CS3640

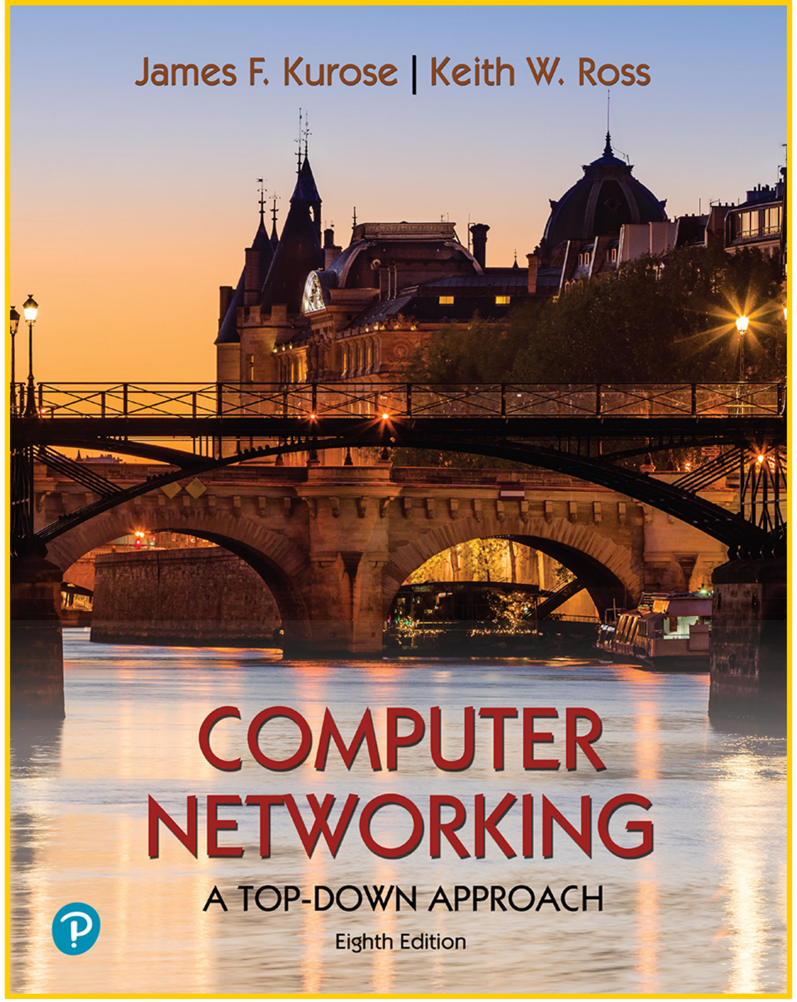
Network Layer (4): Routing Algorithms

Prof. Supreeth Shastri
Computer Science
The University of Iowa

Lecture goals

a technical deep-dive into two classes of routing algorithms used in the Internet

- *Link-State algorithm*
- *Distance Vector algorithm*

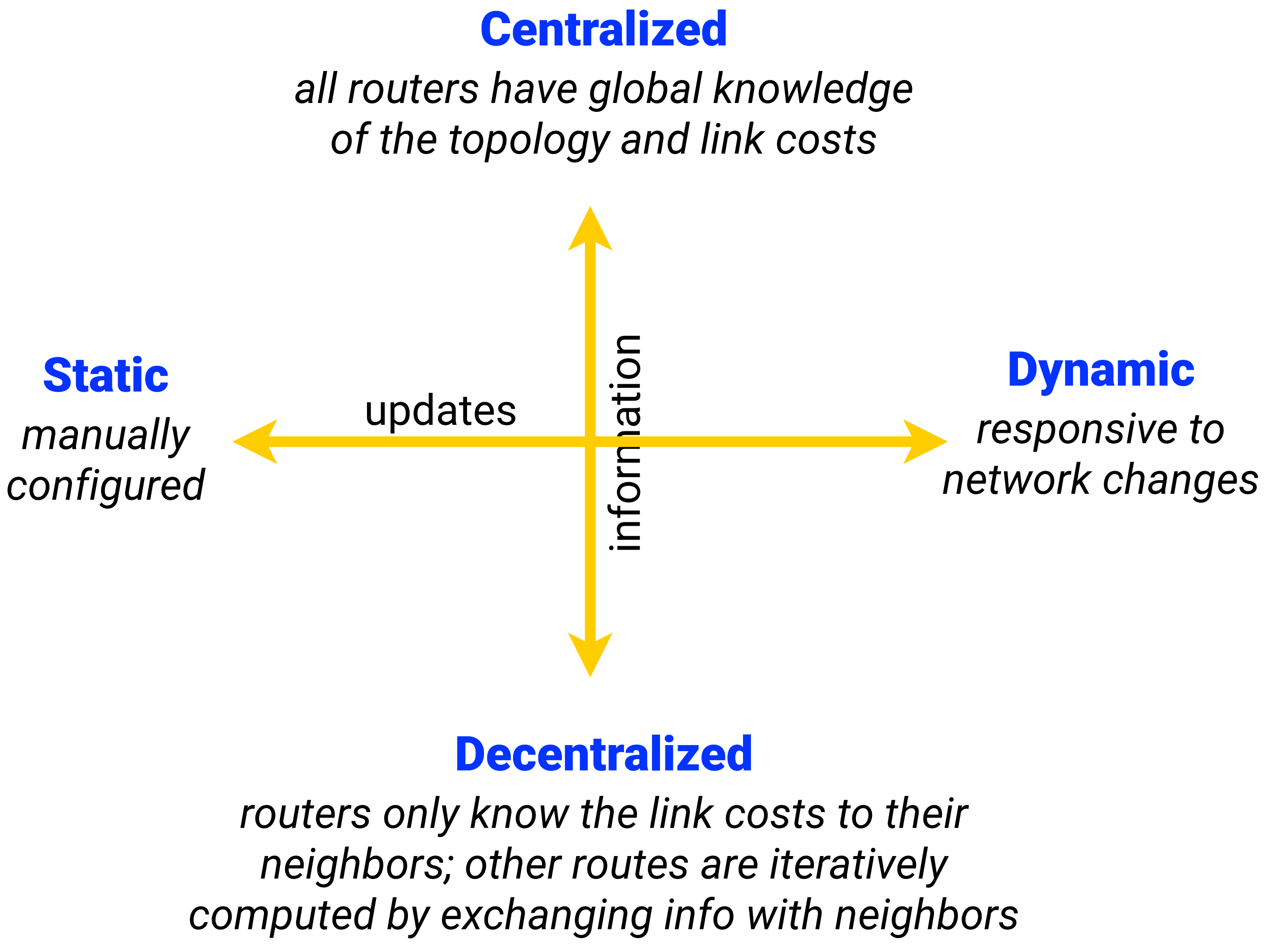


Chapters 5.1 - 5.2

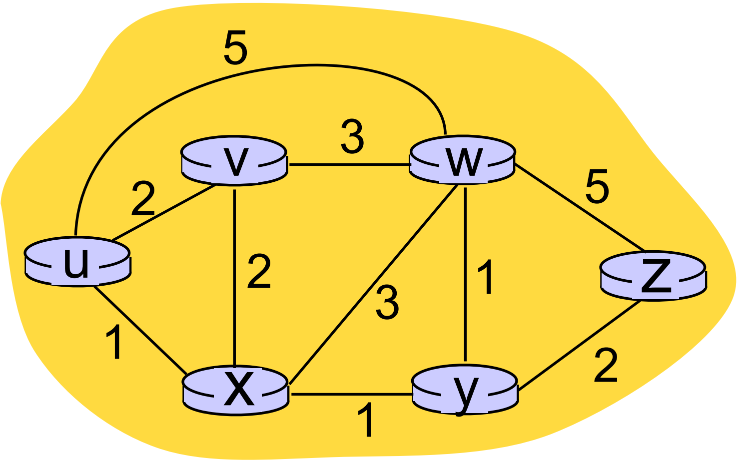
Routing Algorithms

Goal: determine “good” paths 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”, and so on!



