



PES
UNIVERSITY

CELEBRATING 50 YEARS

COMPUTER NETWORKS

TEAM NETWORKS

Department of Computer Science and Engineering

Unit – 4 Network Layer and Internet Protocol

4.5 Introduction to Network Layer Protocols

- DHCP
- ICMP

4.6 IPv6 Addressing

4.7 Introduction to Routing Algorithms

Unit – 4 Network Layer and Internet Protocol

4.5 Introduction to Network Layer Protocols

- DHCP
- ICMP

4.6 IPv6 Addressing

4.7 Introduction to Routing Algorithms

That's actually **two** questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

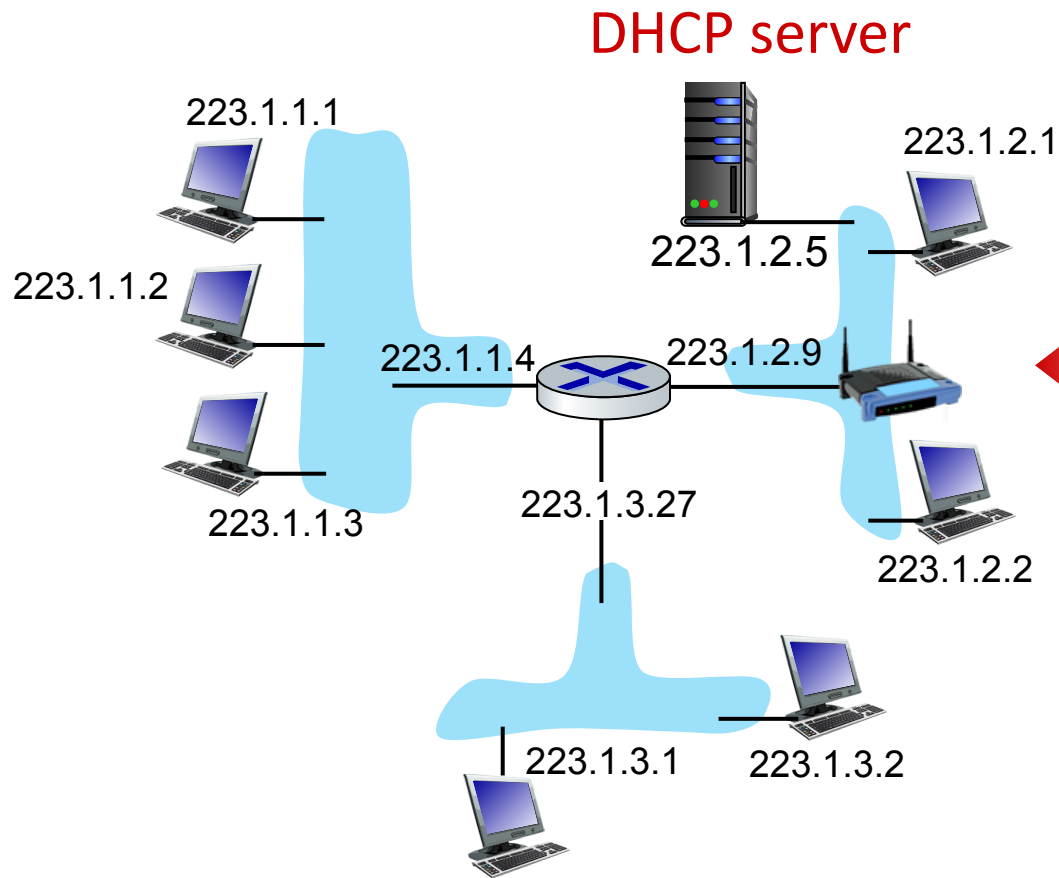
- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- **DHCP**: Dynamic Host Configuration Protocol: dynamically get address from as server
 - “plug-and-play”

goal: host *dynamically* obtains IP address from network server when it “joins” network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

- host broadcasts **DHCP discover** msg [optional]
- DHCP server responds with **DHCP offer** msg [optional]
- host requests IP address: **DHCP request** msg
- DHCP server sends address: **DHCP ack** msg

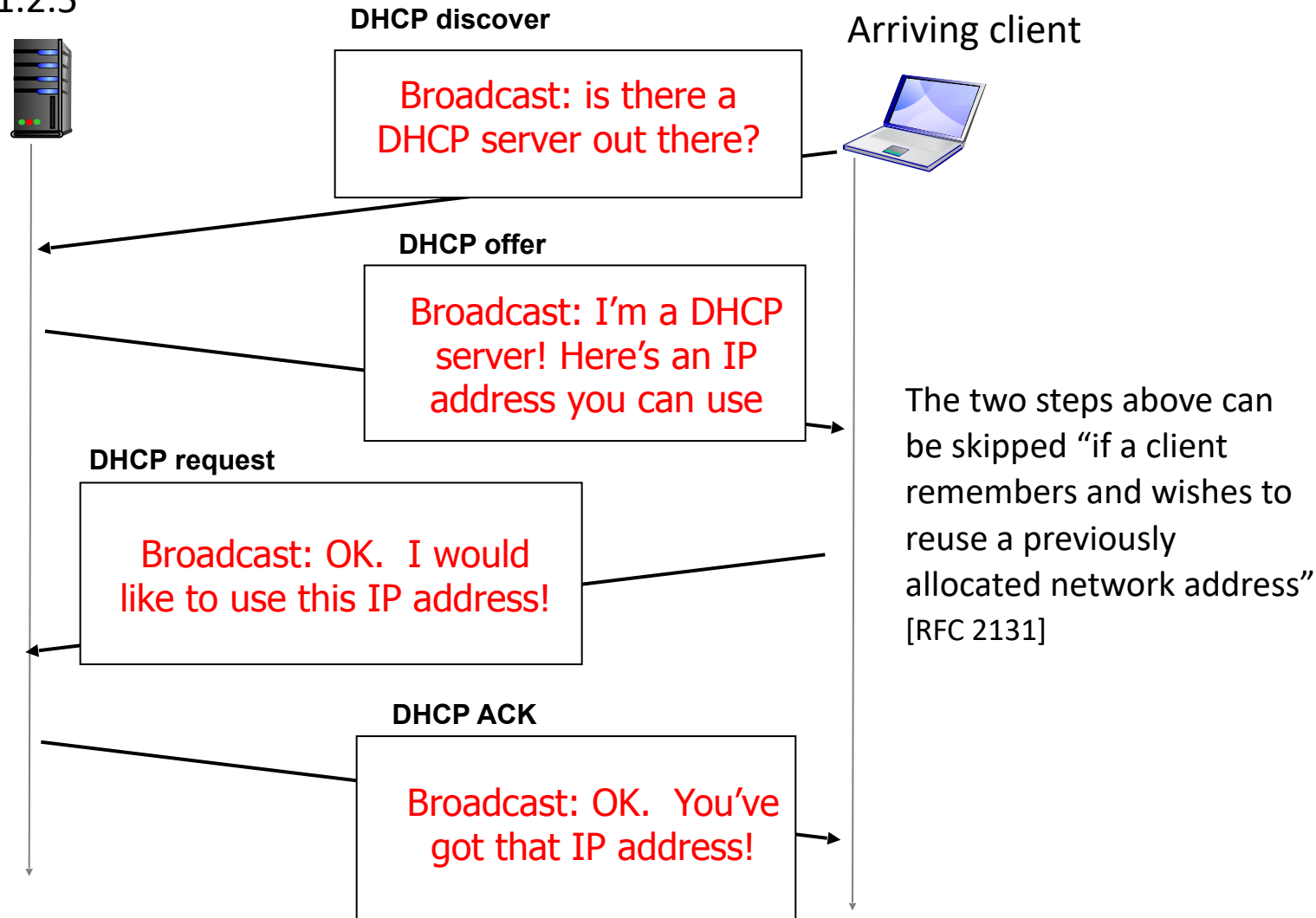


Typically, DHCP server will be co-located in router, serving all subnets to which router is attached



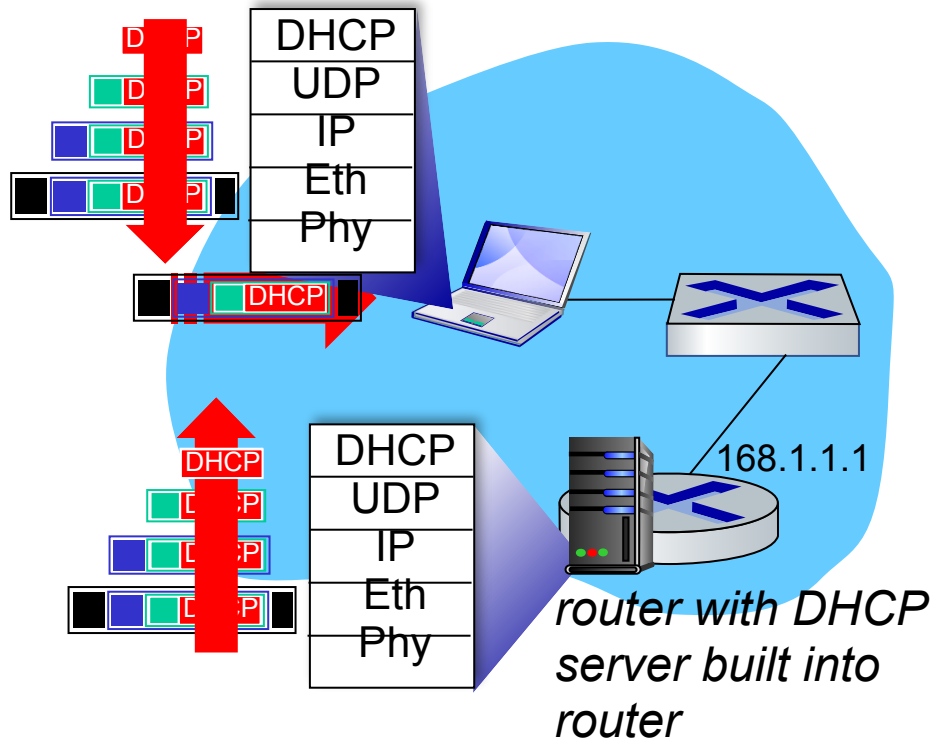
arriving **DHCP client** needs address in this network

DHCP server: 223.1.2.5

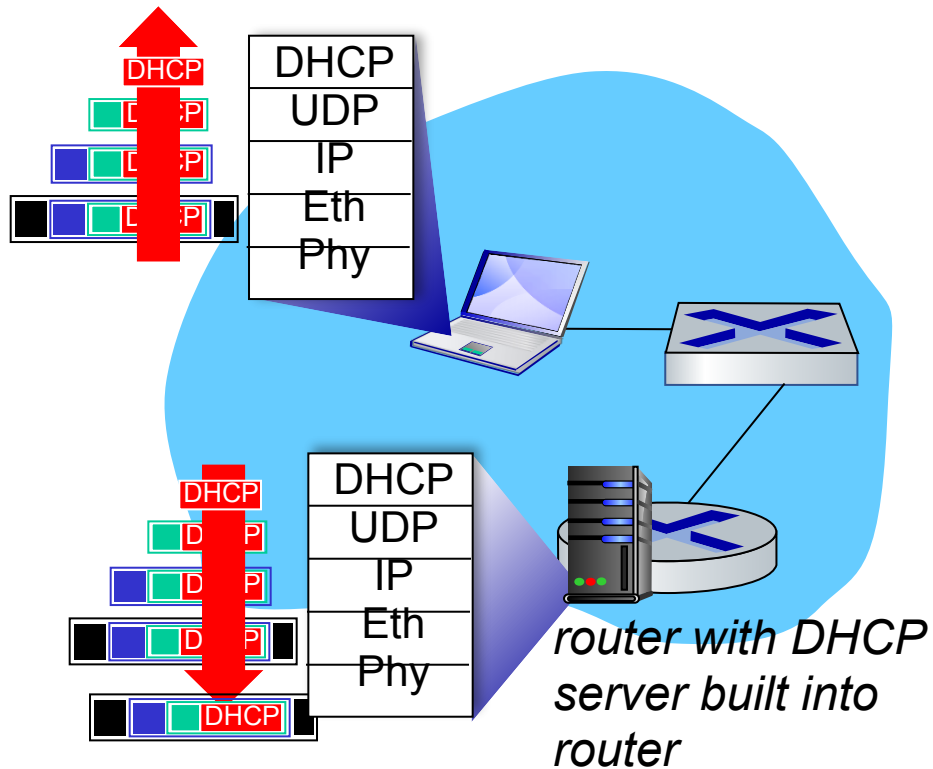


DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)



- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers <http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu, ...) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

Unit – 4 Network Layer and Internet Protocol

4.5 Introduction to Network Layer Protocols

- DHCP
- ICMP

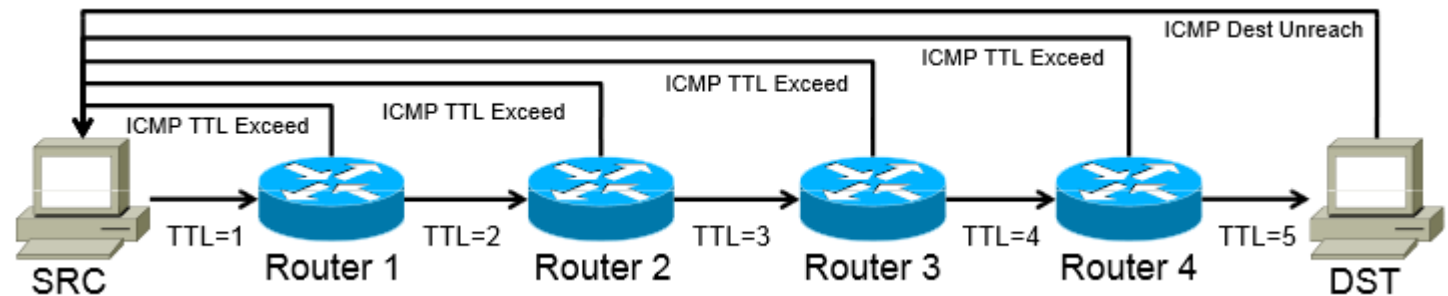
4.6 IPv6 Addressing

4.7 Introduction to Routing Algorithms

- Used by hosts and routers to communicate network-layer information.
- Typical use of ICMP is for error reporting.
- Eg: “Destination network unreachable”
- ICMP messages – carried as IP payloads.
- ICMP is often considered part of IP, but architecturally it lies just above IP.
- ICMP messages have a type and a code field.

