Chapter 5
Network Layer:
Control Plane
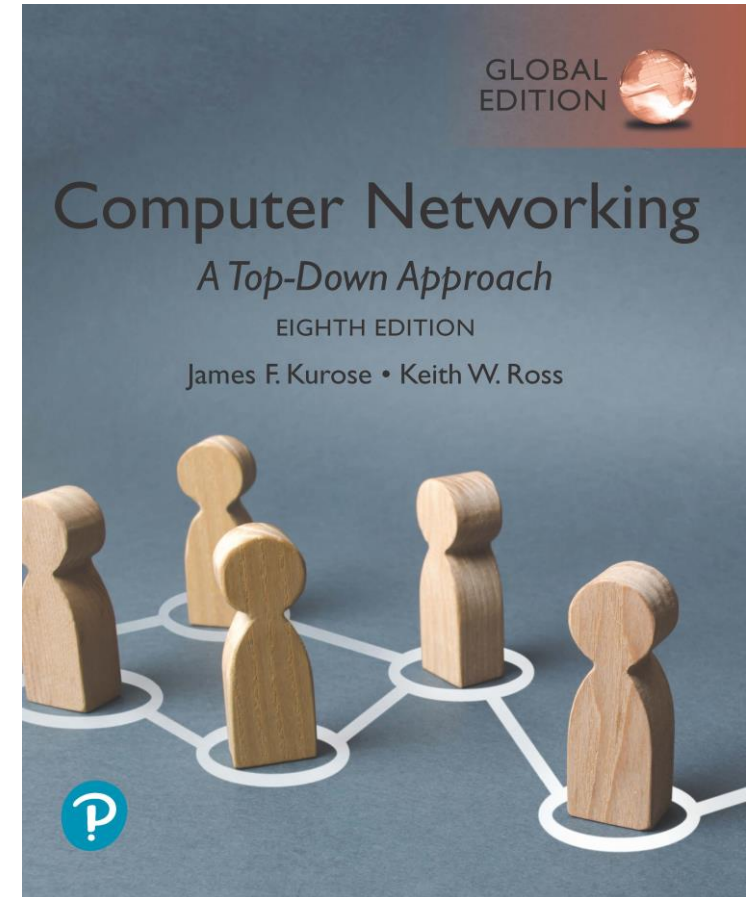
Network layer control plane: our goals

- understand principles behind network control plane:
  - traditional routing algorithms
  - SDN controllers
  - network management, configuration

- instantiation, implementation in the Internet:
  - OSPF, BGP
  - OpenFlow, ODL and ONOS controllers
  - Internet Control Message Protocol: ICMP
  - SNMP, YANG/NETCONF

Network layer: "control plane" roadmap

- **introduction**

- routing protocols
  - link state
  - distance vector

- intra-ISP routing: OSPF

- routing among ISPs: BGP

- SDN control plane

- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

Network-layer functions

- forwarding: move packets from router's input to appropriate router output

data plane

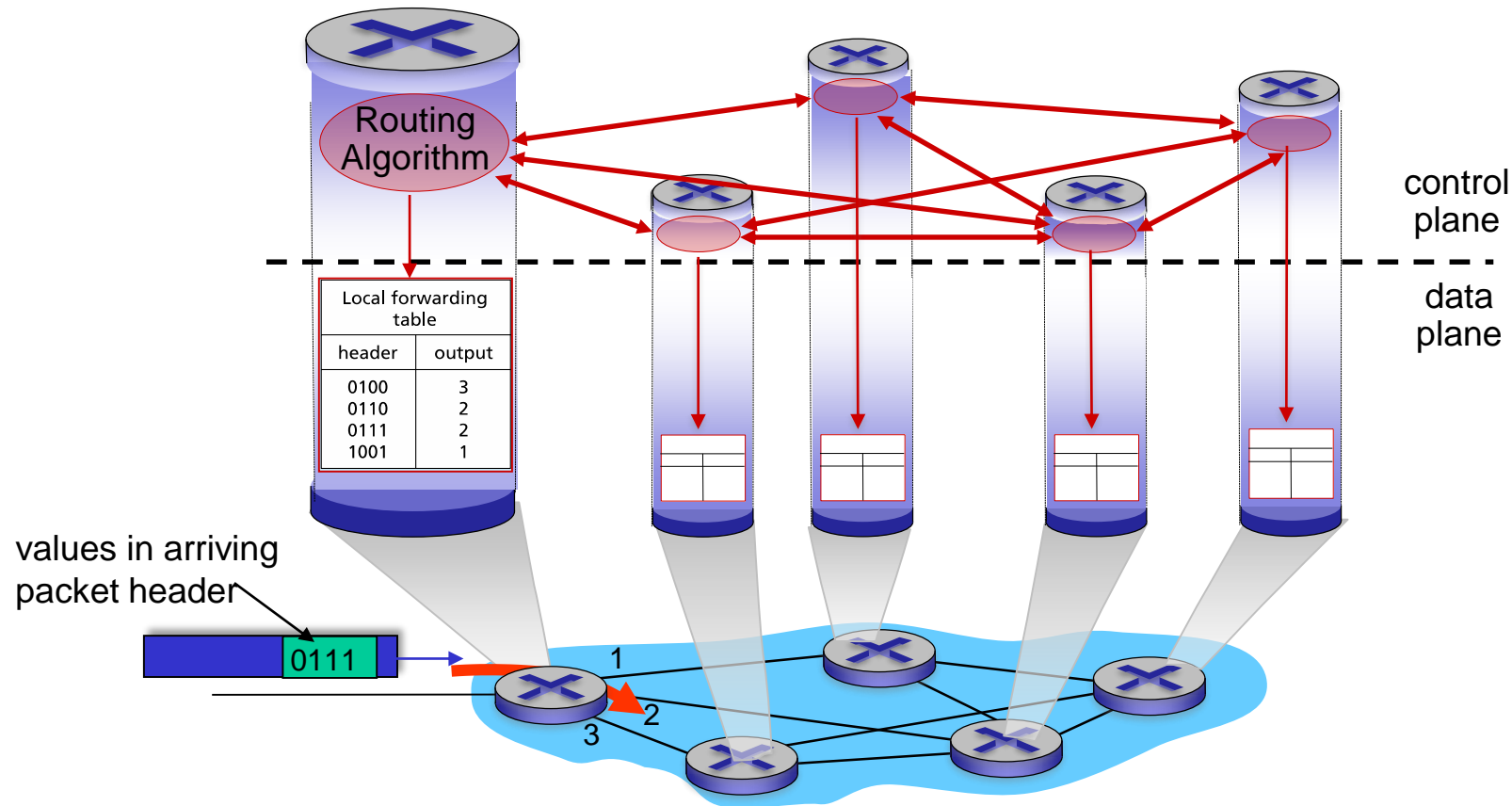- routing: determine route taken by packets from source to destination

control plane

Two approaches to structuring network control plane:
- per-router control (traditional)
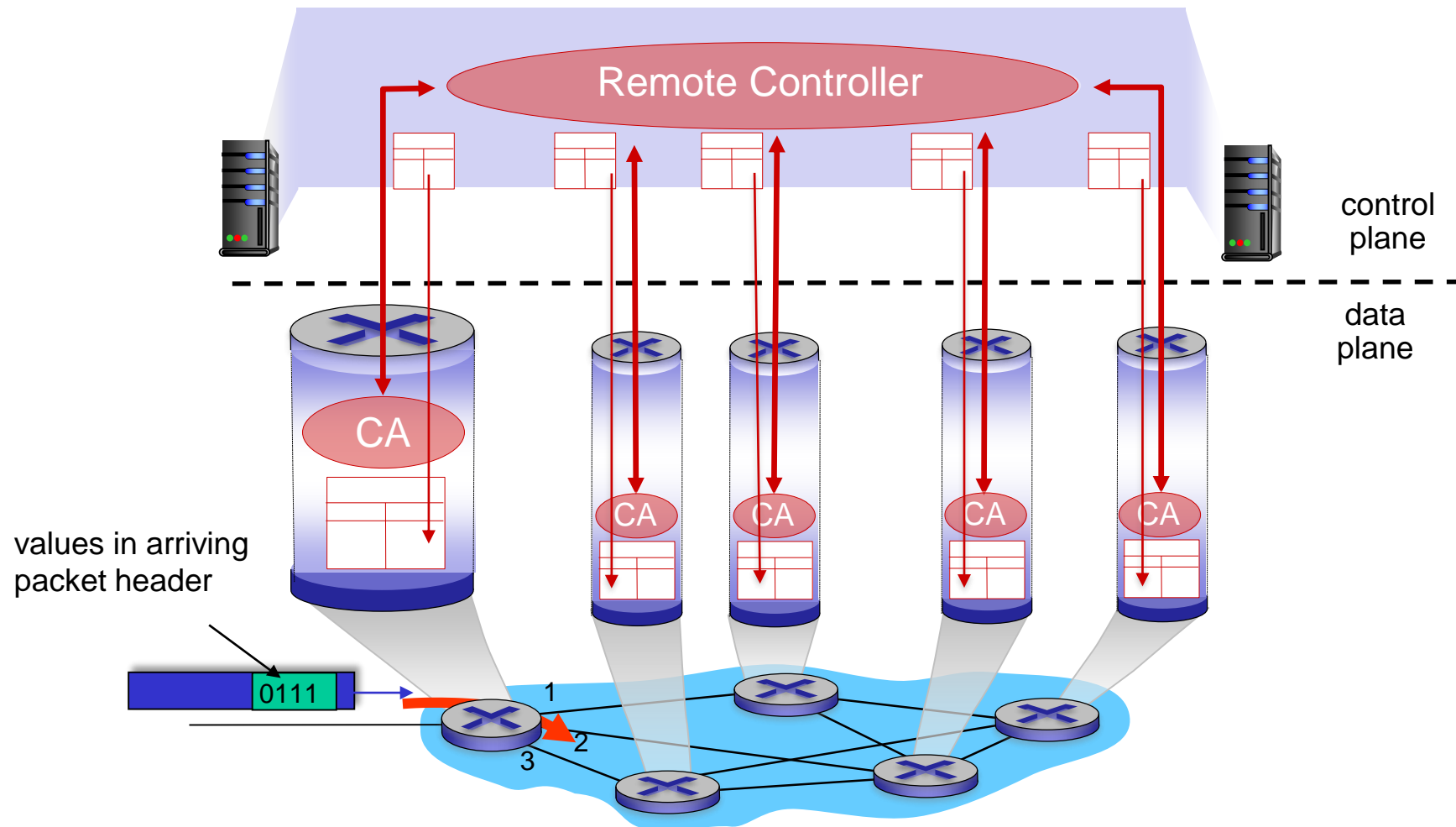- logically centralized control (software defined networking)