Representing Routing via Graph Abstraction



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

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

cost is defined by network operators: they could all be set to 1, or set to reflect a network metric such as bandwidth or 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)* }

Dijkstra's Algorithm

(a link-state routing algorithm)

Dijkstra's Link-State Routing Algorithm

- **Centralized**: network topology, link costs known to *all* nodes (which is accomplished via a *link state broadcast* such that all nodes have same info)
- computes least cost paths from one node ("source") to all other nodes, which generates the *forwarding table* for that node
- **Iterative**: after *k* iterations, we know least cost path to *k* destinations

notation

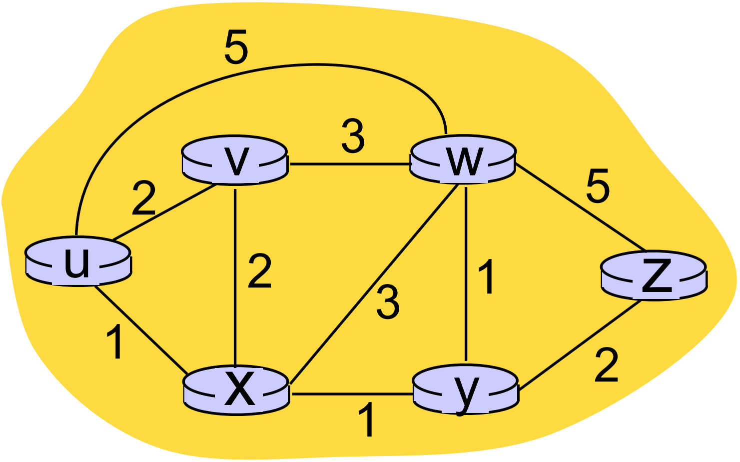
- $C_{a,b}$: direct link cost from node *a* to *b*; = ∞ if not direct neighbors
- *D(a): current* estimate of cost of least-cost-path from source to destination *a*
- *p(a):* predecessor node along path from source to *a*
- *N':* set of nodes whose least-cost-path *definitively* known

```
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) = ∞

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

