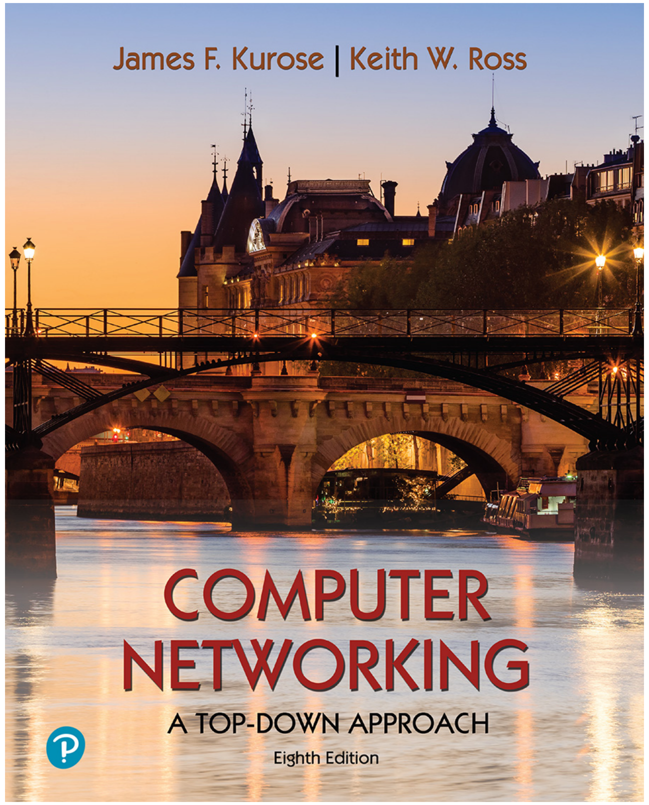
Chapter 5
Network Layer: Control Plane



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

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

IMPORTANT

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

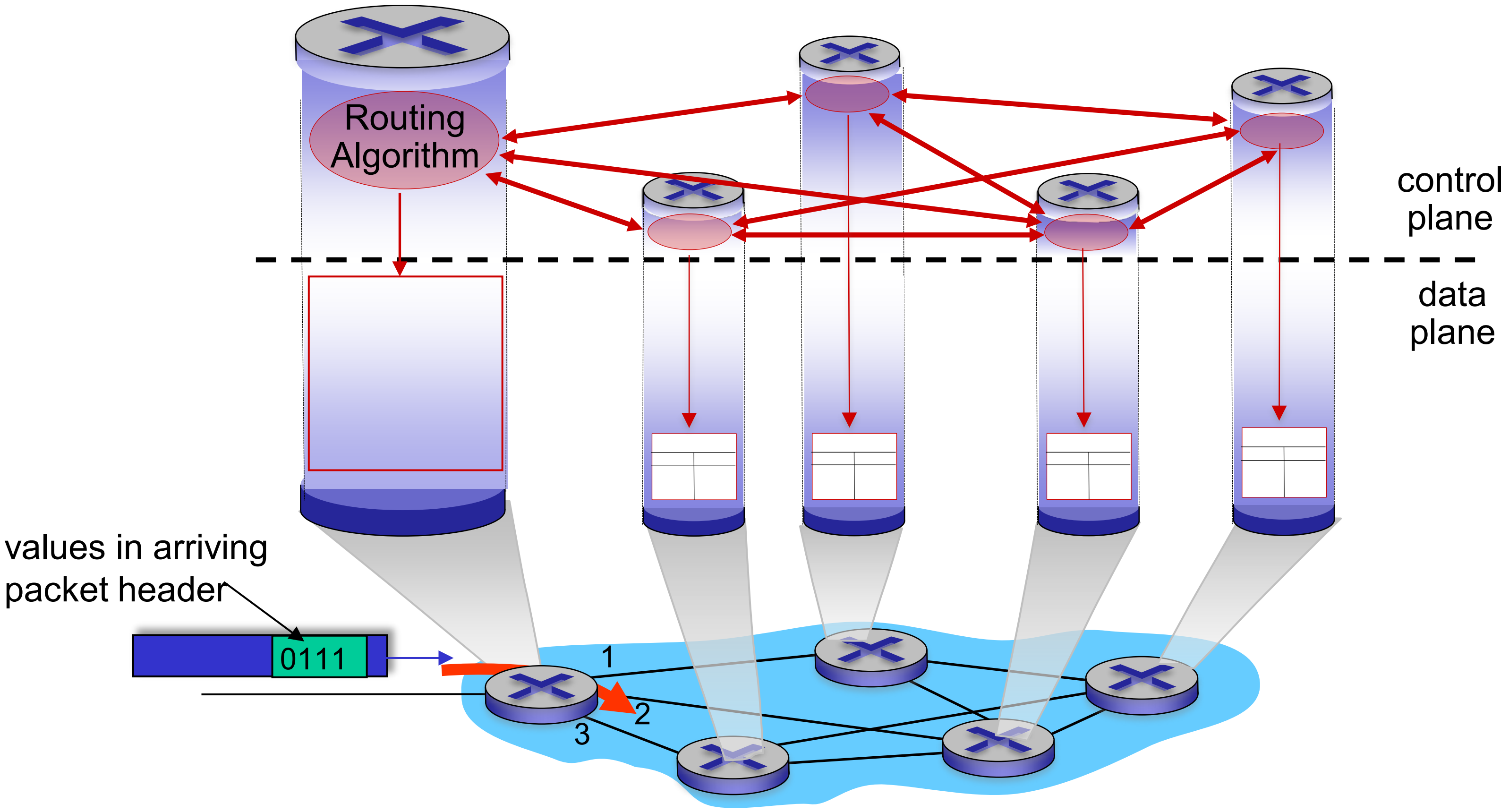
Two approaches to structuring network control plane:

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

IMPORTANT

Per-router control plane

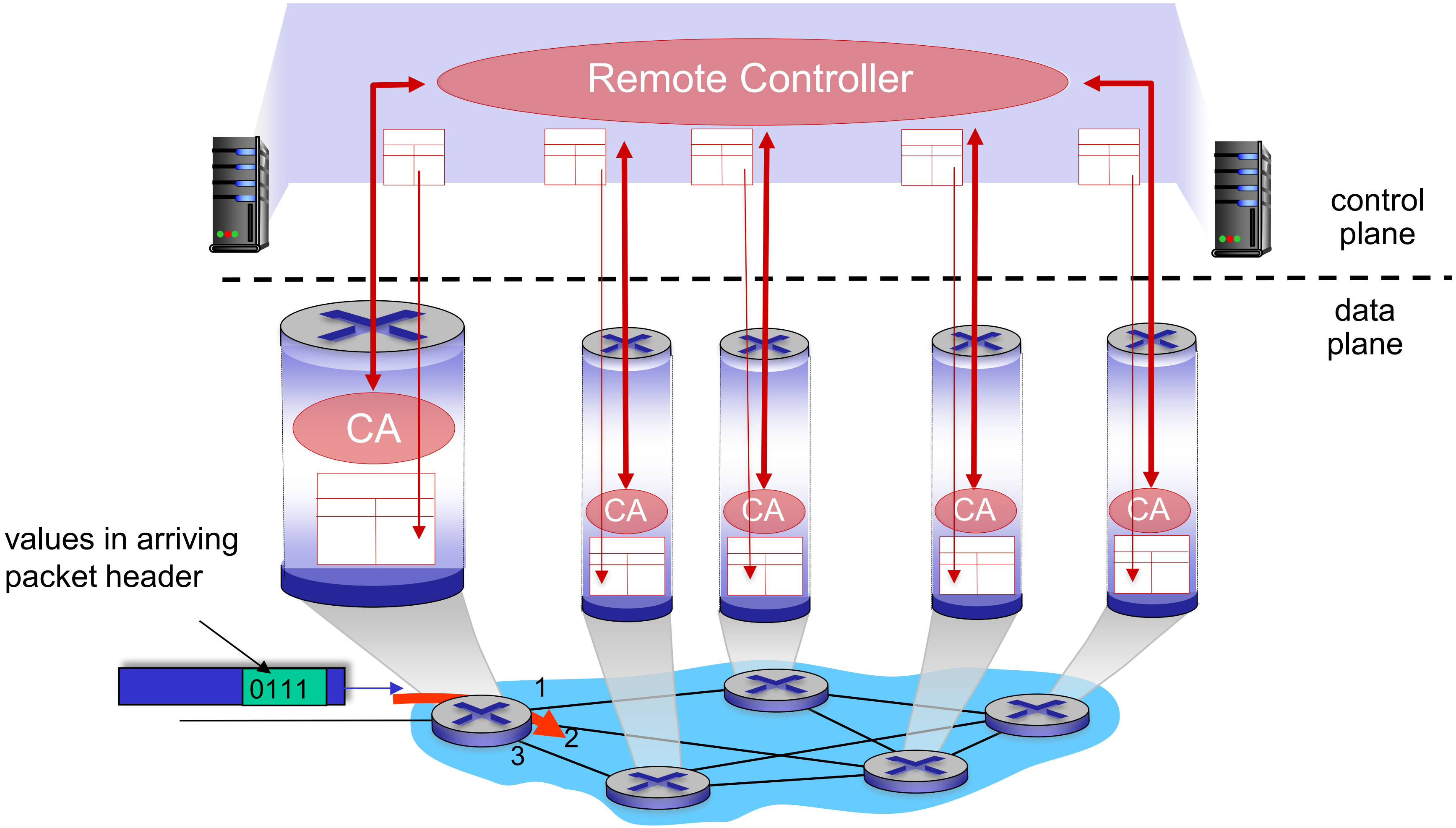
Individual routing algorithm components *in each and every router* interact in the control plane



IMPORTANT

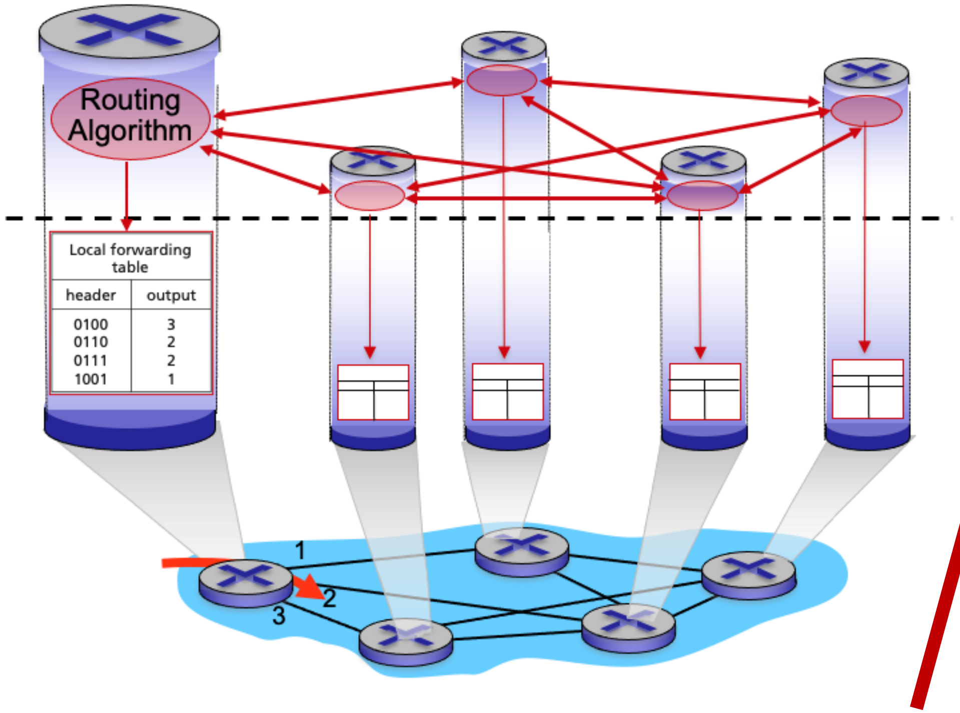
Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