<u>Type</u>	<u>Code</u>	<u>description</u>
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

- Perform – Network diagnostics
- Utilities traceroute and ping – use ICMP.
- Other examples of ICMP messages:
 - Source quench message (congestion)
 - Parameter problem (bad IP header)
 - Time exceeded message (TTL)
 - Destination unreachable
 - Redirection message



Unit – 4 Network Layer and Internet Protocol

4.5 Introduction to Network Layer Protocols

- DHCP
- ICMP

4.6 IPv6 Addressing

4.7 Introduction to Routing Algorithms

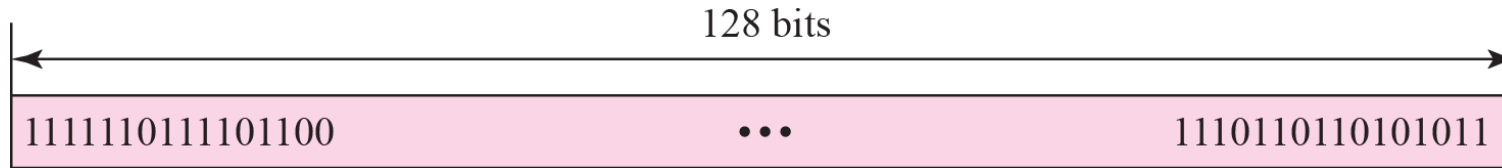
- 128 bits address
- 340 undecillion address (340 + 33 digits)
- IPv6 address space is 2^{96} times of the IPv4 address.
- Made up of eight 16-bit blocks (octet)
- Separated by ‘:’
- Hexadecimal, each hexadecimal character – 4 bits

340,282,366,920,938,463,374,607,431,768,211,456

Rules:

- Discard leading Zero(es)
- If two or more blocks contain consecutive zeroes, omit them all and replace with double colon sign ::

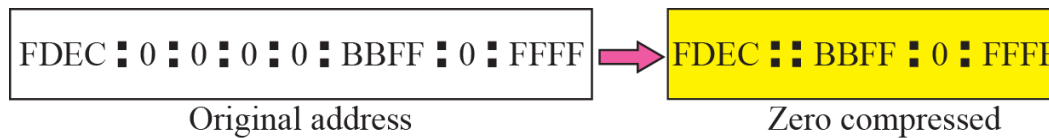
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F



Colon hexadecimal notation

FDEC ■ BA98 ■ 7654 ■ 3210 ■ ADBF ■ BBFF ■ 2922 ■ FFFF

Zero compression



CIDR address



Rule 1

2001:0000:3238:DFE1:0063:0000:0000:FEFB

2001:0000:3238:DFE1:63:0000:0000:FEFB

Rule 2

2001:0000:3238:DFE1:0063:0000:0000:FEFB

2001:0000:3238:DFE1:63::FEFB
2001:0:3238:DFE1:63::FEFB

Show the unabbreviated colon hex notation for the following IPv6 addresses:

- a. An address with 64 0s followed by 64 1s.
- b. An address with 128 0s.
- c. An address with 128 1s.
- d. An address with 128 alternative 1s and 0s.

Solution

- a. 0000:0000:0000:0000:FFFF:FFFF:FFFF:FFFF
- b. 0000:0000:0000:0000:0000:0000:0000:0000
- c. FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
- d. AAAA:AAAA:AAAA:AAAA:AAAA:AAAA:AAAA:AAAA



The following shows the zero contraction version of addresses in previous example.

- a. :: FFFF:FFFF:FFFF:FFFF
- b. ::
- c. FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
- d. AAAA:AAAA:AAAA:AAAA:AAAA:AAAA:AAAA:AAAA



Show abbreviations for the following addresses:

- a. 0000:0000:FFFF:0000:0000:0000:0000:0000
- b. 1234:2346:0000:0000:0000:0000:0000:1111
- c. 0000:0001:0000:0000:0000:0000:1200:1000
- d. 0000:0000:0000:0000:0000:FFFF:24.123.12.6

Solution

- a. 0:0:FFFF::
- b. 1234:2346::1111
- c. 0:1::1200:1000
- d. ::FFFF:24.123.12.6



Decompress the following addresses and show the complete unabbreviated IPv6 address:

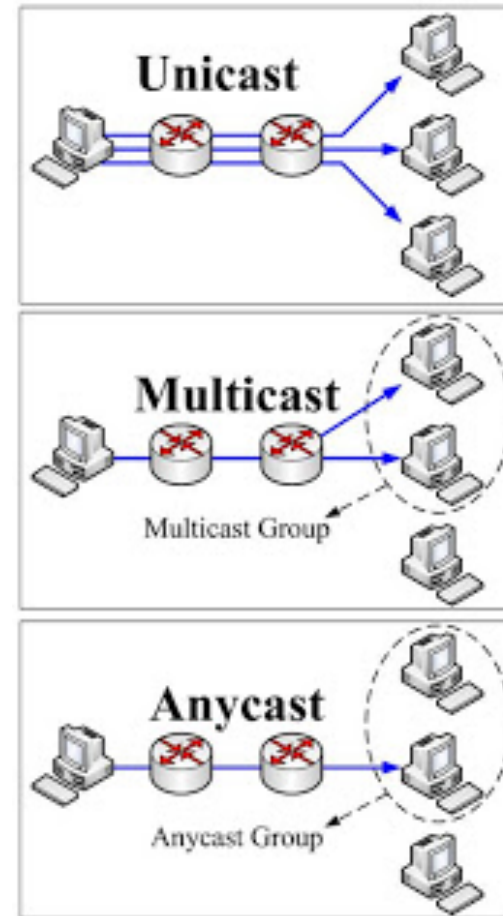
- a. 1111::2222
- b. ::
- c. 0:1::
- d. AAAA:A:AA::1234

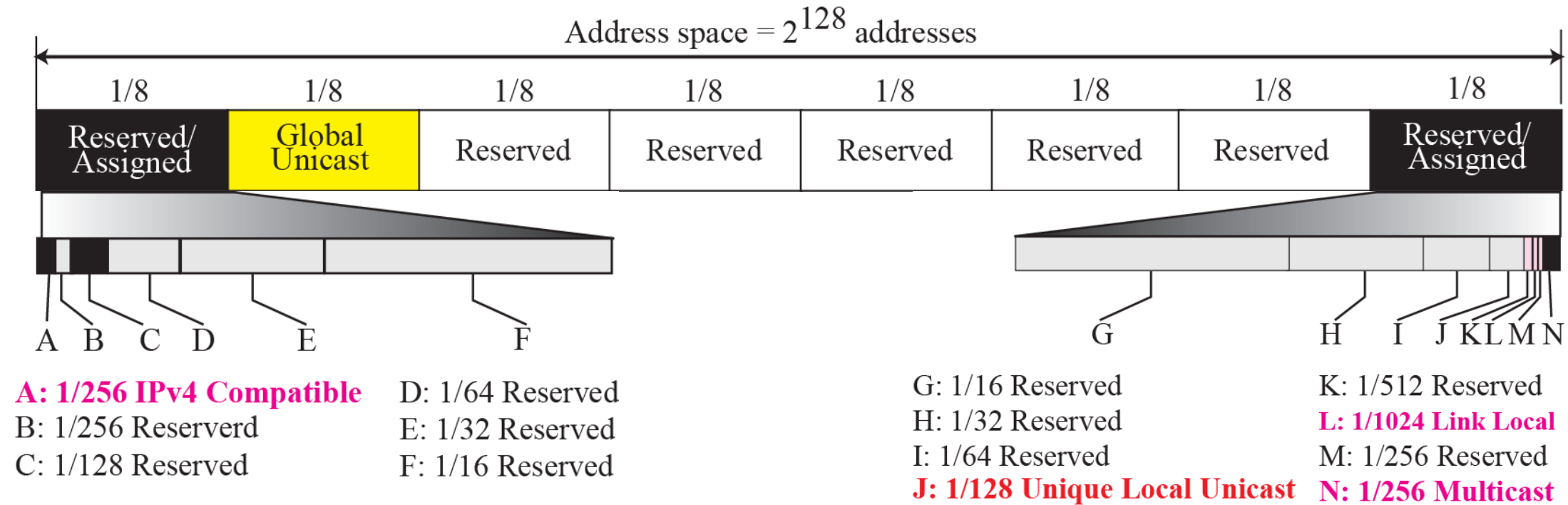
Solution

- a. 1111:0000:0000:0000:0000:0000:0000:2222
- b. 0000:0000:0000:0000:0000:0000:0000:0000
- c. 0000:0001:0000:0000:0000:0000:0000:0000
- d. AAAA:000A:00AA:0000:0000:0000:0000:1234



- In IPv6, a destination address belong to:
 - **Unicast Address (one-to-one)** – defines a single interface (computer or router)
 - **Multicast Address (one-to-many)** – also defines a group of computers, each member of the group receives a copy.
 - **Anycast Address (one-to-nearest)** – defines a group of computers that all share a single address.
 - **Broadcast Address (one-to-all)** – ×





- Larger address space
- Better header format
- New options
- Allowance for extension
- Support for resource allocation
- Support for more security