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						
2						
3						
4						
5						

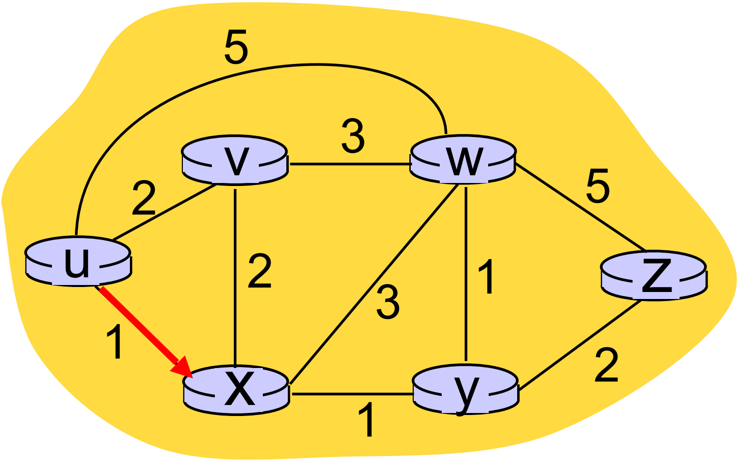


Initialization (step 0):

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

Dijkstra's algorithm: an example

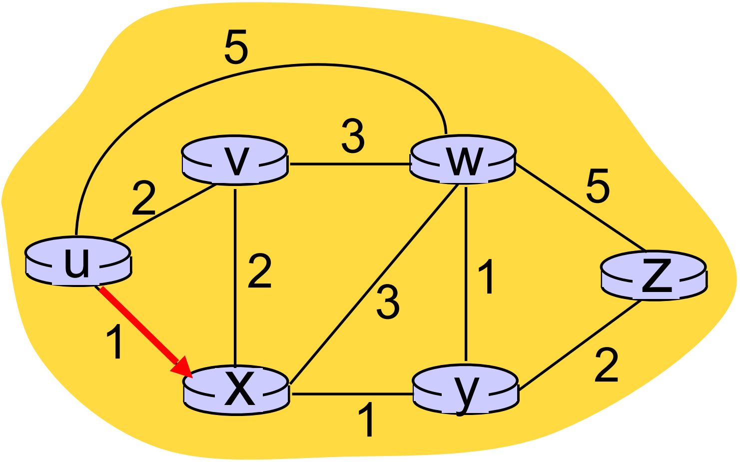
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux					
2						
3						
4						
5						



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

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						
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'* :

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

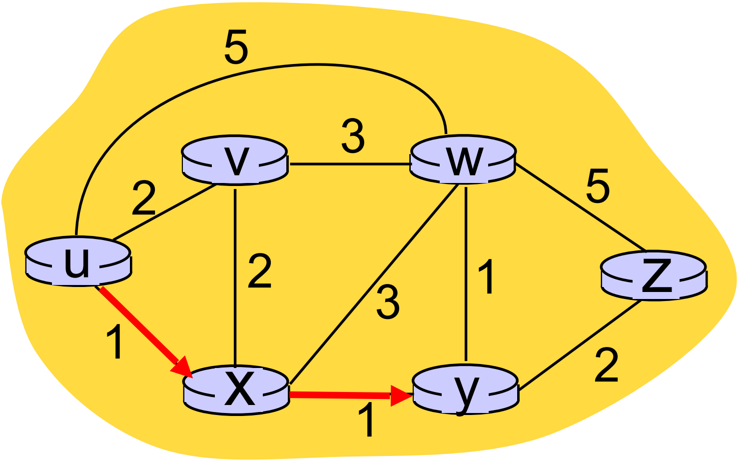
$D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$

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

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

Dijkstra's algorithm: an example

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					
3						
4						
5						



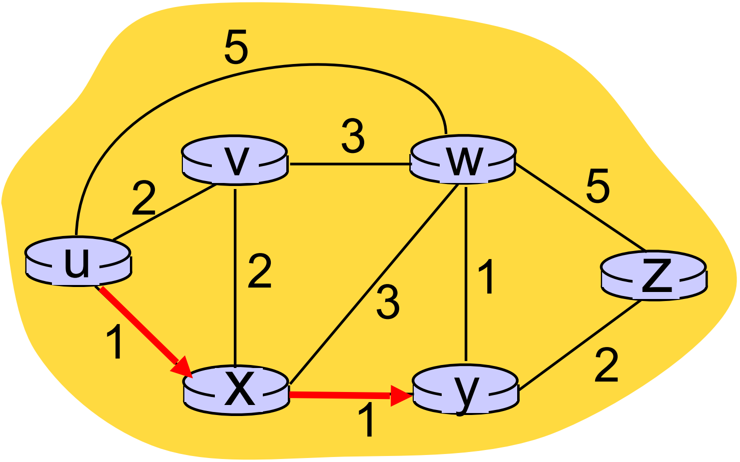
8 *Loop*

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'* :

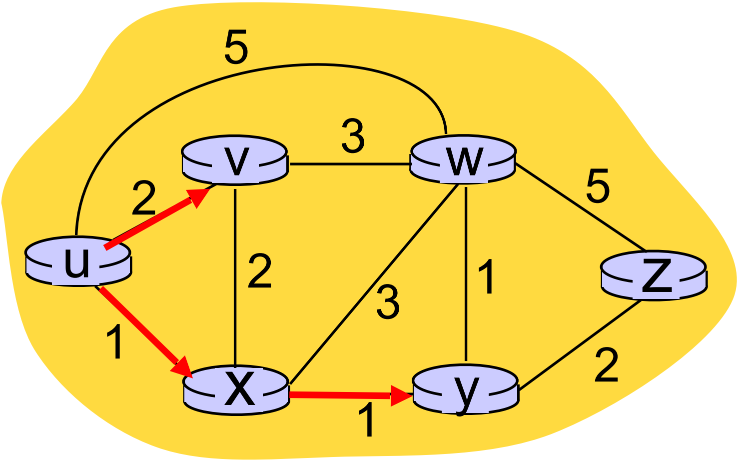
$$D(b) = \min (D(b), D(a) + c_{a,b})$$

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



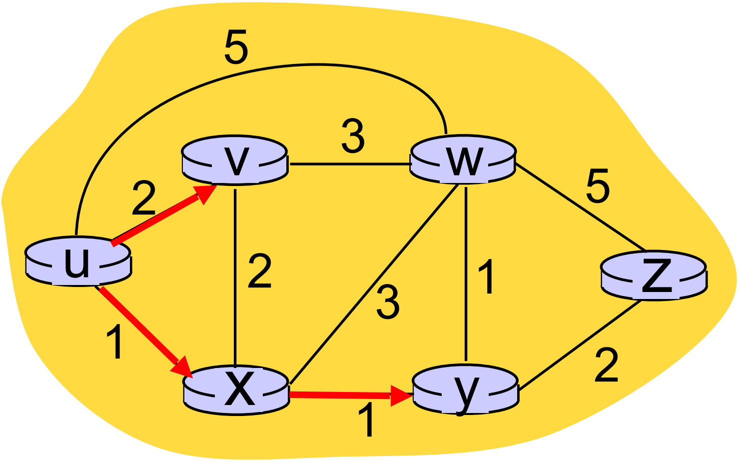
Loop

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