Per-router control plane

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

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

Network layer: "control plane" roadmap

- introduction

- **routing protocols**
  - **link state**
  - **distance vector**

- intra-ISP routing: OSPF

- routing among ISPs: BGP

- SDN control plane

- Internet Control Message Protocol

- network management, configuration
 - SNMP
  - NETCONF/YANG

Routing protocols

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

- **path:** sequence of routers packets traverse from given initial source host to final destination host

- **"good":** least "cost", "fastest", "least congested"

- routing: a "top-10" networking challenge!
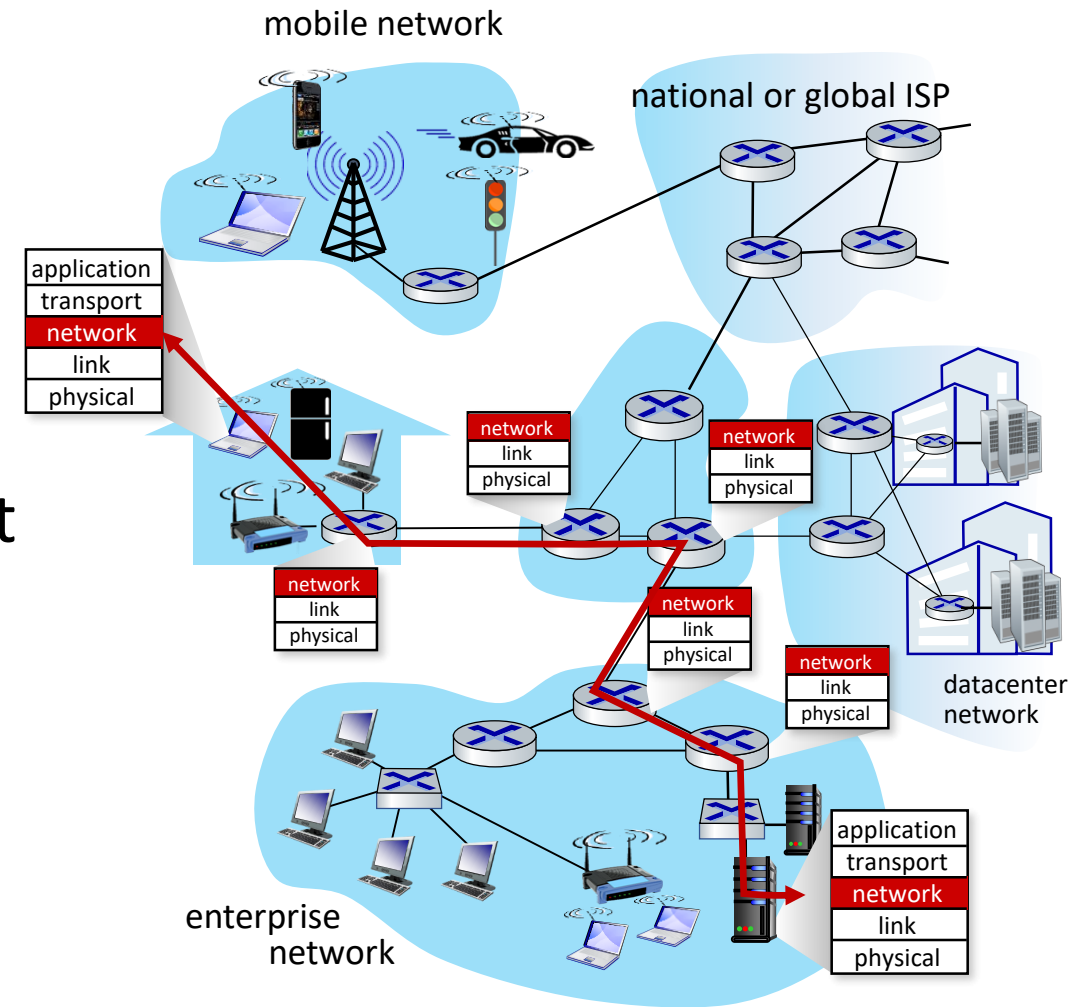
Graph abstraction: link costs
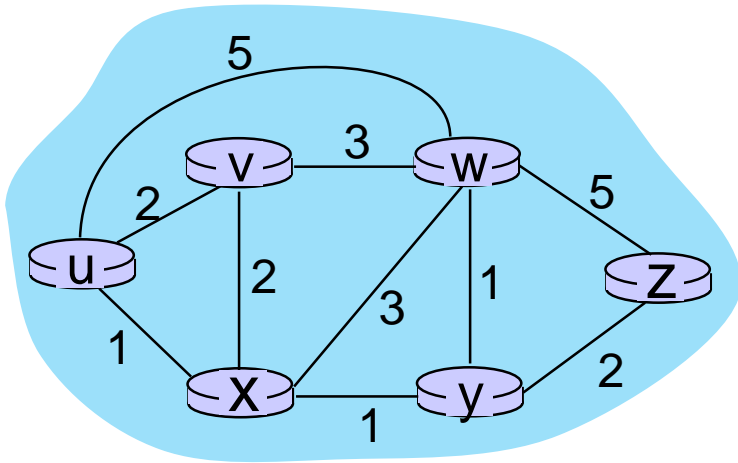
$c_{a,b}$: cost of *direct* link connecting a and b

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

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

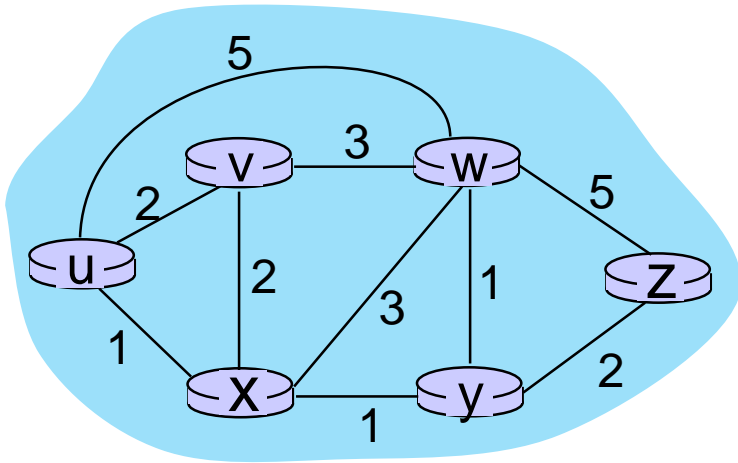graph: *G = (N,E)*

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)* }

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$
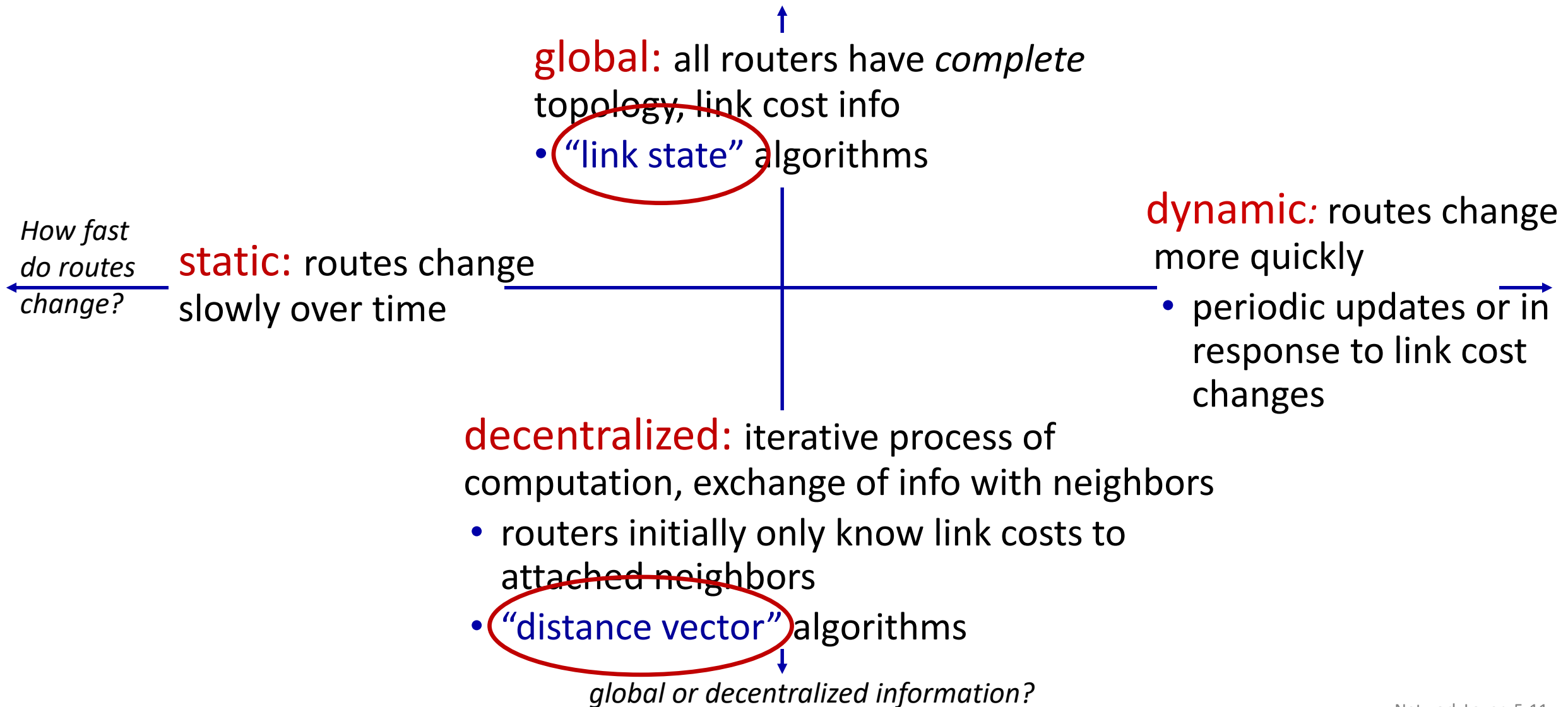
Graph abstraction: link costs

