Information Network

Lecture 11: Network Layer – Control Plane

Holger Thies

KYOTO UNIVERSITY

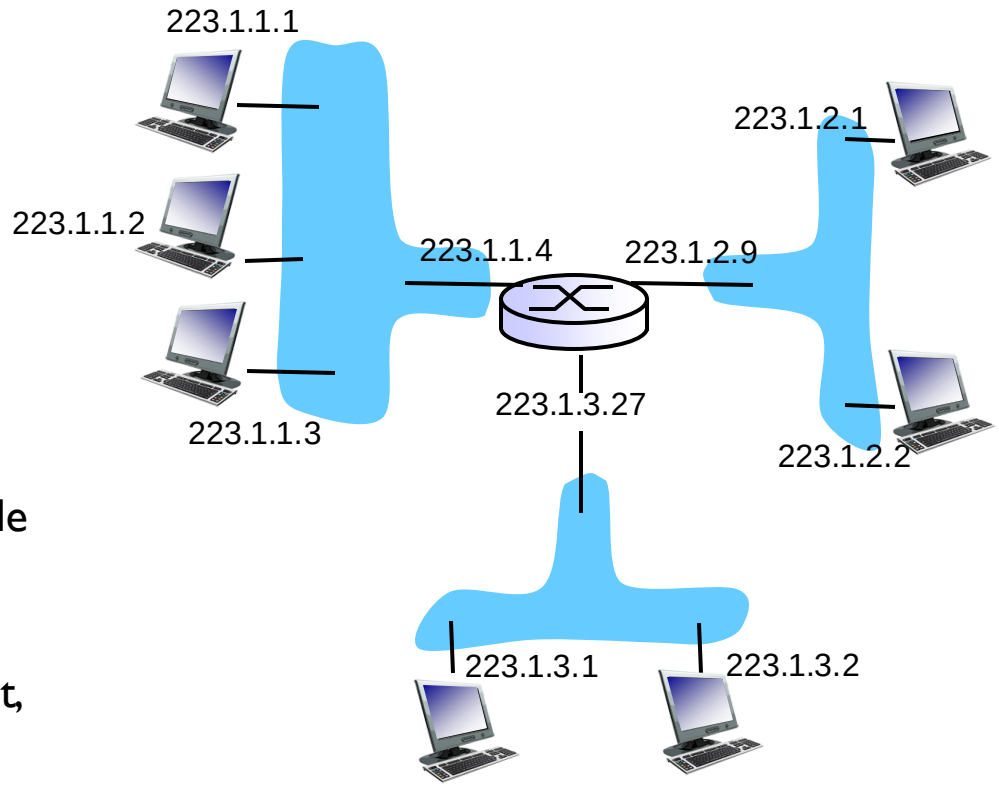
京都大学

IP Addresses

- To send a message from one device to another, each device needs a unique address.
- The internet uses IP addresses to identify devices and route data correctly.
- Each interface has unique 32-bit (IPv4) or 128-bit (IPv6) IP address
  - 32-bit address, expressed as four decimal numbers separated by dots (e.g., 192.168.0.1).
  - 128-bit address, expressed in hexadecimal and separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- IP addresses usually are not permanently tied to a device, i.e., the IP address can change over time.
  - Some servers etc. have static IP addresses but most devices use dynamic IP addresses which are assigned by the network when they are connected and change over time.

IP addressing: introduction

- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*

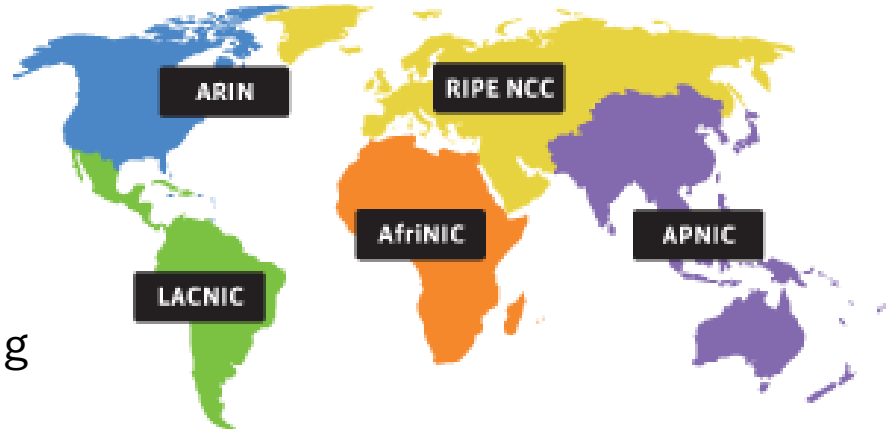


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

京都大学

How to get an IP address



- On the global level, ICANN (Internet Corporation for Assigned Names and Numbers) is responsible for assigning IP Addresses.
- ICANN does not assign individual IP addresses directly but distributes blocks of IP Addresses to 5 Regional Registries which in turn distribute blocks of IP addresses to suborganizations.
- For example, APNIC (Asia-Pacific Network Information Center) is the regional registry for the Asia Pacific Region.
- Under APNIC there are National Internet Registry, e.g., JPNIC (Japan Network Information Center) is responsible for IP addresses in Japan.
- JPNIC acts as a Local Internet Registry within Japan, receiving larger blocks of IP addresses from APNIC and then allocating them to Internet Service Providers, businesses, educational institutions, and government entities in Japan.

How to get an IP address

- Once an organization has obtained a block of IP addresses, it can assign individual IP addresses to hosts and router interfaces.
- IP addresses can be static or dynamic.
- A static address is typically set up manually by the network administrator.
  - Routers often have a fixed IP address that the system administrator configures manually.
- However, most devices receive their IP address automatically using a system called Dynamic Host Configuration Protocol (DHCP)
  - A DHCP server assigns a temporary address to a device whenever it connects to the network.
  - These addresses can change over time.
  - In small and home networks, the DHCP server is usually integrated in the router.

DHCP: Dynamic Host Configuration Protocol

goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- Easy to join and leave a network frequently
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

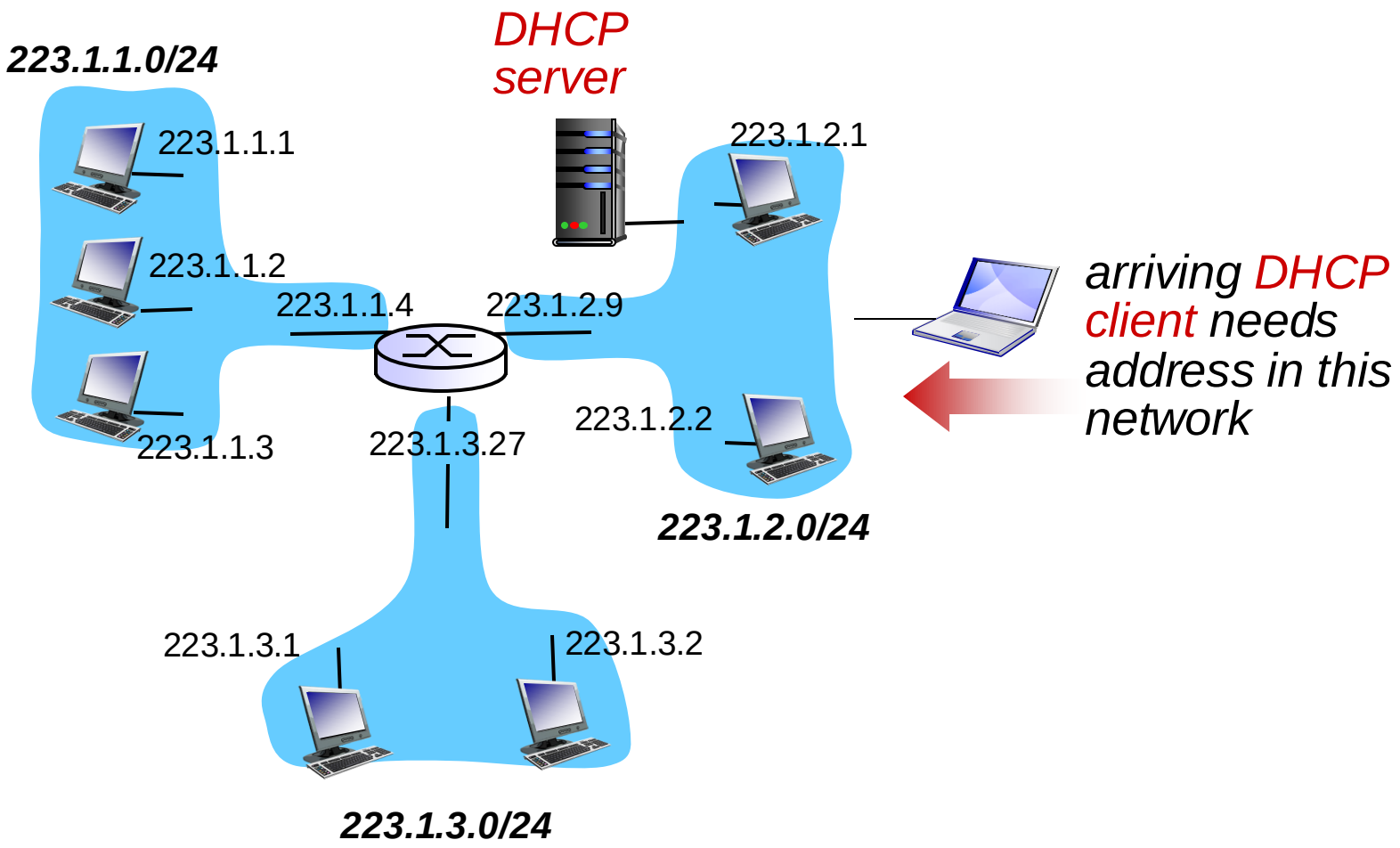
DHCP overview:

- 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

DHCP messages

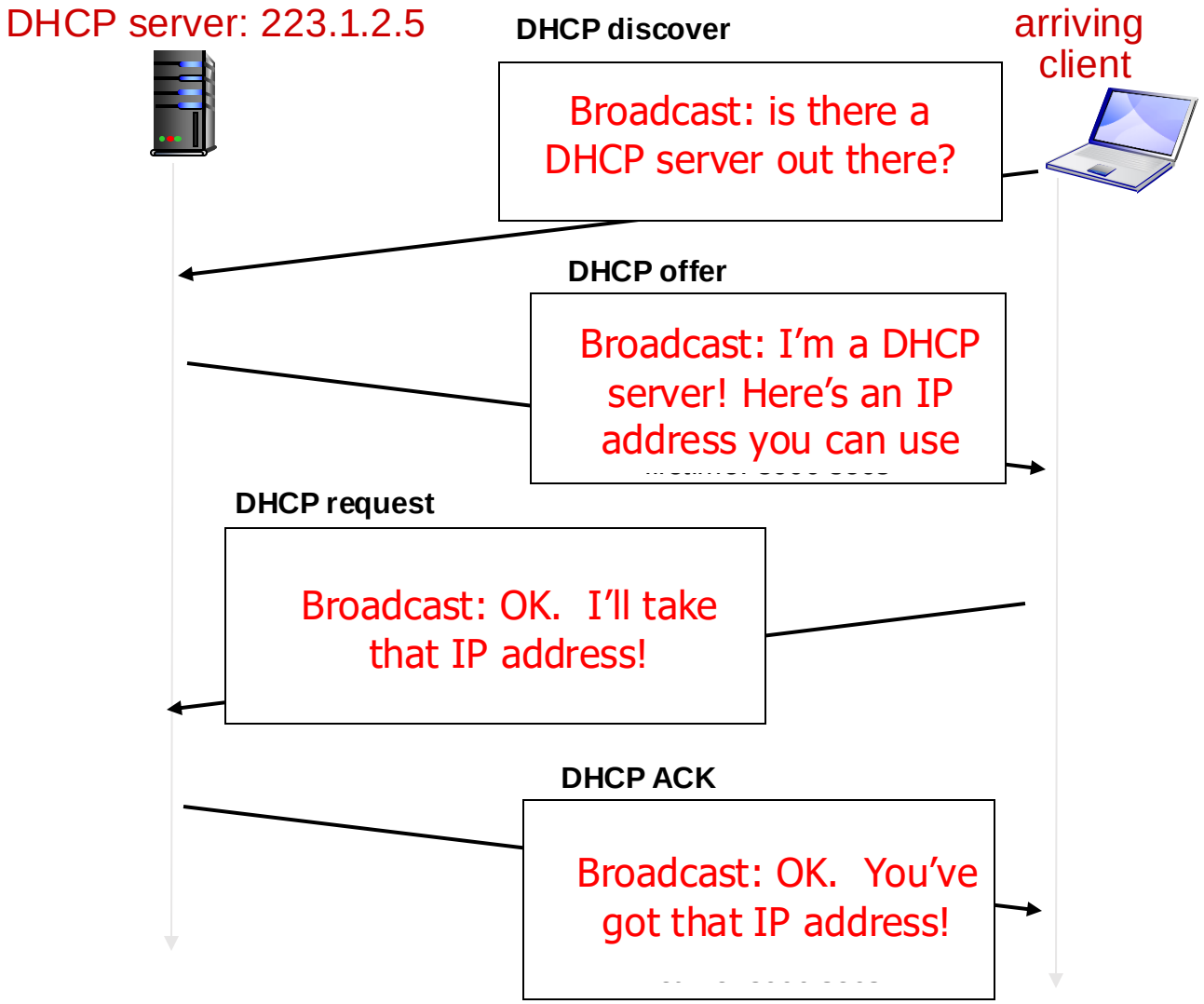
- DHCP server discovery: The client sends an UDP message with destination port 67. This message is broadcasted to all hosts on the subnet by using the broadcasting destination IP address of 255.255.255.255. The sender IP address is set to 0.0.0.0 and the sender port number to 68. The message also contains a transaction ID number to identify the request.
- DHCP server offer: Each DHCP server receiving a discovery message responds with a DHCP offer message that is again broadcasted to all hosts on the subnet. The offer message contains the transaction ID, a proposed IP address for the client, the IP address of the DHCP server and a time how long the IP address will be valid (lifetime).
- DCHP request: The client chooses an IP address among the offers and responds with a DHCP request message. The message contains a copy of the configuration parameters.
- DHCP acknowledgment: The server responds with a DHCP ack message, acknowledging the requested parameters.