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						



Loop

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

add a to N'

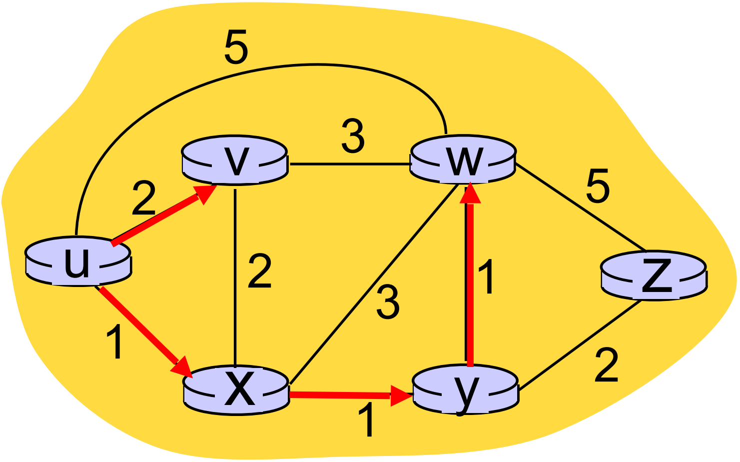
update $D(b)$ for all b adjacent to a and not in N' :

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

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

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						



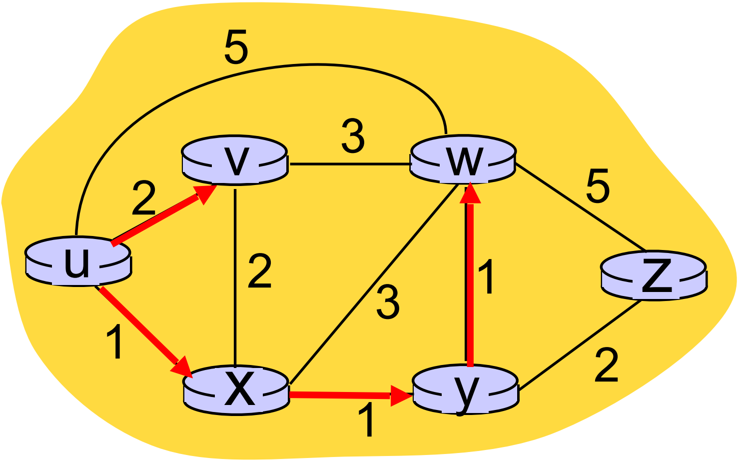
8 *Loop*

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

10 add *a* to *N'*

Dijkstra’s algorithm: an example

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						



Loop

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

add *a* to *N'*

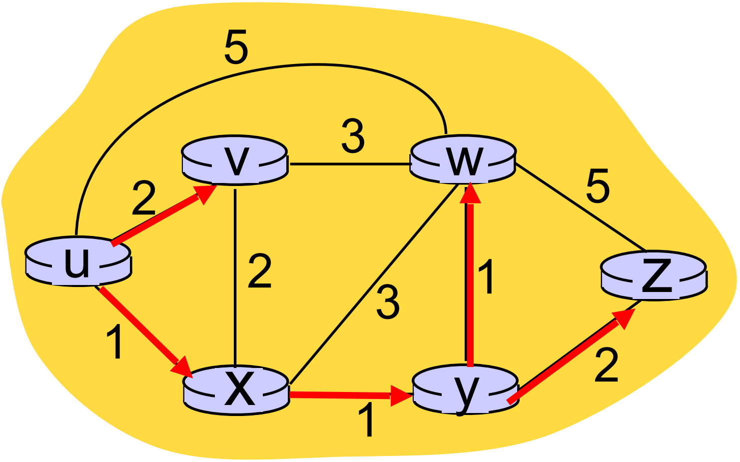
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

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

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					



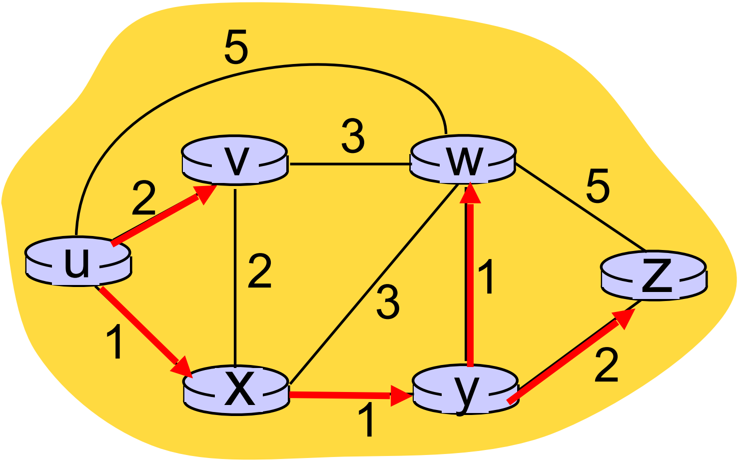
8 *Loop*

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

10 add *a* to *N'*

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					



Loop

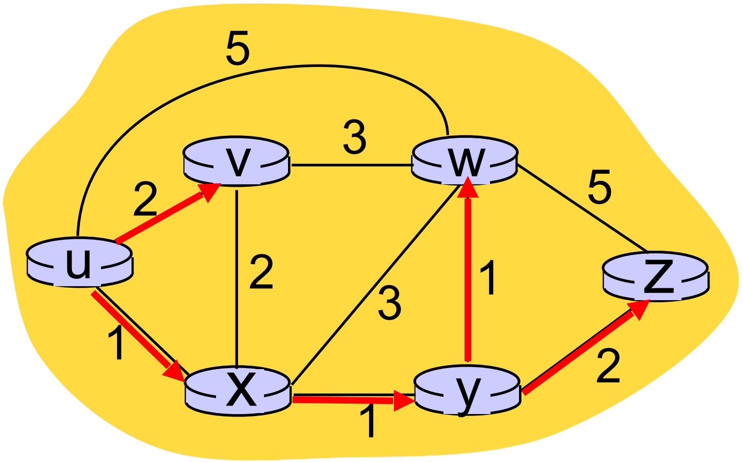
find a not in N' such that $D(a)$ is a minimum

add a to N'

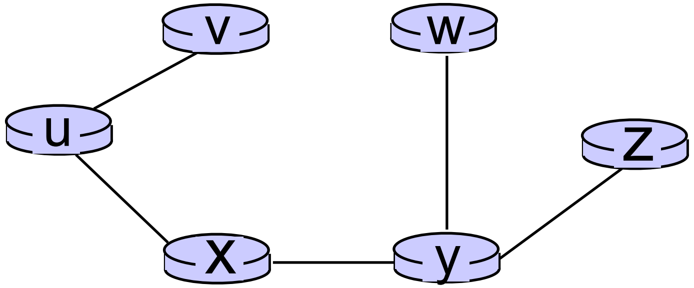