checksum: removed entirely to reduce processing time at each hop

options: allowed, but outside of header, indicated by “Next Header” field

ICMPv6: new version of ICMP

fragmentation/reassembly: no

COMPUTER NETWORKS

IPv6 Datagram Format

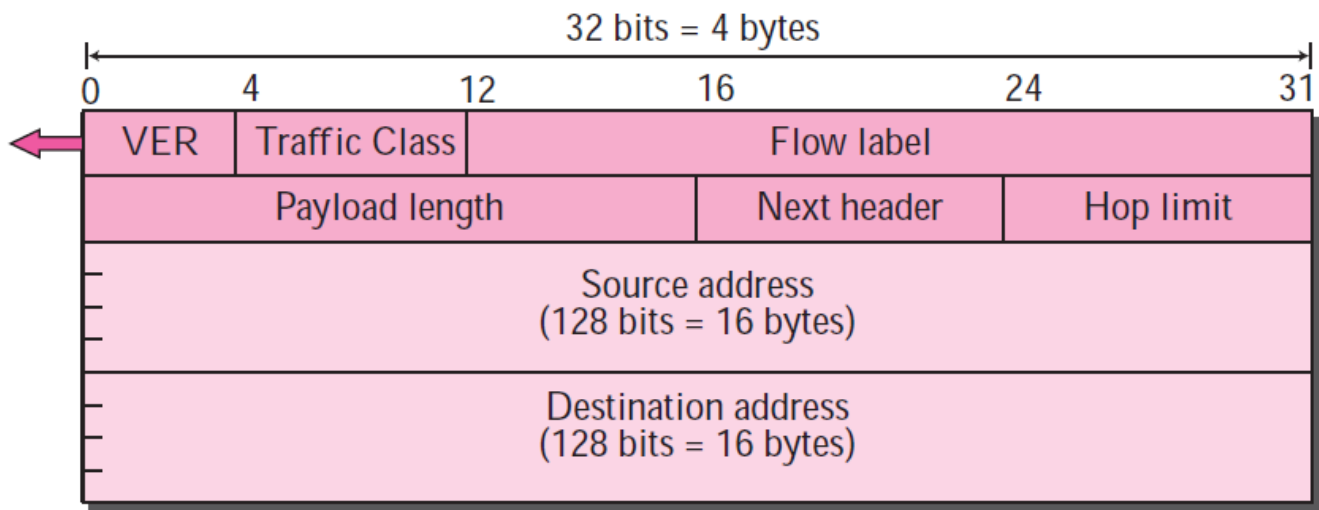
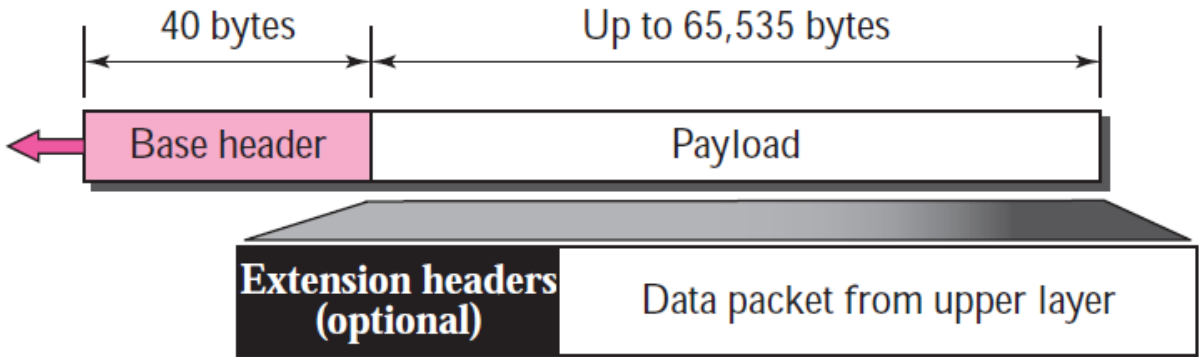
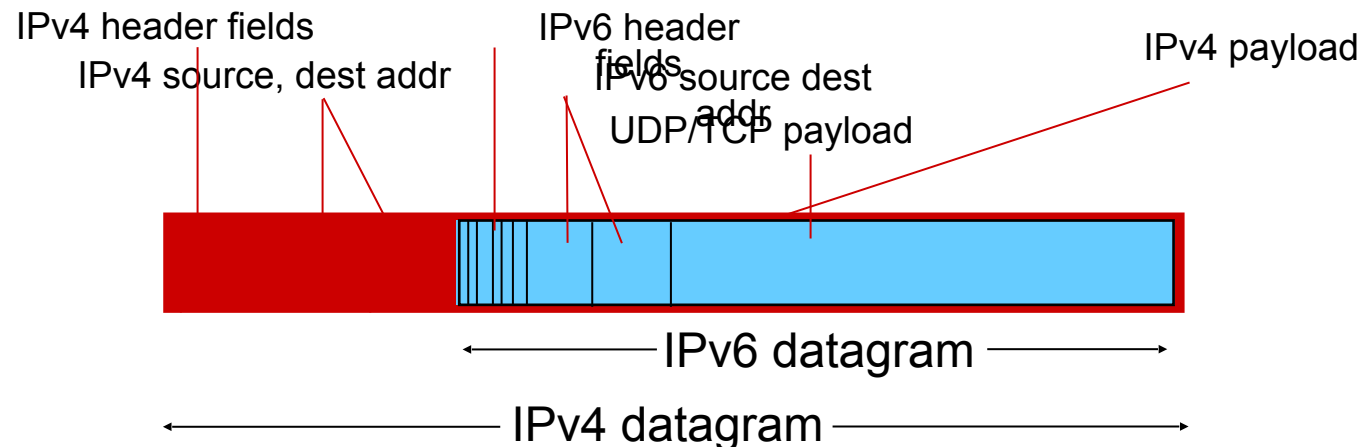


Table 27.1 Next Header Codes

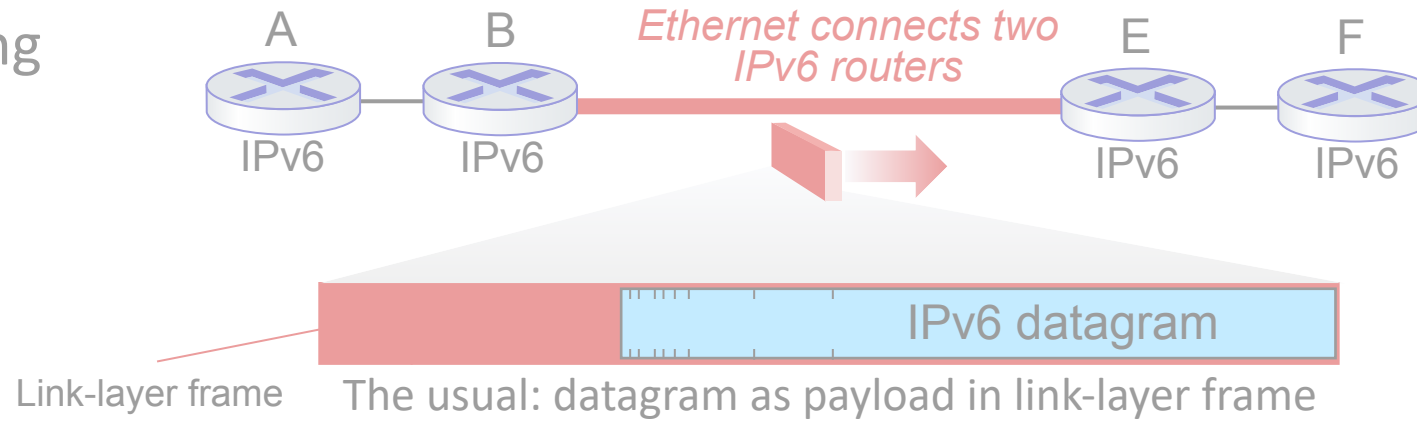
Code	Next Header	Code	Next Header
0	Hop-by-hop option	44	Fragmentation
2	ICMP	50	Encrypted security payload
6	TCP	51	Authentication
17	UDP	59	Null (No next header)
43	Source routing	60	Destination option

Format of the base header

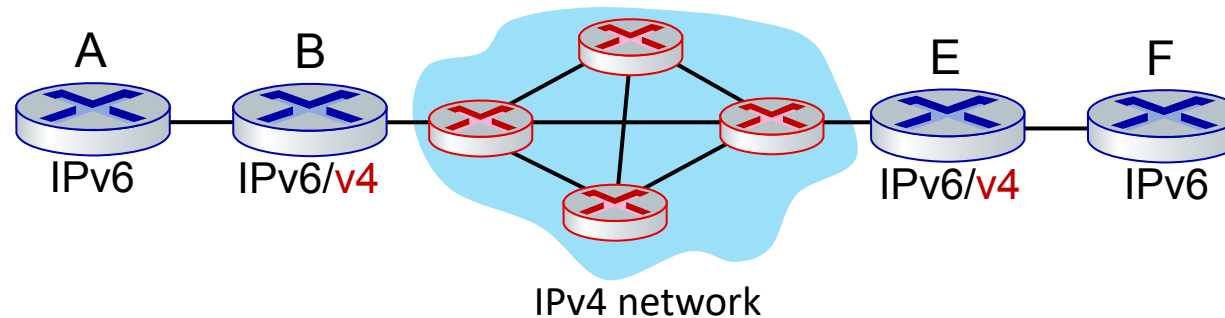
- not all routers can be upgraded simultaneously
 - no “flag days”
 - how will network operate with mixed IPv4 and IPv6 routers?
- **tunneling**: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers (“packet within a packet”)
 - tunneling used extensively in other contexts (4G/5G)



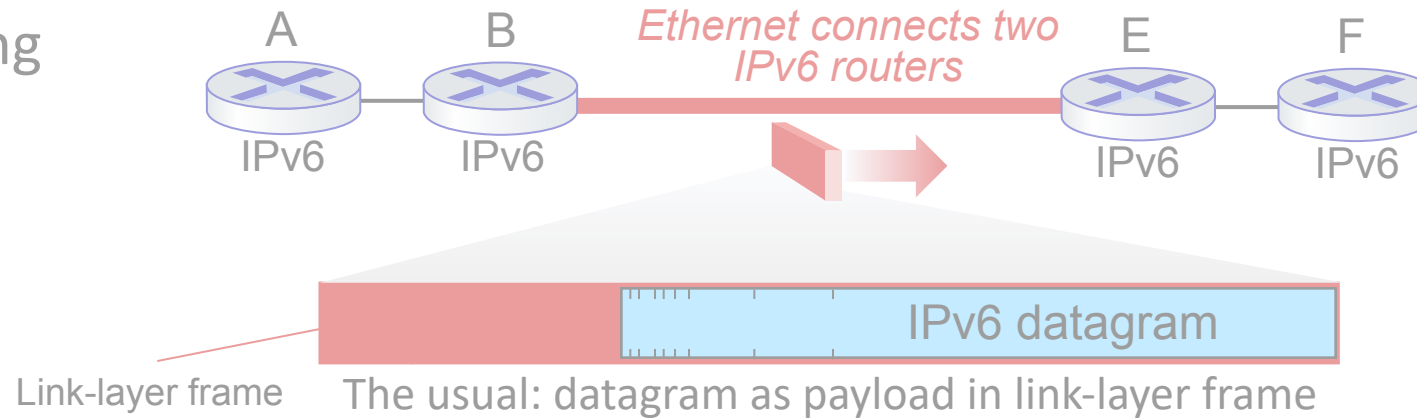
Ethernet connecting
two IPv6 routers:



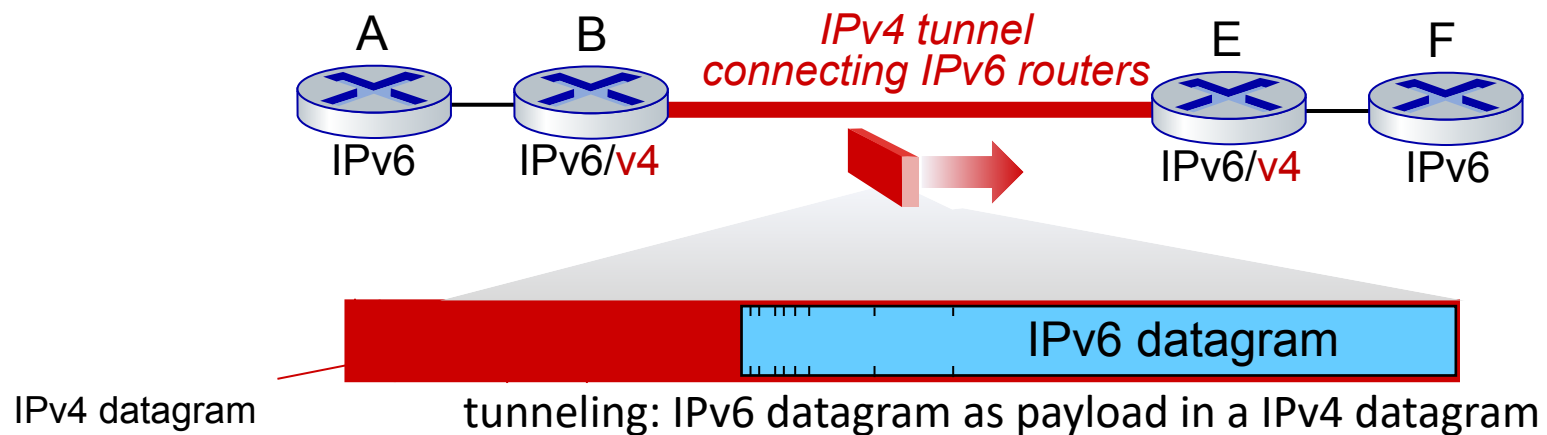
IPv4 network
connecting two
IPv6 routers

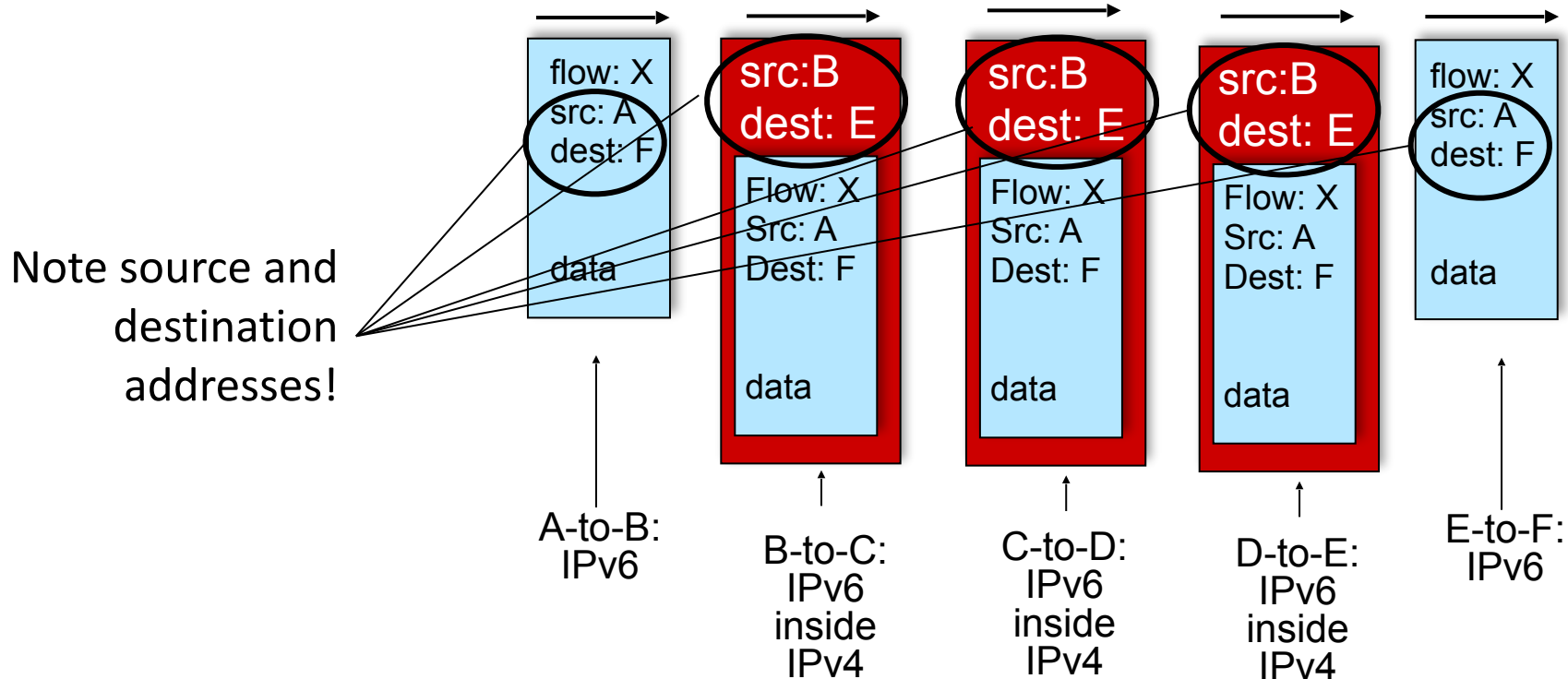
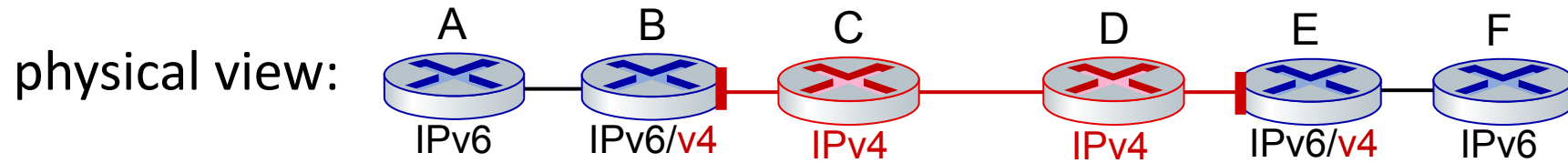


