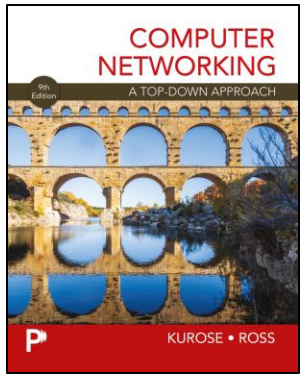


CSC 361
Computer Networks
Network Layer
(Chapter 5 – Control Plane)

Wenjun Yang

Summer 2026

Network Layer: "Control Plane" Roadmap (1 of 10)

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

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

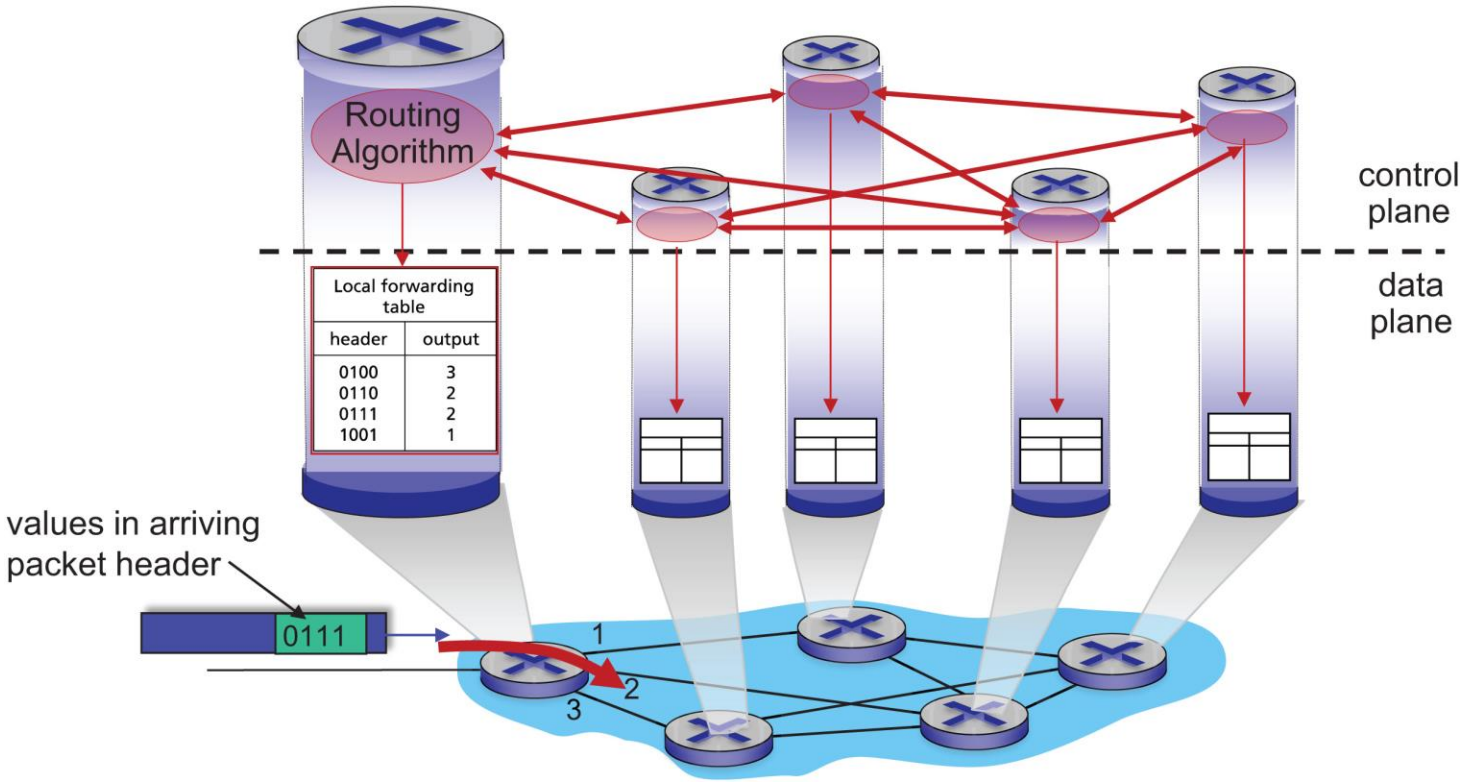
    control plane

Two approaches to structuring network control plane:

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

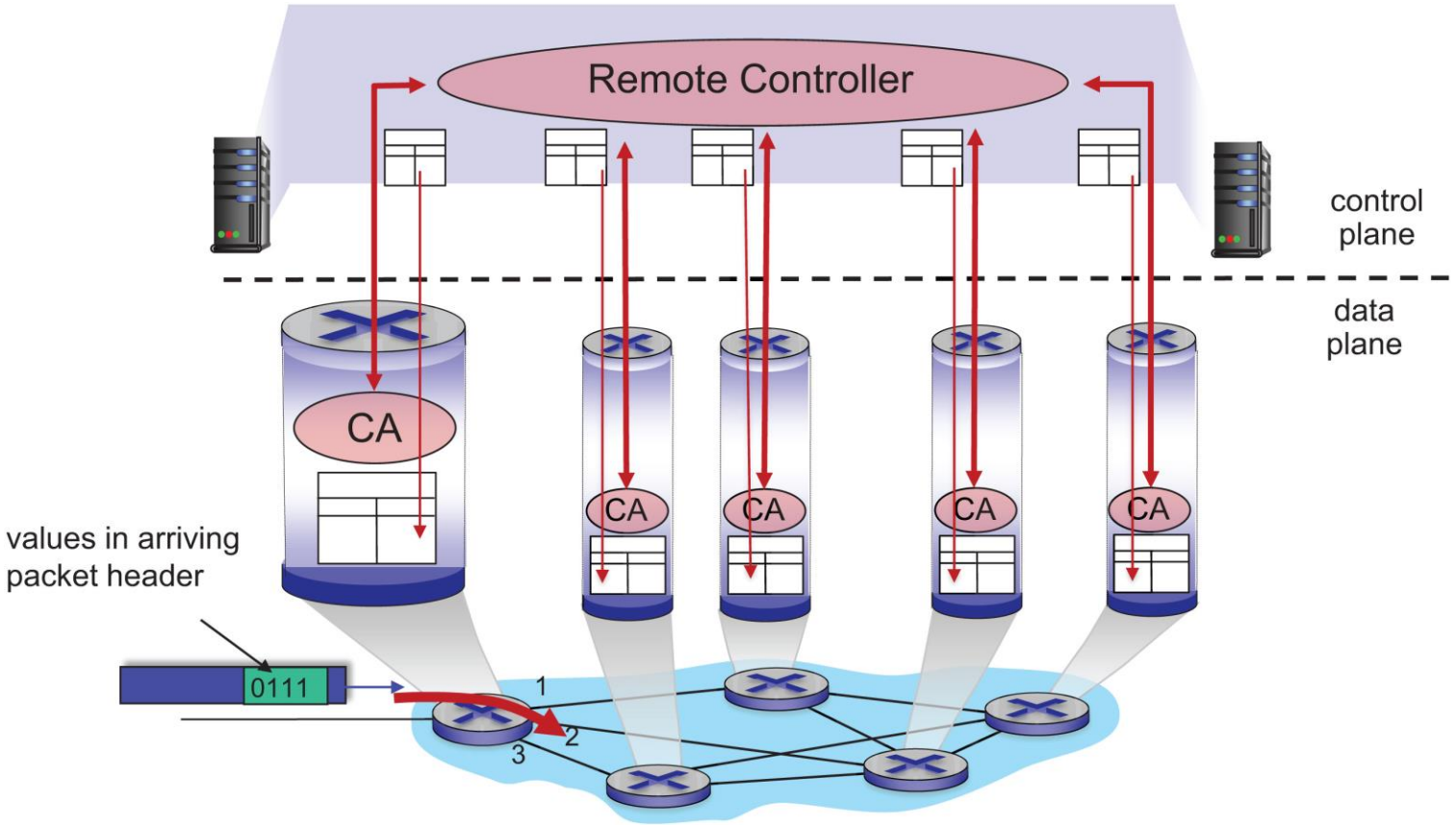
Per-router Control Plane (1 of 2)

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

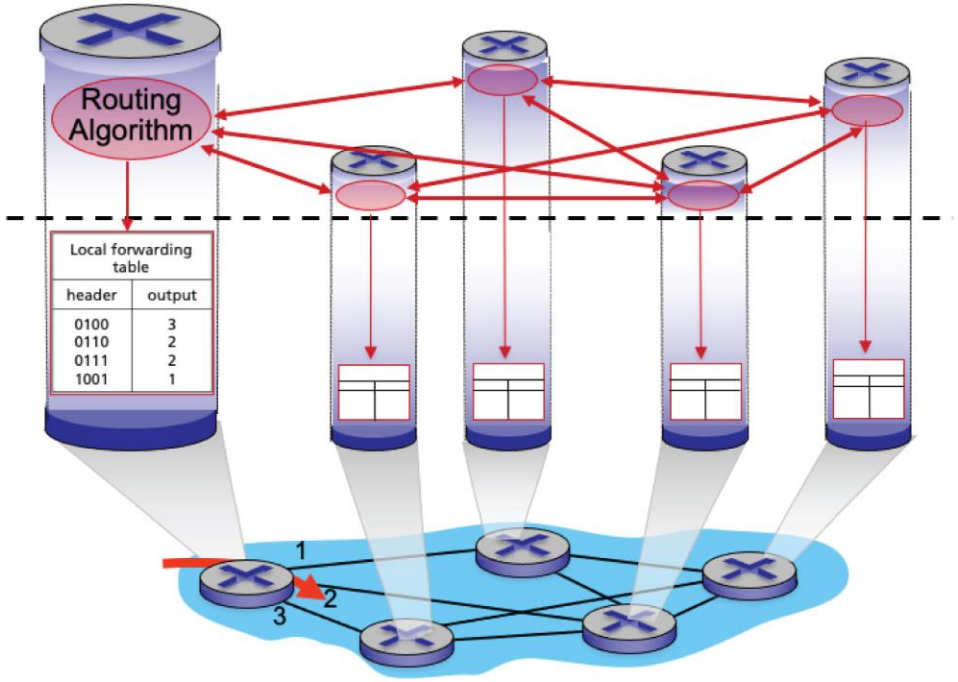


Software-Defined Networking (SDN) Control Plane (1 of 2)

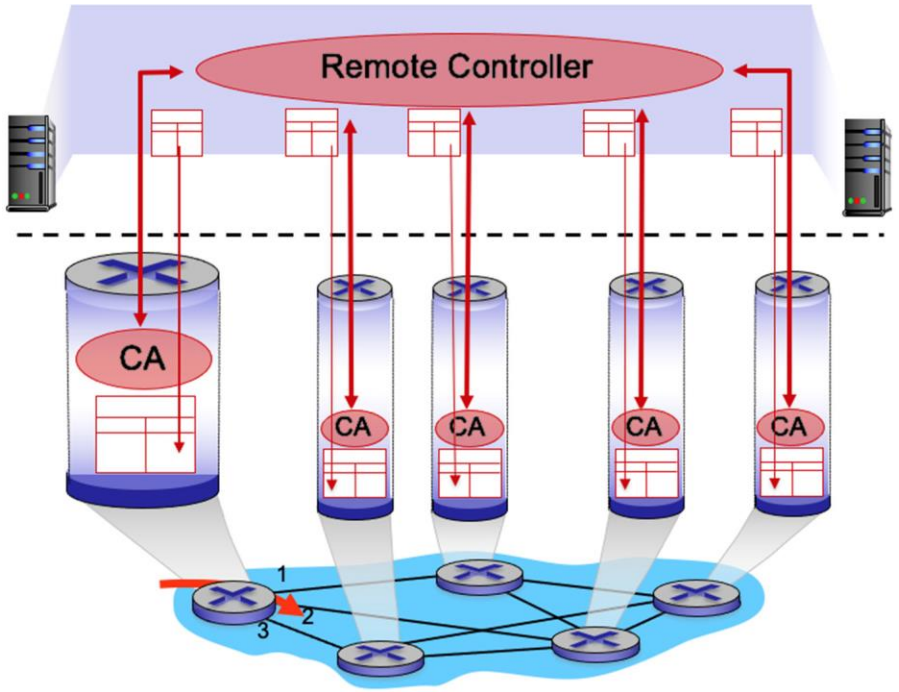
Remote controller computes, installs forwarding tables in routers