update $D(b)$ for all b adjacent to a and not in N' :

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

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*

Comparing Dijkstra and Bellman-Ford Algorithms

	Dijkstra (LS)	Bellman-Ford (DV)
Algorithm structure	Centralized	
Speed of convergence	$O(N^2)$	
Application	Routing within autonomous systems	
Robustness	Route oscillations	

Bellman-Ford Algorithm

(a distance-vector routing algorithm)

Bellman-Ford equation

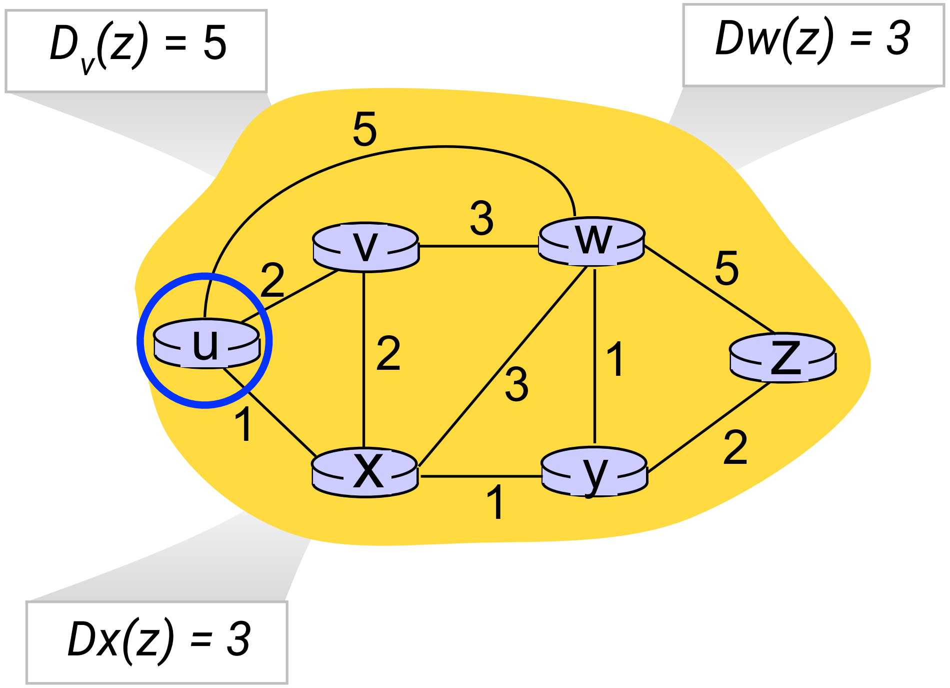
Let $D_x(y)$: cost of least-cost path from x to y.

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

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

direct cost of link from x to v

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



Suppose that *u*'s neighboring nodes *(x,v, and w)* know their cost for destination *z*:

Bellman-Ford equation says:

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

Bellman-Ford Distance Vector Algorithm

Each node:

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

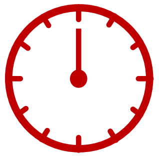
recompute my DV estimates using DV received from neighbor

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

Key Characteristics

