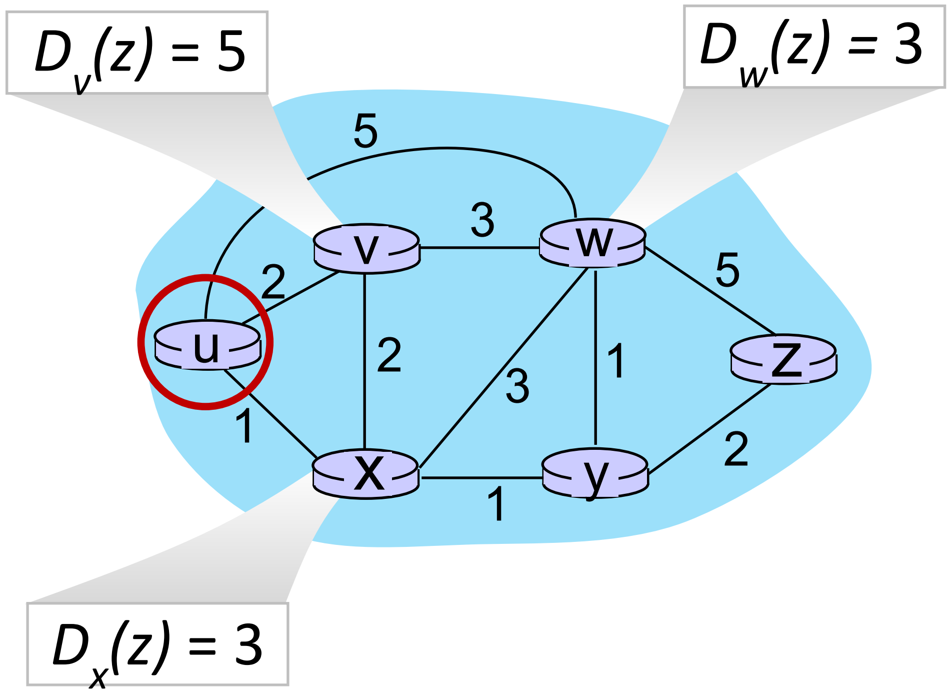
v's estimated least-cost-path cost to *y*

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:

$$\boxed{D_u(z)} = \min \{ c_{u,v} + D_v(z), \\ \boxed{c_{u,x} + D_x(z)}, \\ c_{u,w} + D_w(z) \}$$

$$= \min \{2 + 5, \\ 1 + 3, \\ 5 + 3\} \boxed{= 4}$$

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:

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

recompute my DV estimates using DV received from neighbor

if my DV to any destination has changed, *send my new DV* my neighbors, else do nothing.

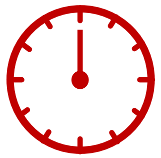
iterative, asynchronous: each local iteration caused by:

- 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!

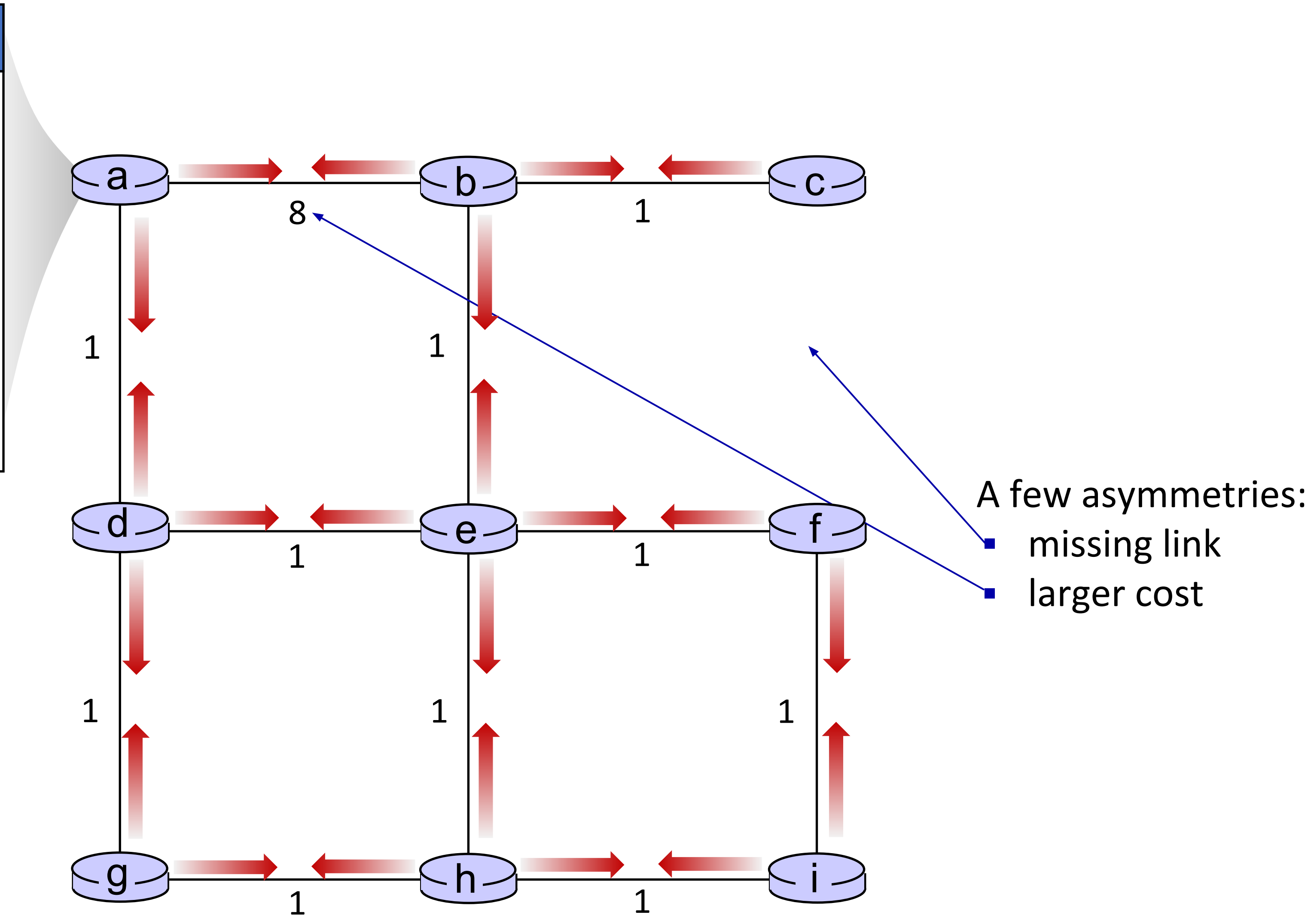
Distance vector: example



t=0

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

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



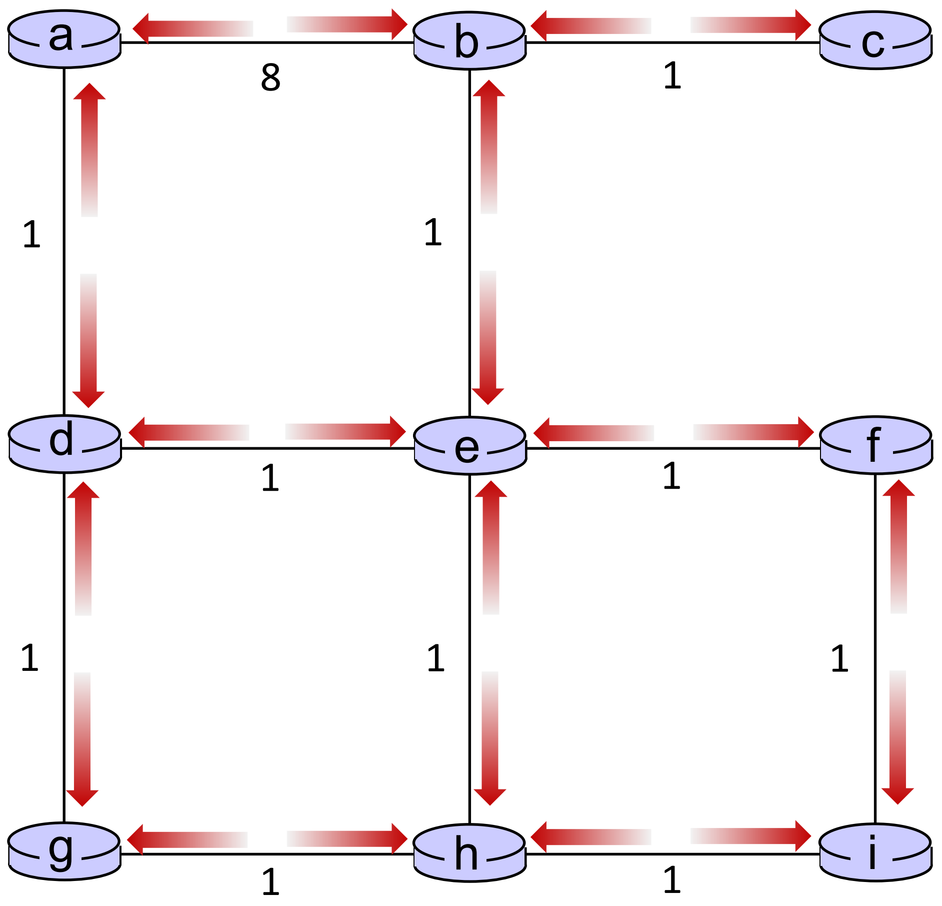
A few asymmetries:
- missing link
- larger cost