IMPORTANT

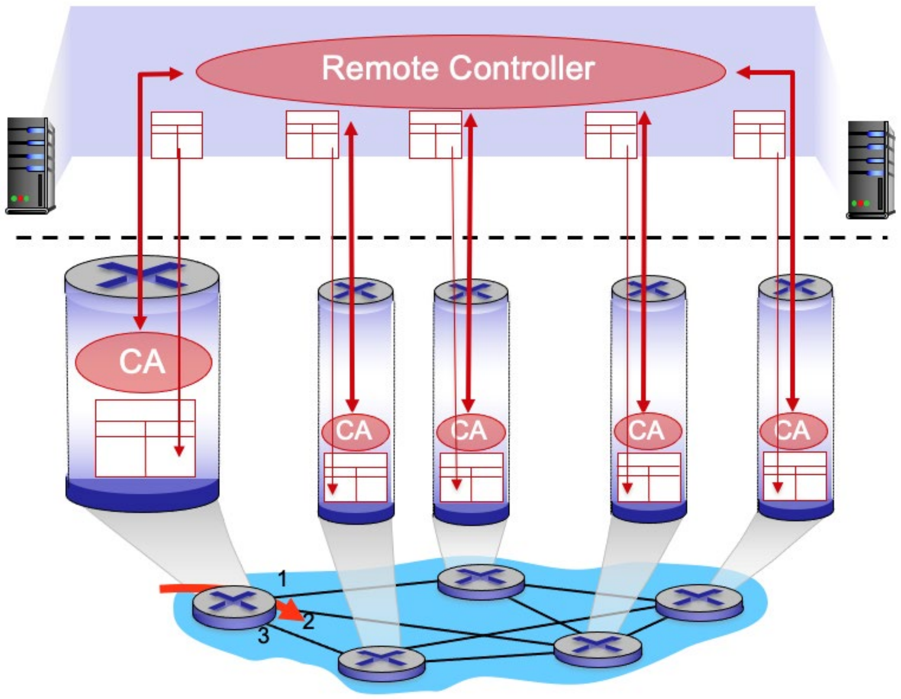
Per-router control plane



Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

SDN control plane



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

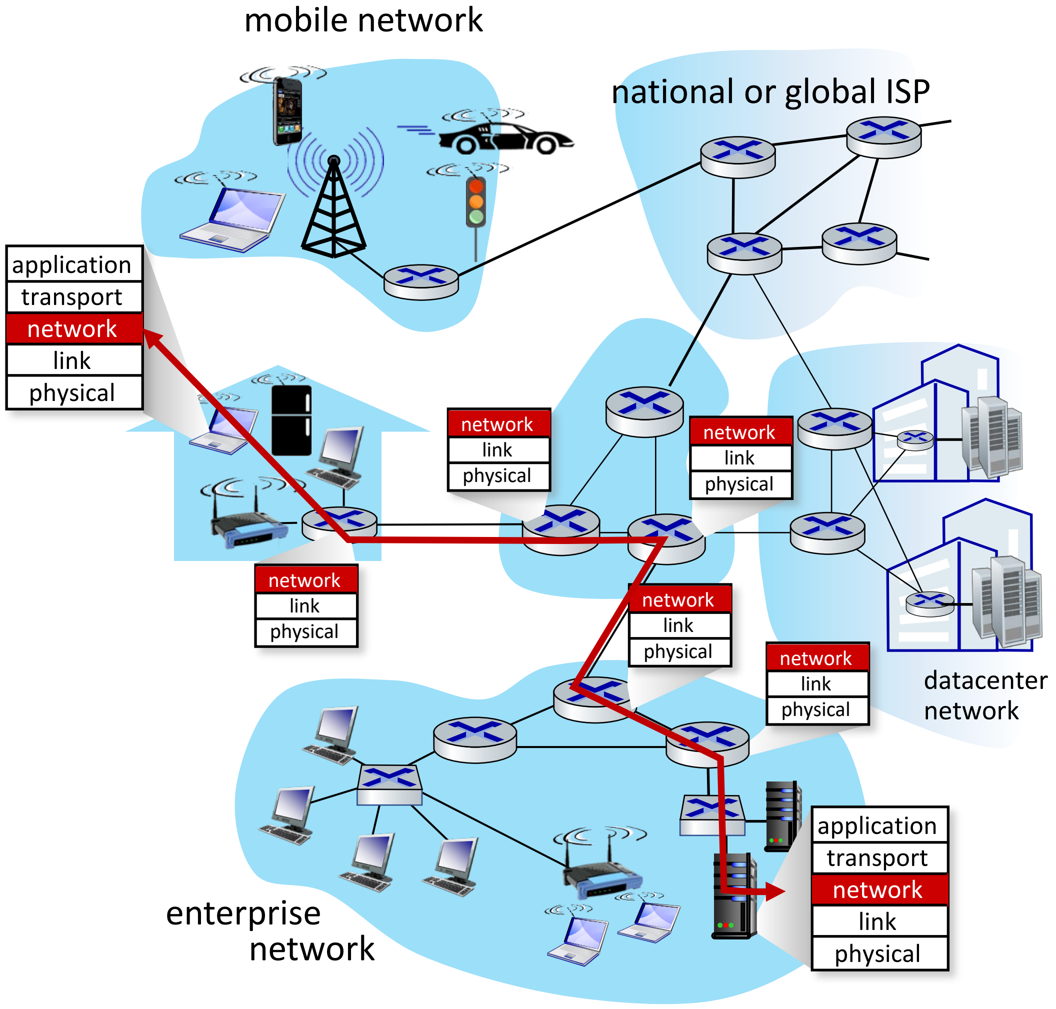
- network management, configuration
  - SNMP
  - NETCONF/YANG

IMPORTANT

Routing protocols

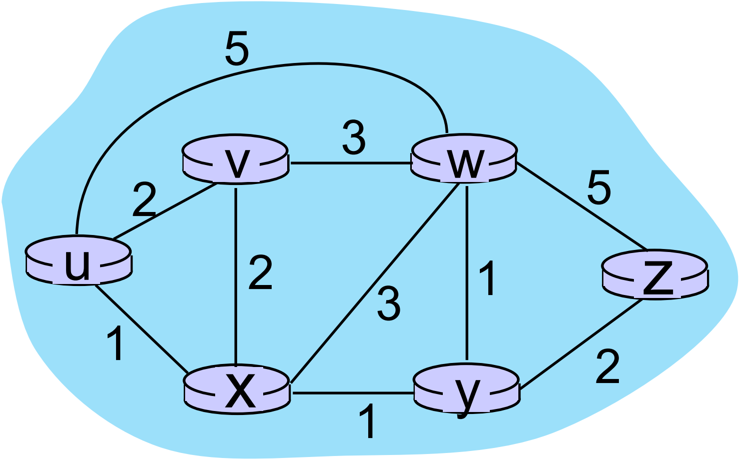
Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- 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!



IMPORTANT

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

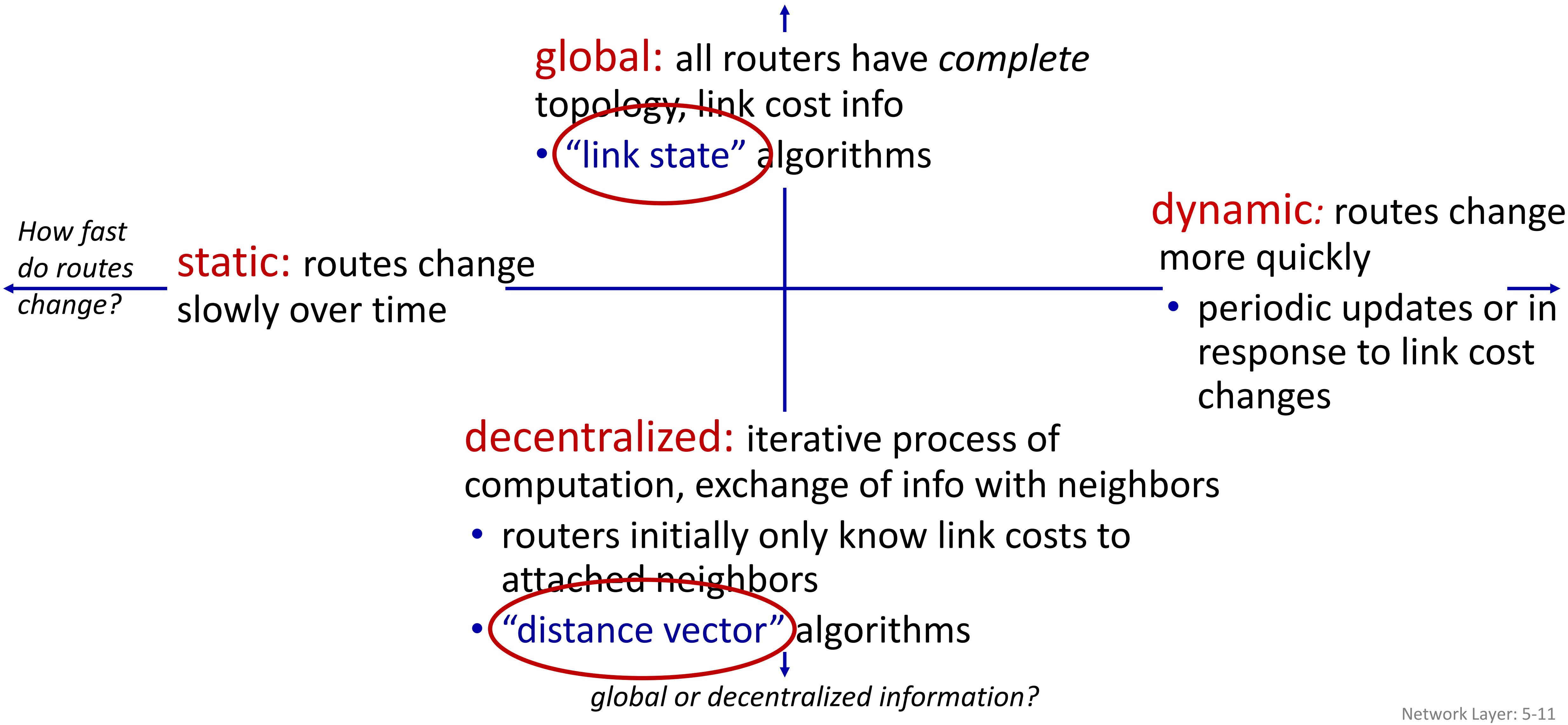
graph: $G = (N,E)$

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)* }

IMPORTANT

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

IMPORTANT

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*; = ∞ 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

IMPORTANT

Dijkstra’s link-state routing algorithm

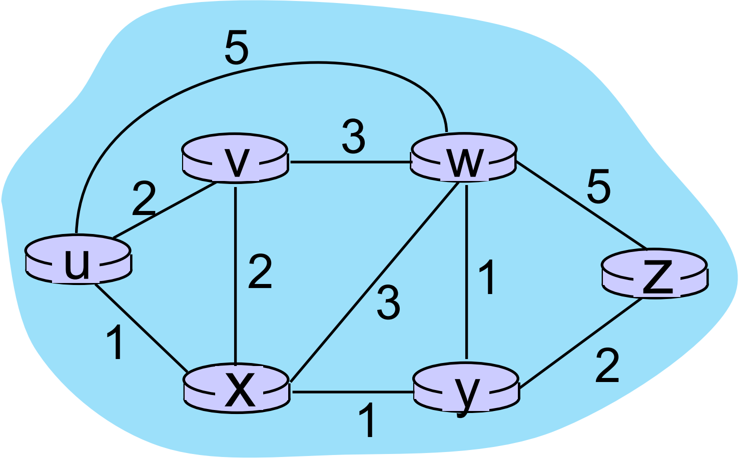
```
Initialization:
  $N' = \{u\}$                          /* compute least cost path from u to all other nodes */
  for all nodes $v$
    if $v$ adjacent to $u$              /* $u$ initially knows direct-path-cost only to  direct neighbors    */
        then $D(v) = c_{u,v}$           /* but may not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

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

IMPORTANT

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						



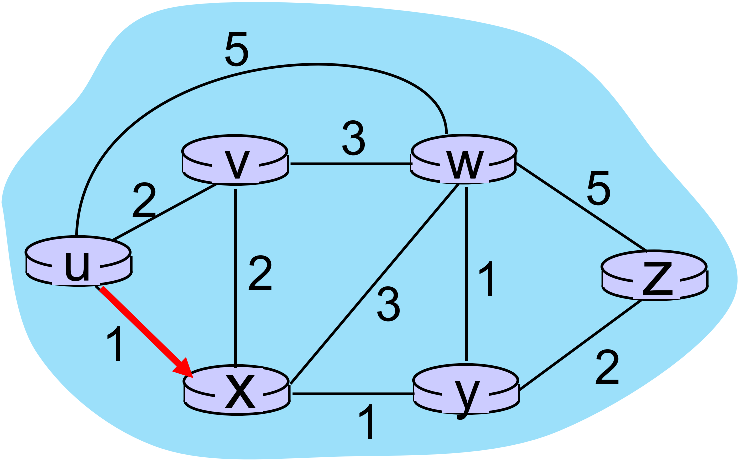
Initialization (step 0):

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

IMPORTANT

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						
3						
4						
5						

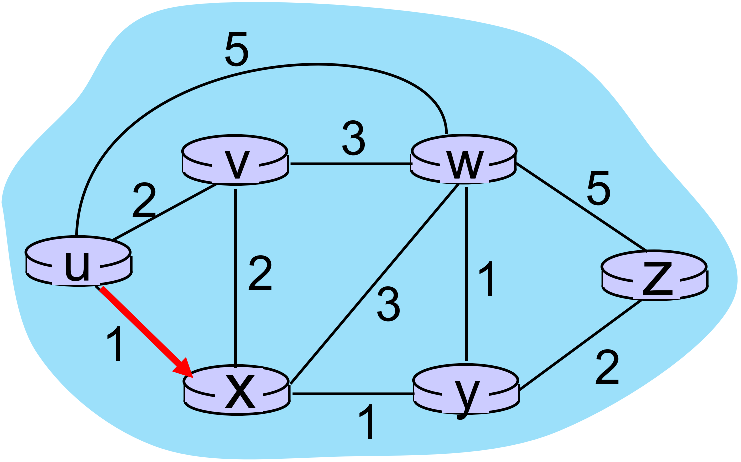


```
Loop
    find a not in N′ such that D(a) is a minimum
    add a to N′
```

IMPORTANT

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						



```
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 $N'$ :
        $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