Per-router control plane



SDN control plane



Network Layer: "Control Plane" Roadmap (2 of 10)

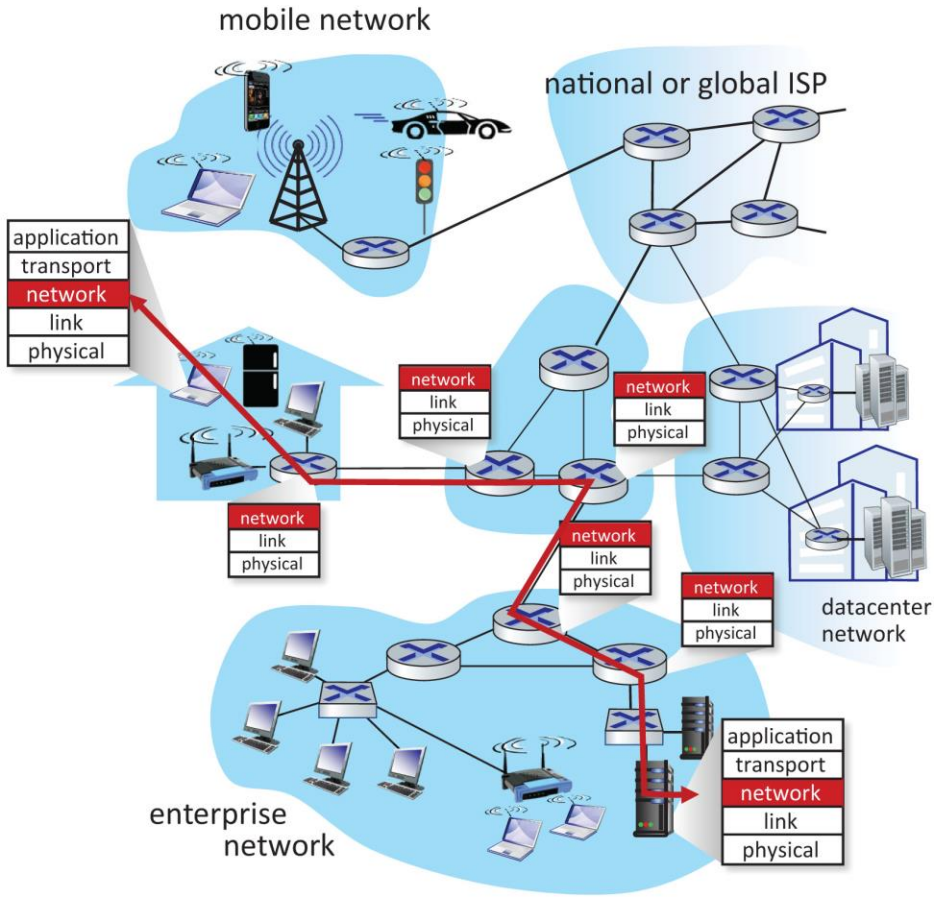
- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP

- Internet Control Message Protocol

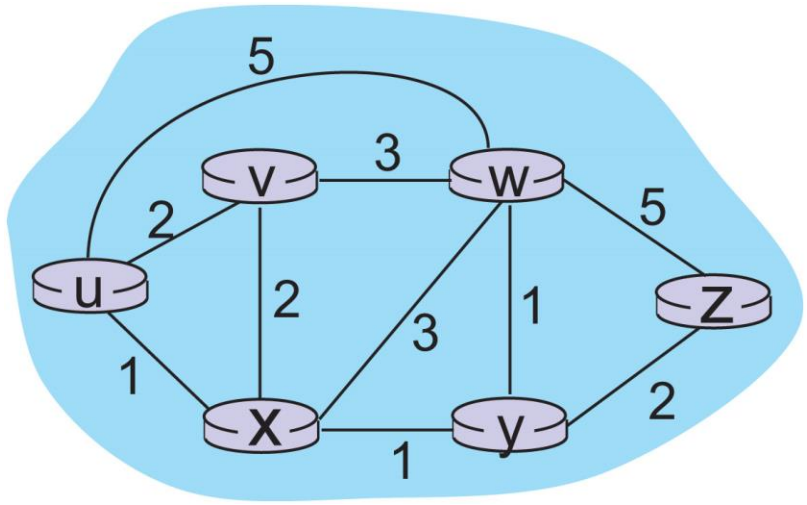
Routing Protocols

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

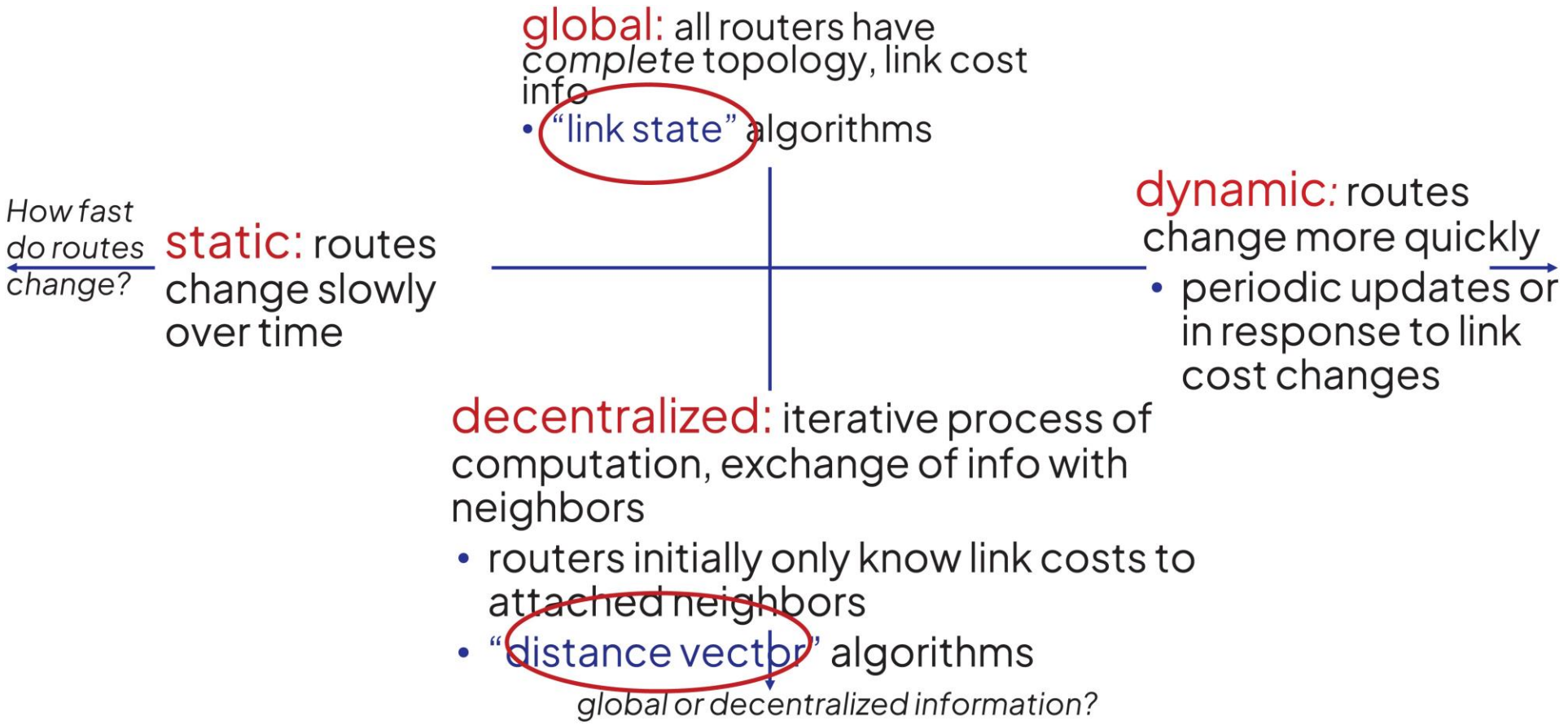
cost defined by network operator: could always be 1, or inversely related to bandwidth, or inversely related to congestion

graph: $G = (N,E)$

N: set of routers $= \{u, v, w, x, y, z\}$

E: set of links $= \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Routing Algorithm Classification



Network Layer: "Control Plane" Roadmap (3 of 10)

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

Dijkstra's Link-State Routing Algorithm (1 of 2)

- centralized: network topology, link costs known to **all** nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives **forwarding table** for that node
- iterative: after *k* iterations, know least cost path to *k* destinations

notation

- $c_{x,y}$: **direct** link cost from node *x* to *y*; $= \infty$ if not direct neighbors
- *D (v)*: **current** estimate of cost of least-cost-path from source to destination *v*
- *p (v)*: predecessor node along path from source to *v*
- *N'*: set of nodes whose least-cost-path **definitively** known

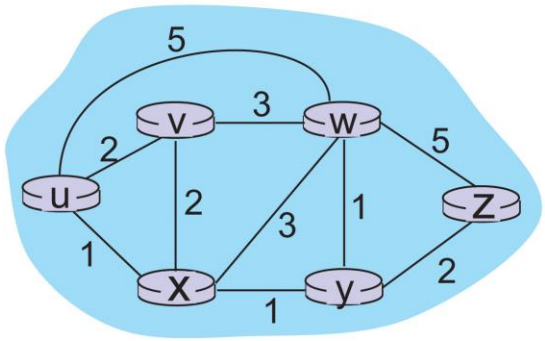
Dijkstra's Link-State Routing Algorithm (2 of 2)

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 (1 of 13)

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						

D(w),p(w)
5,u
4,x
3,y
3,y



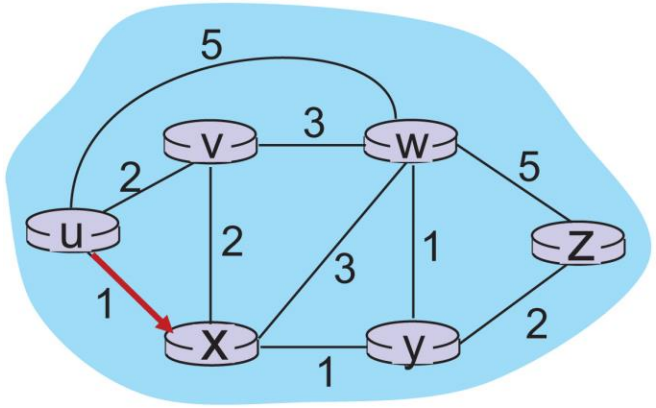
Initialization (step 0):

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

Dijkstra's Algorithm: An Example (2 of 13)