- **Distributed/Decentralized**: routers do not need global knowledge of network topology
- **Iterative**: routes are computed iteratively in response to link cost change or DV updates from neighbors
- **Asynchronous**: routers do not need to synchronize on their route computations, or DV announcements
- **Self-stopping**: neighbors communicate only if necessary, and stop when no notifications are received

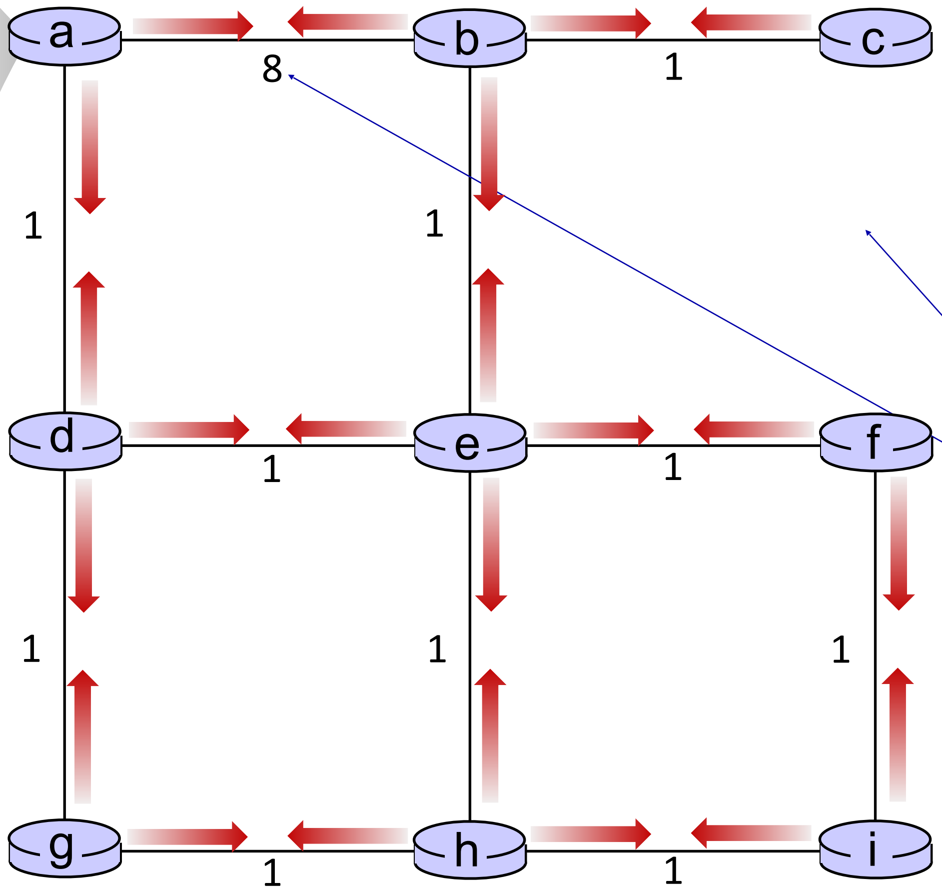
Bellman-Ford Algorithm: an example



t=0

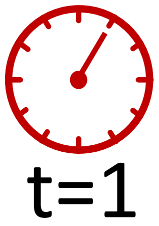
- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

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

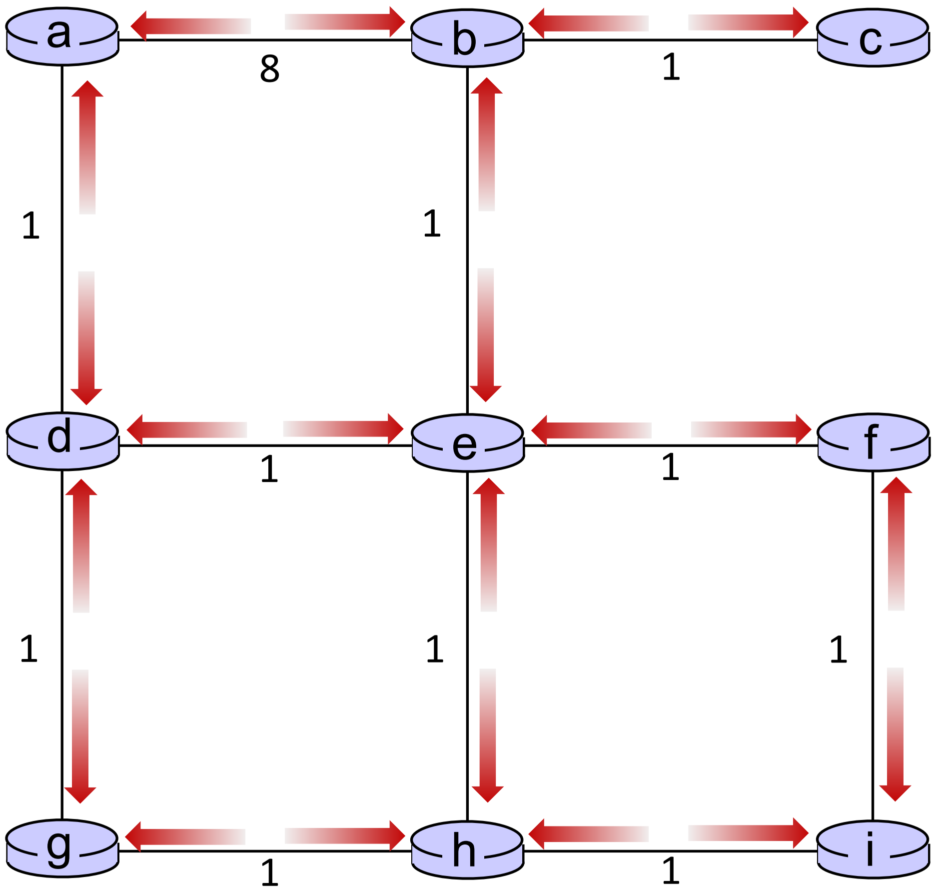


A few asymmetries:
- missing link
- larger cost

Bellman-Ford Algorithm: how iterations work



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

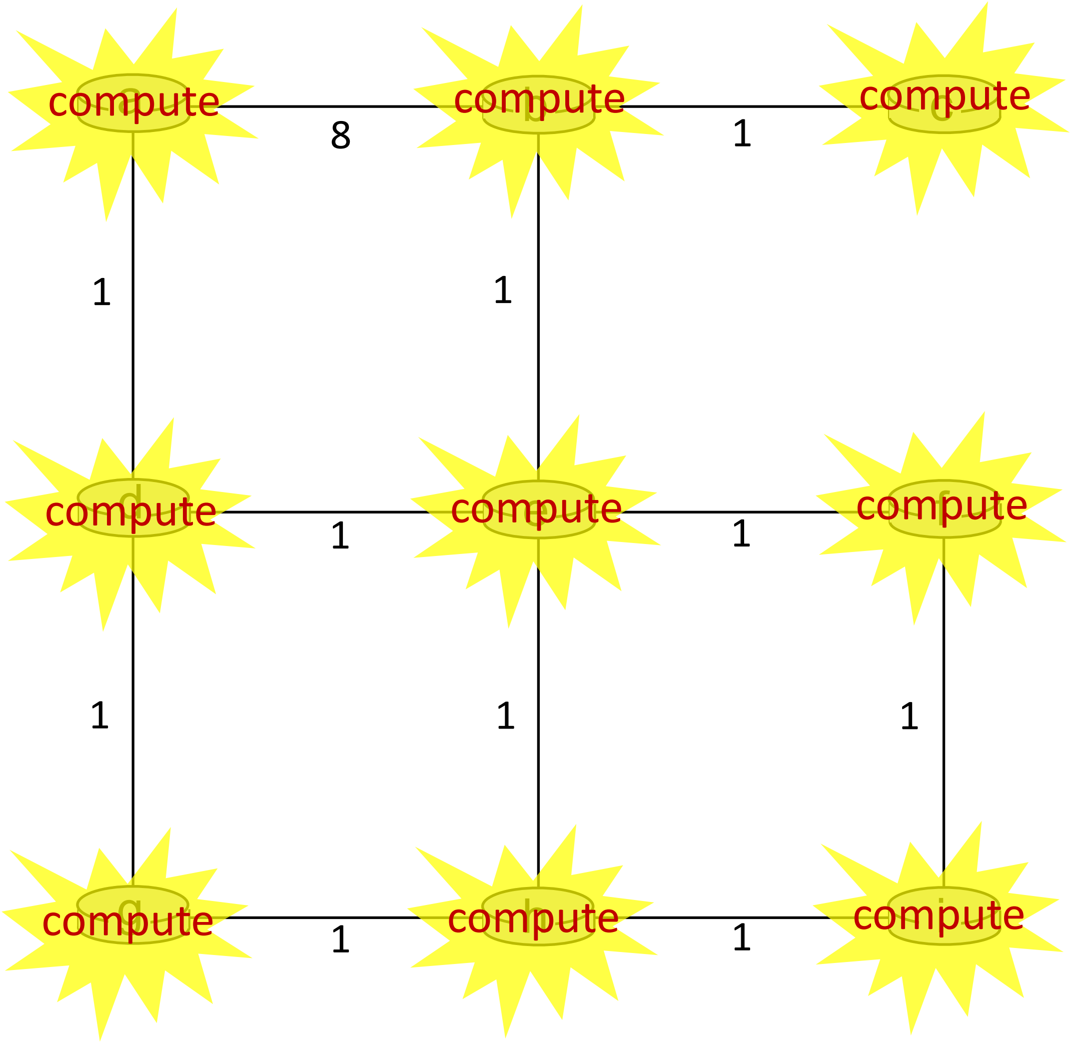


Bellman-Ford Algorithm: how iterations work

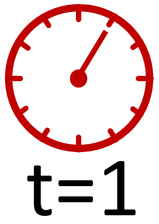
t=1

All nodes:

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

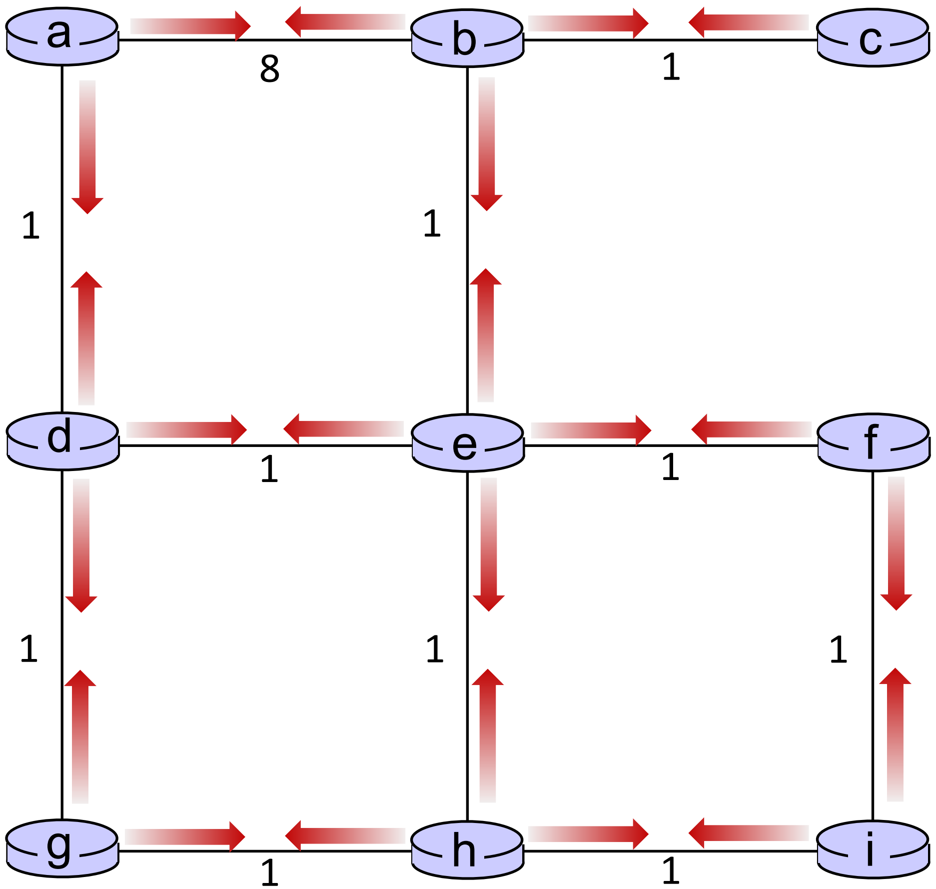


Bellman-Ford Algorithm: how iterations work

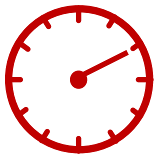


All nodes:

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

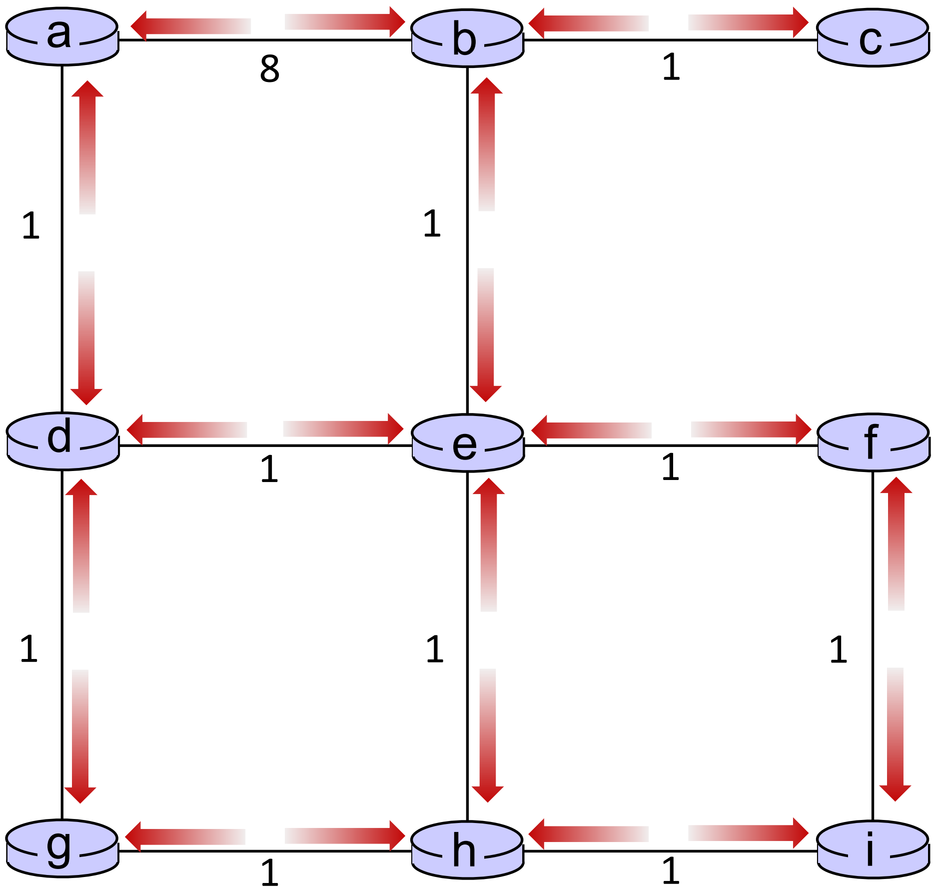


Bellman-Ford Algorithm: how iterations work