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$

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

Routing algorithm classification

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

dynamic: routes change more quickly
- periodic updates or in response to link cost changes

How fast do routes change?

static: routes change slowly over time

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

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' :
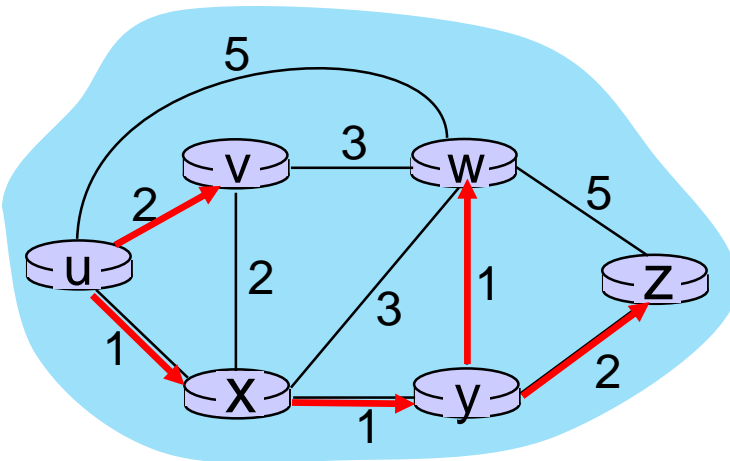12   **$D(v) = \min (D(v),\ D(w) + c_{w,v})$**
13   /* new least-path-cost to v is either old least-cost-path to v or known
14   least-cost-path to w plus direct-cost from w to v */
15 *until all nodes in N'*

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	∞	∞
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					

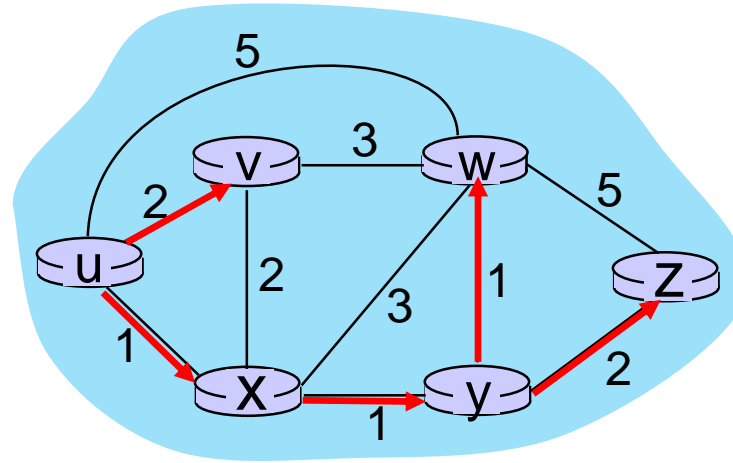Initialization (step 0): For all a: if a adjacent to then $D(a) = c_{u,a}$

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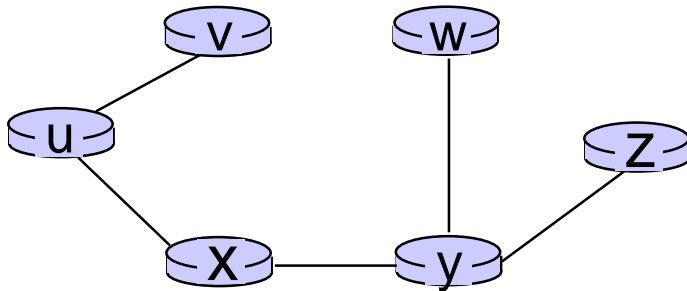update D(b) for all b adjacent to a and not in N' :
D(b) = min (D(b), D(a) + $c_{a,b}$)

Dijkstra's algorithm: an example

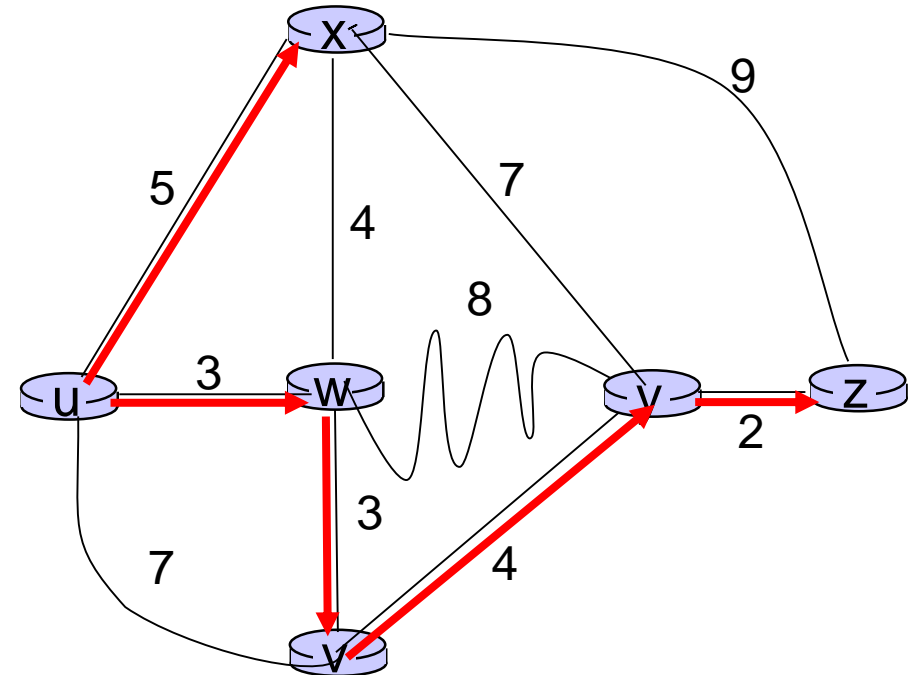resulting least-cost-path tree from u:

resulting forwarding table in u:

destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
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	∞	∞
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
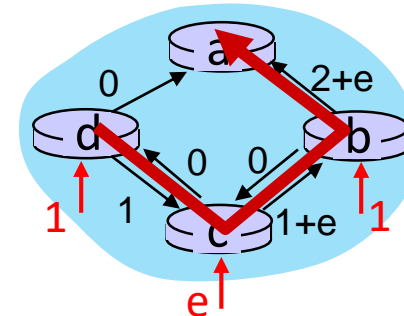
- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
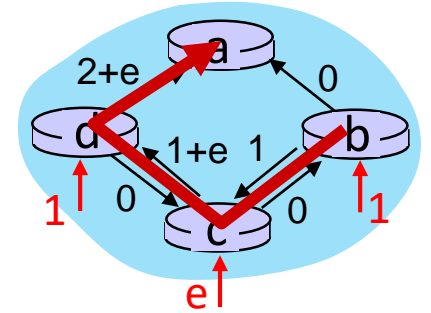 - link costs are directional, and volume-dependent
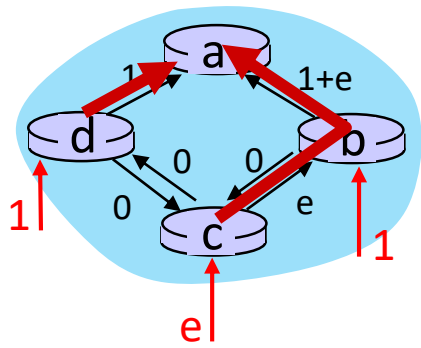
initially
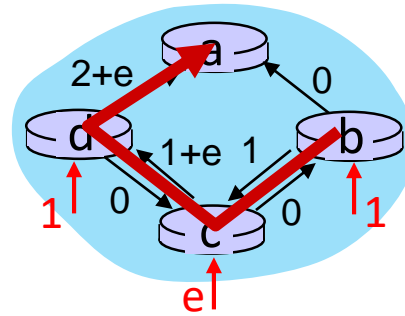
given these costs,
find new routing….
resulting in new costs