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						

D(w),p(w)
5,u
4,x
3,y
3,y

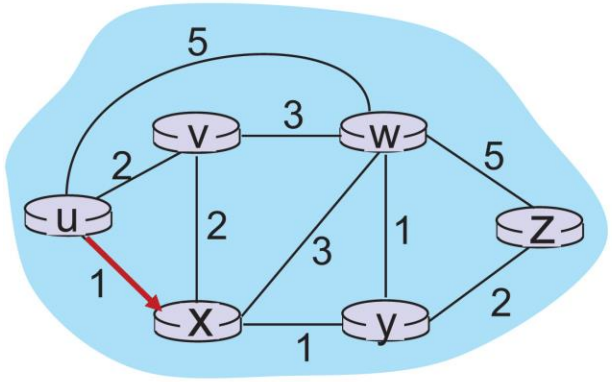



```
8  Loop
9     find a not in N' such that D(a) is a minimum
10    add a to N'
```

Dijkstra's Algorithm: An Example (3 of 13)

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						

D(w),p(w)
5,u
4,x
3,y
3,y



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'* :

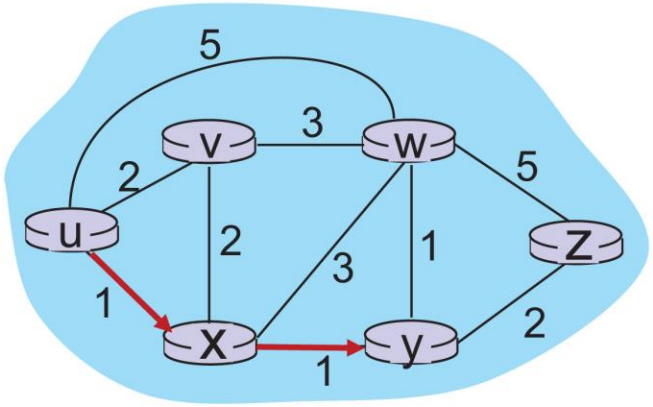
$$D(b) = \min (D(b), D(a) + c_{a,b})$$

$D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$
$D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$ NEW!
$D(y) = \min (D(y), D(x) + c_{x,y}) = \min(\inf, 1+1) = 2$ NEW!

Dijkstra's Algorithm: An Example (4 of 13)

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						

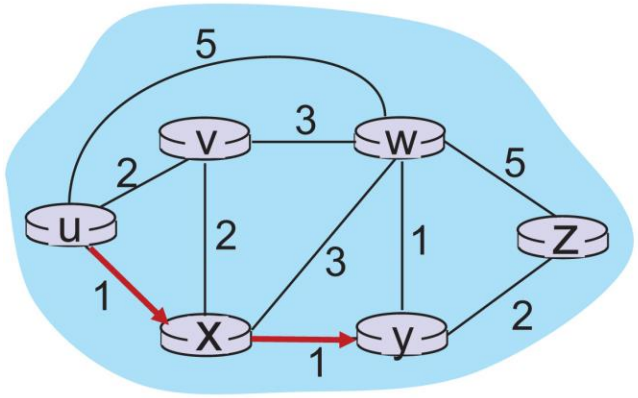
D(w),p(w)
5,u
4,x
3,y
3,y




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

Dijkstra's Algorithm: An Example (5 of 13)

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						



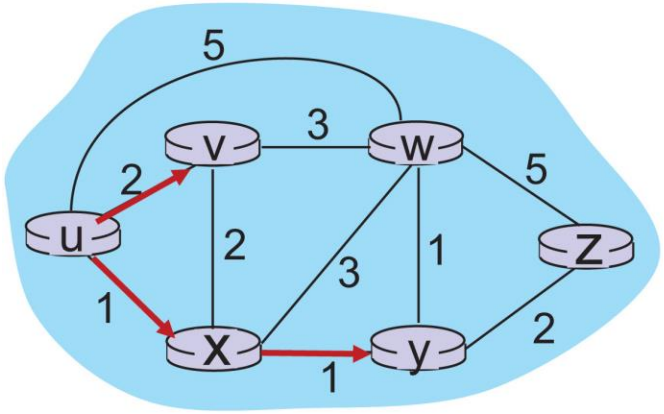

```
8  Loop
9     find a not in N' such that D(a) is a minimum
10    add a to N'
11    update D(b) for all b adjacent to a and not in N' :
         D(b) = min ( D(b), D(a) + c_{a,b} )
```

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

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

Dijkstra's Algorithm: An Example (6 of 13)

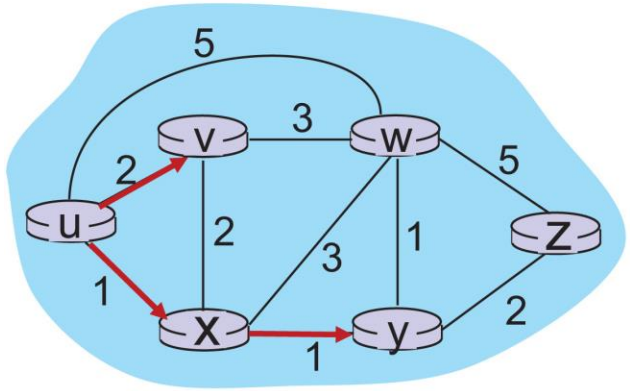
Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						




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

Dijkstra's Algorithm: An Example (7 of 13)

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						
5						



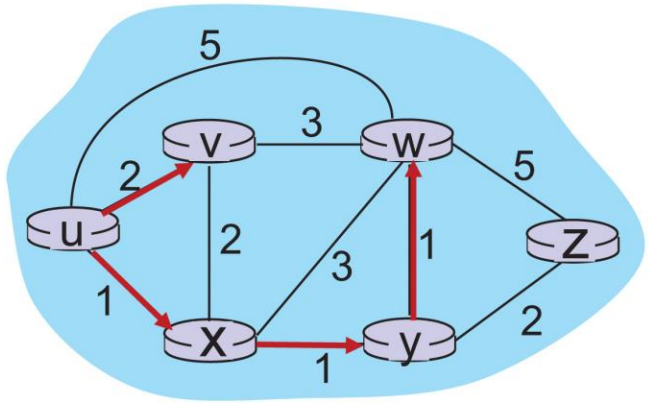
8 *Loop*
9 find *a* not in *N'* such that *D(a)* is a minimum
10 add *a* to *N'*
11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

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

Dijkstra's Algorithm: An Example (8 of 13)

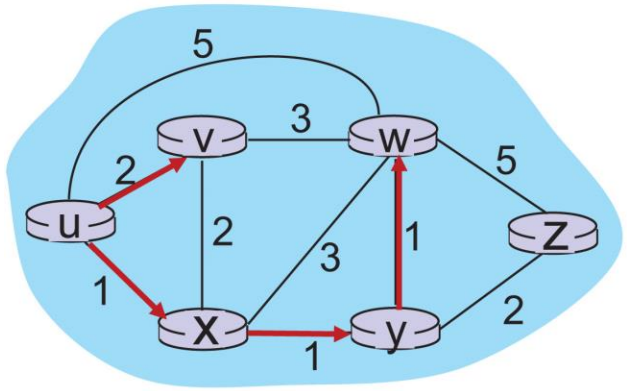
Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5						



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 (9 of 13)

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						



8 *Loop*

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

10 add *a* to *N'*

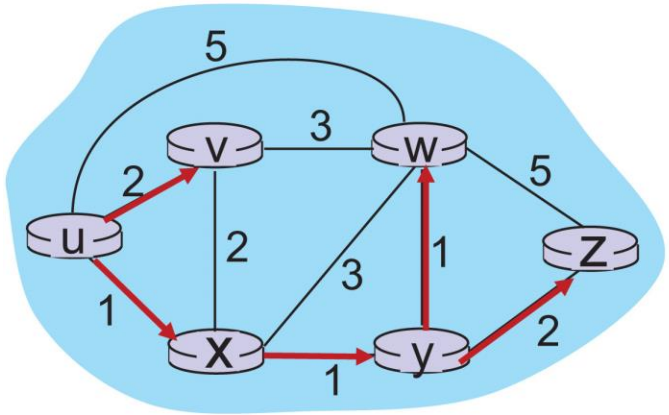
11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

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

Dijkstra's Algorithm: An Example (10 of 13)

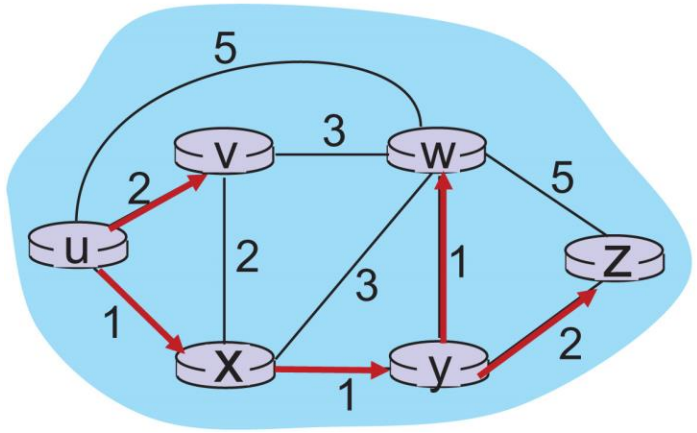
Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



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 (11 of 13)

		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'*

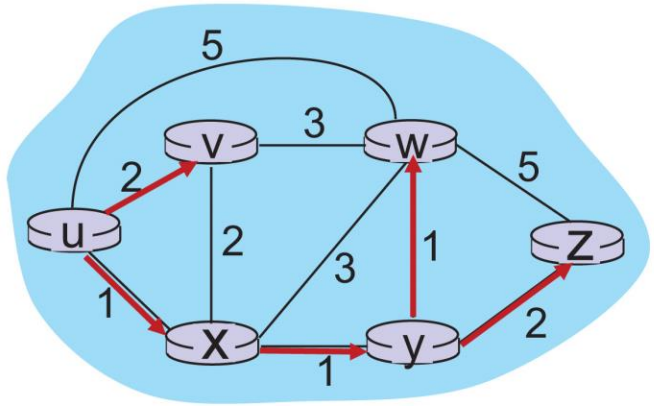
11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

Dijkstra's Algorithm: An Example (12 of 13)

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					

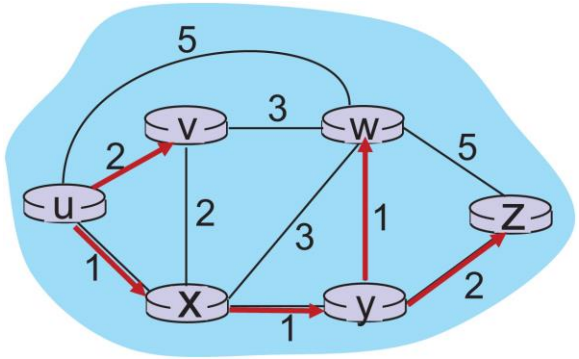
D(w),p(w)
5,u
4,x
3,y
3,y



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

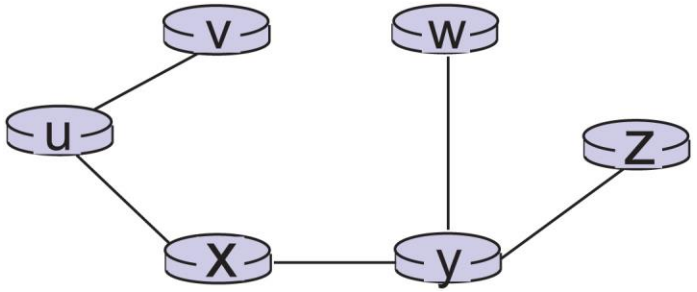
find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :
$D(b) = \min (D(b), D(a) + c_{a,b})$