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

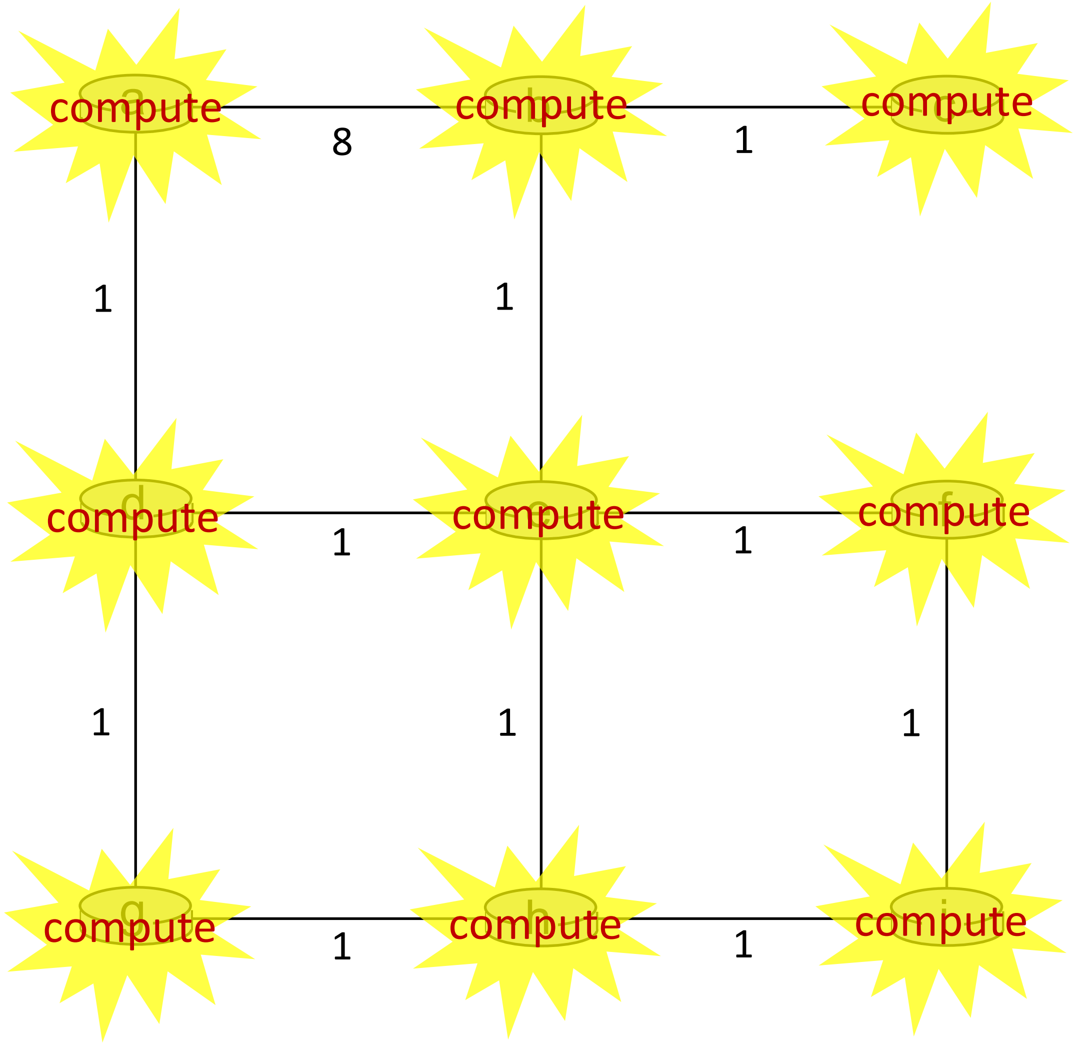


Distance vector example: iteration

t=1

All nodes:

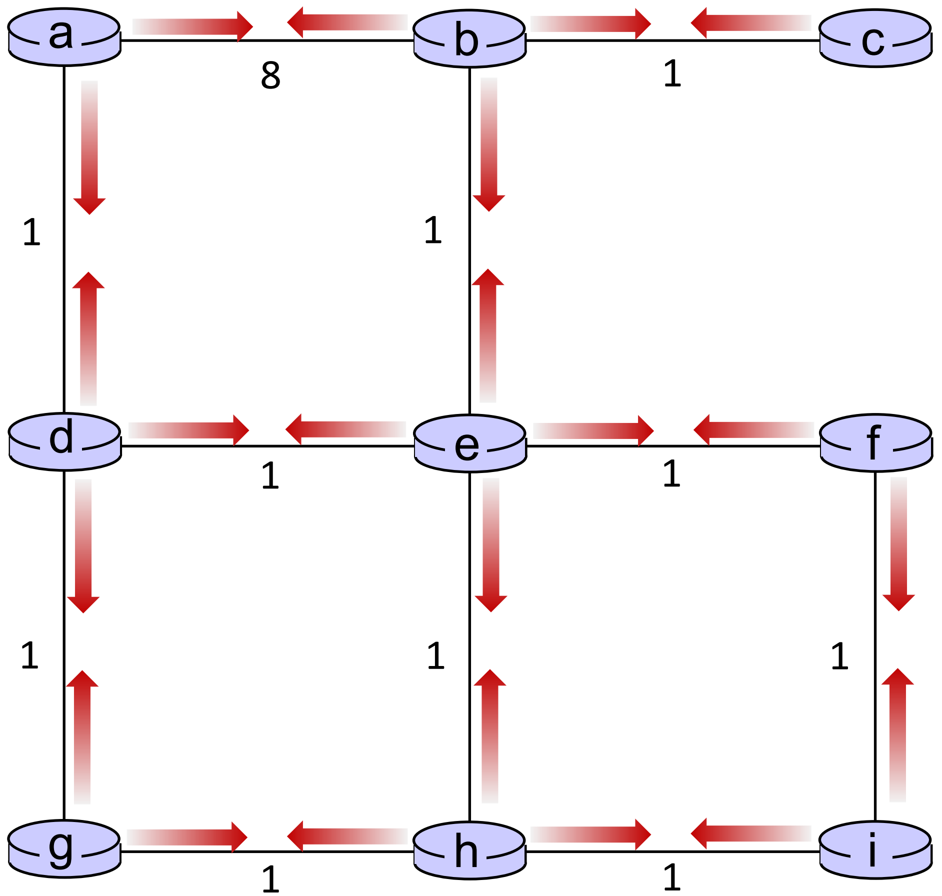
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



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

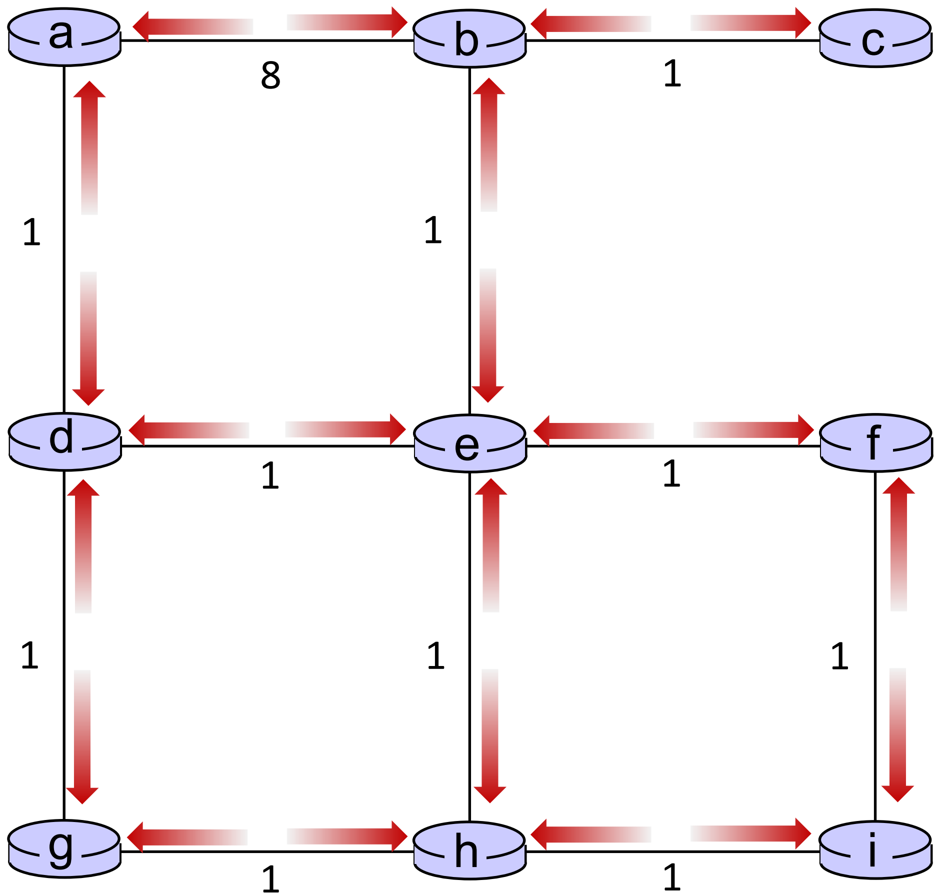


Distance vector example: iteration

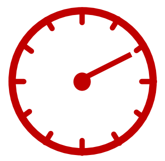
t=2

All nodes:

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



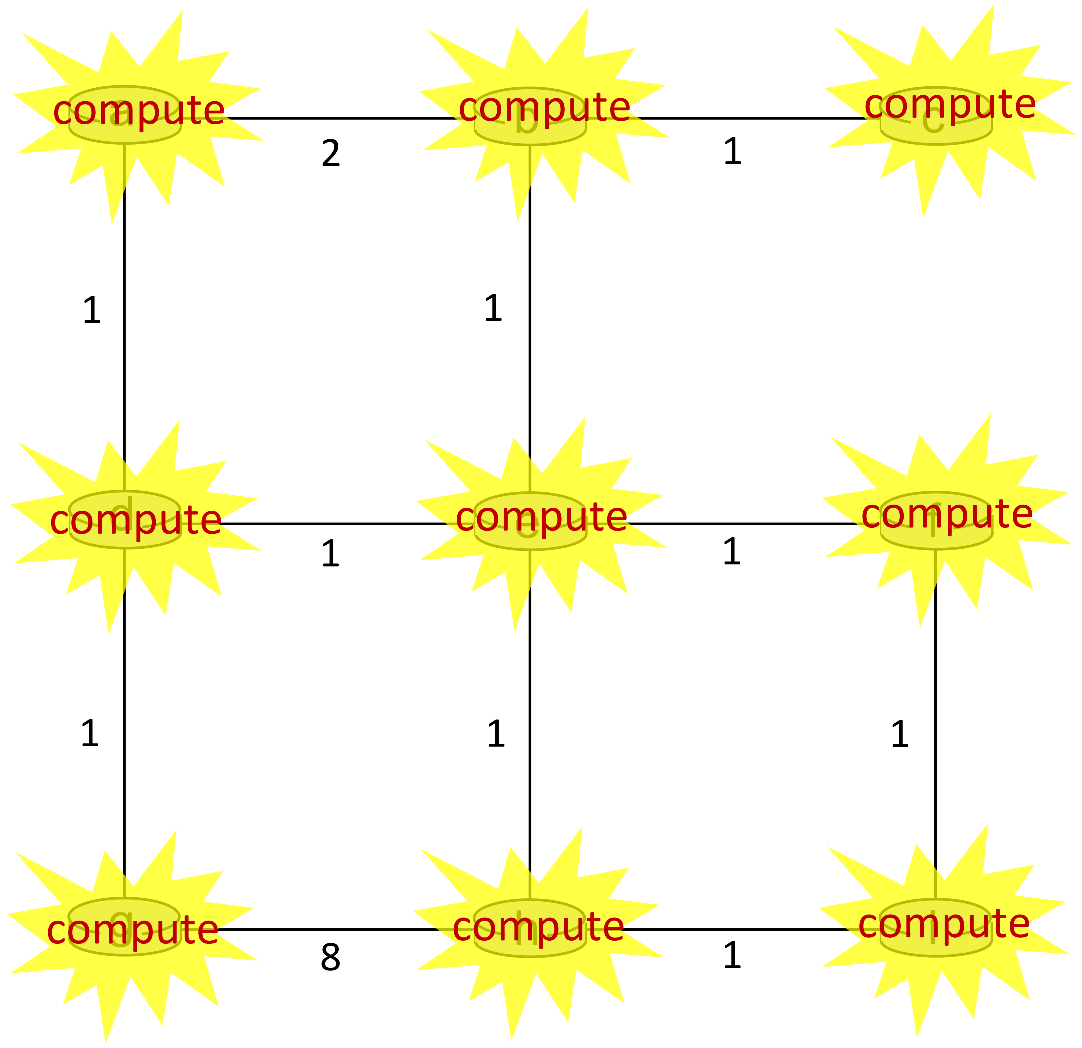
Distance vector example: iteration



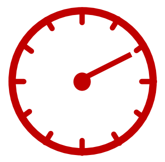
t=2

All nodes:

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



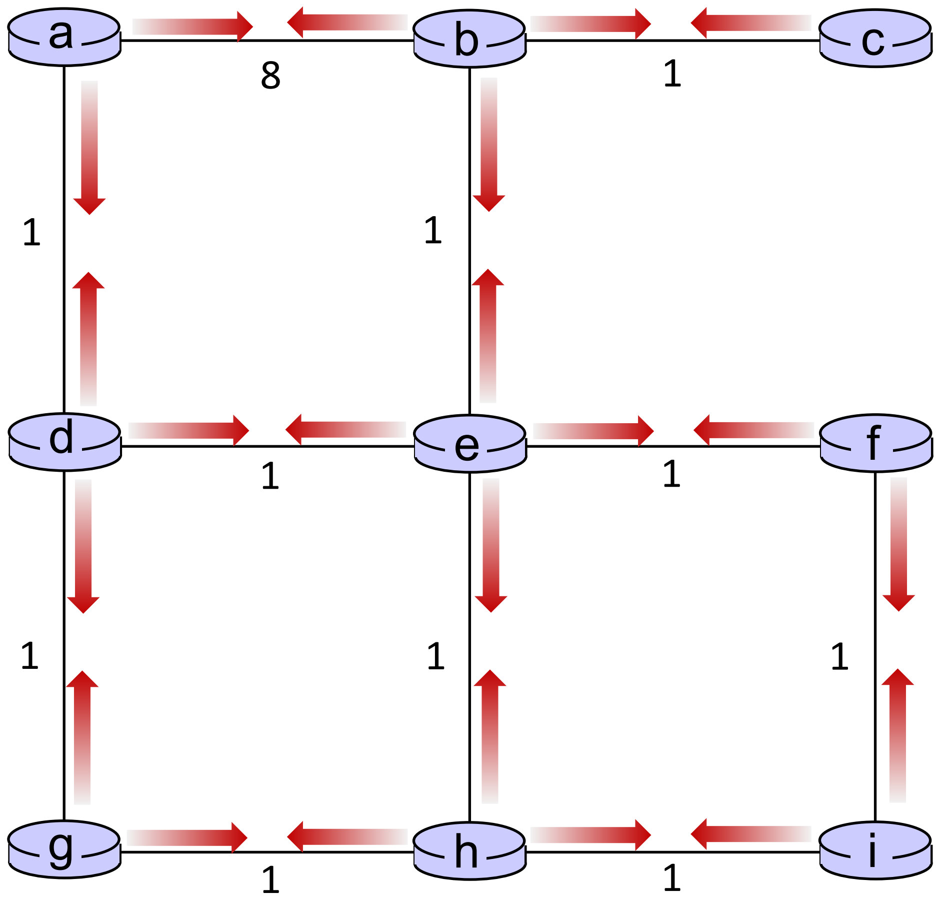
Distance vector example: iteration



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

t=1

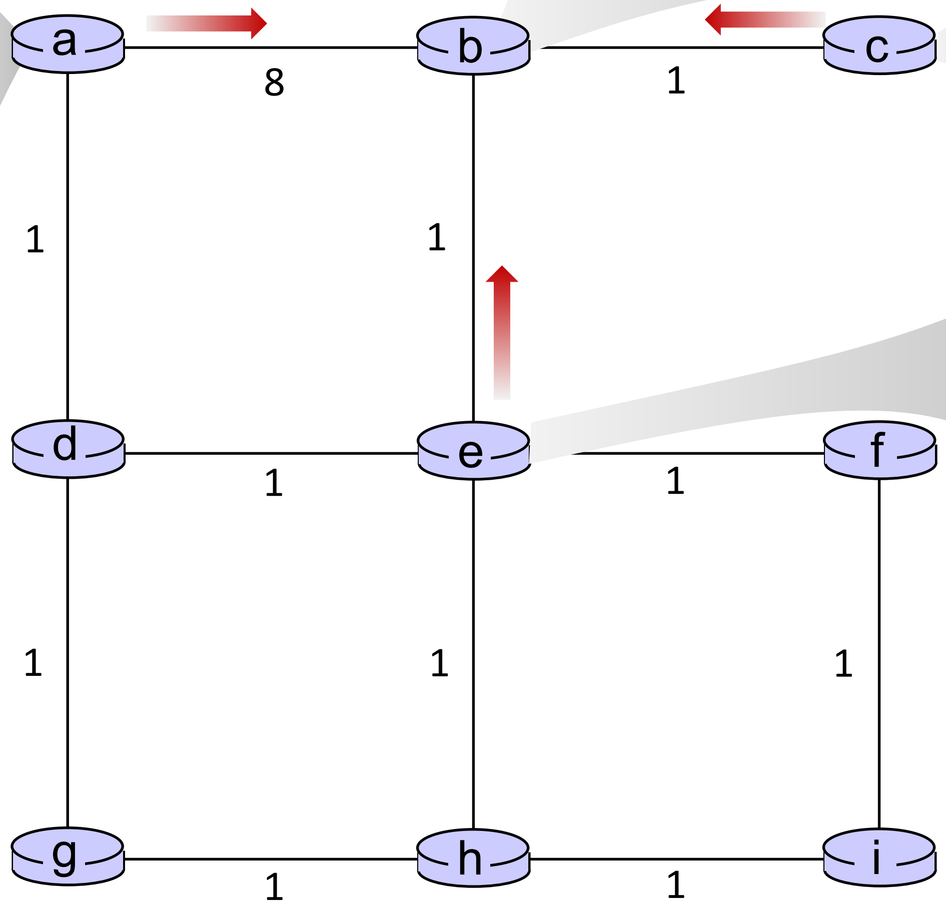
- b receives DVs from a, c, e

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

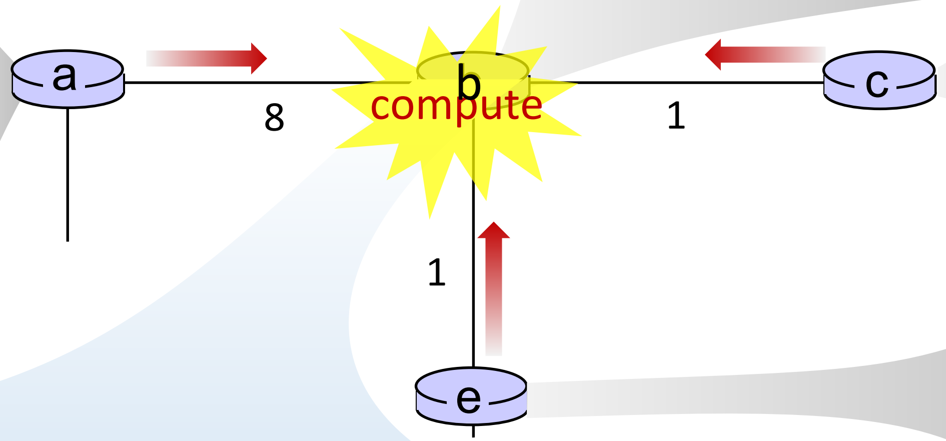
t=1

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

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

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

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

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

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

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

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

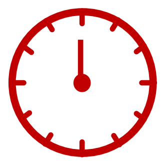
$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2$