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

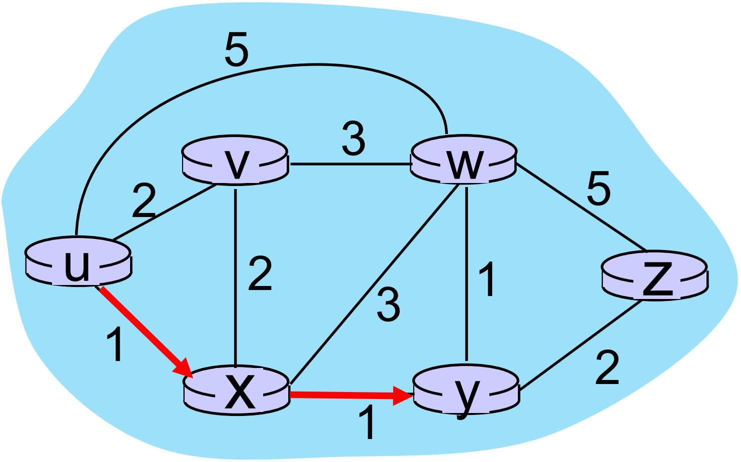
$D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$

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

IMPORTANT

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						

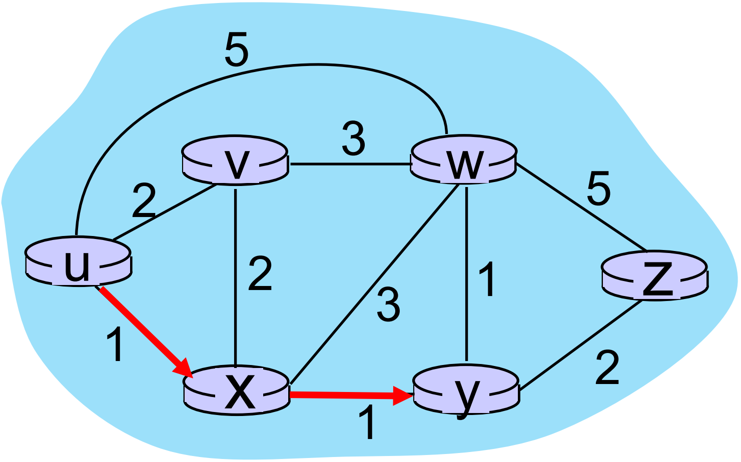


```
Loop
    find $a$ not in $N'$ such that $D(a)$ is a minimum
    add $a$ to $N'$
```

IMPORTANT

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						
4						
5						



```
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 $N'$ :
        $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

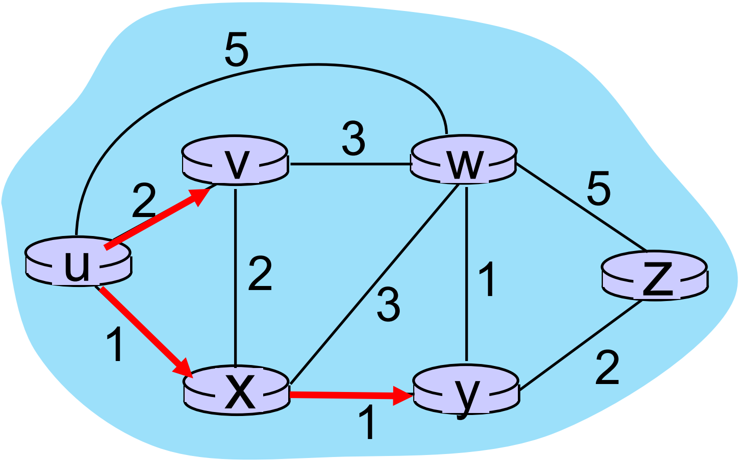
$D(w) = \min (D(w), D(y) + c_{y,w}) = \min (4, 2+1) = 3$

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

IMPORTANT

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					
4						
5						

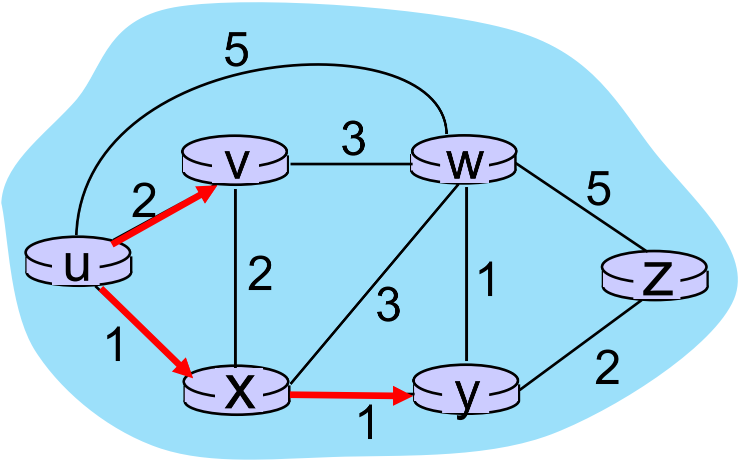


```
8   Loop
9       find $a$ not in $N'$ such that $D(a)$ is a minimum
10      add $a$ to $N'$
```

IMPORTANT

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						



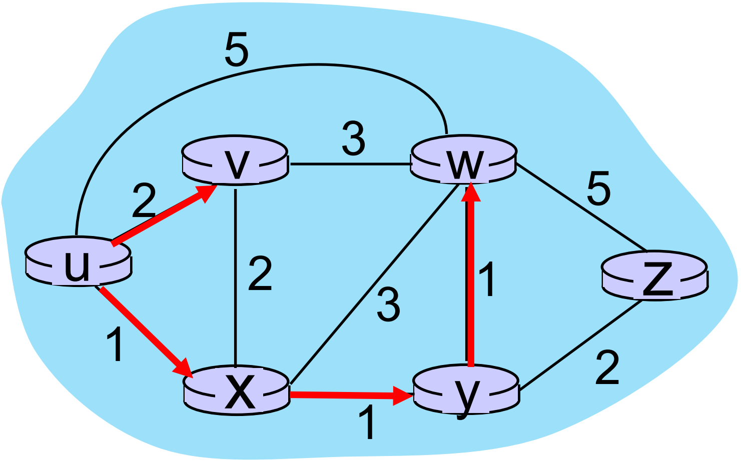
```
Loop
    find $a$ not in $N'$ such that $D(a)$ is a minimum
    add $a$ to $N'$
    update $D(b)$ for all $b$ adjacent to $a$ and not in $N'$ :
        $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

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

IMPORTANT

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					
5						

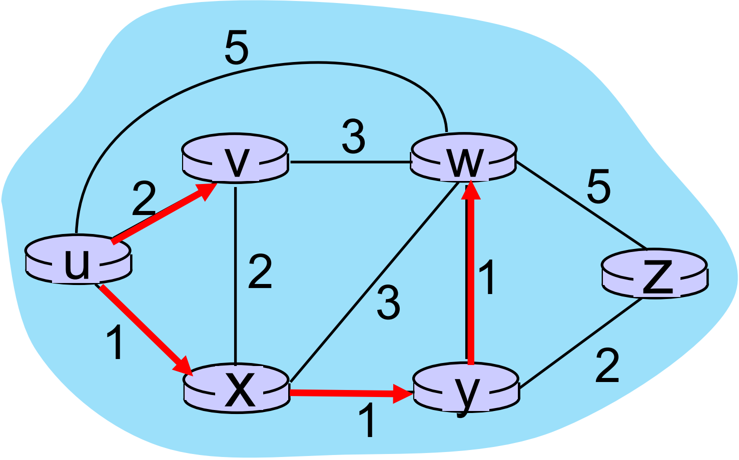


```
Loop
    find $a$ not in $N'$ such that $D(a)$ is a minimum
    add $a$ to $N'$
```

IMPORTANT

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						



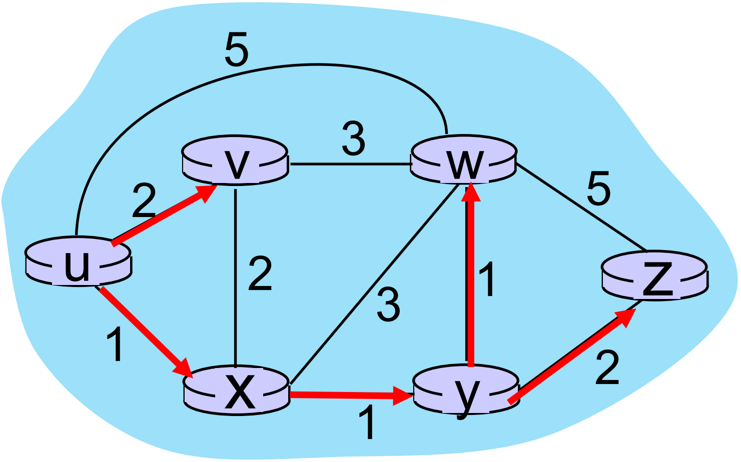
```
Loop
    find a not in N' such that D(a) is a minimum
    add a to N'
    update D(b) for all b adjacent to a and not in N' :
        D(b) = min ( D(b), D(a) + c_{a,b} )
```

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

IMPORTANT

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					

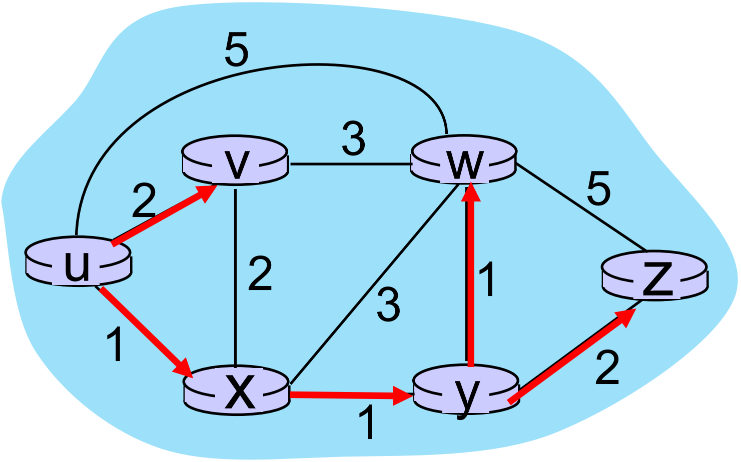


```
Loop
    find a not in N' such that D(a) is a minimum
    add a to N'
```

IMPORTANT

Dijkstra’s algorithm: an example

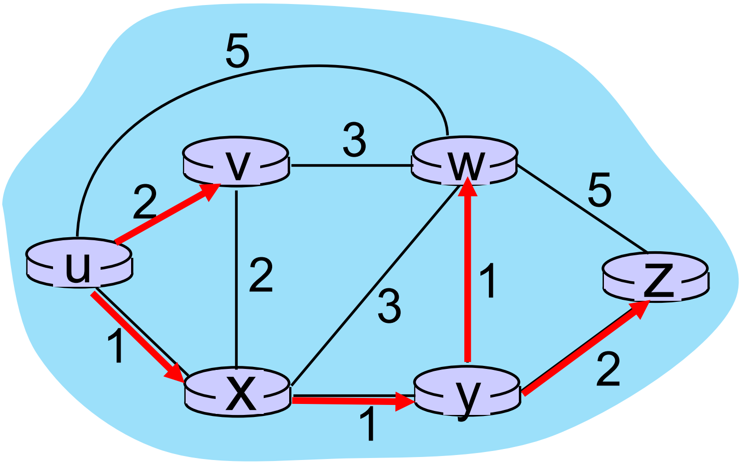
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



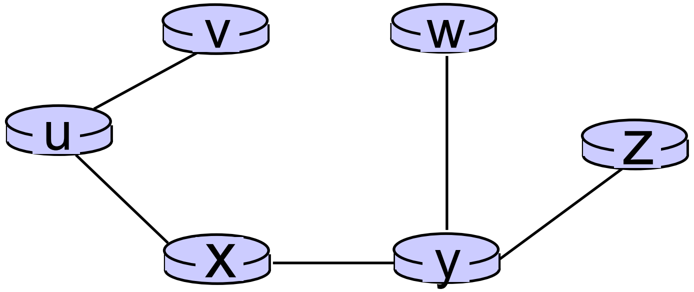
```
Loop
    find a not in N' such that D(a) is a minimum
    add a to N'
    update D(b) for all b adjacent to a and not in N' :
        $D(b) = \min ( D(b), D(a) + c_{a,b} )$
```

IMPORTANT

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)