Ethernet connecting
two IPv6 routers:



IPv4 tunnel
connecting two
IPv6 routers





<https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/state-of-the-internet-ipv6-adoption-visualization.jsp>

Unit – 4 Network Layer and Internet Protocol

4.5 Introduction to Network Layer Protocols

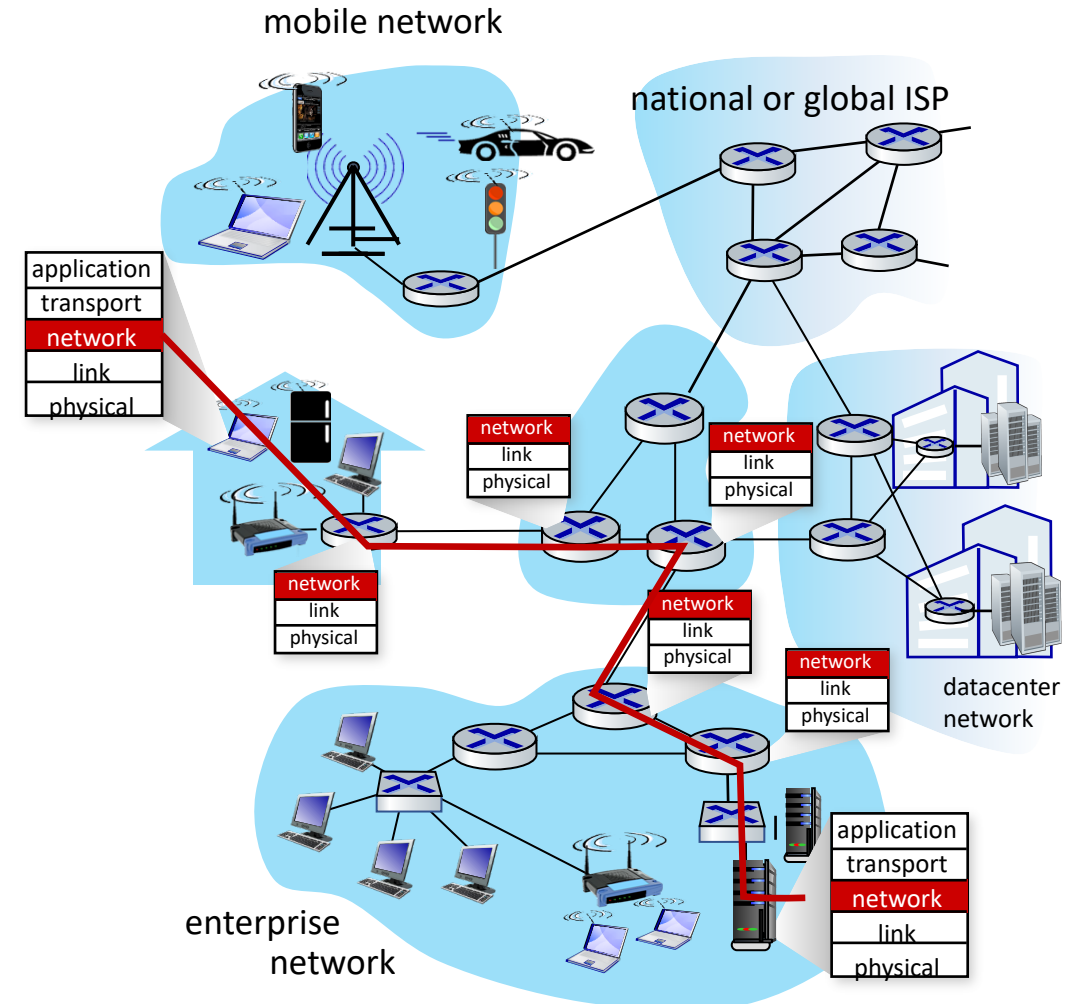
- DHCP
- ICMP

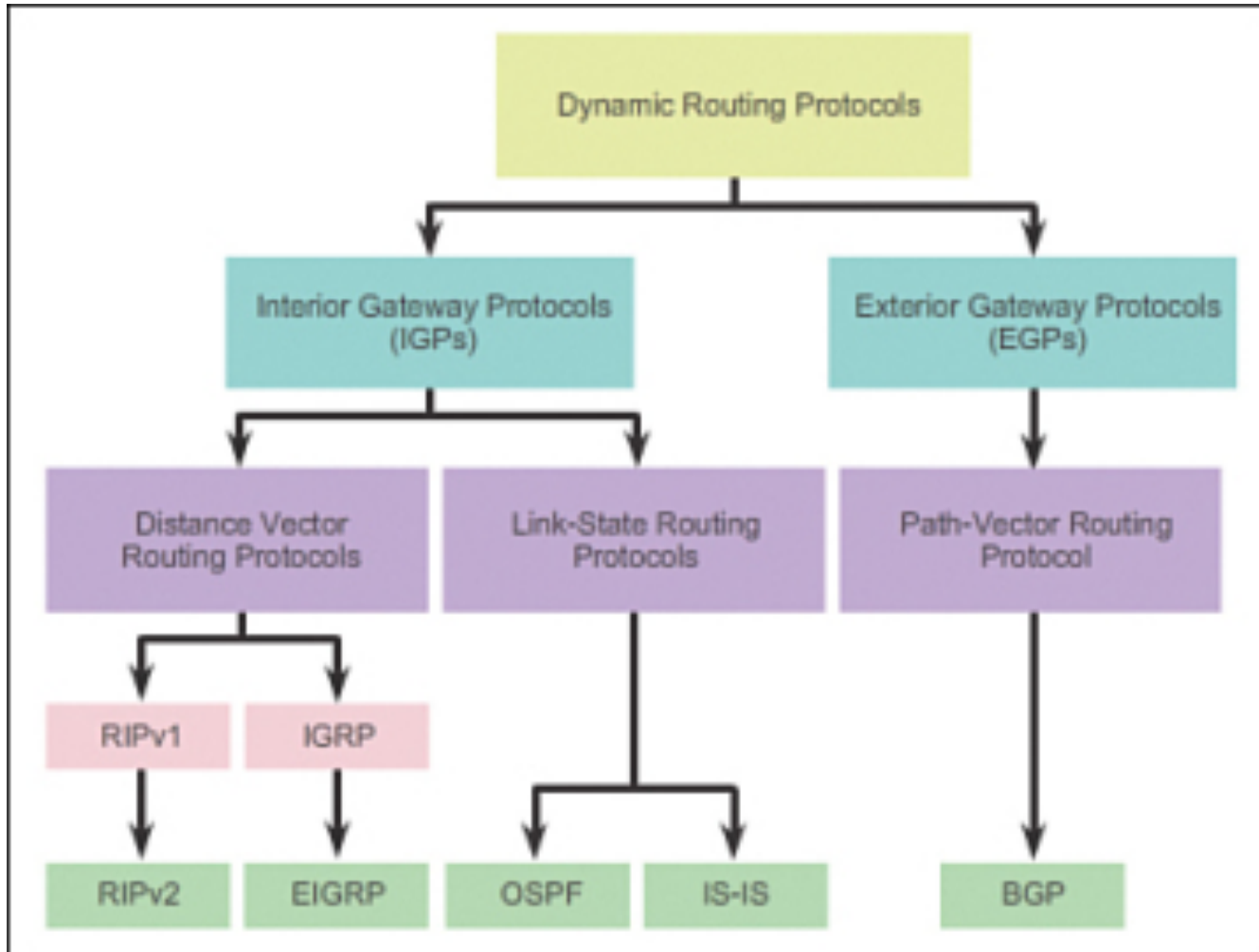
4.6 IPv6 Addressing

4.7 Introduction to Routing Algorithms

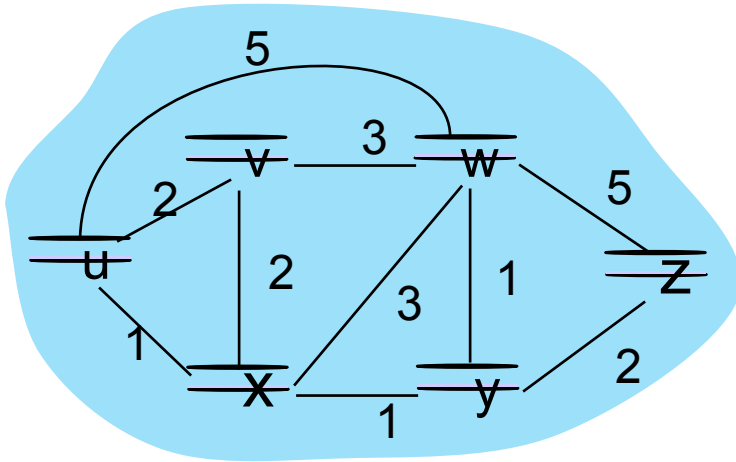
Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

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

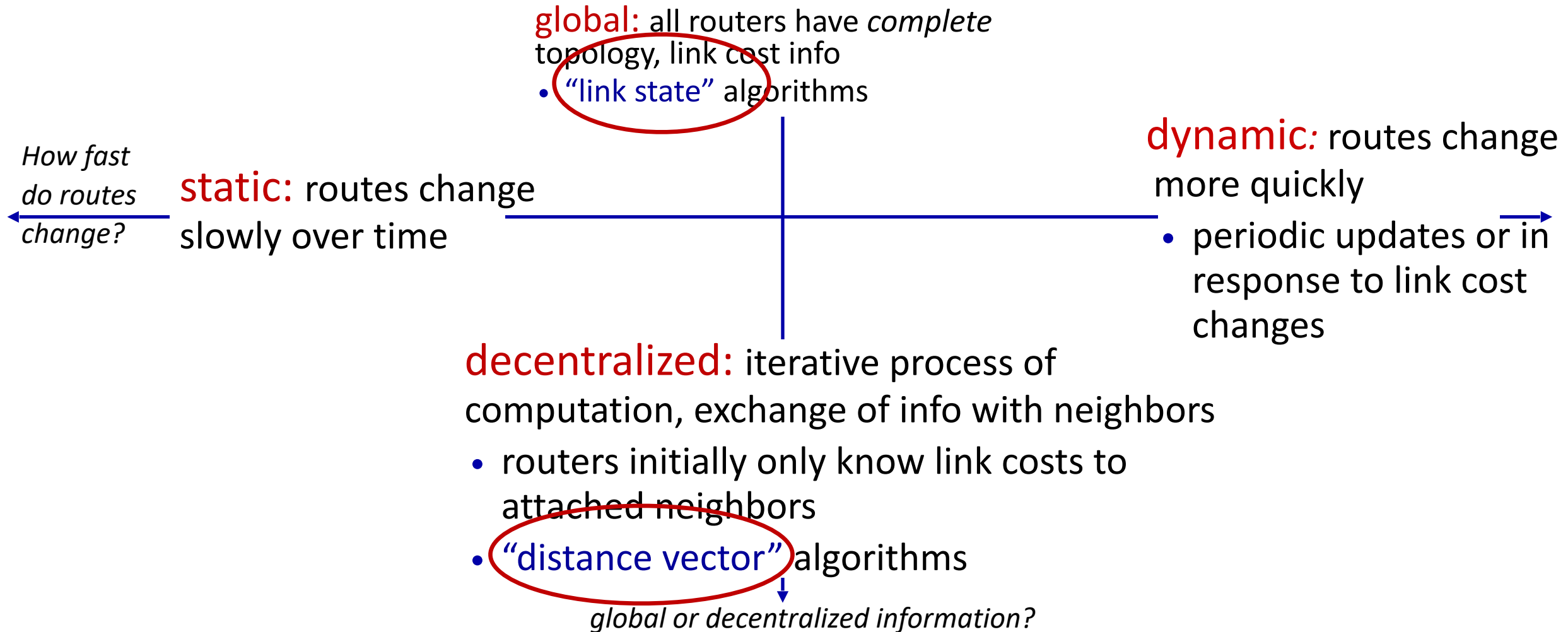
cost defined by network operator:
could always be 1, or inversely related
to bandwidth, or inversely related to
congestion