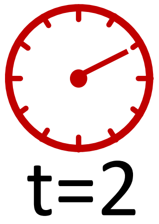


t=2

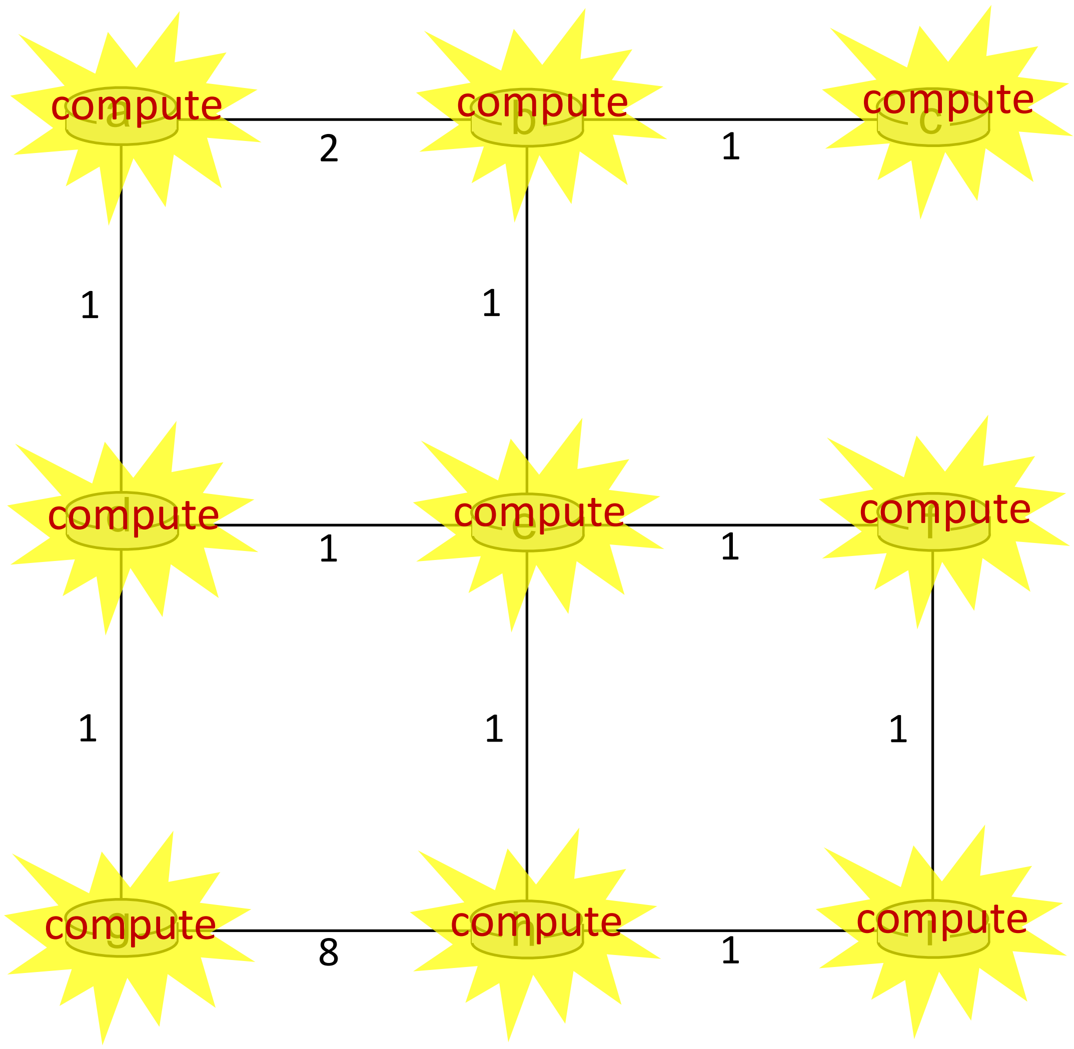
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



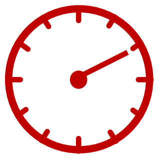
Bellman-Ford Algorithm: how iterations work



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



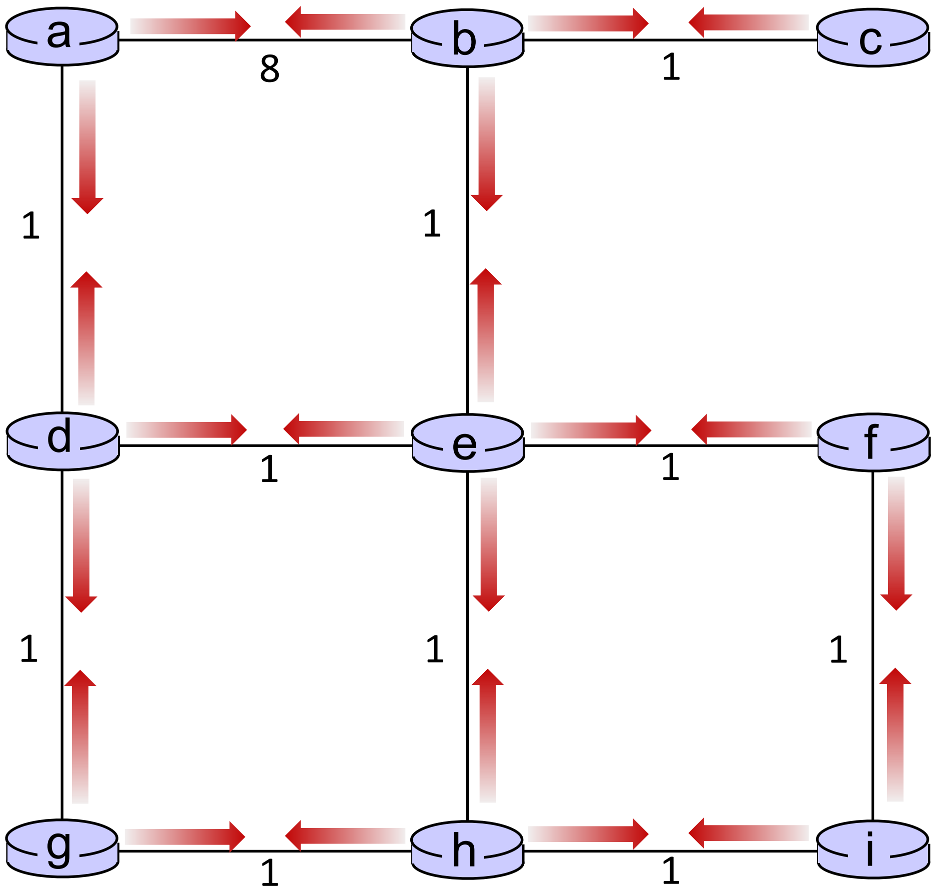
Bellman-Ford Algorithm: how iterations work



t=2

All nodes:

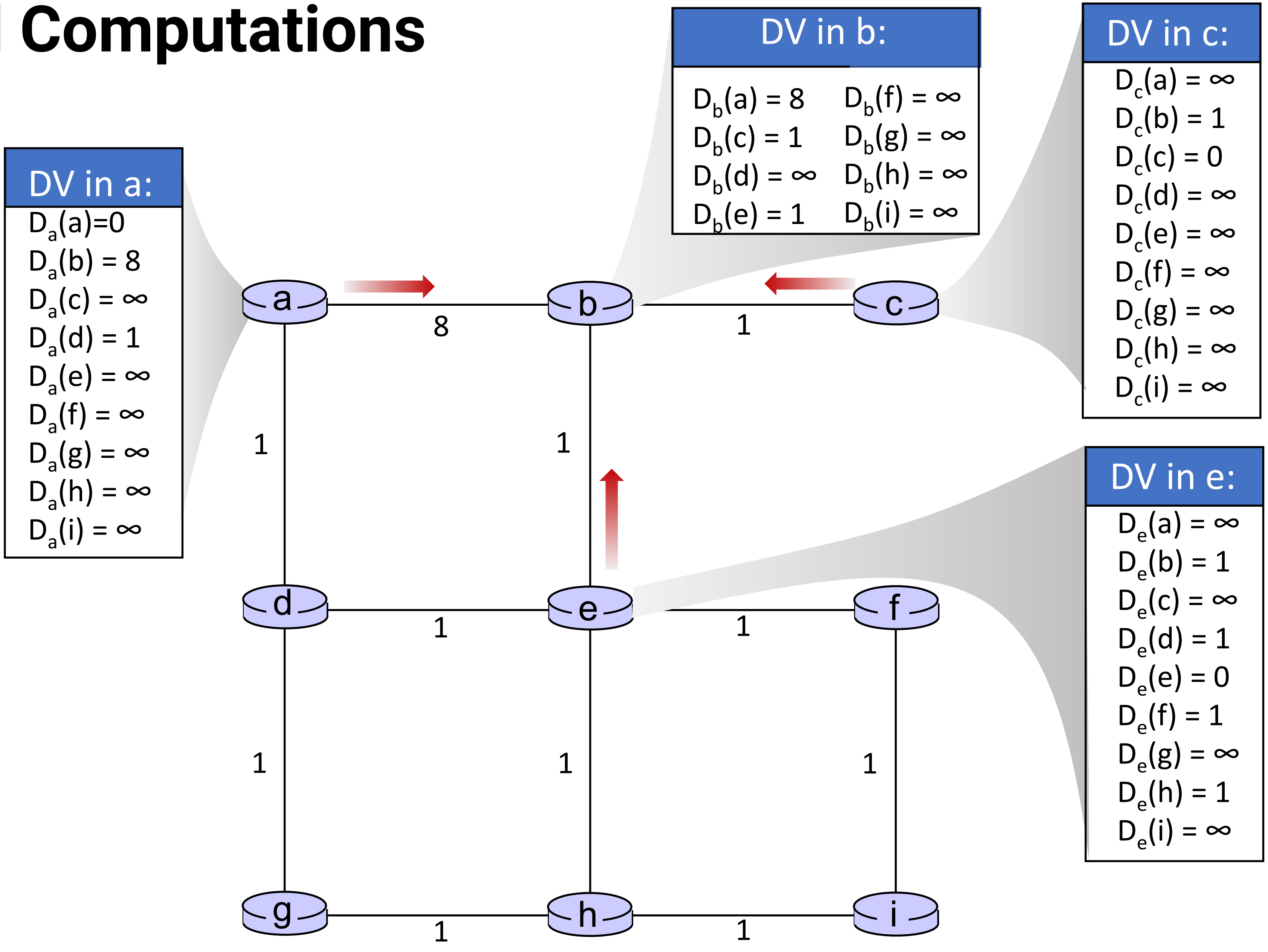
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



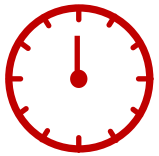
Bellman-Ford Computations

t=0

- b receives DVs from a, c, e



Bellman-Ford Computations



t=1

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

DV in a:

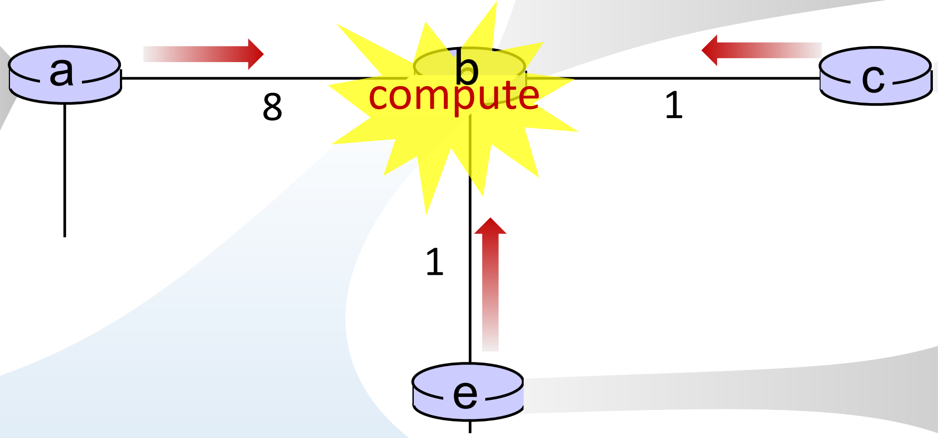
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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:

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

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

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

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

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

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

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

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

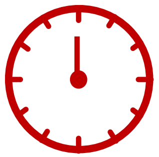
$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$

New DV in b

DV in b:

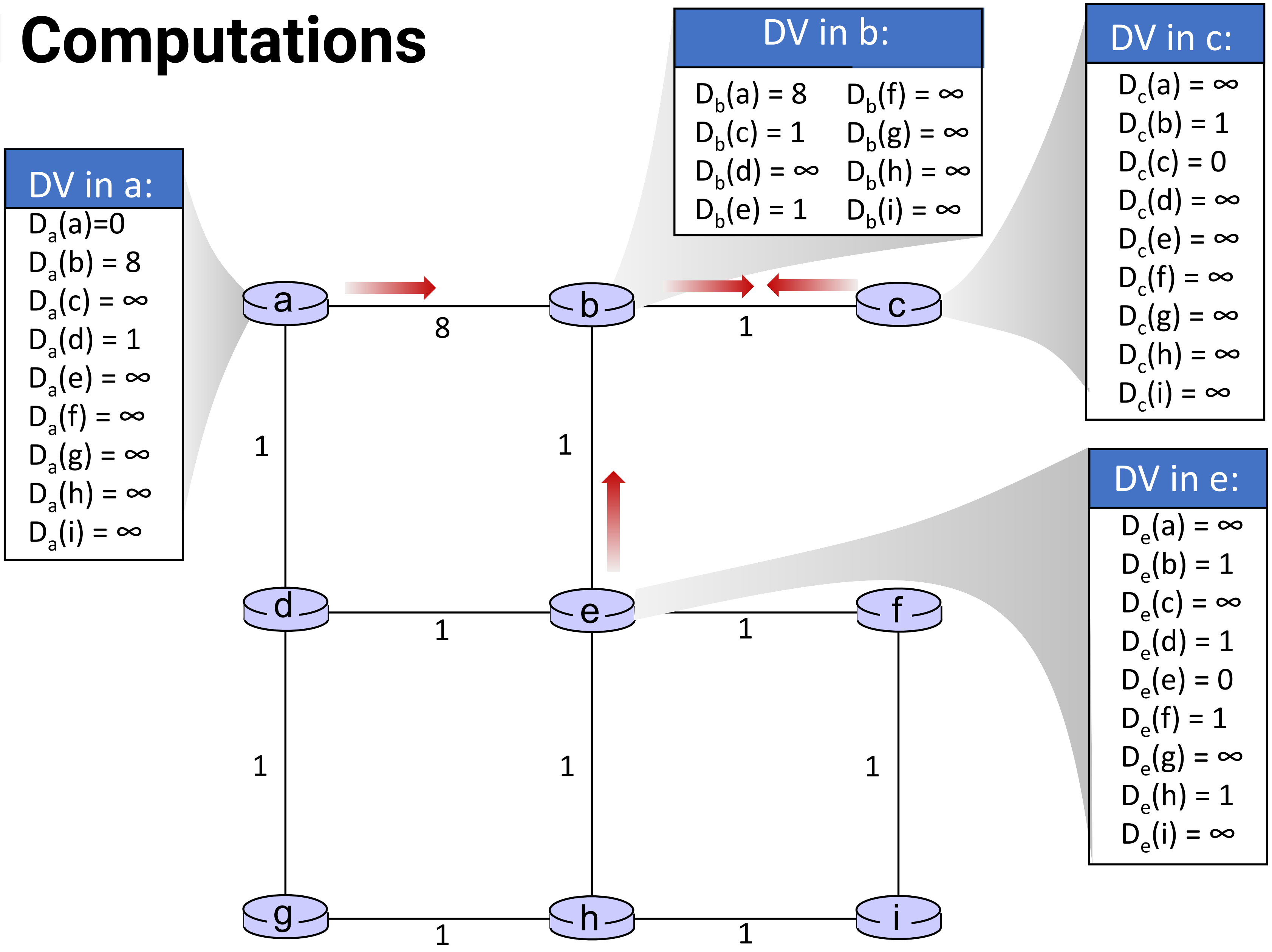
$D_b(a) = 8$	$D_b(f) = 2$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = 2$	$D_b(h) = 2$
$D_b(e) = 1$	$D_b(i) = \infty$

Bellman-Ford Computations



t=1

- c receives DVs from b



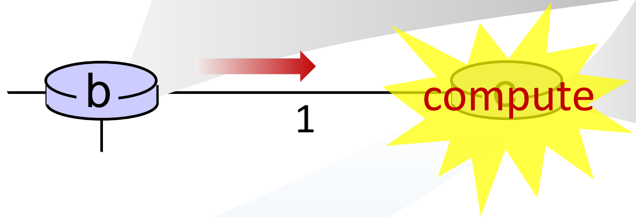
Bellman-Ford Computations

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:

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

Bellman-Ford: Iterative Information Propagation

Iterative communication, computation steps diffuses information through network:

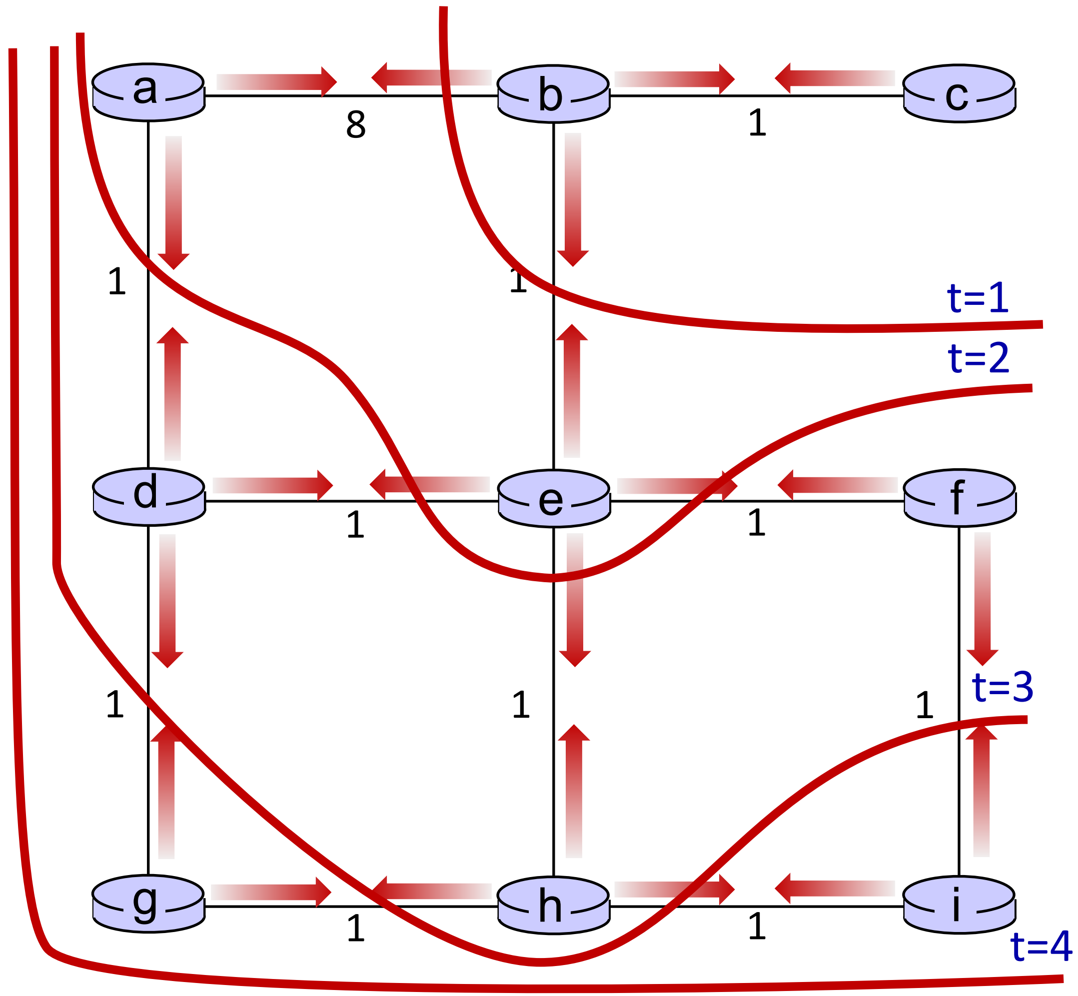
t=0 c's state at t=0 is at c only

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



Comparing Dijkstra and Bellman-Ford Algorithms

	Dijkstra (LS)	Bellman-Ford (DV)
Algorithm structure	Centralized	Decentralized
Speed of convergence	$O(N^2)$	slower than Dijkstra; $O(N*E)$ in worst
Application	Routing within autonomous systems	Routing across autonomous systems
Robustness	Route oscillations	Routing loops; Count-to-infinity

Exercise for you: read about these three routing problems from the textbook

Spot Quiz (ICON)