DHCP client-server scenario



京都大学

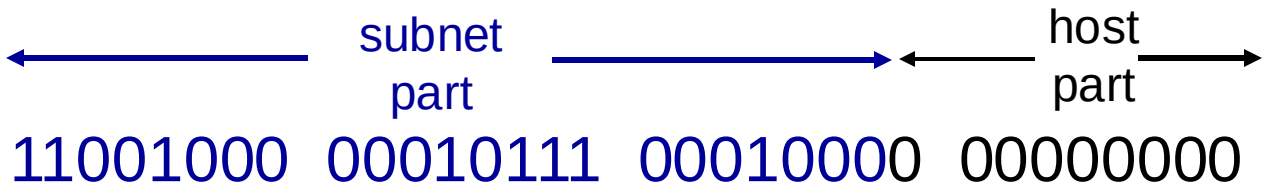
DHCP client-server scenario



IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



200.23.16.0/23

京都大学

IP addresses: how to get one?

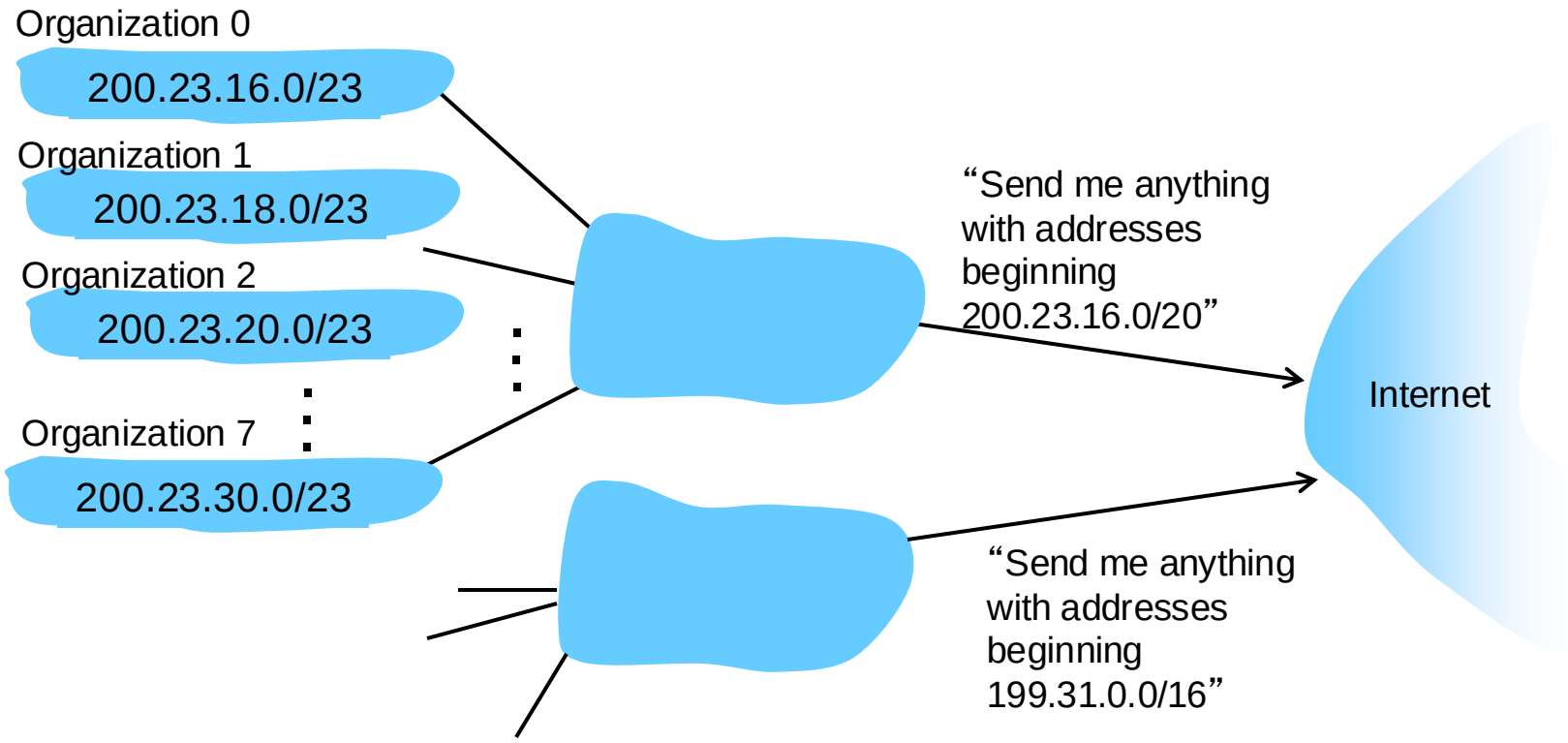
Q: how does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



IPv4 exhaustion

Q: are there enough 32-bit IP addresses?

- 32-bit only allows 4,294,967,296 unique values.
- Furthermore, large blocks are reserved for special use and not available for public allocation.
- ICANN allocated last chunk of IPv4 addresses in 2011.
- All regional registries have by now exhausted their address pools.
- The problem was first recognized in the early 1990.
- IPv6, the successor technology to IPv4, was proposed in 1995 mainly to address this problem.
  - IPv6 uses 128-bit addresses:
    340,282,366,920,938,463,463,374,607,431,768,211,456 ($\sim 3.4 \times 10^{38}$) possible addresses
- However, switching to IPv6 takes a very long time, and is still ongoing 30 years later!
- Some solutions such as NAT (next) have been developed to delay IPv4 address space exhaustion.

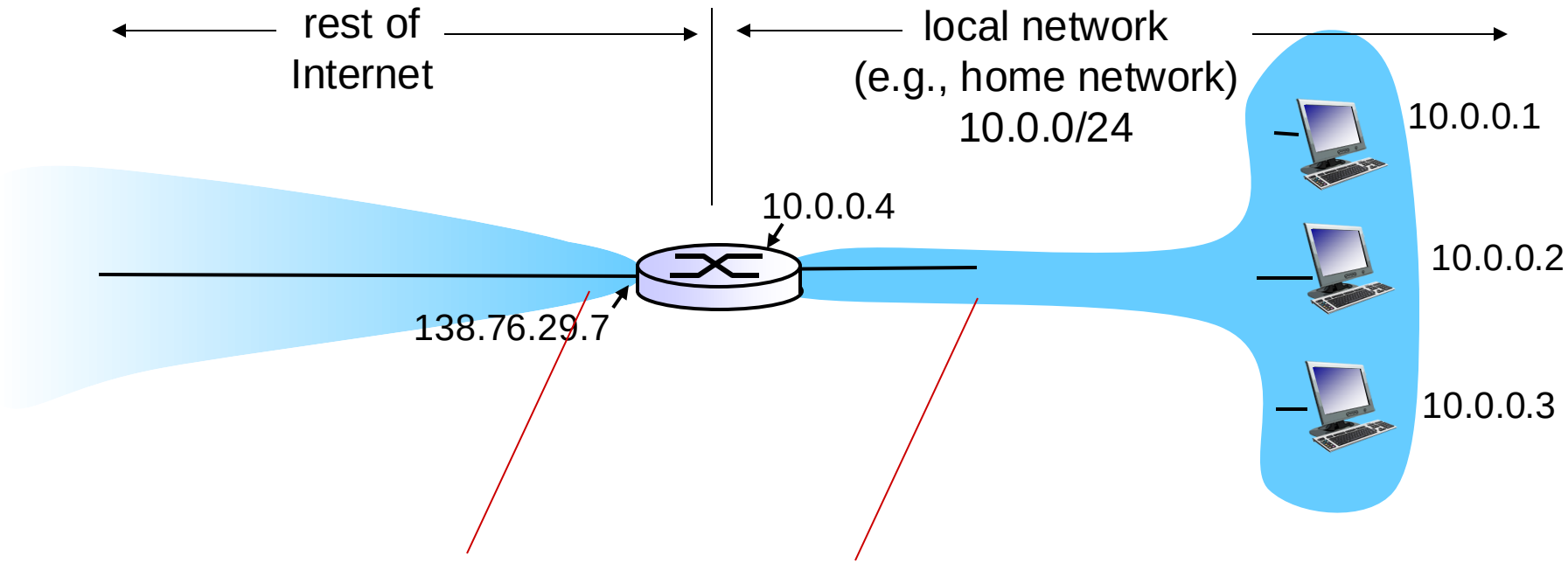
Network Address Translation (NAT)

- Every device needs an IP address.
- Whenever multiple machines need to be connected in a local area network, a range of IP addresses would need to be allocated by the ISP.
- In a typical home or small office network this is not feasible.
- Usually, the router gets assigned only one IP address from the ISP.
- To address multiple interfaces in the network, the router assigns local IP addresses which are only valid within the network.
- When communicating with the outside world local addresses have to be translated to global addresses.
- Network Address Translation (NAT) is a simpler approach to address allocation of IP addresses in a small local area network.

Network Address Translation (NAT)

- NAT is implemented in a NAT-enabled router.
- The NAT-enabled router resides in the home network.
- Inside the home network, local IP addresses starting with 10.0.0 are used for local communication within the private network.
- Local IP addresses only have meaning inside the local network and cannot be used outside.
- The NAT-enabled router looks like a single device for the outside world.
- Thus, there is only one global IP address for the whole local network.
- The NAT-enabled router must translate between local and global addresses.
- For this, the NAT-enabled router uses its own port numbers and a NAT translation table mapping its port numbers to local IP addresses and original port numbers.

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

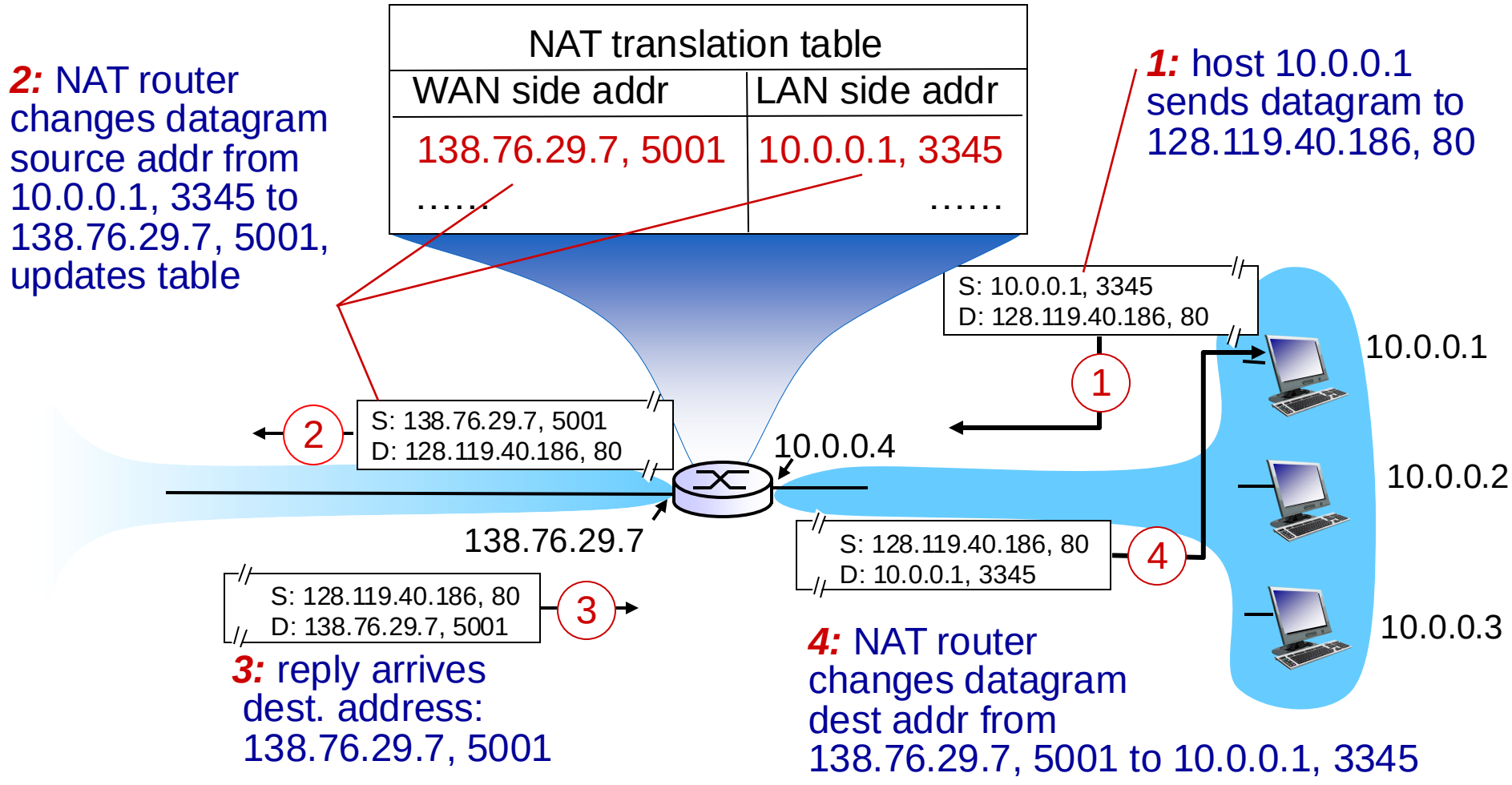
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port number) of every outgoing datagram to (NAT IP address, new port number)
  - . . . remote clients/servers will respond using (NAT IP address, new port number) as destination addr
- *remember (in NAT translation table)* every (source IP address, port number) to (NAT IP address, new port number) translation pair
- *incoming datagrams: replace* (NAT IP address, new port number) in dest fields of every incoming datagram with corresponding (source IP address, port number) stored in NAT table