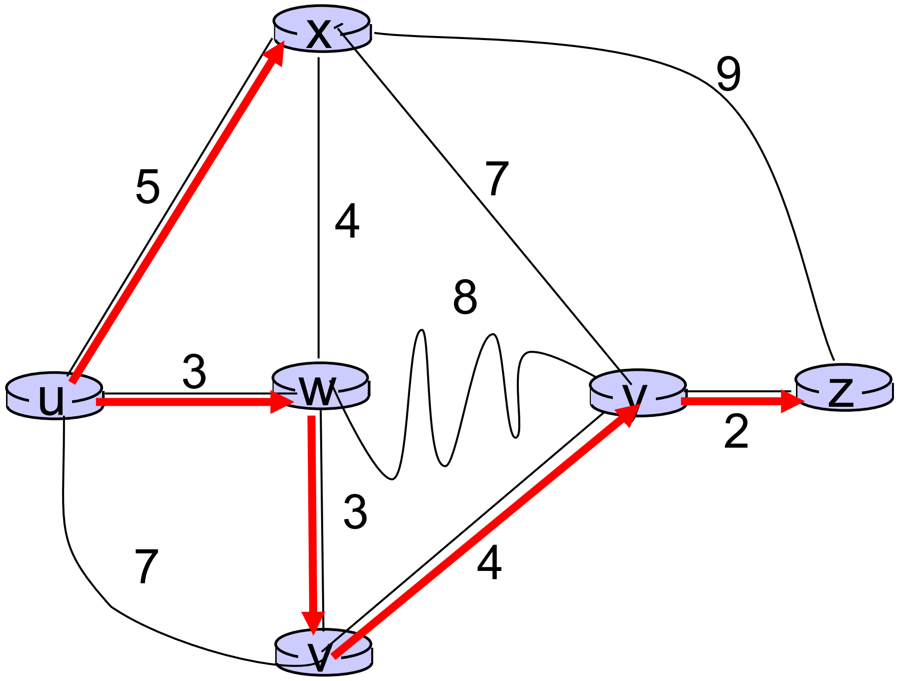
route from *u* to *v* directly

route from u to all other destinations via *x*

IMPORTANT

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)

IMPORTANT

Dijkstra’s algorithm: discussion

algorithm complexity: *n* nodes

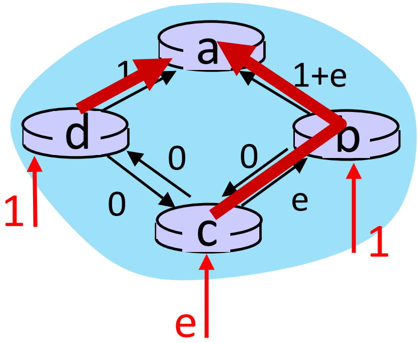
- each of *n* iteration: need to check all nodes, *w*, not in *N*
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

message complexity:

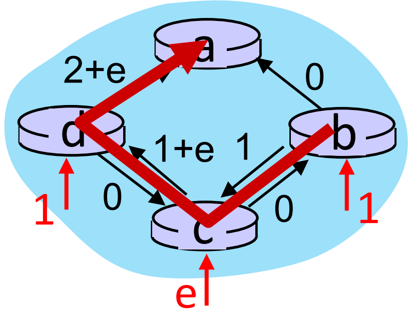
- each router must *broadcast* its link state information to other *n* routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router’s message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Dijkstra’s algorithm: oscillations possible

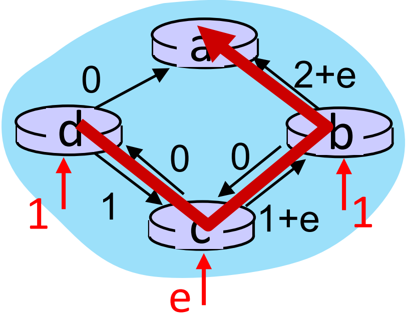
- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - link costs are directional, and volume-dependent



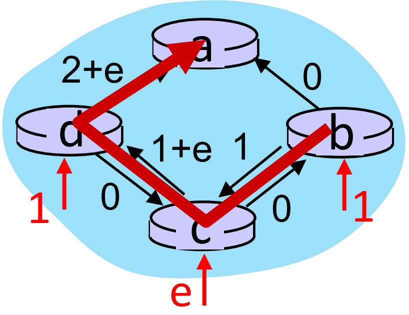
initially



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” 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

IMPORTANT

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

IMPORTANT

Bellman-Ford (BF) equation

- Recall edge relaxation for the one edge connecting v and y:
  - $D_x(y) = \min \{ D_x(y), c_{x,v} + D_v(y) \}$
- Perform edge relaxation for all vertices v connected to x, we have the B-F equation
  - $D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$
- In L. 5.0 we centralized global synchronous version of BF algorithm, where all edges are relaxed in each iteration. Here we consider decentralized asynchronous version of BF algorithm.

IMPORTANT

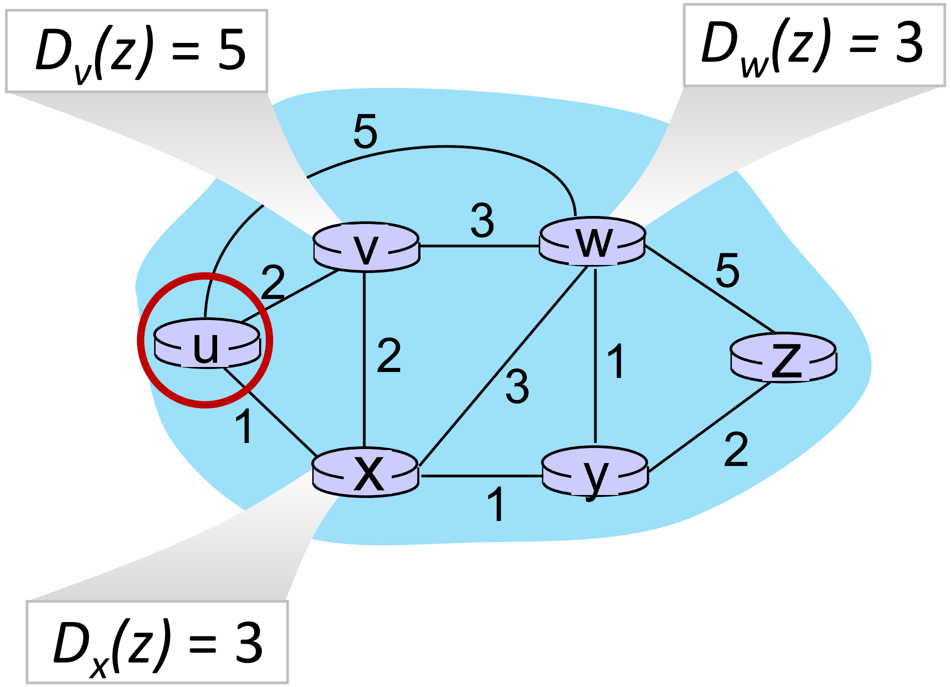
Bellman-Ford Algorithm

- Each router maintains a Distance Vector table containing the distance between itself and All possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.
- Information kept by DV router:
  - Each router has an ID
  - Associated with each link connected to a router, there is a link cost (static or dynamic).
  - Intermediate hops
- Distance Vector Table Initialization:
  - Distance to itself = 0
  - Distance to ALL other routers = ∞

IMPORTANT

Bellman-Ford Example

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



Bellman-Ford equation says:

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

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

IMPORTANT

Distance vector algorithm

key idea:

- Decentralized gossip algorithm based on local information: “I tell my neighbors, you tell yours.”
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when *x* receives new DV estimate from any neighbor *v*, it updates its own DV using B-F equation:

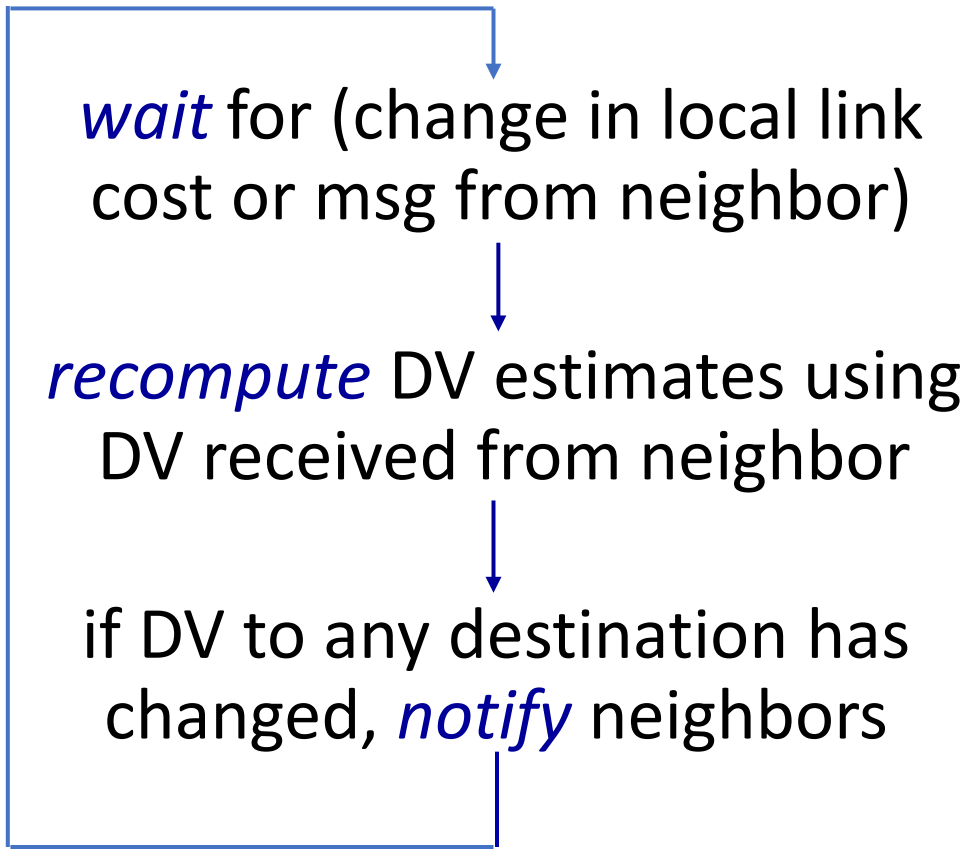
$$D_x(y) \leftarrow min_v\{c_{x,v} + D_v(y)\} \text{ for each node } y \in N$$

- under minor conditions, the estimate $D_x(y)$ *converge to the actual least cost* $\mathrm{d}_\mathrm{x}(\mathrm{y})$

IMPORTANT

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

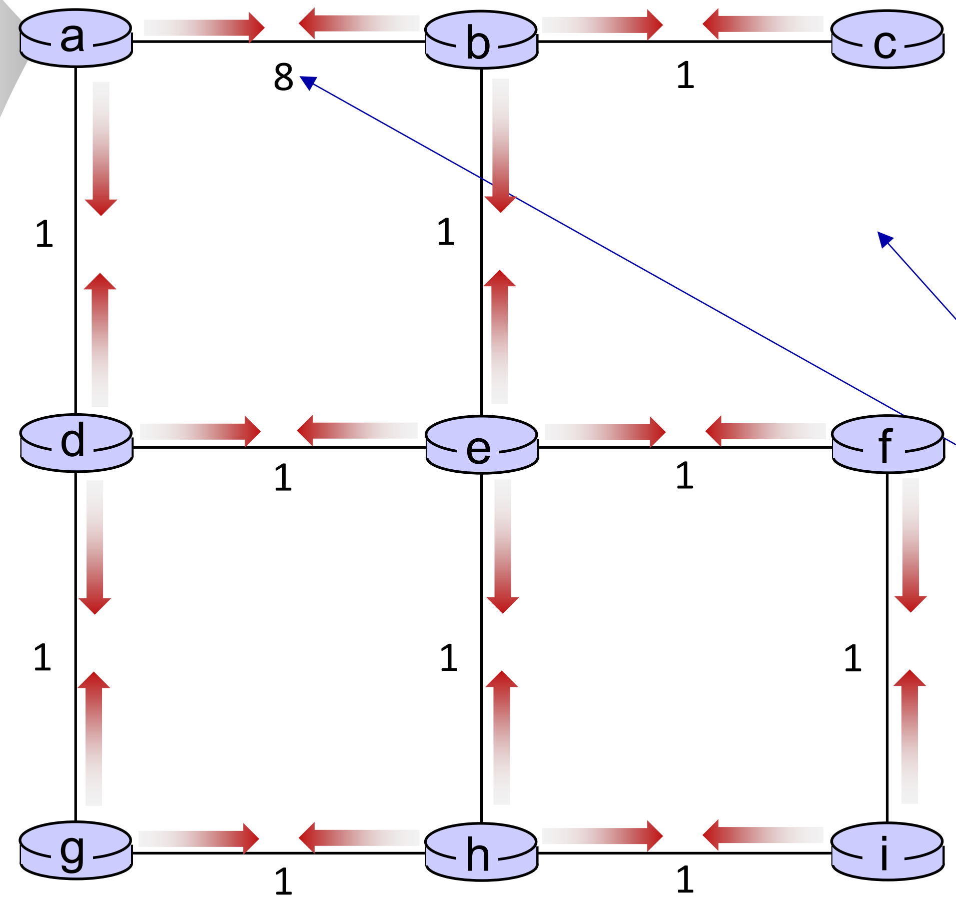
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

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:

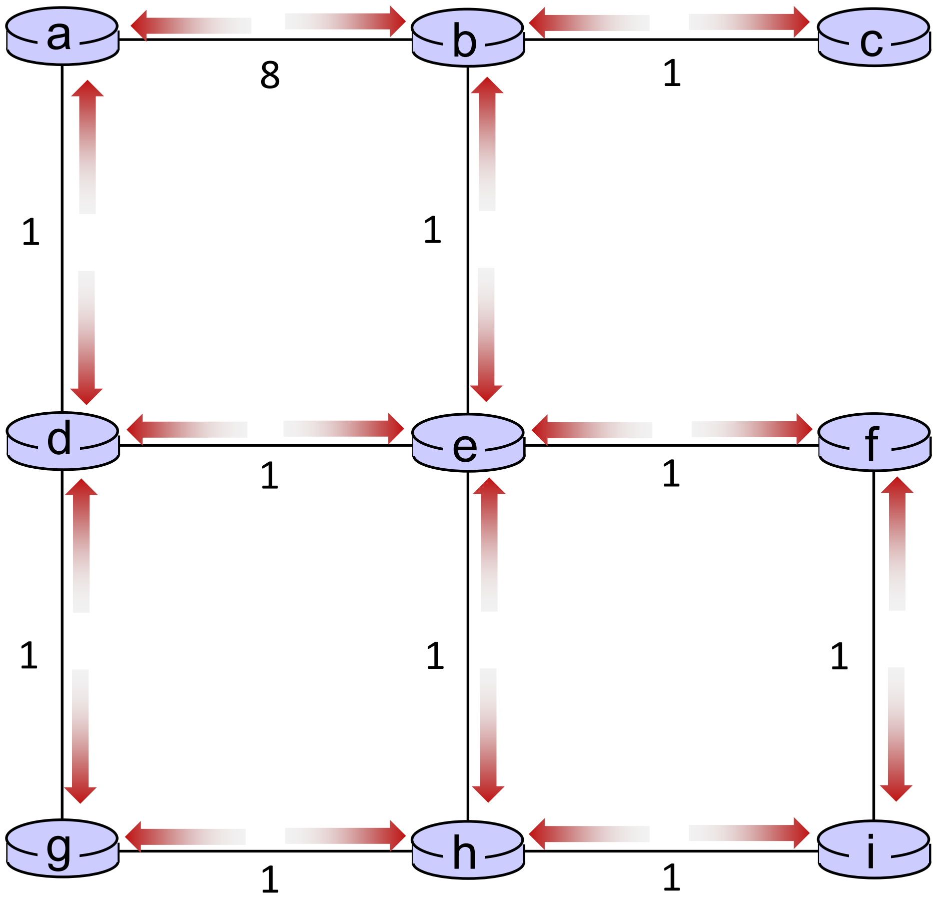
- missing link
- larger cost

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

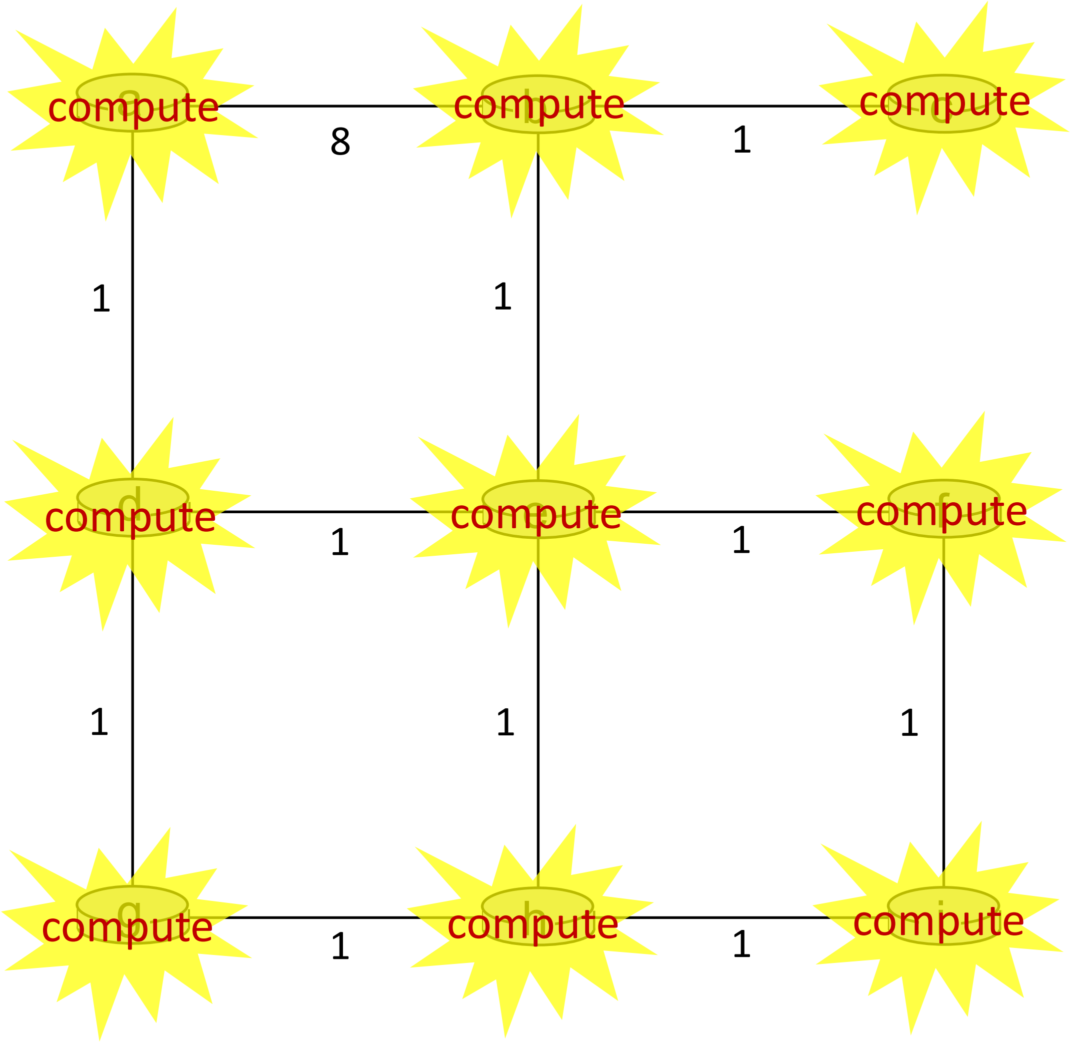


Distance vector example: iteration

t=1

All nodes:

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

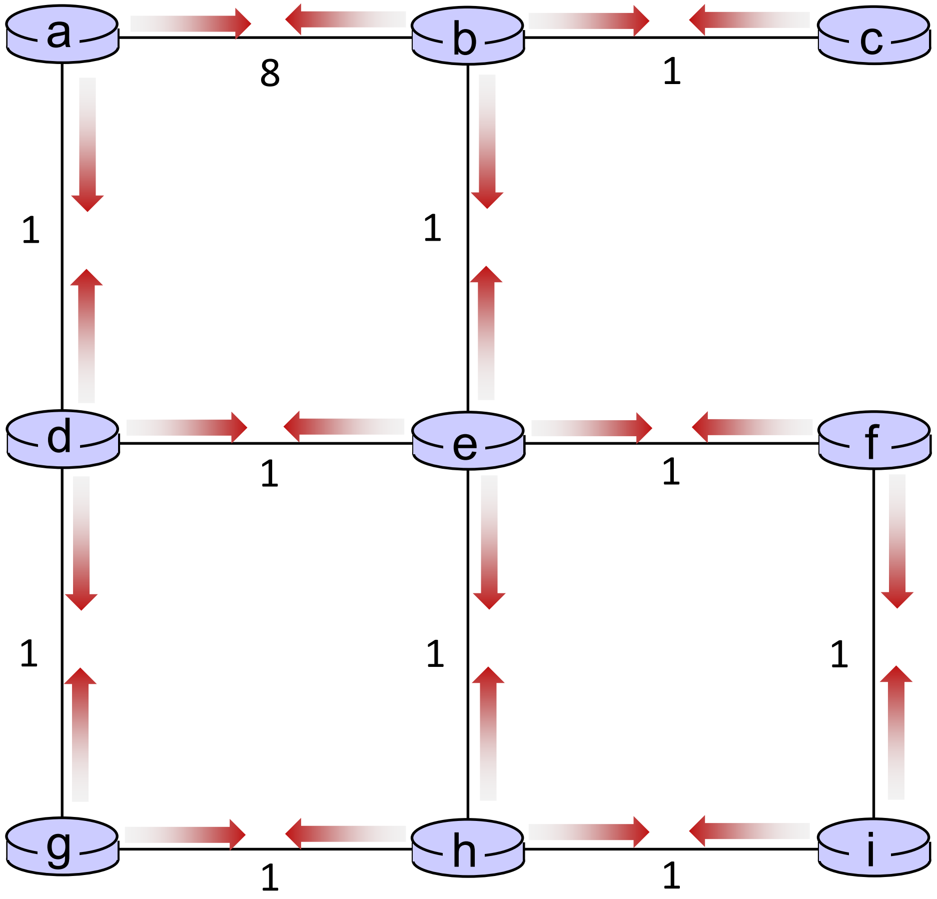


Distance vector example: iteration

t=1

All nodes:

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

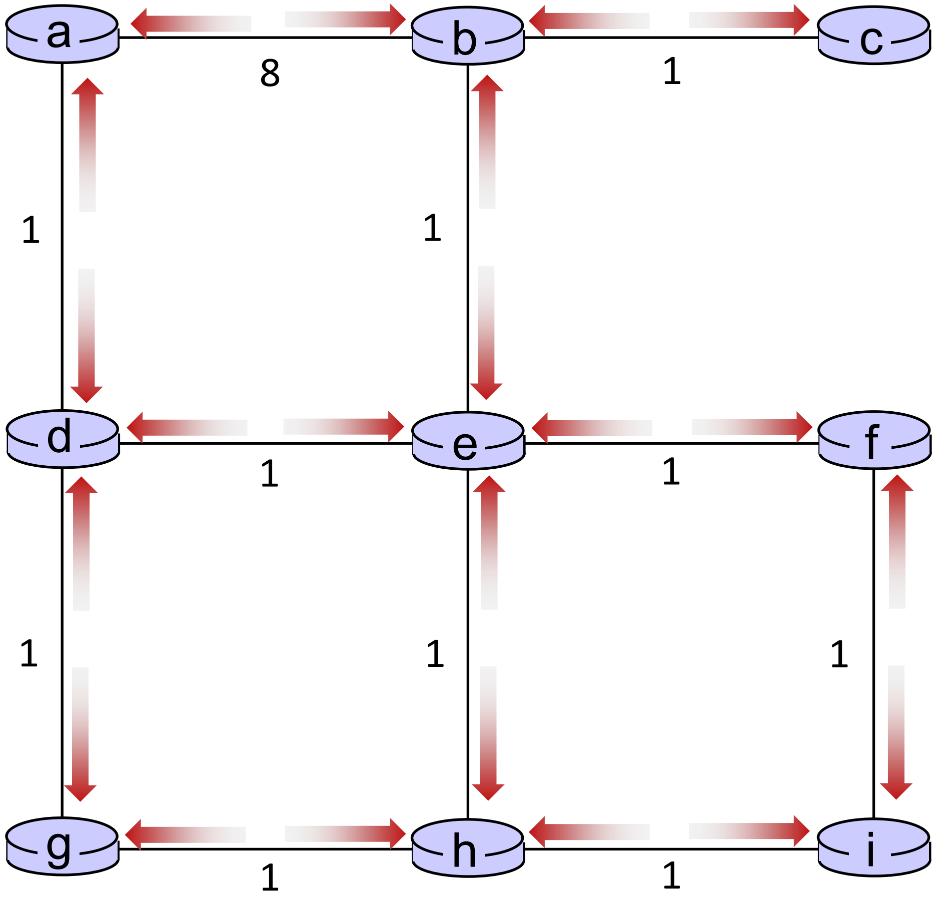


Distance vector example: iteration

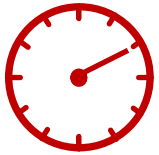
t=2

All nodes:

