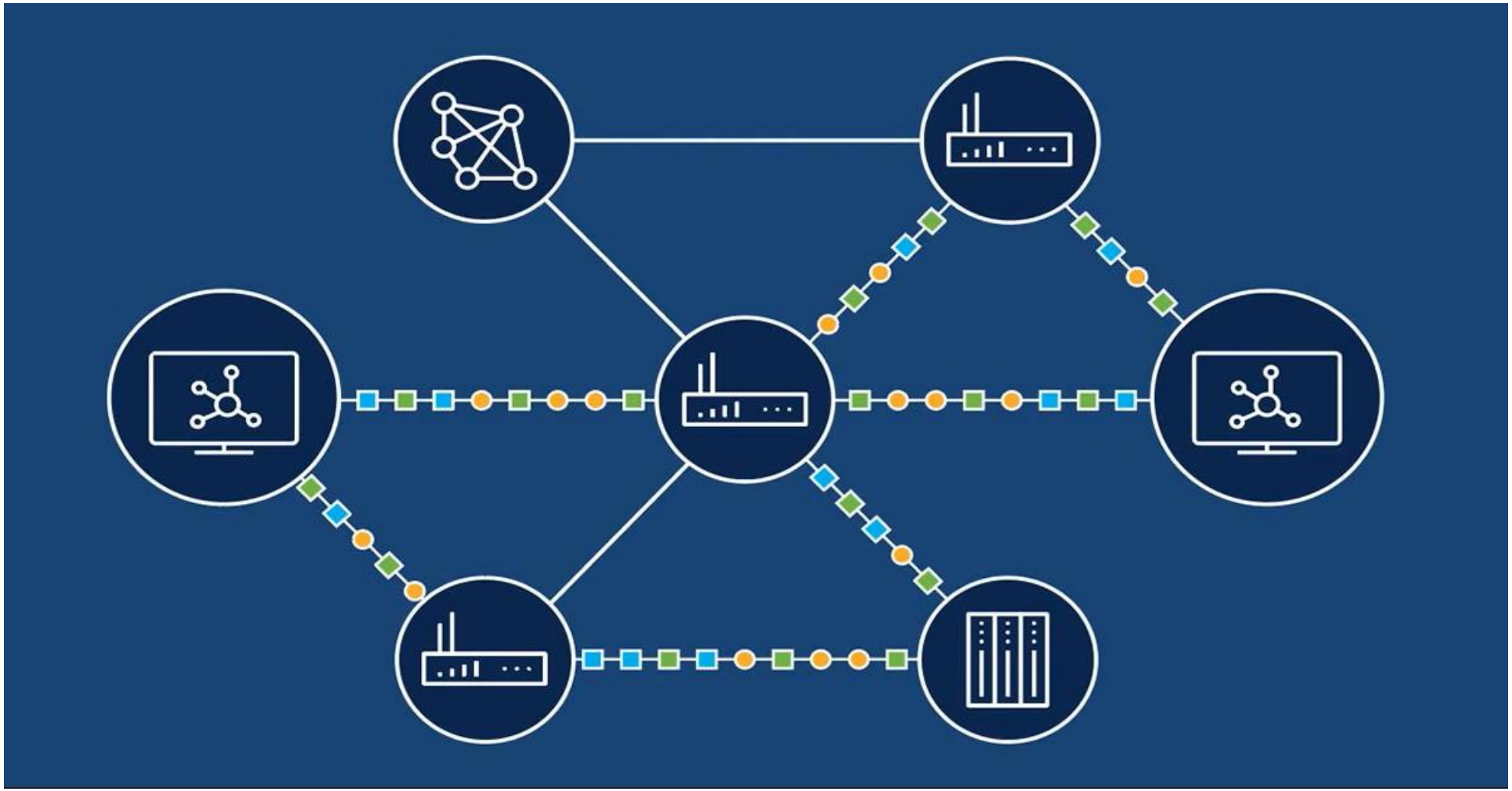
CAN201: Introduction to Networking

Lecture 7 - The Network Layer: Routing Algorithm



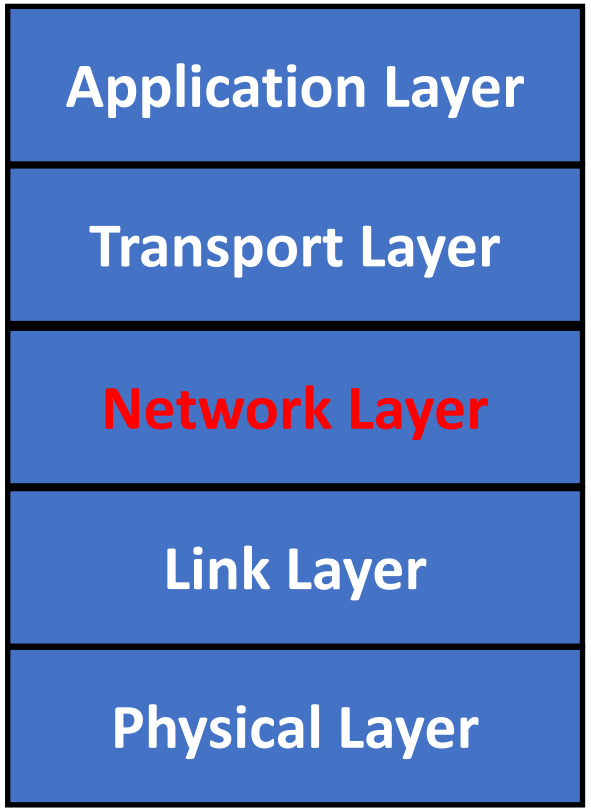
Lecturers: Dr. Gordon Boateng & Dr. Fei Cheng

Important Information

- **Contact:**
  - Email: Gordon.Boateng@xjtlu.edu.cn
  - Office No.: SC554A

- **Office Hours (Strictly via appointment)**
  - **Tuesday: 14:00-15:00**
  - **Wednesday: 14:00-15:00**

Recap: Top-Down Approach

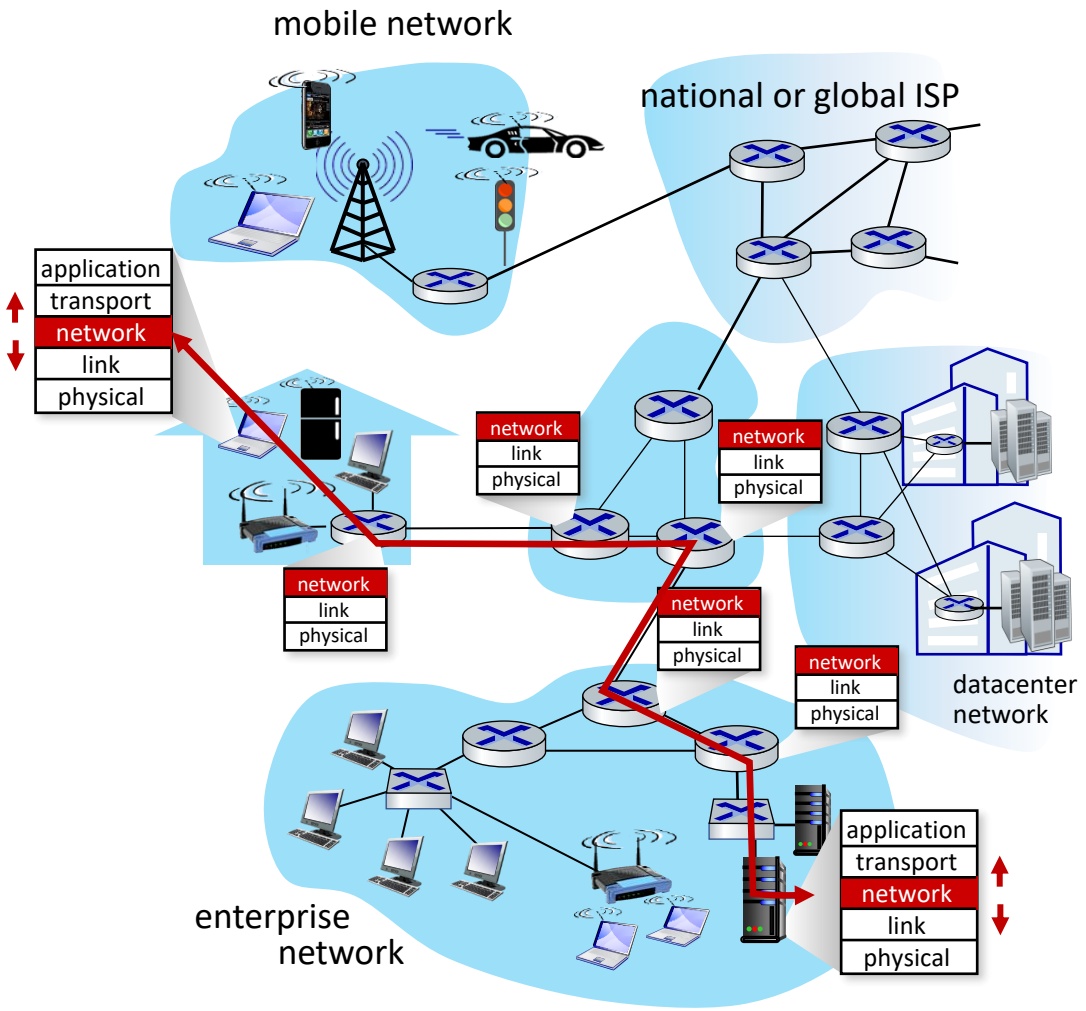


Network Layer: Data Plane

- Overview of Network Layer
- The Internet Protocol: IPv4, Addressing, NAT IPv6
- Generalized Forwarding and SDN
- Routing Protocols

Network-layer Services and Protocols

- Transport segment from sending host to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- router:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two Key Network-layer Functions

network-layer functions:

- *forwarding:* move packets from a router's input link to appropriate router output link
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

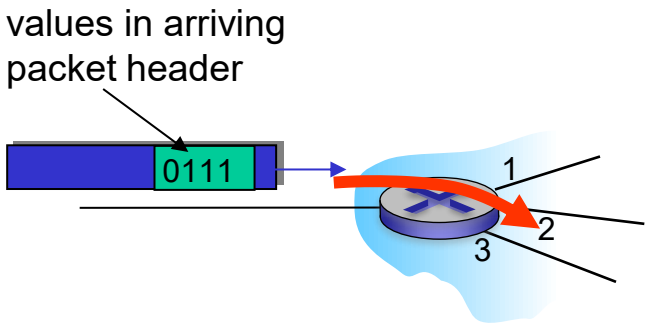
forwarding

routing

Network layer: Data plane, Control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

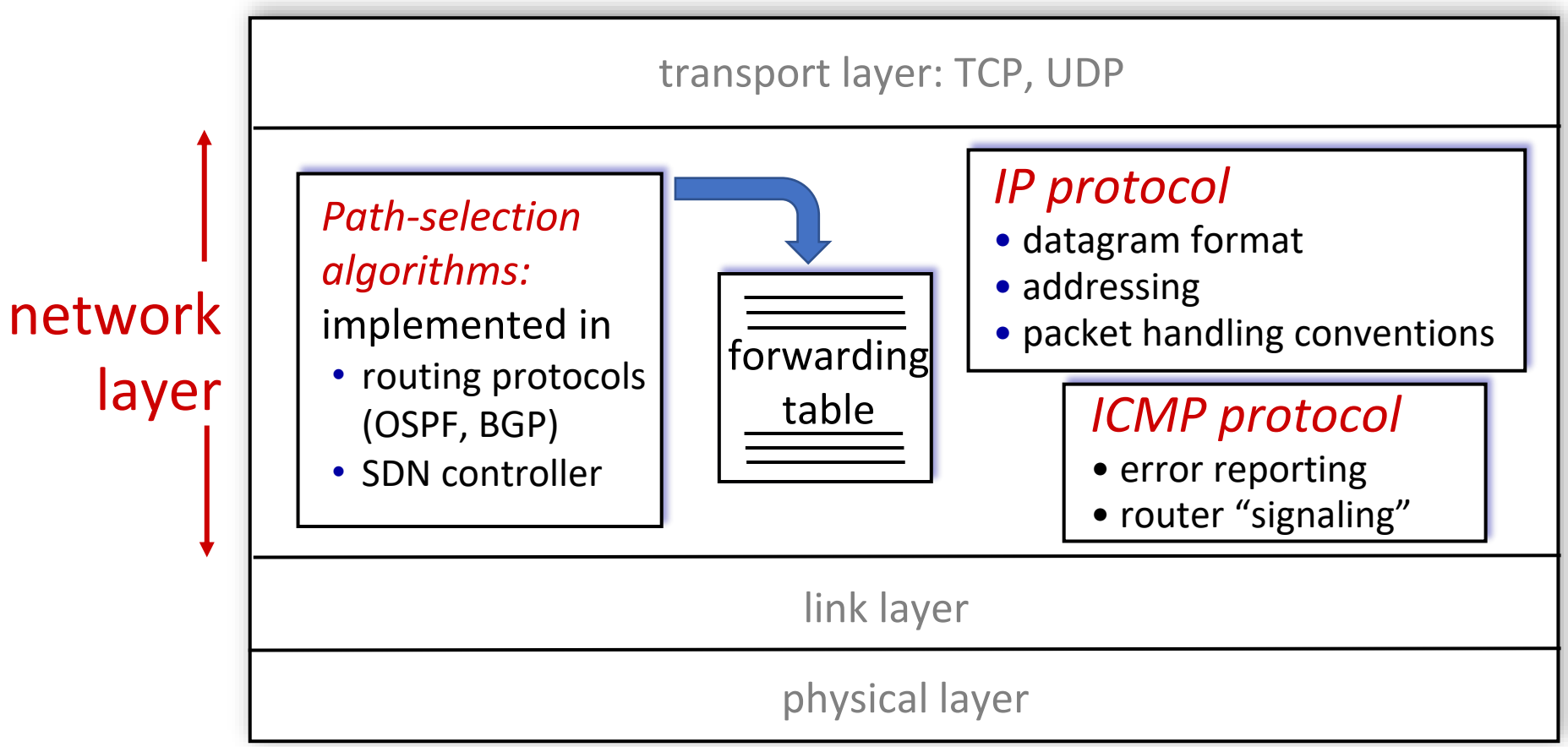
Q1: What is the difference between control plane and data plane?

Network Layer: Data Plane

- Overview of Network Layer
- The Internet Protocol:
  - IPv4, addressing (part 1)
  - NAT, IPv6 (part 2)
- Generalized Forwarding and SDN
- Routing Protocols

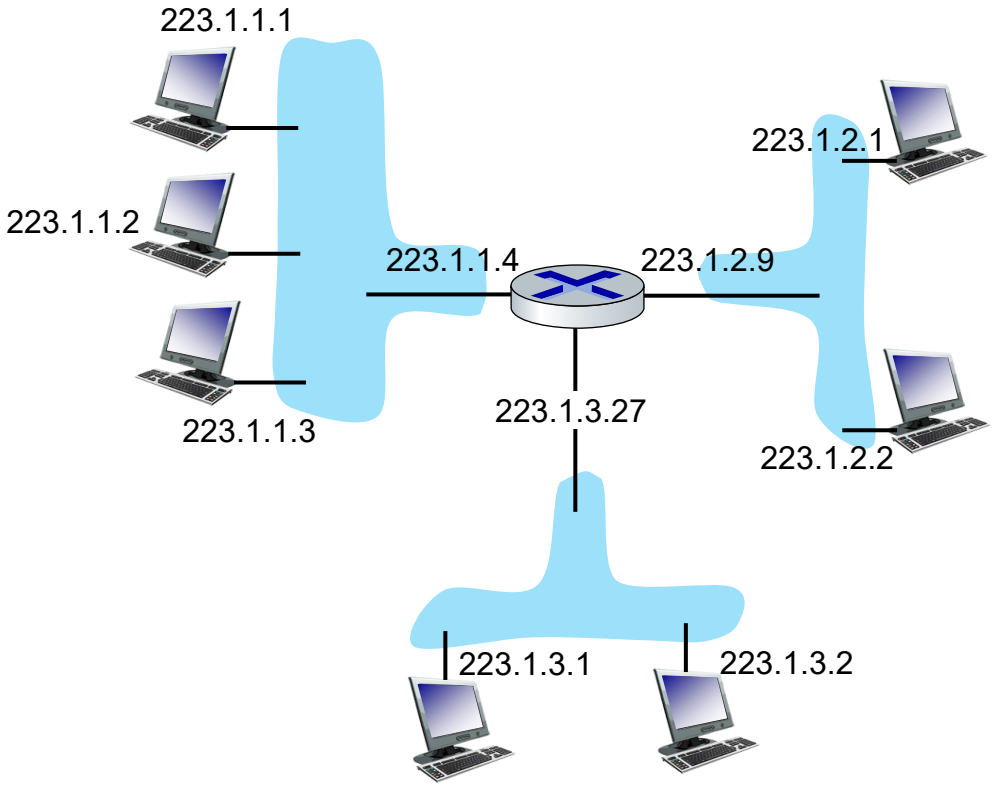
Network Layer: Internet

host, router network layer functions:



IP Addressing: Introduction

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



dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

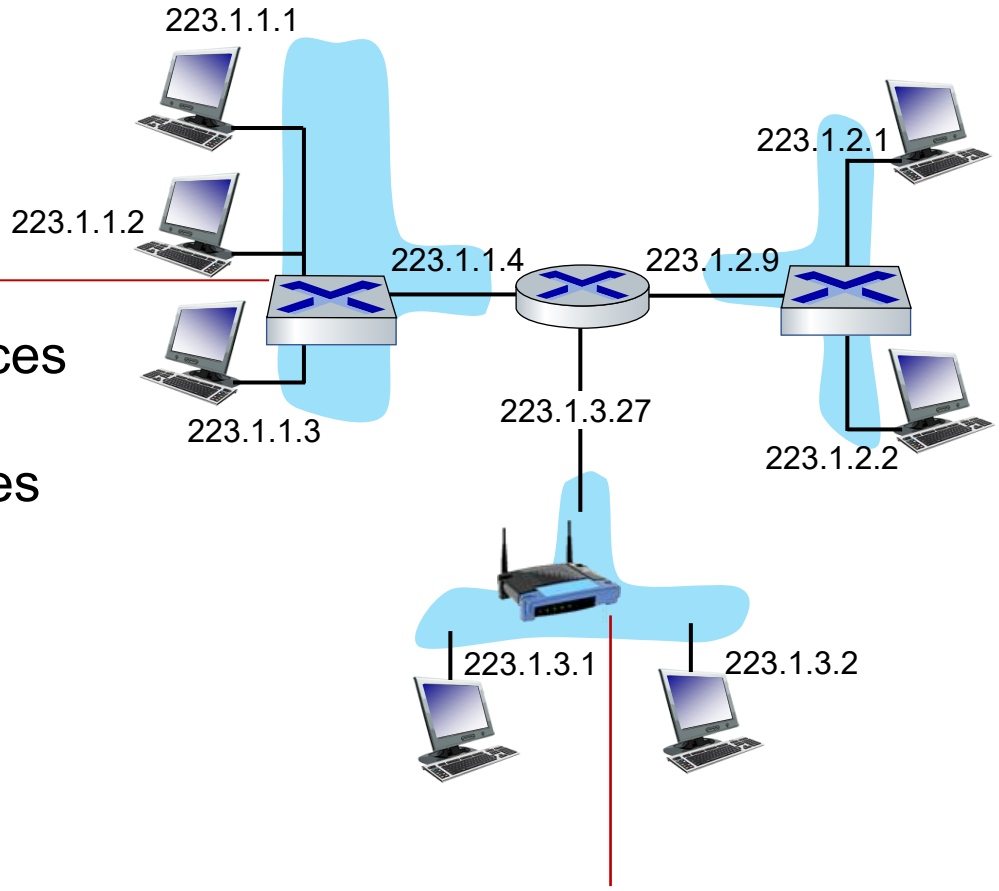
223 1 1 1

Q2: if a router has 5 interfaces, how many IP addresses does it have?

IP Addressing: Introduction (cont'd)

Q: How are interfaces actually connected?

A: wired Ethernet interfaces connected by Ethernet switches

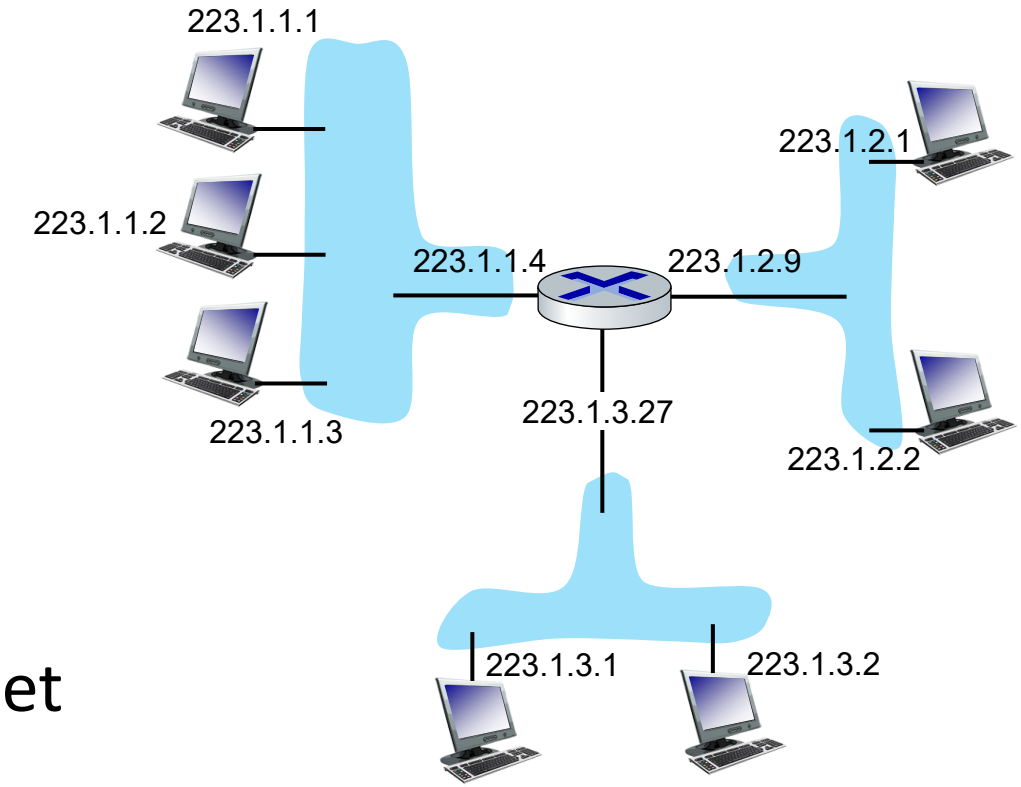


A: wireless WiFi interfaces connected by WiFi base station

Q3: **What is the difference between a router and a switch?**

Subnets

- *What is a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have two parts:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits

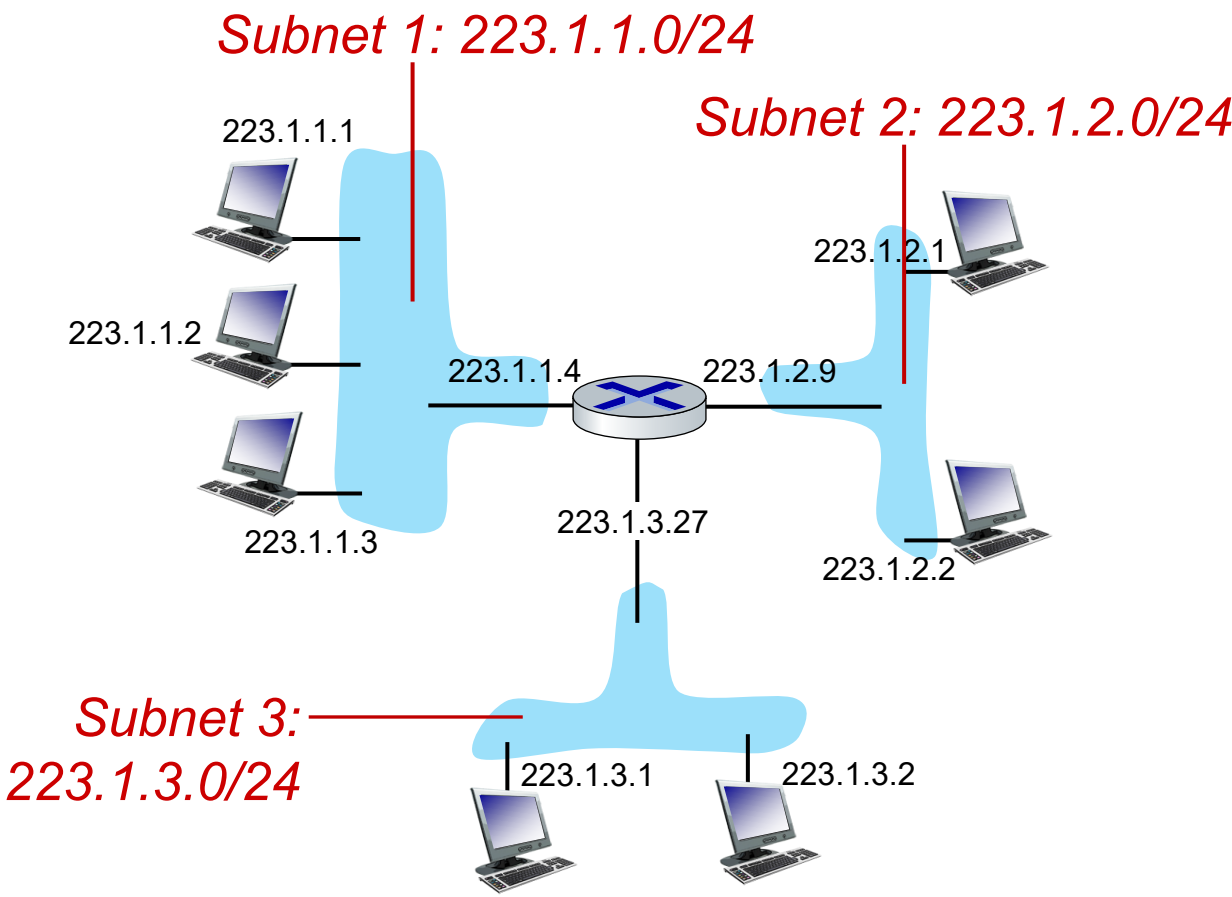


network consisting of 3 subnets

Subnets (cont'd)

Recipe for defining subnets:

- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*

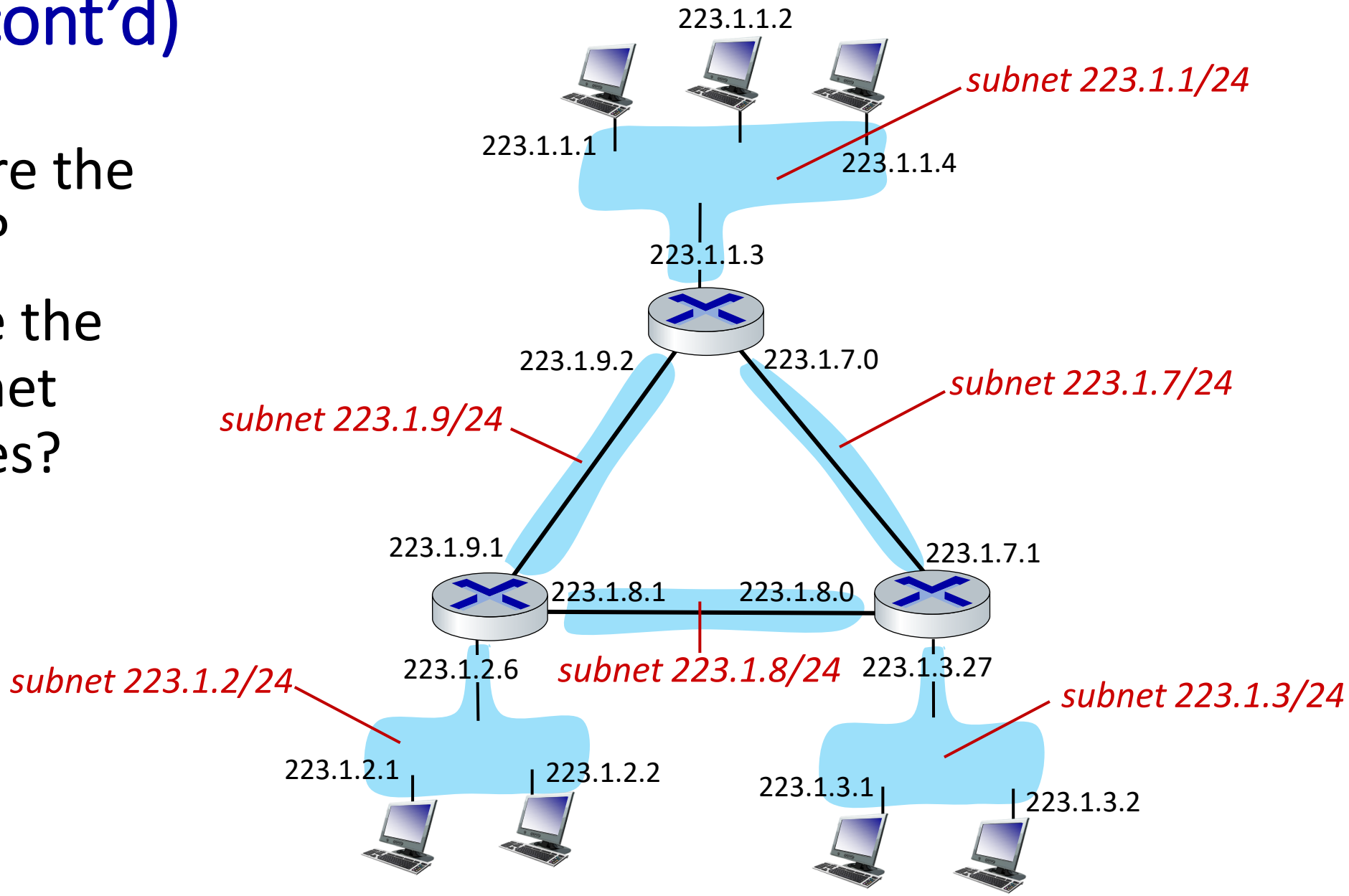


subnet mask: /24

(high-order 24 bits: subnet part of IP address)

Subnets (cont'd)

- where are the subnets?
- what are the /24 subnet addresses?



IP addressing: Network Classes

Classful addressing:

- The network portion of an IP address were constrained to be 8, 16, or 24 bits in length.
- Subnets with 8-, 16-, and 24-bit subnet addresses were known as class A, B and C networks.
- It became problematic
  - A class B (/16) subnet supporting $2^{16} - 2 = 65{,}534$ hosts, which is too large

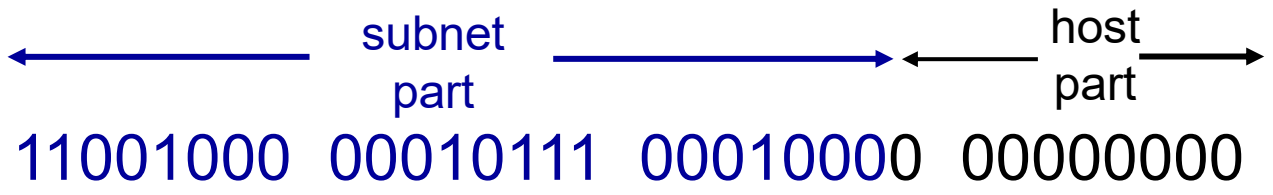
Under classful addressing, an organization with 10,000 hosts, which subnet class should be allocated? What is the problem?

This led to a rapid depletion of the class B address space and poor utilization of the assigned address space.

IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

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



200.23.16.0/23

IP addresses: how to get one?

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., Windows: control-panel->network->configuration->tcp/ip->properties, /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from a server

DHCP: Dynamic Host Configuration Protocol

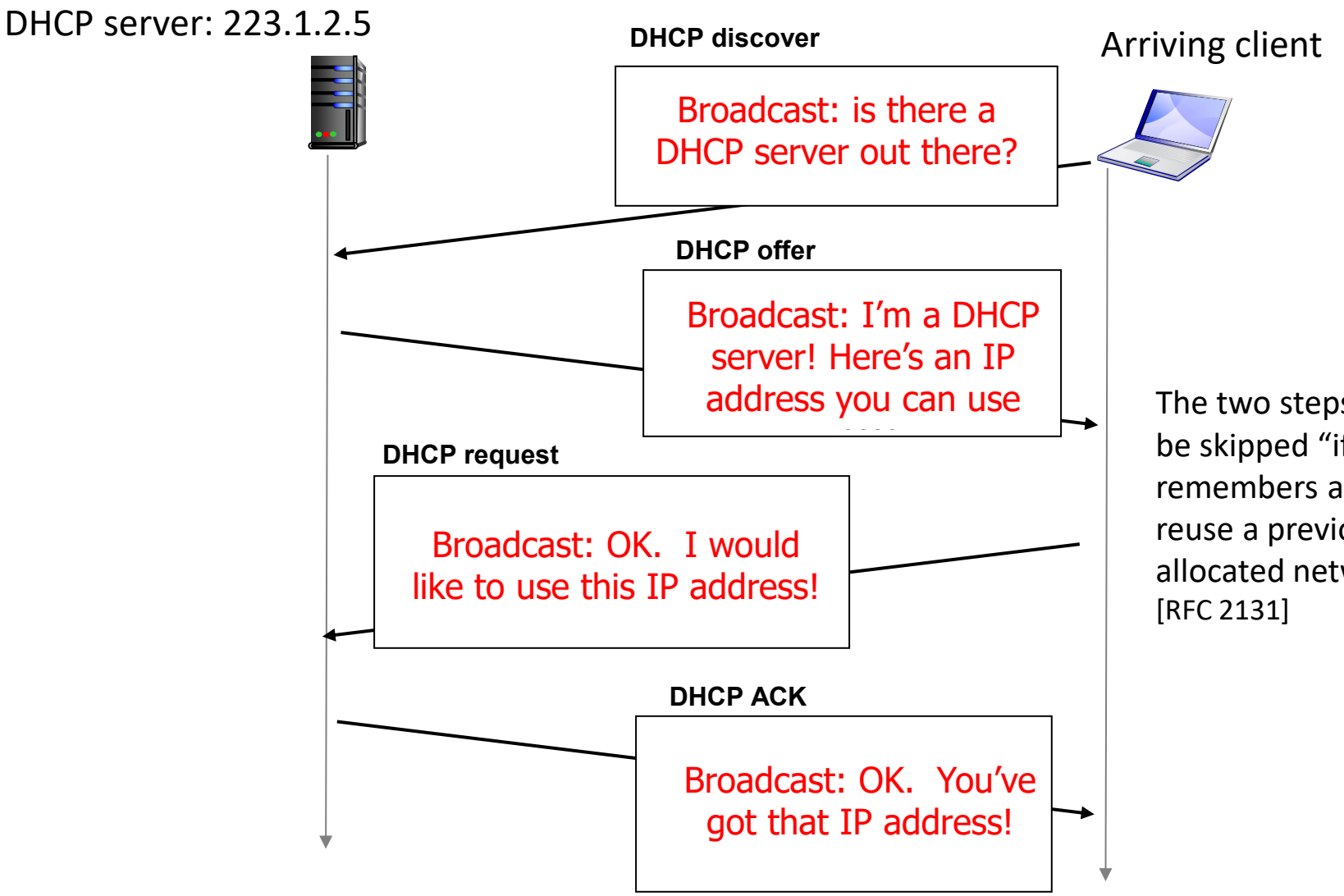
Goal: Allow host to *dynamically* obtain its IP address from network server when it "joins" network

- 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

DHCP Client-Server Scenario



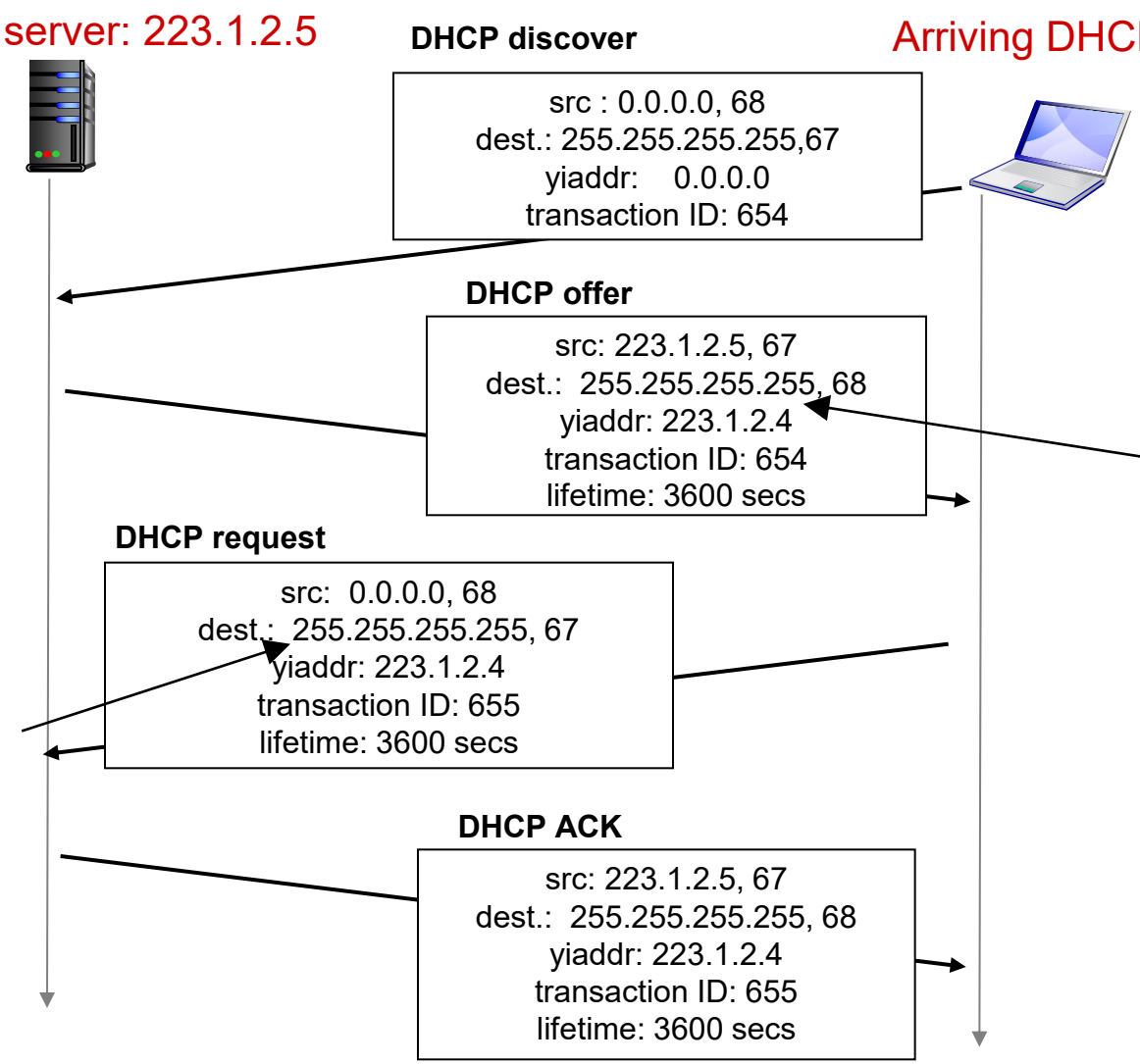
The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP: more than IP addresses

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)

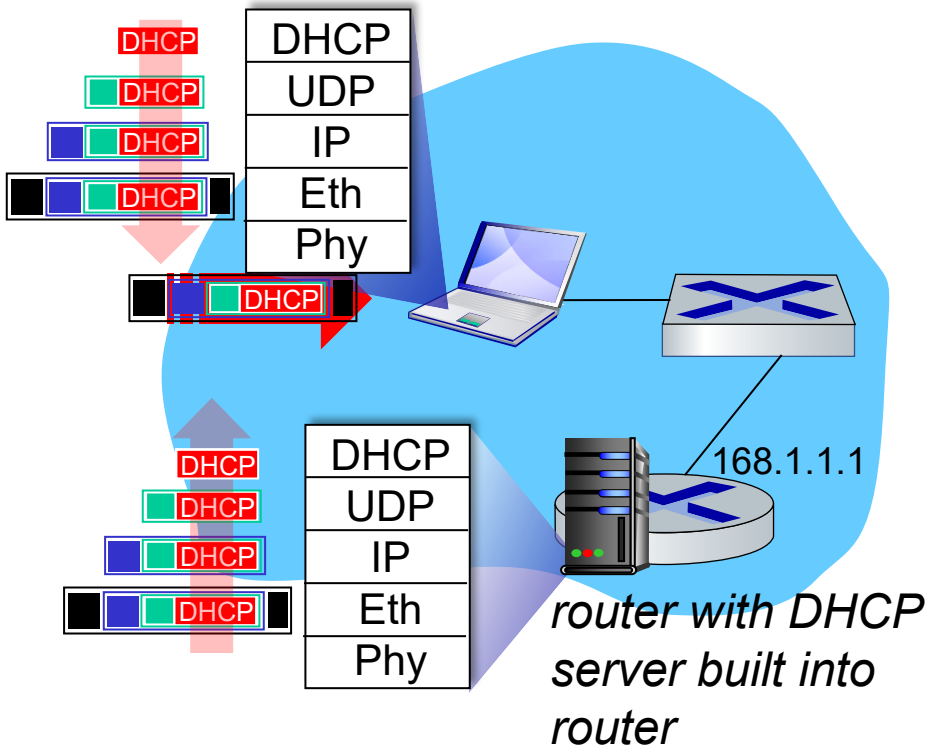
DHCP Client-Server Scenario (cont'd)



Why must this server reply be broadcast?

Why must this client request also be broadcast?

DHCP: Example



- 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: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

IP addresses: how to get one?

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

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

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20

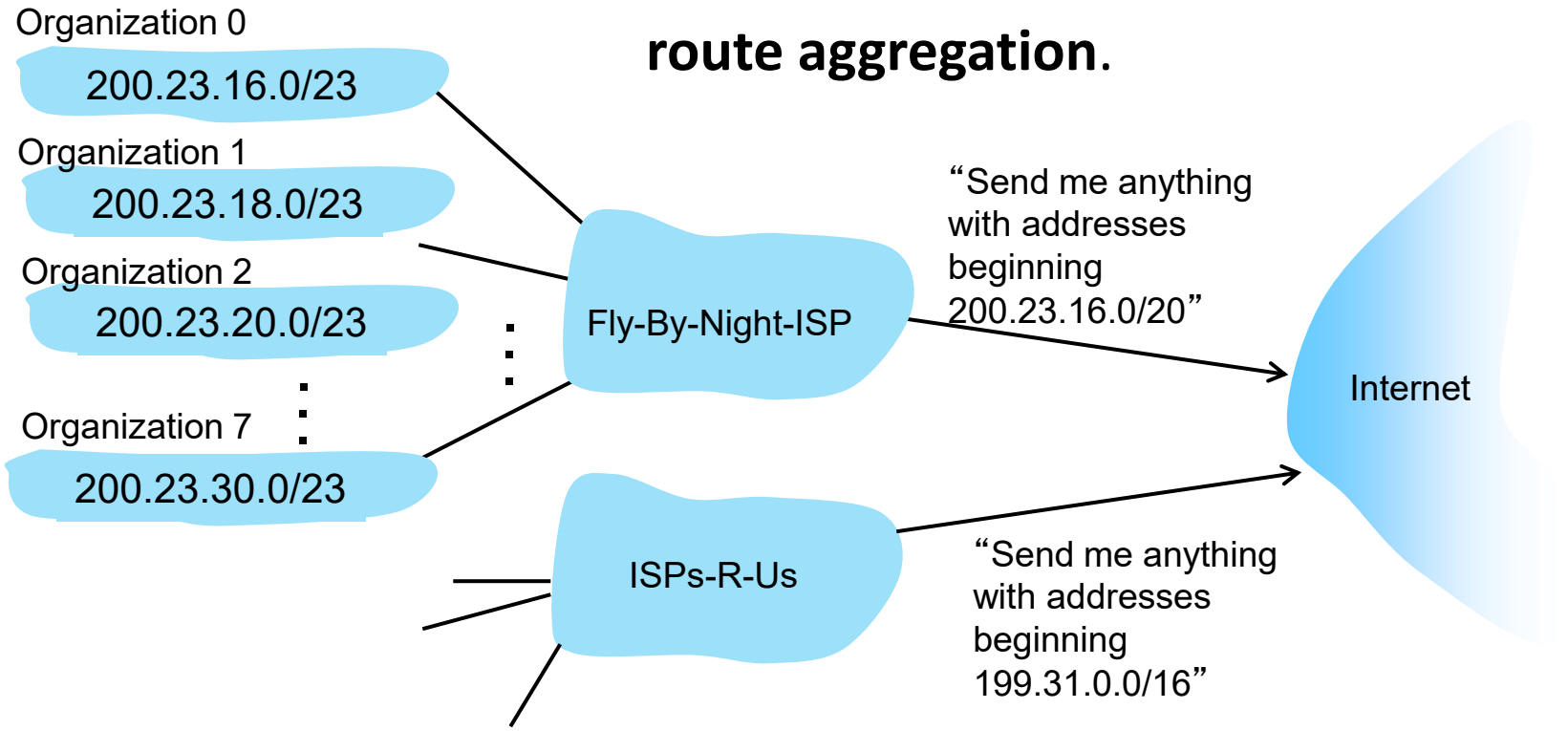
ISP can then allocate out its address space in, say, 8 blocks:

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

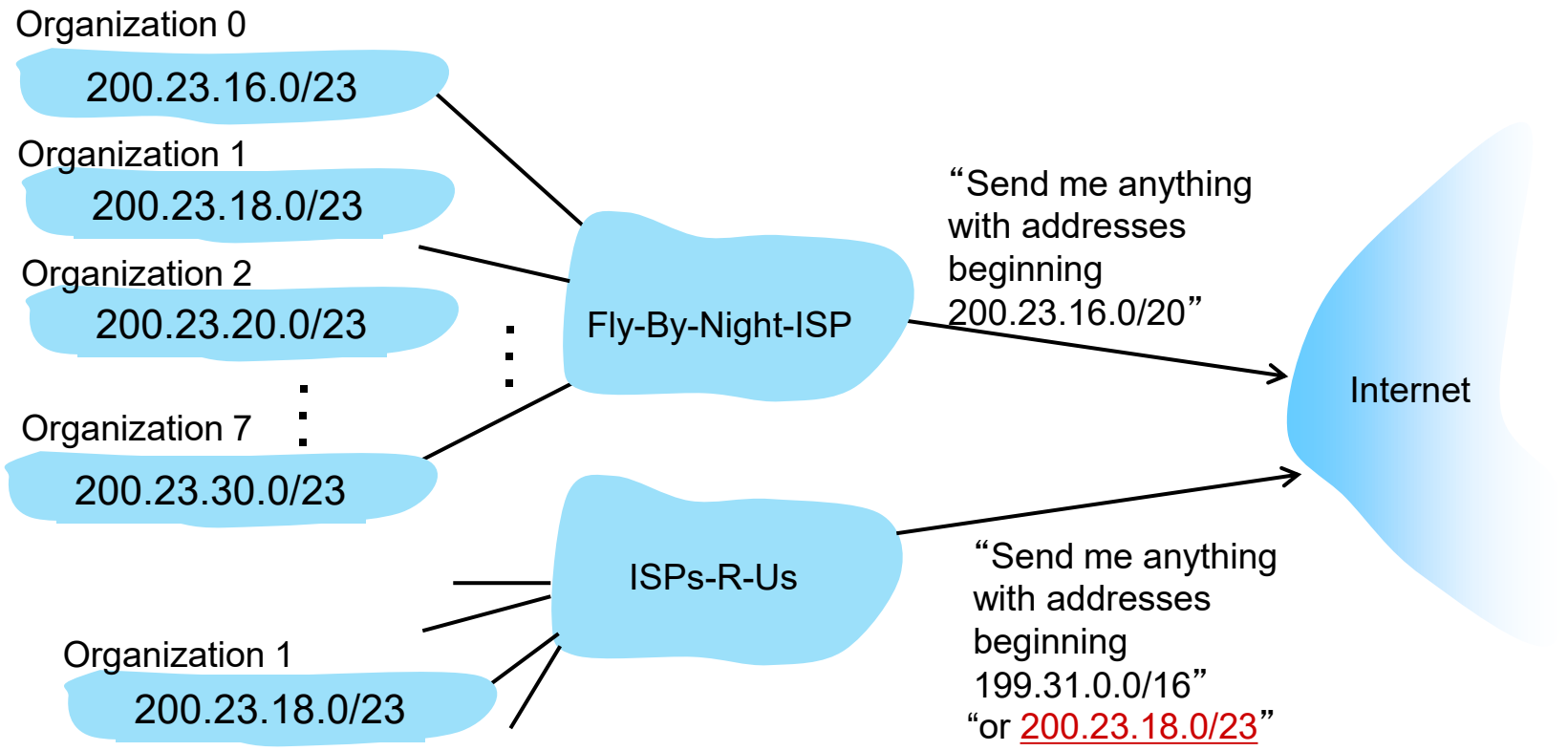
Hierarchical addressing allows efficient advertisement of routing information:

The ability to use a single prefix to advertise multiple networks is often referred to as **address aggregation** or **route aggregation**.



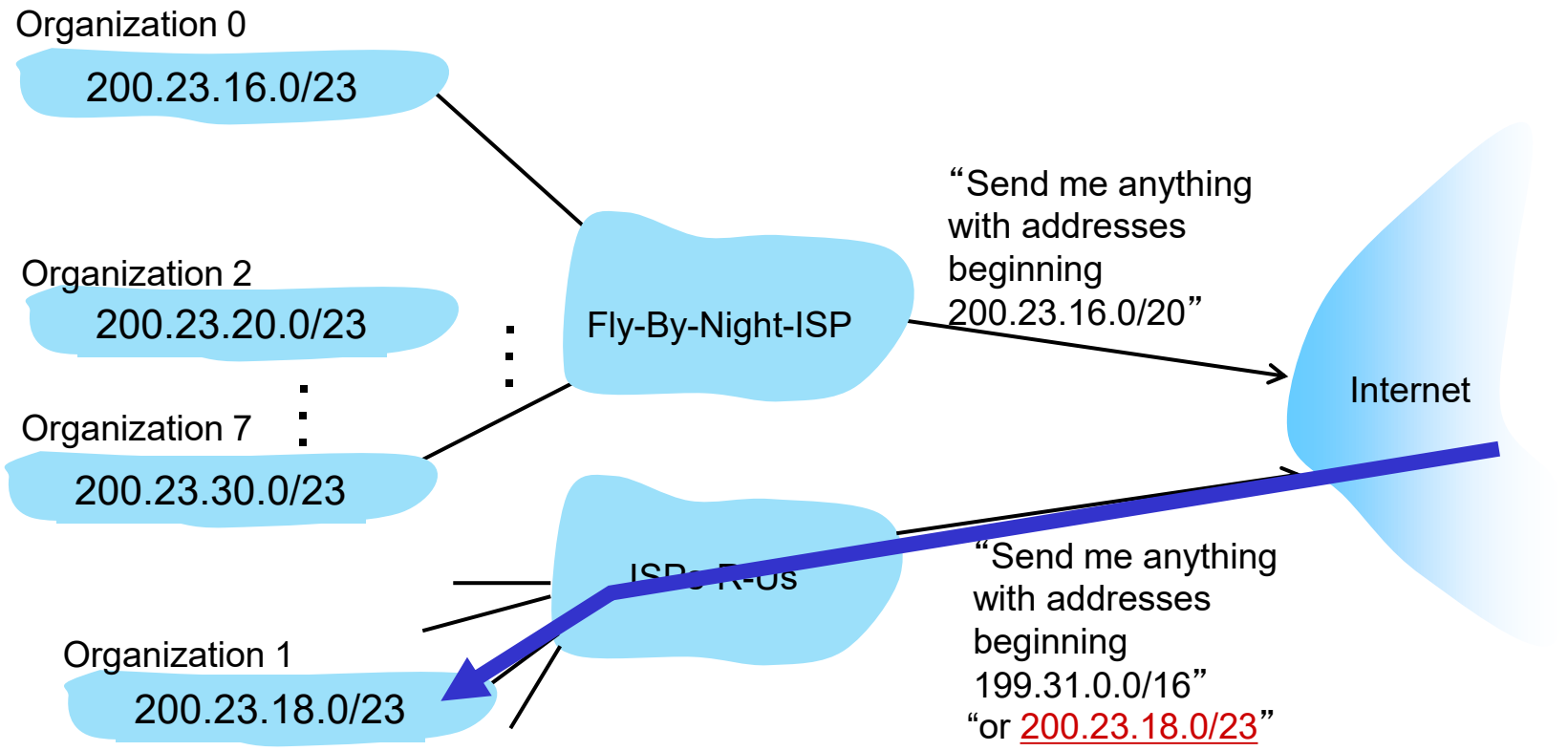
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



IP addressing: last words ...

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?

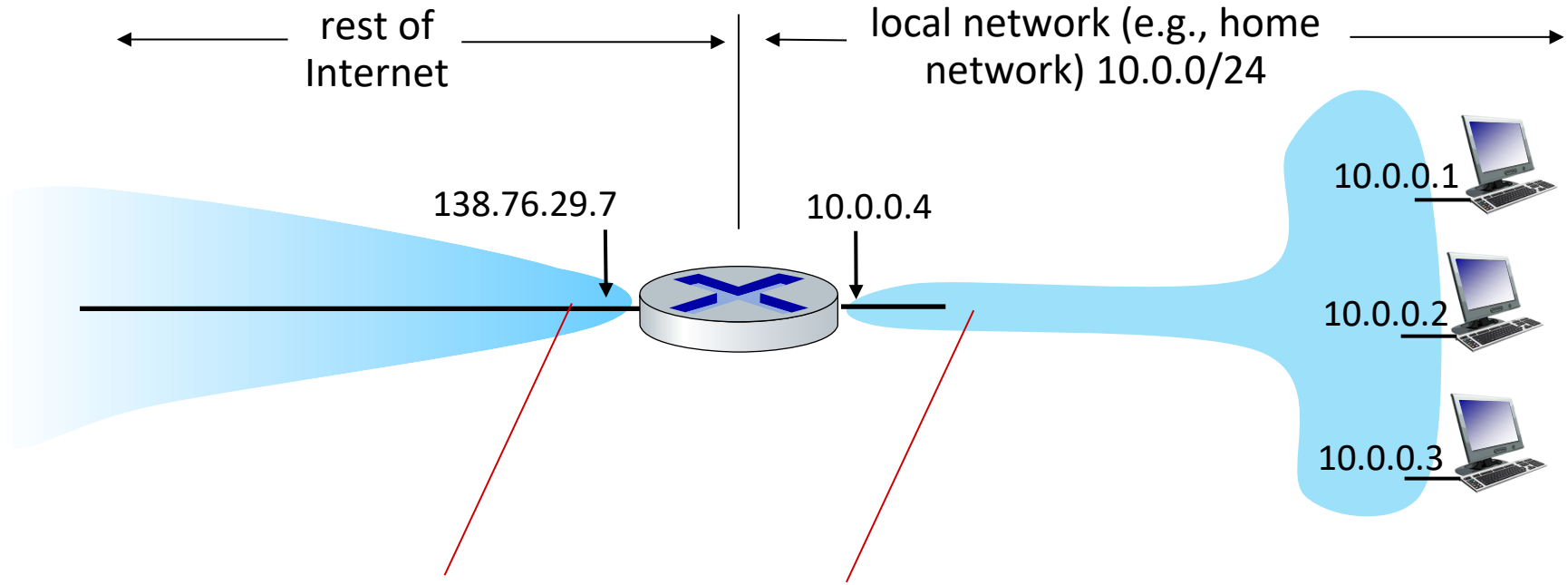
- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

Network Layer: Data Plane

- Overview of Network Layer
- The Internet Protocol:
  - IPv4, addressing (part 1)
  - NAT, IPv6 (part 2)
- Generalized Forwarding and SDN
- Routing Protocols

NAT: Network Address Translation

NAT: All devices in the local network share just one IPv4 address as far as the outside world is concerned



all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *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 (cont'd)

- all devices in local network have 32-bit addresses in a "private" IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  - just one IP address needed from provider ISP for *all* devices
  - can change addresses of host in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - security: devices inside local network not directly addressable, visible by outside world

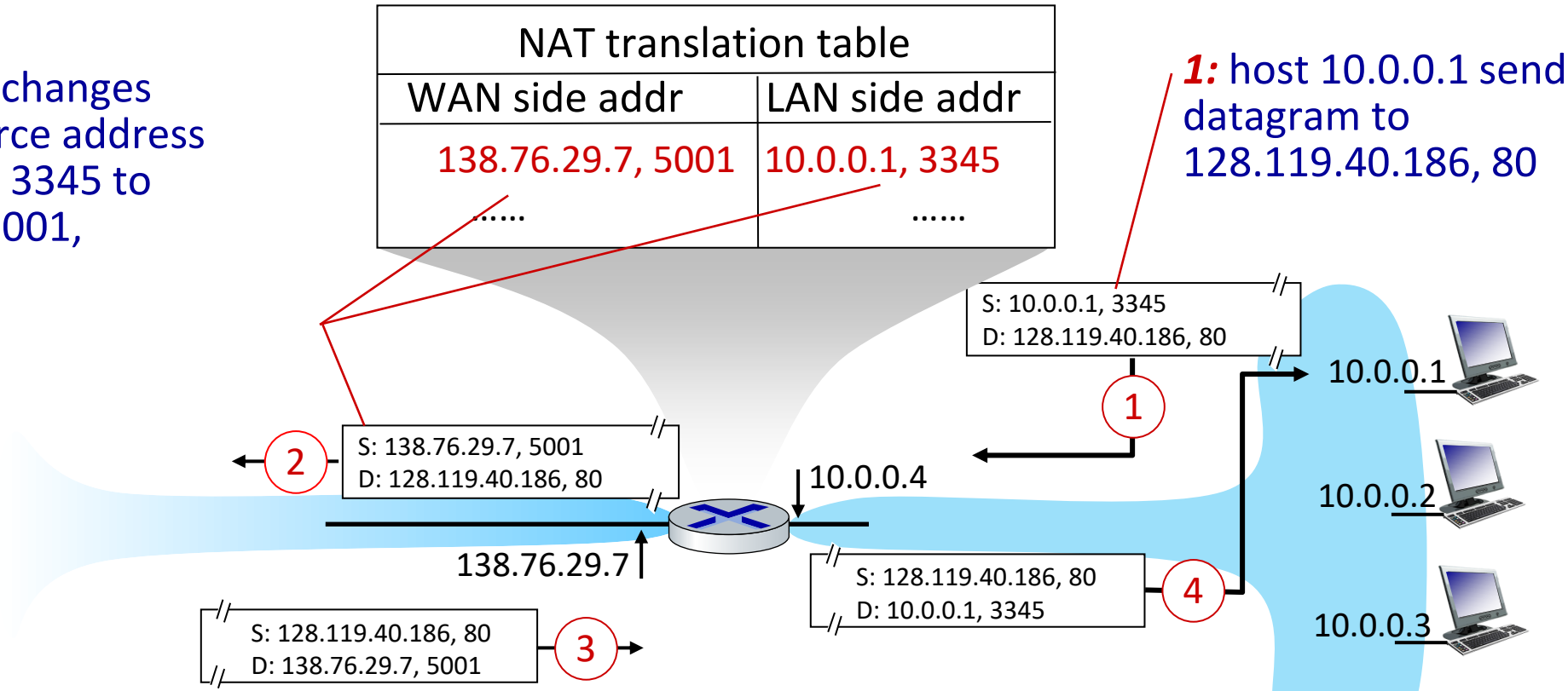
NAT: Network Address Translation (cont'd)

implementation: NAT router must (transparently):

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

NAT: Network Address Translation (cont'd)

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: reply arrives, destination address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT: Network Address Translation (cont'd)

- NAT has been controversial:
  - routers "should" only process up to layer 3
  - address "shortage" should be solved by IPv6
  - violates end-to-end argument (port # manipulation by network-layer device)
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G cellular nets

IPv6: Motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- Additional motivation:
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS

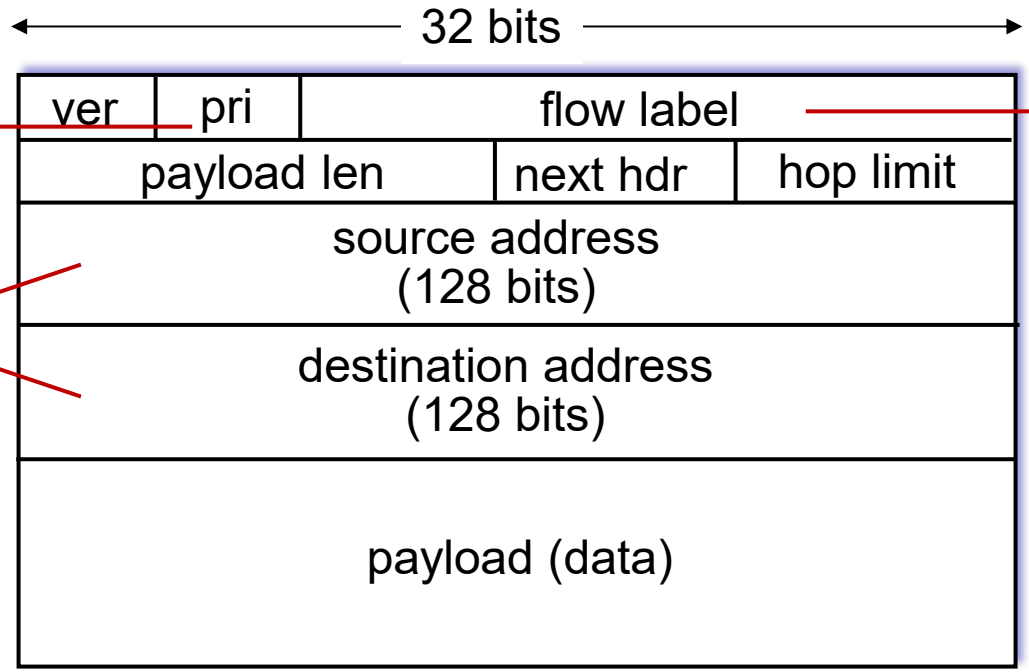
IPv6 datagram format:

- Fixed-length 40-byte header
- No fragmentation allowed (at intermediate routers, see "packet too big" new ICMP message type)

IPv6 Datagram Format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



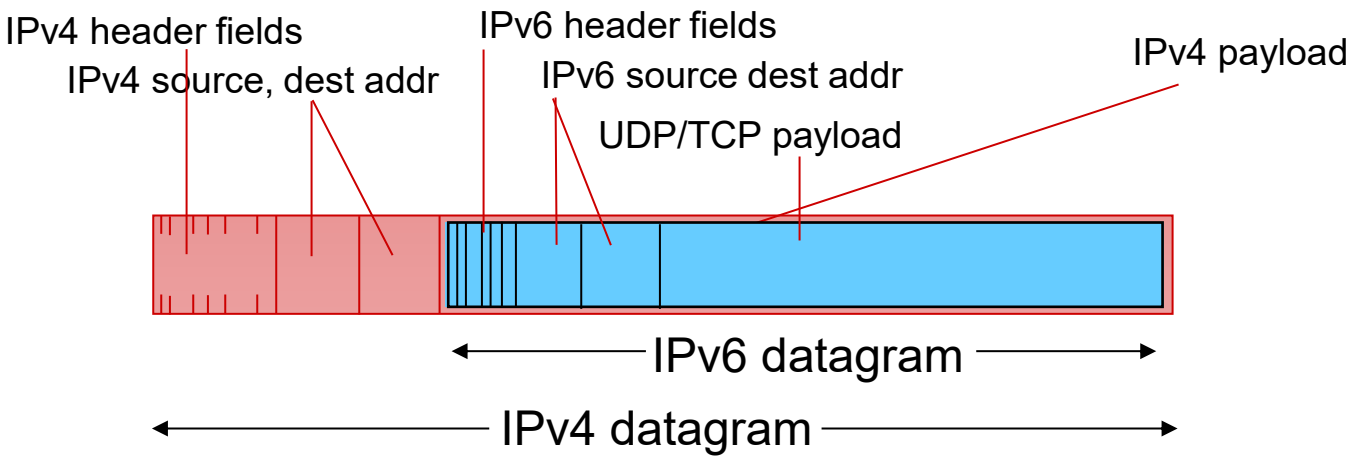
flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

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)

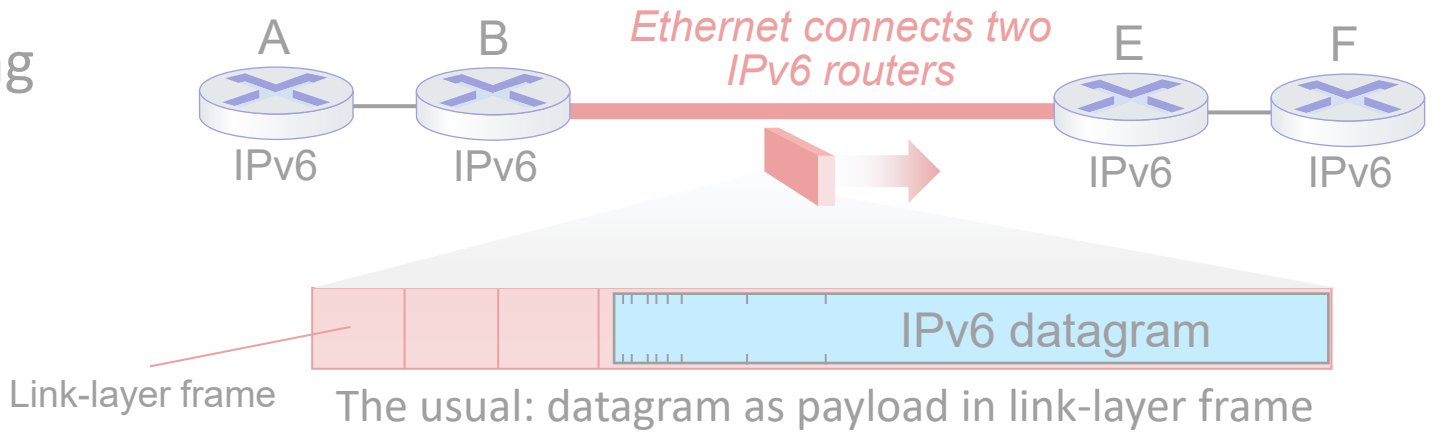
Transition from IPv4 to IPv6

- 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)

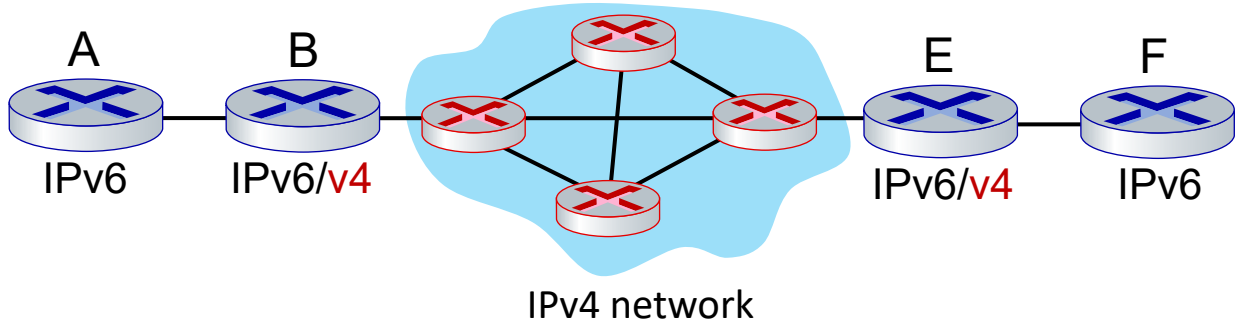


Tunneling and Encapsulation

Ethernet connecting two IPv6 routers:

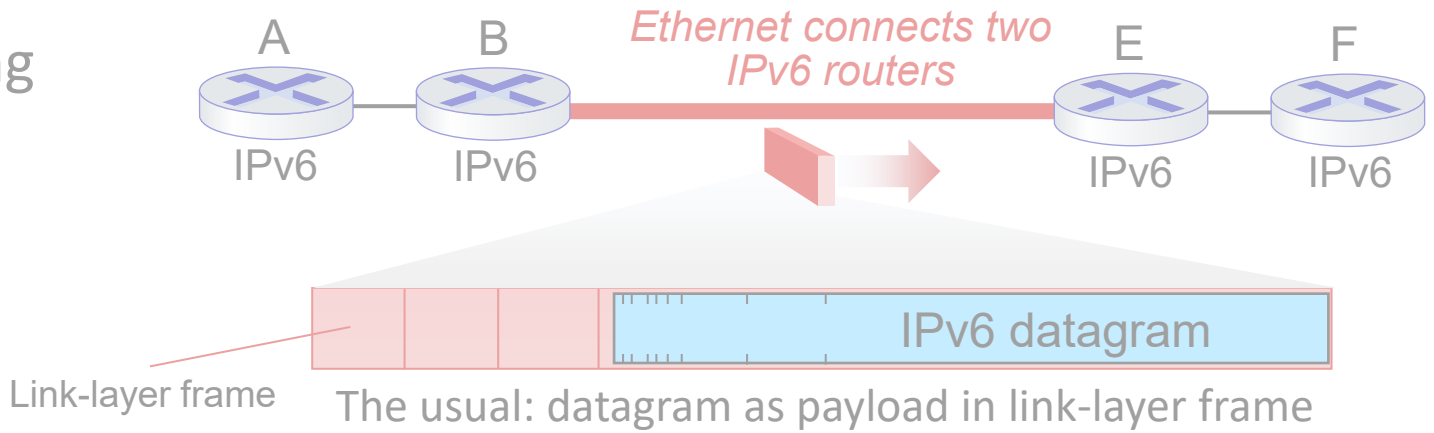


IPv4 network connecting two IPv6 routers

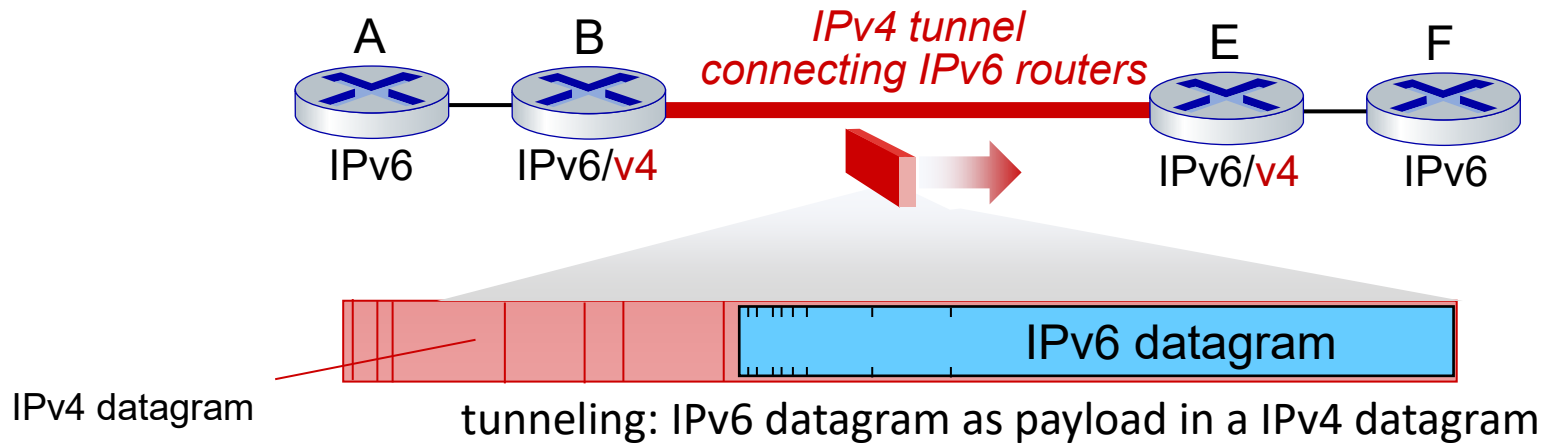


Tunneling and Encapsulation

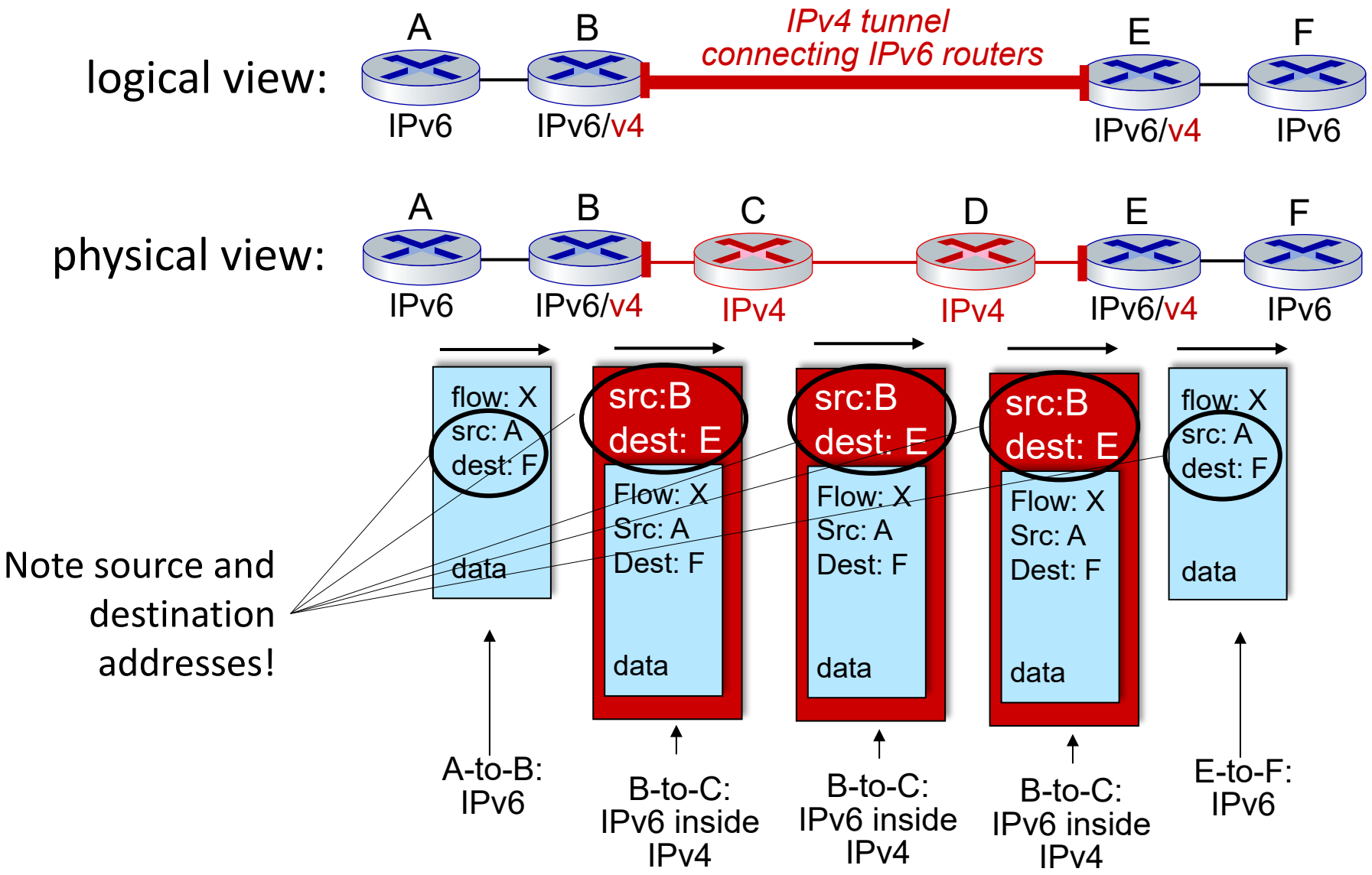
Ethernet connecting two IPv6 routers:



IPv4 tunnel connecting two IPv6 routers



Tunneling

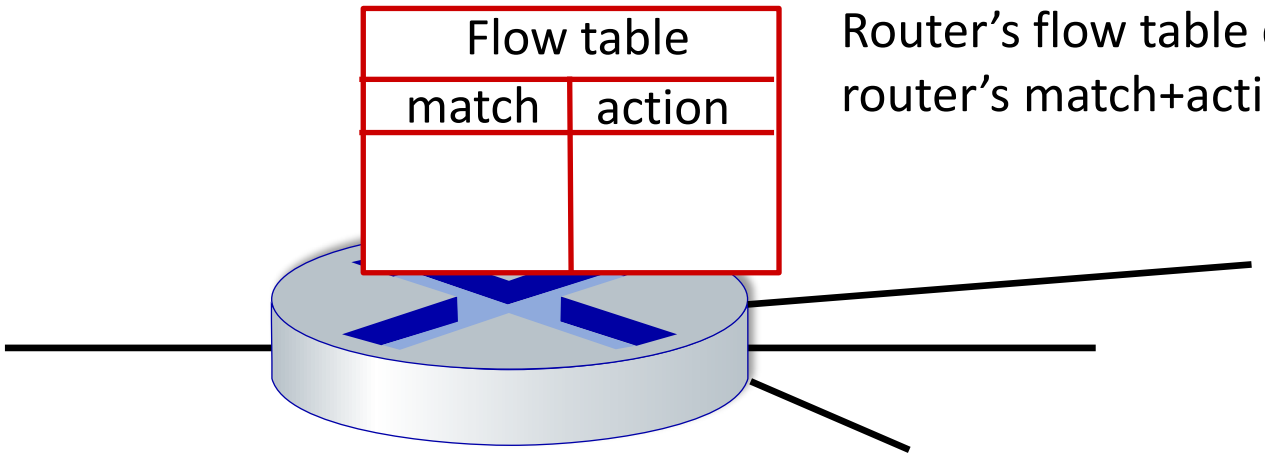

logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
Note source and
destination
addresses!
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Network Layer: Data Plane

- Overview of Network Layer
- The Internet Protocol: IPv4, Addressing, NAT IPv6
- Generalized Forwarding and SDN
- Routing Protocols

Flow Table Abstraction

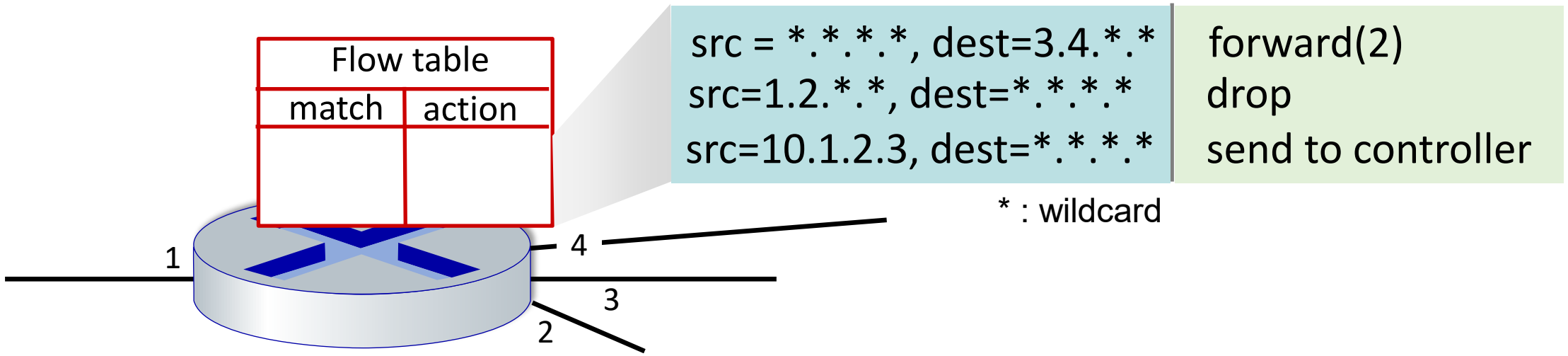
- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



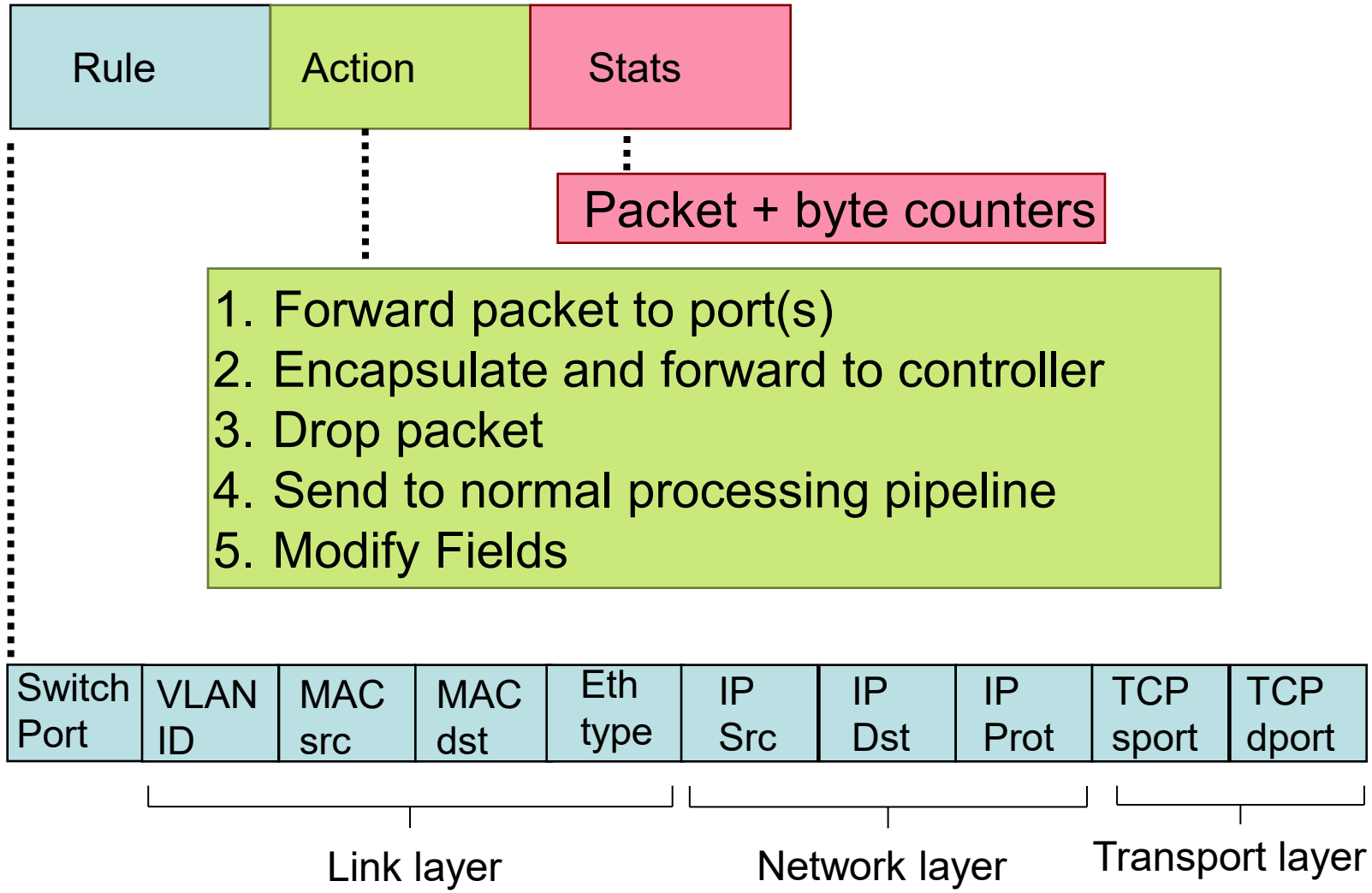
Router's flow table define router's match+action rules

Flow Table Abstraction

- flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - Pattern: match values in packet header fields
  - Actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets



OpenFlow: Flow Table Entries



OpenFlow: Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

OpenFlow: Examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow Abstraction

- match+action: abstraction unifies different kinds of devices

Router

- *match:* longest destination IP prefix
- *action:* forward out a link

Switch

- *match:* destination MAC address
- *action:* forward or flood

Firewall

- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT

- *match:* IP address and port
- *action:* rewrite address and port

Network Layer: Data Plane

- Overview of Network Layer
- The Internet Protocol: IPv4, Addressing, NAT IPv6
- Generalized Forwarding and SDN
- Routing Protocols

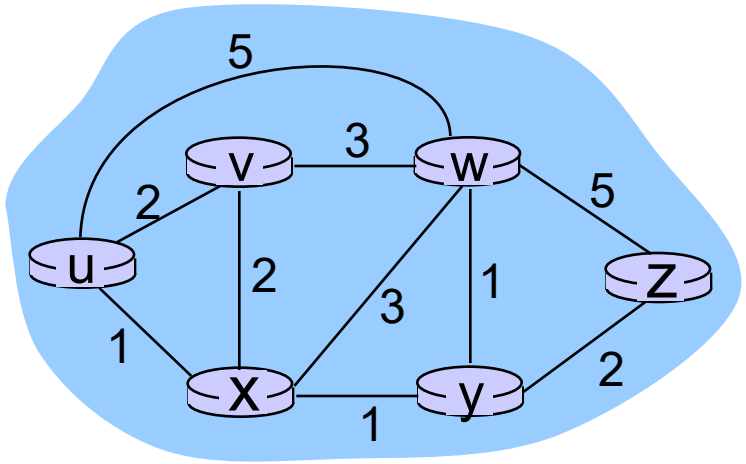
Routing Protocols

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

- **Path**: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested"
- Routing: a "top-10" networking challenge!

Graph Abstraction of the Network

A graph is often used to formulate routing problems..



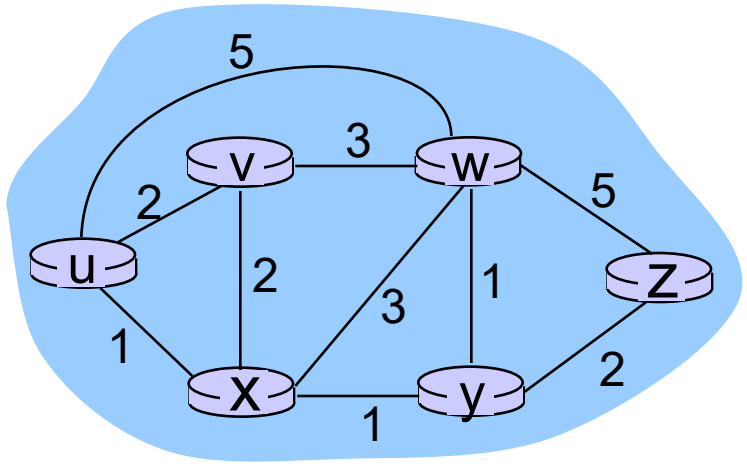
graph: G = (N,E)

N = set of nodes (routers) = { u, v, w, x, y, z }

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

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph Abstraction: Link Costs



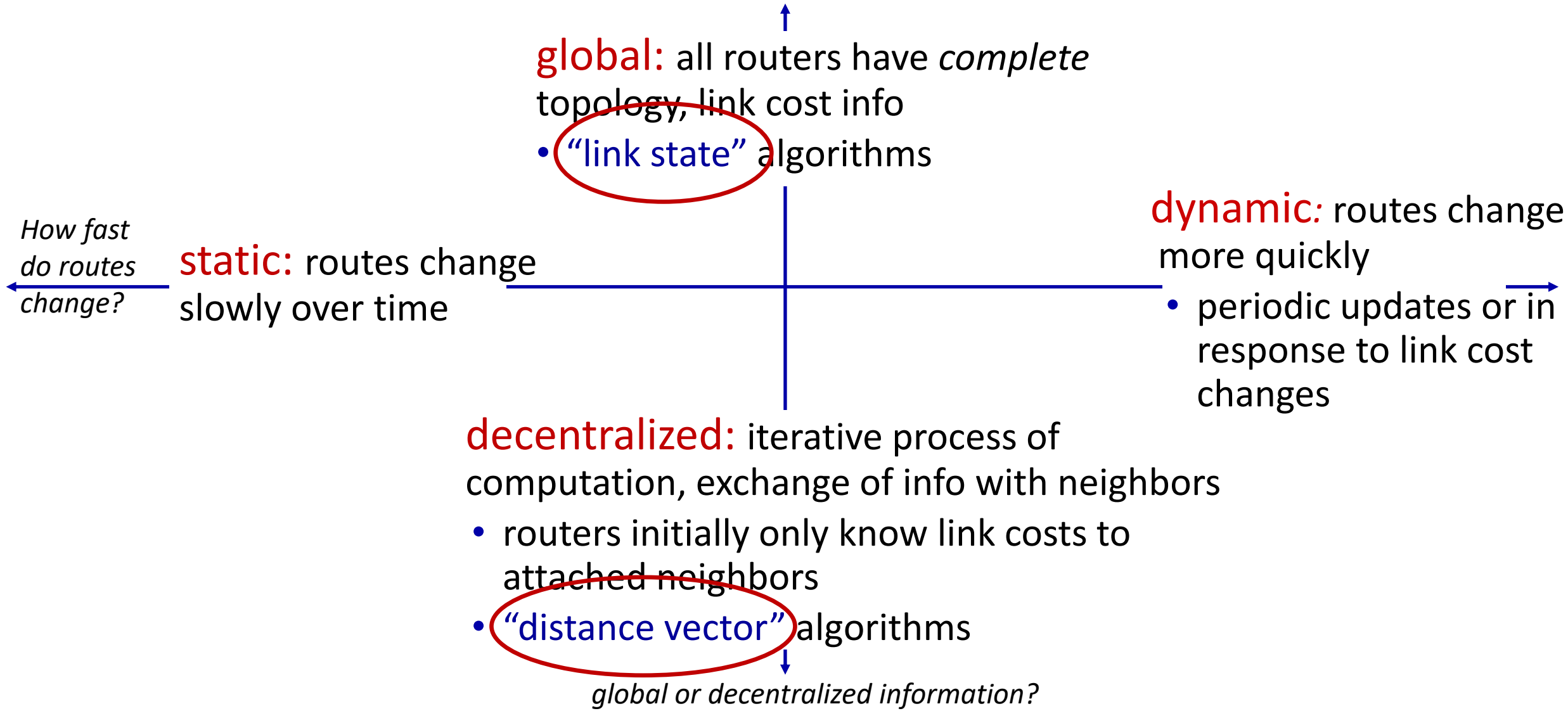
c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

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

Routing Algorithm Classification



A Link-State Routing Algorithm - Dijkstra

Dijkstra's algorithm

- **Net topology and 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 dest.'s**

Notations:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to dest. v
- N': set of nodes whose least cost path is definitively known

Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
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 cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```

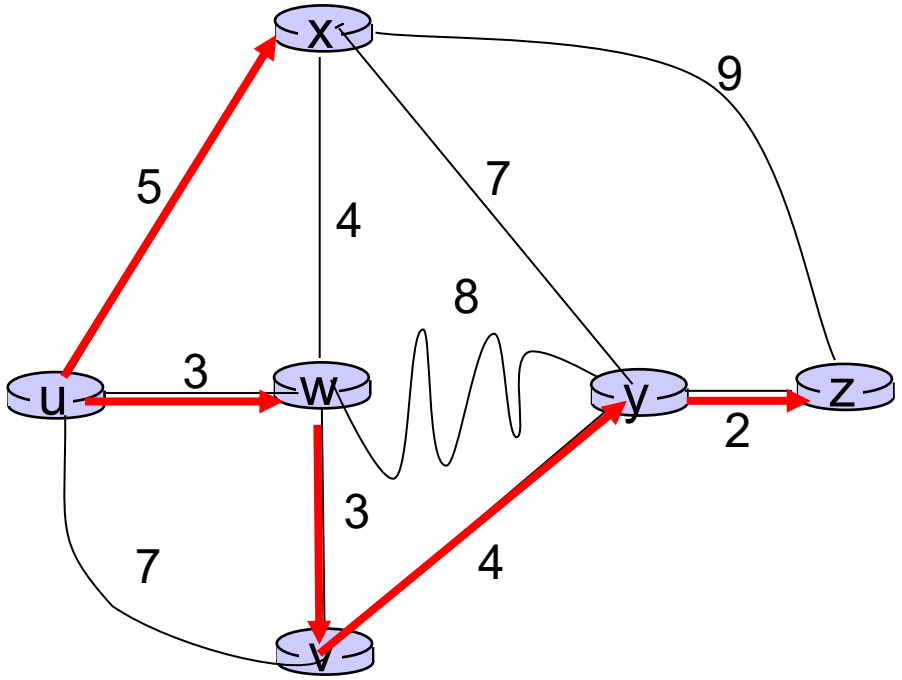
Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijkstra's Algorithm: Example 1

computing routing table in u:

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

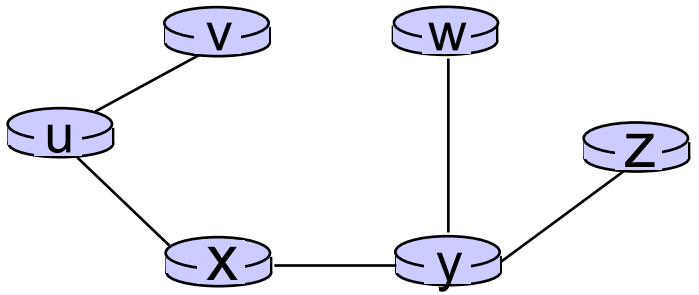


notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's Algorithm: Example 2 - result

resulting shortest-path tree from u:



resulting forwarding table in u:

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

Distance Vector Algorithm

- Bellman-Ford equation (dynamic programming)

  let

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

  then

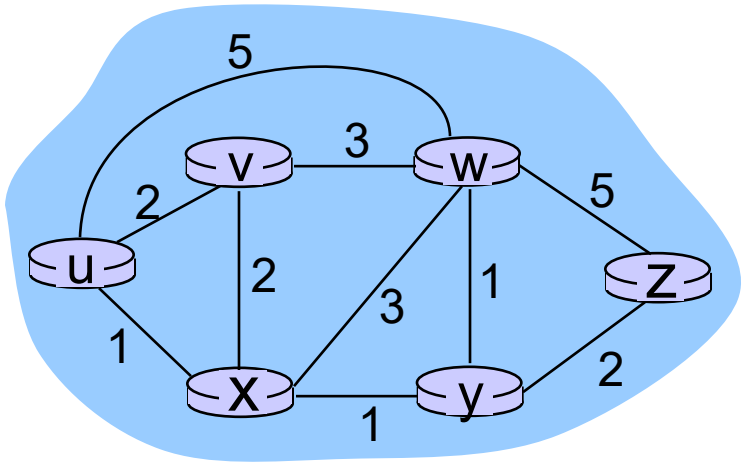
$$d_x(y) = \min_v\{ c(x,v) + d_v(y) \}$$

cost from neighbor v to destination y

cost to neighbor v

min_v is taken over all neighbors v of x

Bellman-Ford Example



B-F equation:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

From u's perspective estimated:

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

Node achieving minimum is the next hop in shortest path, used in forwarding table

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:

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

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

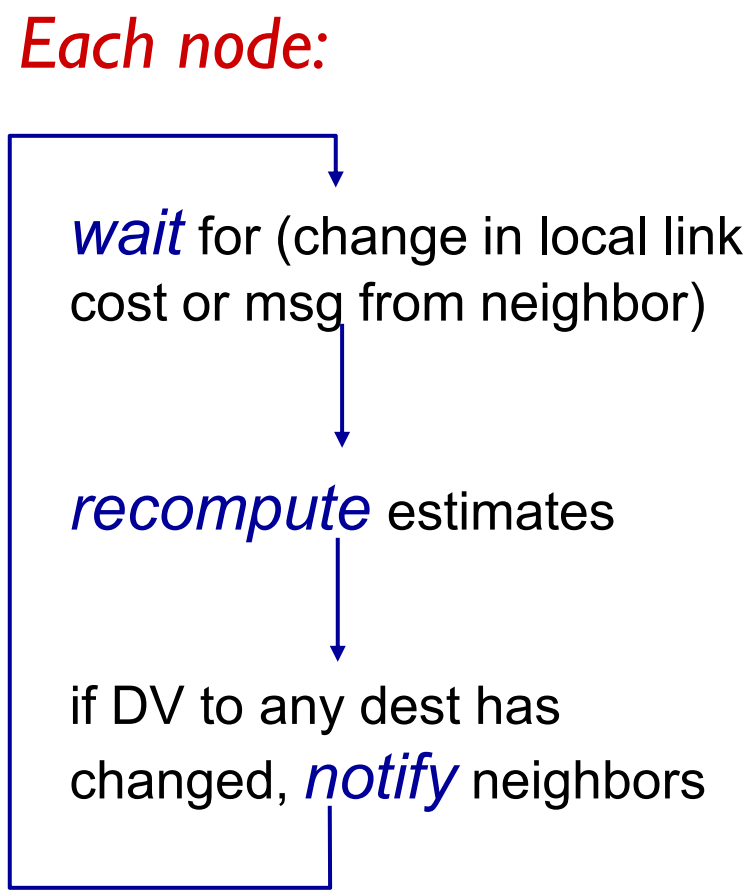
Distance Vector Algorithm

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
  - Its neighbors then further notify their neighbors if necessary



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

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

node x table

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

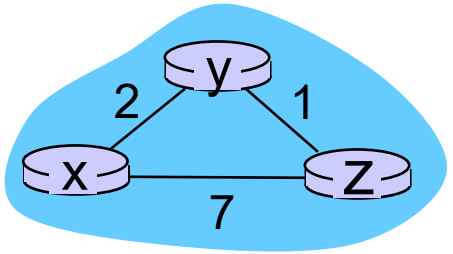
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

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

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

node x table

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

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

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

node y table

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

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

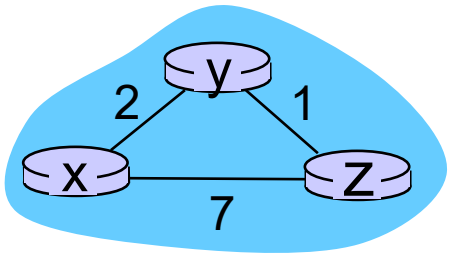
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

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

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



time