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

New DV in b

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



t=1

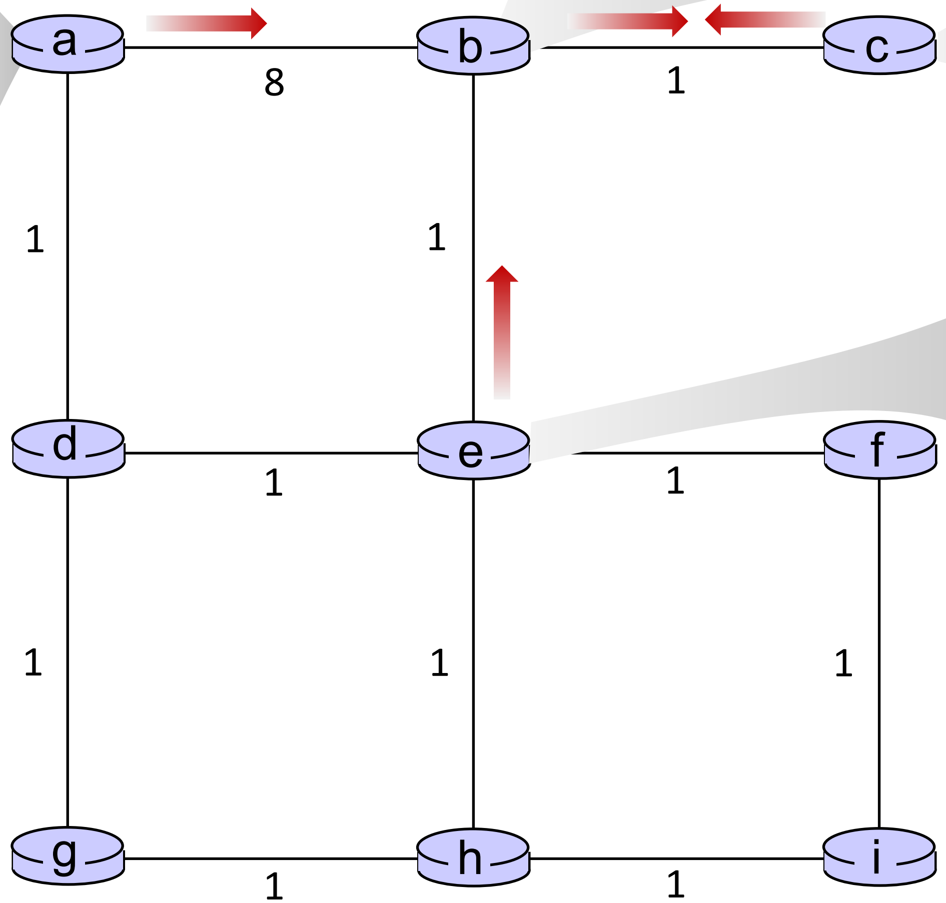
- c receives DVs from b

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

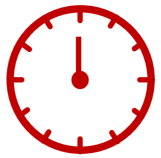
DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

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

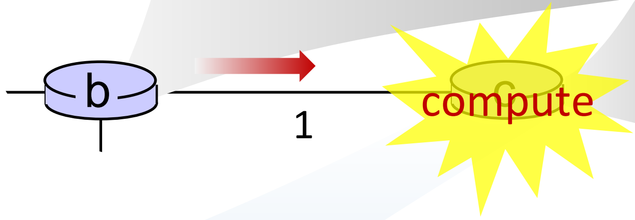


Distance vector example



t=1

- c receives DVs from b computes:



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:

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

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

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

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

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

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

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

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

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

New DV in c

DV in c:

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

Distance vector example n

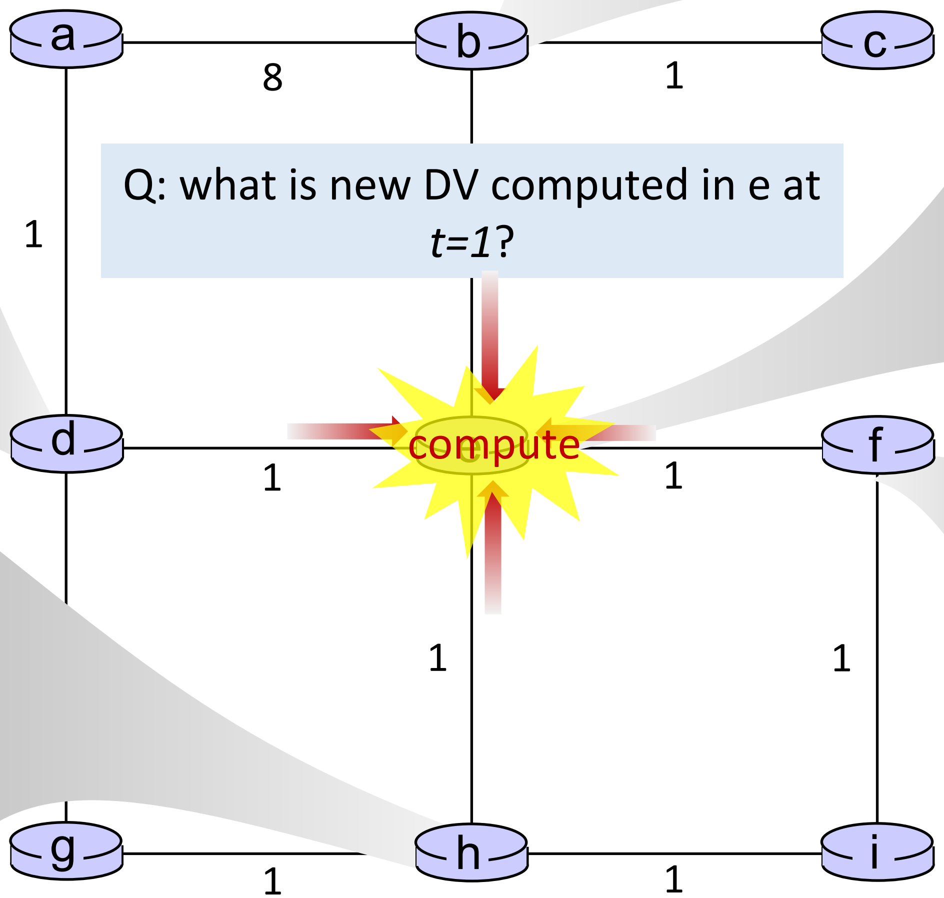
t=1

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

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

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



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

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:

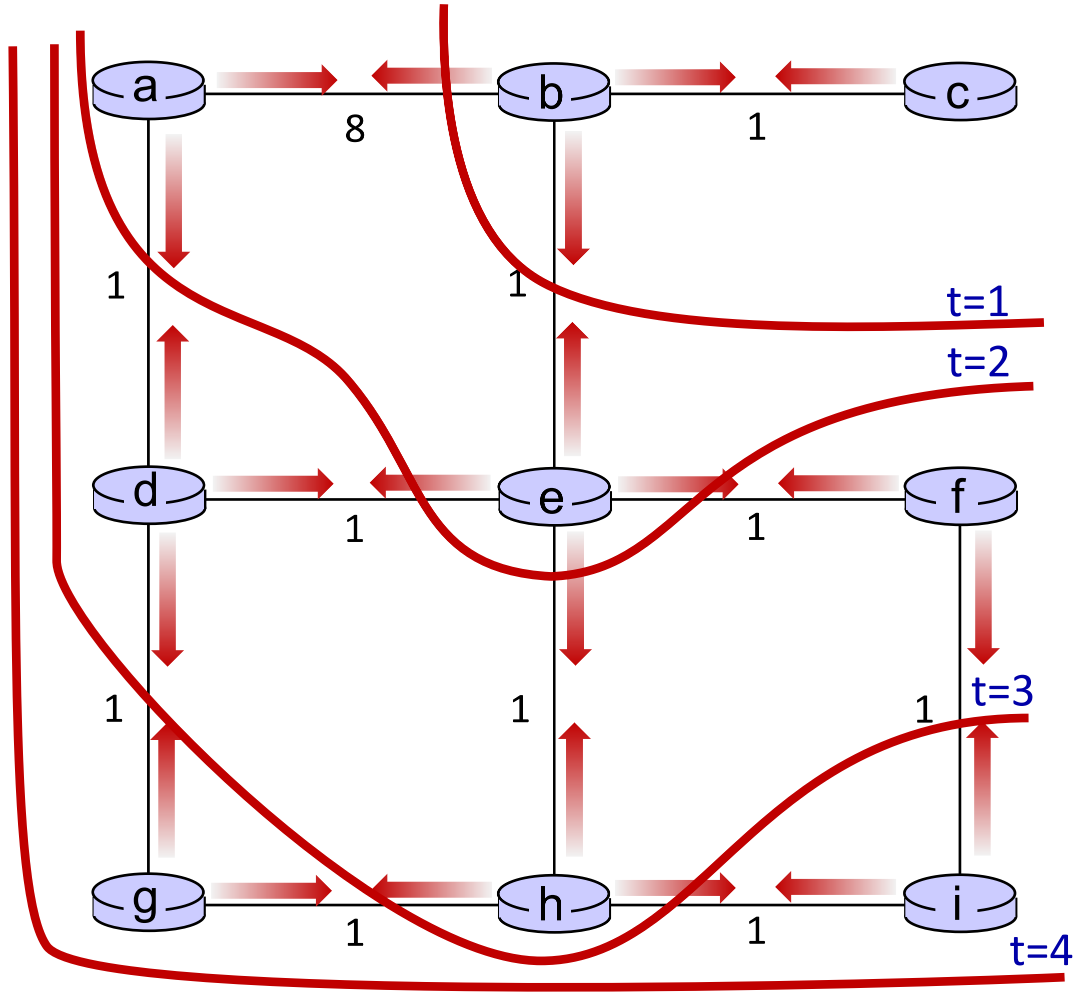
t=0 c’s state at t=0 is at c only

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

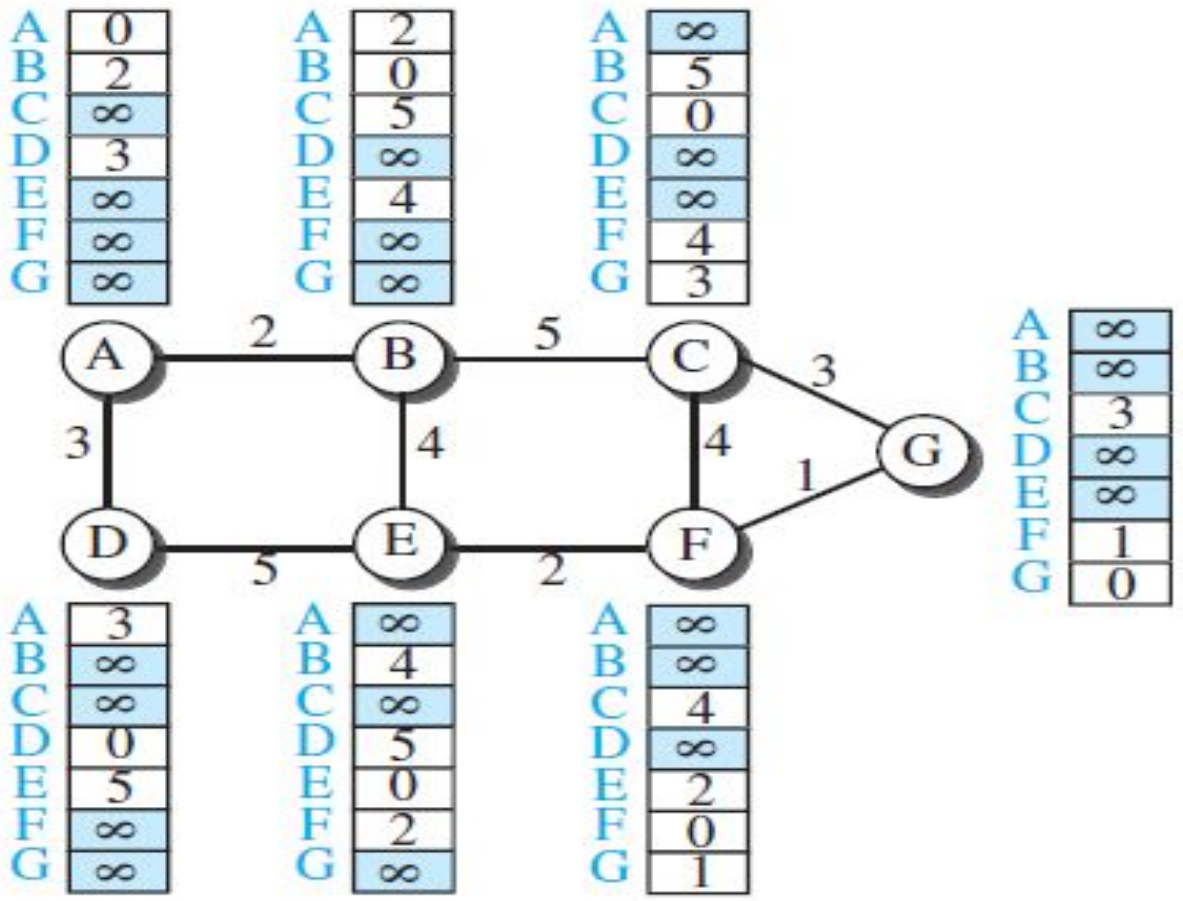
t=2 c’s state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well

t=3 c’s state at t=0 may influence distance vector computations up to **3** hops away, i.e., at d, f, h

t=4 c’s state at t=0 may influence distance vector computations up to **4** hops away, i.e., at g, i



Initial Distance Vector for an internet in the Figure



A	0
B	2
C	∞
D	3
E	∞
F	∞
G	∞

A	2
B	0
C	5
D	∞
E	4
F	∞
G	∞

A	∞
B	5
C	0
D	∞
E	∞
F	4
G	3

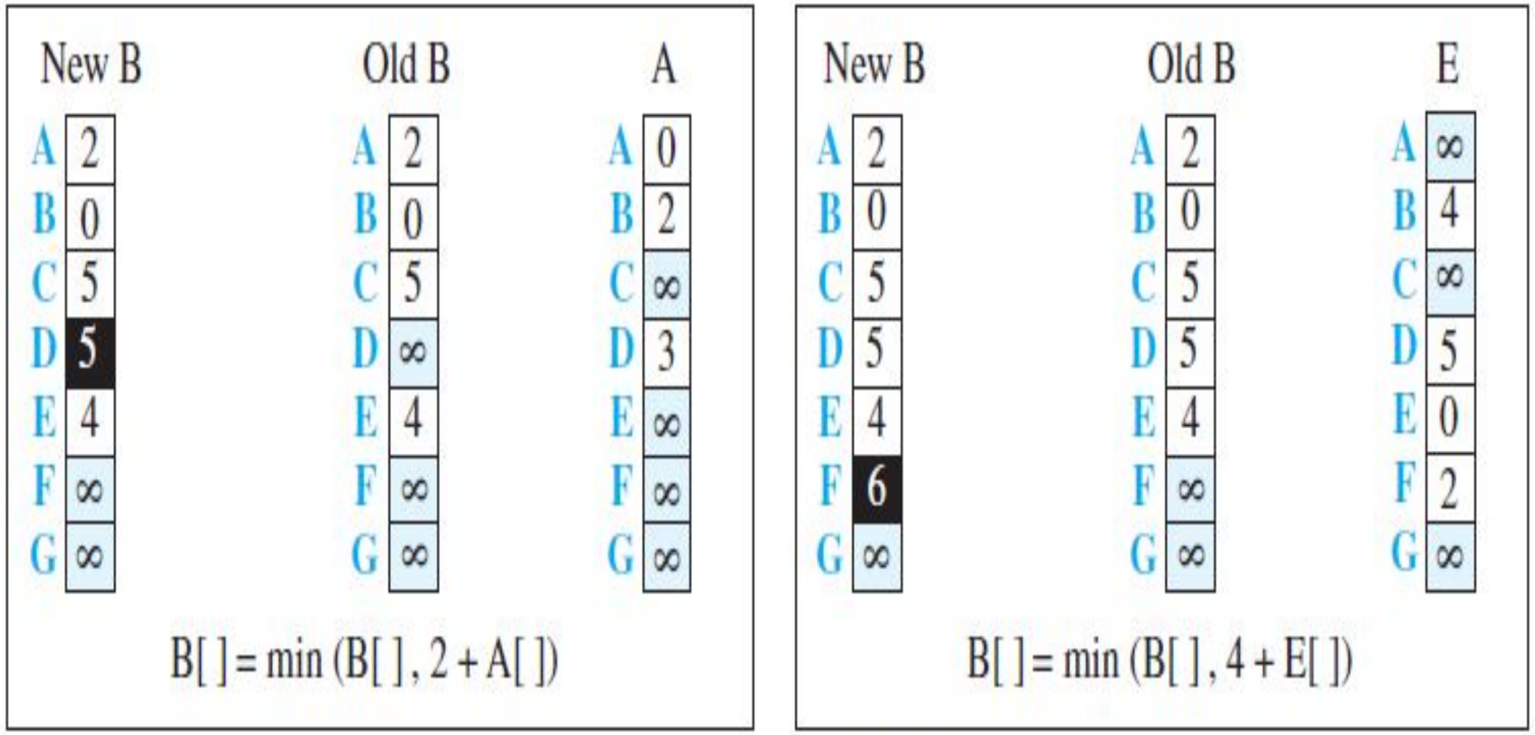
A	∞
B	∞
C	3
D	∞
E	∞
F	1
G	0

A	3
B	∞
C	∞
D	0
E	5
F	∞
G	∞

A	∞
B	4
C	∞
D	5
E	0
F	2
G	∞

A	∞
B	∞
C	4
D	∞
E	2
F	0
G	1

Updating Distance Vectors



	New B	Old B	A
A	2	2	0
B	0	0	2
C	5	5	∞
D	5	∞	3
E	4	4	∞
F	∞	∞	∞
G	∞	∞	∞

B[ ] = min (B[ ], 2 + A[ ])

a. First event: B receives a copy of A's vector.