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



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 ; $= \infty$ 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

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

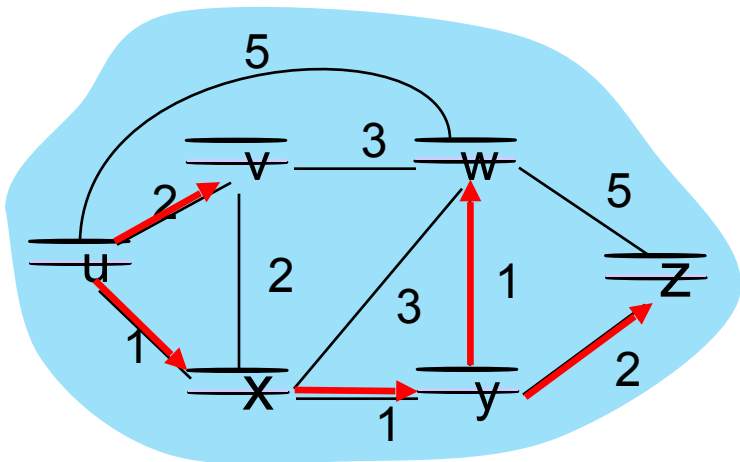
13 /* new least-path-cost to v is either old least-cost-path to v or known

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	u, x	2, u	4, x		2, x	∞
2	u, x, y	2, u	3, y			4, y
3	u, x, y, v		3, y			4, y
4	u, x, y, v, w					4, y
5	u, x, y, v, w, z					



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

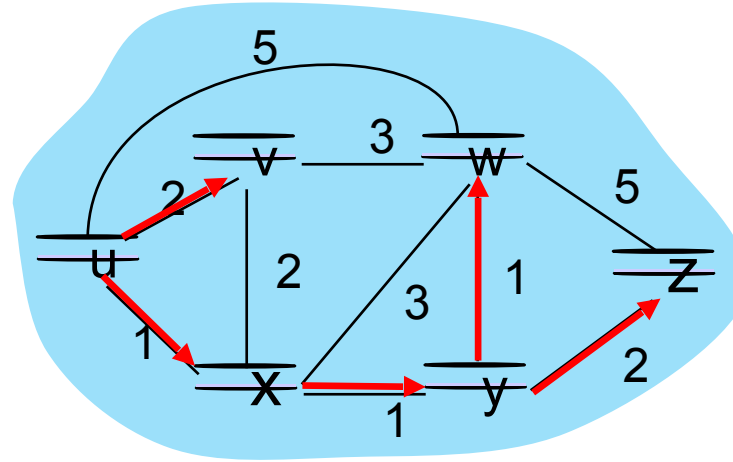
find a not in N' such that $D(a)$ is a minimum

add a to N'

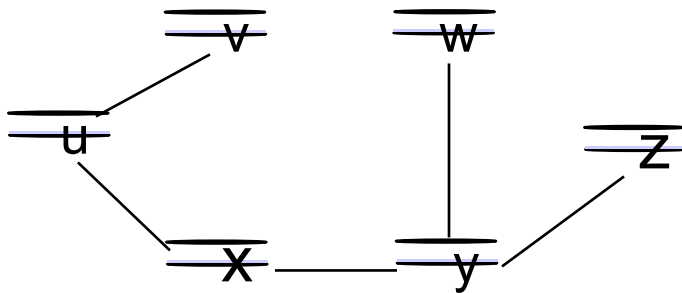
update $D(b)$ for all b adjacent to a and not in N' :

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

Example contd..



resulting least-cost-path tree from u:



resulting forwarding table in u:

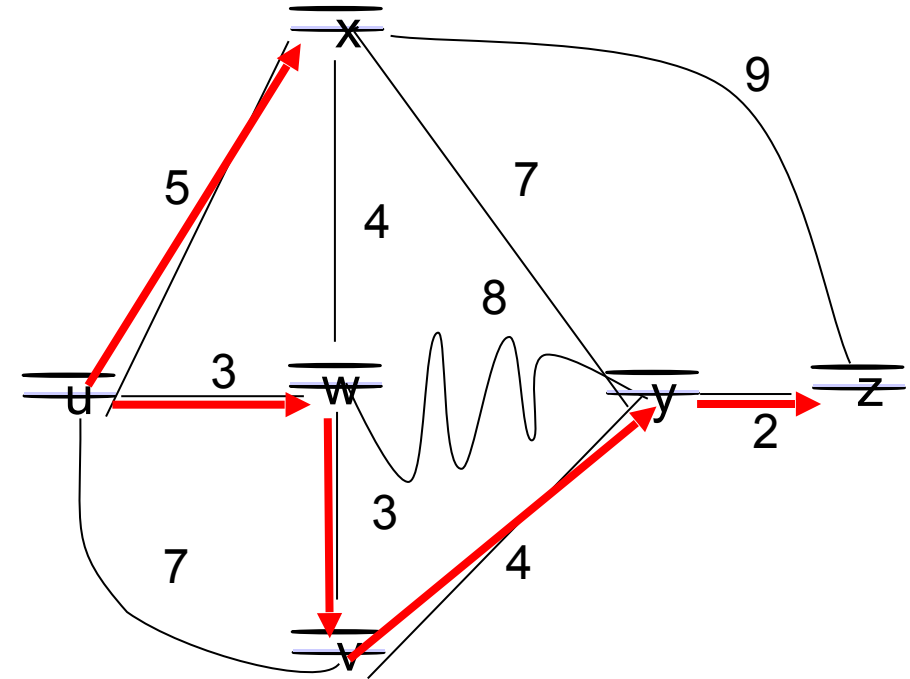
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

route from u to v directly

route from u to all other destinations via x

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	∞	∞
2	uwvx	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)

Routing Protocols - Bellman-Ford Distance Vector Routing Algorithm

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:

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

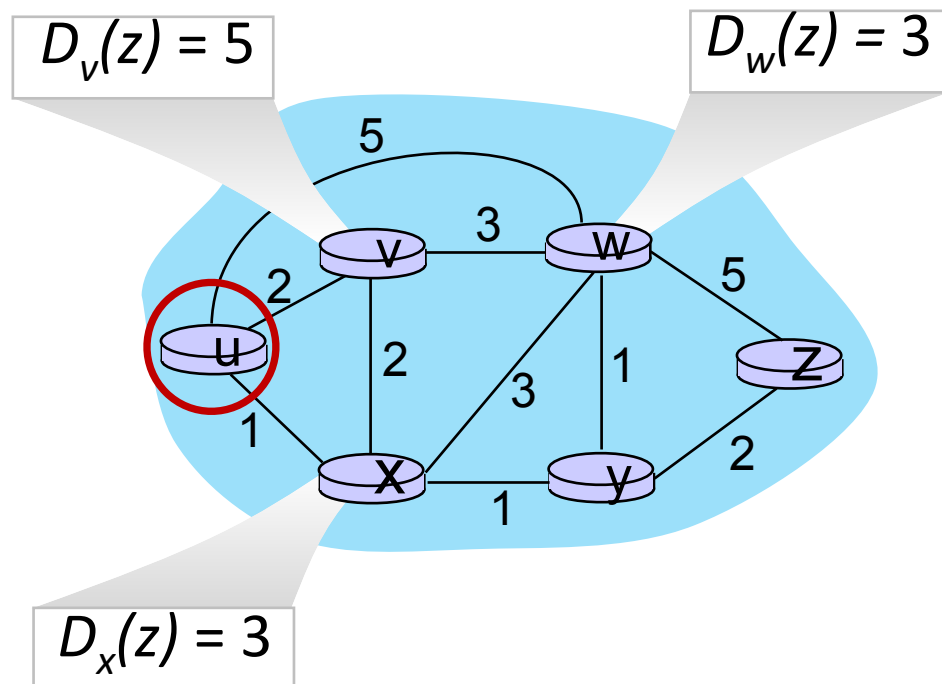
min taken over all neighbors v of x

direct cost of link from x to v

v 's 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:

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

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

Distance vector algorithm:

key idea:

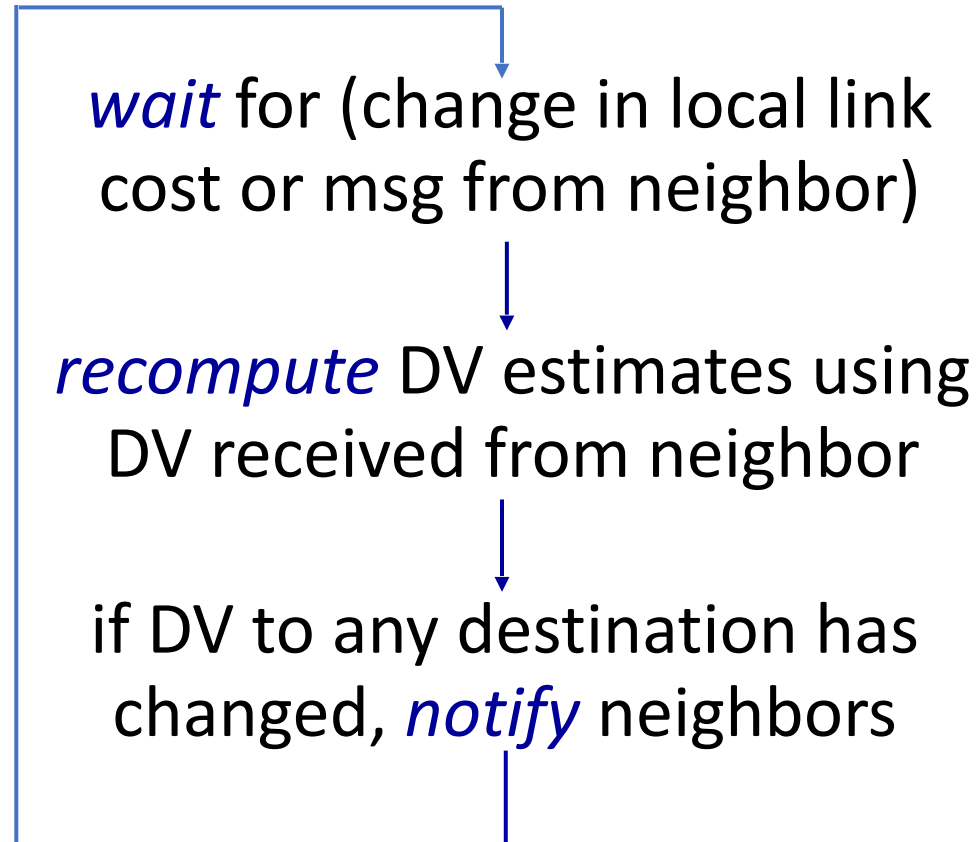
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

- under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm:

each node:



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!