NAT: network address translation



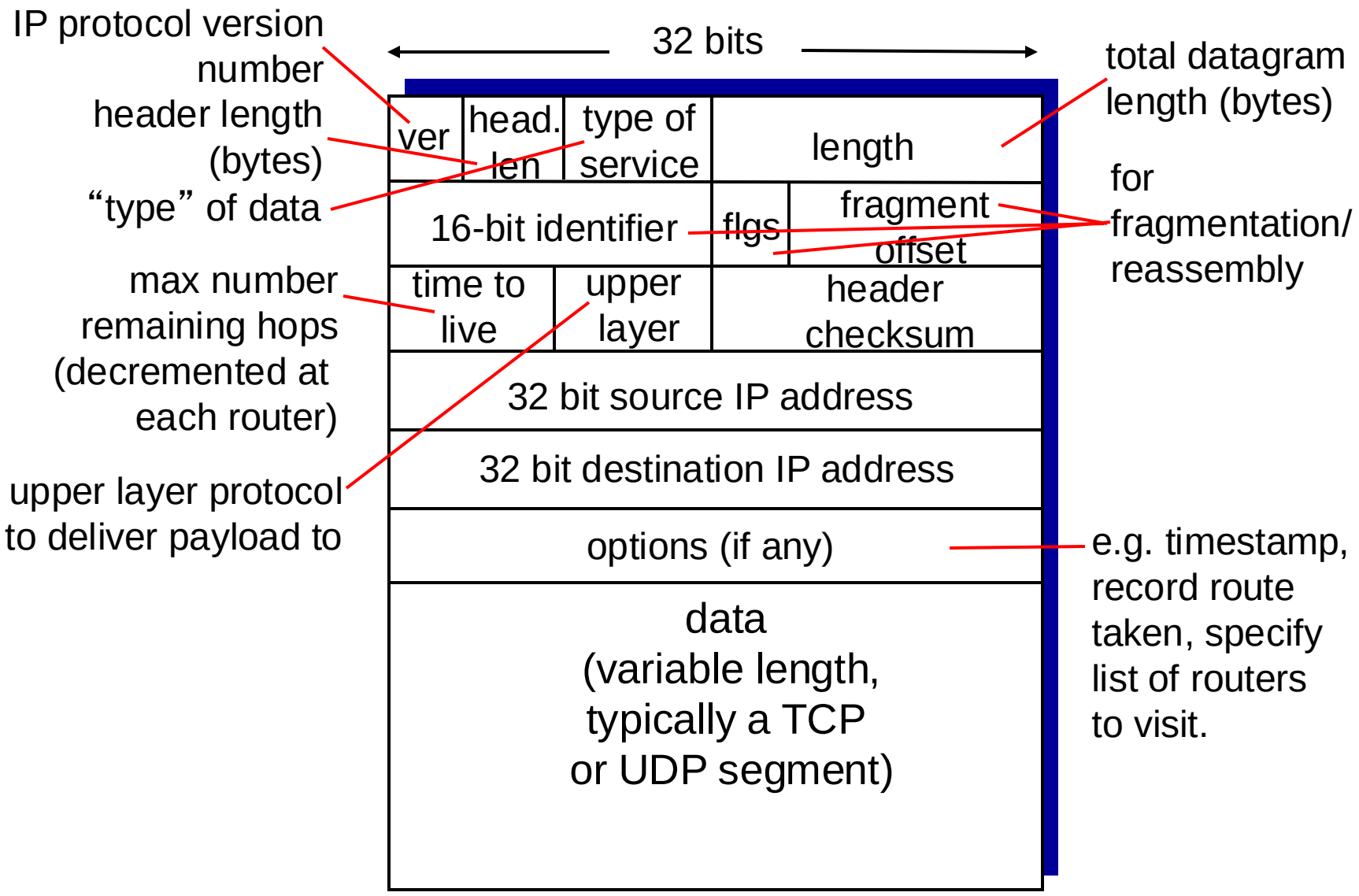
IPv6: motivation

- *initial motivation:* 32-bit address space would be completely allocated (only around 4.3 billion addresses available).
- IPv6 uses 128 bit addresses
- additional motivation:
  - Simpler header format to speed processing/forwarding
  - Built-in security
  - Support for modern network applications

IPv6 datagram format:

  - fixed-length 40 byte header, no additional options

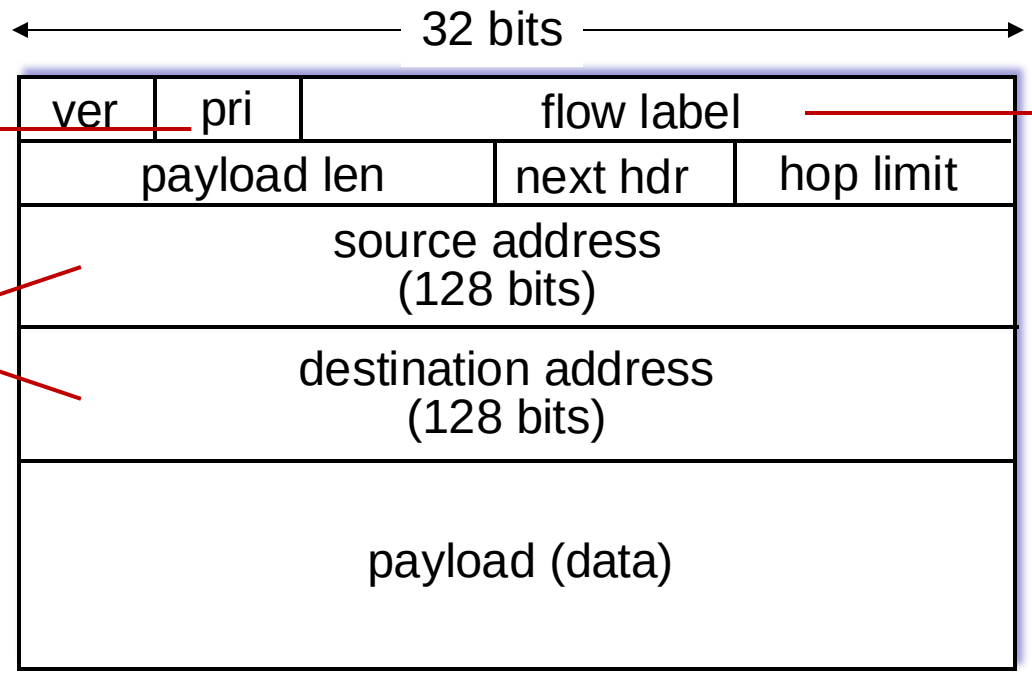
IP datagram format



IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



flow label: identify datagrams in same "flow." (concept of "flow" used for different purposes).

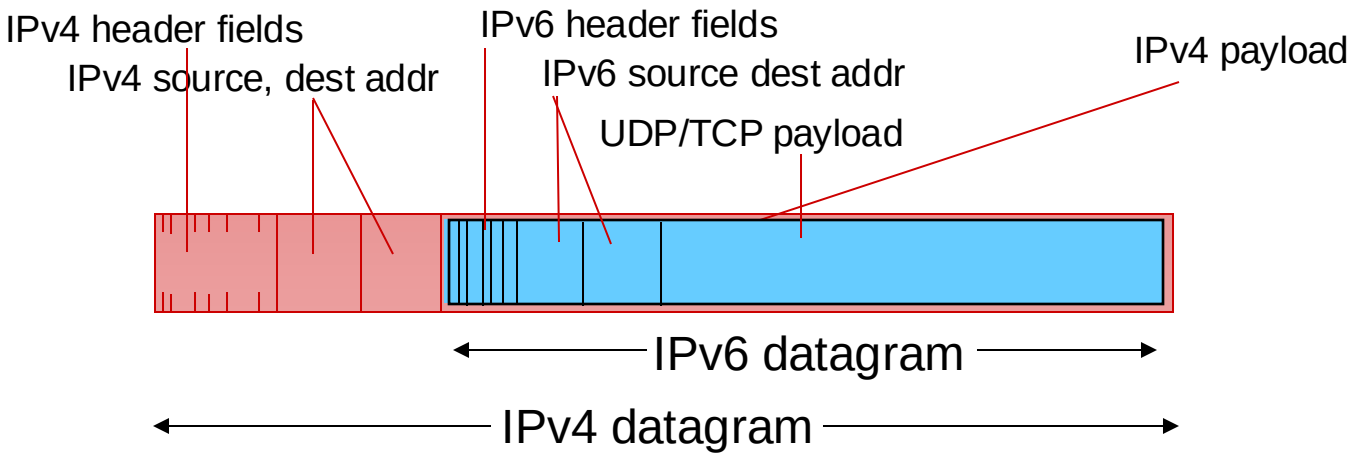
What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

Transition from IPv4 to IPv6

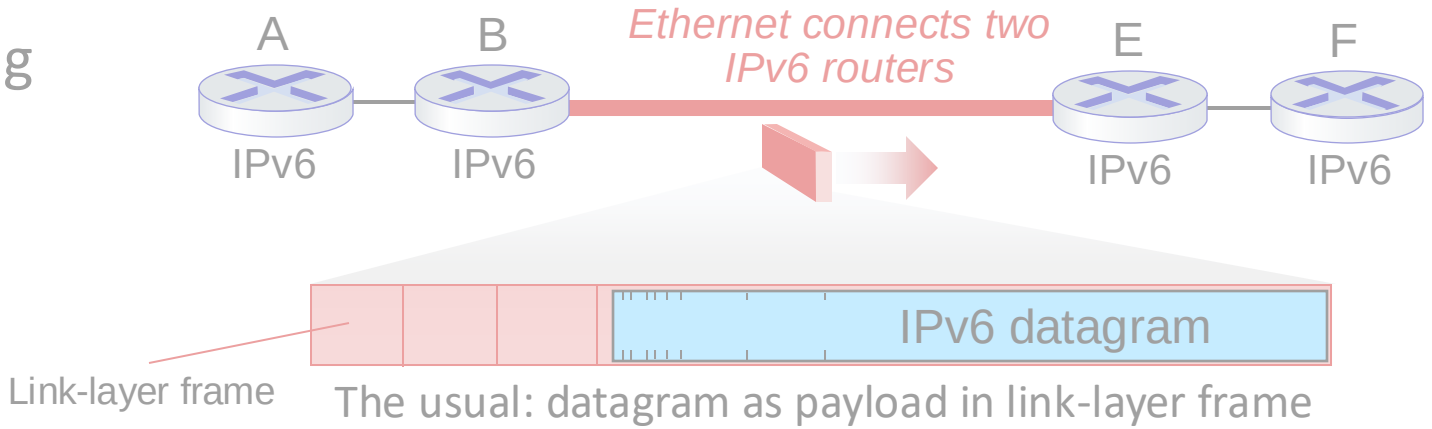
- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?

- tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers (“packet within a packet”)
  - tunneling used extensively in other contexts (4G/5G)

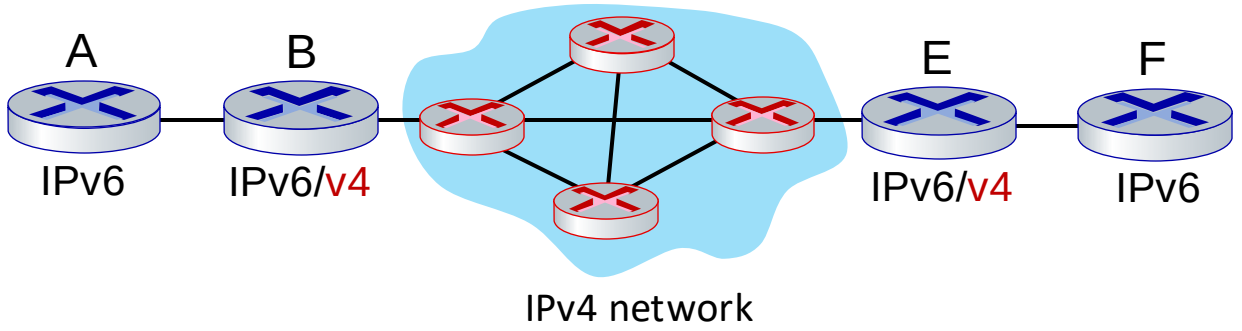


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



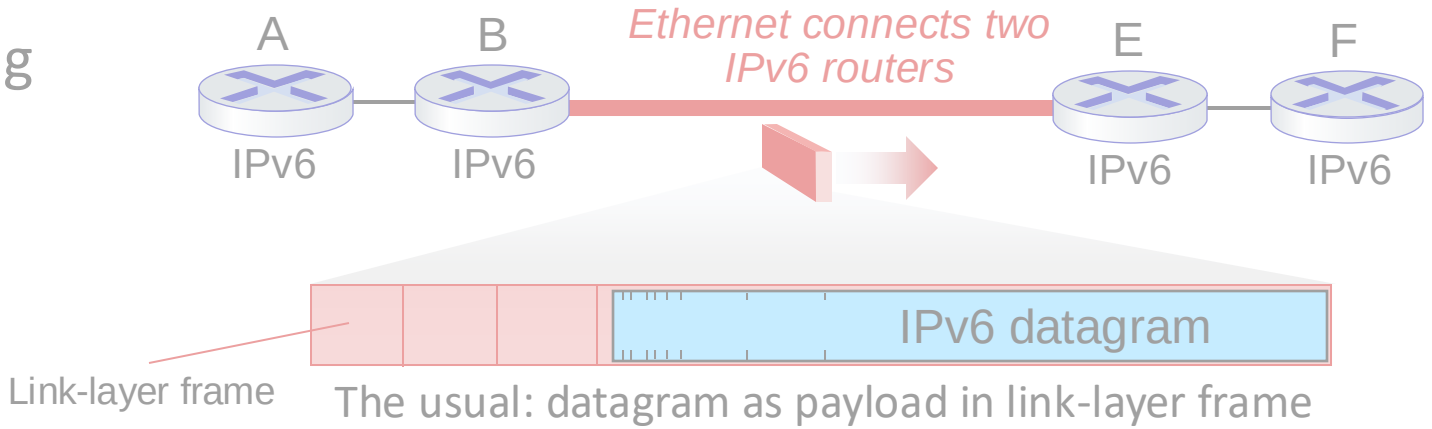
IPv4 network connecting two IPv6 routers



京都大学

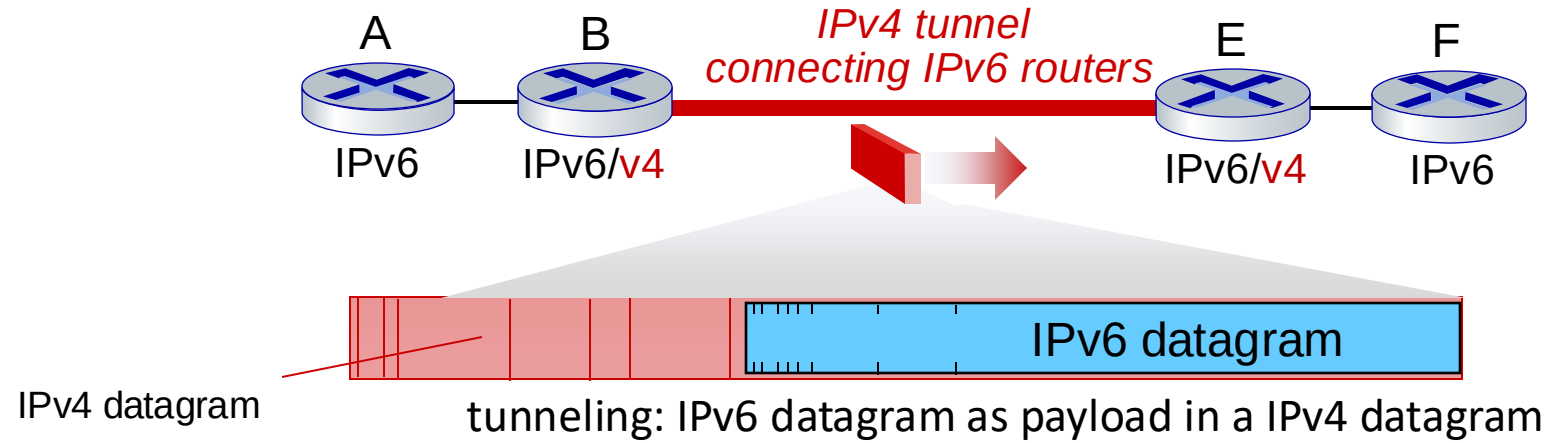
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



The usual: datagram as payload in link-layer frame

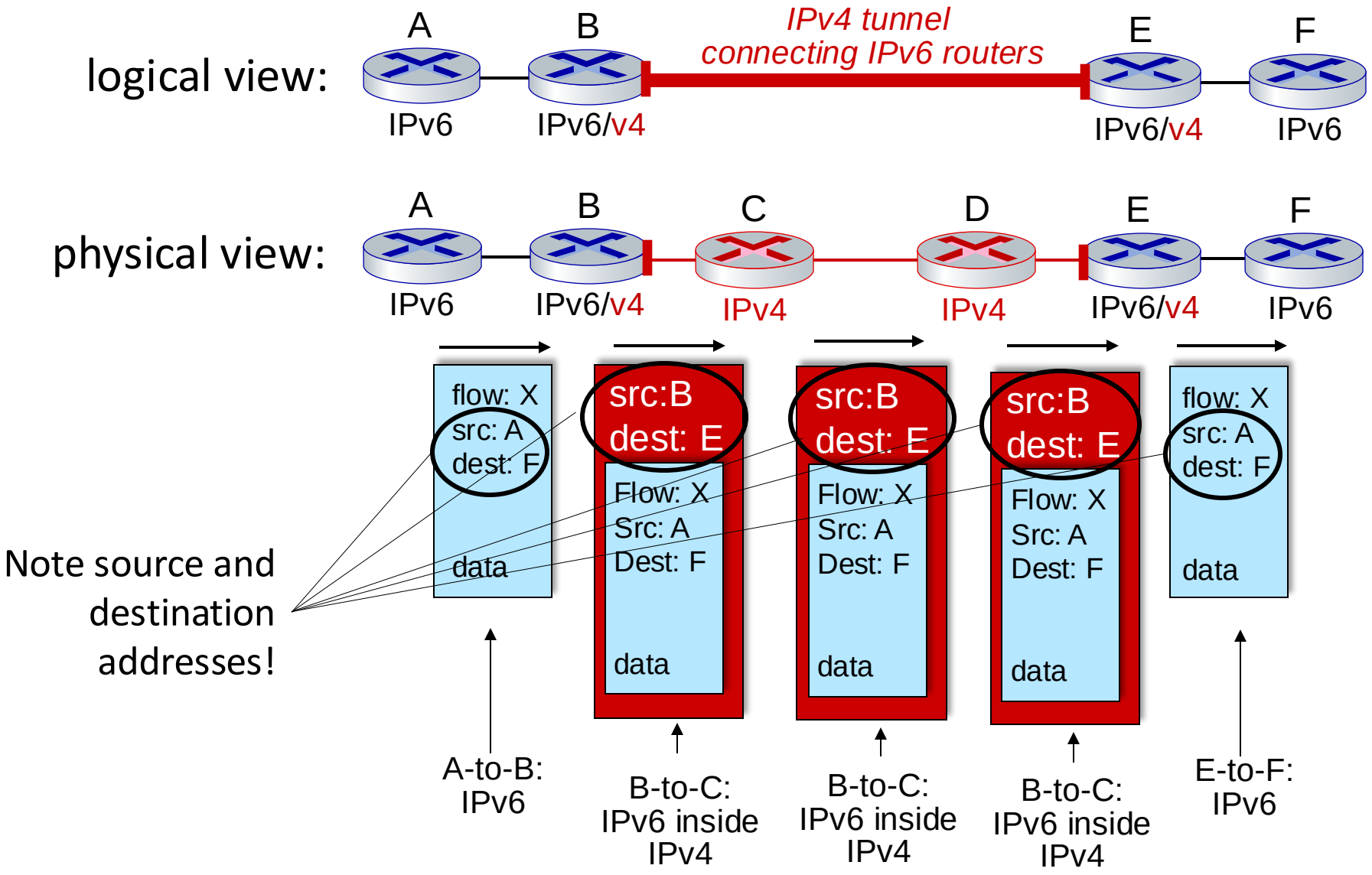
IPv4 tunnel connecting two IPv6 routers



tunneling: IPv6 datagram as payload in a IPv4 datagram

京都大学

Tunneling

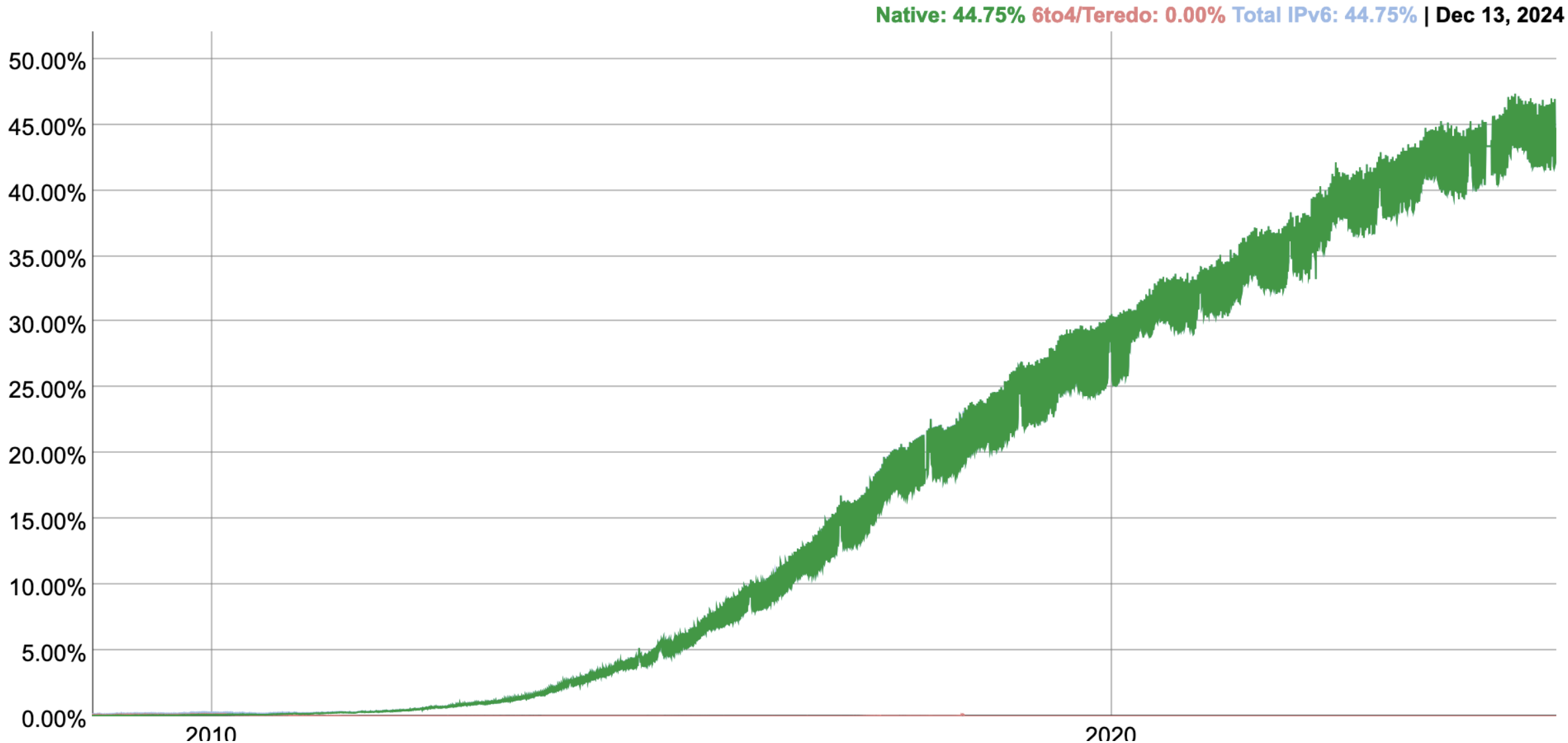


IPv6: adoption

- *Long (long!) time for deployment*
  - Almost 30 years since IPv6 was proposed!
  - Still only around 50% of traffic on the internet uses IPv6 in Dec 2024 (according to google statistics).
  - Think of application-level changes in last 30 years: WWW, Facebook, streaming media, Skype, …

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.



京都大学

Summary of the data plane

- We have now covered the data plane functions of the network layer
- The data plane contains functions that are performed for every router to forward packets arriving on an input link to one of the router's output links.
- We also studied the IPv4 protocol in detail and learned how IP addresses are obtained.
- Next, we will study the network's layer control plane which deals with network-wide logic (routing).

Control plane

5.1 introduction

5.2 routing protocols

- link state
- distance vector

Network-layer functions

Recall: two 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

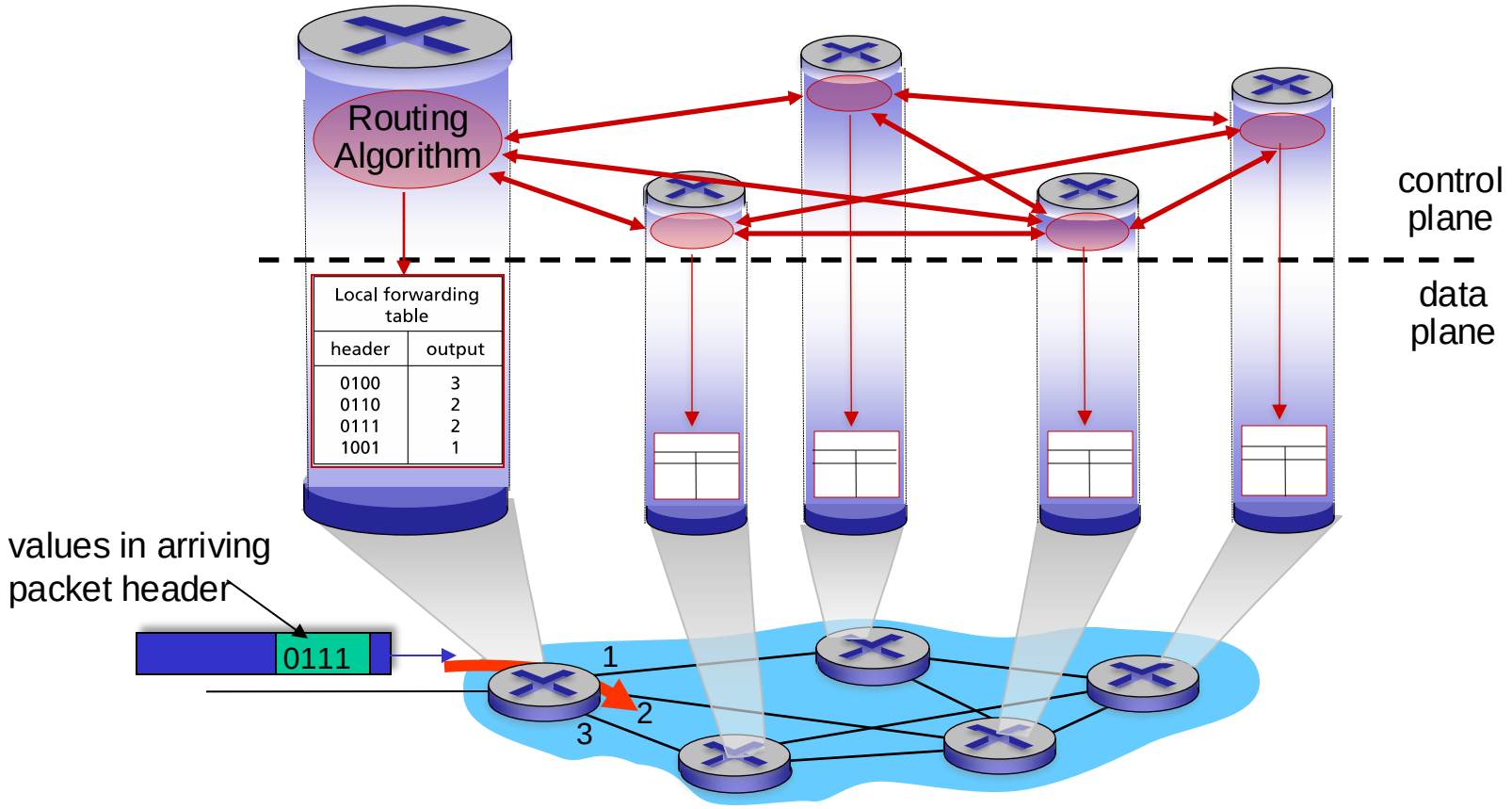
control plane

Two approaches to structuring network control plane:

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

Per-router control plane

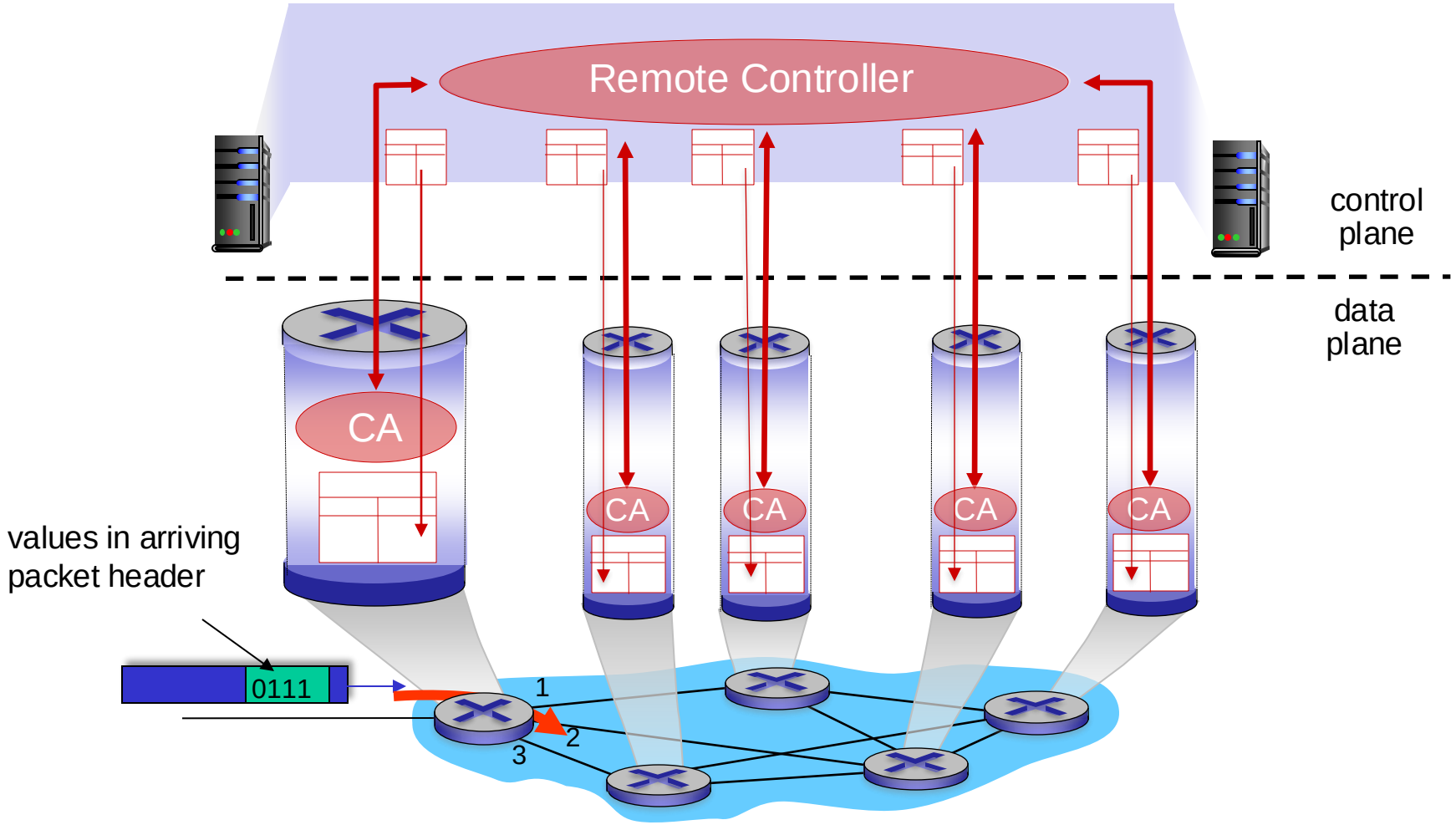
Individual routing algorithm components *in each and every router* interact in the control plane



京都大学

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

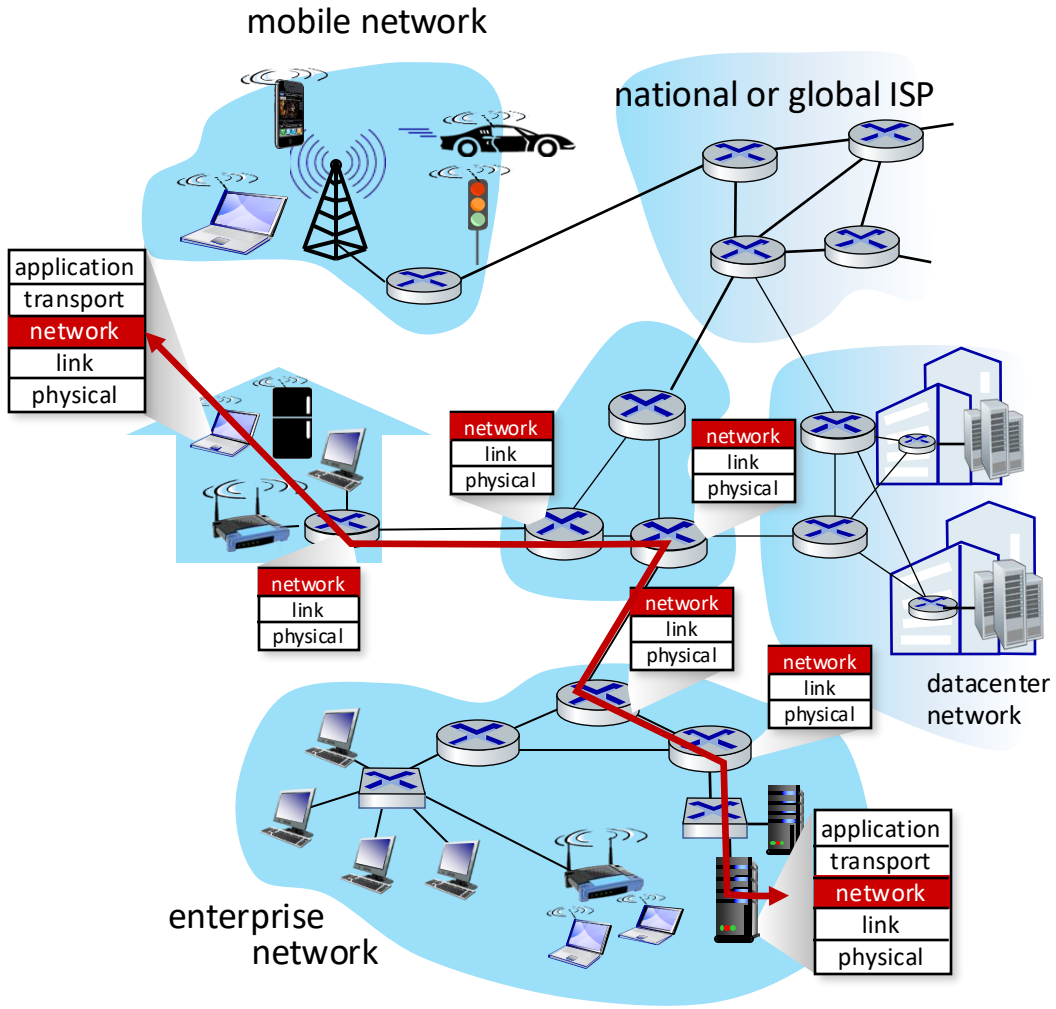


京都大学

Routing protocols

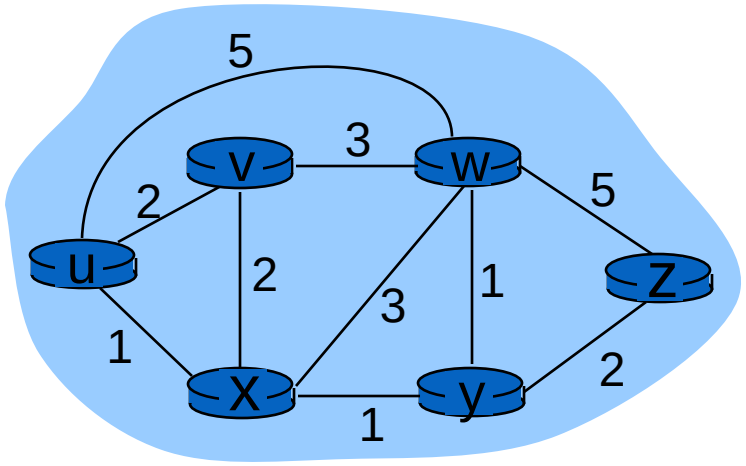
Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets traverse from given initial source host to final destination host
- "good": least "cost", "fastest", "least congested"



京都大学

Graph abstraction of the network

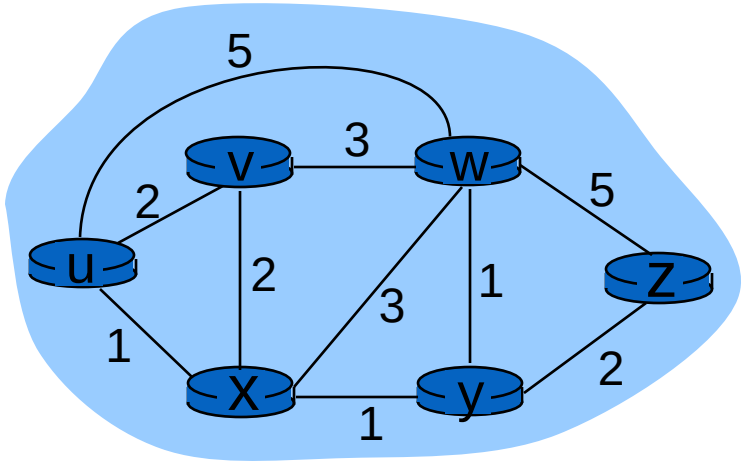


graph: G = (N,E)

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) }

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

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
Sometimes least-cost path is also called **shortest path**.

Classification of routing algorithms

Centralized routing algorithm:

- Computes the least-cost path between source and destination using complete, global knowledge about the network.
- The algorithm knows all the nodes and edges of the graphs, and each of the edge costs.
- Often called link-state algorithm because the algorithm must be aware of the costs of each of the links in the network.

Classification of routing algorithms

Decentralized routing algorithm:

- Every router only has information about its neighbors.
- Can exchange information with neighbors.
- Calculation done in an iterative, distributed manner.
- Each router starts with only information about the cost to its direct neighbors.
- By receiving information from its neighbors, the router gradually calculates the least-cost path to other destinations.
- We will study the distance vector algorithm.

Classification of routing algorithms

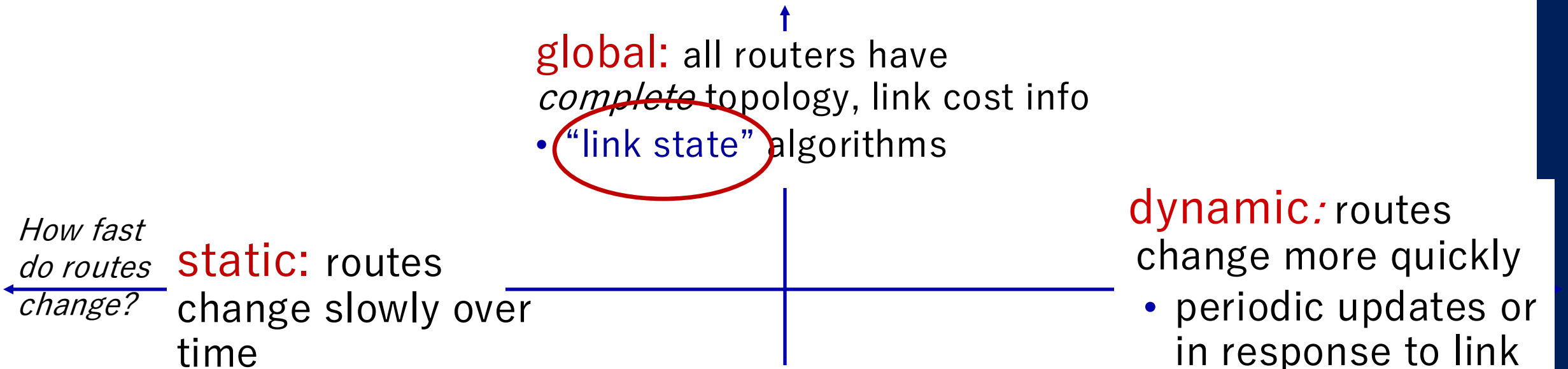
Static routing algorithms:

- Routes usually do not change or only change very slowly over time.
- Routing tables are only updated if requested manually.

Dynamic routing algorithms:

- Routes can change dynamically, for example to adapt to traffic or changes in the network.
- More responsive to network changes but also more susceptible to problems and needs more complex algorithm.

Routing algorithm classification



global: all routers have *complete* topology, link cost info

- "link state" algorithms

How fast do routes change?

static: routes change slowly over time

dynamic: routes change more quickly

- periodic updates or in response to link cost changes

decentralized: iterative process of computation, exchange of info with neighbors

- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

京都大学

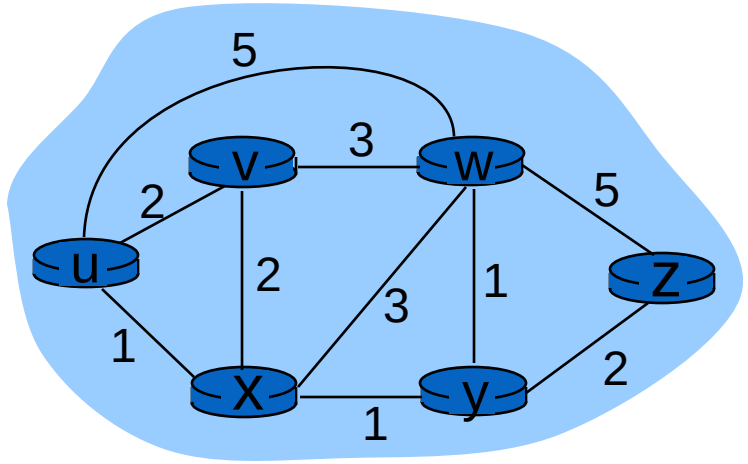
Today's lecture

5.1 introduction

5.2 routing protocols

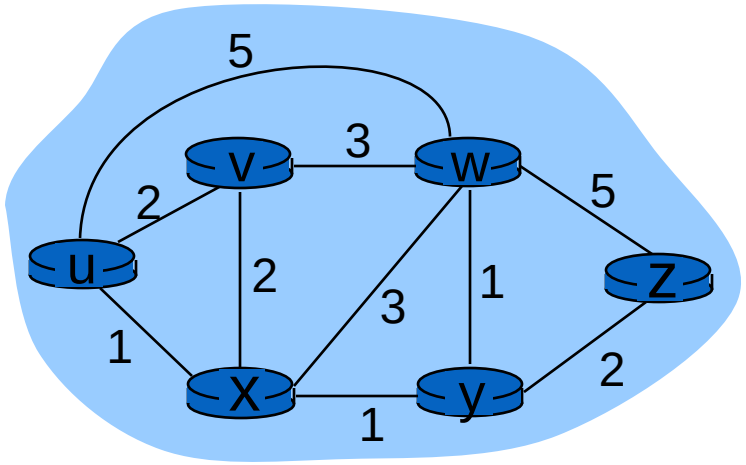
- link state
- distance vector

Link state algorithm



- In a link state algorithm, the algorithm knows the network topology and all link costs.
- In practice, each node has to broadcast link-state packets to all other nodes in the network.
- A link-state packet contains the identities of the node's neighbors and the costs to each neighbor.
- After receiving the packets, all nodes have an identical and complete view of the network.
- Each node can then run the link-state algorithm to find the least-cost path to any other node.

Dijkstra's algorithm



- Dijkstra's algorithm computes the least cost to go from a source node s to each of the other nodes.
- For example, if the source node is u, it computes the table below:

destination	cost	previous node
v	2	u
x	1	u
y	2	x
w	3	y
z	4	y

A link-state routing algorithm

Dijkstra's algorithm

- 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

Today's lecture

5.1 introduction

5.2 routing protocols

- link state
- distance vector

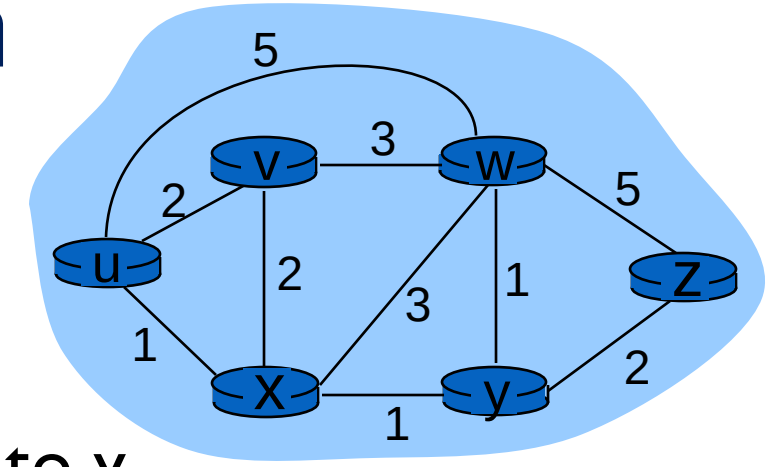
Distance vector algorithm

- The Link-state algorithm uses global information.
- We will next discuss the distance vector algorithm which is **iterative**, **asynchronous** and **distributed**.
- Distributed: Each node receives information from its neighbors and then distributes it back to its neighbors.
- Iterative: The process continues until no more new information is available.
- Asynchronous: not all nodes need to operate at the same time.

Distance vector algorithm

- Each router has knowledge about the network and shares it through the entire network.
- Each router sends its knowledge to its direct neighbors.
- The router uses the information it gets from its neighbors to update its own routing table.
- The information is shared with the neighbors at regular interval, for example, every 30 seconds.

Distance vector algorithm



Bellman-Ford equation

let

$d_x(y)$:= cost of least-cost path from x to y

then

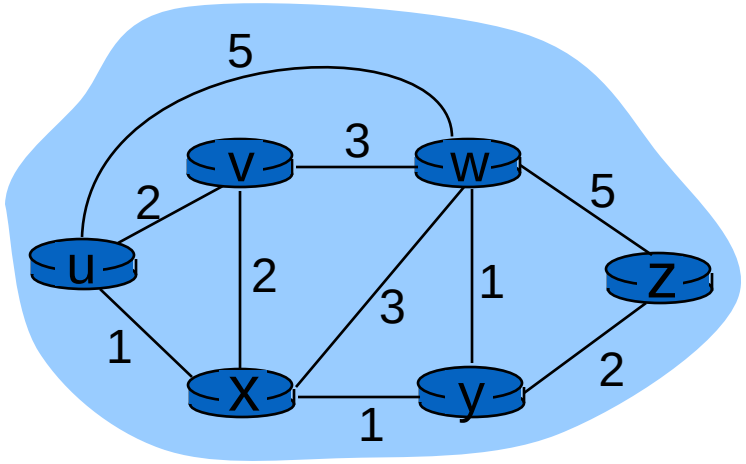
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F 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 is next hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: c(x,v)
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains

    $\mathbf{D}_v = [D_v(y): y \in N]$

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 neighbor, it updates its own DV using B-F equation:

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

❖ The estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$ (under some minor natural conditions)

Distance vector algorithm

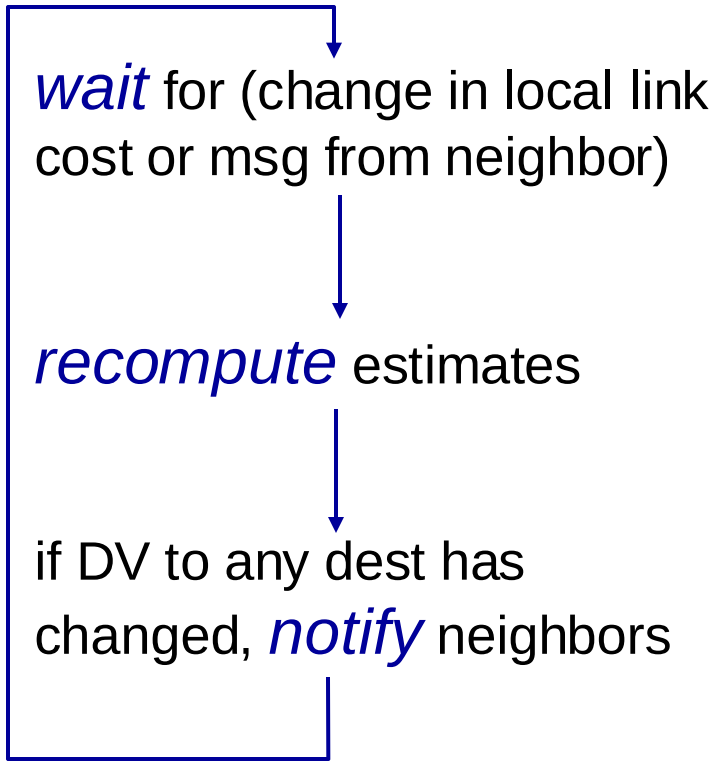
iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

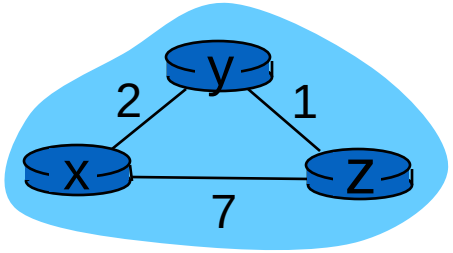
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

京都大学

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

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

node x table

from \ 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 \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

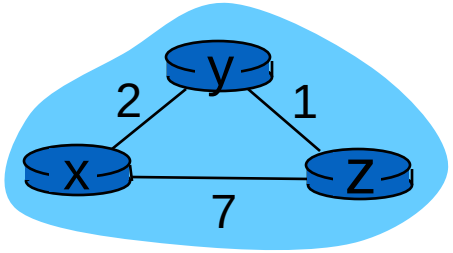
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

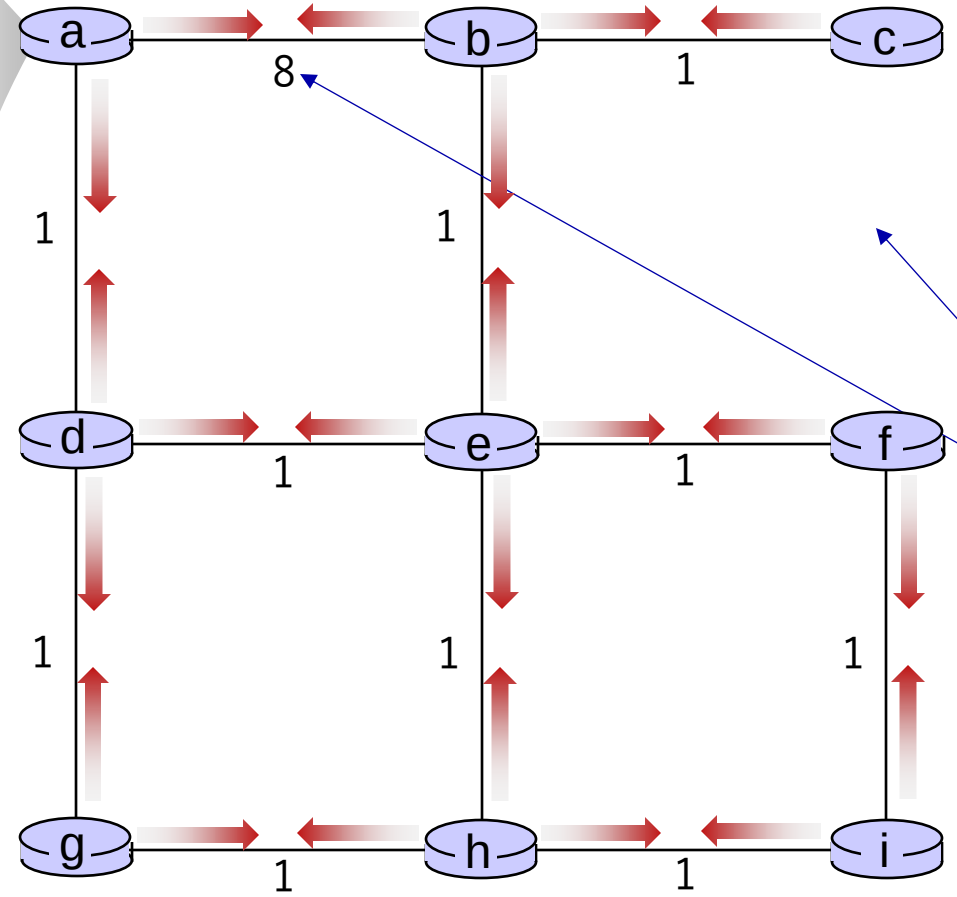
京都大学

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

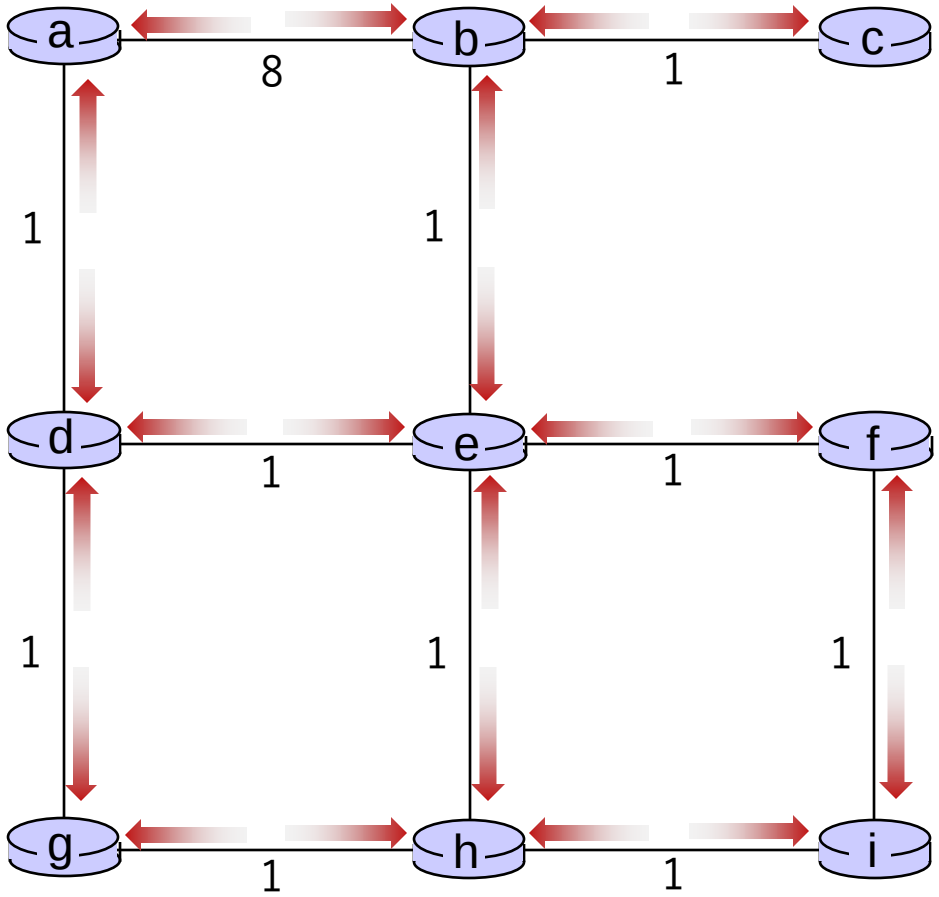
京都大学

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



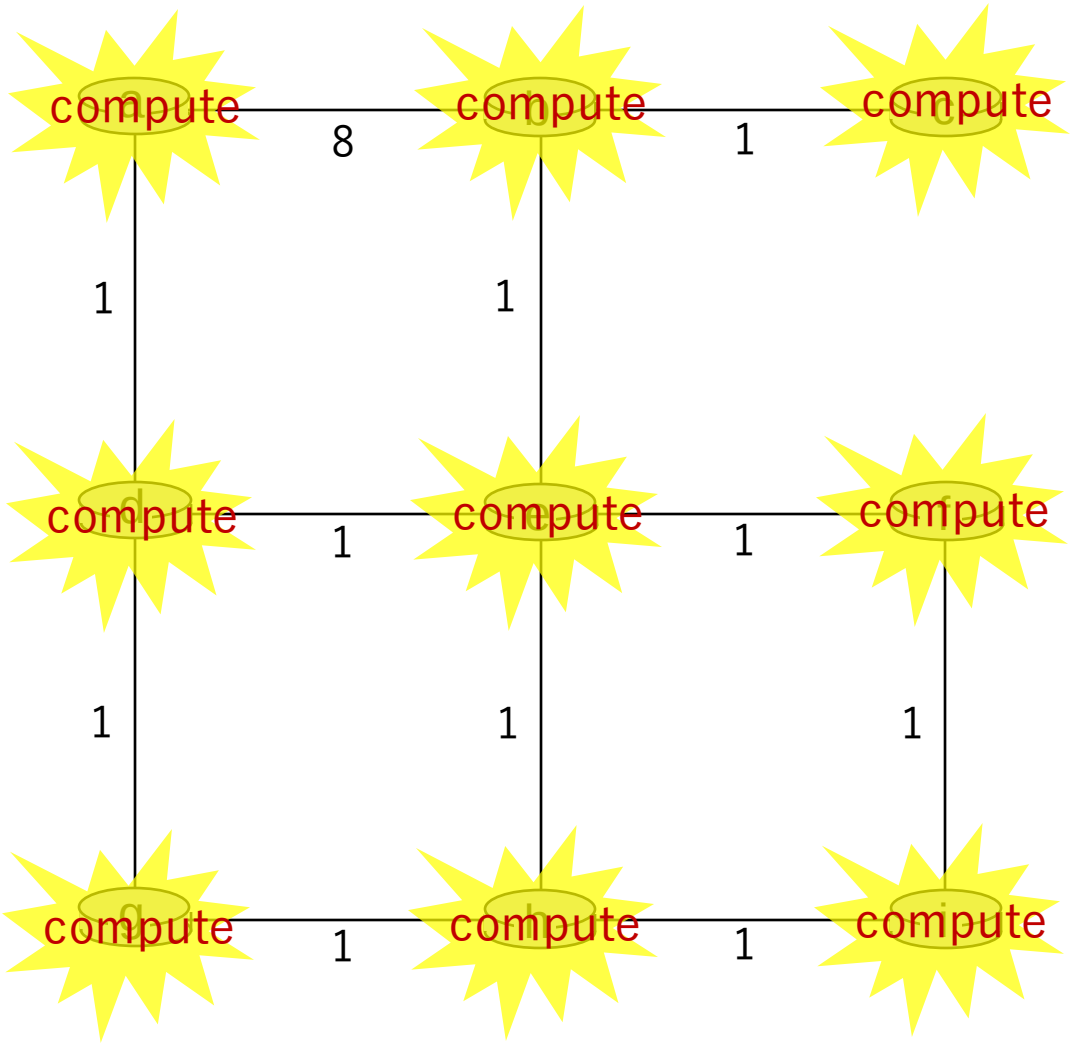
京都大学

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



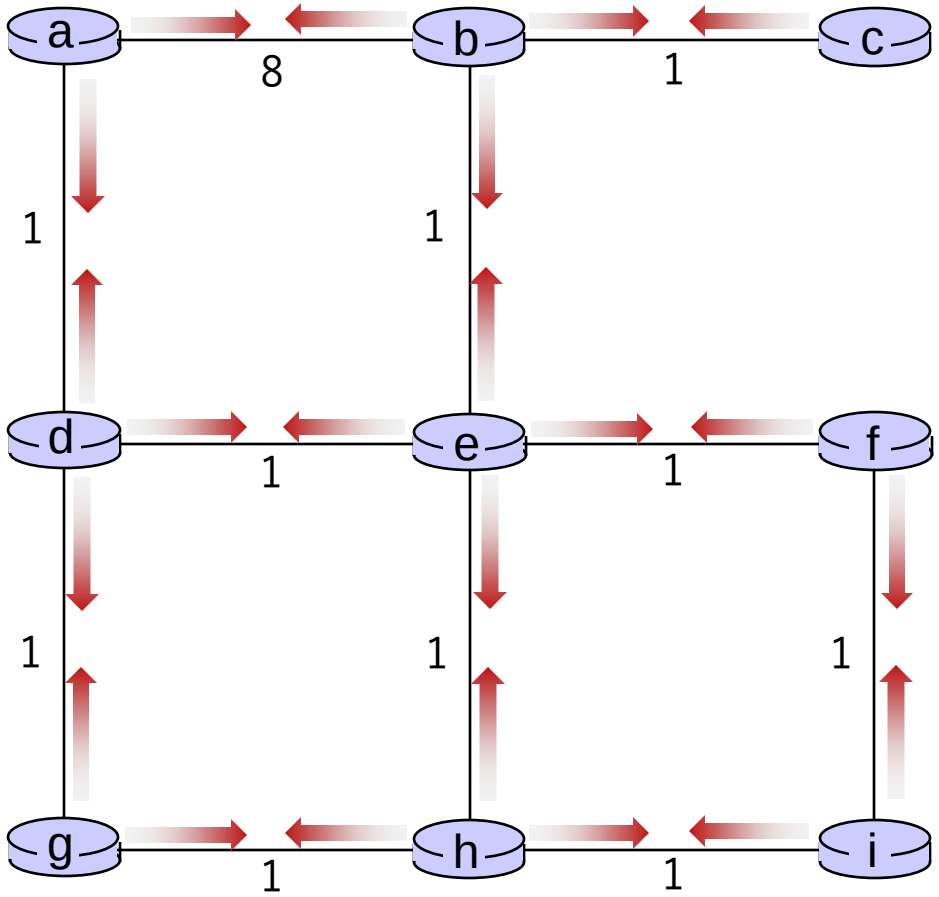
京都大学

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



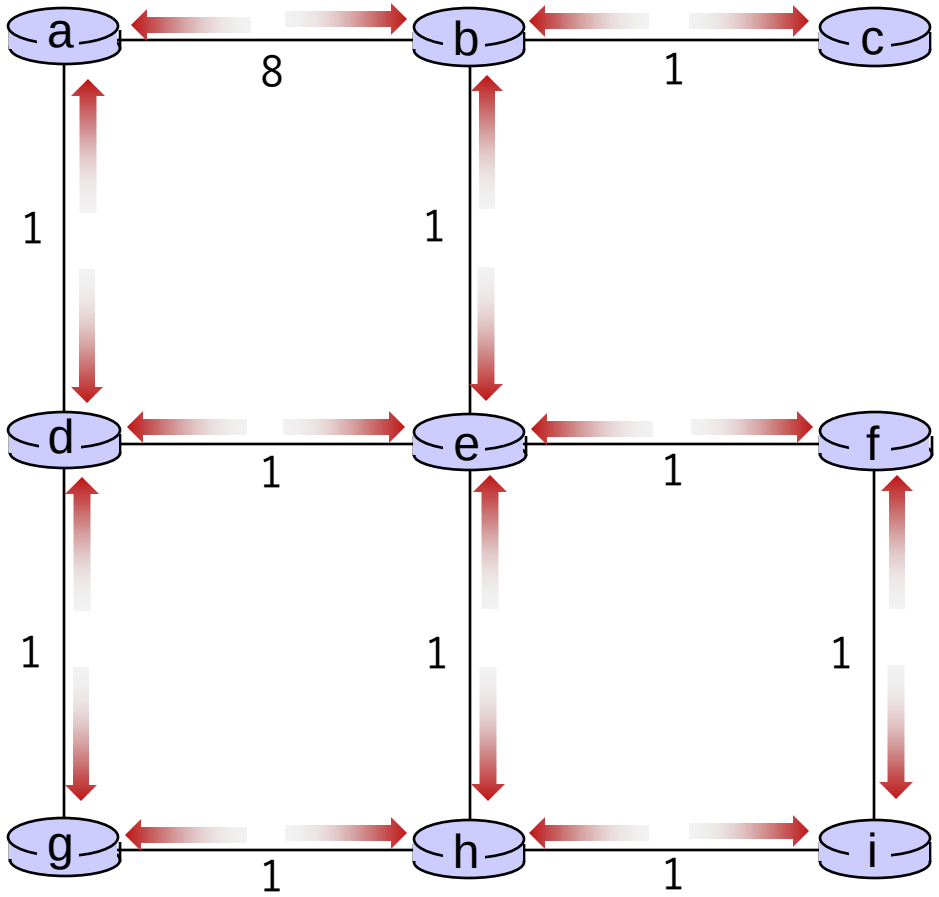
京都大学

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



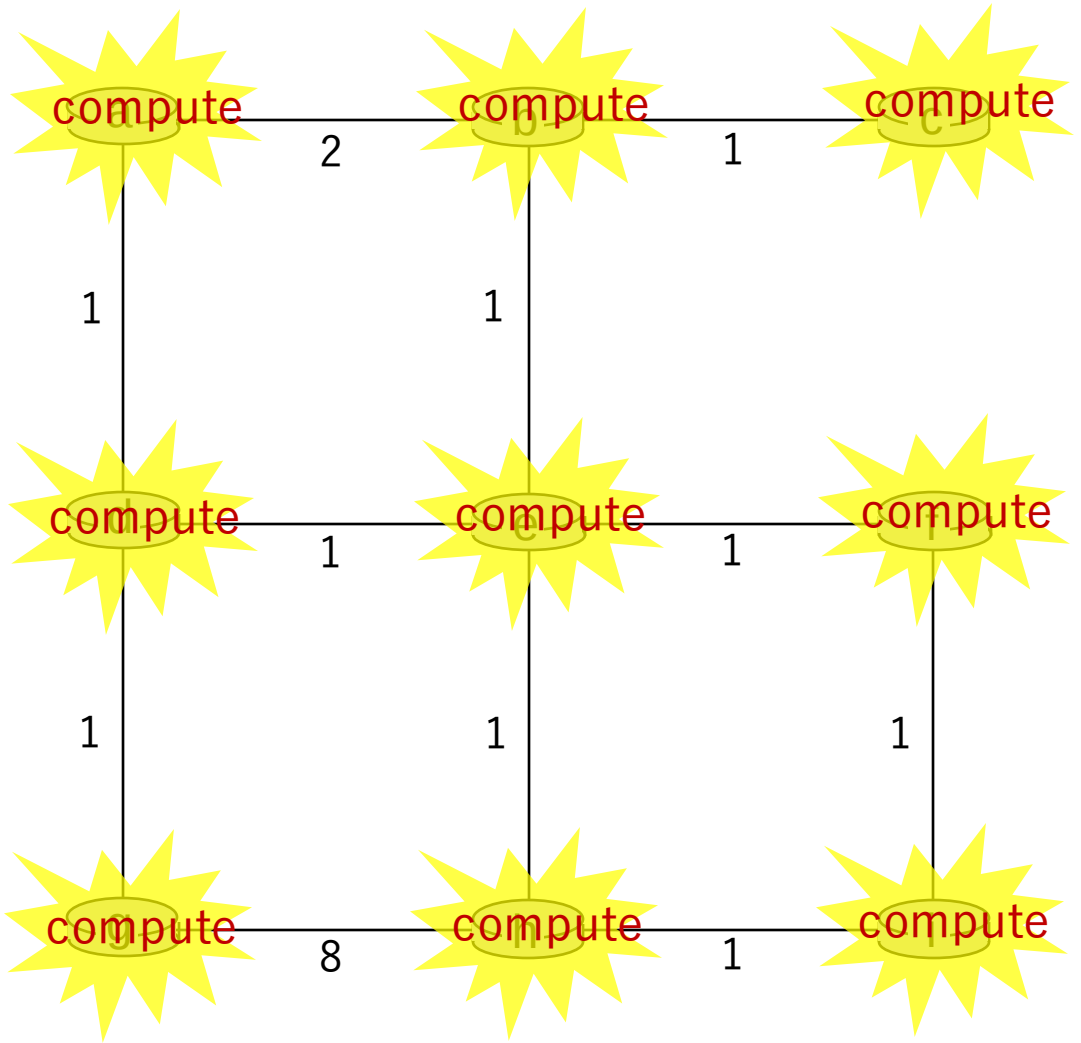
京都大学

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



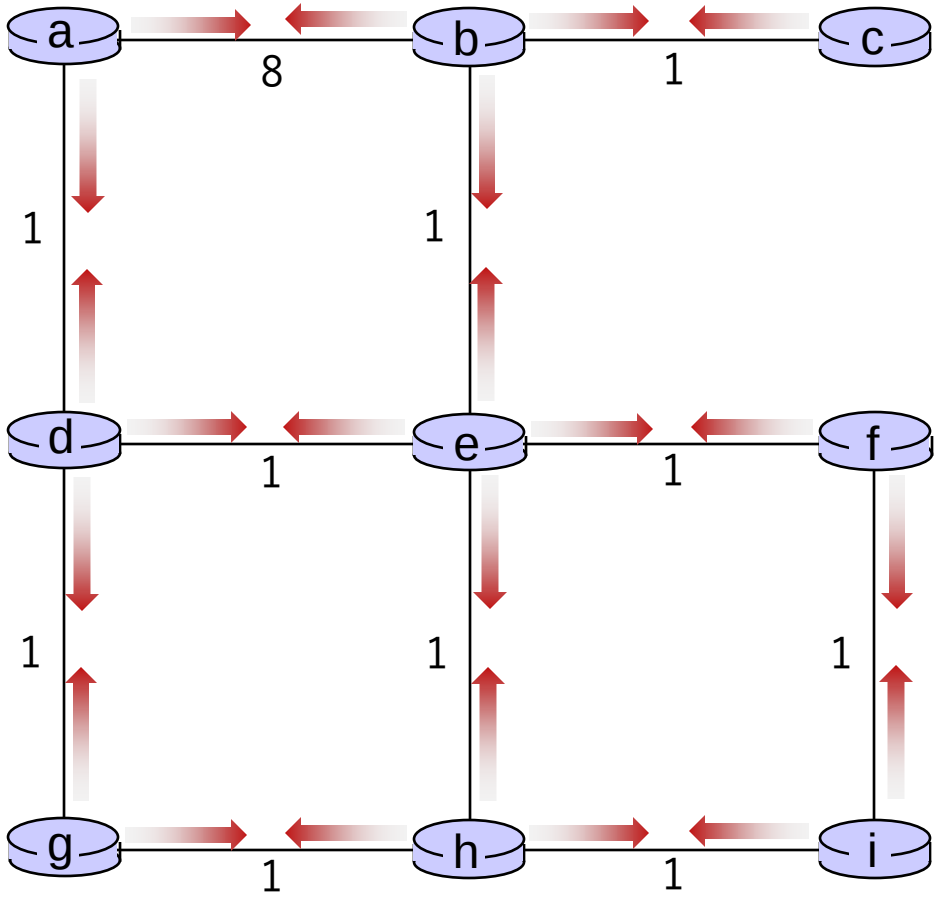
京都大学

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



京都大学

Distance vector example: iteration

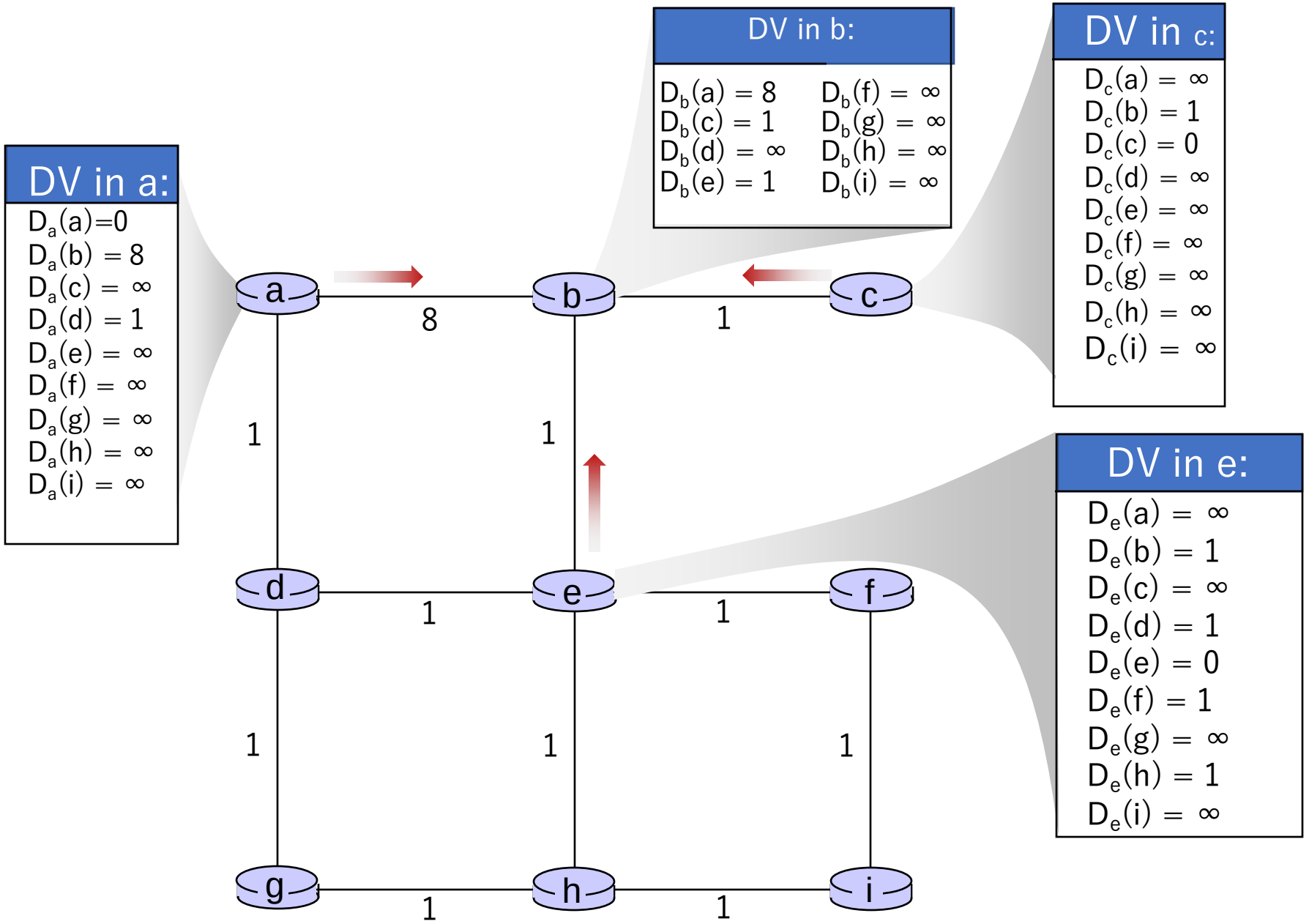
…. and so on

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

京都大学

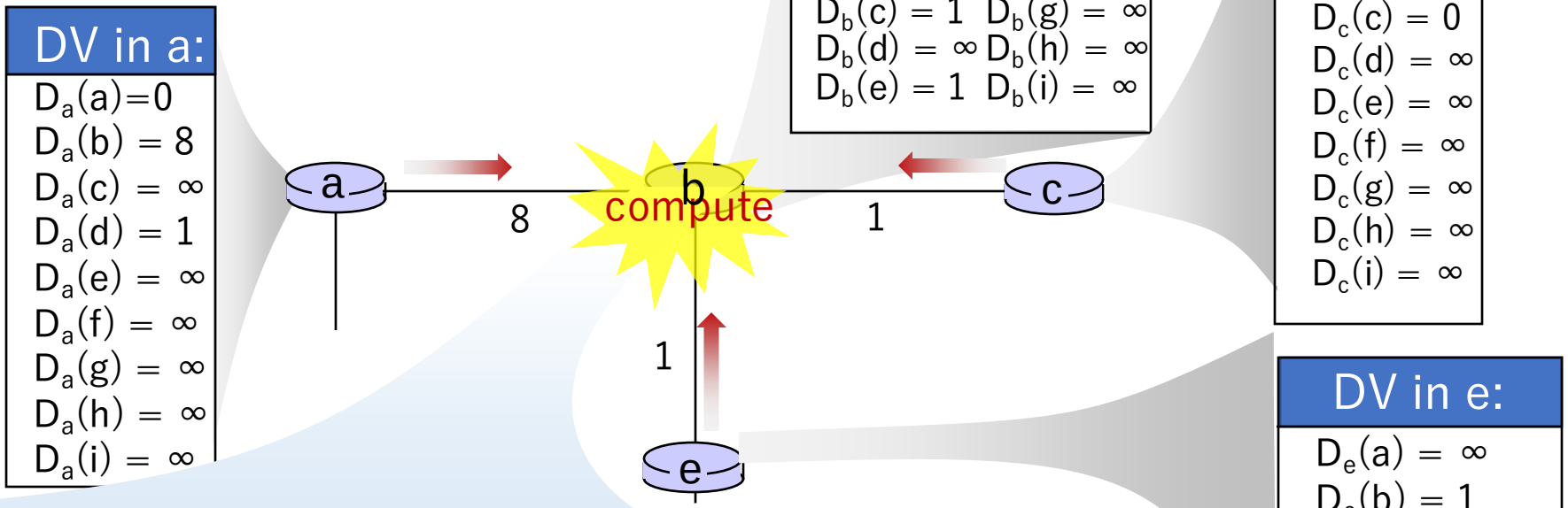
t=1

- b receives DVs from a, c, e



t=1

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



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

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

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

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

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

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

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

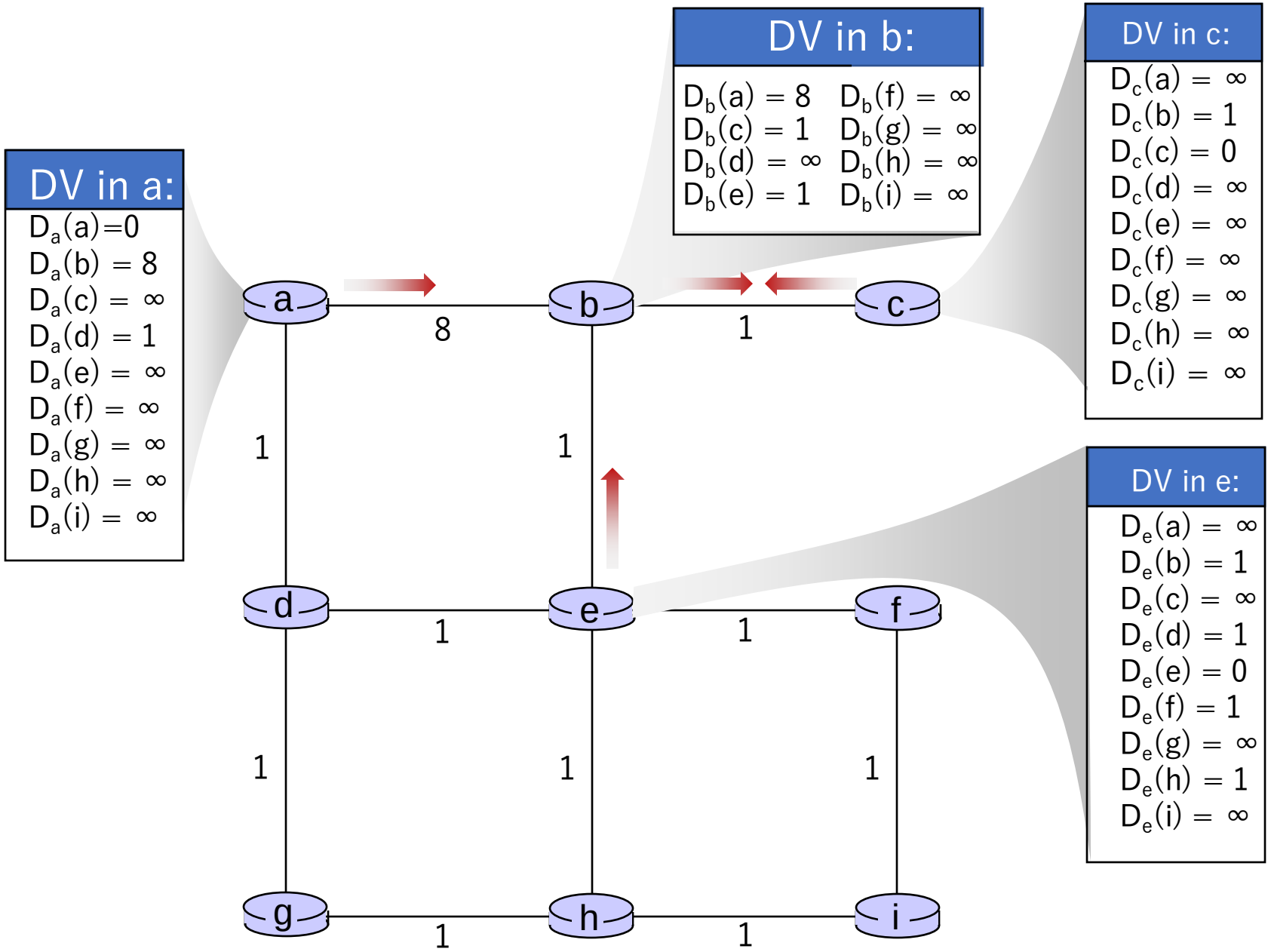
$D_b(i) = \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$

DV in b:

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

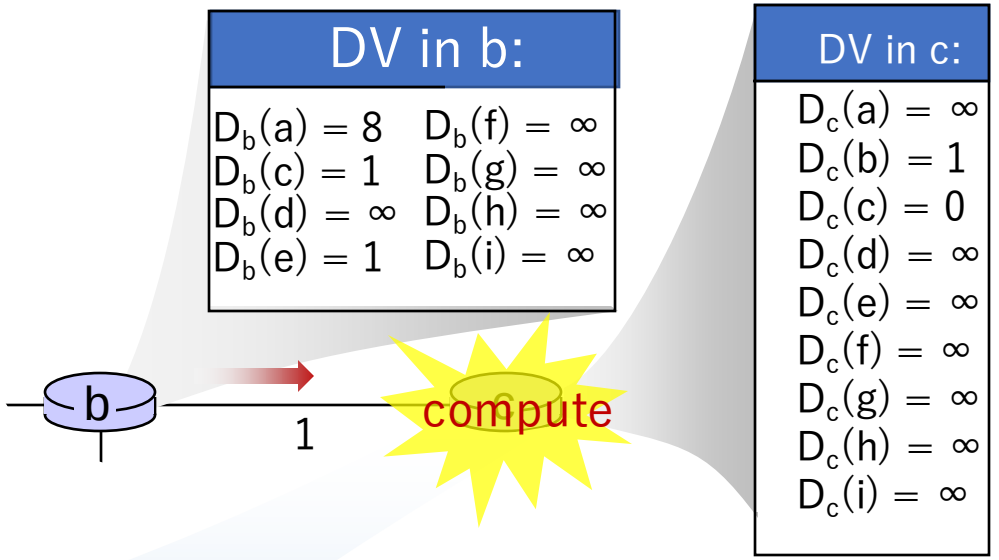
t=1

- c receives DVs from b



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:

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

京都大学

Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

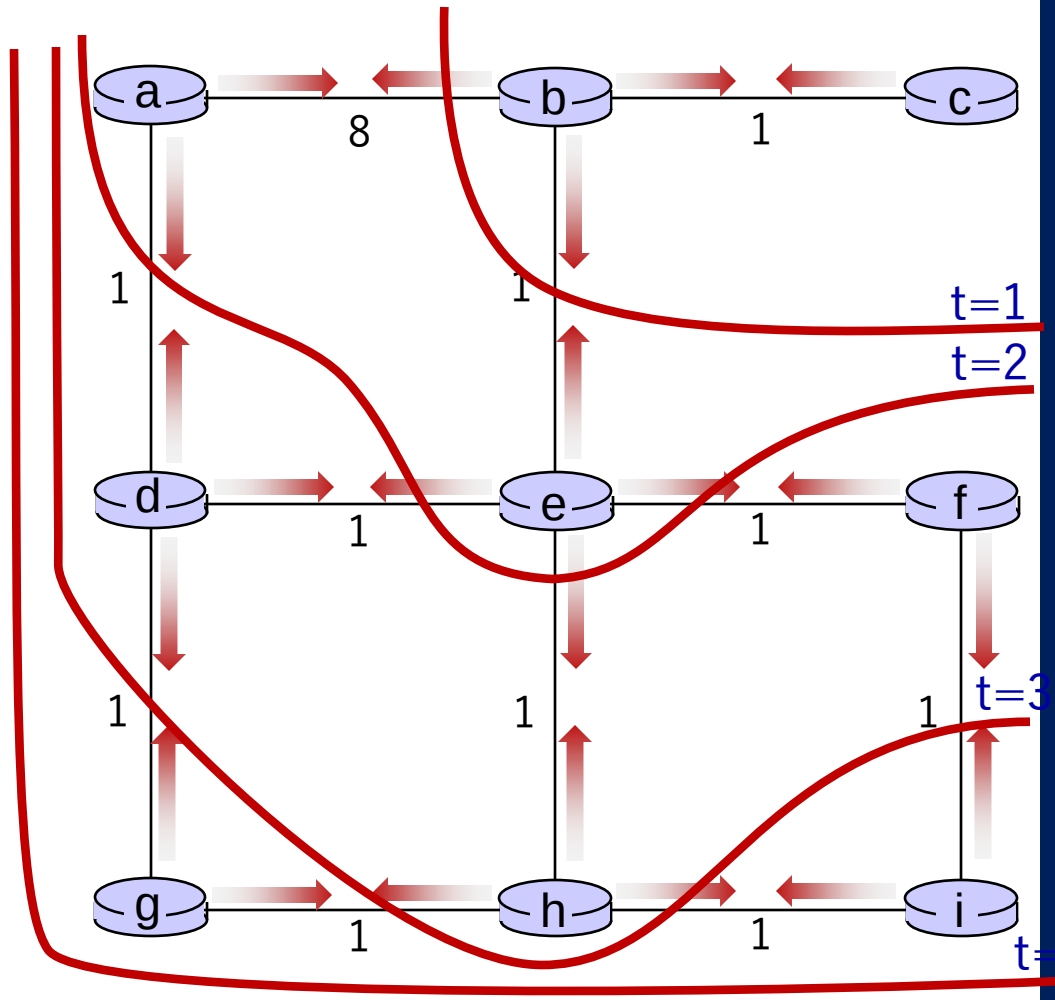
t=0 c's state at t=0 is at c only

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



京都大学

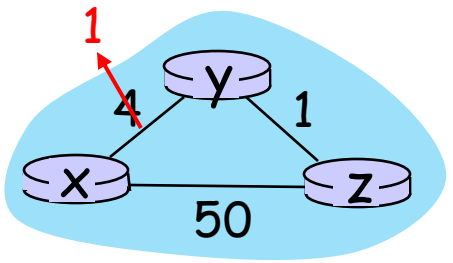
Link cost changes and link failure

- When a node running the distance vector algorithm detects a change, it updates its distance vector.
- If there is a change in a least-cost path, it informs its neighbors

Distance vector: link cost changes

link cost changes:

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



"good news travels fast"

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

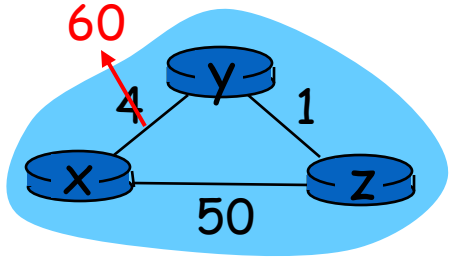
t_1: z receives update from y, updates its DV, computes new least cost to x, sends its neighbors its DV.

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

link cost changes:

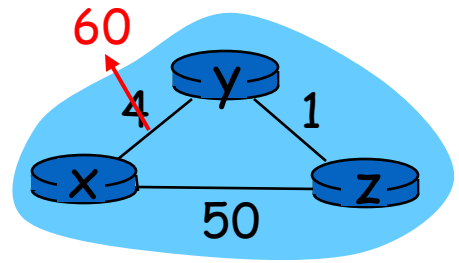
- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes



Distance vector: link cost changes

Before link cost changes:

$$D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$$



Node y detects link cost change:

$$D_y(x) = \min\{60 + 0, 1 + 5\} = 6$$

Now y informs z about the link cost change.

Node z updates its distance vector:

$$D_z(x) = \min\{50 + 0, 1 + 6\} = 7$$

Node z informs y about the link cost change.

$$D_y(x) = \min\{60 + 0, 1 + 7\} = 8$$

Node y informs z about the new cost and so on