Dijkstra's Algorithm: An Example (13 of 13)



D(w),p(w)
5,u
4,x
3,y
3,y

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*

Network Layer: "Control Plane" Roadmap (4 of 10)

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

Distance Vector Algorithm (1 of 2)

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:

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

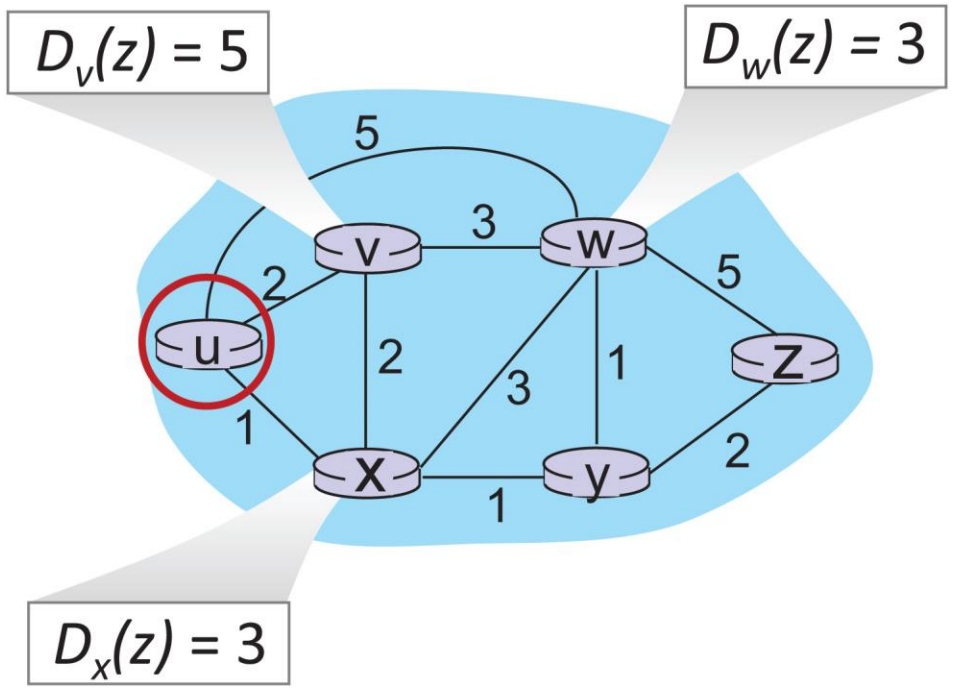
min taken over all neighbors v of x

direct cost of link from x to v

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

Bellman-Ford Example

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



Bellman-Ford equation says:

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

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

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

Distance Vector Algorithm (2 of 2)

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 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

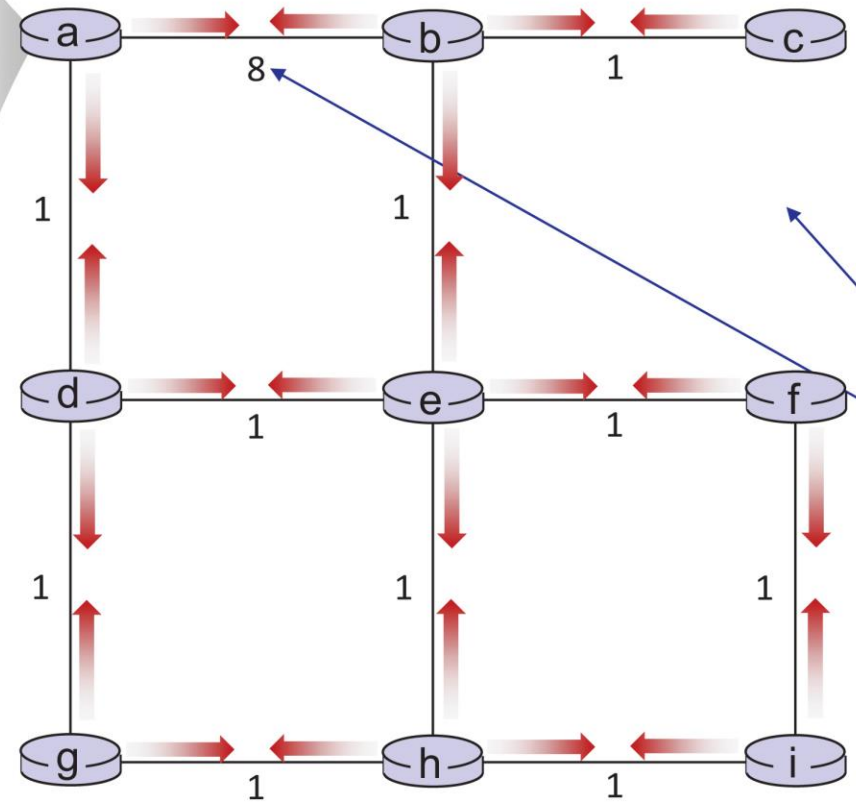
- neighbors then notify their neighbors – **only if necessary**
- no notification received, no actions taken!

Distance Vector: Example

t=0

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

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



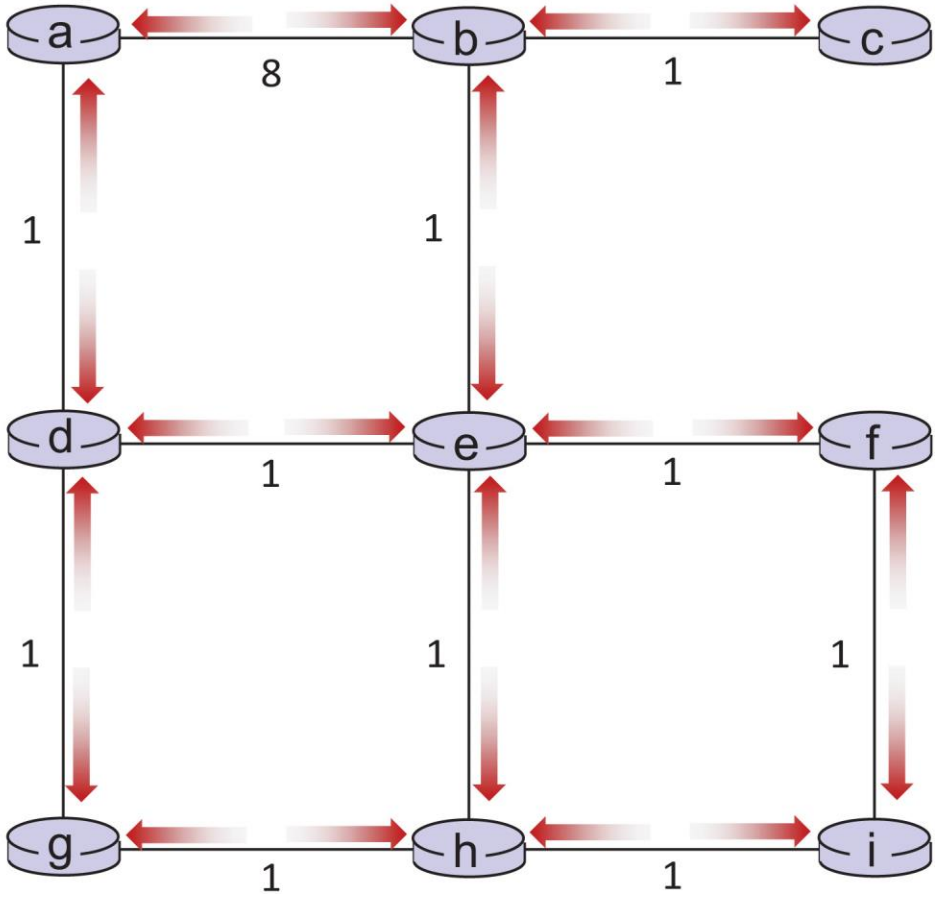
A few asymmetries:
- missing link
- larger cost

Distance Vector Example: Iteration (1 of 7)

t=1

All nodes:

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

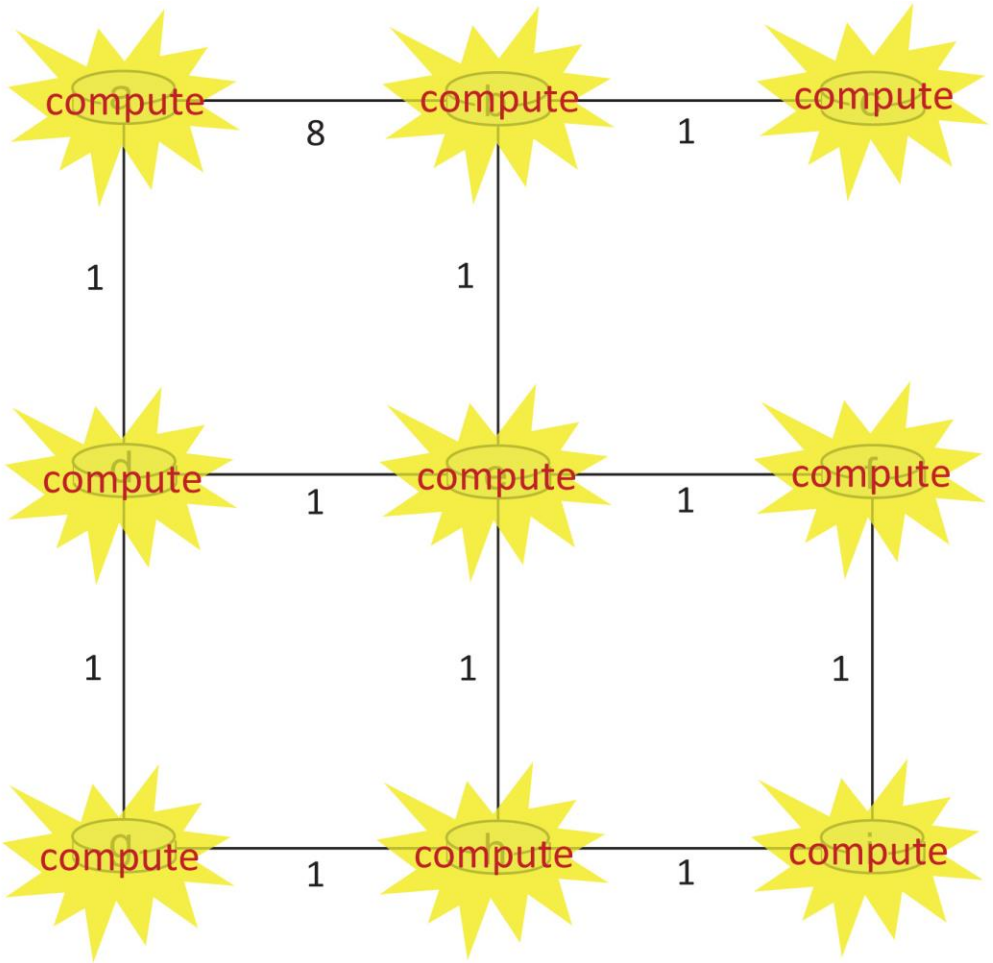


Distance Vector Example: Iteration (2 of 7)

t=1

All nodes:

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

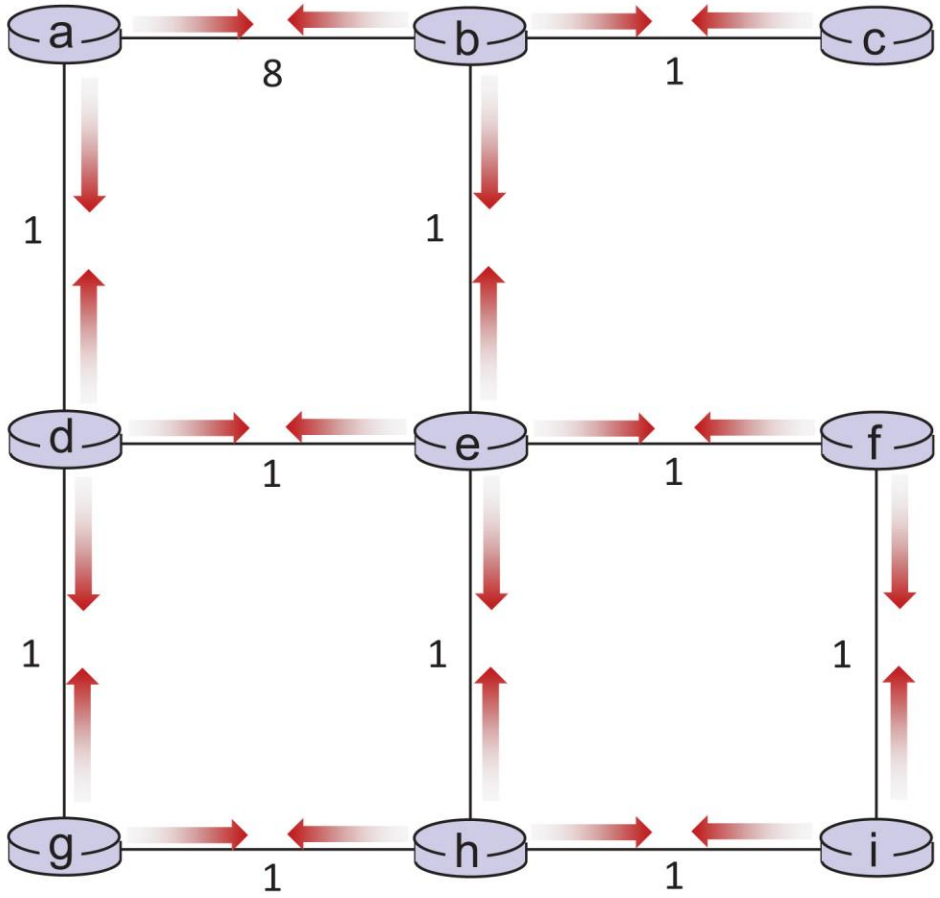


Distance Vector Example: Iteration (3 of 7)

t=1

All nodes:

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

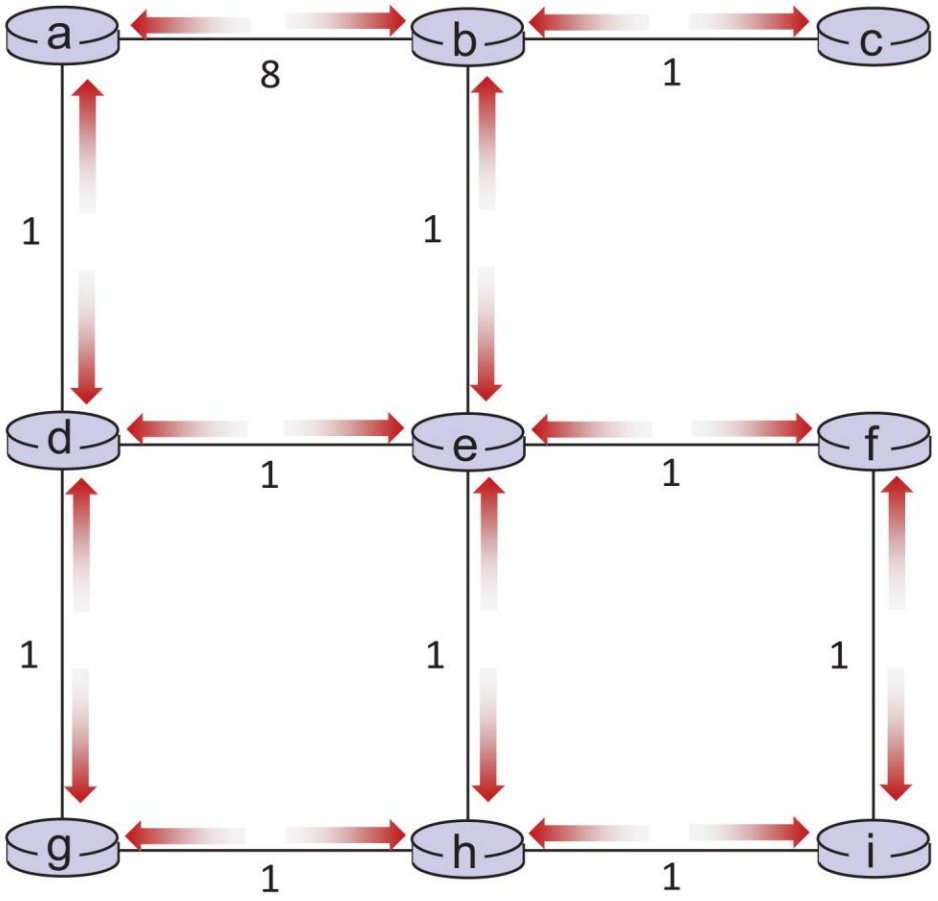


Distance Vector Example: Iteration (4 of 7)

t=2

All nodes:

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

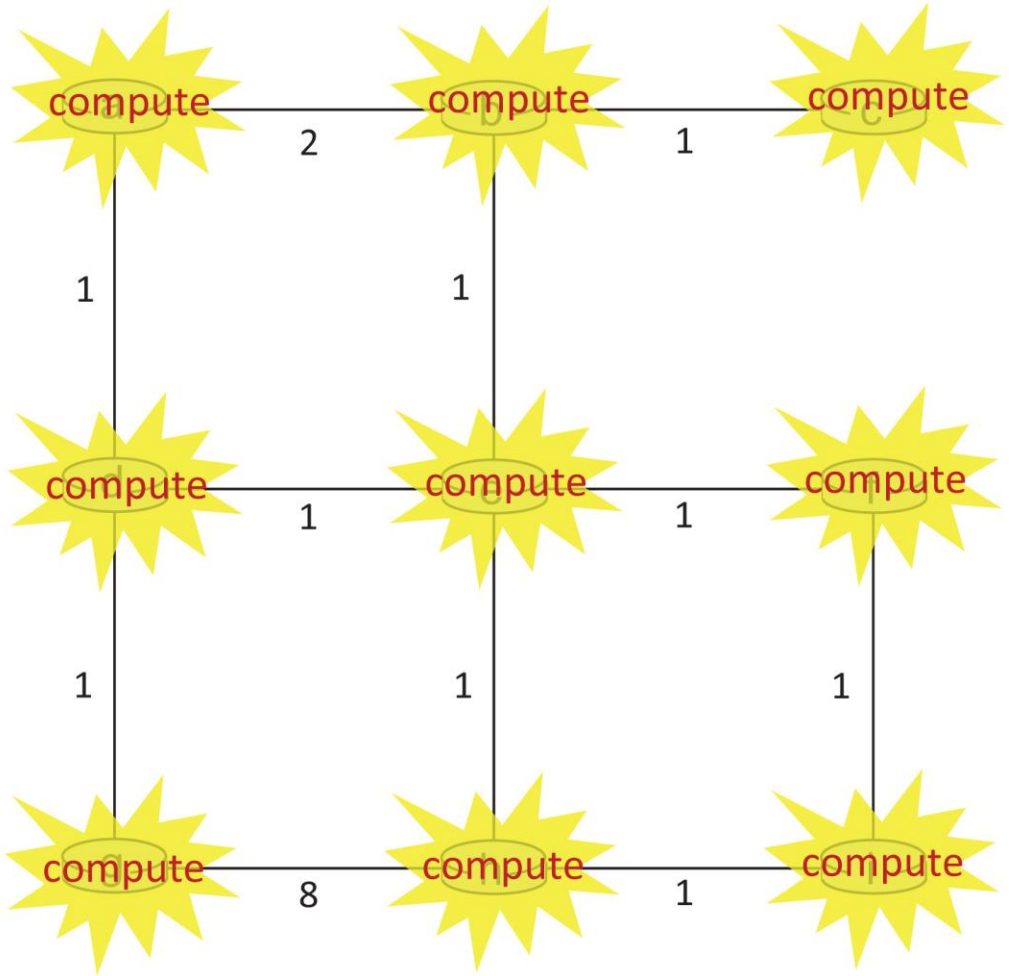


Distance Vector Example: Iteration (5 of 7)

t=2

All nodes:

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

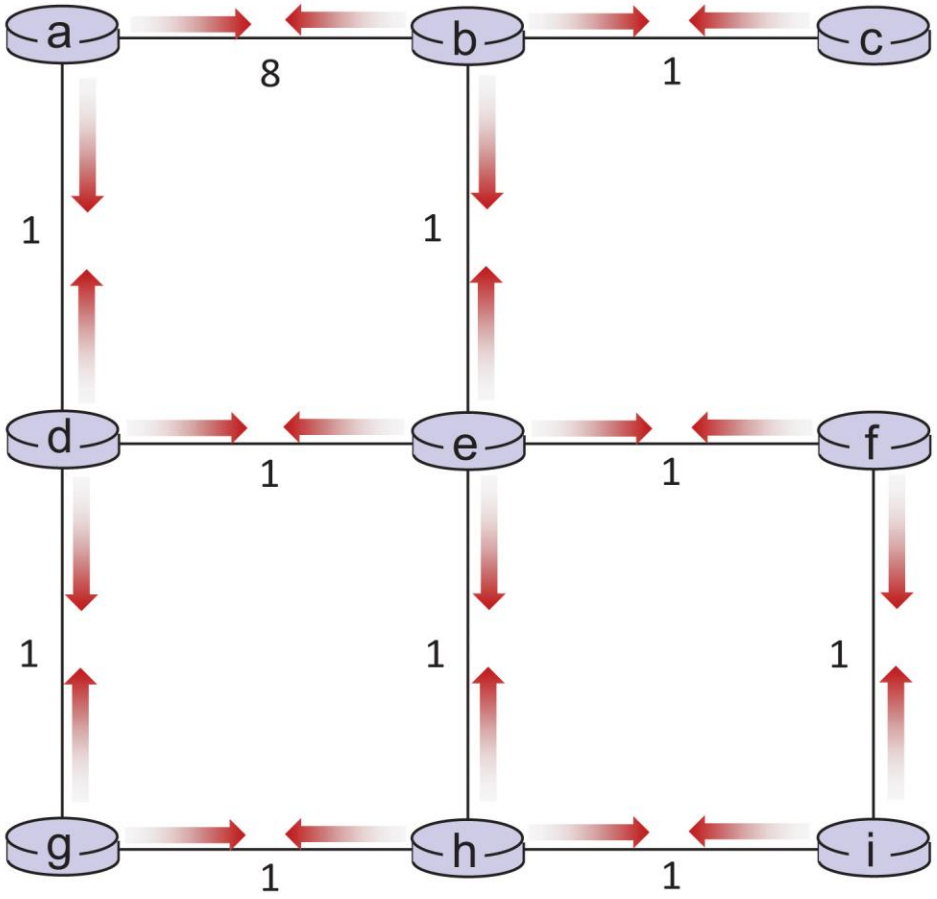


Distance Vector Example: Iteration (6 of 7)

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 (7 of 7)