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



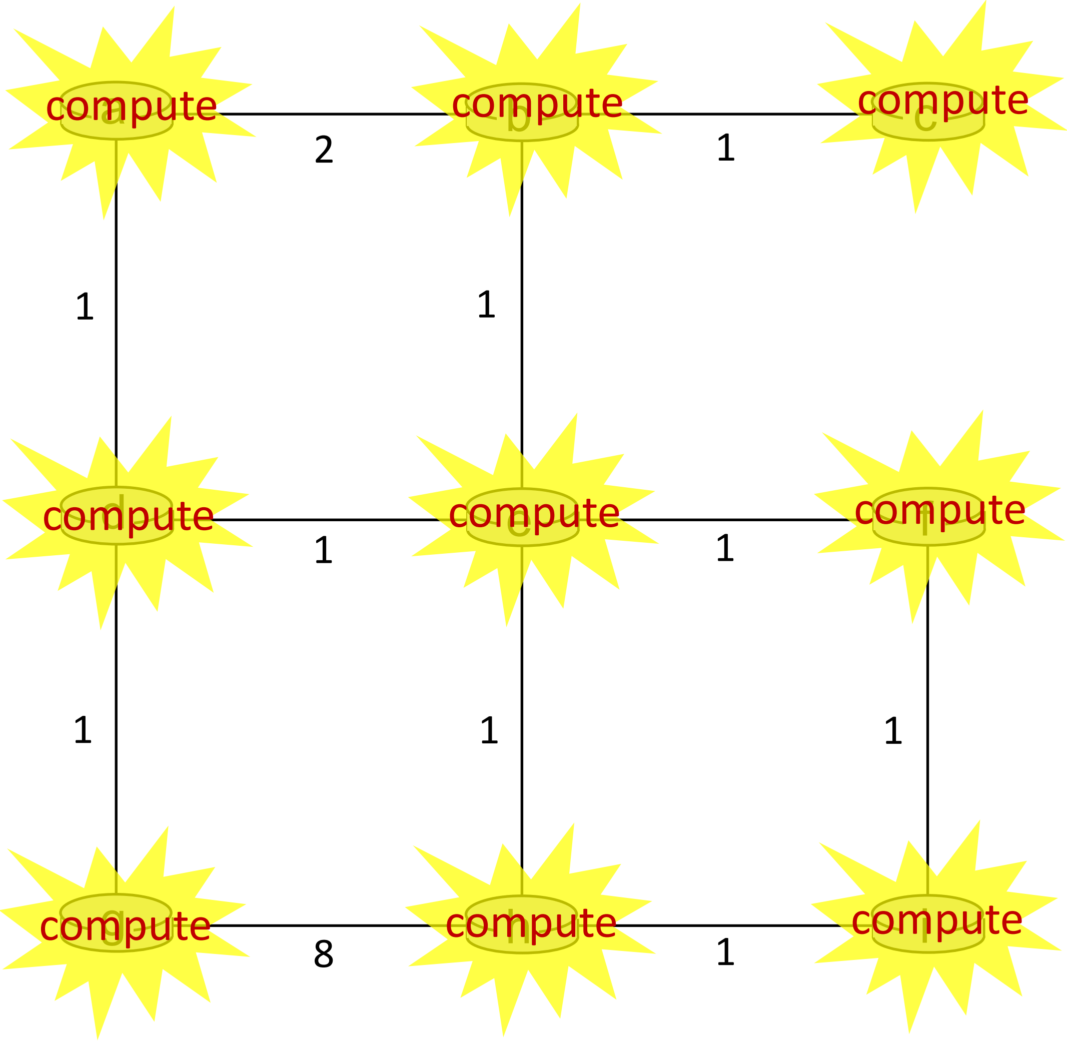
Distance vector example: iteration



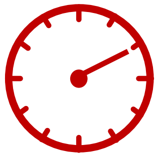
t=2

All nodes:

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



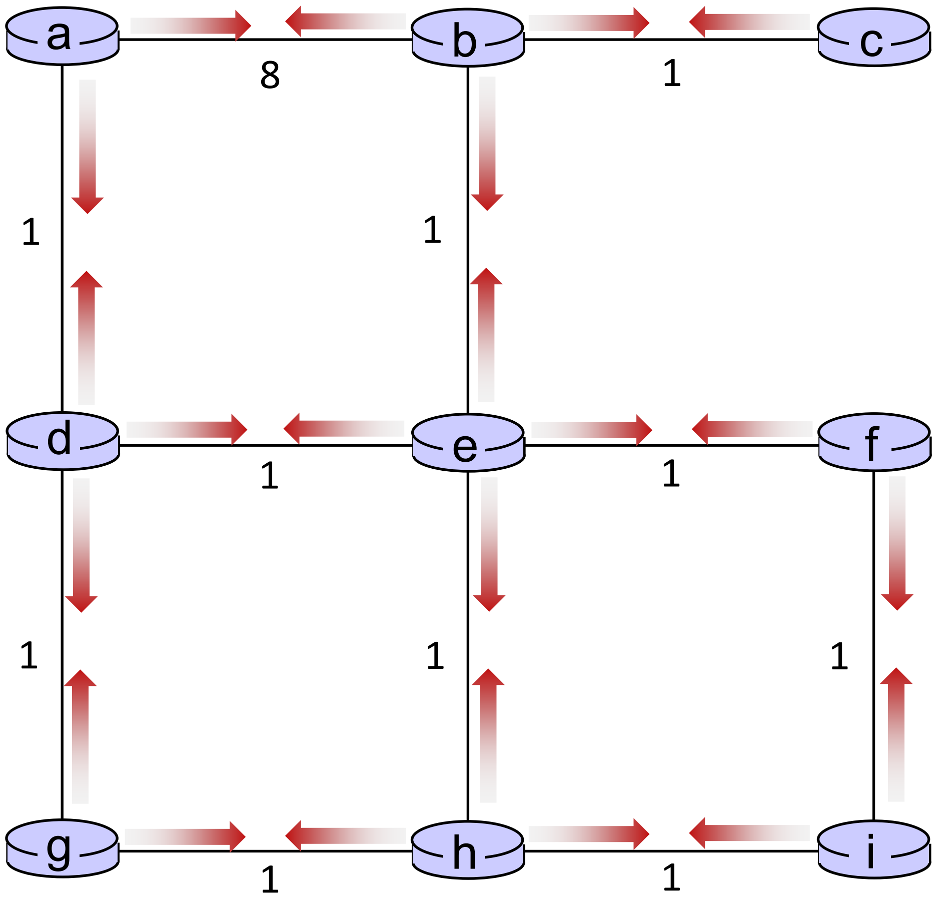
Distance vector example: iteration



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

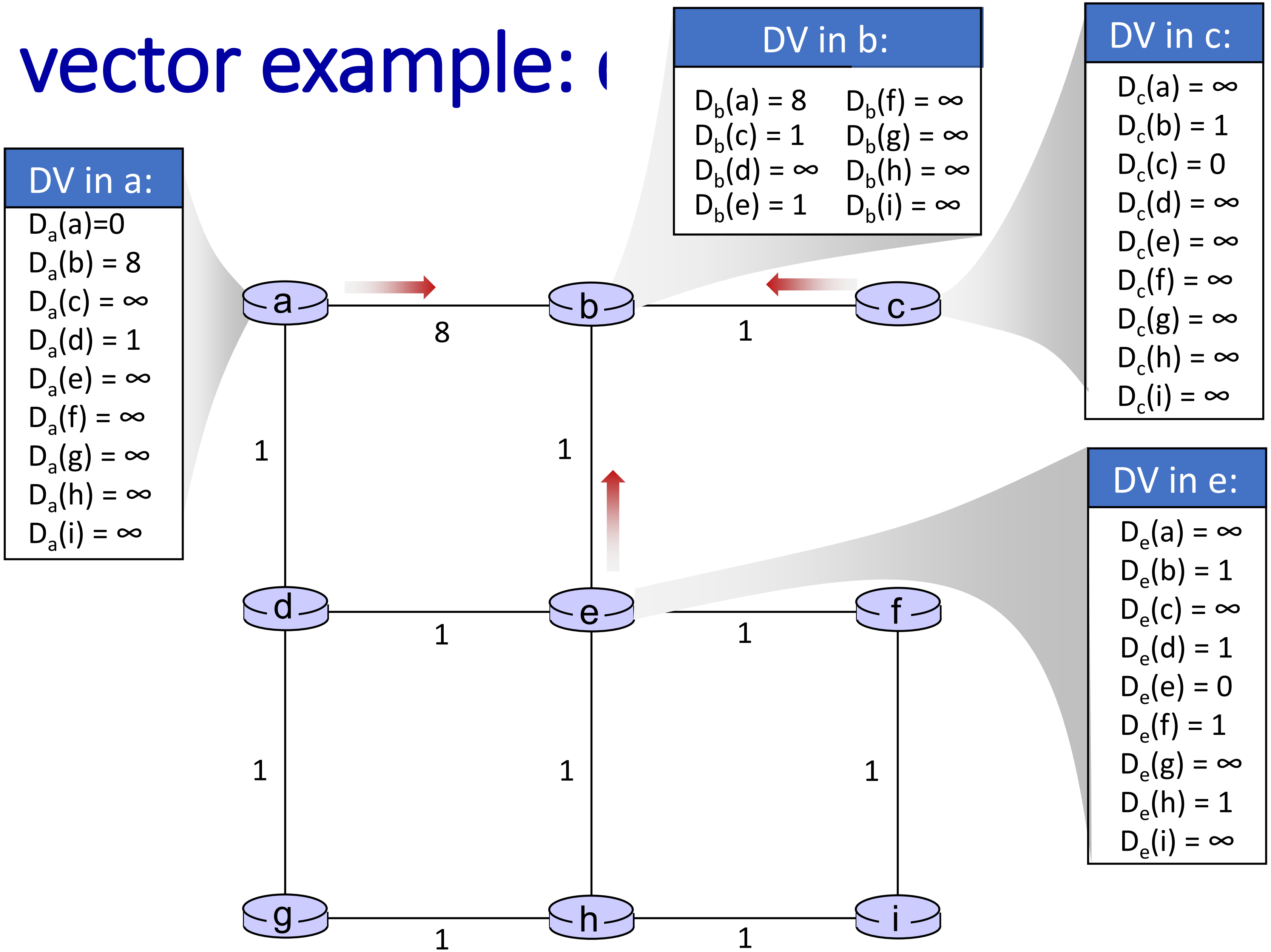
…. and so on

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:

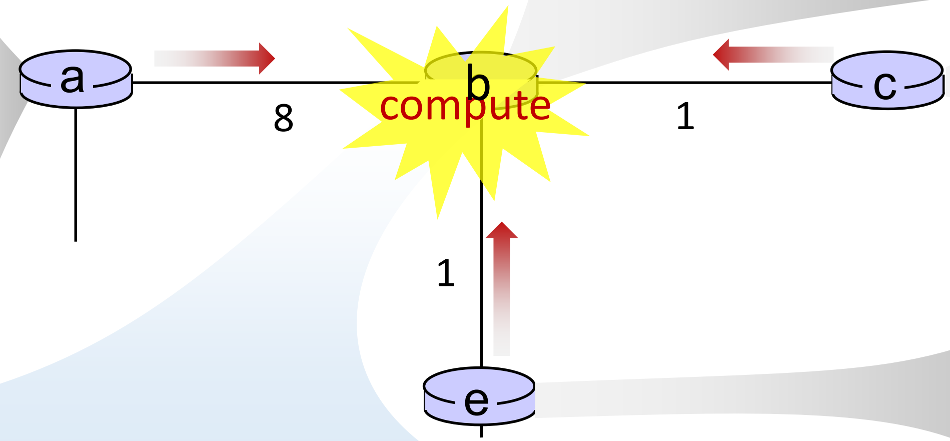
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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:

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

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

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

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

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

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

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

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

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

DV in b:

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

Distance vector example:

t=1

- c receives DVs from b

DV in a:

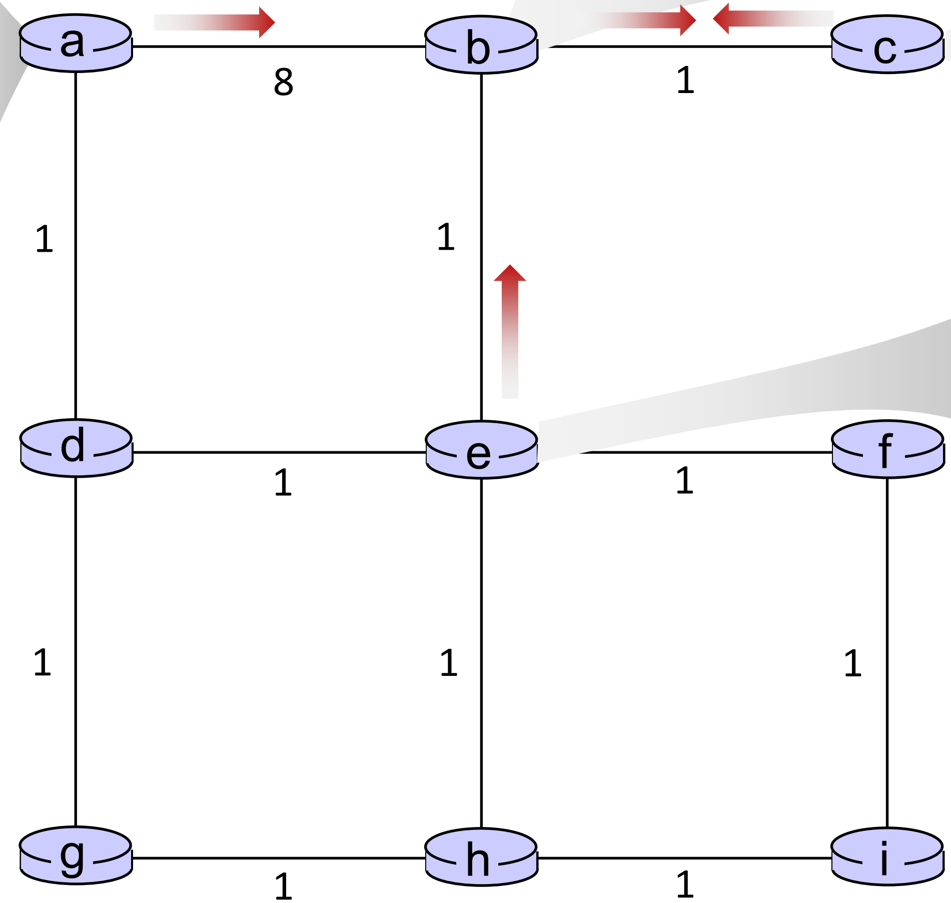
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:

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

DV in c:

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



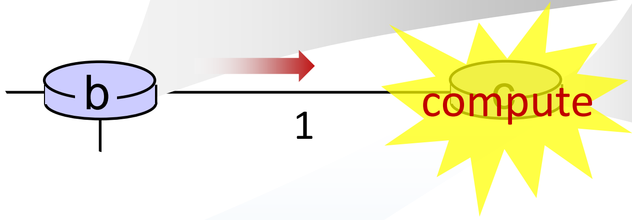
DV in e:

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

Distance vector example:

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: (

t=1

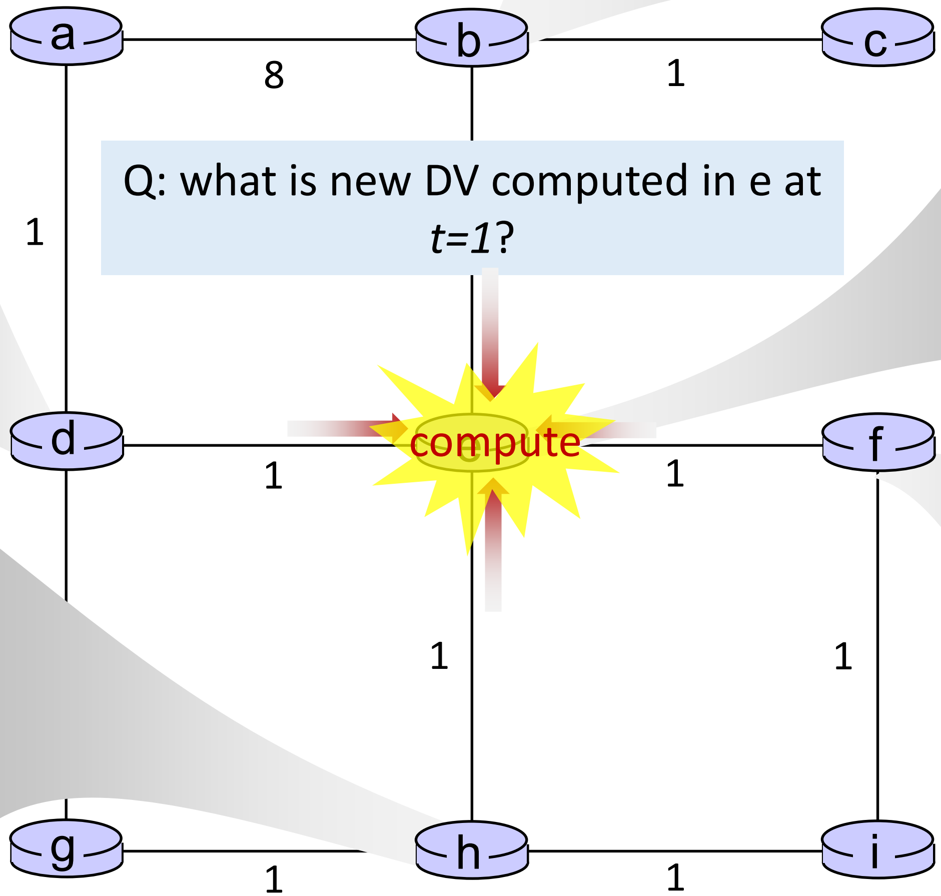
- e receives DVs from b, d, f, h

DV in d:

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

DV in h:

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



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

DV in b:

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

DV in e:

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

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:

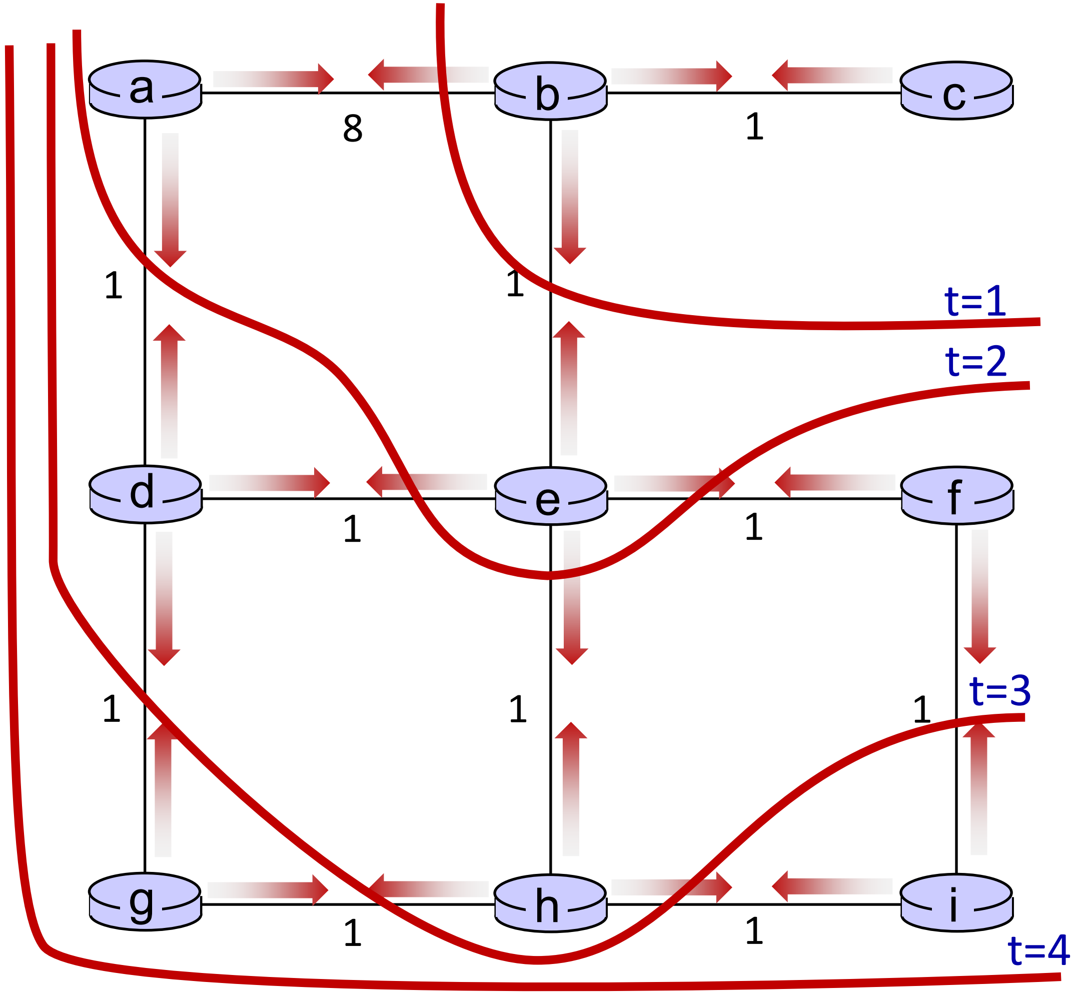
t=0 c's state at t=0 is at c only

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



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	∞	∞	∞

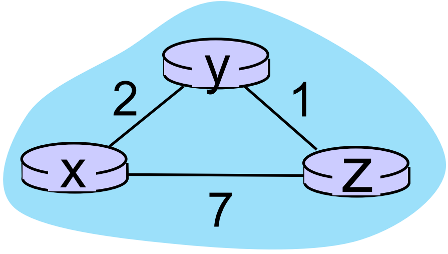
$D_z()$

		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

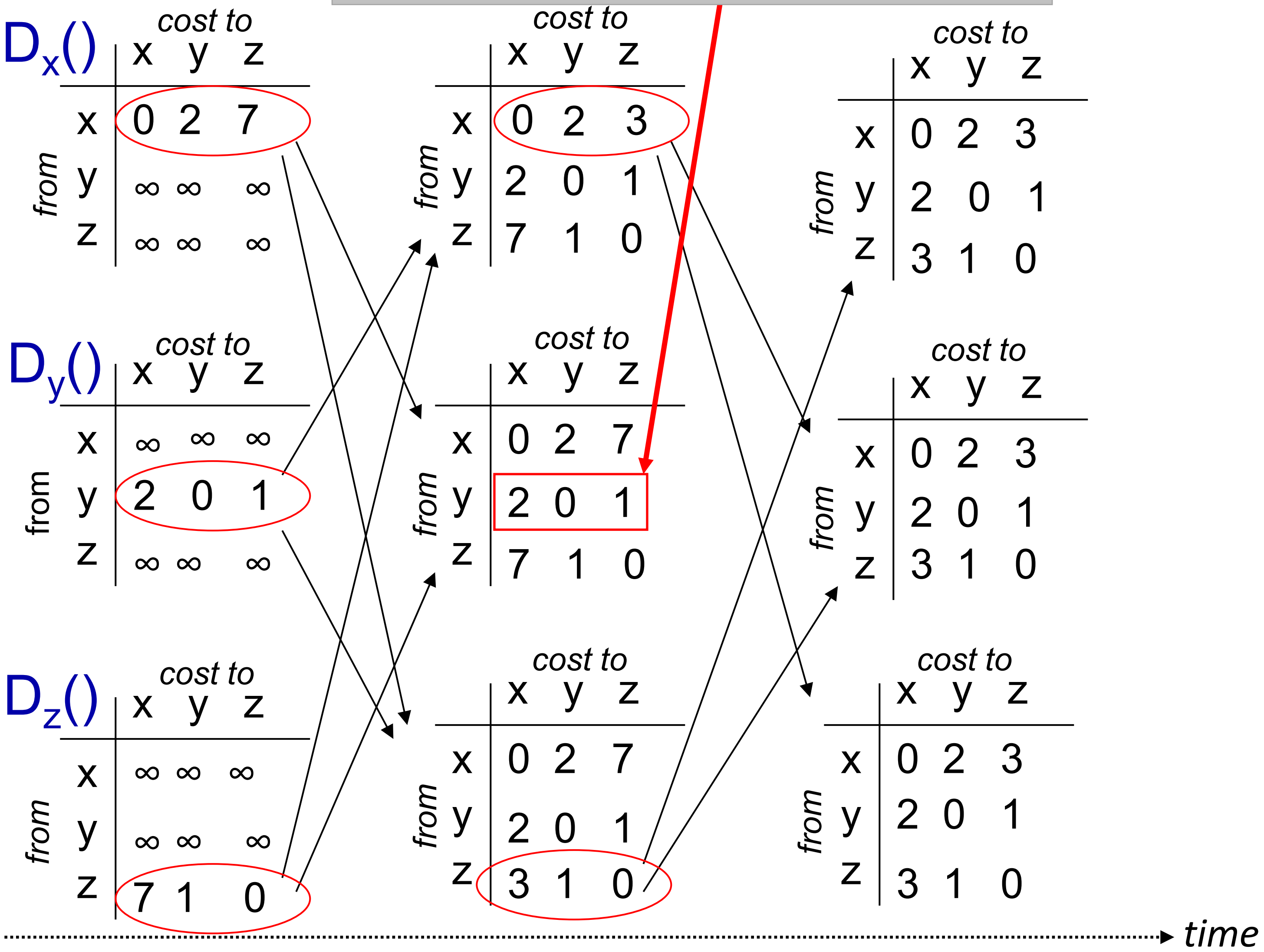
$$D_x(z) = \min\{c_{x,y} + D_y(z),\ c_{x,z} + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

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



time

Distance vector: another example



y’s DV did not change after 1st iteration, so do not propagate its DV to neighbors

$D_x()$

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

$D_y()$

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

$D_z()$

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

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

cost to

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

cost to

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

cost to

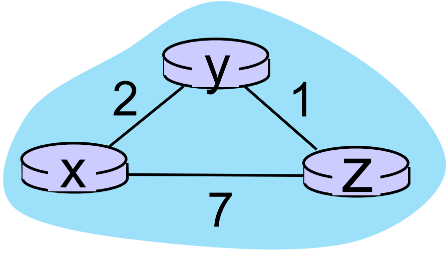
from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

cost to

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

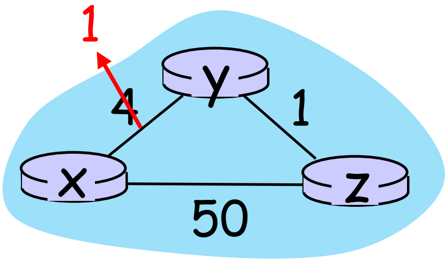
cost to

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



IMPORTANT

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"

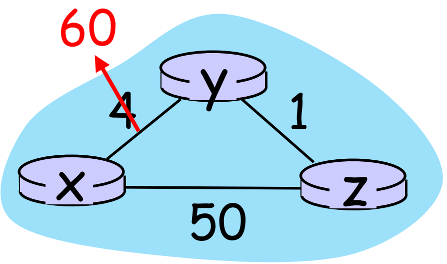
t_0 : *y* detects link-cost change, updates its DV $D_y(x)$=1, informs its neighbors.

t_1 : *z* receives update from *y*, updates its DV, computes new least cost to *x* to be min(50, 1+1)=2, 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