Routing Protocols - Bellman-Ford Distance Vector Routing Algorithm

Distance vector: example

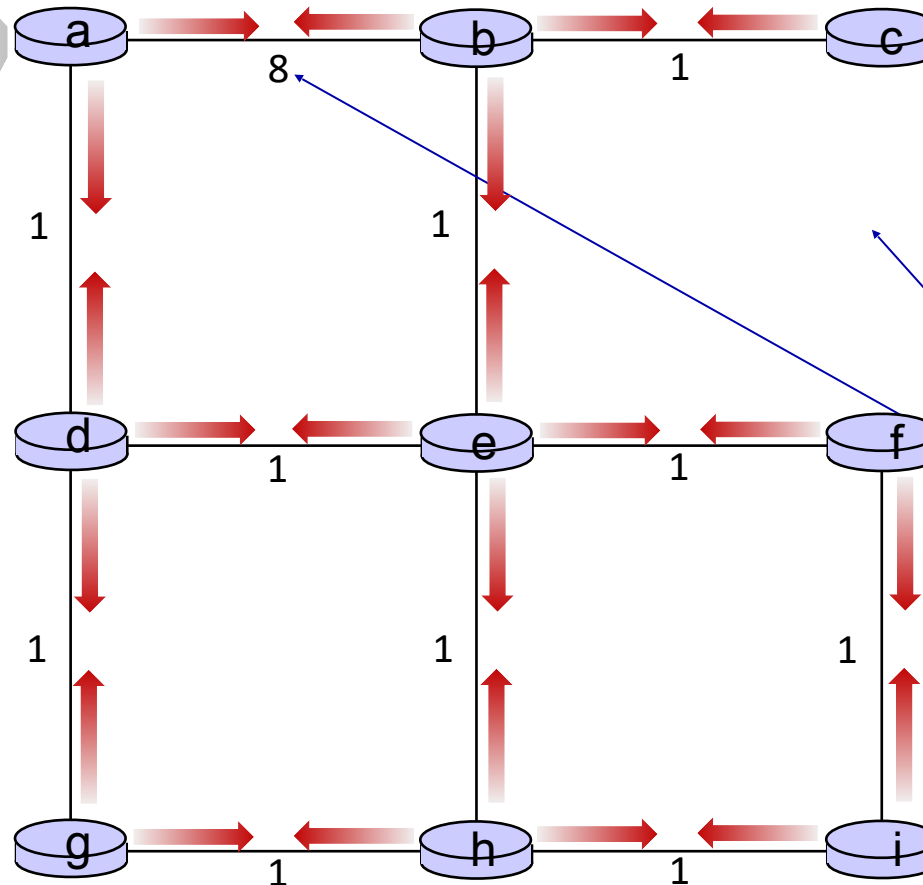


t=0

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

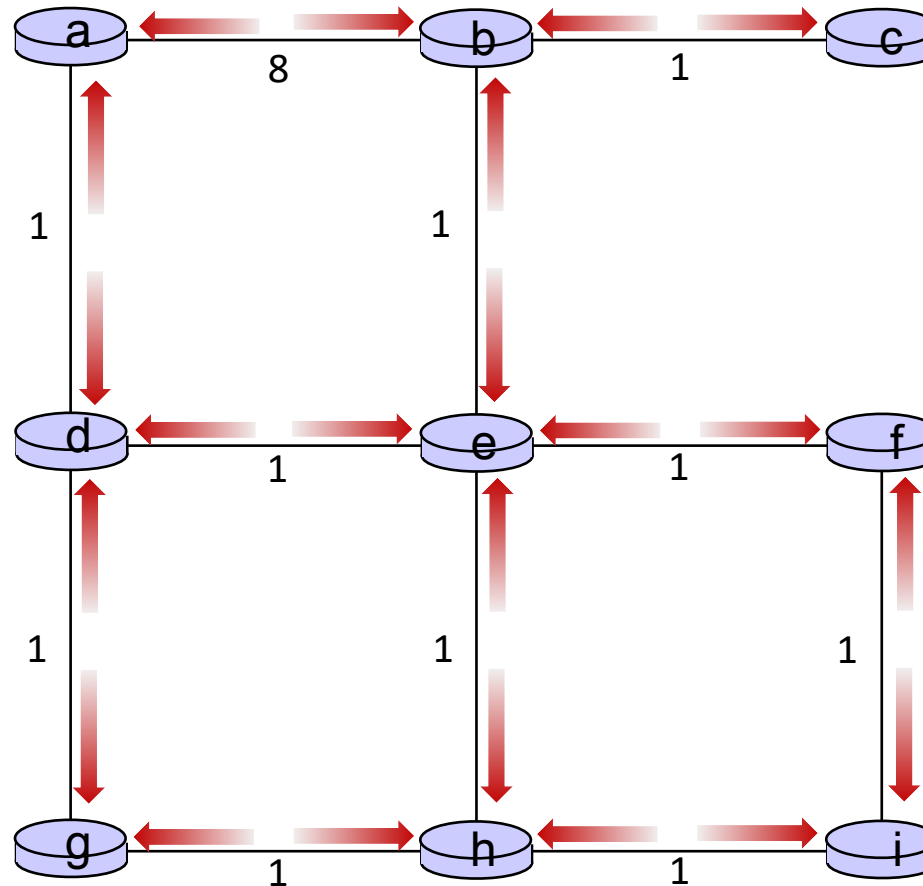
DV in a:

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

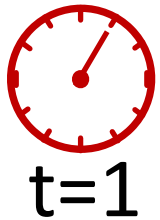


A few asymmetries:

- missing link
- larger cost

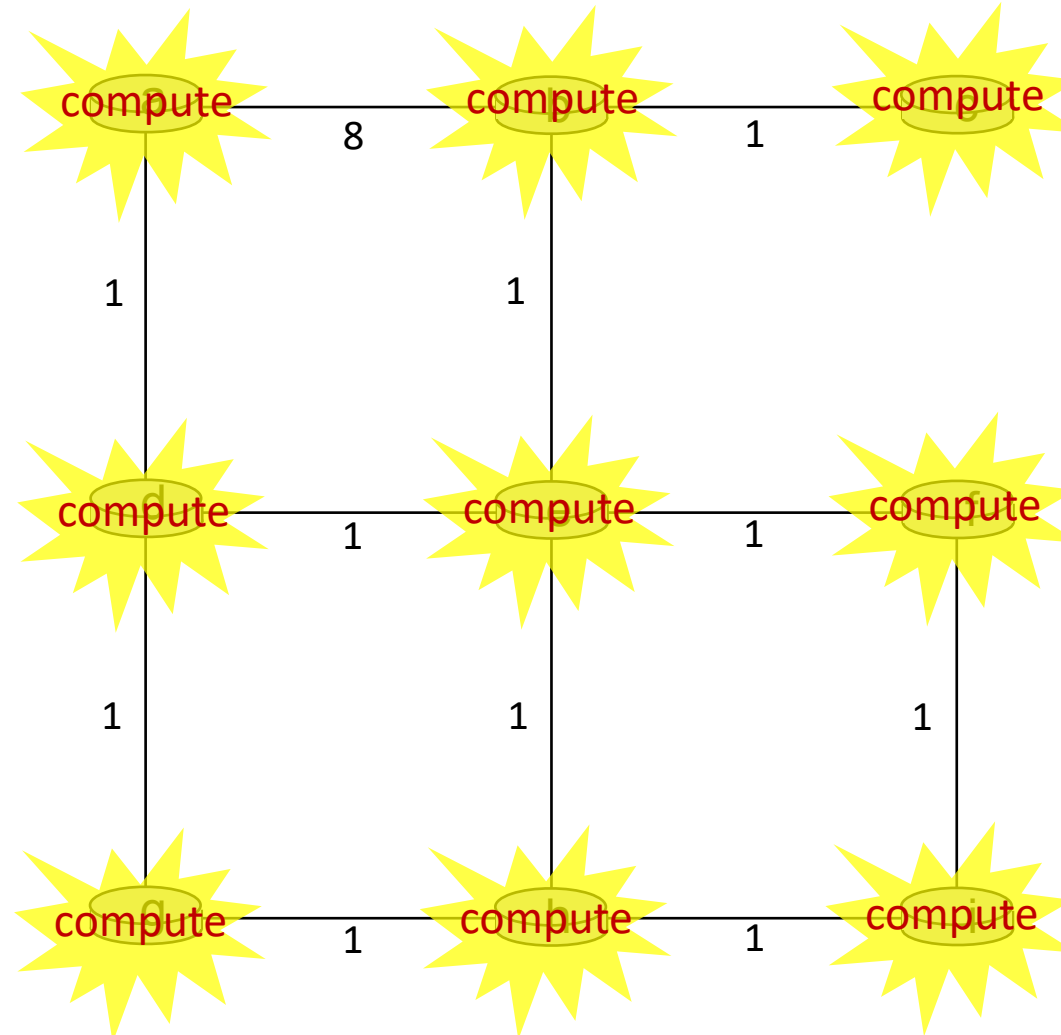


Distance vector: Iteration



All nodes:

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

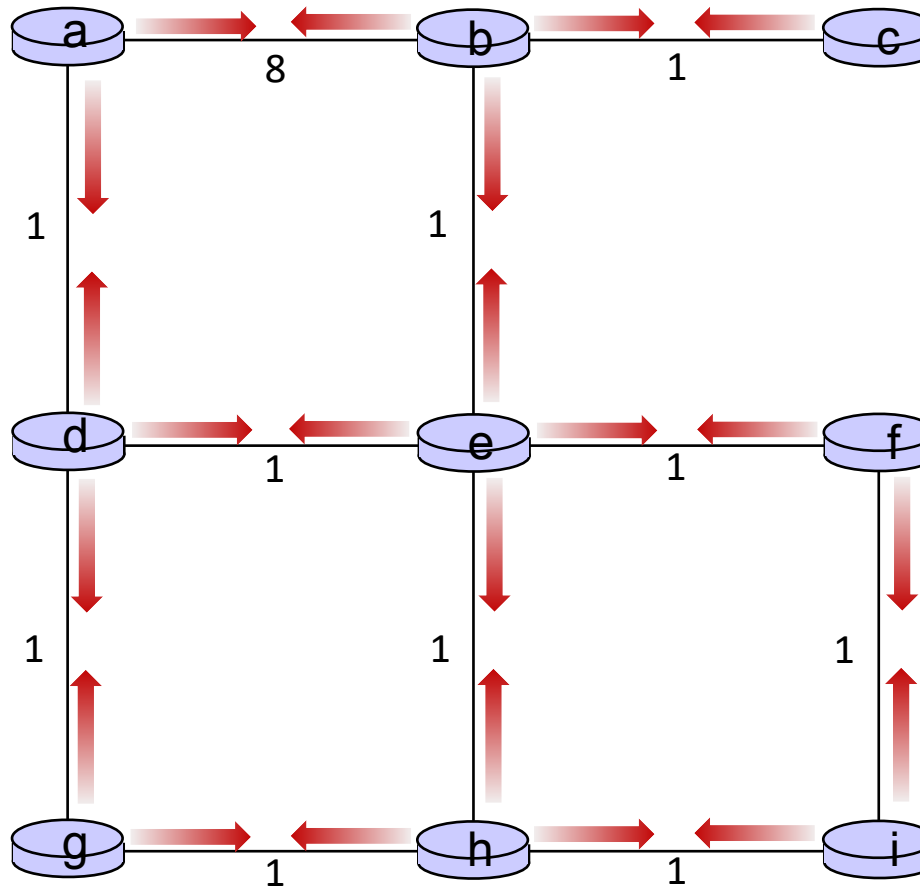


Distance vector: Iteration

 $t=1$

All nodes:

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



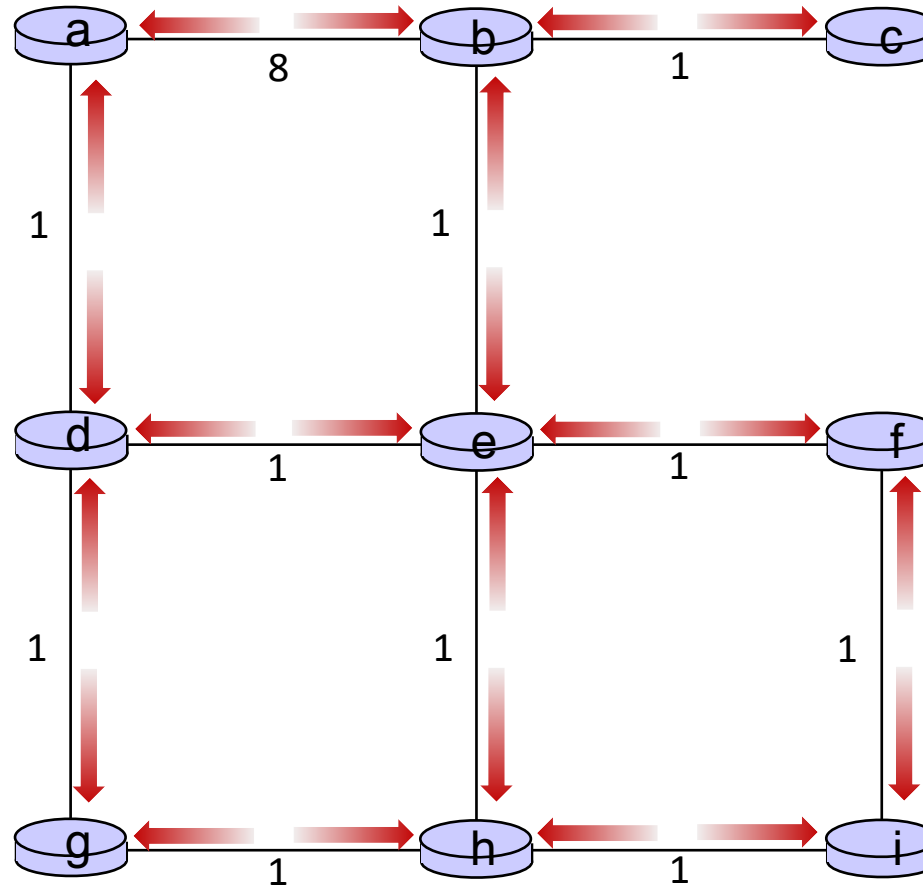
Distance vector: Iteration



t=2

All nodes:

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



Routing Protocols - Bellman-Ford Distance Vector Routing Algorithm

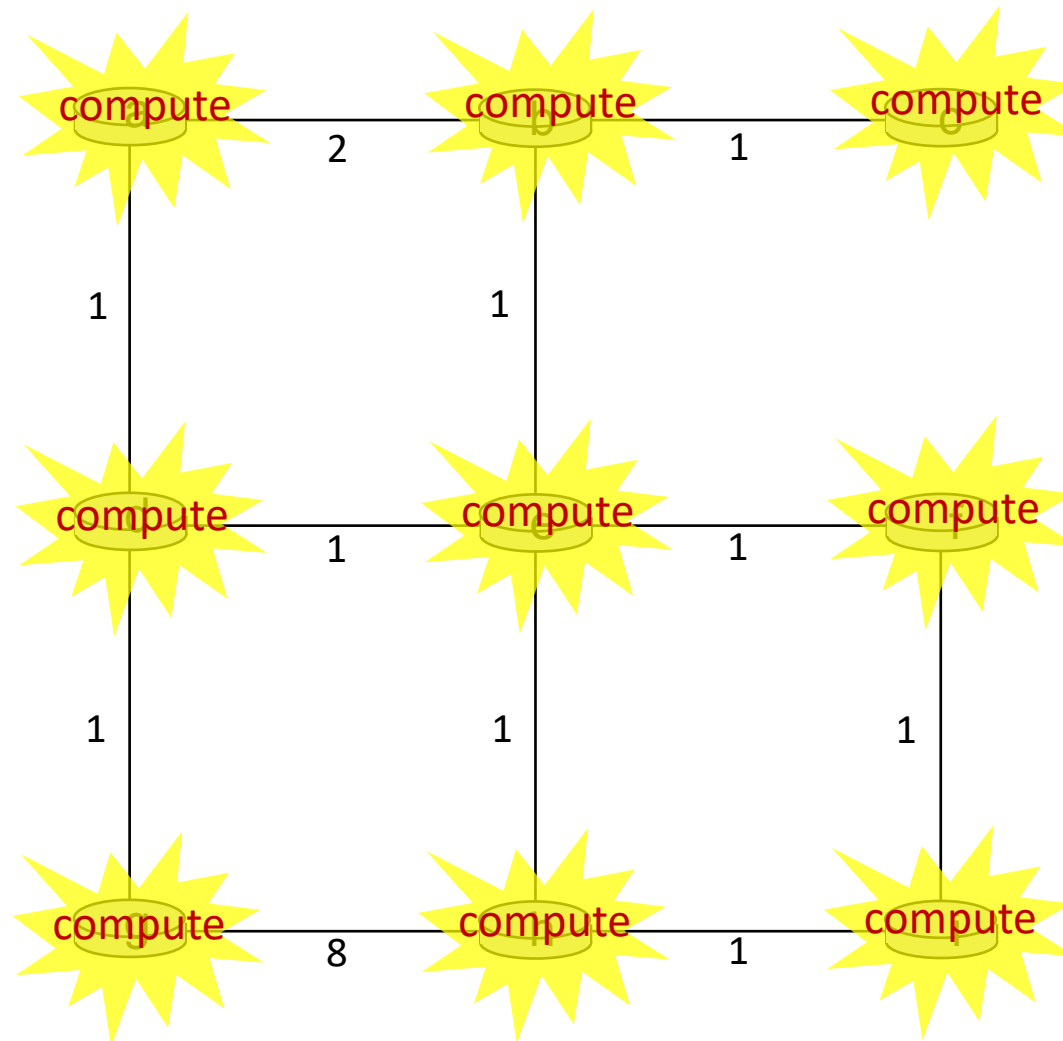
Distance vector: Iteration



t=2

All nodes:

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



Routing Protocols - Bellman-Ford Distance Vector Routing Algorithm

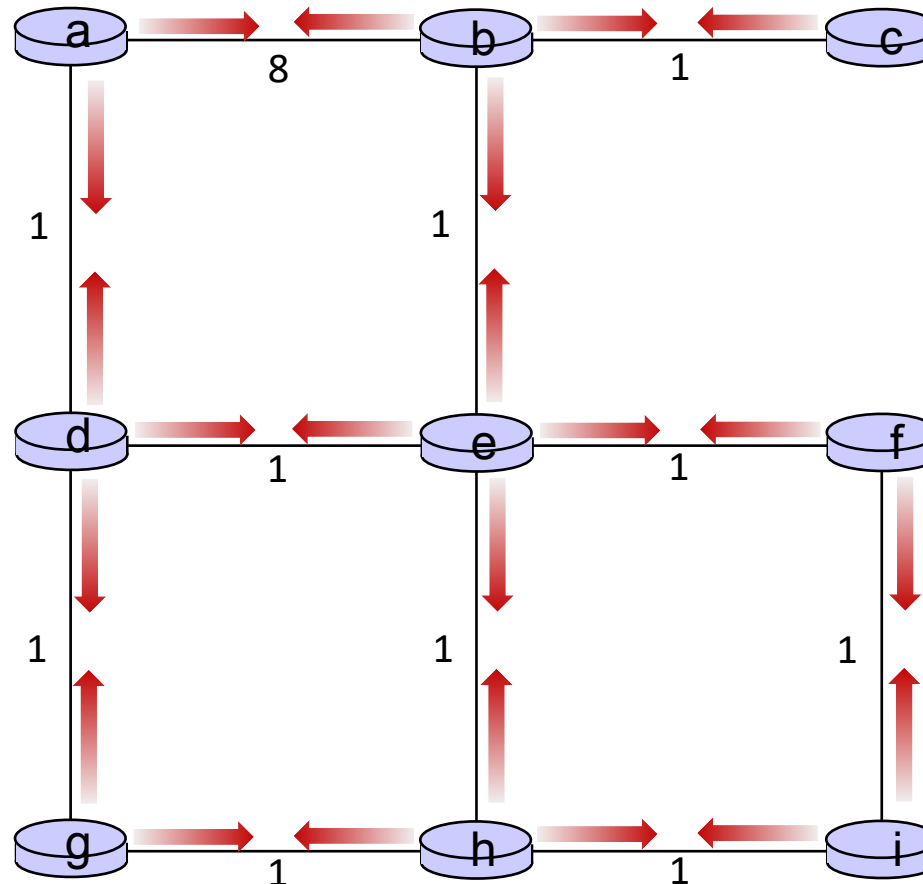
Distance vector: Iteration



t=2

All nodes:

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



.... and so on

Distance vector example: computation



t=1

- b receives DVs from a, c, e

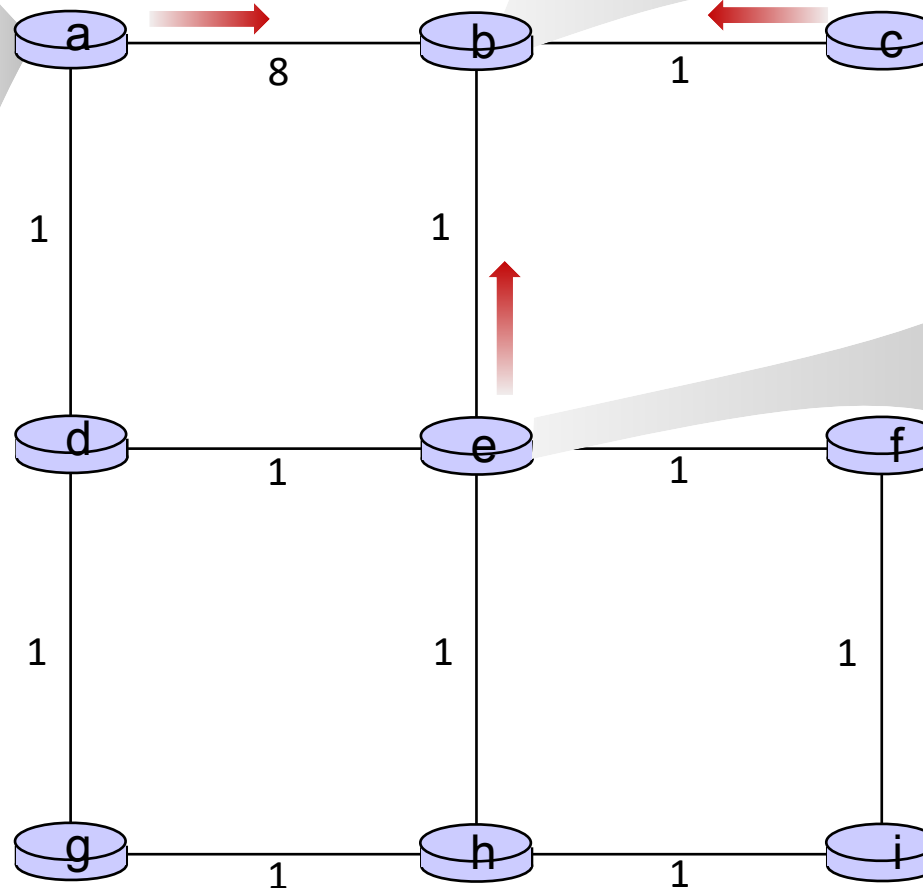
Let's next take a look at the iterative *computations* at nodes

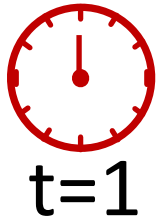
DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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

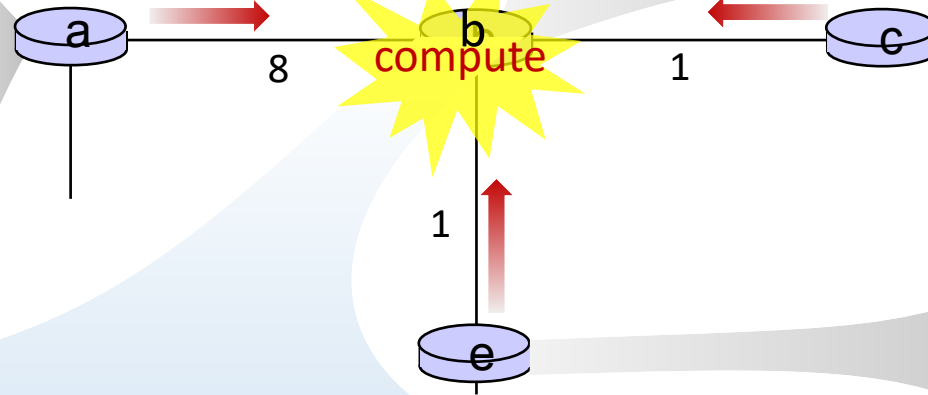




- b receives DVs from a, c, e, computes:

$$\begin{aligned}
 D_b(a) &= \min\{c_{b,a} + D_a(a), c_{b,c} + D_c(a), c_{b,e} + D_e(a)\} = \min\{8, \infty, \infty\} = 8 \\
 D_b(c) &= \min\{c_{b,a} + D_a(c), c_{b,c} + D_c(c), c_{b,e} + D_e(c)\} = \min\{\infty, 1, \infty\} = 1 \\
 D_b(d) &= \min\{c_{b,a} + D_a(d), c_{b,c} + D_c(d), c_{b,e} + D_e(d)\} = \min\{9, 2, \infty\} = 2 \\
 D_b(e) &= \min\{c_{b,a} + D_a(e), c_{b,c} + D_c(e), c_{b,e} + D_e(e)\} = \min\{\infty, \infty, 1\} = 1 \\
 D_b(f) &= \min\{c_{b,a} + D_a(f), c_{b,c} + D_c(f), c_{b,e} + D_e(f)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(g) &= \min\{c_{b,a} + D_a(g), c_{b,c} + D_c(g), c_{b,e} + D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty \\
 D_b(h) &= \min\{c_{b,a} + D_a(h), c_{b,c} + D_c(h), c_{b,e} + D_e(h)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(i) &= \min\{c_{b,a} + D_a(i), c_{b,c} + D_c(i), c_{b,e} + D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty
 \end{aligned}$$

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



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

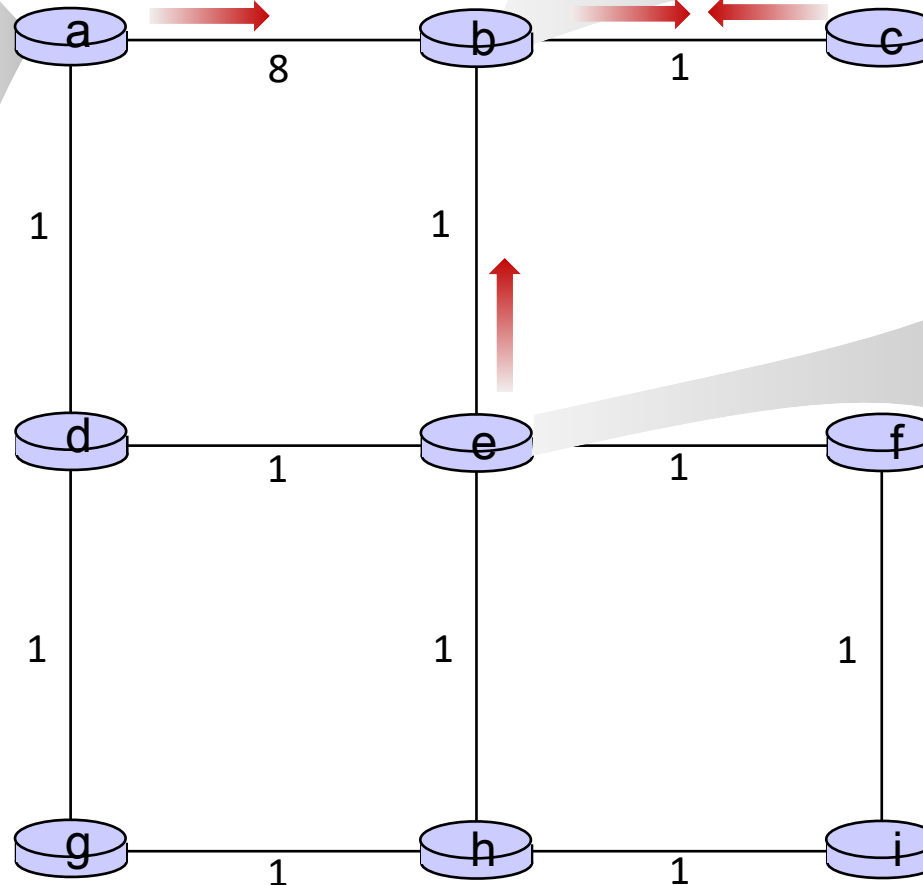
DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

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



- c receives DVs from b

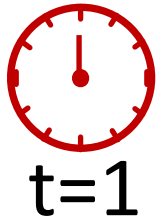
DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



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

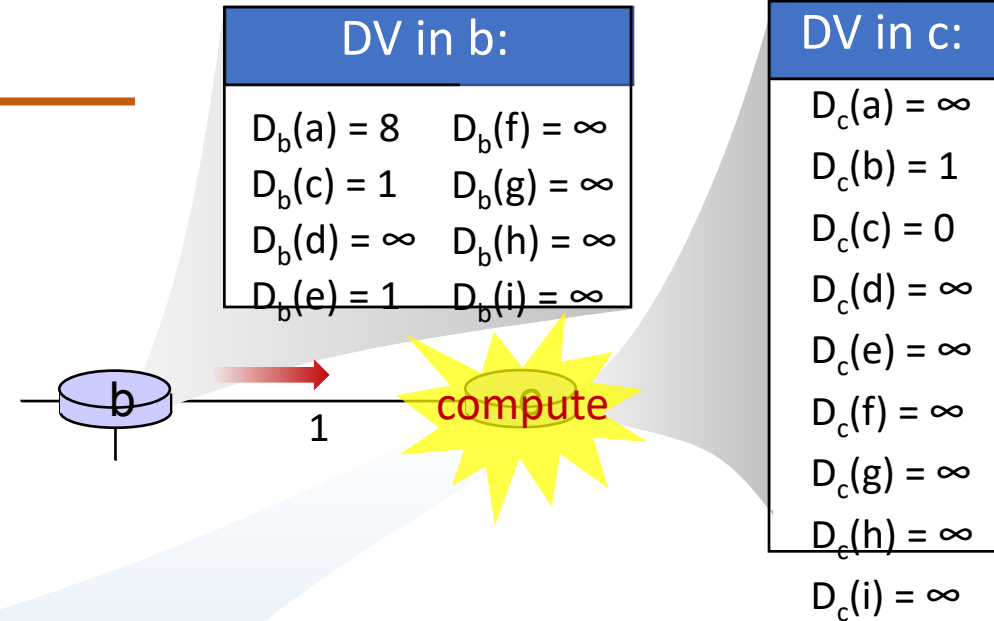
DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

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



- c receives DVs from b computes:

$$\begin{aligned}
 D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\
 D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\
 D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\
 D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\
 D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\
 D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\
 D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\
 D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty
 \end{aligned}$$



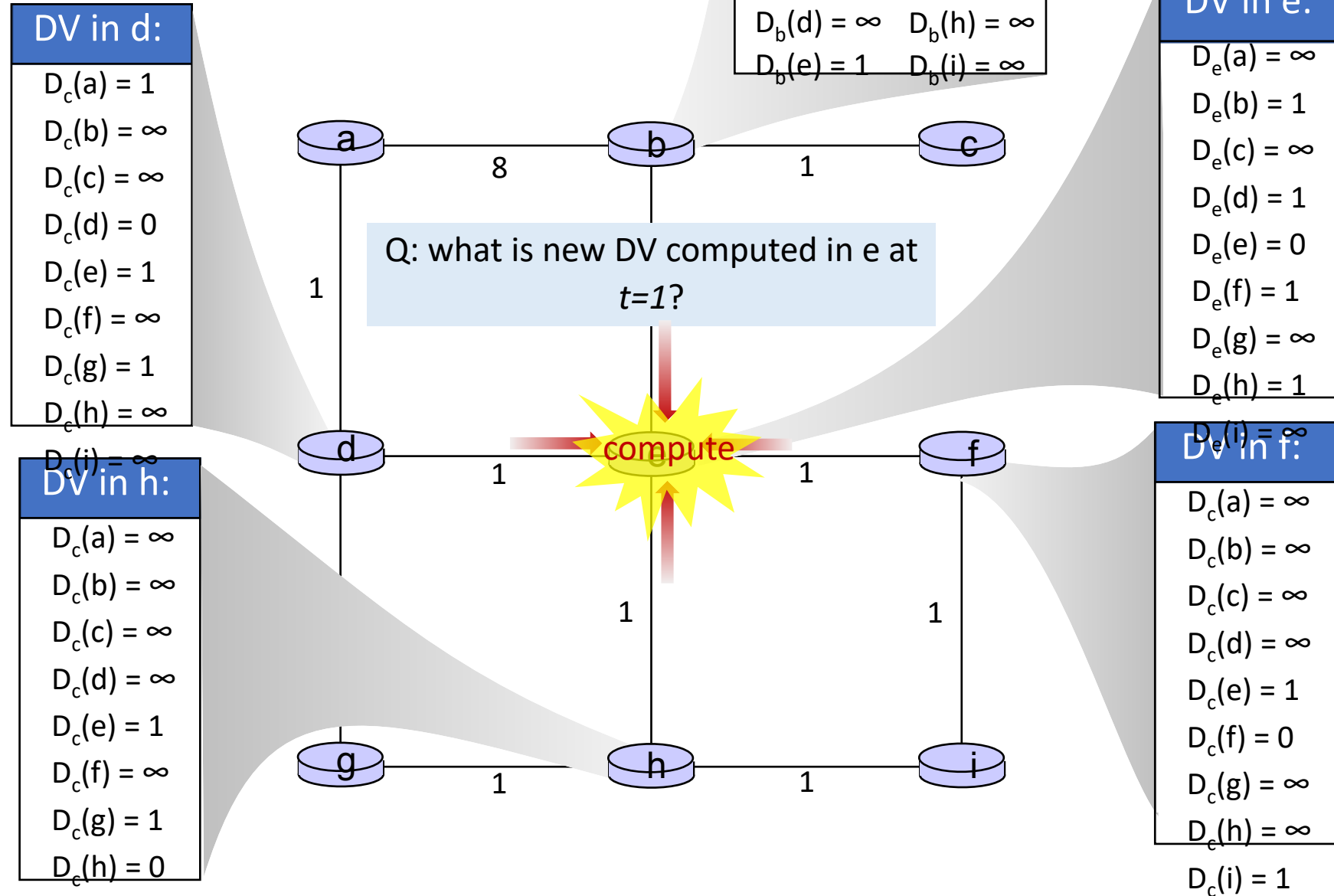
DV in c:
$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$

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








$t=1$

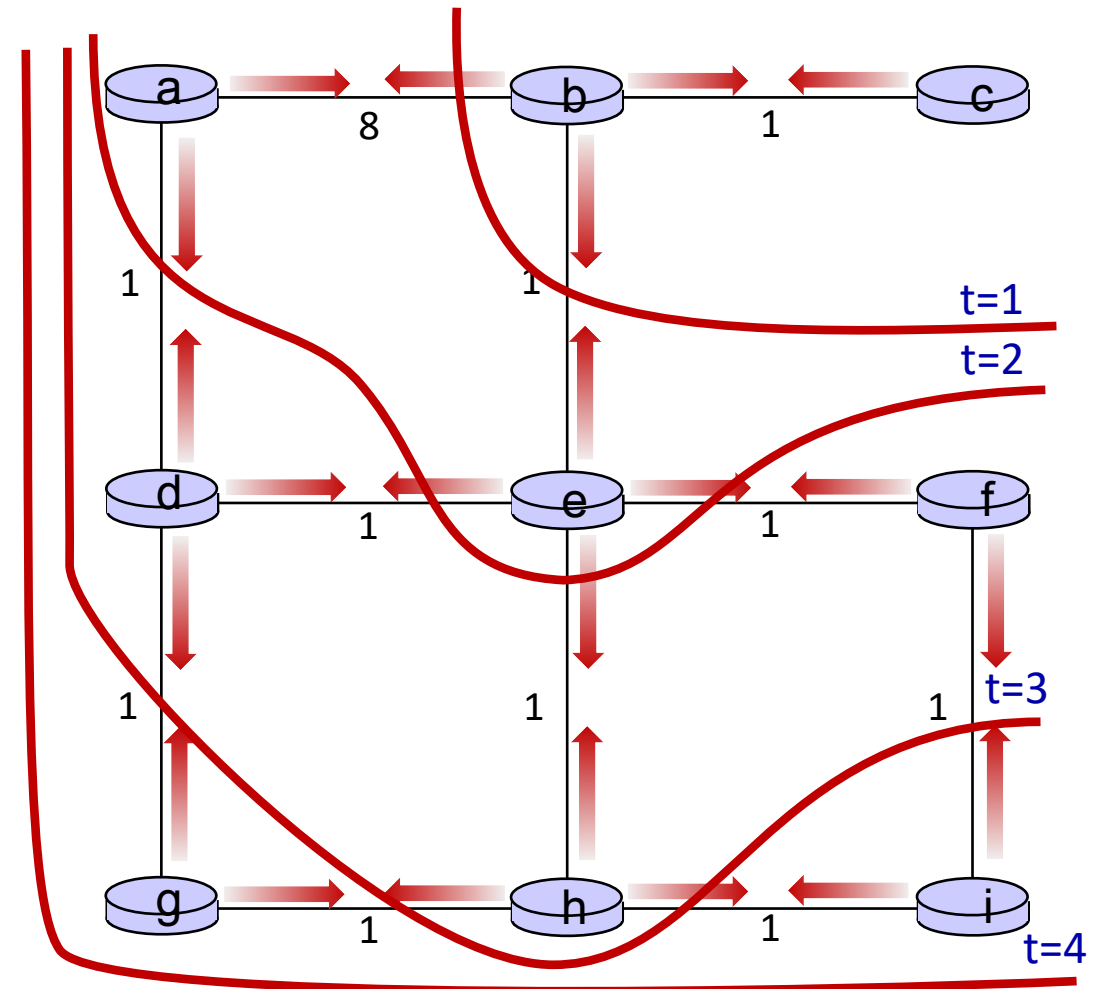
- e receives DVs from b, d, f, h



Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

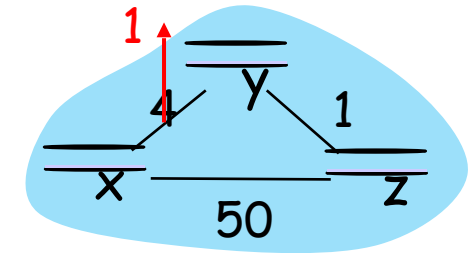
-  $t=0$ c's state at $t=0$ is at c only
-  $t=1$ c's state at $t=0$ has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
-  $t=2$ c's state at $t=0$ may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
-  $t=3$ c's state at $t=0$ may influence distance vector computations up to **3** hops away, i.e., at 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: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors



t_0 : y detects link-cost change, updates its DV, informs its neighbors.

“good news
travels fast”

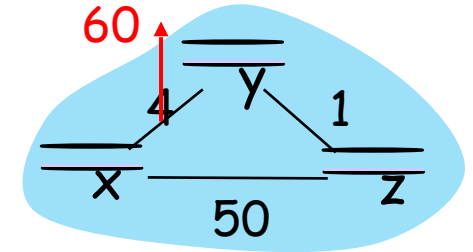
t_1 : z receives update from y , updates its table, computes new least cost to x , sends its neighbors its DV.

t_2 : y receives z 's update, updates its distance table. y 's least costs do *not* change, so y does *not* send a message to z .

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- “bad news travels slow” – count-to-infinity problem:
 - y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes “my new cost to x will be 6, via z); notifies z of new cost of 6 to x.
 - z learns that path to x via y has new cost 6, so z computes “my new cost to x will be 7 via y), notifies y of new cost of 7 to x.
 - y learns that path to x via z has new cost 7, so y computes “my new cost to x will be 8 via y), notifies z of new cost of 8 to x.
 - z learns that path to x via y has new cost 8, so z computes “my new cost to x will be 9 via y), notifies y of new cost of 9 to x.
 - ...
- see text for solutions. *Distributed algorithms are tricky!*



	Distance Vector	Link State
Primary principle	Send entire routing table to its neighbors	Only provides link state information
Learning about network	Learn about network only from neighbors	Learn about network from all routers
Building the routing table	Based on inputs from only neighbors	Based on complete database collected from all routers
Advertisement of updates	Sends periodic updates every 30-90 seconds – Broadcasts updates	Use triggered updates, only when there is a change – Multicasts updates
Routing loops	Vulnerable	Less prone to routing loops

	Distance vector	Link State
Convergence (stabilization)	Slow	Fast
Resources	Less CPU power and memory	More CPU power and memory required
Cost		More than Distance vector
Scalability		More scalable than distance vector
Examples	RIP, IGRP	OSPF, IS-IS



Thank You
For Your Attention



PES
UNIVERSITY

CELEBRATING 50 YEARS

THANK YOU

TEAM NETWORKS

Department of Computer Science and Engineering