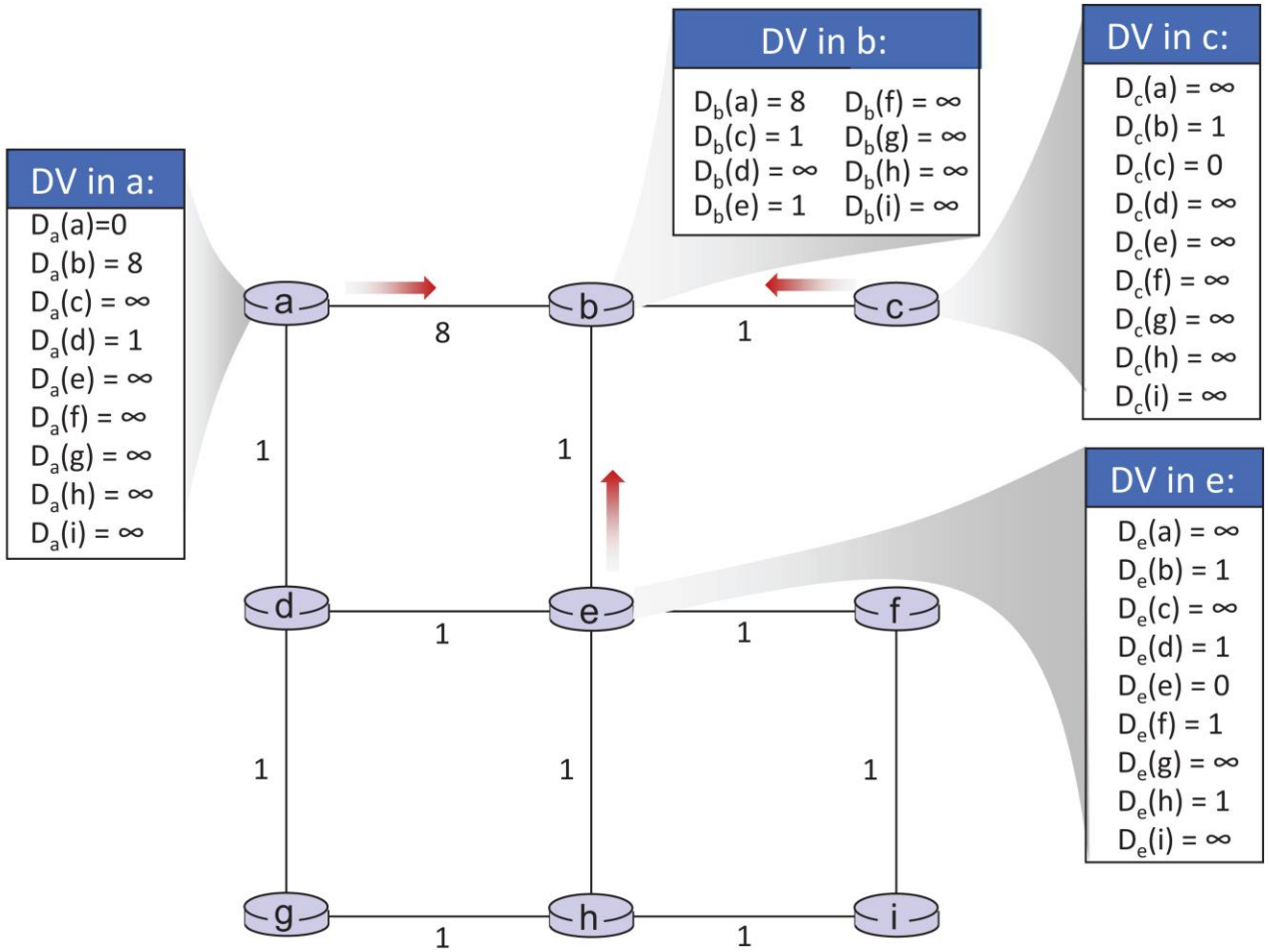
…. and so on

Let's next take a look at the iterative **computations** at nodes

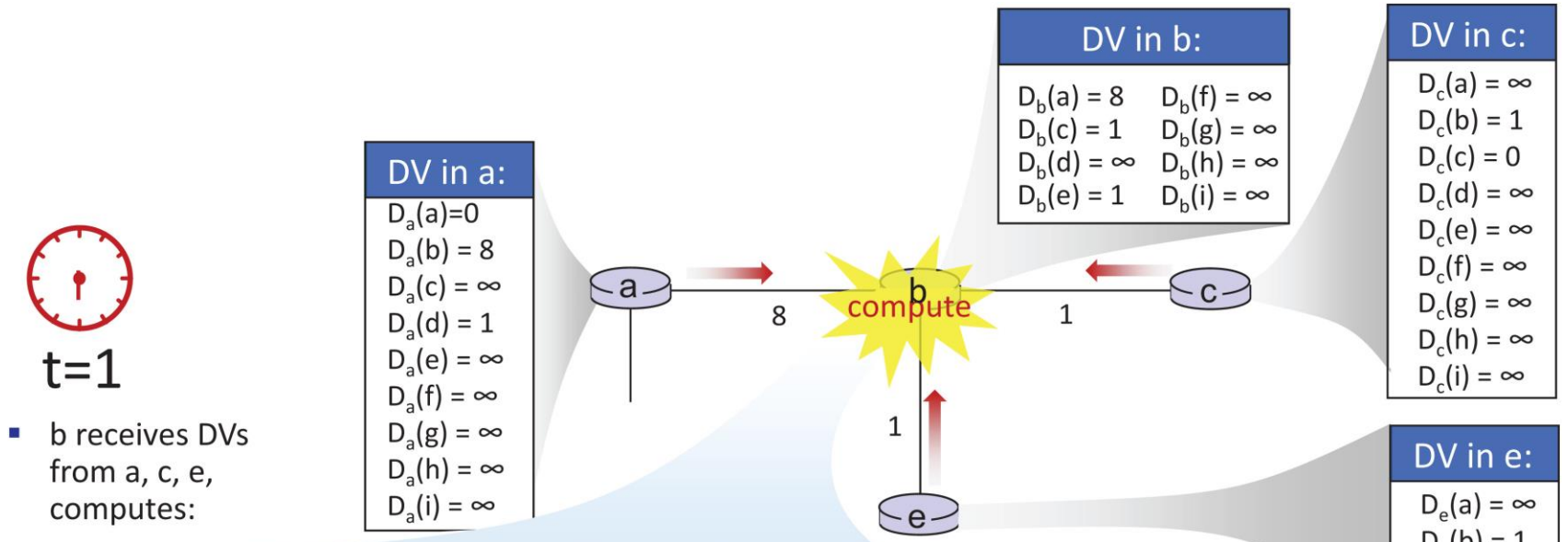
Distance Vector Example: Computation (1 of 5)

t=1

- b receives DVs from a, c, e



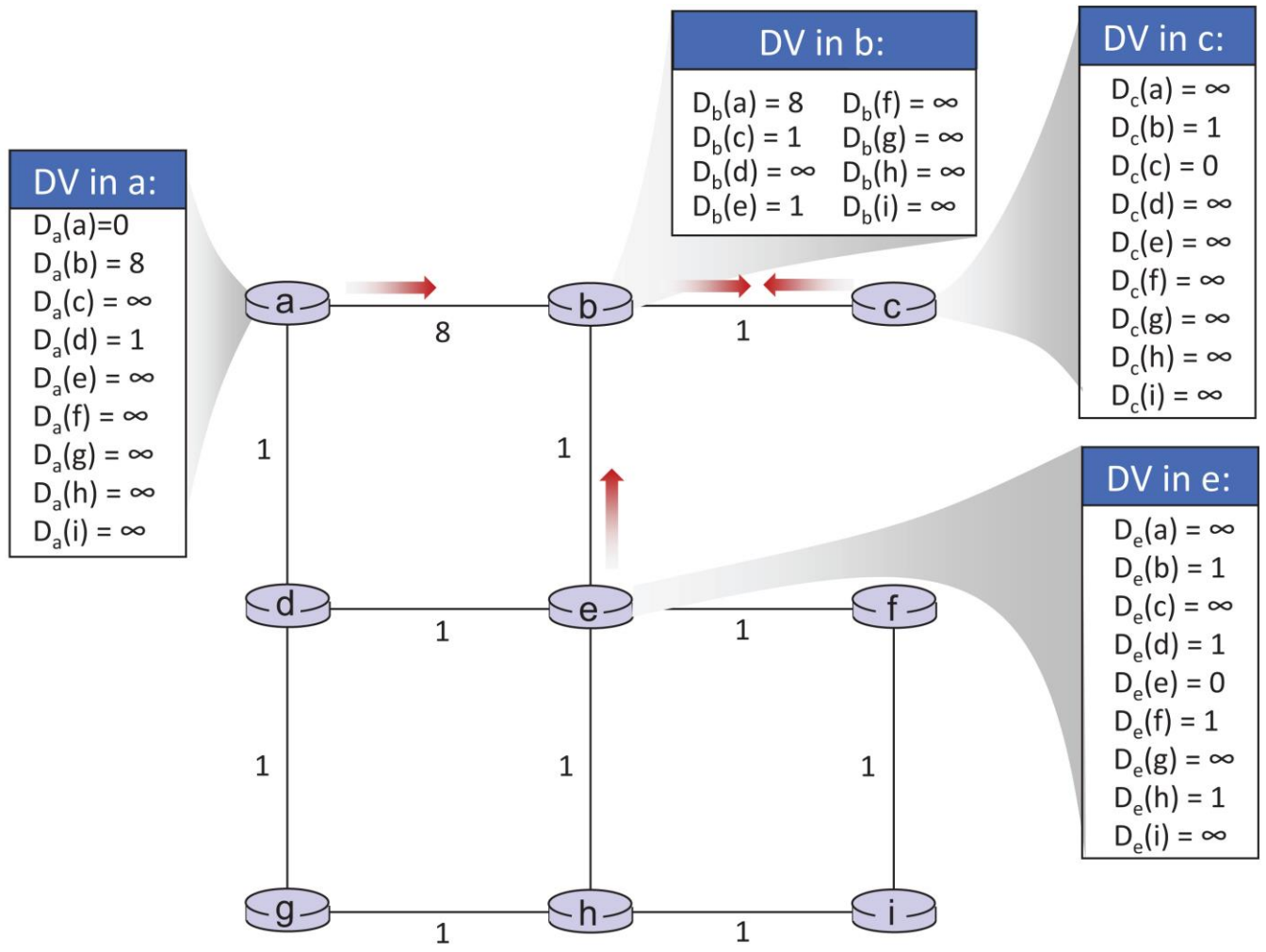
Distance Vector Example: Computation (2 of 5)



Distance Vector Example: Computation (3 of 5)

t=1

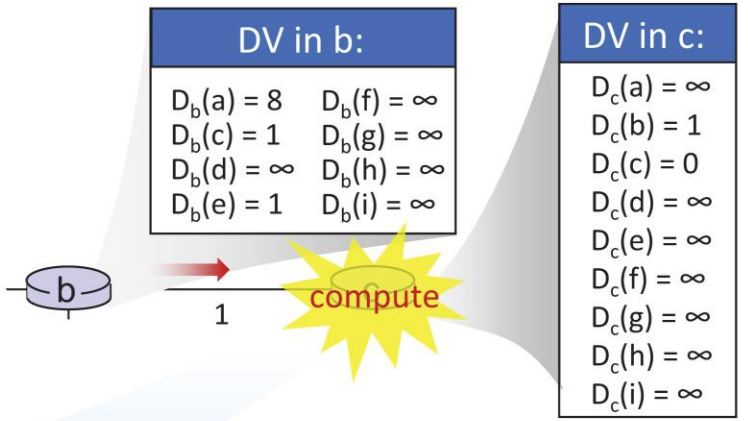
- c receives DVs from b



Distance Vector Example: Computation (4 of 5)