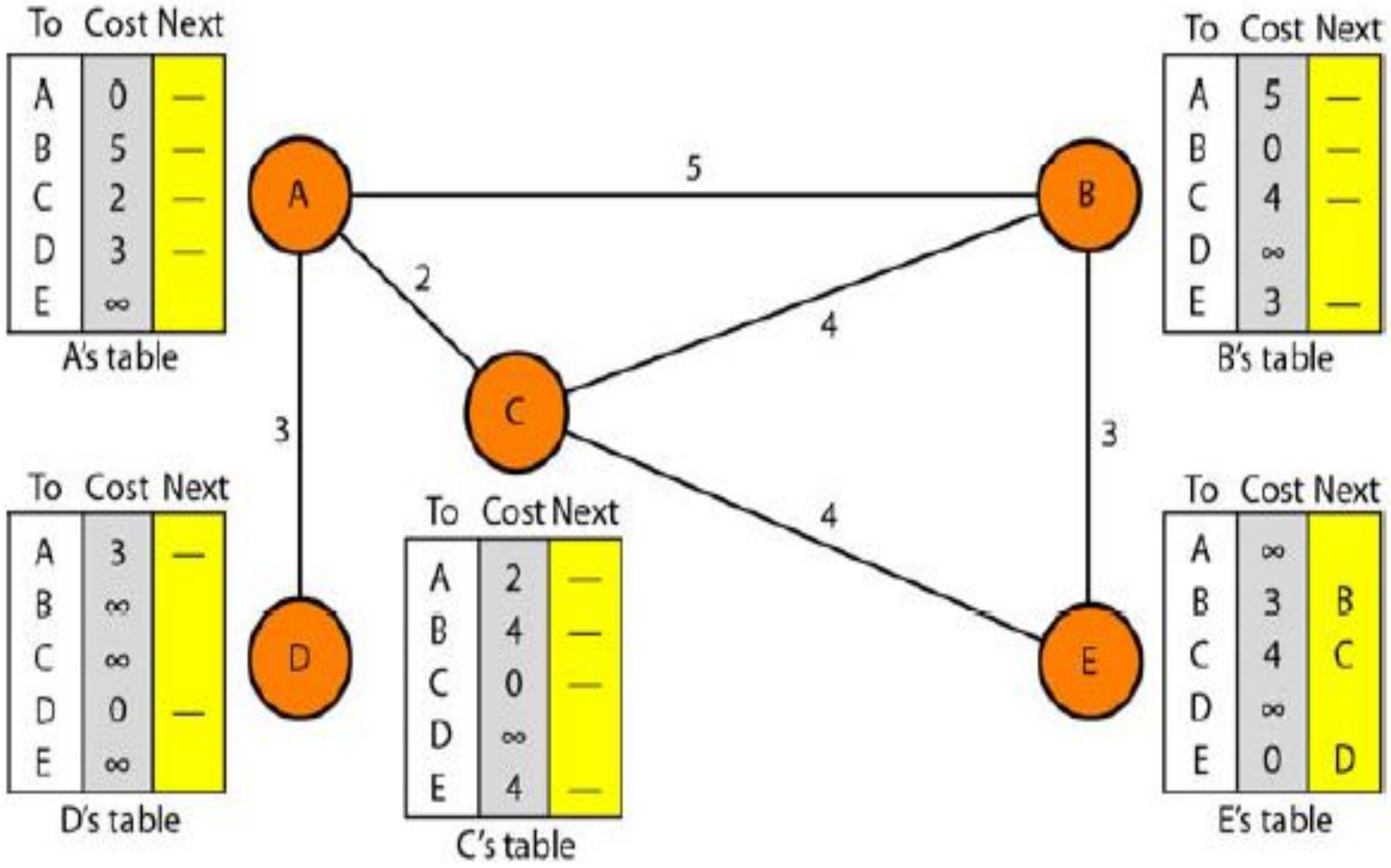
	New B	Old B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	6	∞	2
G	∞	∞	∞

B[ ] = min (B[ ], 4 + E[ ])

b. Second event: B receives a copy of E's vector.

Note:
X[ ]: the whole vector

Distance Vector Routing: Table Initialization



To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

A's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	∞	
E	3	—

B's table

To	Cost	Next
A	3	—
B	∞	
C	∞	
D	0	—
E	∞	

D's table

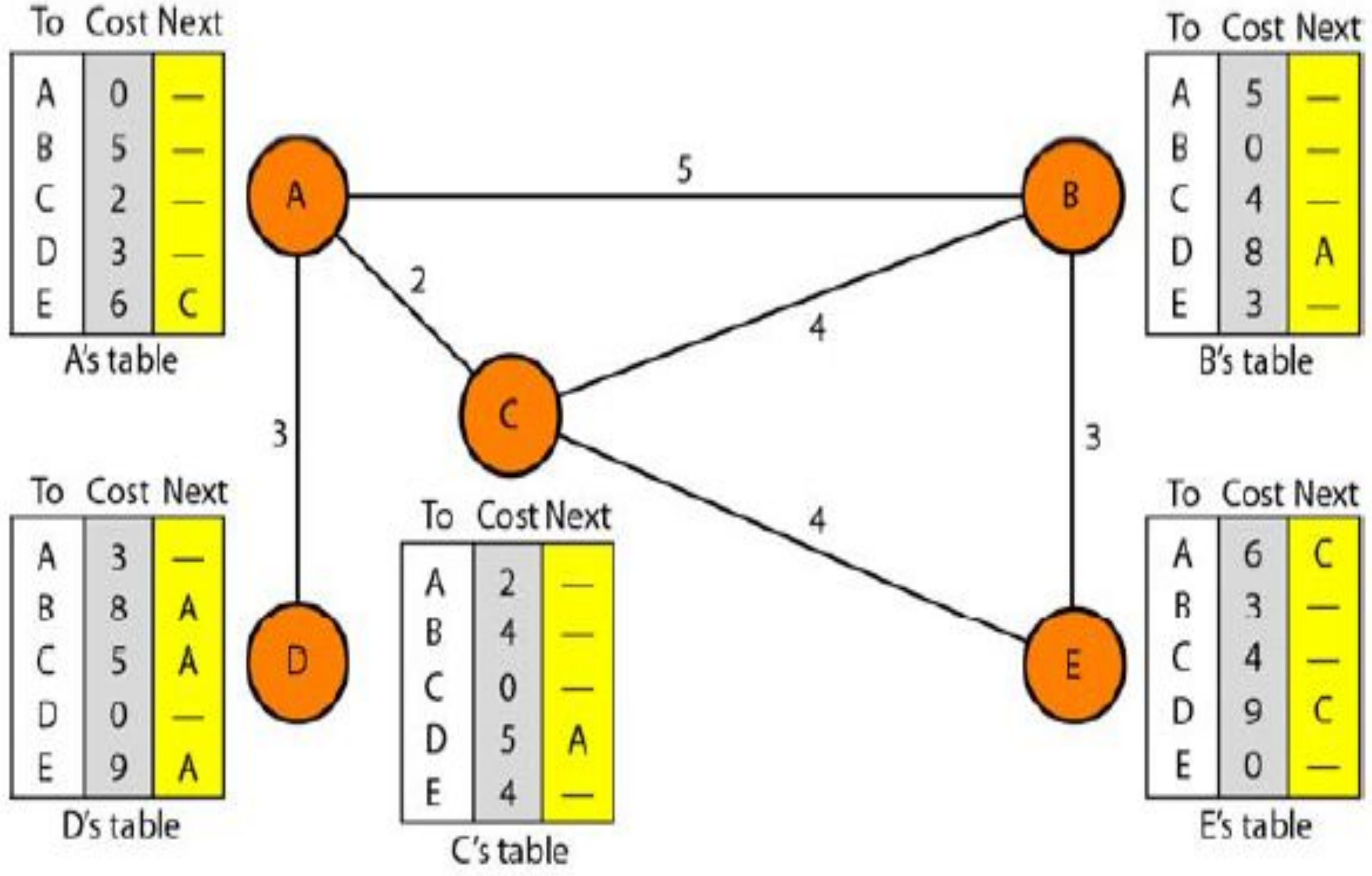
To	Cost	Next
A	2	—
B	4	—
C	0	—
D	∞	
E	4	—

C's table

To	Cost	Next
A	∞	
B	3	B
C	4	C
D	∞	
E	0	D

E's table

Distance Vector Routing: Sharing



To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

A's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	8	A
E	3	—

B's table

To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

D's table

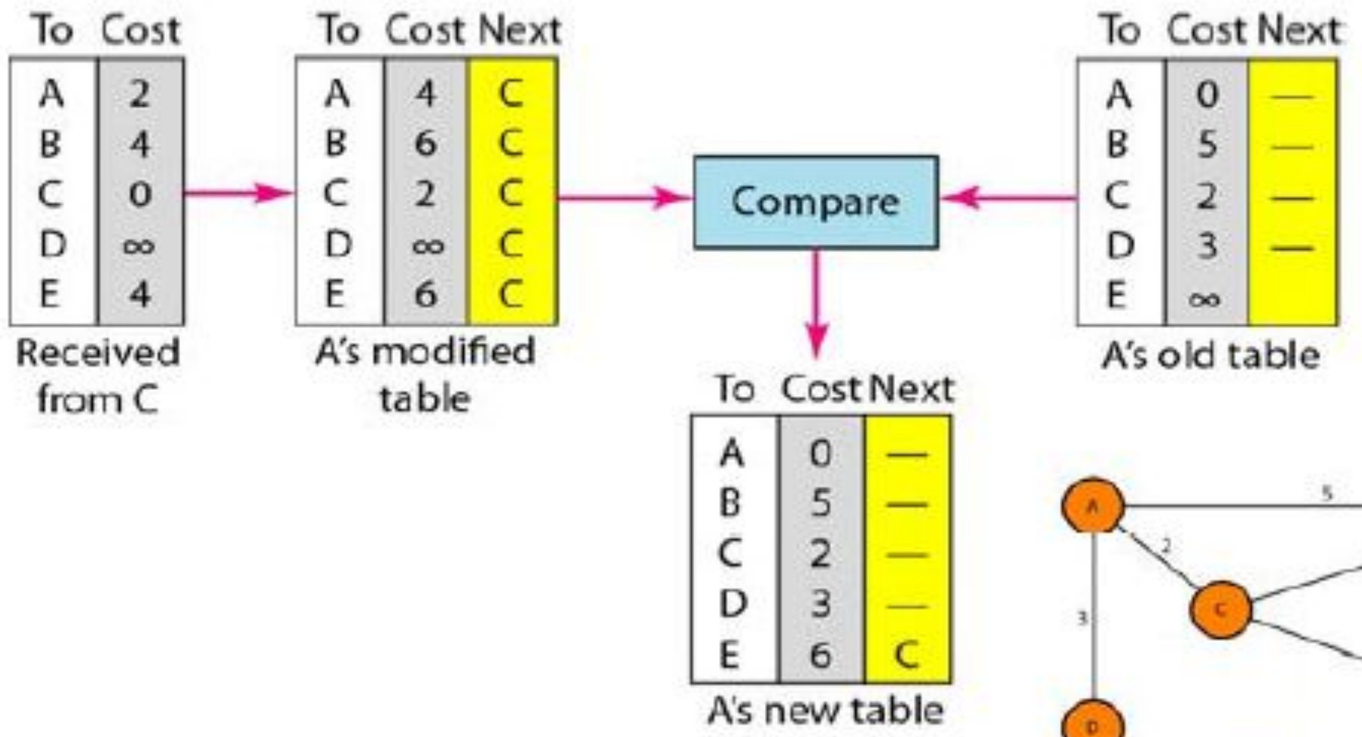
To	Cost	Next
A	2	—
B	4	—
C	0	—
D	5	A
E	4	—

C's table

To	Cost	Next
A	6	C
B	3	—
C	4	—
D	9	C
E	0	—

E's table

Distance Vector Routing: Updating



To	Cost
A	2
B	4
C	0
D	∞
E	4

Received from C

To	Cost	Next
A	4	C
B	6	C
C	2	C
D	∞	C
E	6	C

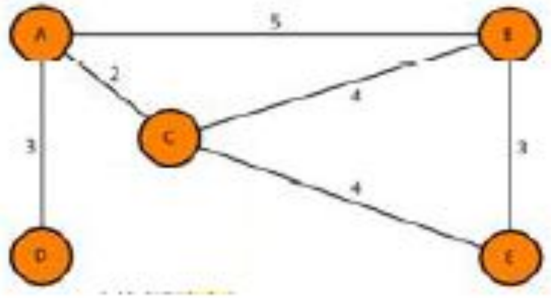
A's modified table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

A's old table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

A's new table



In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change

Updating Routing Table

- **If the next-node entry is different**
  - The receiving node chooses the row with the smaller cost
  - If there is a tie, the old one is kept

- **If the next-node entry is the same**
  - i.e. the sender of the new row is the provider of the old entry
  - The receiving node chooses the new row, even though the new value is infinity.

When to Share

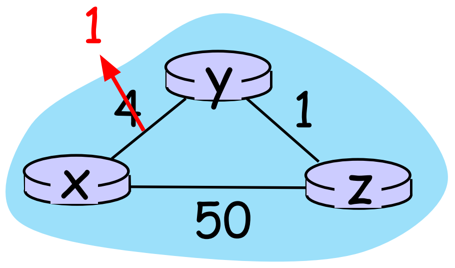
- **Periodic Update**
  - A node sends its routing table, normally 30 seconds, in a periodic update

- **Triggered Update**
  - A node sends its routing table to its neighbors any time when there is a change in its routing table
    - 1. After updating its routing table, or
    - 2. Detects some failure in the neighboring links

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.

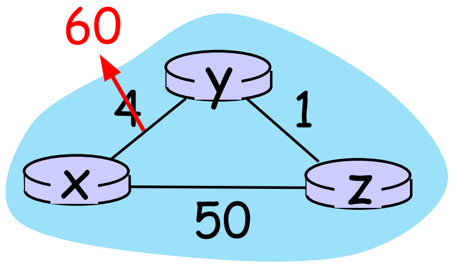
t_1 : *z* receives update from *y*, updates its DV, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its DV. *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!*



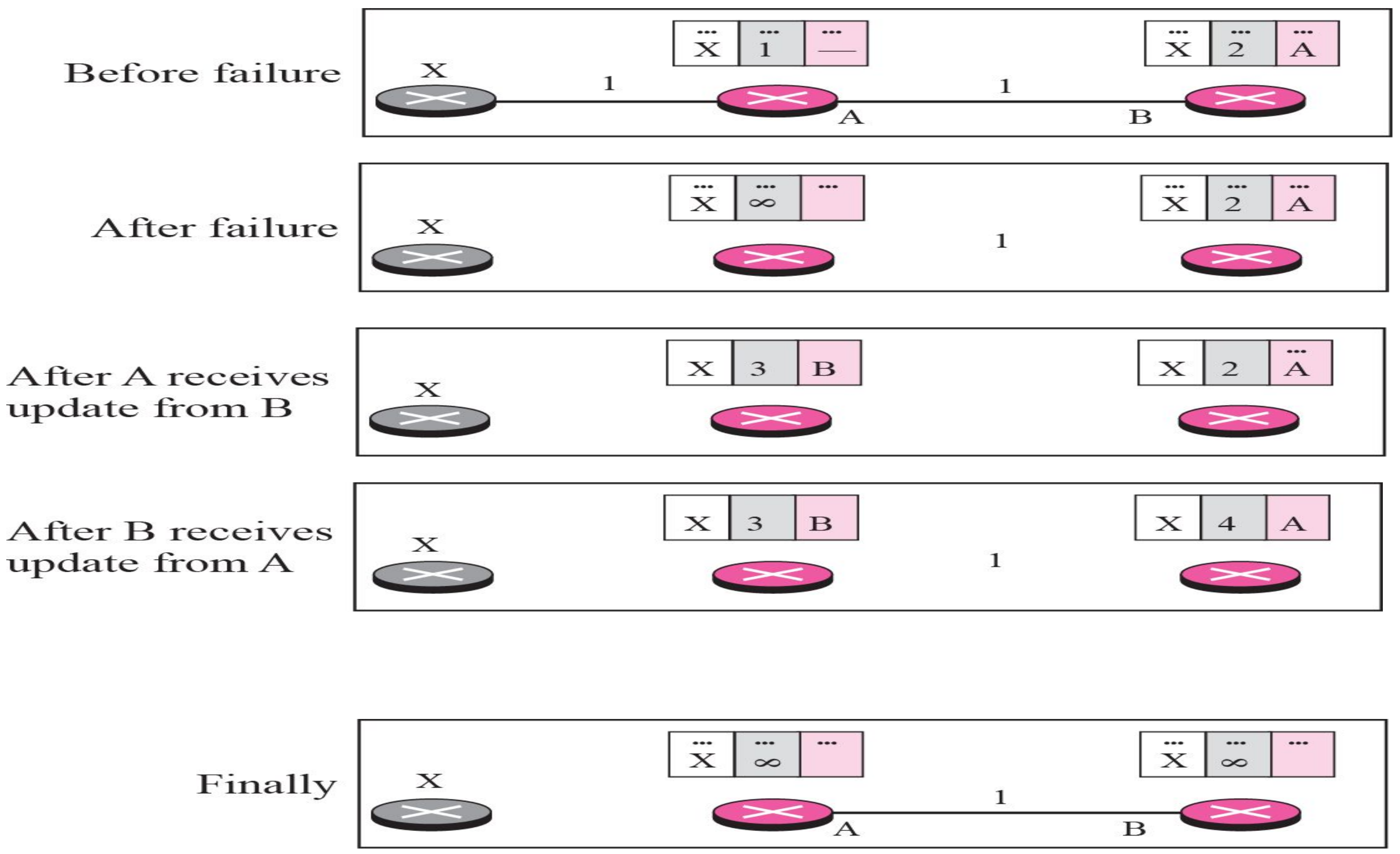
Count to Infinity

- A problem with distance-vector routing is that
  - any decrease in cost (good news) propagates quickly,
  - but any increase in cost (bad news) will propagate slowly.

- For a routing protocol to work properly, if a link is broken (cost becomes infinity),
  - every other router should be aware of it immediately,
  - but in distance-vector routing, this takes some time.

- **The problem is referred to as *count to infinity***
- It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

Count to Infinity: Two node loop

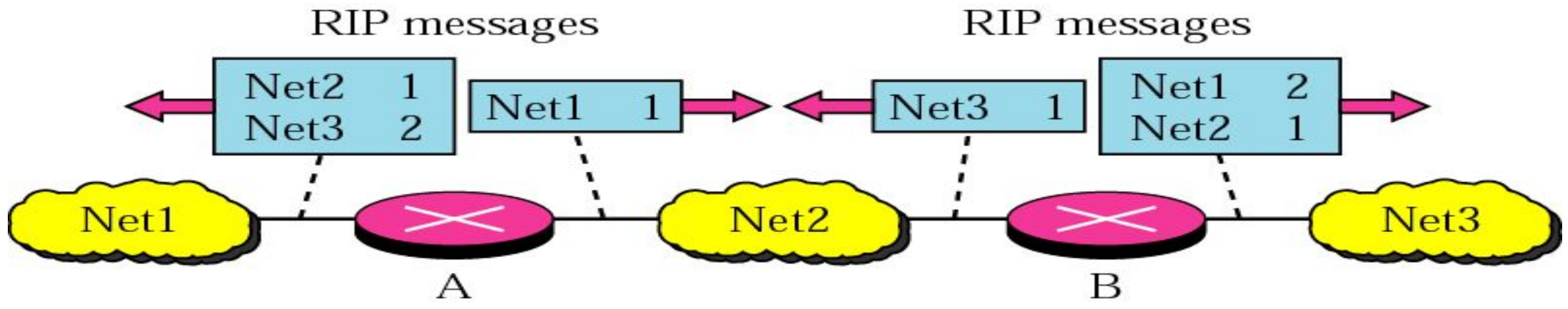


Most implementations define 16 as infinity

Solutions for Two-Node Instability

Split Horizon

- Split horizon is a method of preventing a routing loop in a network.
- **Simple principle:** Information about the routing for a particular packet is never sent back in the direction from which it was received.
- So, instead of flooding the table through each interface, each node sends only part of its table through each interface



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:

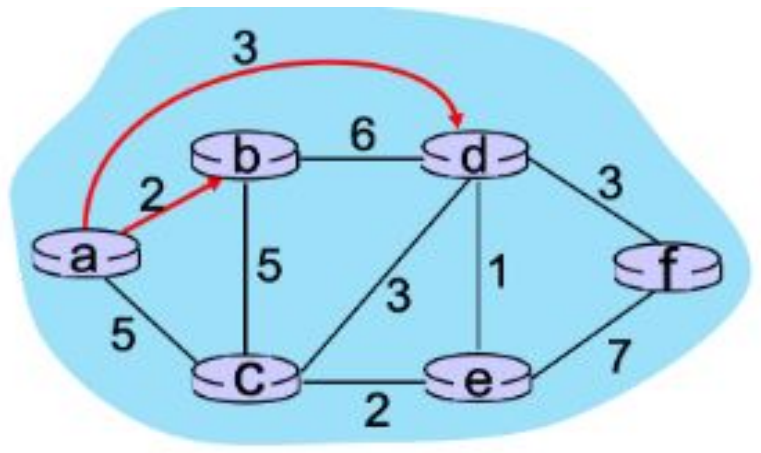
- DV router can advertise incorrect *path* cost (“I have a *really* low-cost path to everywhere”): *black-holing*
- each router’s DV is used by others: error propagate thru network

DIJKSTRA’S LINK-STATE ROUTING ALGORITHM.

Consider the graph shown below and the use of Dijkstra’s algorithm to compute a least cost path from a to all destinations. Suppose that nodes b and d have already been added to N’.

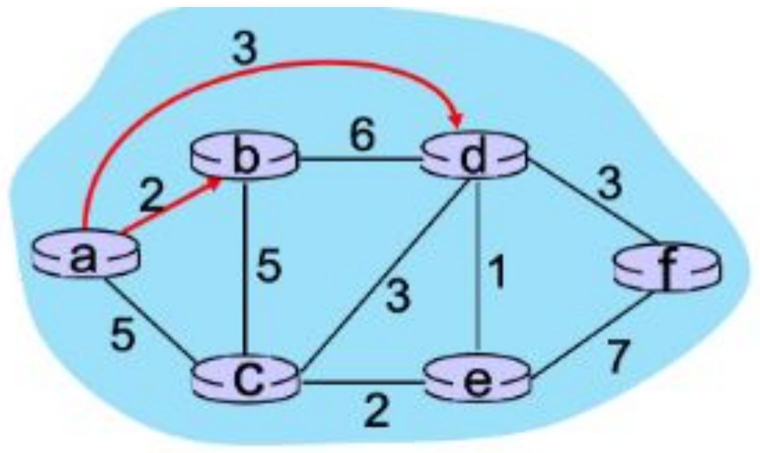
What is the next node to be added to N’?

What is the *path cost* to the next node to be added to N'



DIJKSTRA'S LINK-STATE ROUTING ALGORITHM.

Consider the graph shown below and the use of Dijkstra's algorithm to compute a least cost path from a to all destinations. Suppose that nodes b and d have already been added to N'. What is the next node to be added to N'?



Distance vector: another example

$D_x()$

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

$D_y()$

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

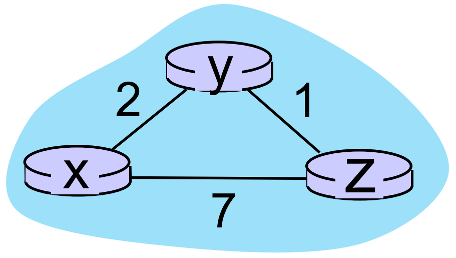
$D_z()$

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

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

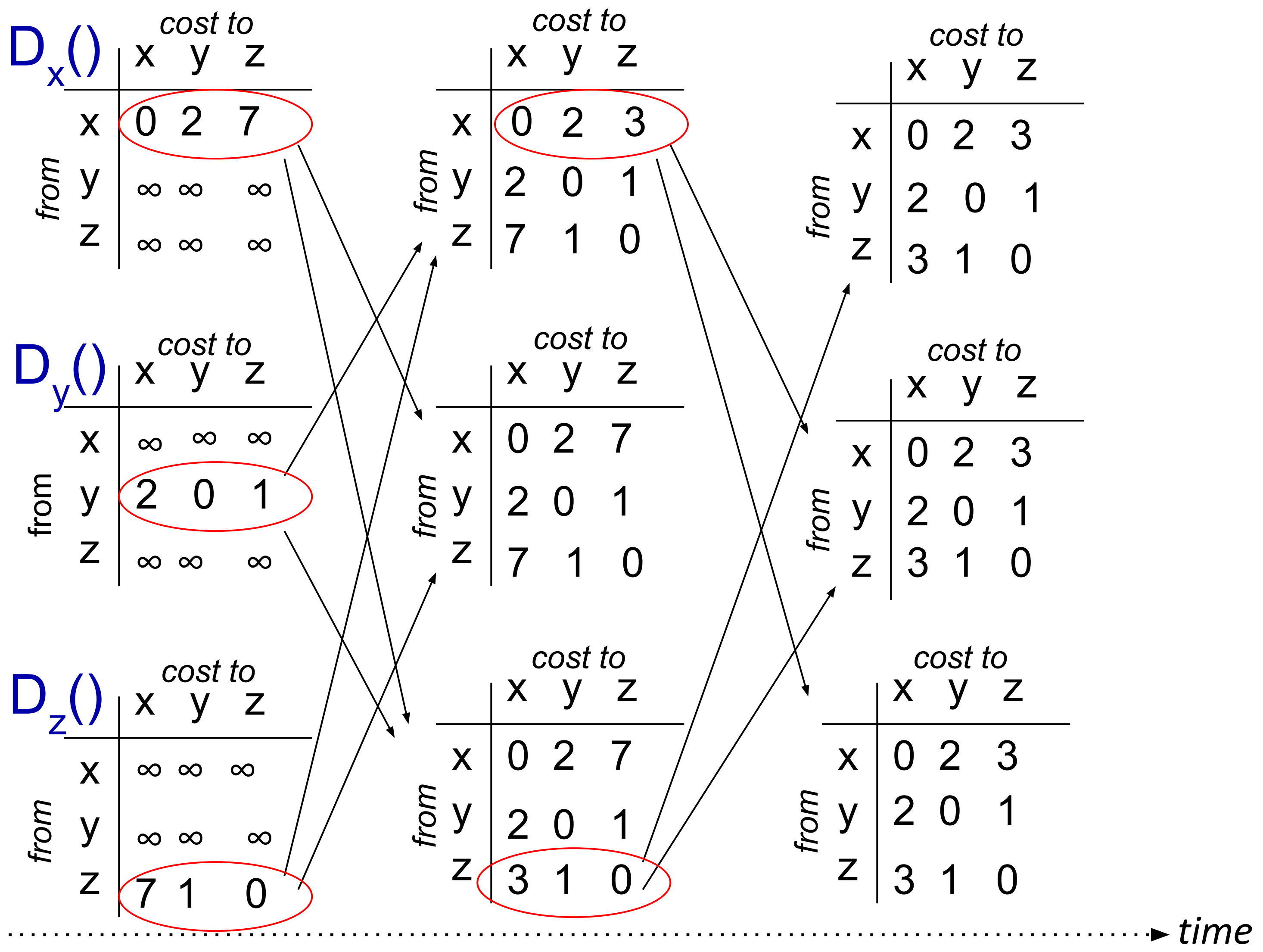
$$D_x(z) = \min\{c_{x,y} + D_y(z),\ c_{x,z} + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

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



time

Distance vector: another example



$D_x()$

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

$D_y()$

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

$D_z()$

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

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

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

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

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

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

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