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

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

Network layer: "control plane" roadmap

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

- network management, configuration
  - SNMP
  - NETCONF/YANG

Distance vector algorithm

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

Bellman-Ford equation

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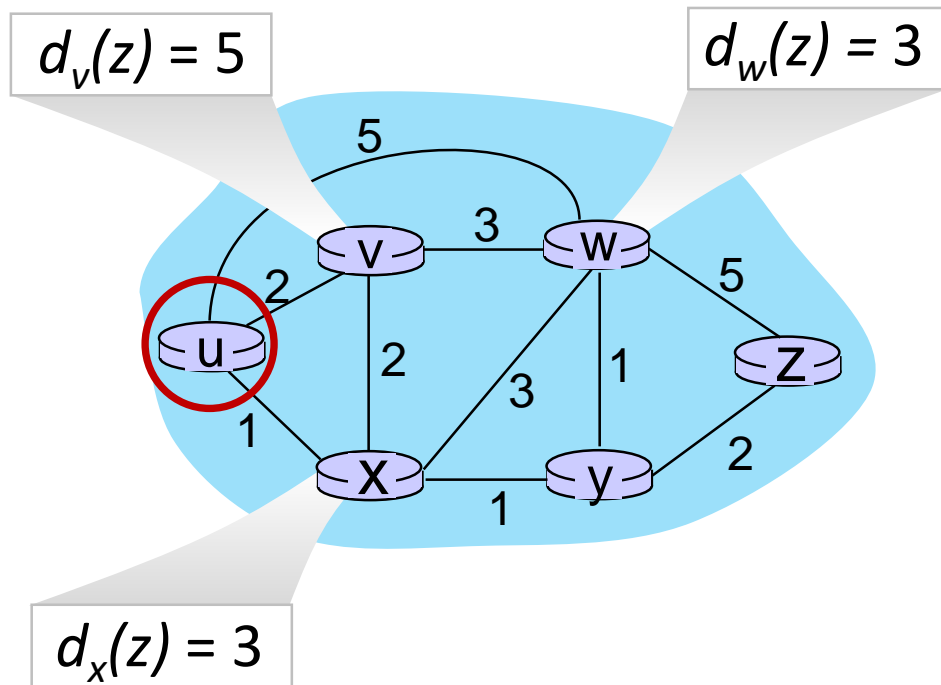Then:

$$d_x(y) = \min_v \{ c_{x,v} + d_v(y) \}$$

---

min taken over all neighbors *v* of *x*

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

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

Bellman-Ford Example

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

$d_v(z) = 5$

$d_w(z) = 3$

$d_x(z) = 3$

Bellman-Ford equation says:

$$d_u(z) = \min \{ c_{u,v} + d_v(z),$$
$$c_{u,x} + d_x(z),$$
$$c_{u,w} + d_w(z) \}$$

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

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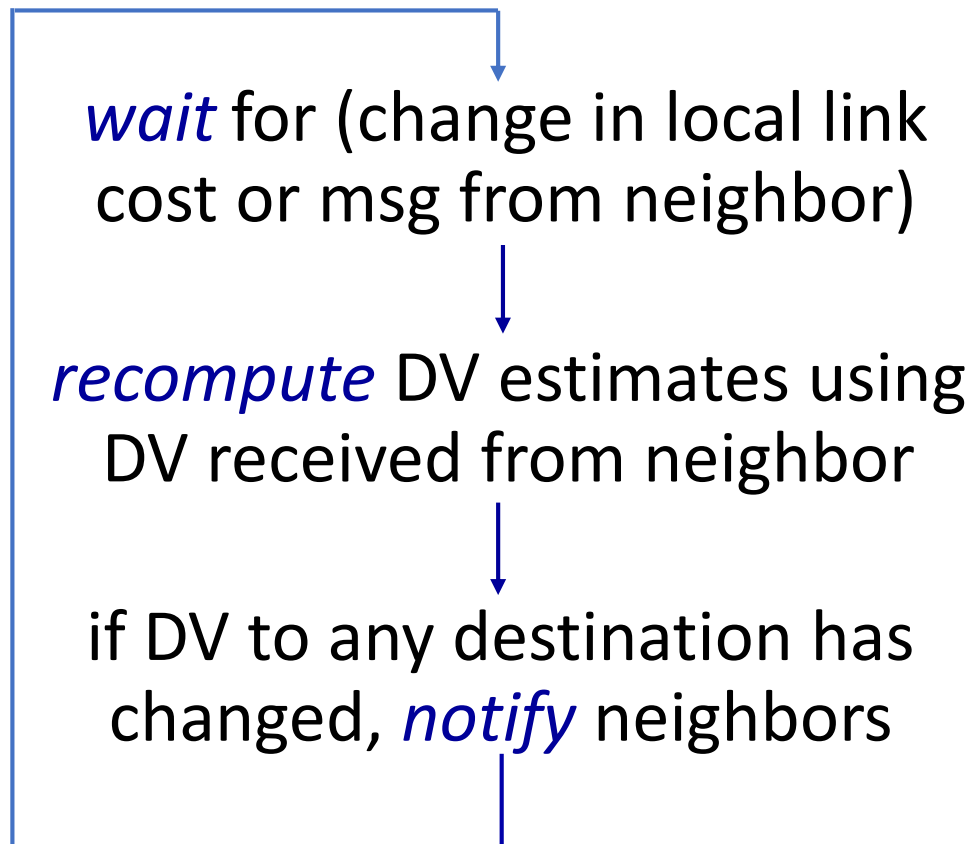
Distance vector algorithm

key idea:

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D_x}$ = $[D_x(y)\colon y \in N]$
- 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 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!

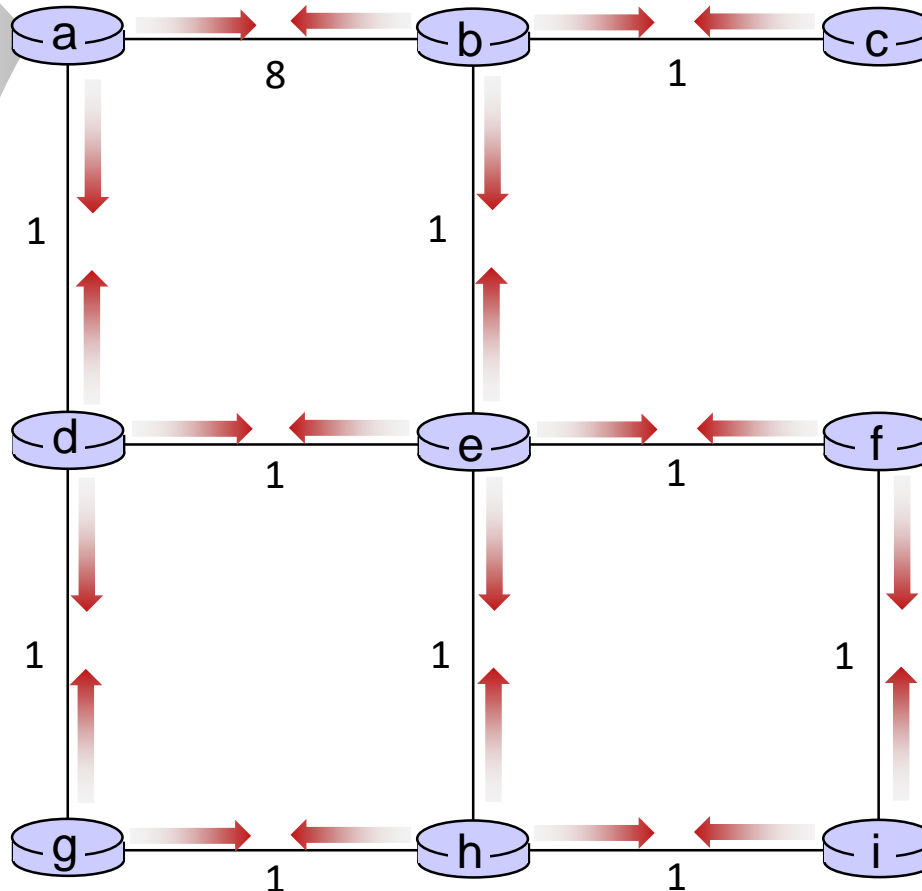# Distance vector: example

t=0

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

- All nodes have distance estimates to nearest neighbors (only)

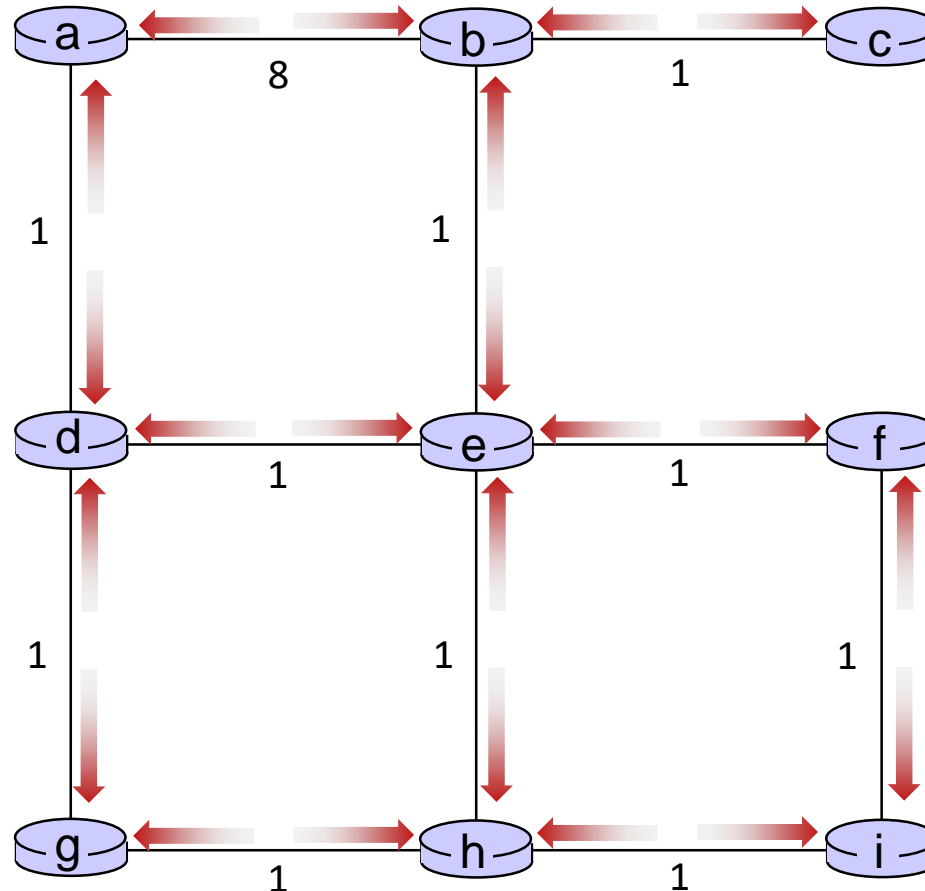- All nodes send their local distance vector to their neighbors