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:

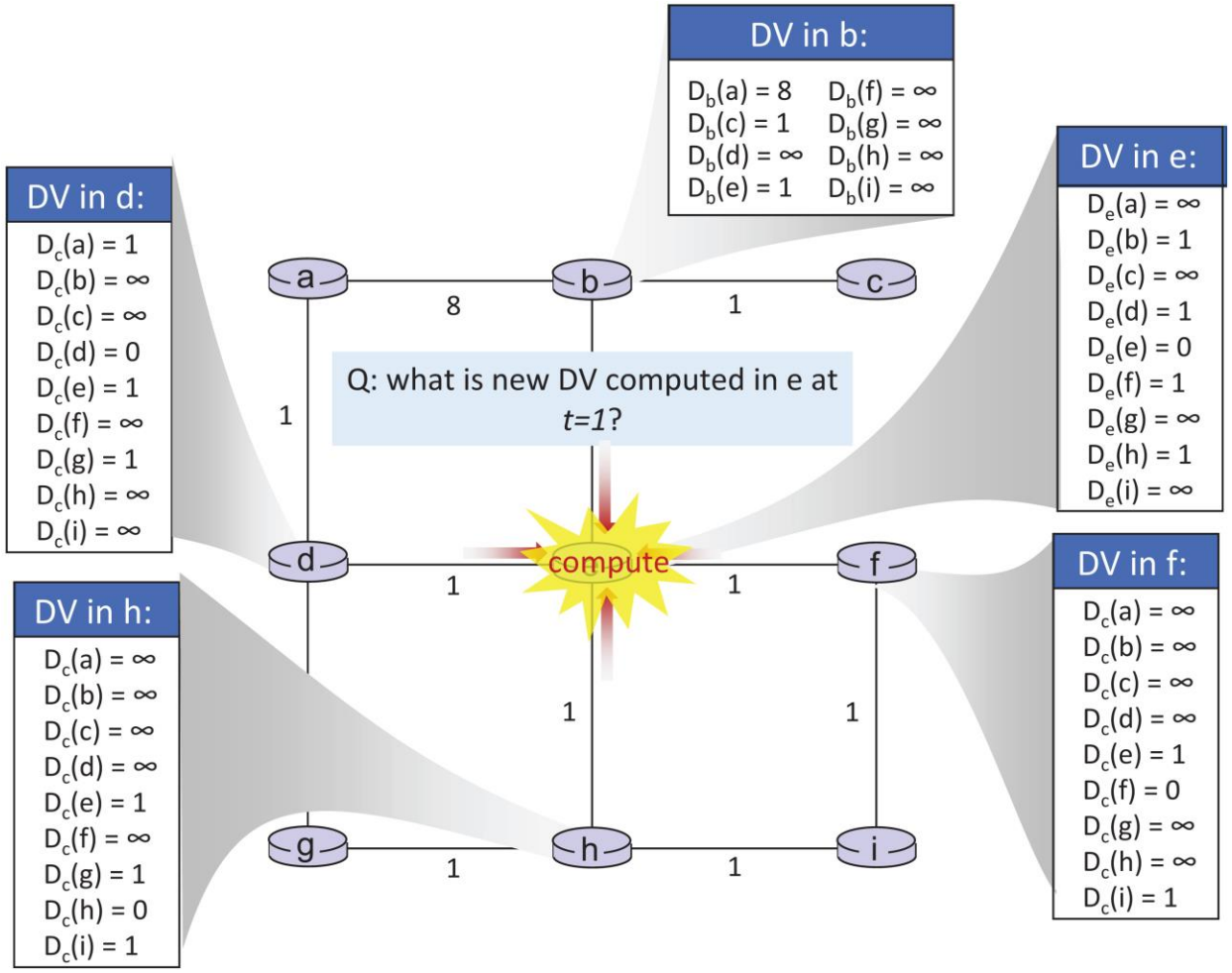
$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

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

Distance Vector Example: Computation (5 of 5)

t=1

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



Distance Vector: State Information Diffusion

Iterative communication, computation steps diffuses information through network:

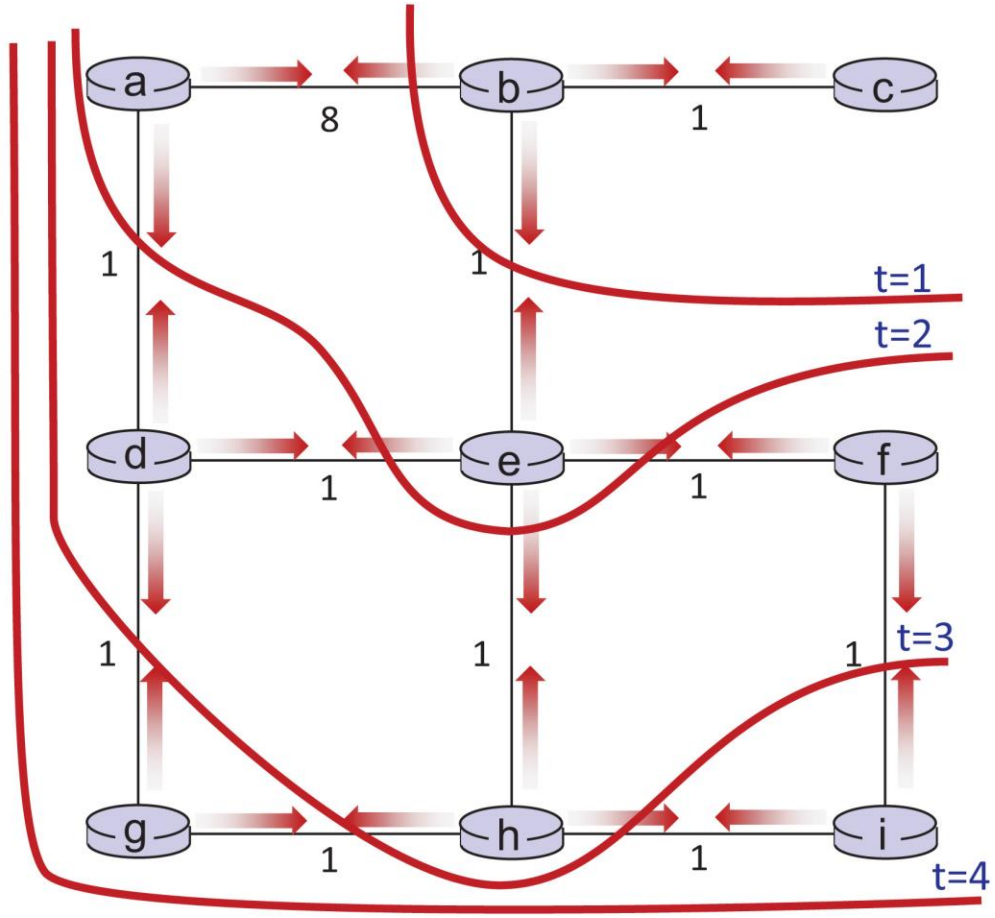
t=0 c's state at t=0 is at c only

t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b

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

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

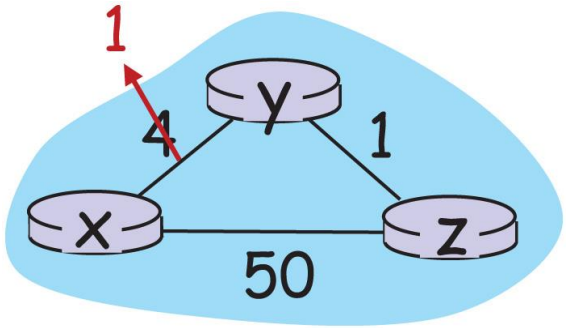
t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at g, i



Distance Vector: Link Cost Changes (1 of 2)

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*.

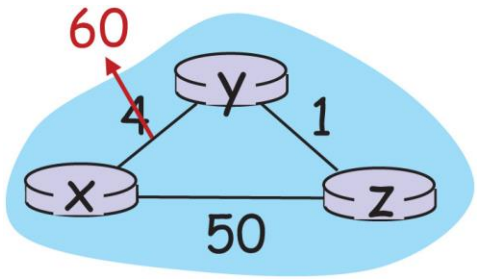
Distance Vector: Link Cost Changes (2 of 2)

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!**



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

Network Layer: "Control Plane" Roadmap (5 of 10)

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- Internet Control Message Protocol

Making Routing Scalable

our routing study thus far - idealized

- all routers identical
- network "flat"

… not true in practice

scale: billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy:

- Internet: a network of networks
- each network admin may want to control routing in its own network

Internet Approach to Scalable Routing

aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")

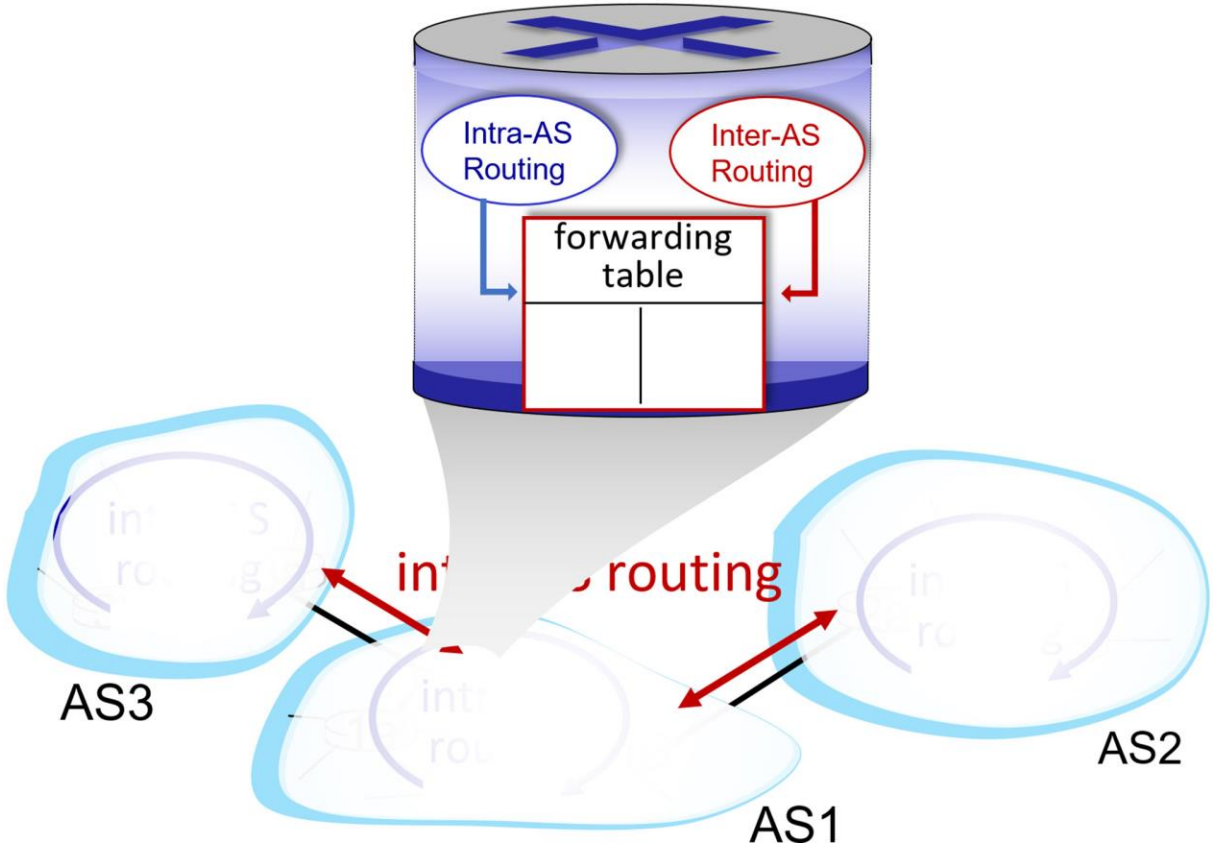
intra-AS (aka "intra-domain"): routing among routers **within same AS ("network")**