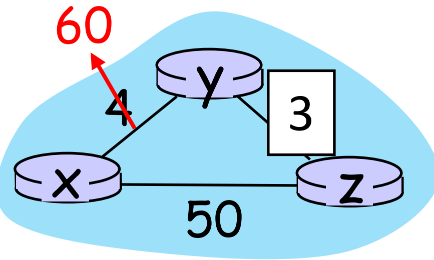
IMPORTANT

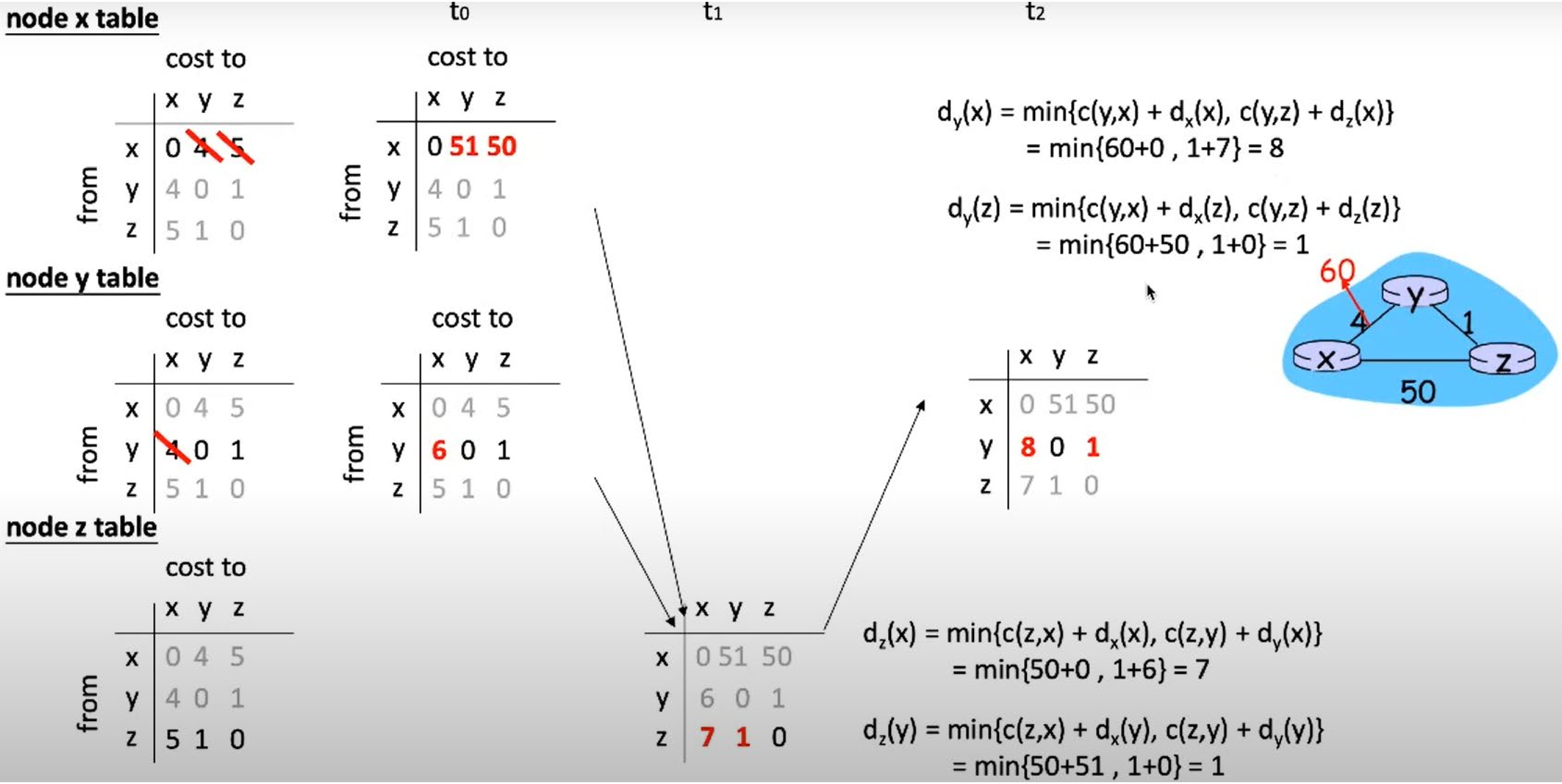


link cost changes:

- node detects local link cost change
- “bad news travels slowly” :

- *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 $D_y(x) = \min\{c_{y,x}+ D_x(x), c_{y,z}+ D_z(x)\}=\min\{60+0, 1+5\}=6$ “my new cost to x will be 6 via z); notifies *z* of new cost of $D_y(x)=6$ to *x.*
- *z* learns that path to *x* via *y* has new cost 6, so *z* computes $D_z(x) = \min\{c_{z,x}+ D_x(x), c_{z,y}+ D_y(x)\}=\min\{50+0, 1+6\}=7$ “my new cost to *x* will be $D_z(x)=7$ 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 $D_y(x) = \min\{c_{y,x}+ D_x(x), c_{y,z}+ D_z(x)\}=\min\{60+0, 1+7\}=8$ “my new cost to *x* will be 8 via z), notifies *z* of new cost of $D_y(x)=8$ to *x.*
- *z* learns that path to *x* via *y* has new cost 8, so *z* computes $D_z(x) = \min\{c_{z,x}+ D_x(x), c_{z,y}+ D_y(x)\}=\min\{50+0, 1+8\}=9$ “my new cost to *x* will be $D_z(x)=9$ via y), notifies *y* of new cost of 9 to *x.*
- The iterations will stop when $D_z(x)$ reaches 50 ($D_z(x)$, $D_y(x)$ going from 5 to 50 in step of 1). And then $D_y(x)$ will reach 51.
- Q: what if link cost between y and z is 3 instead of 1?
- Q: ($D_z(x), D_y(x)$ going from 5 to 8 to 11 to 14… until reaching or exceeding 50 in step of 3





node x table

cost to

from	x	y	z
x	0	~~4~~	~~5~~
y	4	0	1
z	5	1	0

cost to

from	x	y	z
x	0	51	50
y	4	0	1
z	5	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	~~4~~	0	1
z	5	1	0

cost to

from	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

node z table

cost to

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

$$d_z(x) = \min\{c(z,x) + d_x(x),\ c(z,y) + d_y(x)\}$$
$$= \min\{50+0\ ,\ 1+6\} = 7$$

$$d_z(y) = \min\{c(z,x) + d_x(y),\ c(z,y) + d_y(y)\}$$
$$= \min\{50+51\ ,\ 1+0\} = 1$$

$$d_y(x) = \min\{c(y,x) + d_x(x),\ c(y,z) + d_z(x)\}$$
$$= \min\{60+0\ ,\ 1+7\} = 8$$

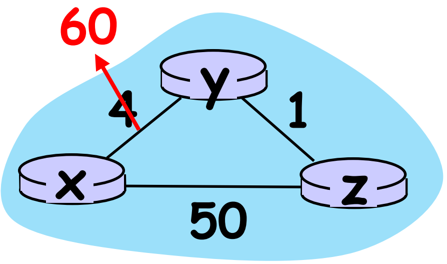
$$d_y(z) = \min\{c(y,x) + d_x(z),\ c(y,z) + d_z(z)\}$$
$$= \min\{60+50\ ,\ 1+0\} = 1$$

	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

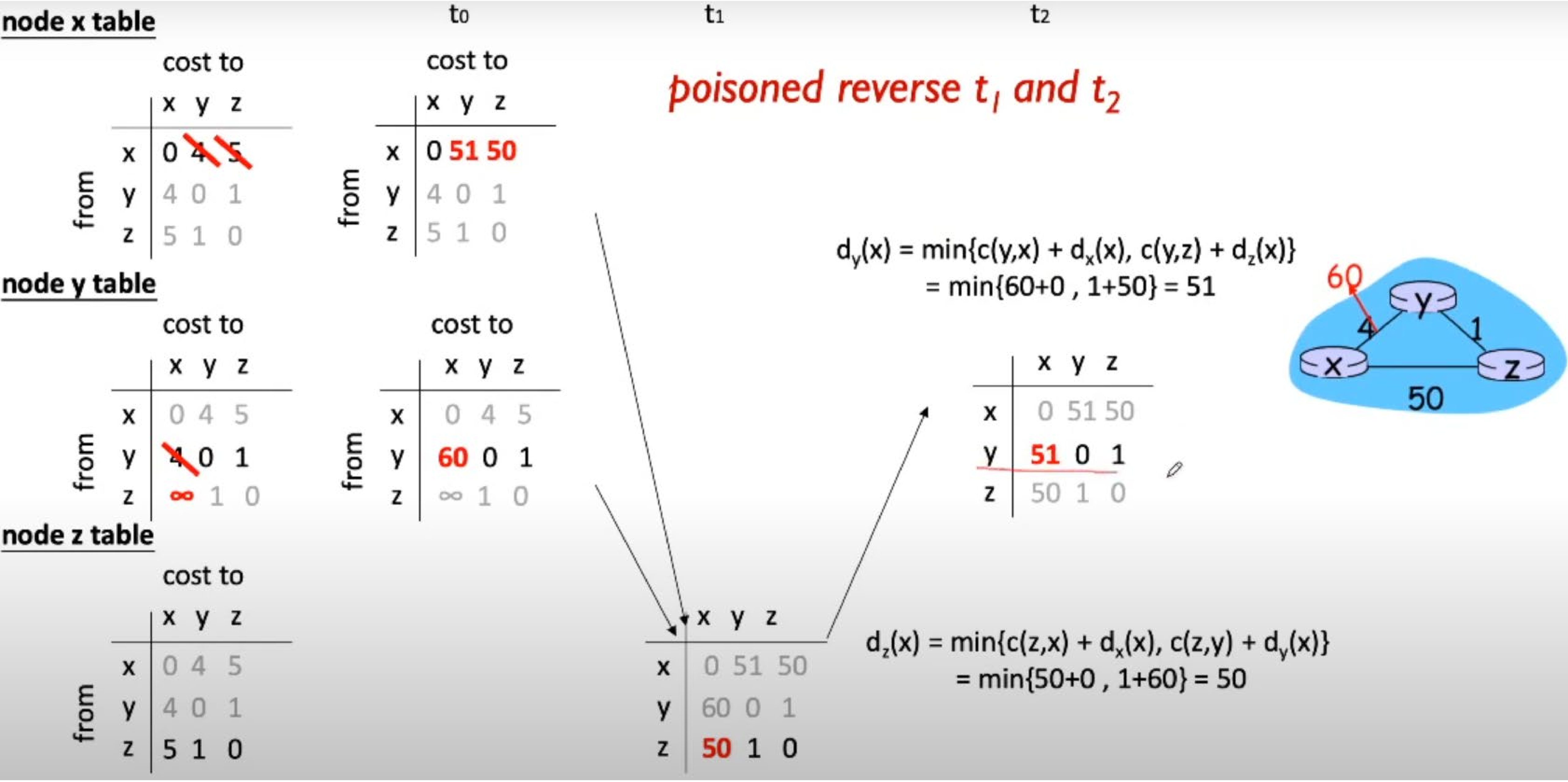
ch52 ep#12 DV Bad News Travel Slow Part II
https://www.youtube.com/watch?v=kLmhxtL2FRI

IMPORTANT

Solution: Poisoned Reverse



- If z routes through y to get to x:
  - In y’s routing table, set z's distance to x $D_z(x)=\infty$ (so y won't route to x via z, removing the circular dependency)
- *y* sees direct link to *x* has new cost 60, so *y* computes $D_y(x)$ = min{$c_{y,x}$+ $D_x(x)$, $c_{y,z}$+ $D_z(x)$}=min{$60+0$, $1+\infty$}=60 “my new cost to x will be 60 via direct link; notifies *z* of new cost of $D_y(x)$=60 to *x.*
- *z* learns that path to *x* via *y* has new cost 60, so *z* computes $D_z(x)$ = min{$c_{z,x}$+ $D_x(x)$, $c_{z,y}$+ $D_y(x)$}=min{$50+0$, $1+60$}=50 “my new cost to *x* will be $D_z(x)$=50 via direct link, notifies *y* of new cost of 50 to *x.*
- *y* learns that path to *x* via *z* has new cost 50, so *y* computes $D_y(x)$ = min{$c_{y,x}$+ $D_x(x)$, $c_{y,z}$+ $D_z(x)$}=min{$60+0$, $1+50$}=51 “my new cost to *x* will be 51 via z, notifies *z* of new cost of $D_y(x)=51$ to *x.*
- Algorithm has converged.



poisoned reverse t_1 and t_2

node x table

t_0

cost to

from	x	y	z
x	0	~~4~~	~~5~~
y	4	0	1
z	5	1	0

t_1

cost to

from	x	y	z
x	0	51	50
y	4	0	1
z	5	1	0

node y table

t_0

cost to

from	x	y	z
x	0	4	5
y	~~4~~	0	1
z	∞	1	0

t_1

cost to

from	x	y	z
x	0	4	5
y	60	0	1
z	∞	1	0

t_2

$$d_y(x) = \min\{c(y,x) + d_x(x),\ c(y,z) + d_z(x)\}$$
$$= \min\{60+0\ ,\ 1+50\} = 51$$

	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node z table

t_0

cost to

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

t_1

	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

$$d_z(x) = \min\{c(z,x) + d_x(x),\ c(z,y) + d_y(x)\}$$
$$= \min\{50+0\ ,\ 1+60\} = 50$$

ch52 ep#13 DV Poisoned Reverse
https://www.youtube.com/watch?v=UcTgNZwGmRU

Summary

- Link State: Cost of link to neighbors sent to the entire network. Large # of small messages.
  - *n* routers, O(n^2) messages sent among all routers
  - Dijkstra's algorithm is used to compute the shortest path using the link state
- Distance Vectors: Distance to all nodes in the network sent to neighbors. Small # of large messages
  - Message exchange between neighbors; convergence time varies
  - Bellman Ford's algorithm is used to compute shortest paths using distance vectors