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
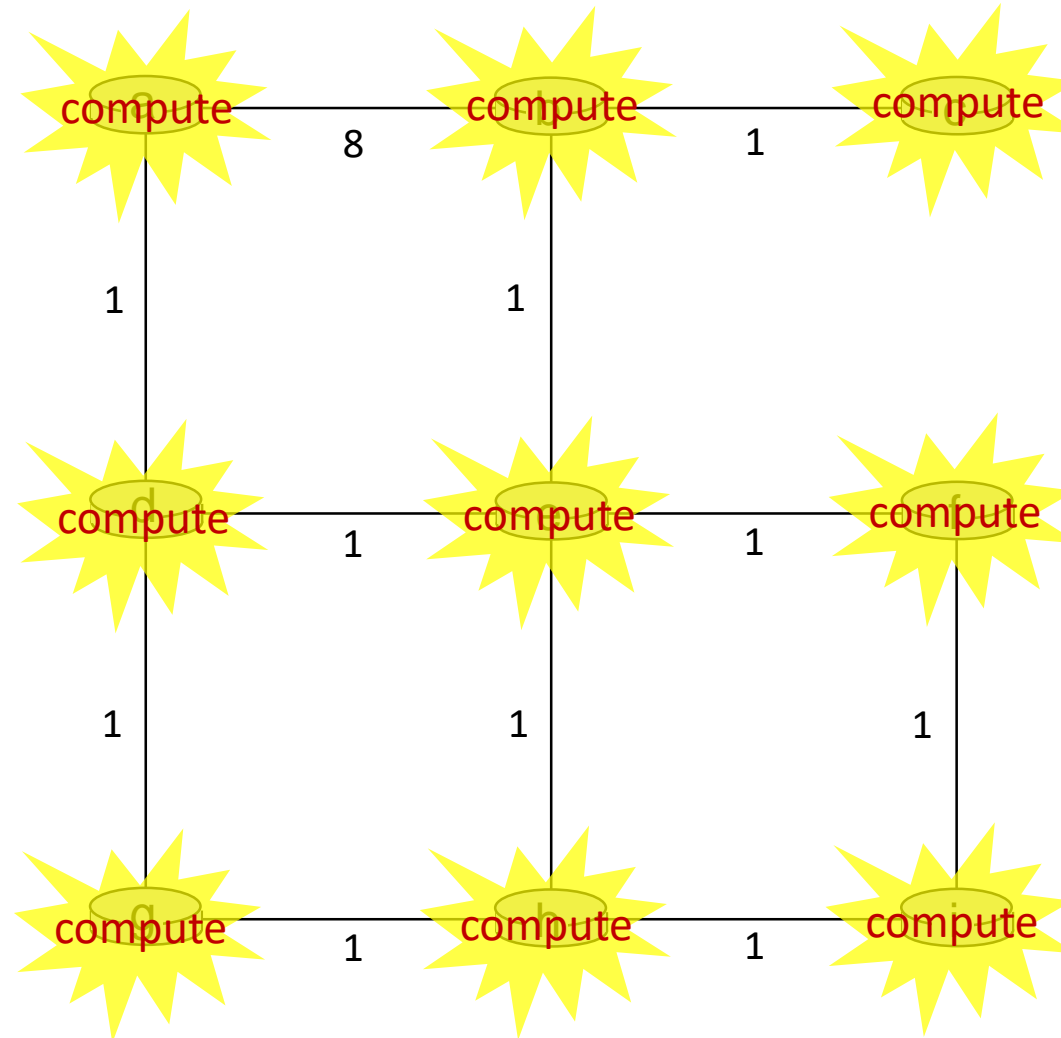
Distance vector example: iteration

t=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
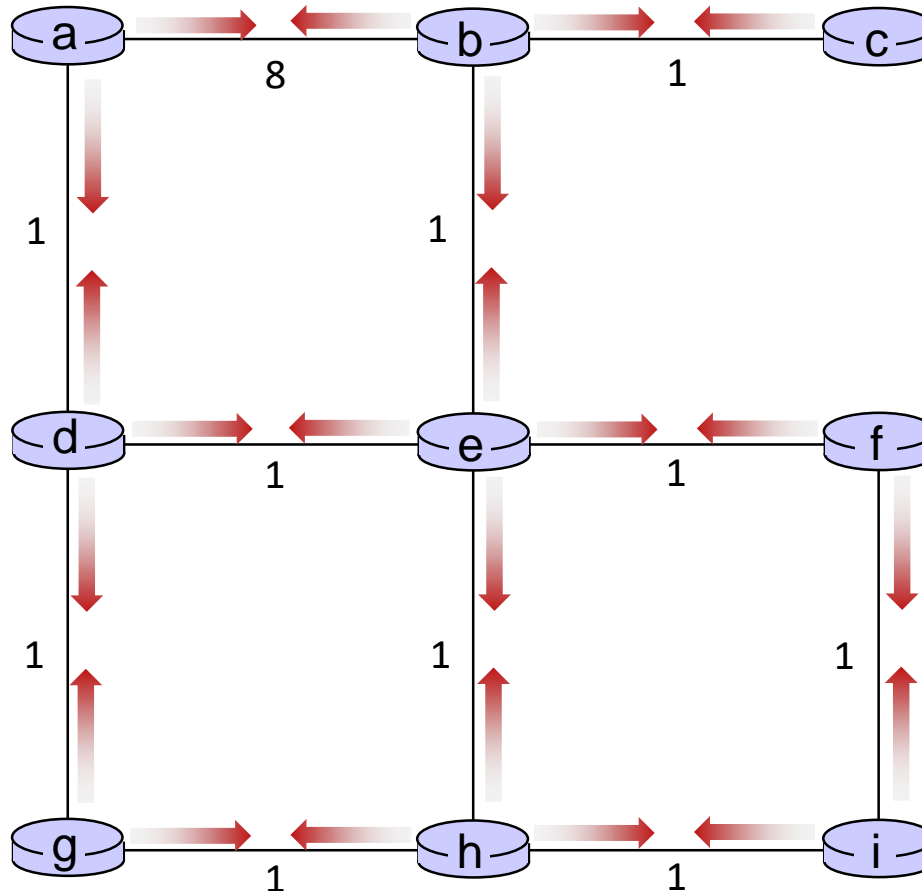- send their new local distance vector to neighbors

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
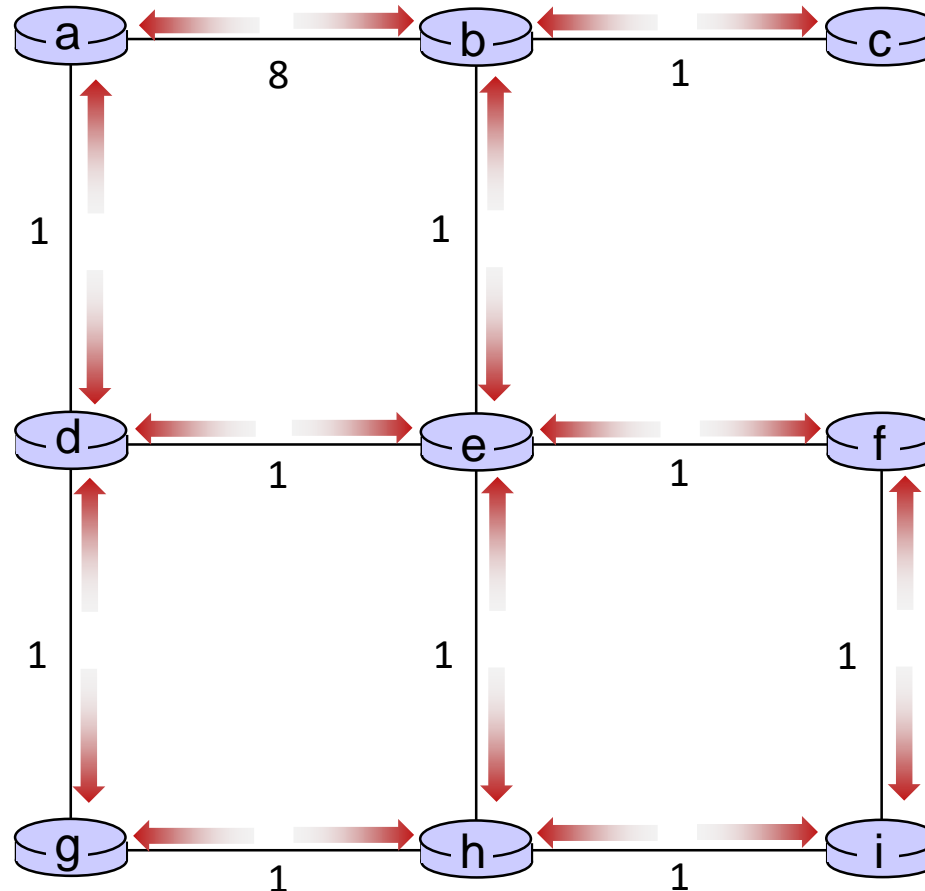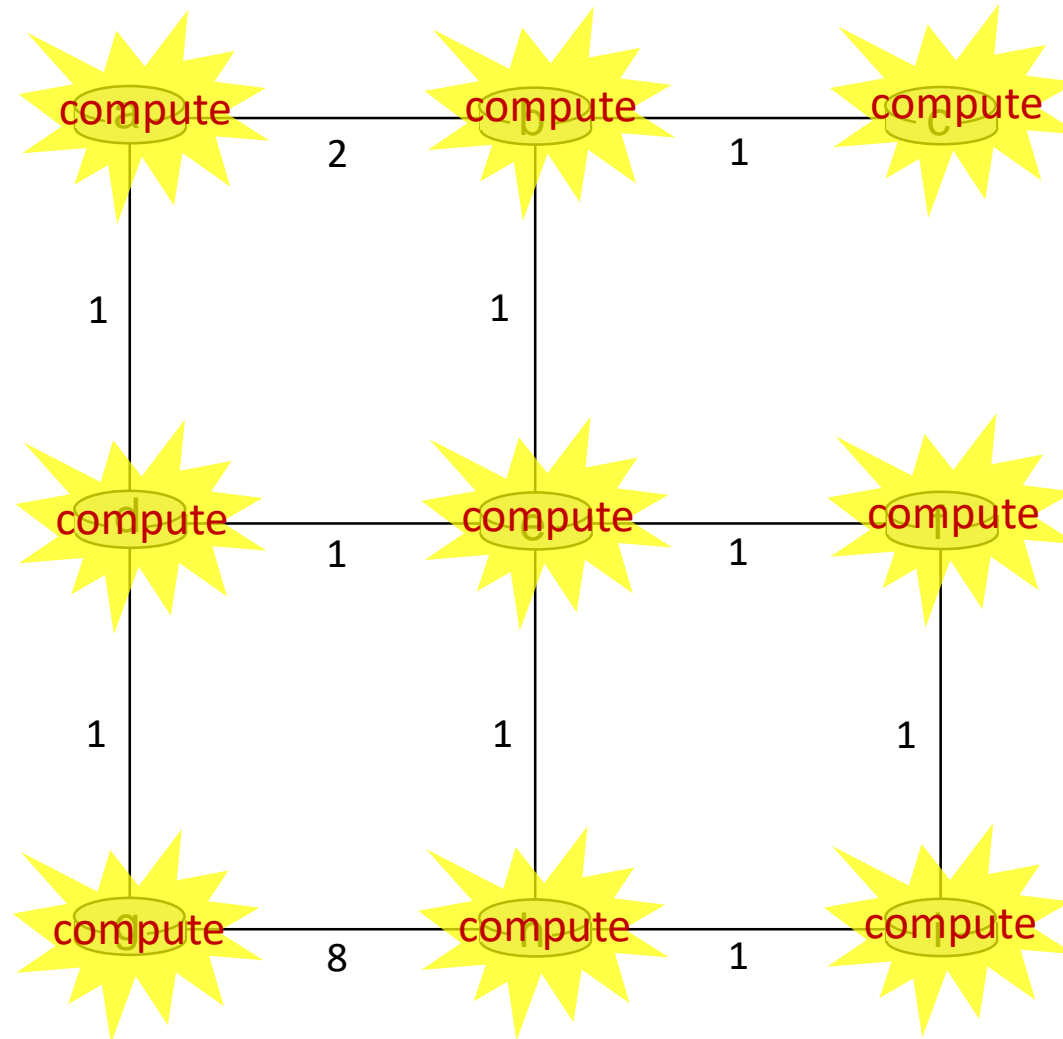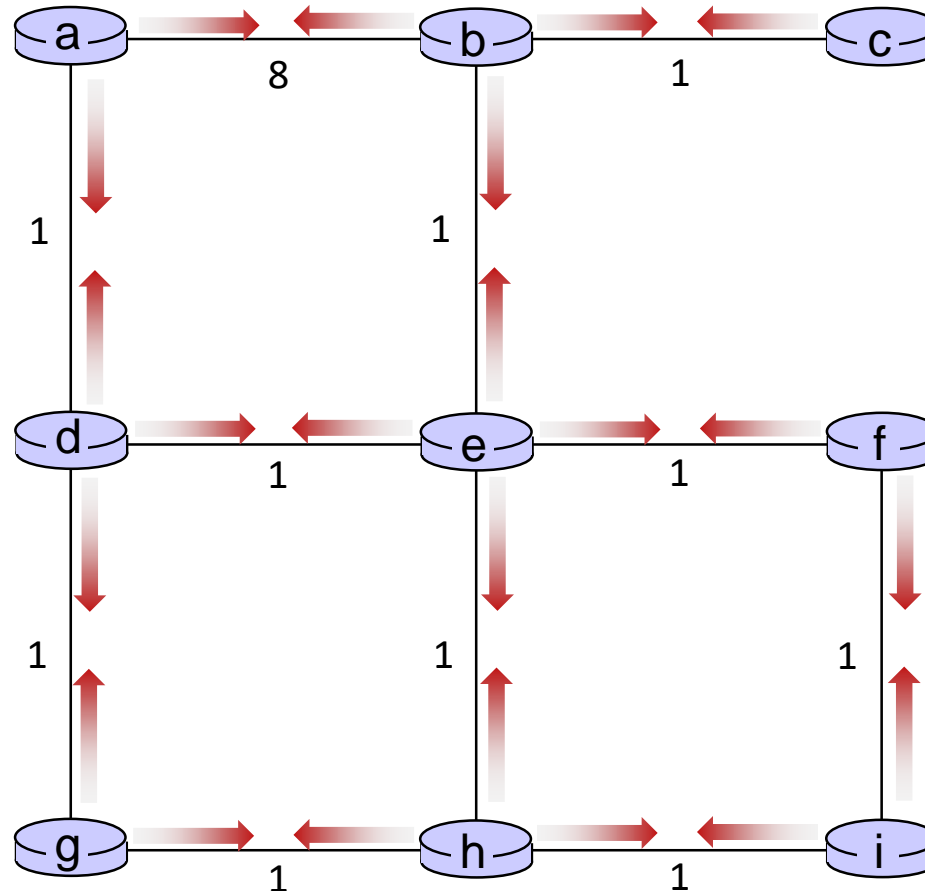
Distance vector example: iteration

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: c

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:

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

t=1

- b receives DVs from a, c, e

Distance vector example: c

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:

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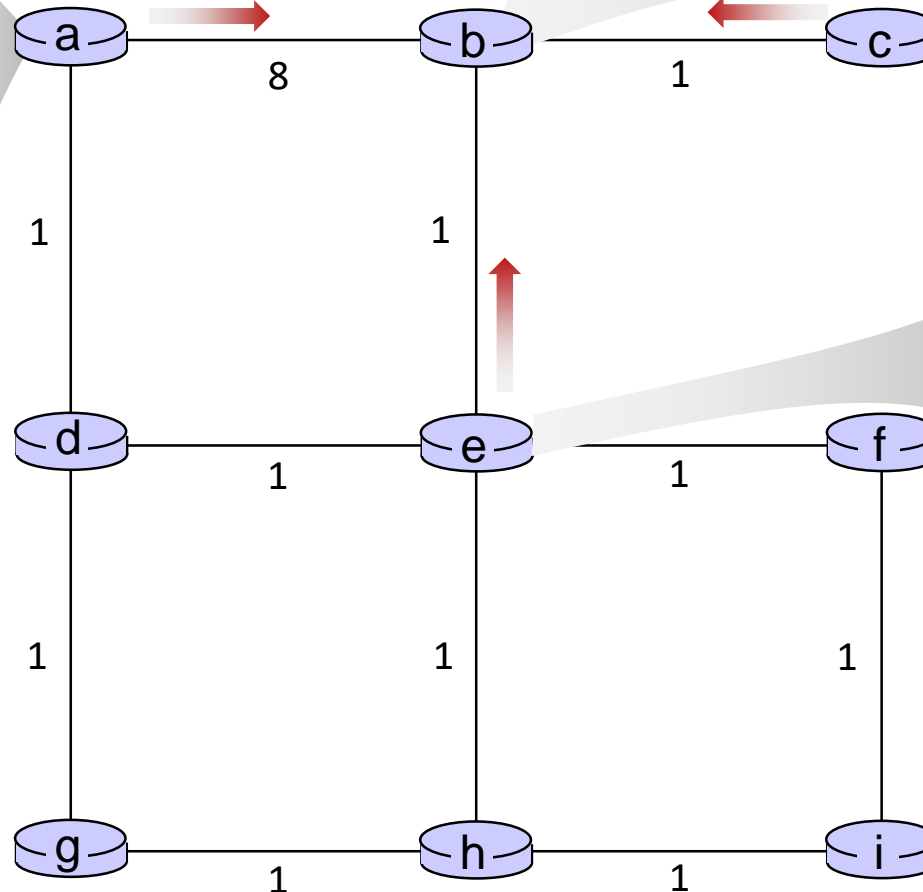$D_b(d) = \infty$ $D_b(h) = \infty$
$D_b(e) = 1$ $D_b(i) = \infty$

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:

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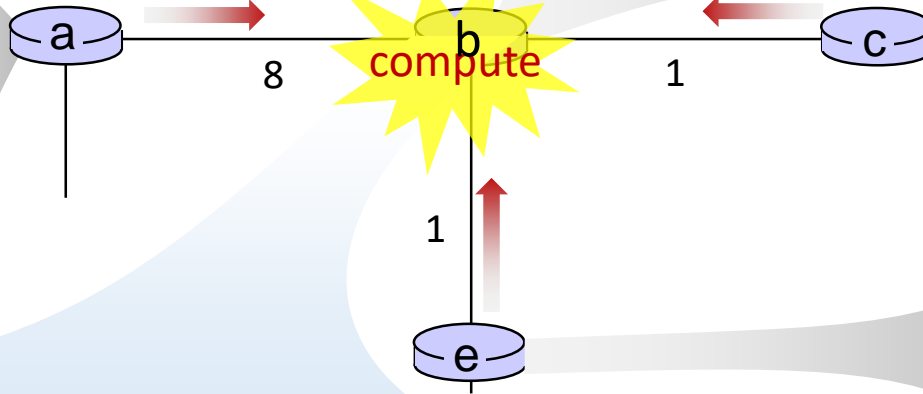$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

t=1

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

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

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

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

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

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

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

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

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

DV in b:

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

a ——8—— b (compute) ——1—— c

e ——1—— b

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:

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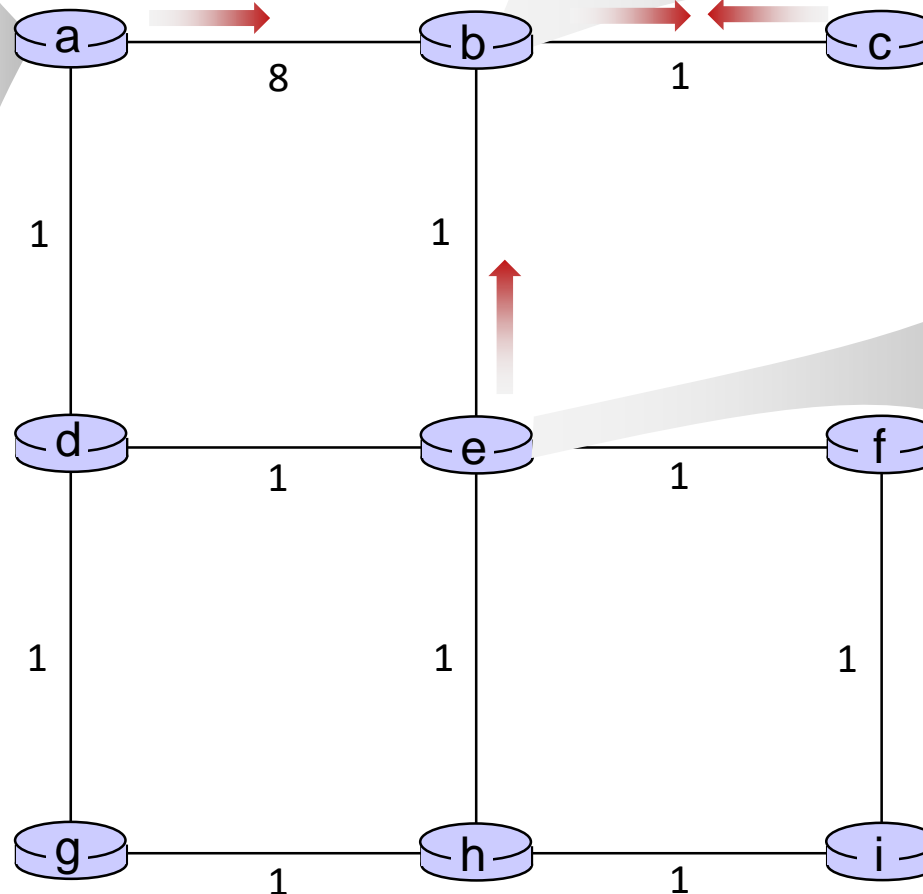$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

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 e:

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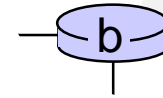$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

t=1

- c receives DVs from b

Distance vector example: c

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:

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

b — compute

1

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:

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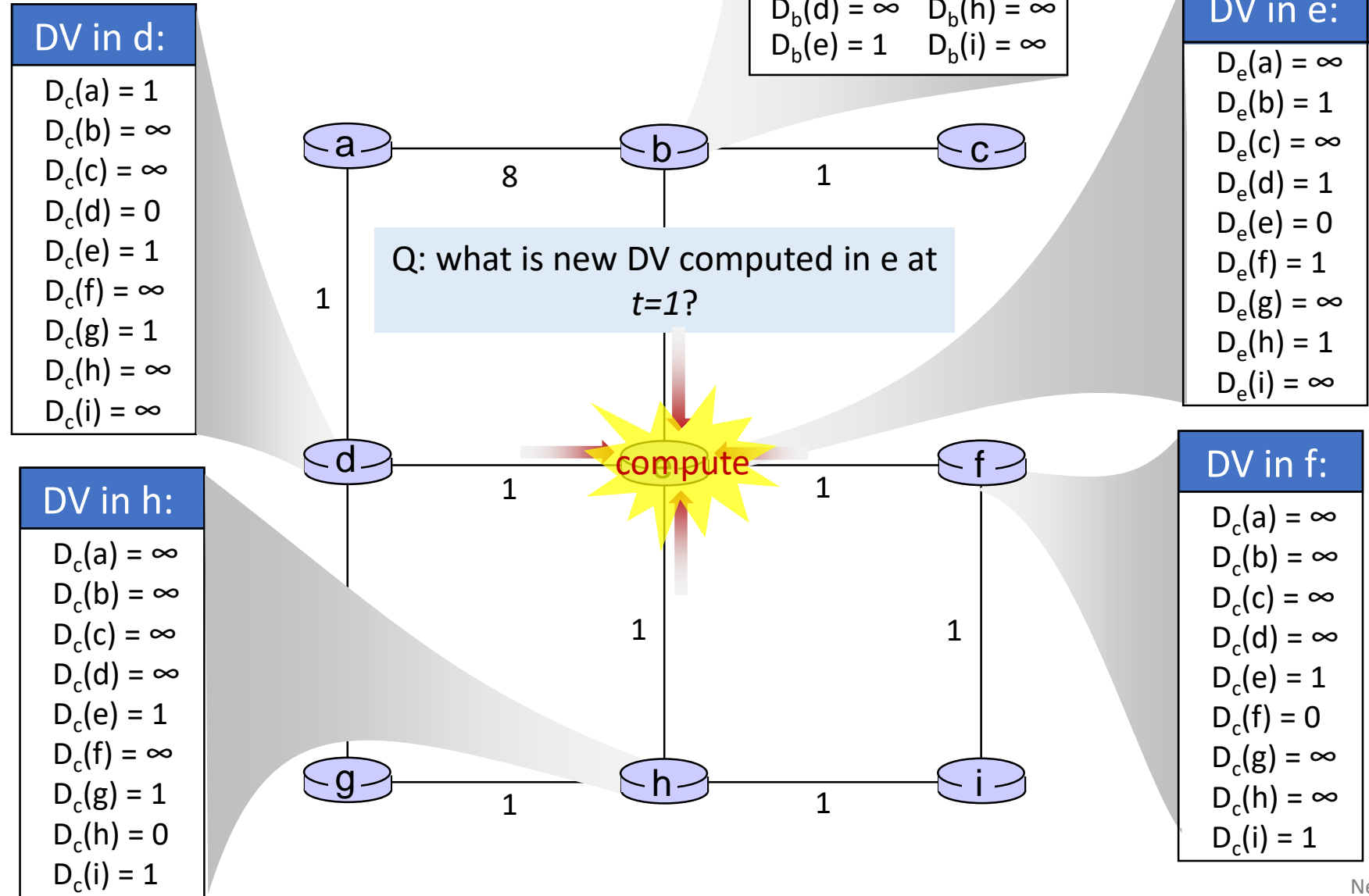$D_c(h) = \infty$
$D_c(i) = \infty$

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

Distance vector example: (

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:

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

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:

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

t=1

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

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

compute

a — 8 — b — 1 — c
1
d — 1 — compute — 1 — f
1   1
g — 1 — h — 1 — i
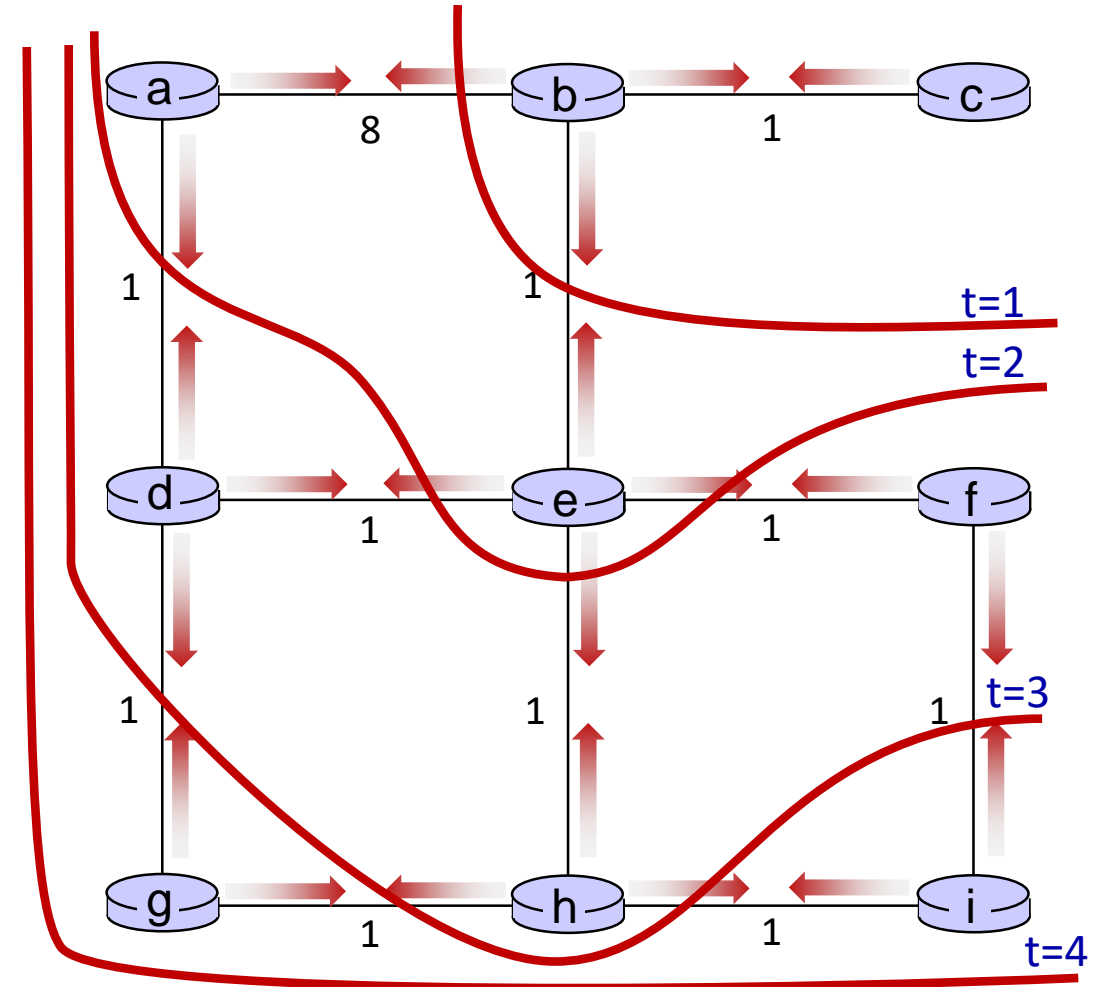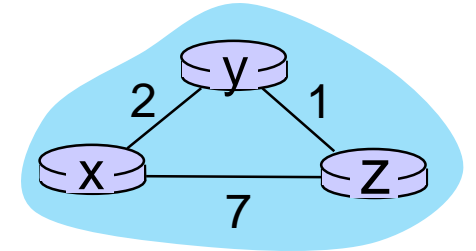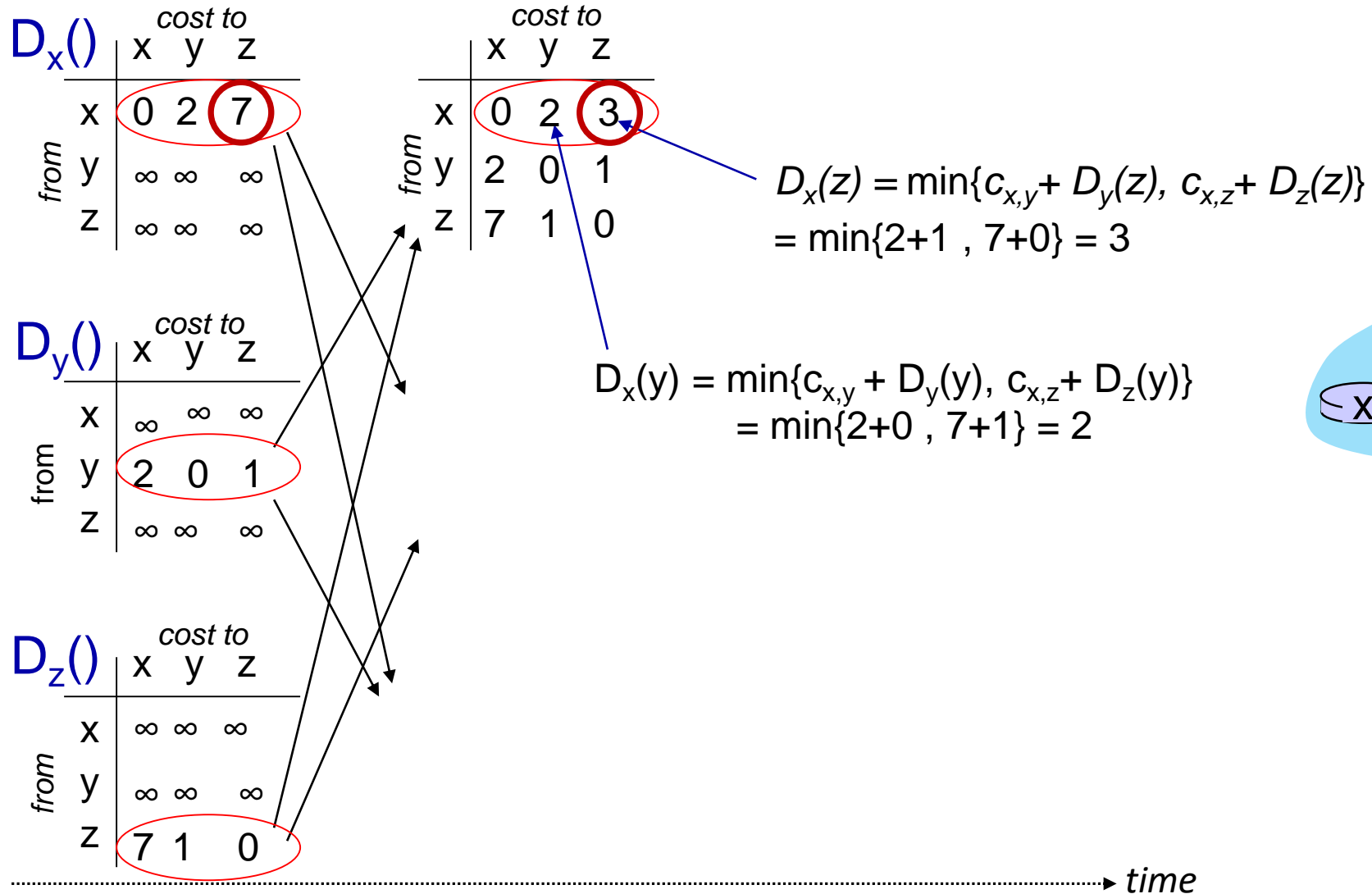
Distance vector: state information diffusion

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: another example

$D_x()$

cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from

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

from

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

$D_y()$

cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from

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

$D_z()$

cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from

→ *time*

Distance vector: another example
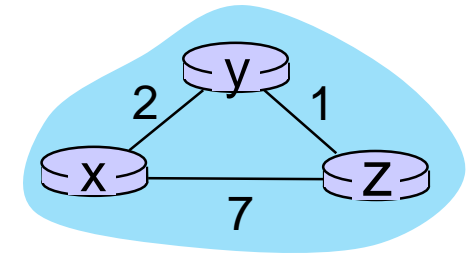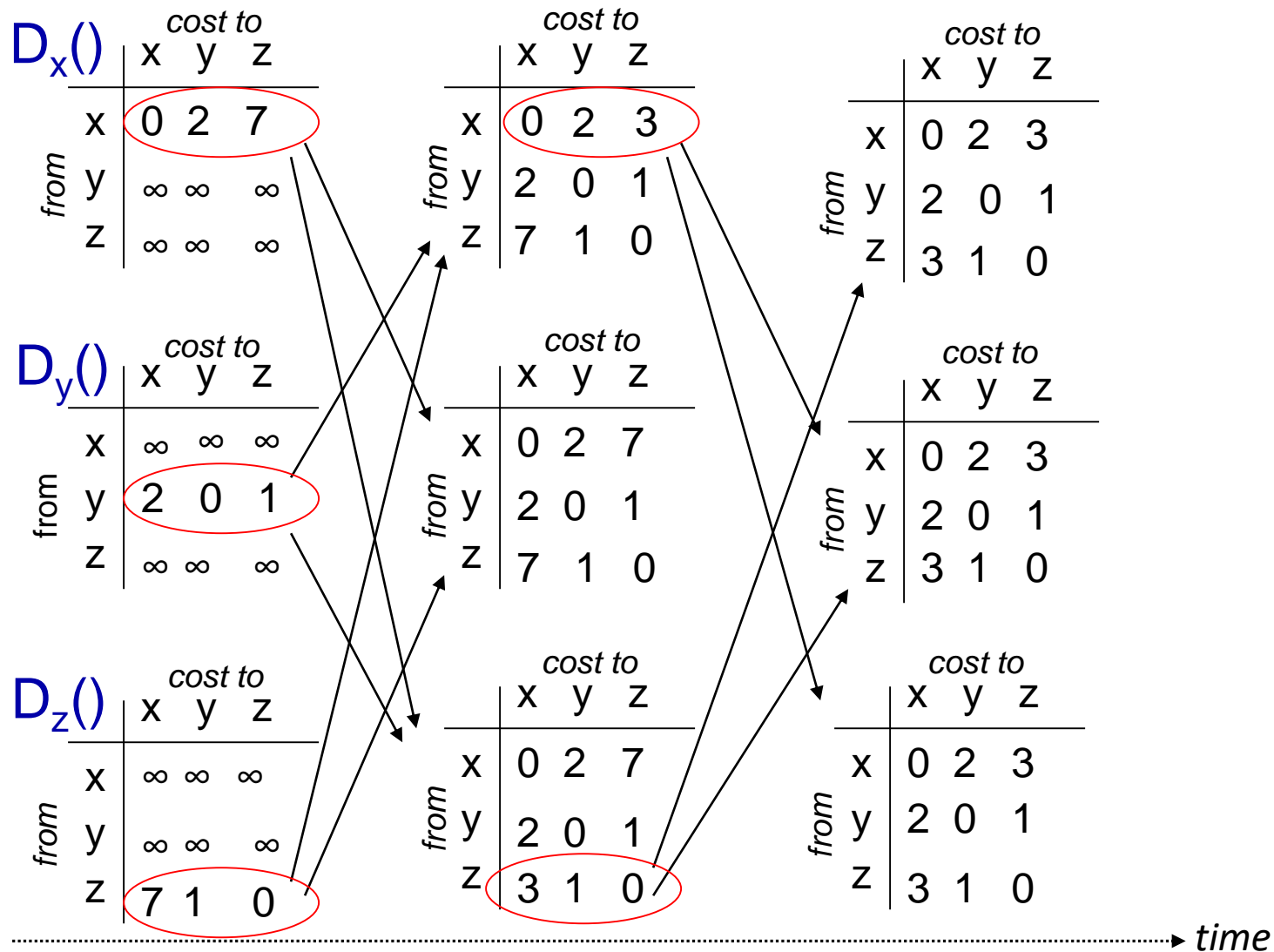
Comparison of LS and DV algorithms

message complexity

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

DV: exchange between neighbors; convergence time varies

speed of convergence

LS: O(n^2) algorithm, O(n^2) messages
- may have oscillations

DV: convergence time varies
- may have routing loops
- count-to-infinity problem

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

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

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