- all routers in AS must run same intra-domain protocol
- routers in different AS can run different intra-domain routing protocols
- gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

inter-AS (aka "inter-domain"): routing **among** AS'es

- gateways perform inter-domain routing (as well as intra-domain routing)

Interconnected ASes (1 of 2)



forwarding table configured by intra- and inter-AS routing algorithms

- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

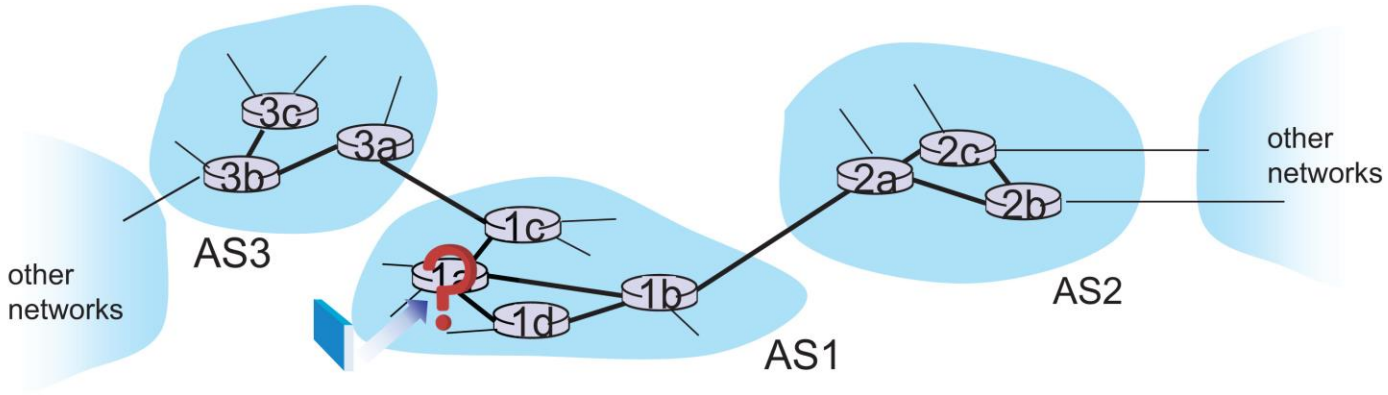
Inter-AS Routing: a Role in Intradomain Forwarding

- suppose router in AS1 receives datagram destined outside of AS1:

  - router should forward packet to gateway router in AS1, but which one?

AS1 inter-domain routing must:

1. learn which destinations reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1



Intra-AS Routing: Routing Within an AS

most common intra-AS routing protocols:

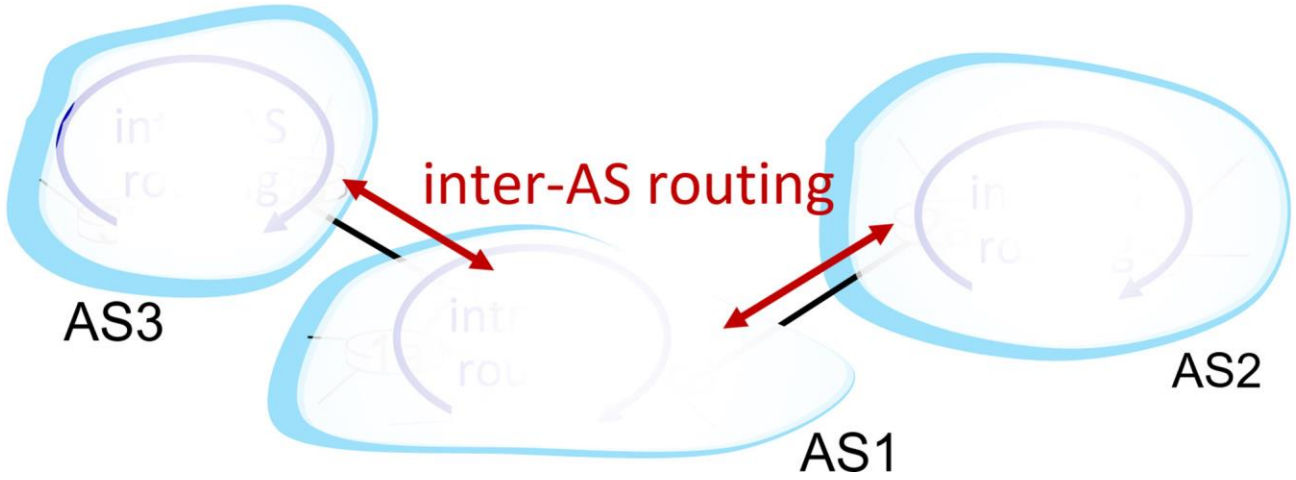
- RIP: Routing Information Protocol [RFC 1723]
  - classic DV: DVs exchanged every 30 secs
  - no longer widely used
- EIGRP: Enhanced Interior Gateway Routing Protocol
  - DV based
  - formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
- OSPF: Open Shortest Path First [RFC 2328]
  - link-state routing
  - IS-IS protocol (ISO standard, not RFC standard) essentially same as OSPF

Network Layer: "Control Plane" Roadmap (6 of 10)

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

Interconnected ASes (2 of 2)



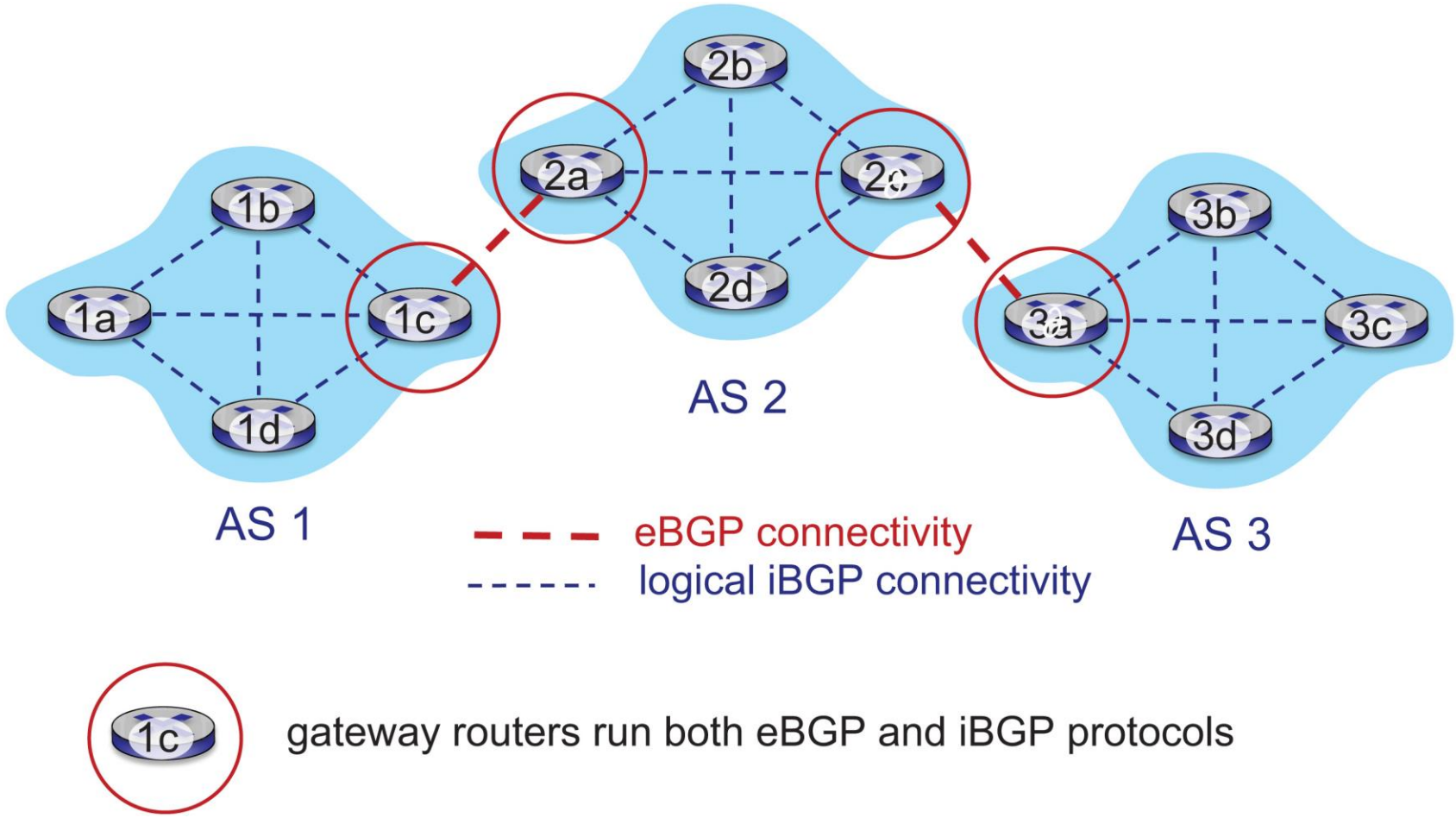
intra-AS (aka "intra-domain"): routing among routers **within same AS ("network")**

inter-AS (aka "inter-domain"): routing **among** AS'es

Internet Inter-AS Routing: BGP

- BGP (Border Gateway Protocol): **the** de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet: **"I am here, here is who I can reach, and how"**
- BGP provides each AS a means to:
  - obtain destination network reachability info from neighboring ASes (eBGP)
  - determine routes to other networks based on reachability information and **policy**
  - propagate reachability information to all AS-internal routers (iBGP)
  - advertise (to neighboring networks) destination reachability info

eBGP, iBGP Connections



Why Different Intra-, Inter-AS Routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its network
- intra-AS: single admin, so policy less of an issue

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy dominates over performance

Network Layer: "Control Plane" Roadmap (8 of 10)

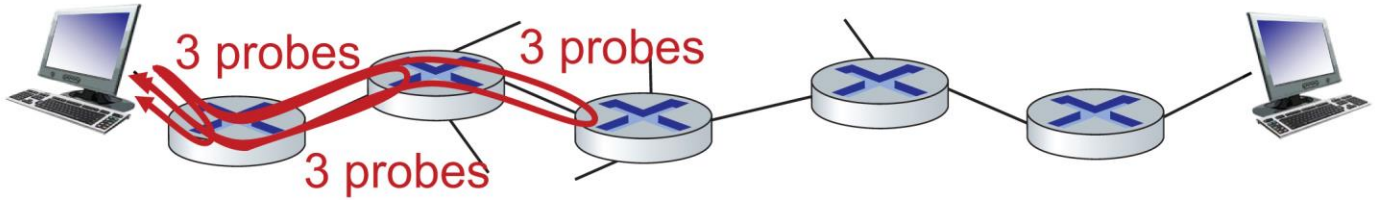
- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- Internet Control Message Protocol

ICMP: Internet Control Message Protocol

- used by hosts and routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP messages carried in IP datagrams
- **ICMP message**: type, code plus first 8 bytes of IP datagram causing error

Type	Code	Description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP



- source sends sets of UDP segments to destination
  - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in *n*th set arrives to nth router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops