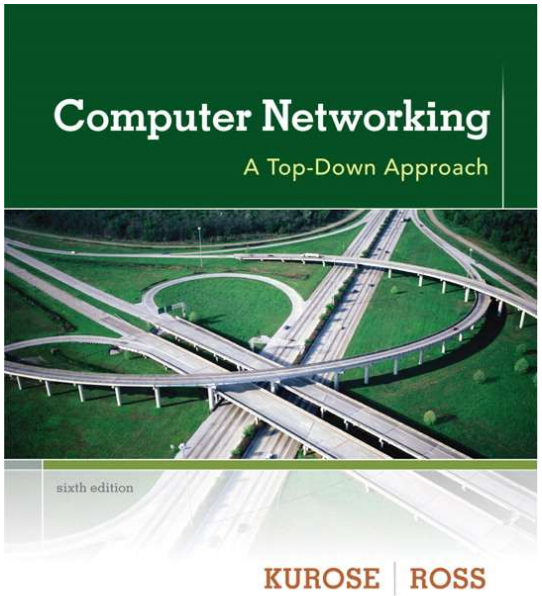
Chapter 4
Network Layer



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chapter 4: network layer

chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast
- instantiation, implementation in the Internet

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

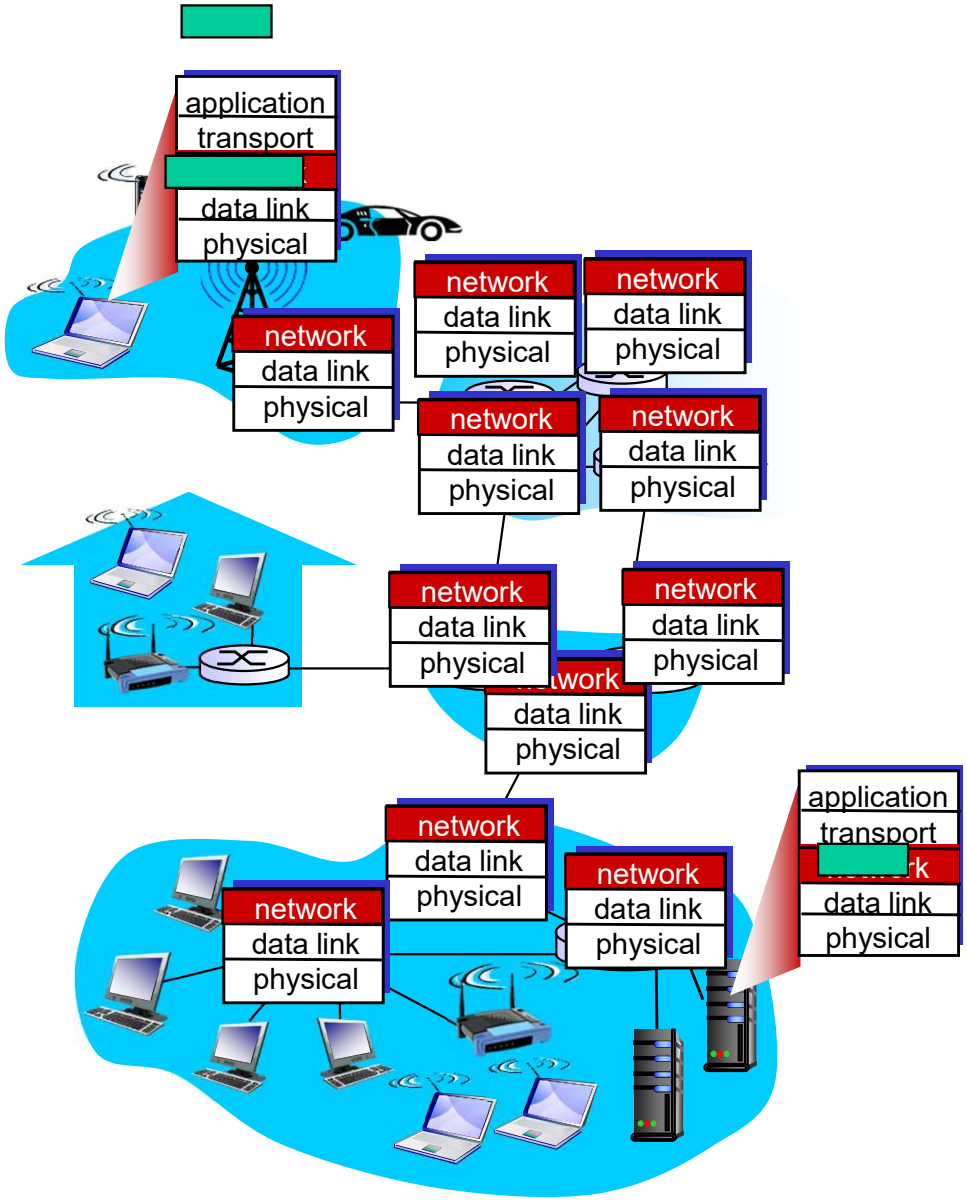
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



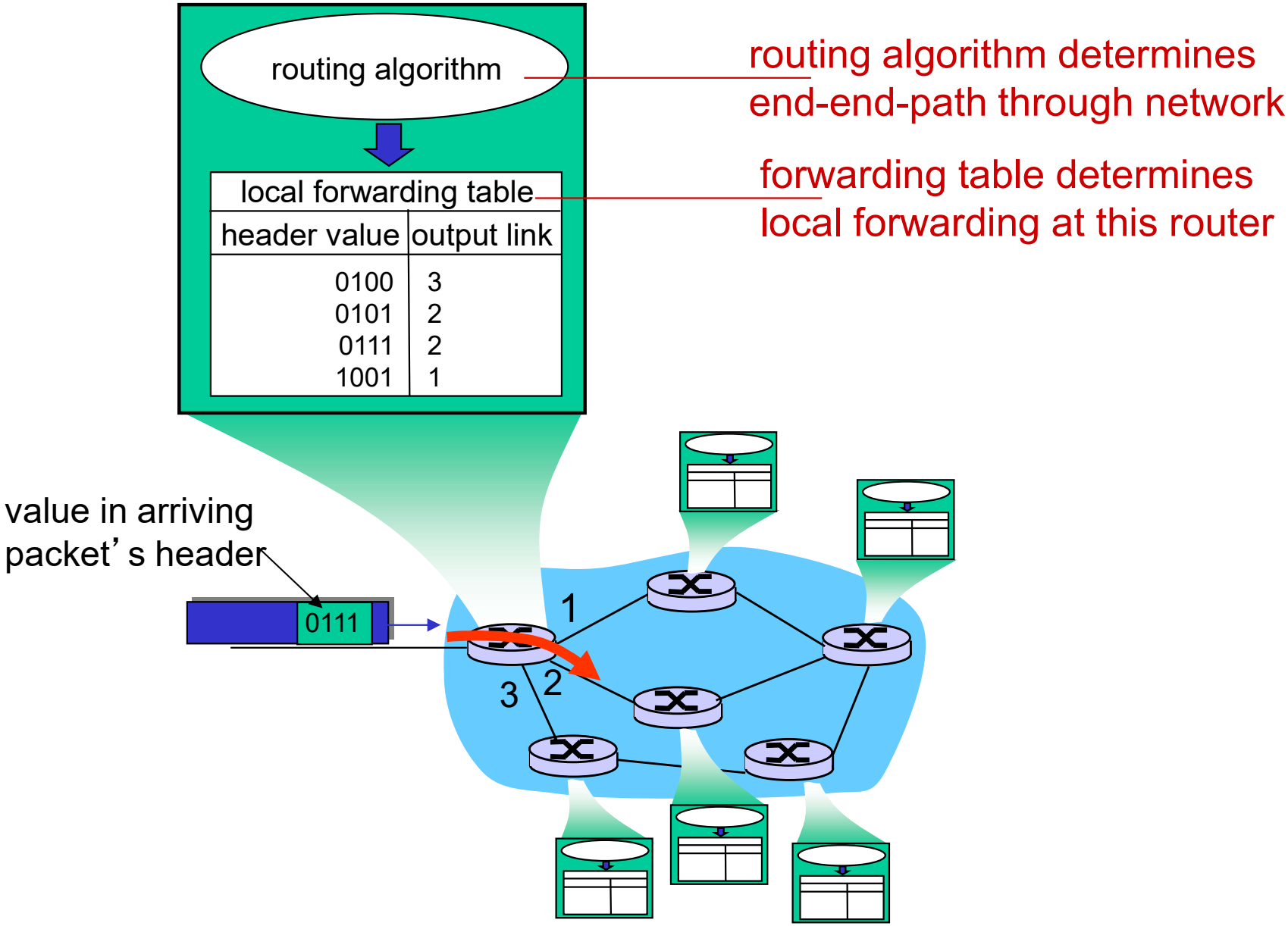
Two key network-layer functions

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

analogy:

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

Interplay between routing and forwarding



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

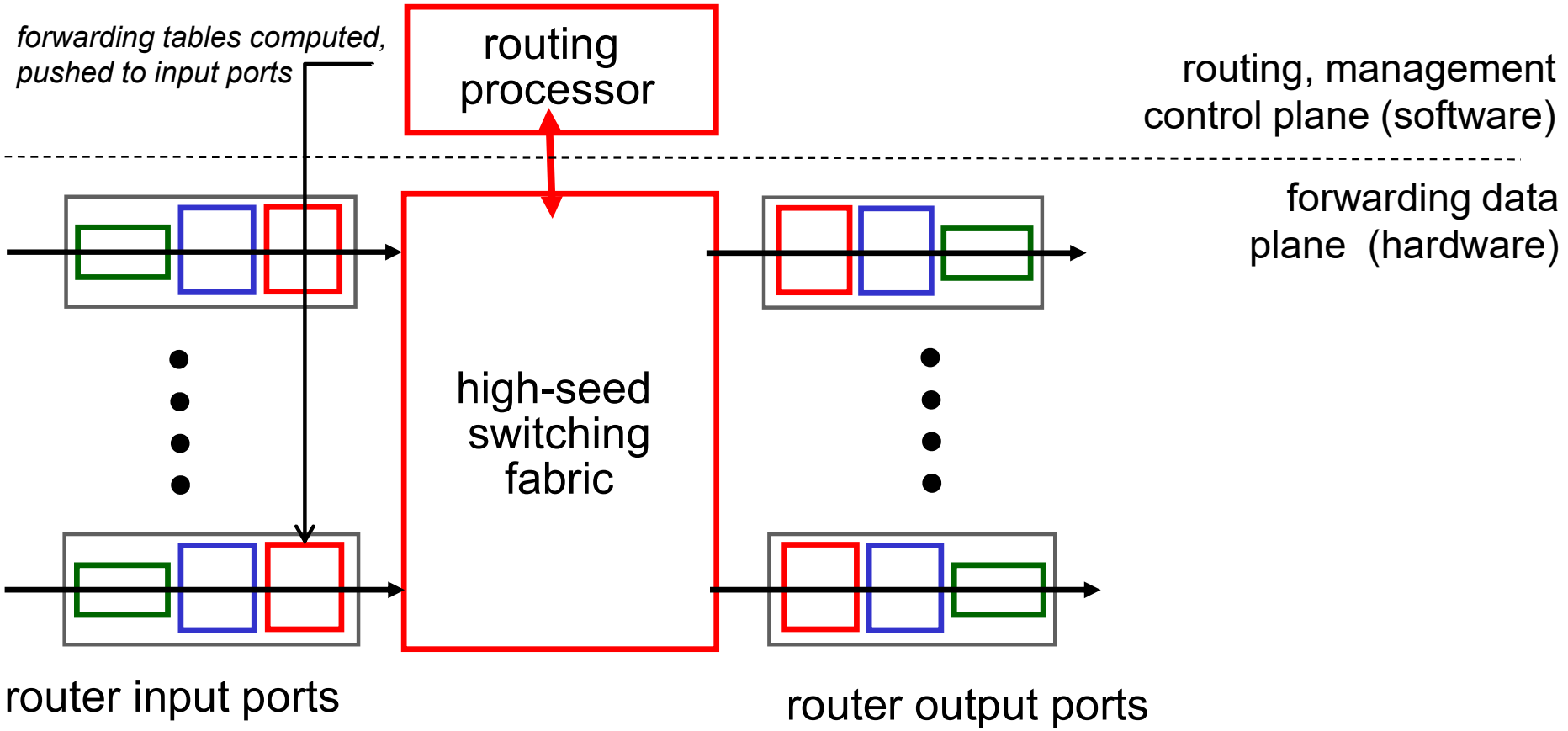
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

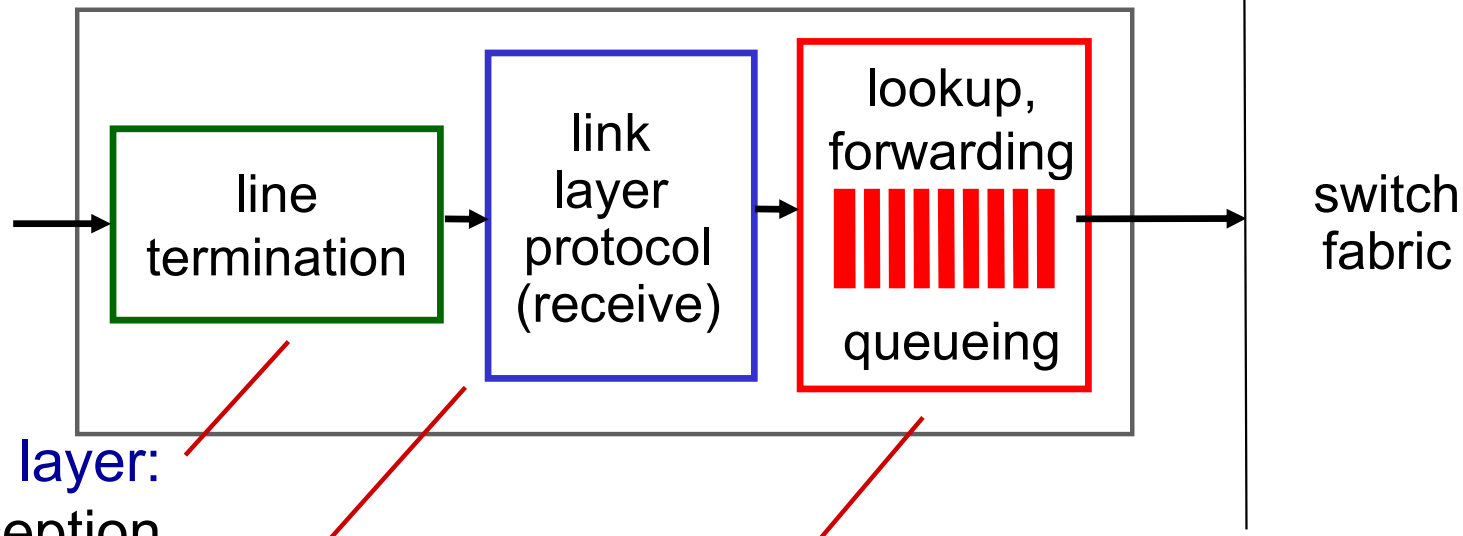
Router architecture overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Input port functions



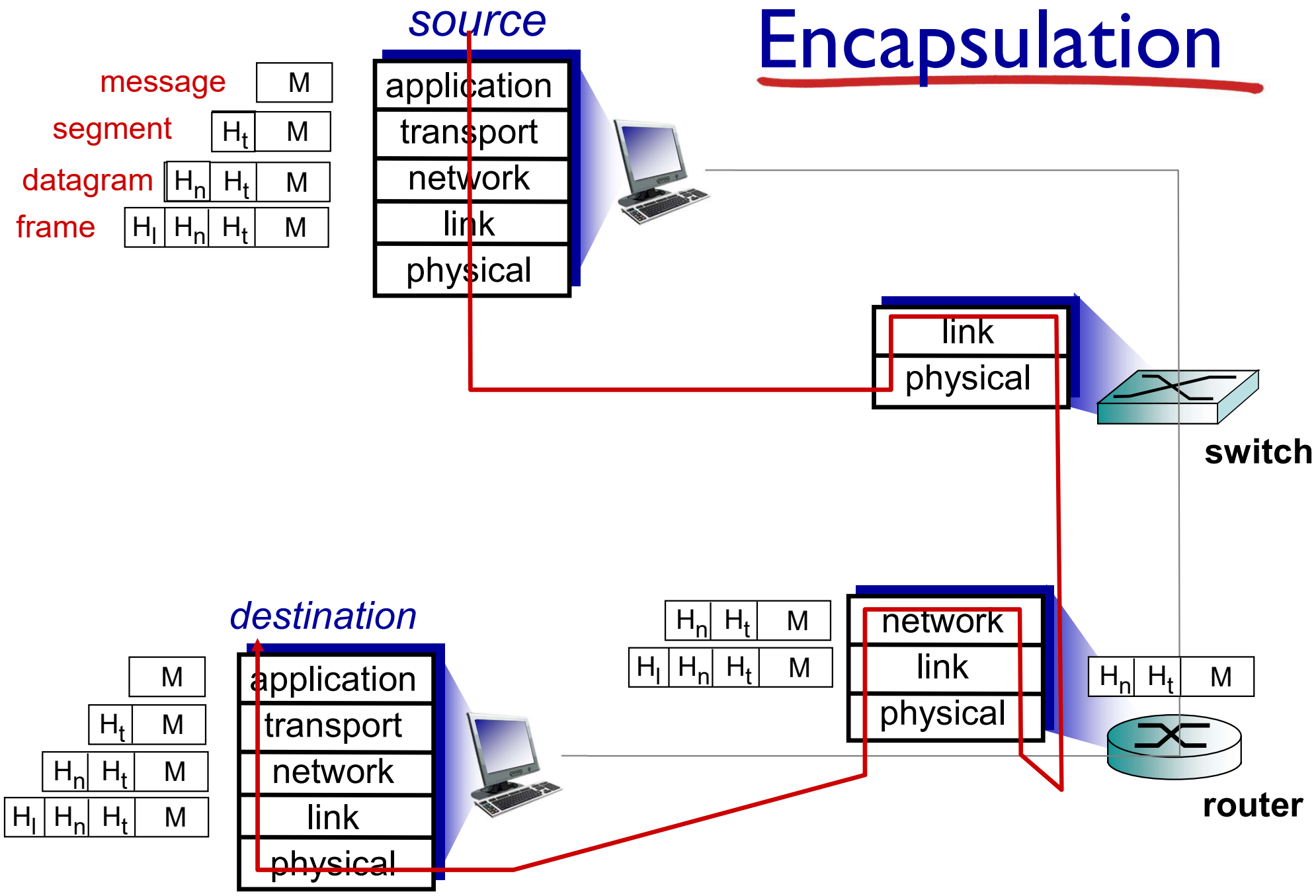
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

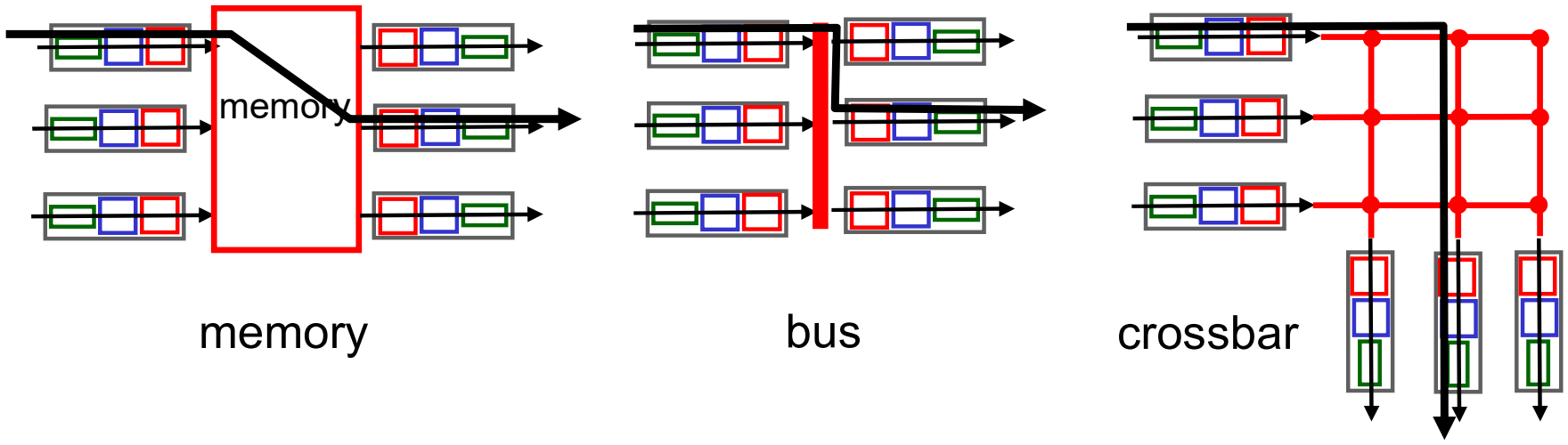
- given datagram dest., lookup output port using forwarding table in input port memory (*"match plus action"*)
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Encapsulation



Switching fabrics

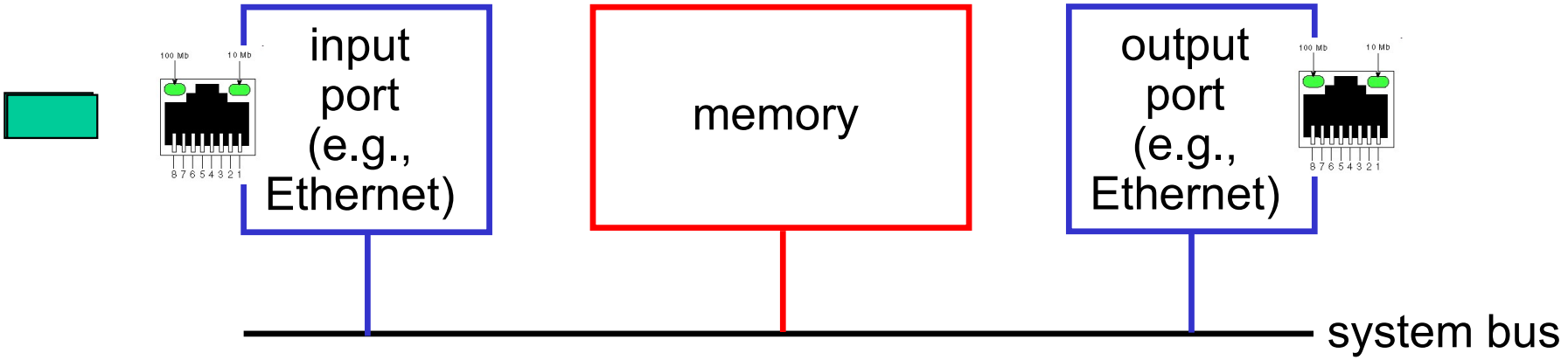
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

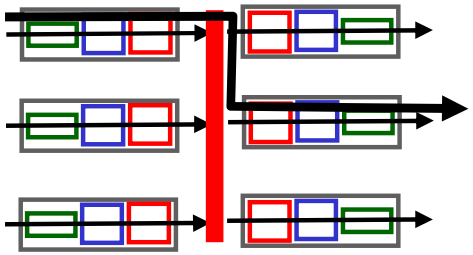
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via a bus

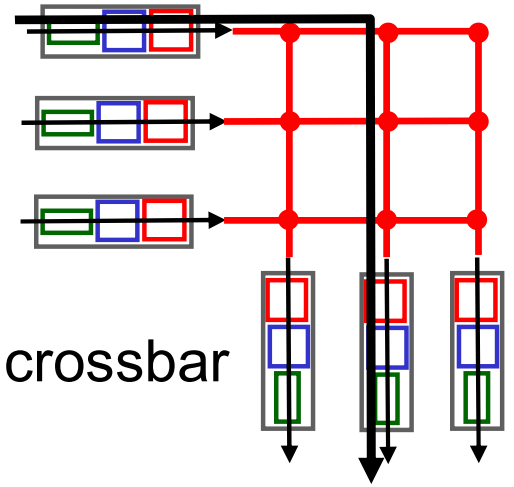
- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



bus

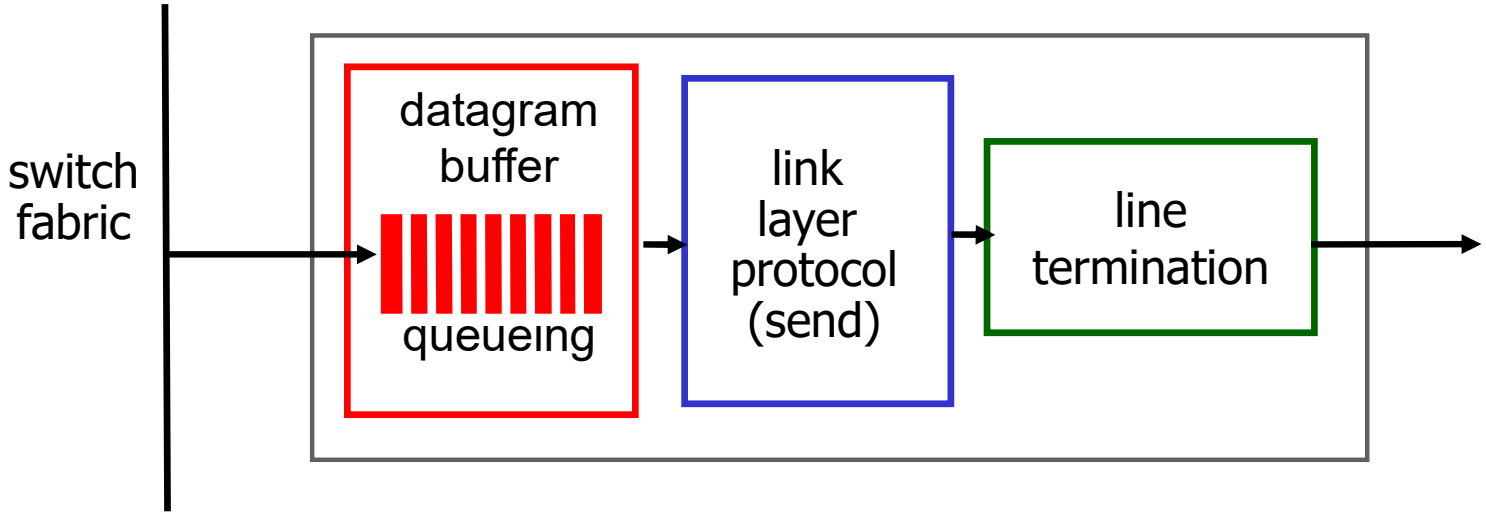
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network



Output ports

This slide in HUGELY important!



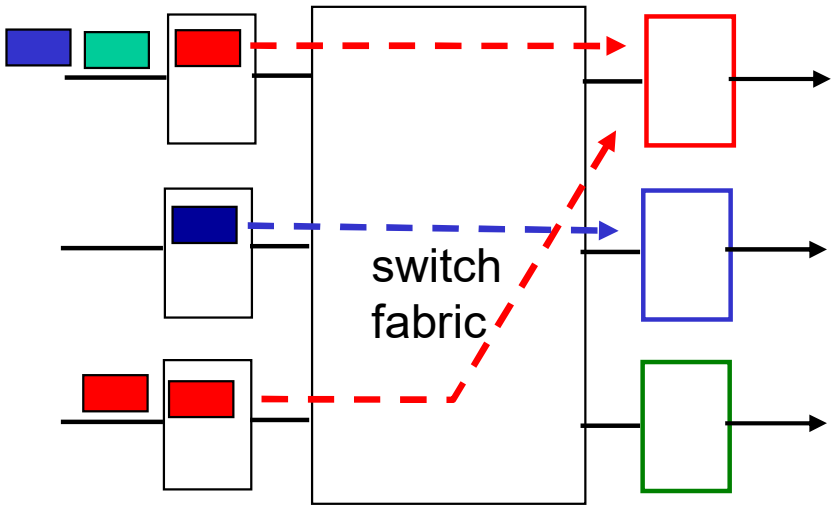
- *buffering* required from fabric faster rate

Datagram (packets) can be lost due to congestion, lack of buffers

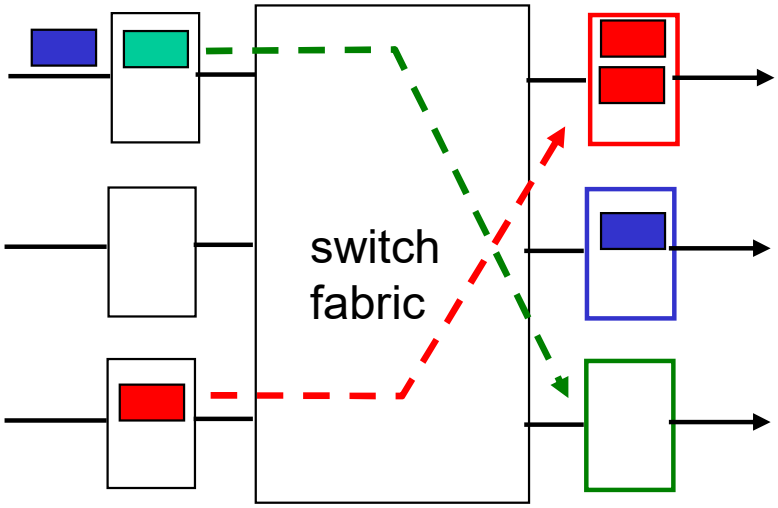
- *scheduling* datagrams

Priority scheduling – who gets best performance, network neutrality

Output port queueing



at *t*, packets more from input to output



one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

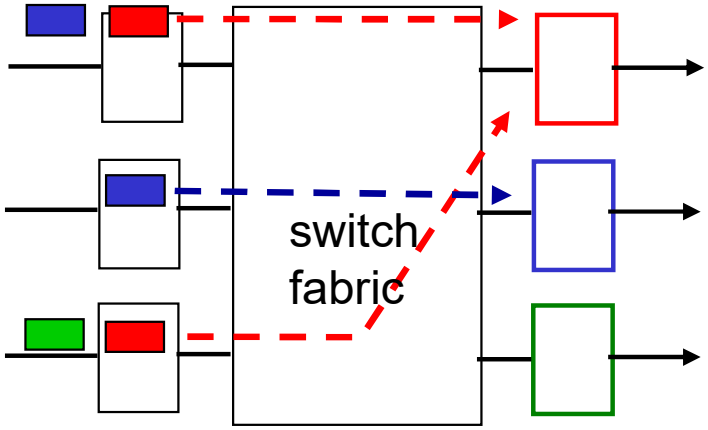
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with *N* flows, buffering equal to

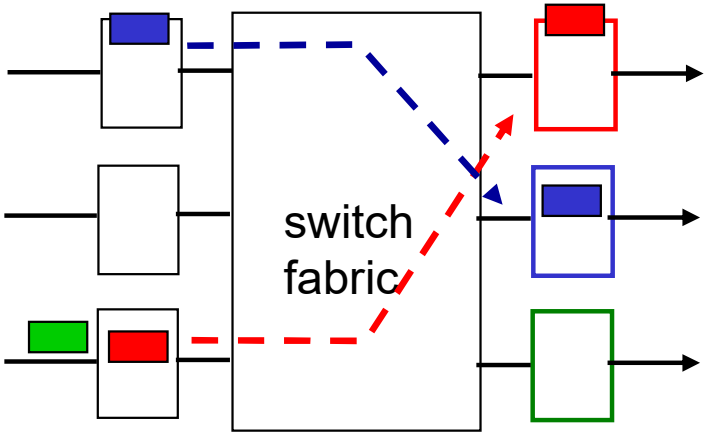
$$\frac{RTT \cdot C}{\sqrt{N}}$$

Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
  - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet
experiences HOL
blocking

Exercise : Datagram Loss

- ให้สรุปว่า Datagram Loss จากที่ใดได้บ้าง และ เกิดจากอะไร

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

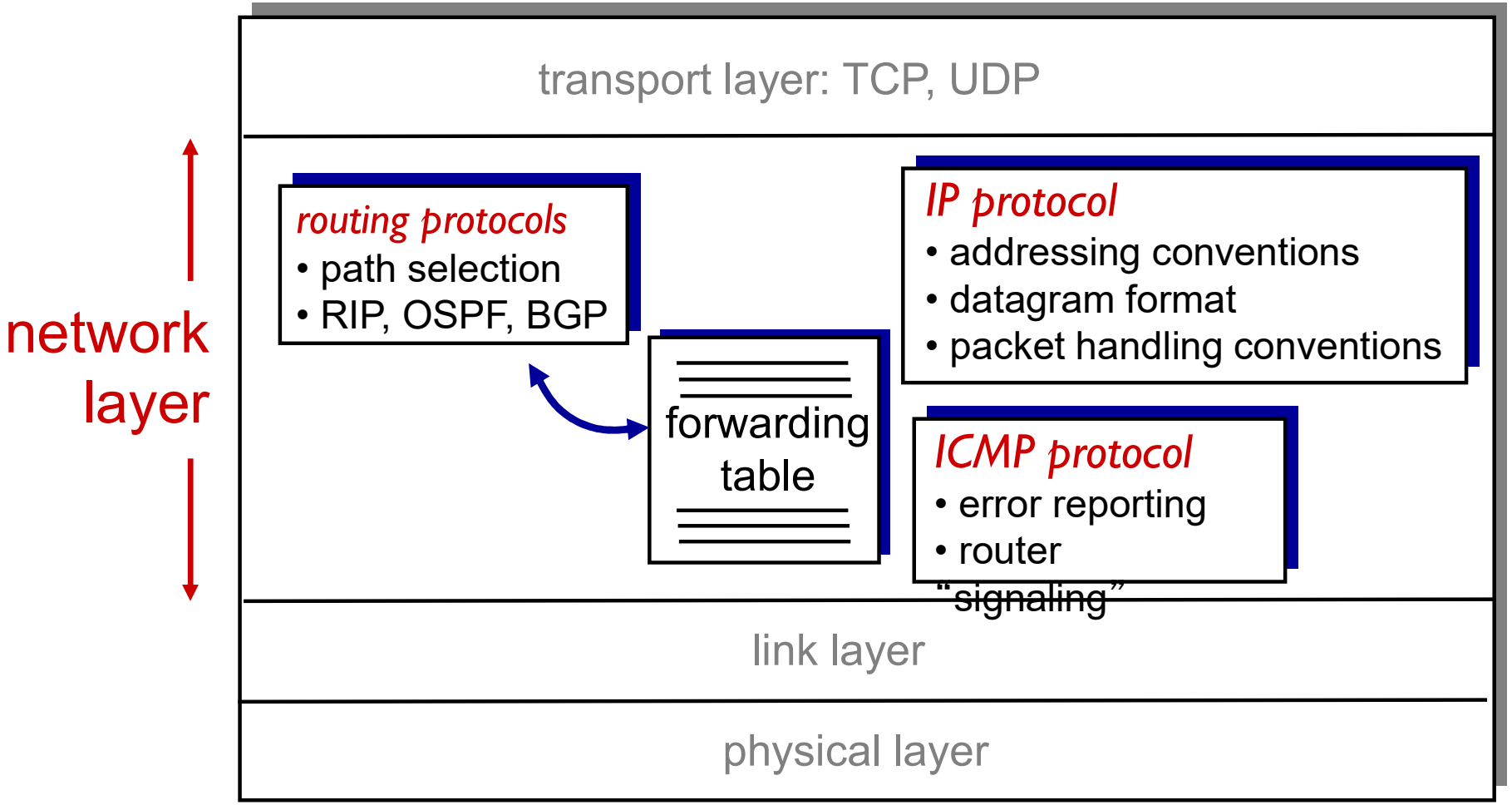
4.6 routing in the Internet

- RIP
- OSPF
- BGP

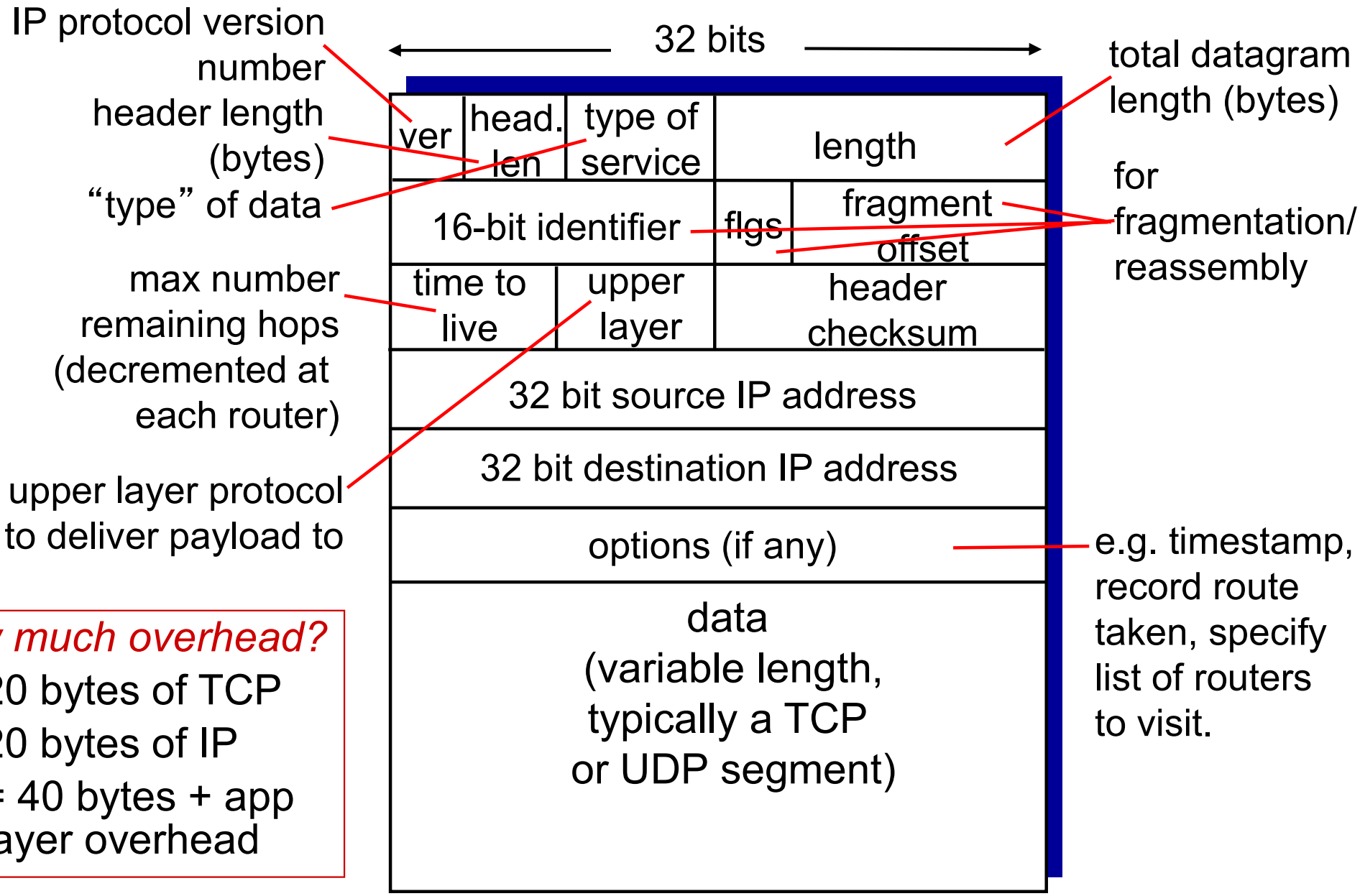
4.7 broadcast and multicast routing

The Internet network layer

host, router network layer functions:



IP datagram format

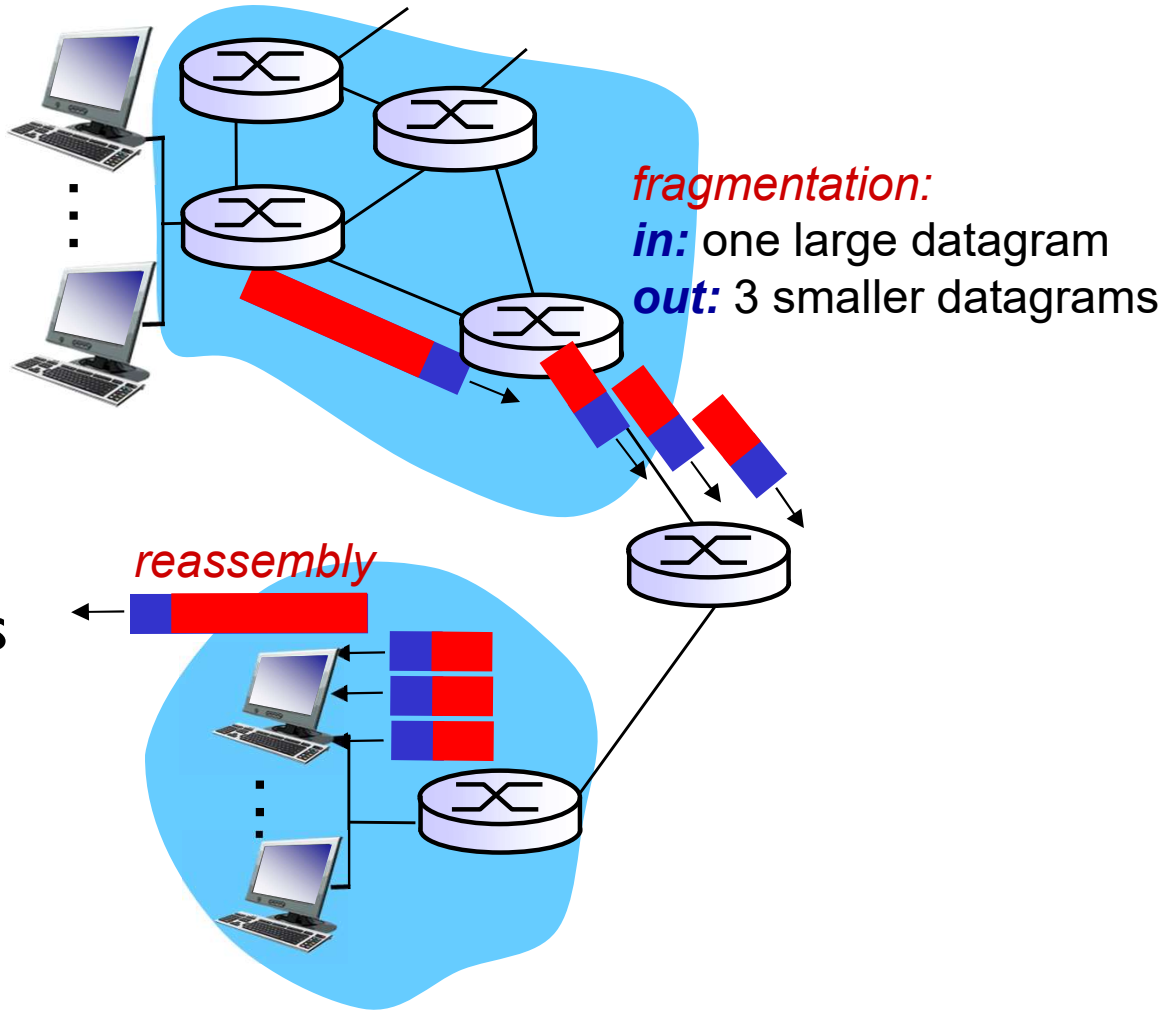


how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP fragmentation, reassembly

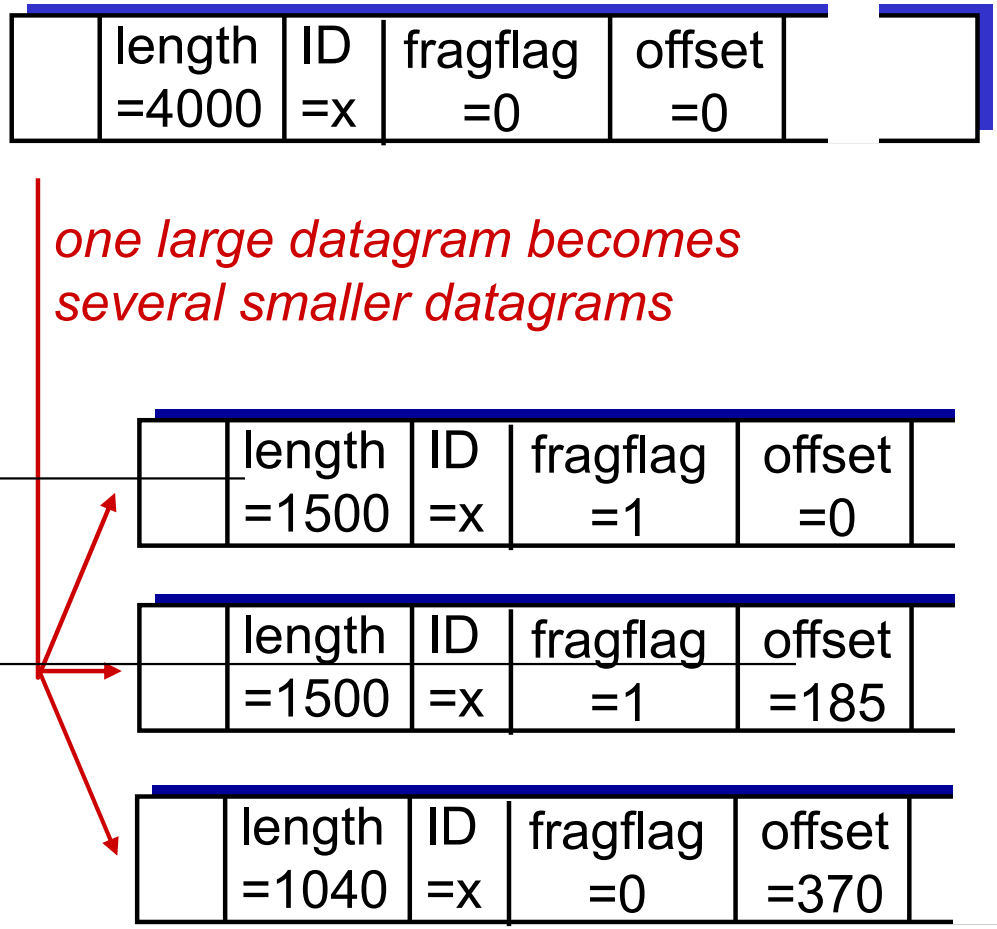
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

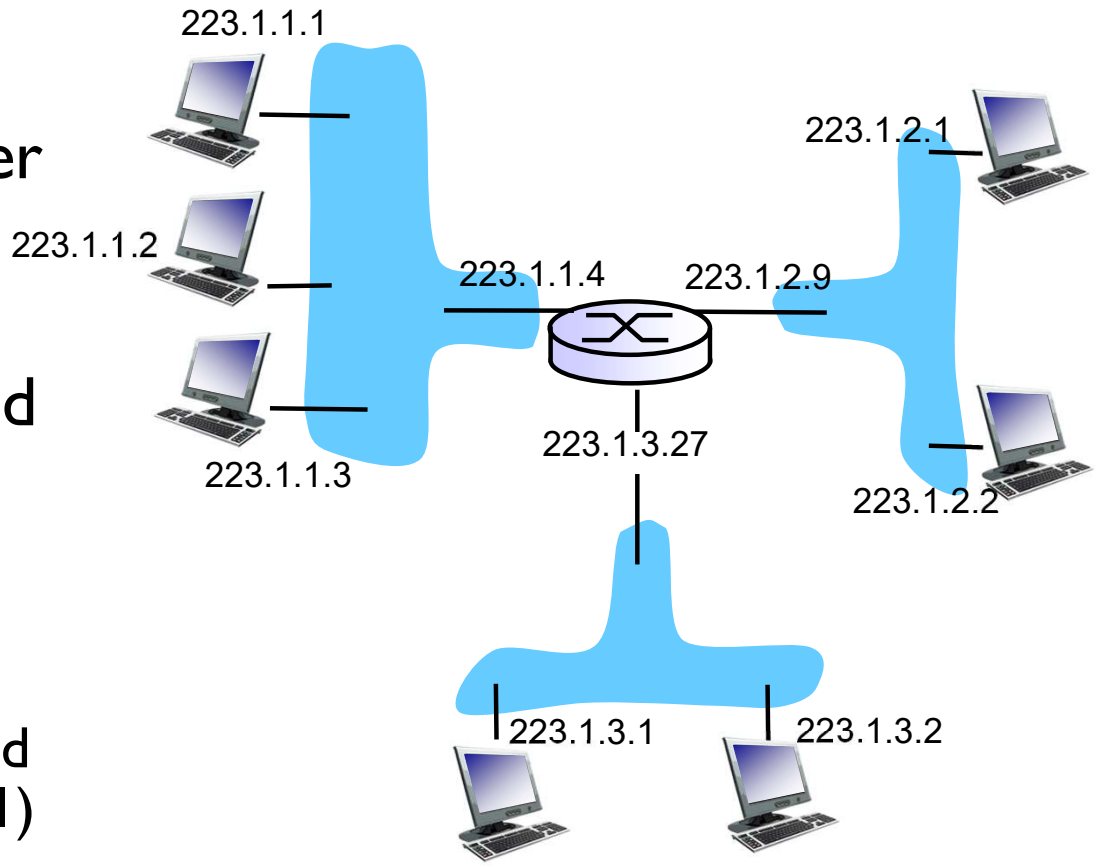
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

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*



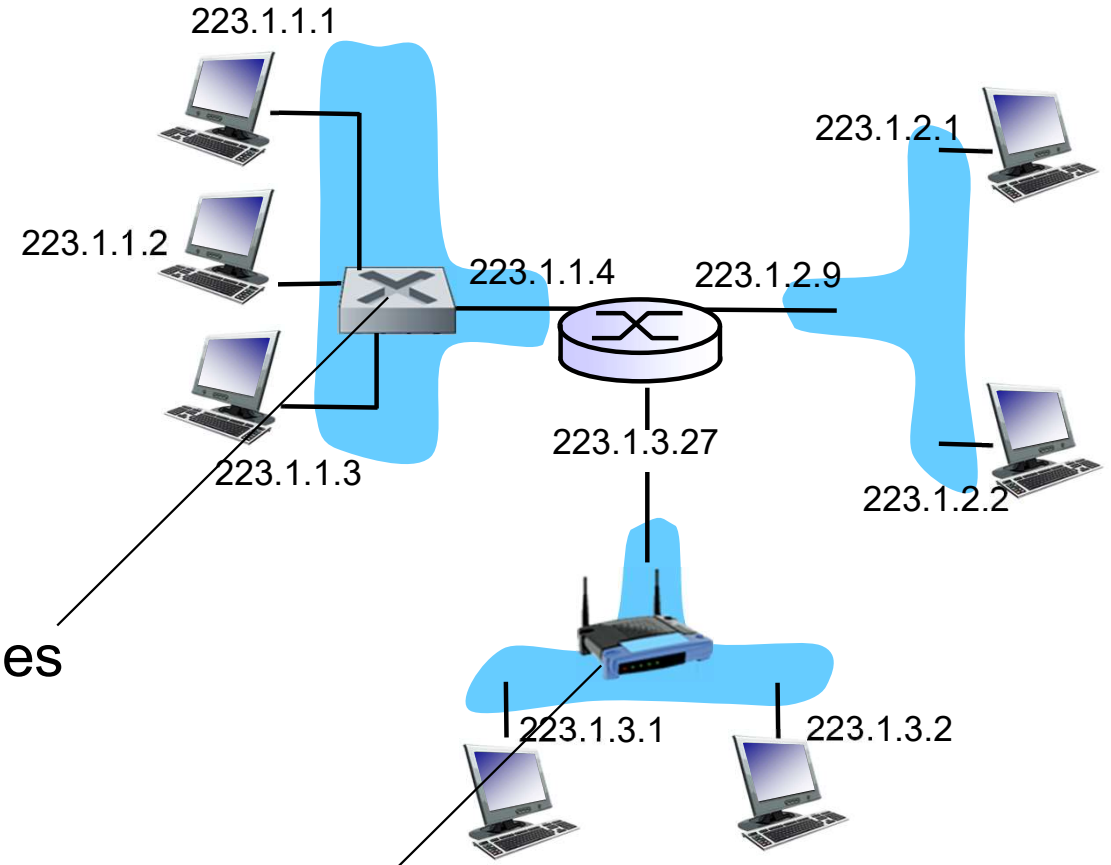
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 5, 6.



A: wired Ethernet interfaces connected by Ethernet switches

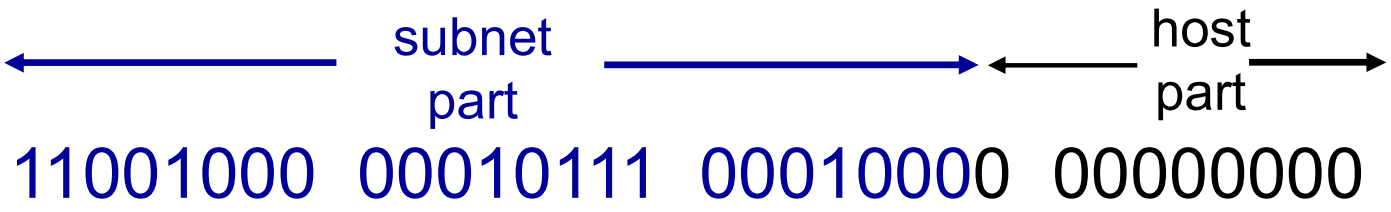
For now: don't need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station

IP addressing: CIDR

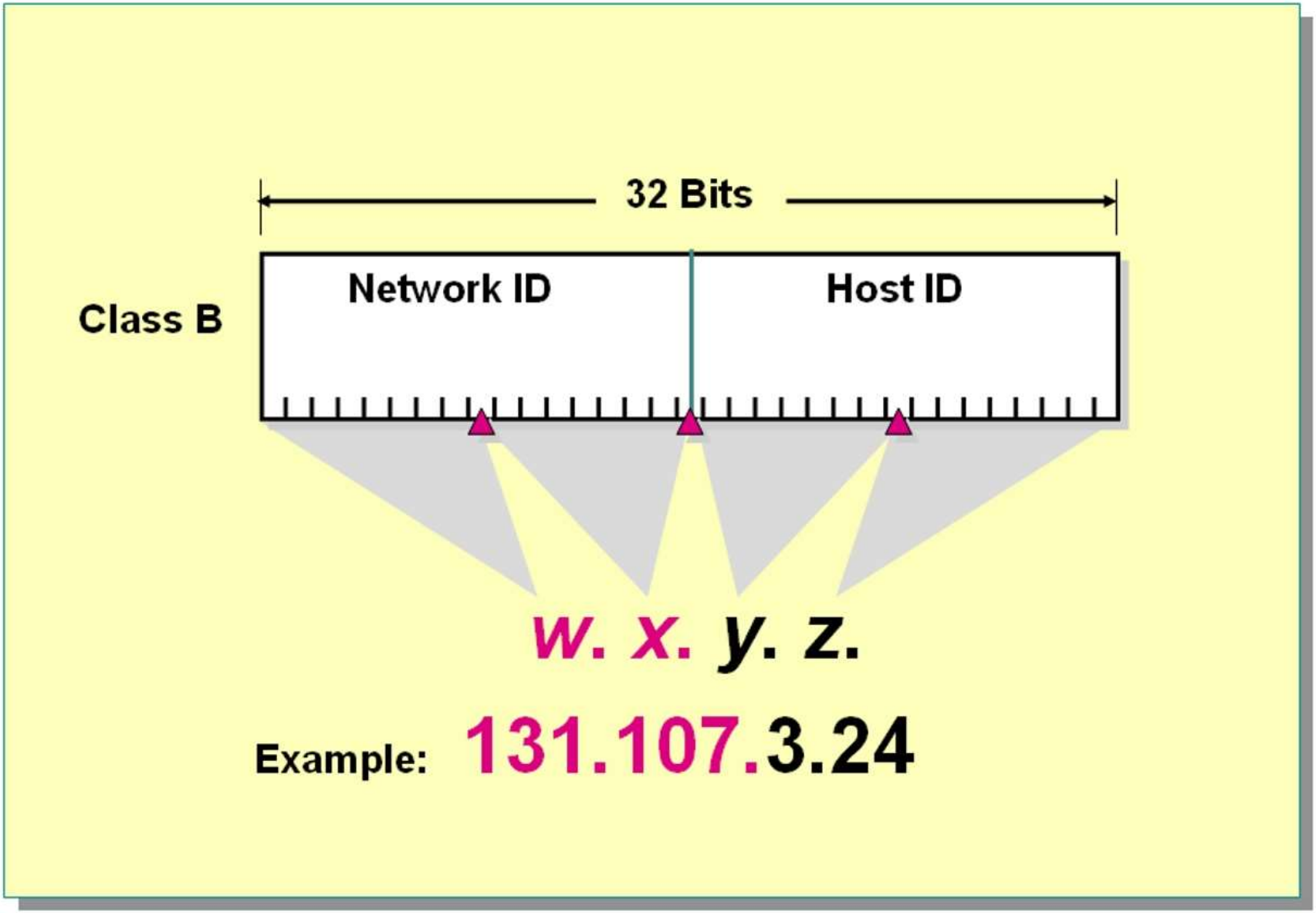
CIDR: Classless InterDomain Routing

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

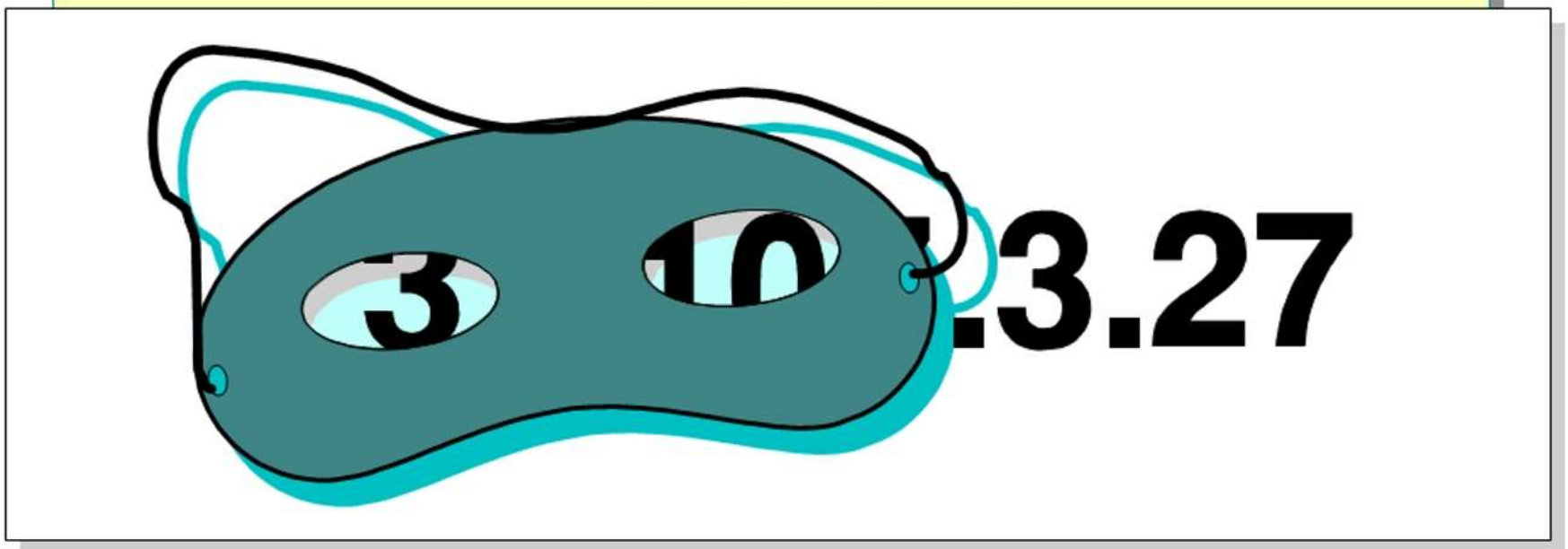


200.23.16.0/23

Network ID



Subnet Mask



- Distinguishes the Network ID from the Host ID
- Used to Specify Whether the Destination Host is Local or Remote

Finding Network ID

- **Local and Destination Host's Subnet Masks Are ANDed**
  - 1 AND 1 = 1
  - Other combinations = 0
  - If ANDed results of source and destination hosts match, the destination is local.

IP Address	10011111	11100000	00000111	10000001
Subnet Mask	11111111	11111111	00000000	00000000
Result	10011111	11100000	00000000	00000000

Example : Finding Network ID

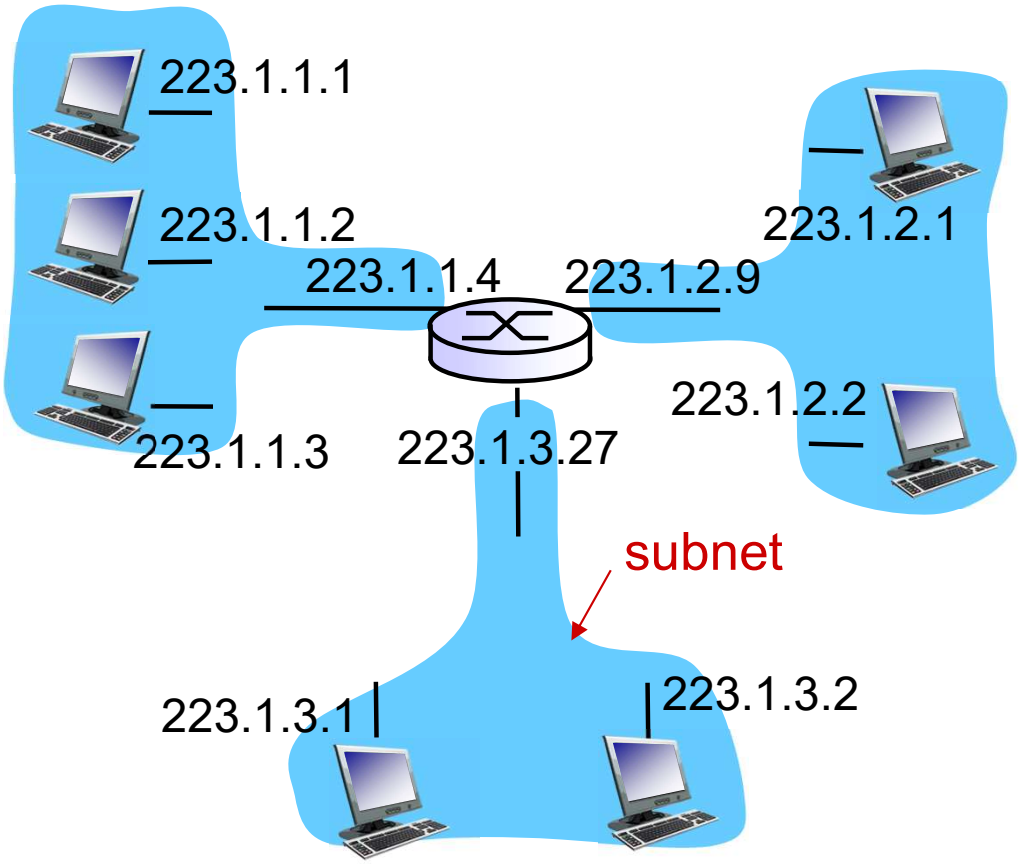
- กำหนด IP Address 201.180.56.5 ให้หา Network ID สำหรับ /28 (เขียนอีกแบบ 255.255.255.240)
  - แปลงเป็นฐาน 2 ได้เป็น 11001001.10110100.00111000.00000101
  - แบ่งเป็น 28 กับ 4
  - 11001001.10110100.00111000.0000 0101
  - เติม 0 ให้ครบ 32 บิตเหมือนเดิม
  - 11001001.10110100.00111000.00000000
  - Network ID = 201.180.56.0, Host ID = 5

Exercise : Finding Network ID

- กำหนด IP Address 23.56.7.91 ให้หา Network ID สำหรับ /8 /12 /20 /22

Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *what's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

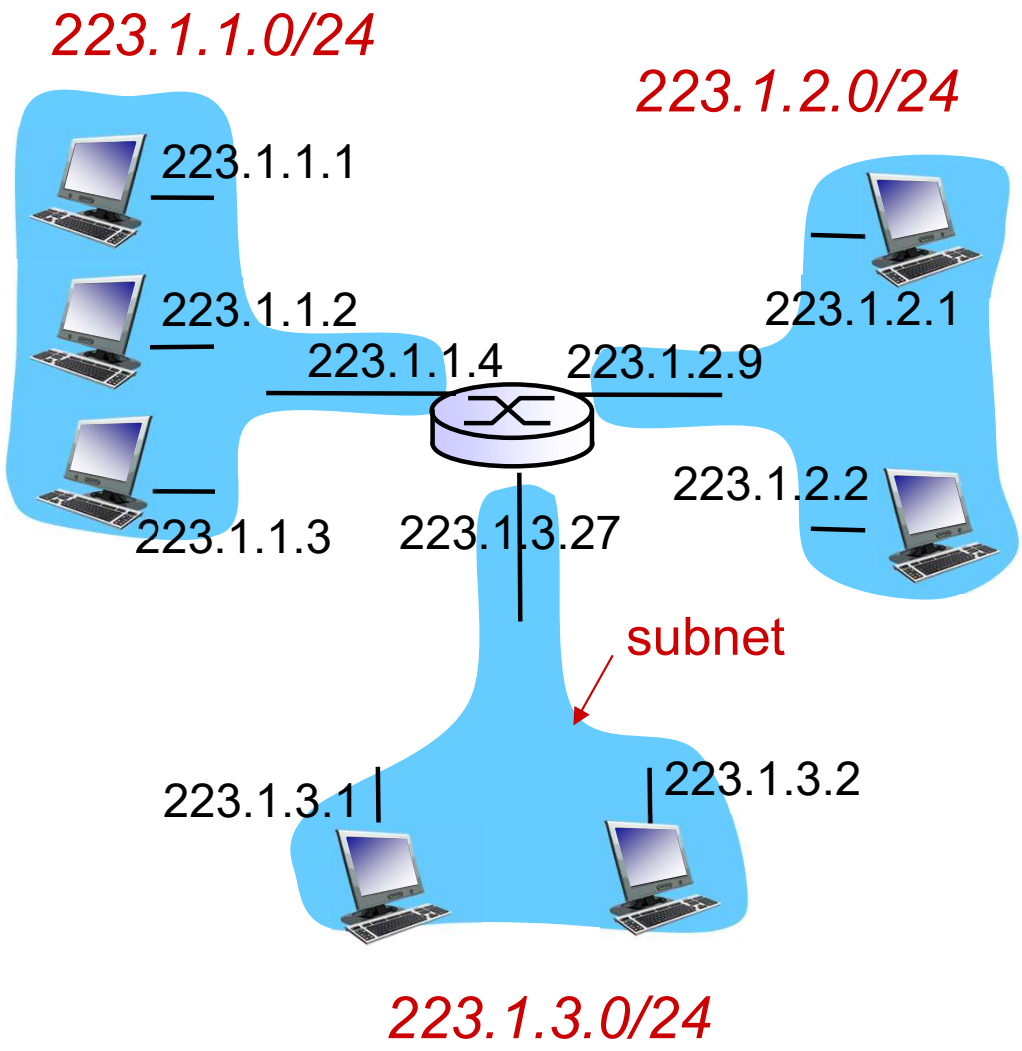


network consisting of 3 subnets

Subnets

recipe

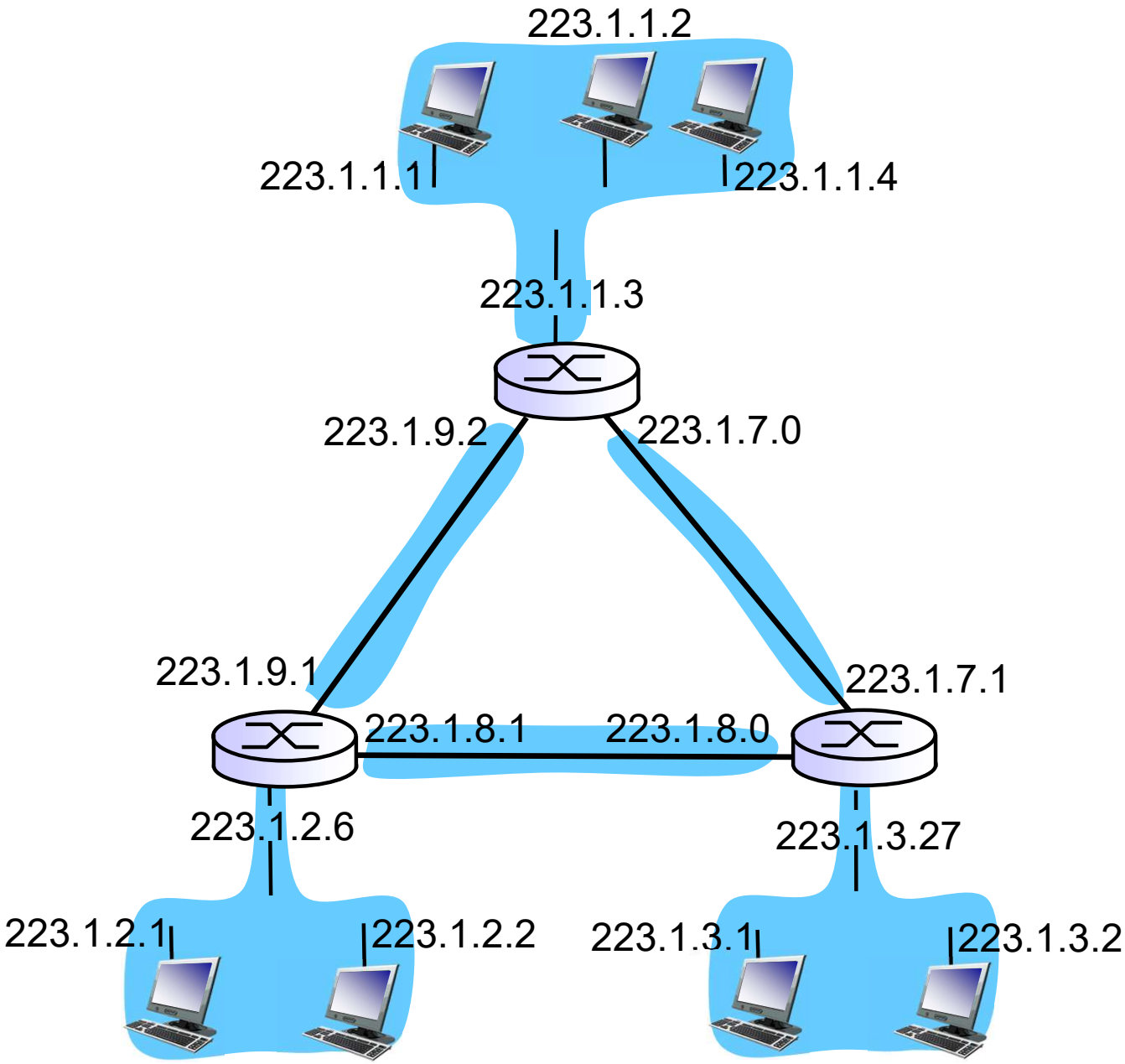
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

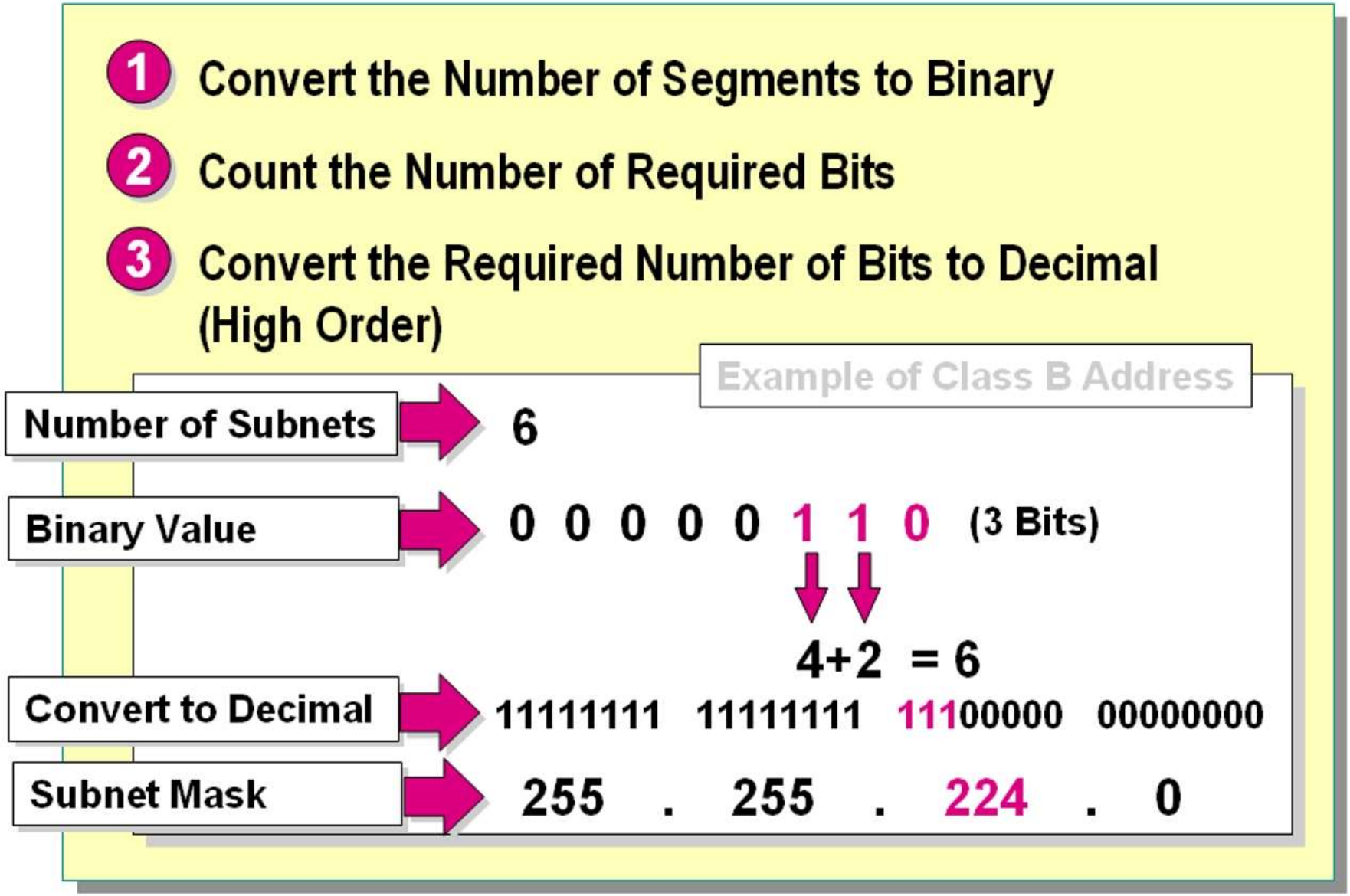
how many?



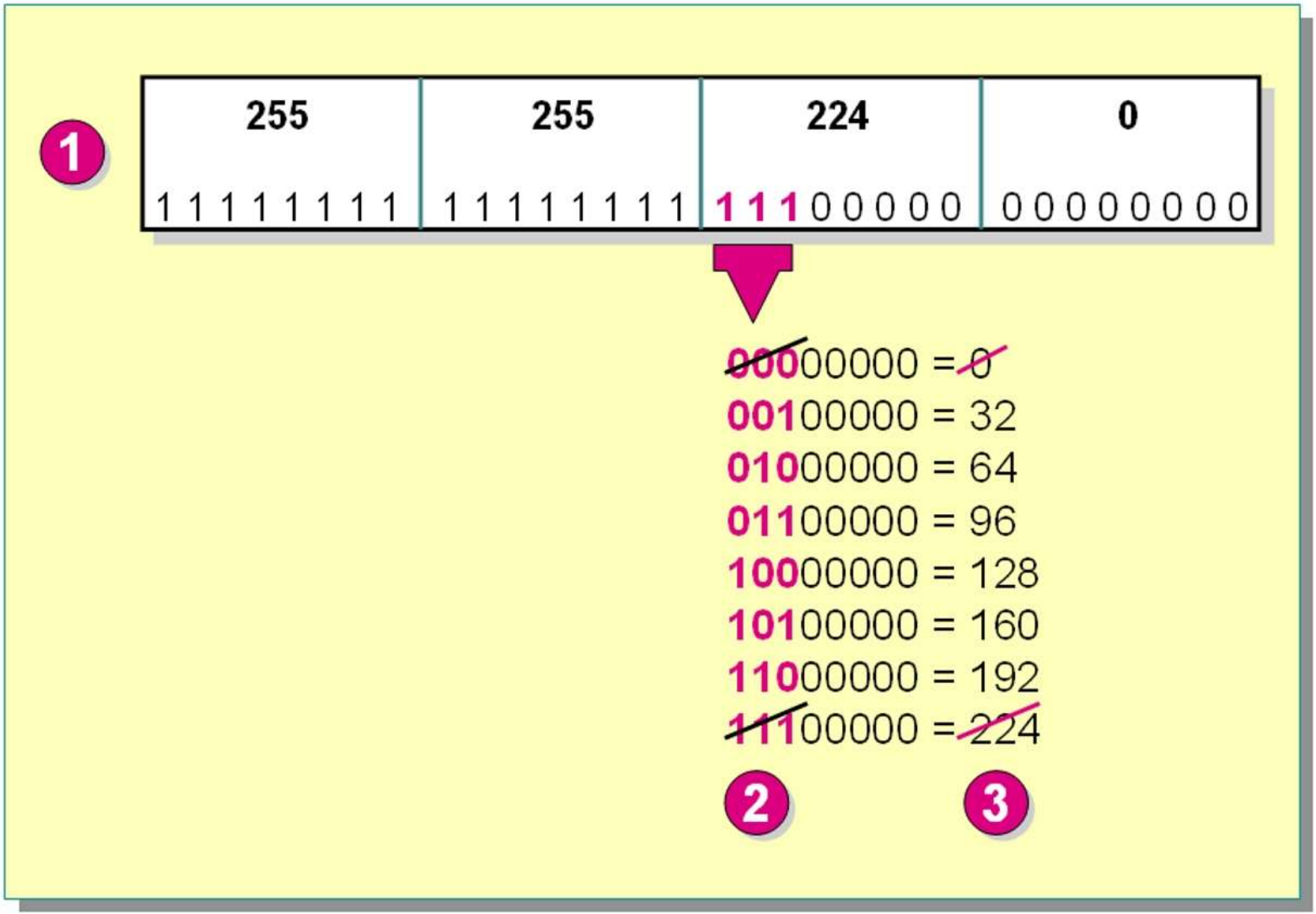
Subnets

- **Determine the Number of Required Network IDs**
  - One for each subnet
  - One for each wide-area network connection
- **Determine the Number of Required Host IDs per Subnet**
  - One for each TCP/IP host
  - One for each router interface
- **Define One Subnet Mask Based on Requirements**
- **Define a Unique Subnet ID for Each Physical Segment Based on the Subnet Mask**
- **Define Valid Host IDs for Each Subnet Based on the Subnet ID**

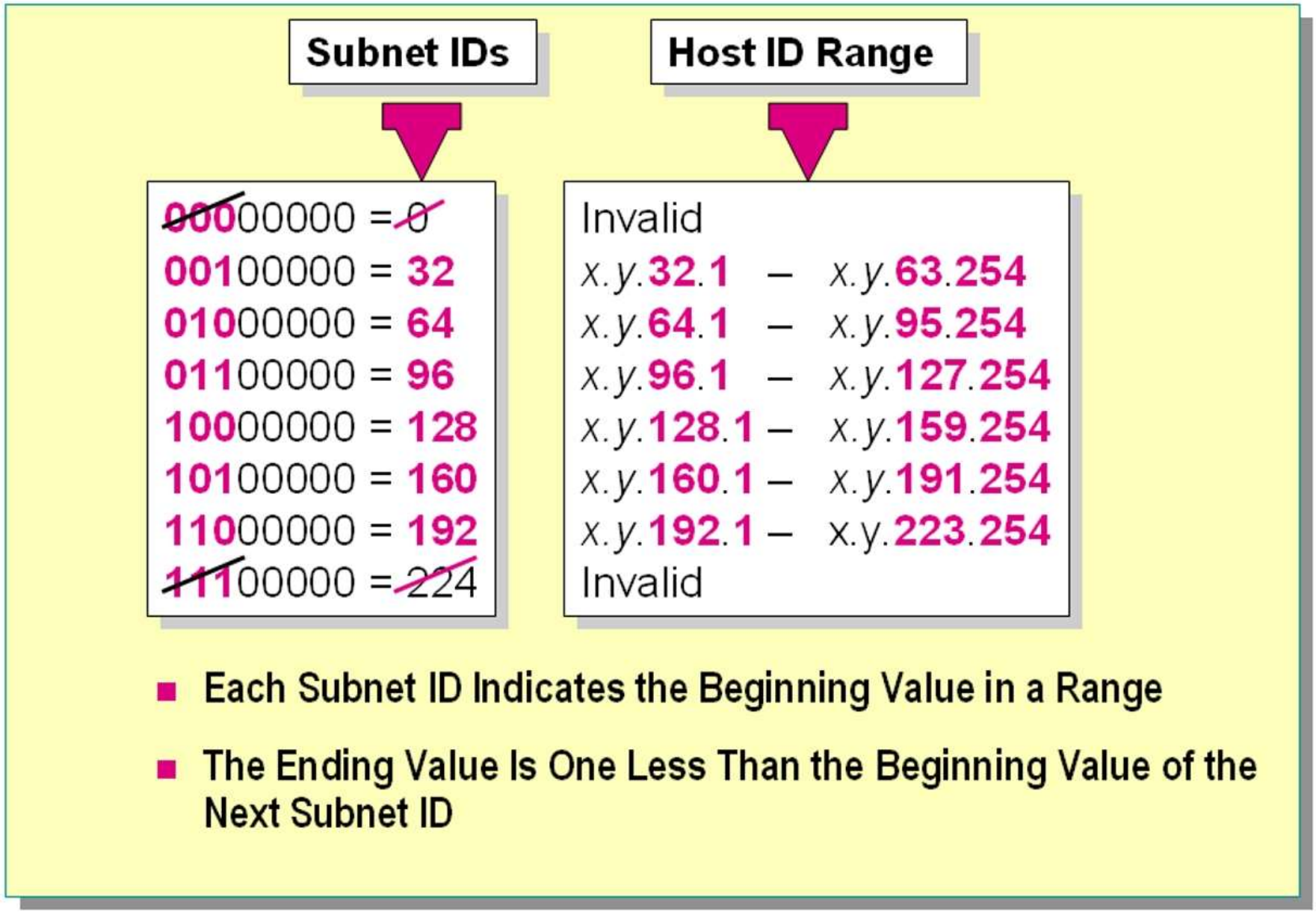
Subnets


1 Convert the Number of Segments to Binary
2 Count the Number of Required Bits
3 Convert the Required Number of Bits to Decimal (High Order)
Example of Class B Address
Number of Subnets
6
Binary Value
0 0 0 0 0 1 1 0 (3 Bits)
4+2 = 6
Convert to Decimal
11111111 11111111 11100000 00000000
Subnet Mask
255 . 255 . 224 . 0

Subnets


1
255
255
224
0
11111111
11111111
11100000
00000000
00000000 = 0
00100000 = 32
01000000 = 64
01100000 = 96
10000000 = 128
10100000 = 160
11000000 = 192
11100000 = 224
2
3

Subnets



- Each Subnet ID Indicates the Beginning Value in a Range
- The Ending Value Is One Less Than the Beginning Value of the Next Subnet ID

Subnets

/n	Mask	/n	Mask	/n	Mask	/n	Mask
/1	128.0.0.0	/9	255.128.0.0	/17	255.255.128.0	/25	255.255.255.128
/2	192.0.0.0	/10	255.192.0.0	/18	255.255.192.0	/26	255.255.255.192
/3	224.0.0.0	/11	255.224.0.0	/19	255.255.224.0	/27	255.255.255.224
/4	240.0.0.0	/12	255.240.0.0	/20	255.255.240.0	/28	255.255.255.240
/5	248.0.0.0	/13	255.248.0.0	/21	255.255.248.0	/29	255.255.255.248
/6	252.0.0.0	/14	255.252.0.0	/22	255.255.252.0	/30	255.255.255.252
/7	254.0.0.0	/15	255.254.0.0	/23	255.255.254.0	/31	255.255.255.254
/8	255.0.0.0	/16	255.255.0.0	/24	255.255.255.0	/32	255.255.255.255

Example : Subnetting

- กำหนด Network Address 161.246.5.0 ต้องการแบ่งเป็น 5 Subnet ให้หา Network ID ของแต่ละ Subnet และ IP Address ที่ใช้ได้สำหรับแต่ละ Subnet
- 5 = 101 ดังนั้น subnet mask = 11111111.11111111.11111111.11100000 = 255.255.255.224 หรือ /27
  - เครือข่ายที่ 1 = 161.246.5.33-161.246.5.62 (ID = 161.246.5.32)
  - เครือข่ายที่ 2 = 161.246.5.65-161.246.5.94 (ID = 161.246.5.64)
  - เครือข่ายที่ 3 = 161.246.5.97-161.246.5.126 (ID = 161.246.5.96)
  - เครือข่ายที่ 4 = 161.246.5.129-161.246.5.158 (ID = 161.246.5.128)
  - เครือข่ายที่ 5 = 161.246.5.161-161.246.5.190 (ID = 161.246.5.160)

Exercise : Subnetting

- กำหนด Network Address 161.246.5.0 ต้องการแบ่งเป็น 14 Subnet ให้หา Network ID ของแต่ละ Subnet และ IP Address ที่ใช้ได้สำหรับแต่ละ Subnet

Exercise : Subnetting

All 16 of the Possible /28 Networks for 161.246.5.*

Network Address	Usable Host Range	Broadcast Address:
161.246.5.0	161.246.5.1 - 161.246.5.14	161.246.5.15
161.246.5.16	161.246.5.17 - 161.246.5.30	161.246.5.31
161.246.5.32	161.246.5.33 - 161.246.5.46	161.246.5.47
161.246.5.48	161.246.5.49 - 161.246.5.62	161.246.5.63
161.246.5.64	161.246.5.65 - 161.246.5.78	161.246.5.79
161.246.5.80	161.246.5.81 - 161.246.5.94	161.246.5.95
161.246.5.96	161.246.5.97 - 161.246.5.110	161.246.5.111
161.246.5.112	161.246.5.113 - 161.246.5.126	161.246.5.127
161.246.5.128	161.246.5.129 - 161.246.5.142	161.246.5.143
161.246.5.144	161.246.5.145 - 161.246.5.158	161.246.5.159
161.246.5.160	161.246.5.161 - 161.246.5.174	161.246.5.175
161.246.5.176	161.246.5.177 - 161.246.5.190	161.246.5.191
161.246.5.192	161.246.5.193 - 161.246.5.206	161.246.5.207
161.246.5.208	161.246.5.209 - 161.246.5.222	161.246.5.223
161.246.5.224	161.246.5.225 - 161.246.5.238	161.246.5.239
161.246.5.240	161.246.5.241 - 161.246.5.254	161.246.5.255

Exercise : Subnetting

- กำหนด Network Address 161.246.5.0 ต้องการให้มี Subnet ละ 20 เครื่อง จะแบ่งได้มากที่สุดกี่ Subnet และมี Address อะไรบ้าง

Exercise : Subnetting

- Network Address 161.246.5.0 มี IP 161.246.5.1 – 161.246.5.254 จำนวน 253 เครื่อง
- ต้องการ Subnet ละ 20 เครื่อง = 253 / 20 = 12 แต่การแบ่ง Subnet ที่มีจำนวน < 12 มีเพียง /27 ที่มีจำนวน Subnet มากที่สุด = 8 Subnet

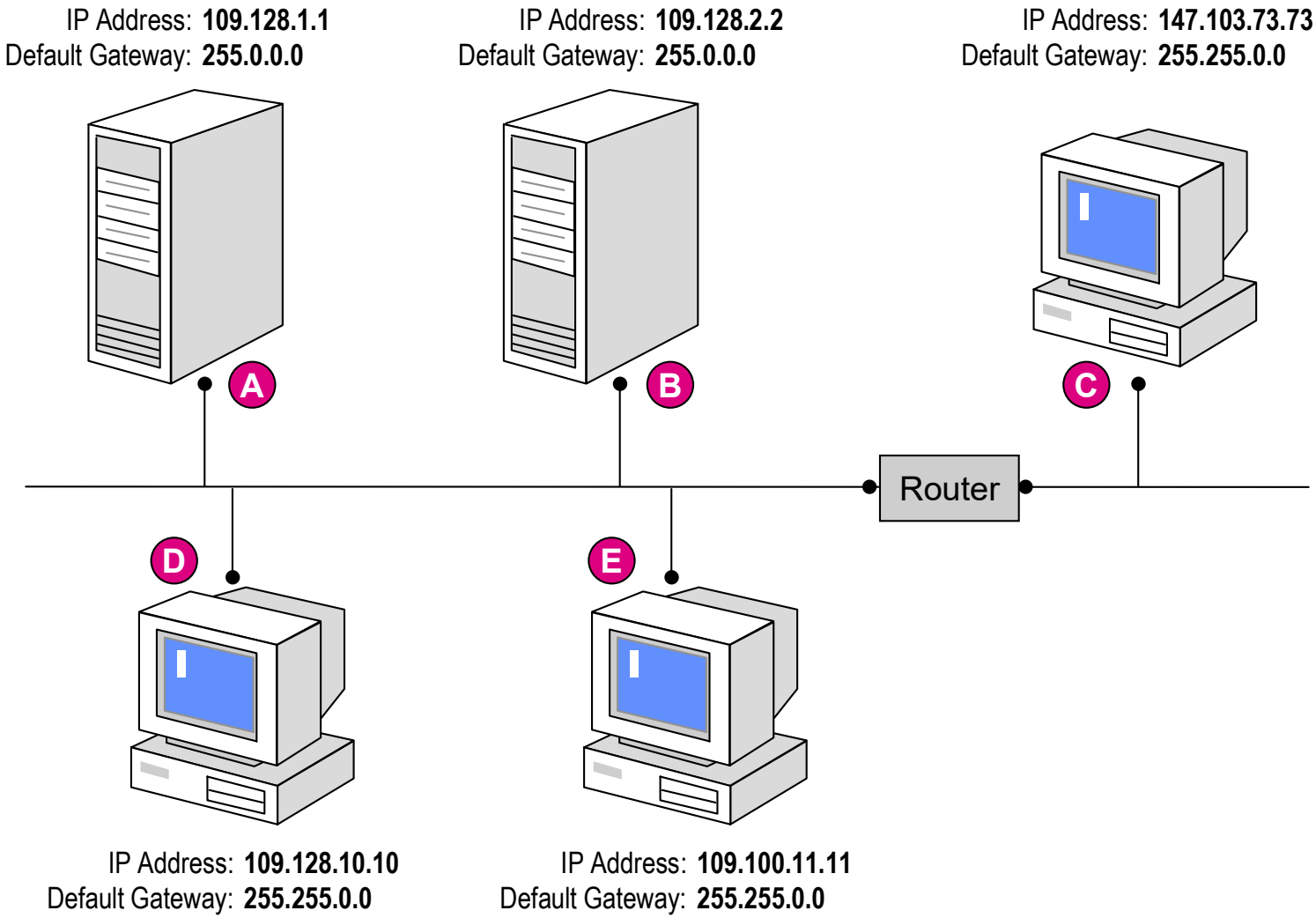
Network Address	Usable Host Range	Broadcast Address:
161.246.5.0	161.246.5.1 - 161.246.5.30	161.246.5.31
161.246.5.32	161.246.5.33 - 161.246.5.62	161.246.5.63
161.246.5.64	161.246.5.65 - 161.246.5.94	161.246.5.95
161.246.5.96	161.246.5.97 - 161.246.5.126	161.246.5.127
161.246.5.128	161.246.5.129 - 161.246.5.158	161.246.5.159
161.246.5.160	161.246.5.161 - 161.246.5.190	161.246.5.191
161.246.5.192	161.246.5.193 - 161.246.5.222	161.246.5.223
161.246.5.224	161.246.5.225 - 161.246.5.254	161.246.5.255

Subnetting

Network Bits	Subnet Mask	Bits Borrowed	Subnets	Hosts/Subnet
16	255.255.0.0	0	0	65534
17	255.255.128.0	1	2	32766
18	255.255.192.0	2	4	16382
19	255.255.224.0	3	8	8190
20	255.255.240.0	4	16	4094
21	255.255.248.0	5	32	2046
22	255.255.252.0	6	64	1022
23	255.255.254.0	7	128	510
24	255.255.255.0	8	256	254
25	255.255.255.128	9	512	126
26	255.255.255.192	10	1024	62
27	255.255.255.224	11	2048	30
28	255.255.255.240	12	4096	14
29	255.255.255.248	13	8192	6
30	255.255.255.252	14	16384	2

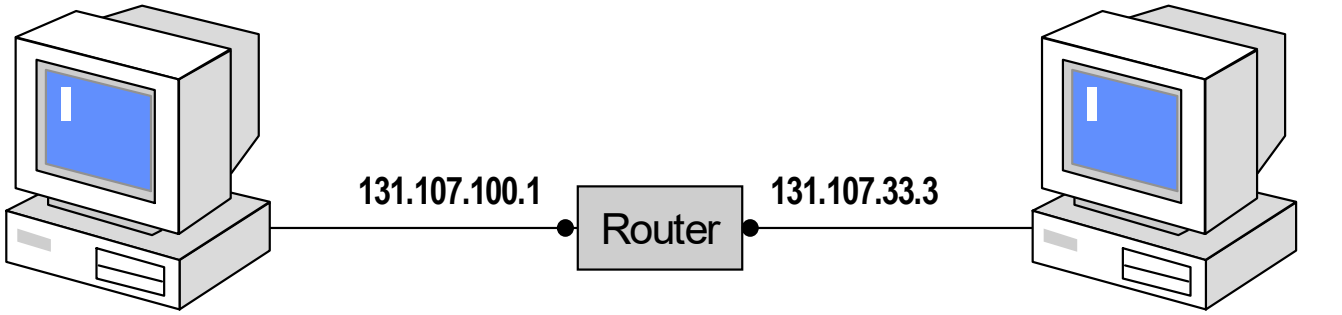
Exercise : Subnetting

- คอมพิวเตอร์ต่อไปนี้ เครื่องใด Config ผิด



Exercise : Subnetting

- คอมพิวเตอร์ 2 เครื่องนี้ติดต่อกันได้หรือไม่



IP Address: **131.107.100.27**
Subnet Mask: **255.255.0.0**
Default Gateway: **131.107.100.1**

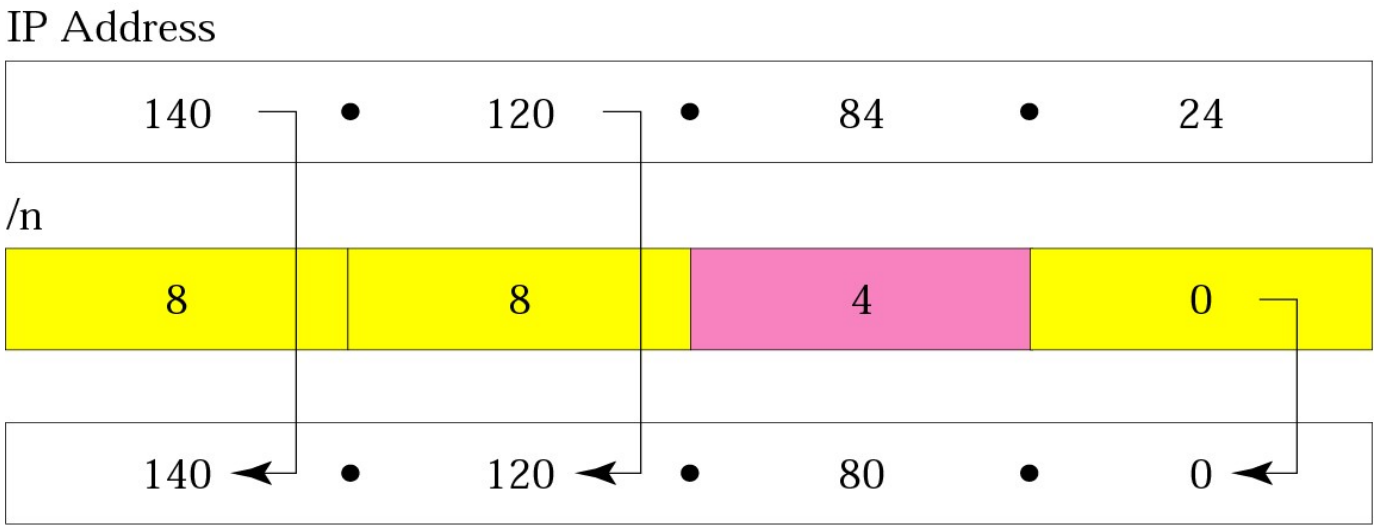
IP Address: **131.107.33.7**
Subnet Mask: **255.255.0.0**
Default Gateway: **131.107.33.3**

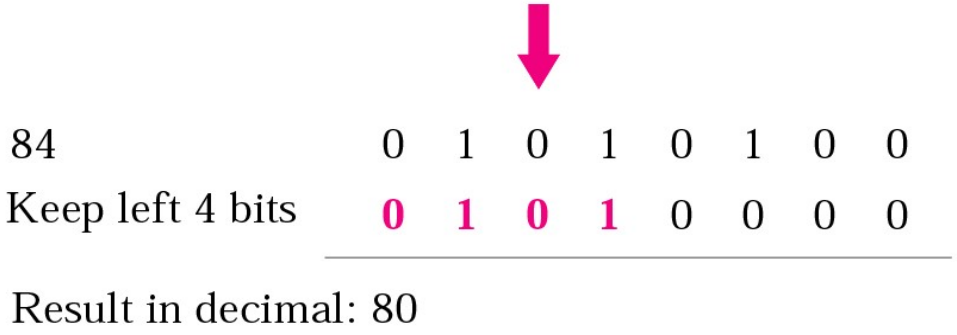
Exercise : Subnetting

- ให้หา Address แรกของบล็อก ถ้า IP Address คือ
- 140.120.84.24/20
- 167.199.170.82/27

Exercise : Subnetting

- ให้หา Address แรกของบล็อก ถ้า IP Address คือ
- 140.120.84.24/20





Exercise : Subnetting

- ให้หา Address แรกของบล็อก ถ้า IP Address คือ
- 140.120.84.24/20

	128	64	32	16	8	4	2	1
Write 84 as sum of:	0	64	0	16	0	4	0	0
Select only leftmost 4:	0	64	0	16				
Add to find the result:	80							

Exercise : Subnetting

- ให้หาจำนวน Host ที่มีได้มากที่สุด ใน Subnet ถ้า IP Address คือ 140.120.84.24/20
- ให้หาจำนวน Host ที่มีได้มากที่สุด ใน Subnet ถ้า IP Address คือ 190.87.140.202/29
- หา Address สุดท้ายของ Subnet 140.120.84.24/20

Exercise : Subnetting

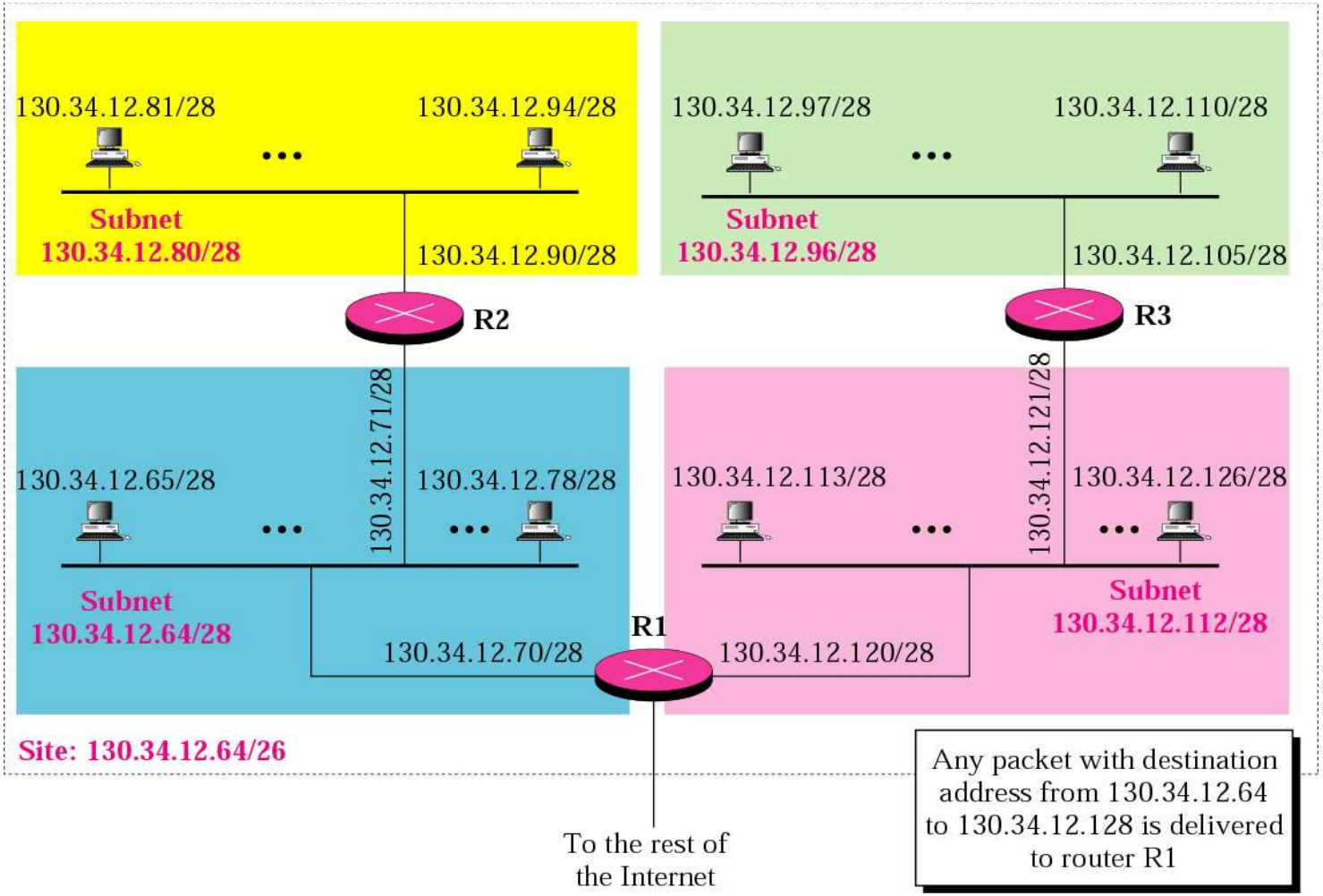
- ในองค์กรแห่งหนึ่งได้รับ Internet Address 130.34.12.64/26 และต้องการแบ่งออกเป็น 4 Subnet ให้หา Subnet Mask ขององค์กรนี้และแต่ละช่วงของ Address

Exercise : Subnetting

- 130.34.12.64/26 และต้องการแบ่งออกเป็น 4 Subnet
- ก่อนอื่นต้องตรวจสอบว่าเป็น Address แรกหรือไม่
  - IP : 10000010.00100010.00001100.01 000000
  - MASK : 11111111.11111111.11111111.11 000000
- /28 = Subnet ละ 16
  - Subnet 1 = 64 = 140.34.12.64/28 (65-78)
  - Subnet 2 = 64 -> 64+16 = 80 = 130.34.12.80/28 (81-94)
  - Subnet 3 = 80 -> 80+16 = 96 = 130.34.12.96/28 (97-110)
  - Subnet 3 = 96 -> 96+16 = 112 = 130.34.12.112/28 (113-126)

Exercise : Subnetting

- 130.34.12.64/26 และต้องการแบ่งออกเป็น 4 Subnet



Exercise : Subnetting

- ในองค์กรแห่งหนึ่งได้รับ Internet Address 14.24.74.0/24 และต้องการแบ่งออกเป็น Subnet ดังนี้
  - 2 Subnet มี 62 Addresses
  - 2 Subnet มี 30 Addresses
  - 3 Subnet มี 14 Addresses
  - 4 Subnet มี 2 Addresses
- ให้เขียนระบบเครือข่ายขององค์กรนี้

Exercise : Subnetting

- ก่อนอื่นต้องตรวจสอบว่าเป็น Address แรกหรือไม่
  - IP : 00001110.00011000.01001010.00000000
  - MASK : 11111111.11111111.11111110.00000000
- ทั้งหมดมี Host = 254 Address
  - 2 Subnet มี 62 Address = 62 x 2
  - 2 Subnet มี 30 Address = 30 x 2
  - 3 Subnet มี 14 Address = 14 x 2
  - 4 Subnet มี 2 Address = 2 x 4
- 64 + 64 + (32+32) + (16+16+(4+4+4+4))

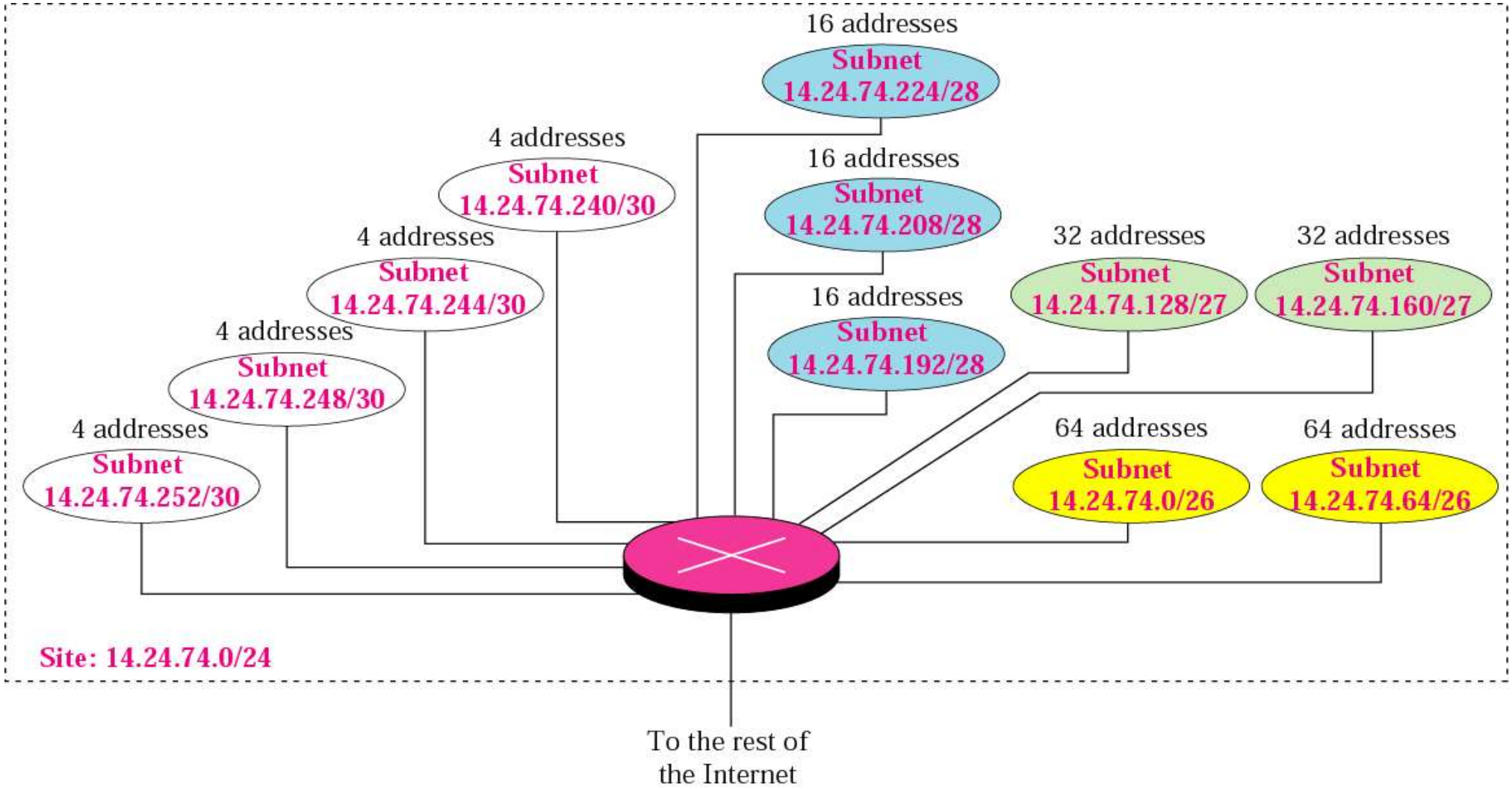
Exercise : Subnetting

- เอา /24 แบ่ง 4 = /26 (0, 64, 128, 192)
- 2 Subnet มี 62 Address ต้องใช้ /26 (64)
  - 14.24.74.1 –> 14.24.74.62 (14.24.4.0/26)
  - 14.24.74.65 -> 14.24.74.126 (14.24.74.64/26)
- จากนั้นเอา ช่วง 128 มาแบ่งเป็น 2 Subnet = /27 (128, 160)
- 2 Subnet มี 30 Address ต้องใช้ /27 (32)
  - 14.24.74.129 -> 14.24.74.158 (14.24.74.128/27)
  - 14.24.74.161 -> 14.24.74.190 (14.24.74.160/27)

Exercise : Subnetting

- จากนั้นเอาช่วง 192 (/26) มาแบ่งเป็น 4 Subnet (/28) แต่ใช้ 3
- 3 Subnet มี 14 Address ต้องใช้ /28 (16)
  - 14.24.74.193 -> 14.24.74.206 (14.24.74.192/28)
  - 14.24.74.209 -> 14.24.74.222 (14.24.74.208/28)
  - 14.24.74.225 -> 14.24.74.238 (14.24.74.224/28)
- และสุดท้ายเอา 14.24.74.240/28 มา แบ่งเป็น 4 Subnet (/30)
  - 14.24.74.241 -> 14.24.74.242 (14.24.74.240/30)
  - 14.24.74.245 -> 14.24.74.246 (14.24.74.244/30)
  - 14.24.74.249 -> 14.24.74.250 (14.24.74.248/30)
  - 14.24.74.253 -> 14.24.74.254 (14.24.74.252/30)

Exercise : Subnetting

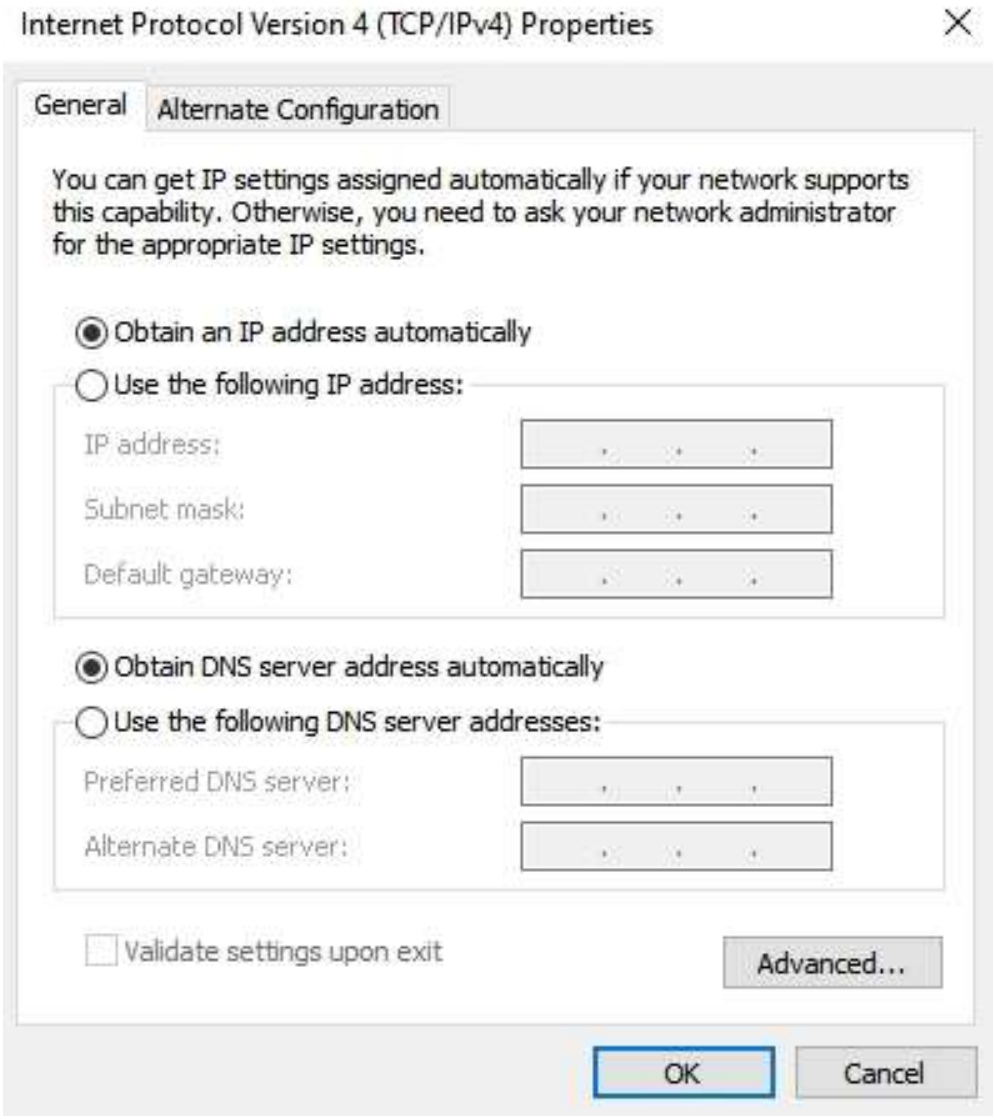


IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

IP addresses: how to get one?



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 want to join network (more shortly)

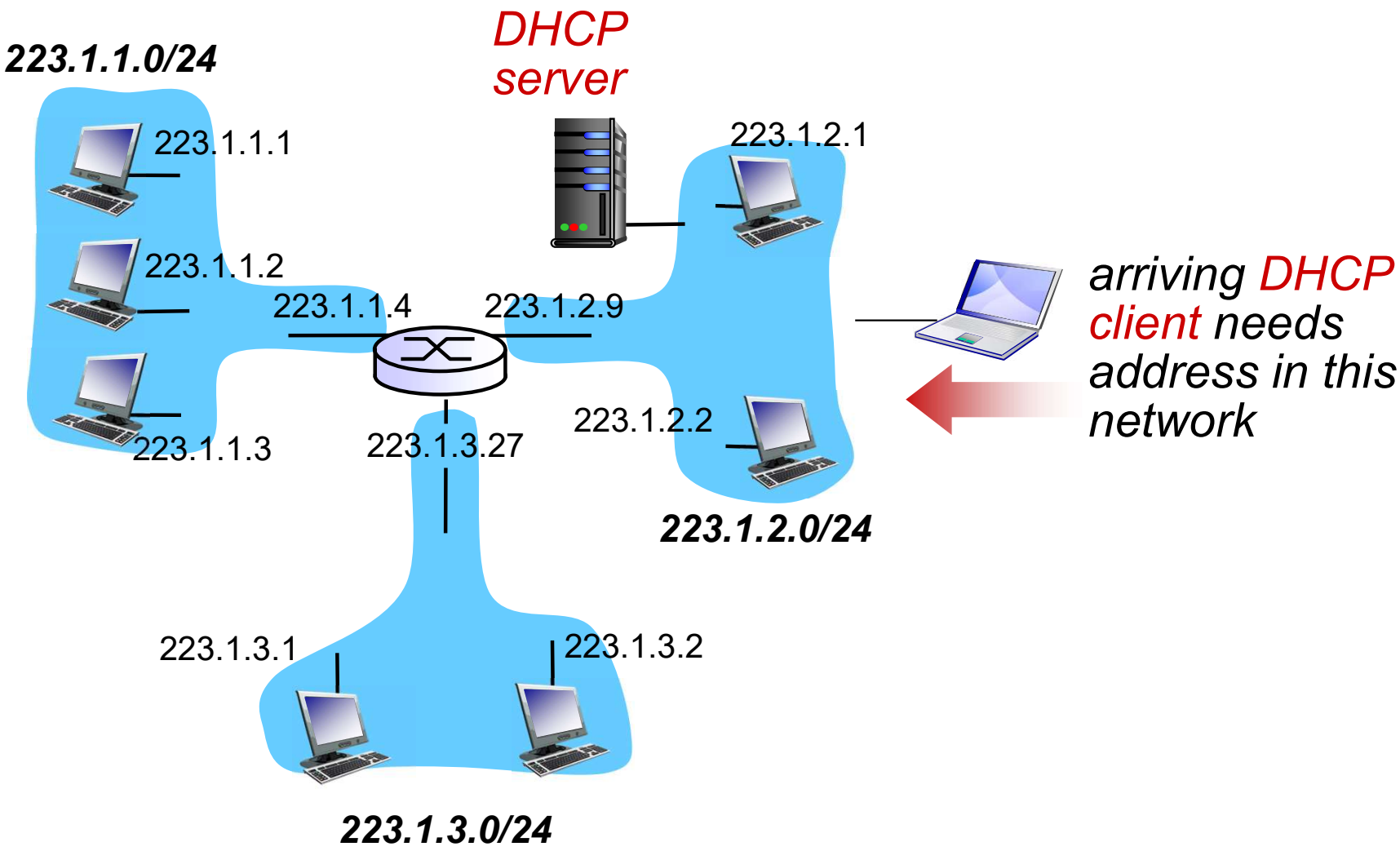
DHCP overview:

- 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

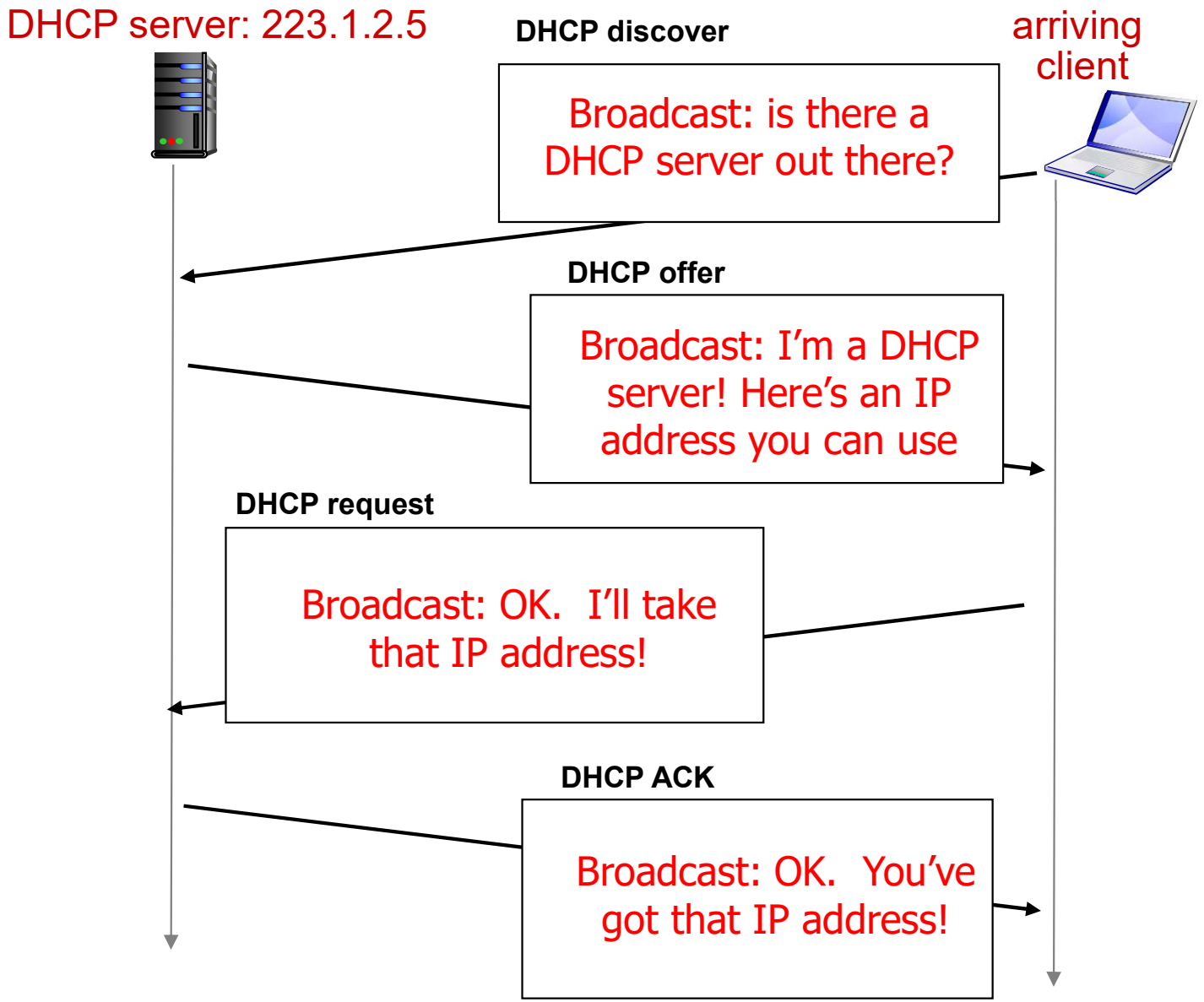
Exercise : DHCP

- จากคอมพิวเตอร์ให้ใช้คำสั่ง
  - ipconfig แล้วจด IPv4
- จากนั้นใช้คำสั่ง
  - ipconfig /release แล้วตรวจสอบ IPv4 อีกทีว่าเป็นเลขอะไร (เป็นการคืน IP Address ไป)
  - ใช้ Internet ได้หรือไม่
- จากนั้นใช้คำสั่ง
  - ipconfig /renew แล้วตรวจสอบ IPv4 (เป็นการขอ IP Address ใหม่)
  - ใช้ Internet ได้หรือไม่ แล้วดู Life Time
- ทำทุกคน แล้ว post ใน chat

DHCP client-server scenario



DHCP client-server scenario



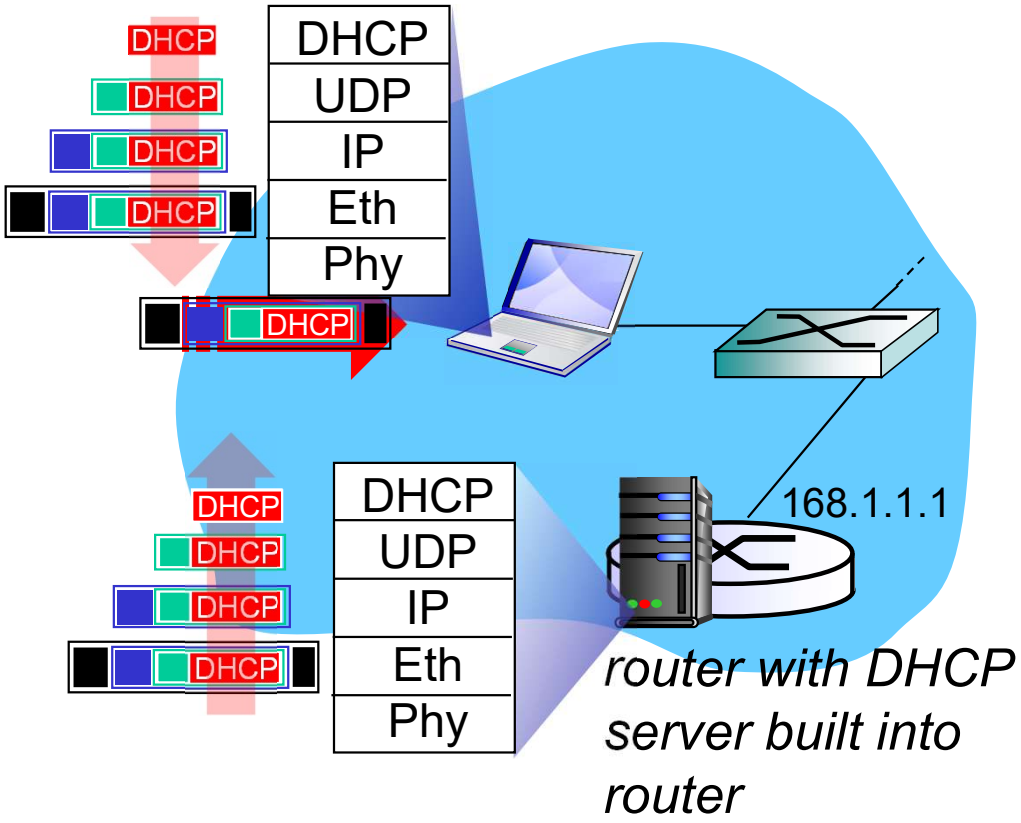
IP Address config

DHCP: more than IP addresses

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

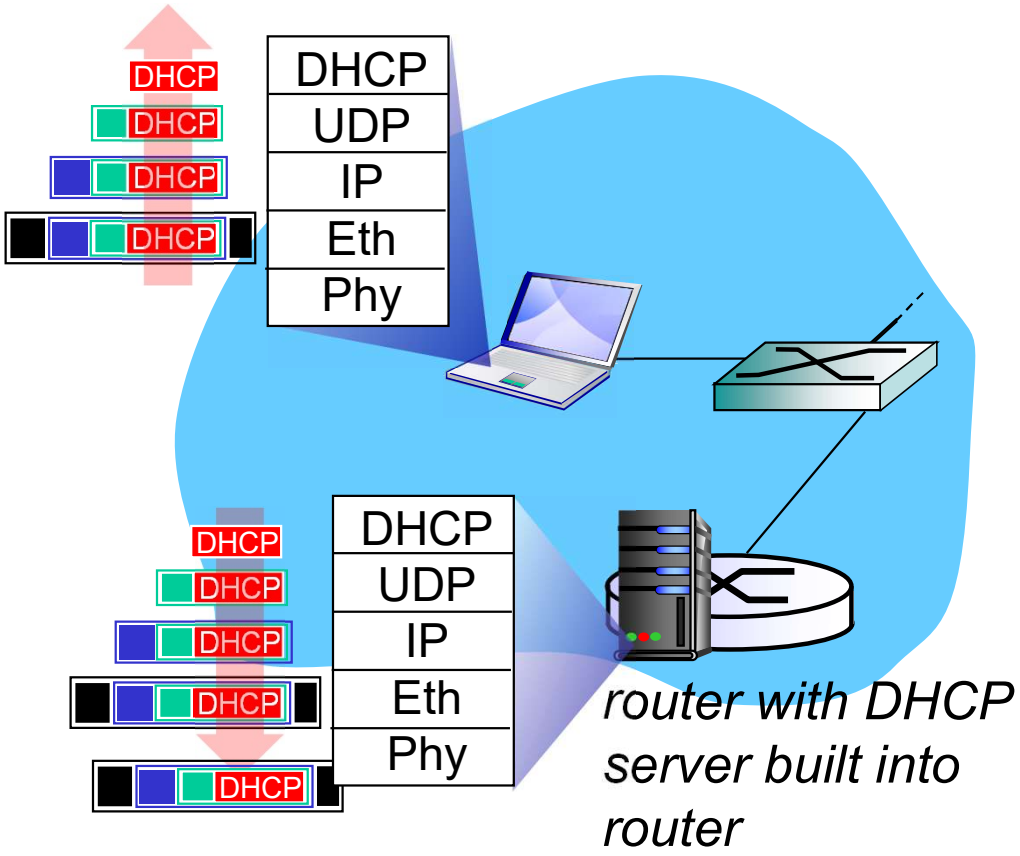
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame 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

Exercise : DHCP

- เปิด wireshark ให้รับเฉพาะ dhcp
- จากคอมพิวเตอร์ให้ใช้คำสั่ง
  - ipconfig
  - จากนั้นใช้คำสั่ง ipconfig /release
  - จากนั้นใช้คำสั่ง ipconfig /renew
- ตรวจสอบ Packet เทียบกับเนื้อหาที่เรียน

DHCP: Wireshark output (home LAN)

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

Exercise : DHCP

- ให้เขียนข้อมูลใน DHCP packet โดย IP ที่ offer คือ 192.168.1.72, server=192.168.1.254, router,dns=192.168.1.1/24, lease=60m

DHCP DISCOVER
Src IP :
Dest IP :
Client Address :
Your Client Address :
Server Address :
Option :

DHCP OFFER
Src IP :
Dest IP :
Client Address :
Your Client Address :
Server Address :
Option :

DHCP REQUEST
Src IP :
Dest IP :
Client Address :
Your Client Address :
Server Address :
Option :

DHCP ACK
Src IP :
Dest IP :
Client Address :
Your Client Address :
Server Address :
Option :

IP addresses: how to get one?

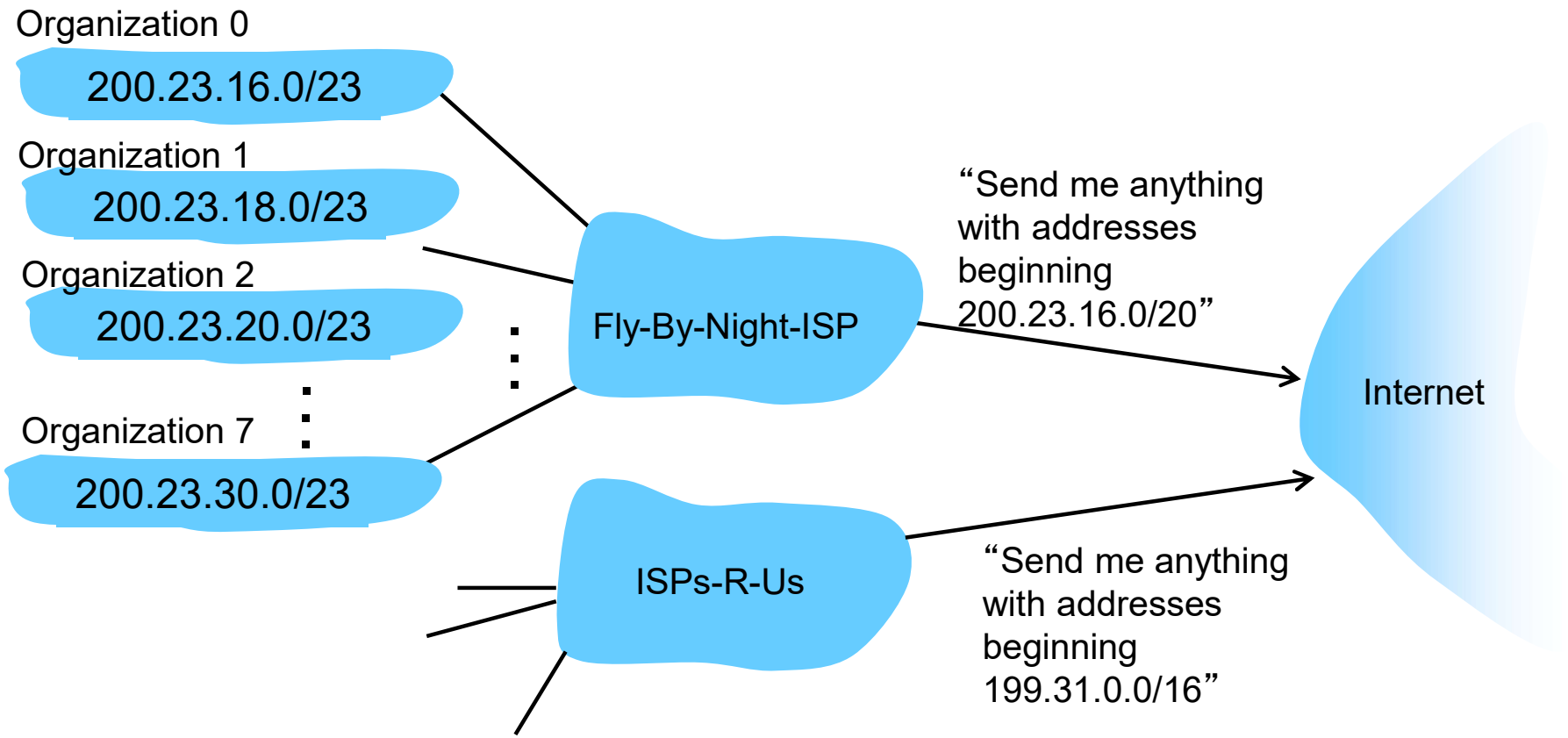
Q: how does *network* get subnet part of IP addr?

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

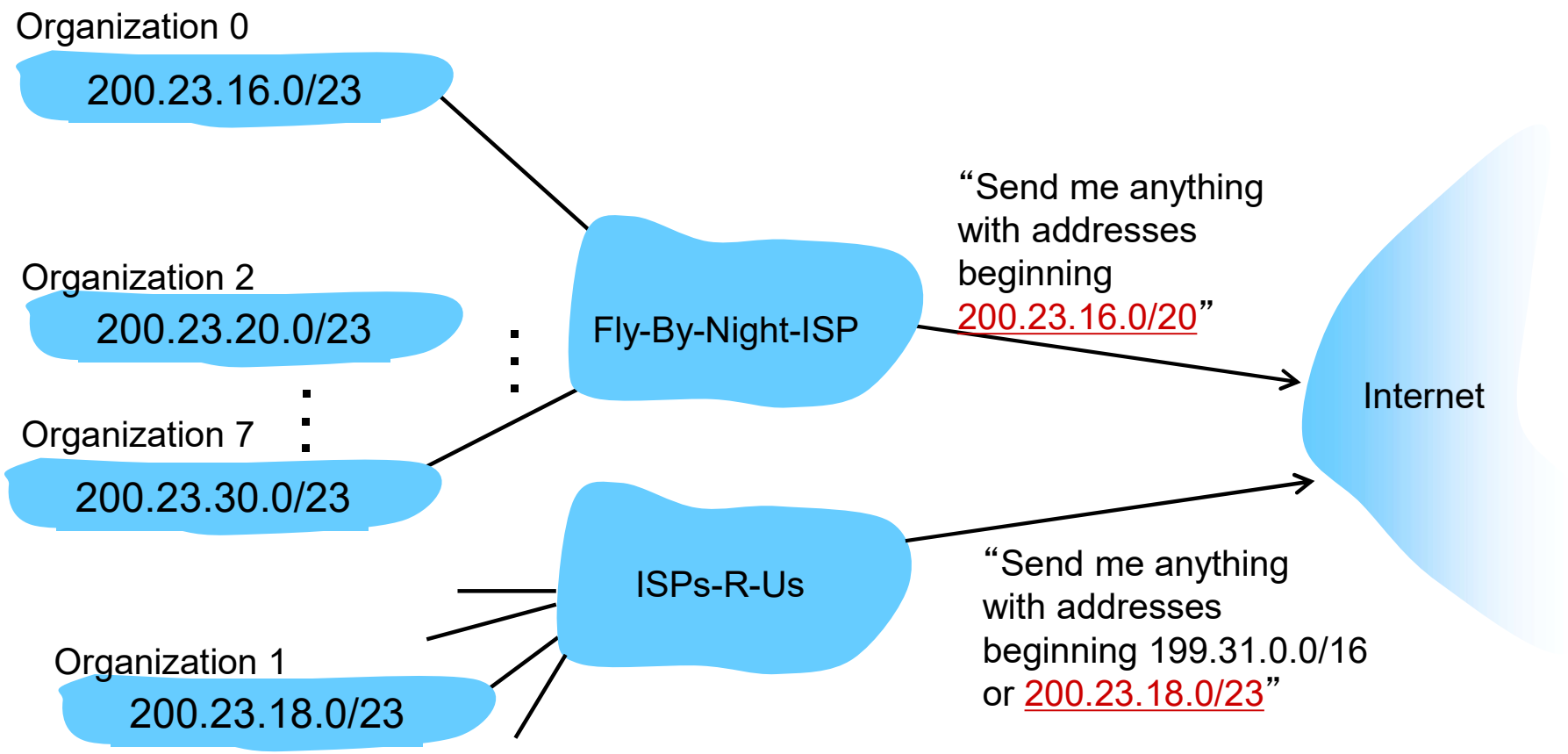
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



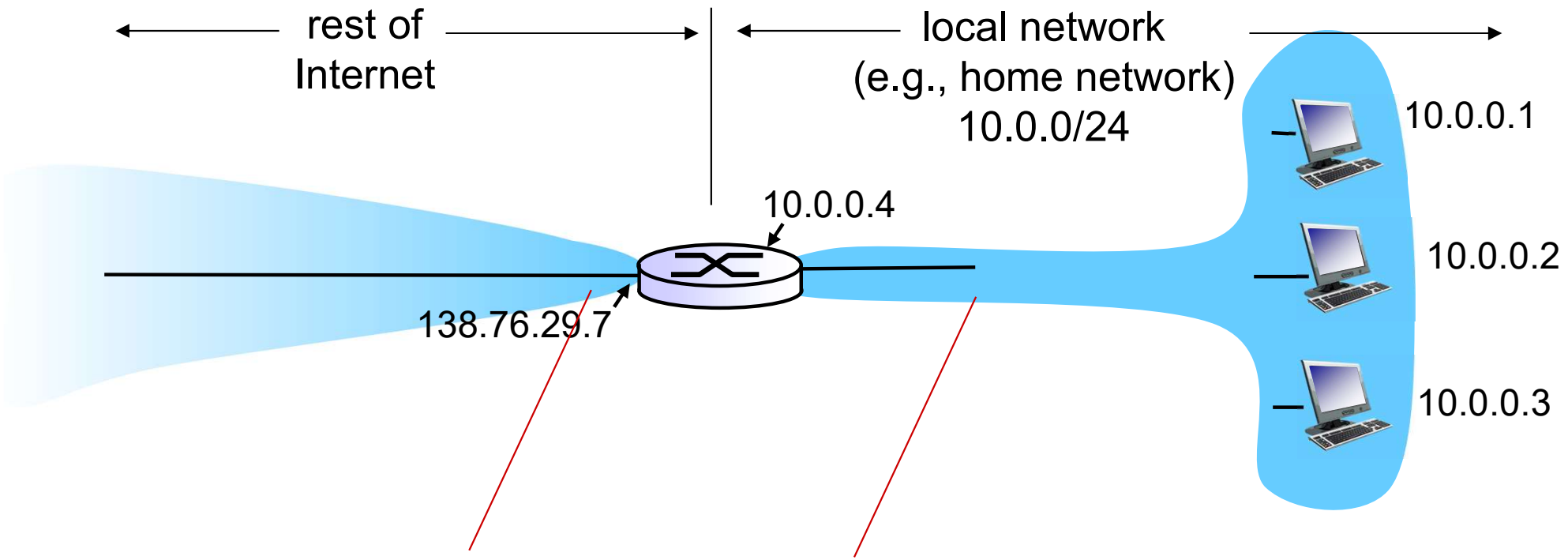
IP addressing: the last word...

Q: how does an ISP get block of addresses?

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

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

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

- ตัวอย่างการใช้ NAT เช่นการเข้าถึง Internet จาก Private IP
- Private IP คือ IP Address ที่ใช้งานได้เฉพาะภายในองค์กร
- ทุกองค์กรสามารถนำ Private IP ไปใช้ได้โดยไม่ต้องขอใช้งาน
- มาตรฐานสงวนไม่ให้ใช้ Private IP บน Public network จึงไม่สามารถ Route ได้ใน Internet จึงต้องใช้ NAT

Private IP address space	
From	**To**
10.0.0.0	10.255.255.255
172.16.0.0	172.31.255.255
192.168.0.0	192.168.255.255

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:

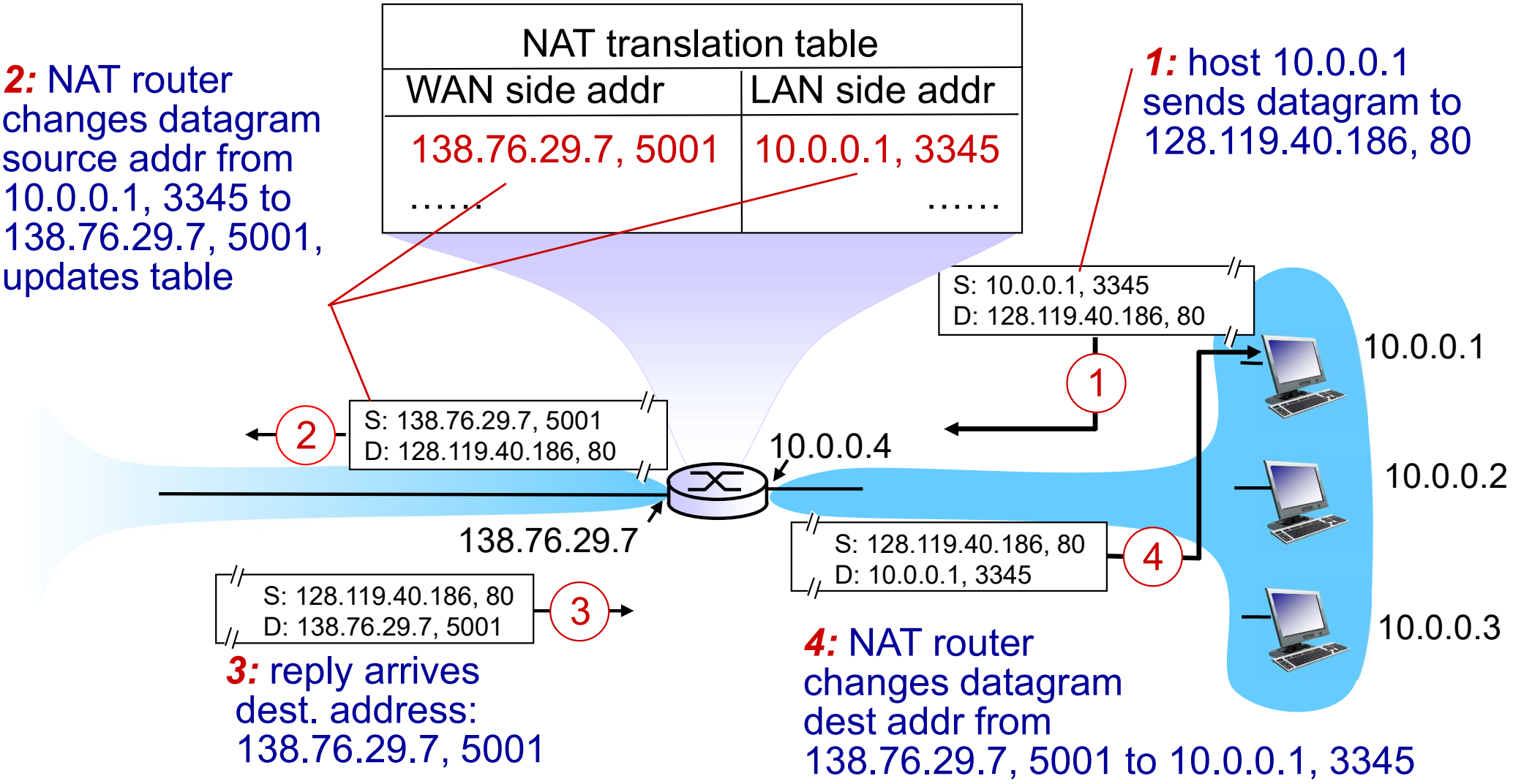
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *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 dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

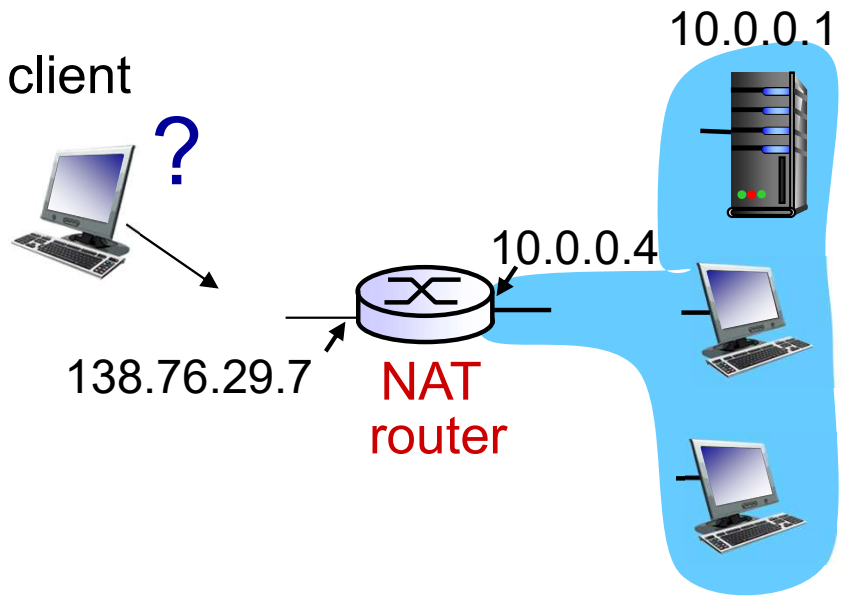


Exercise : NAT

- เครือข่ายหนึ่งคล้ายกับ Slide ก่อนหน้า สมมติว่า ISP ได้ให้ IP Address 24.34.112.235 และใช้ Private IP เป็น 192.168.1.0/24
  - ให้กำหนด IP Address ให้กับทุก Interface ใน Private Network
  - สมมติว่าแต่ละเครื่องมี TCP connection ขาออกจำนวน 2 ครั้ง ไปยัง port 80 ของ IP 128.119.40.86 จงแสดง NAT Translation Table ที่เป็นไปได้

NAT traversal problem

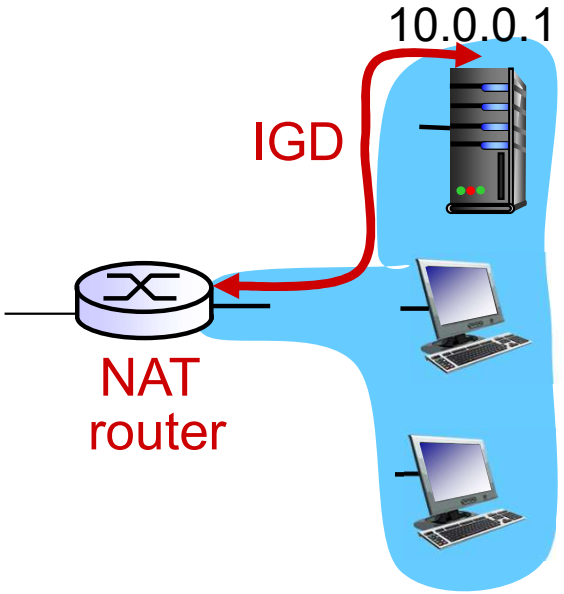
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution 1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

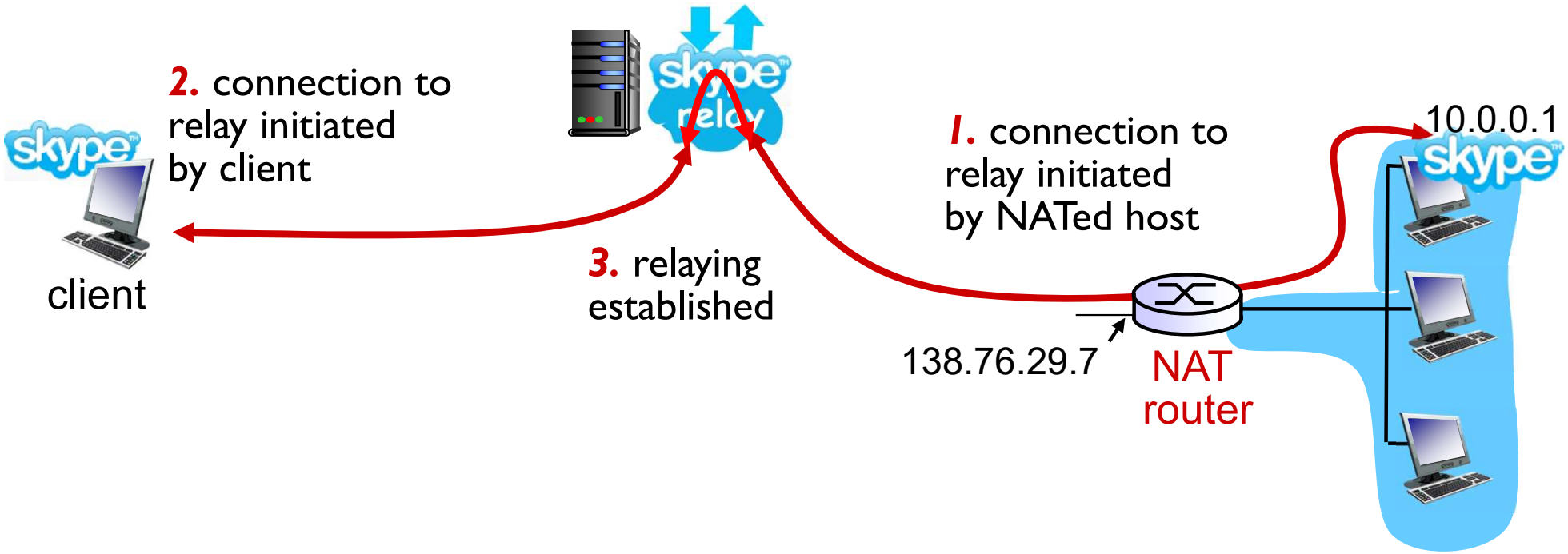
- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



NAT traversal problem

- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

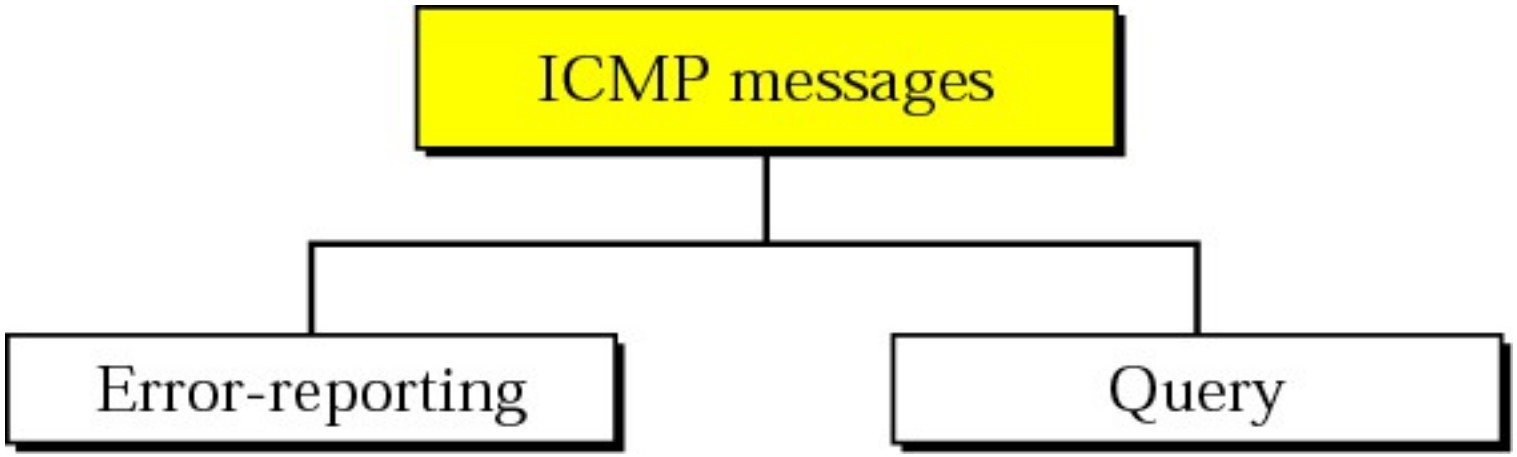
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

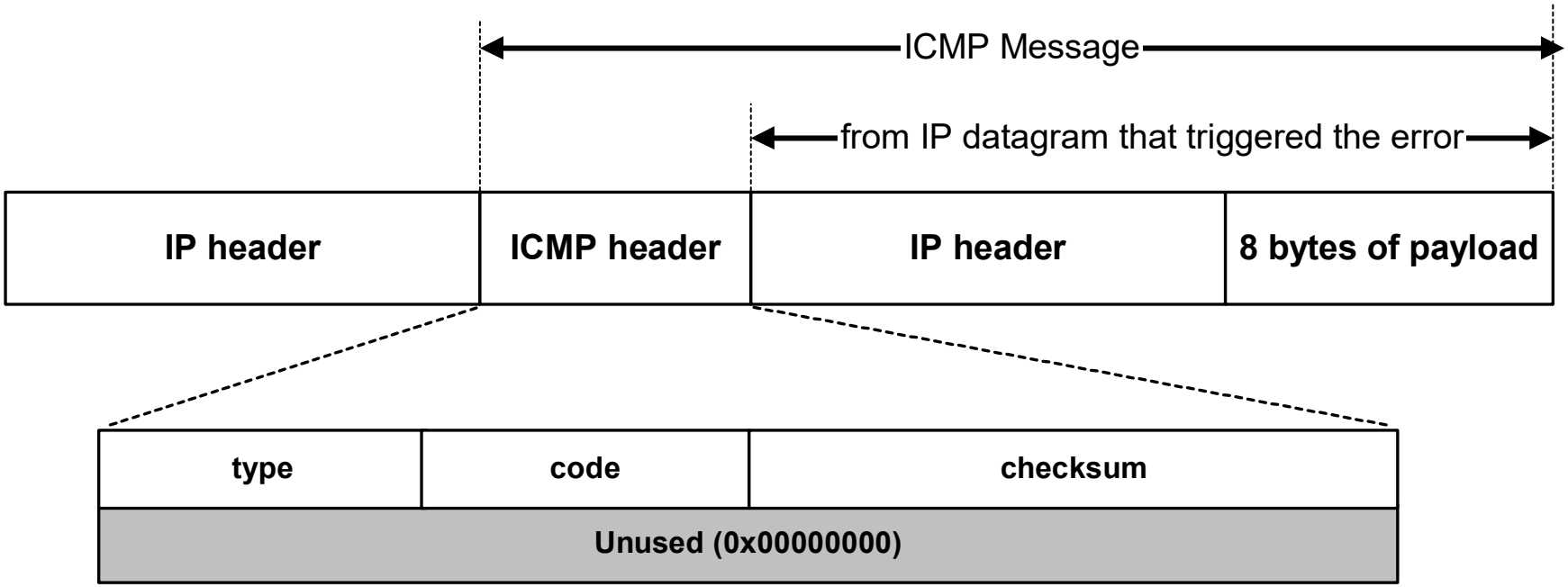
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

ICMP: internet control message protocol

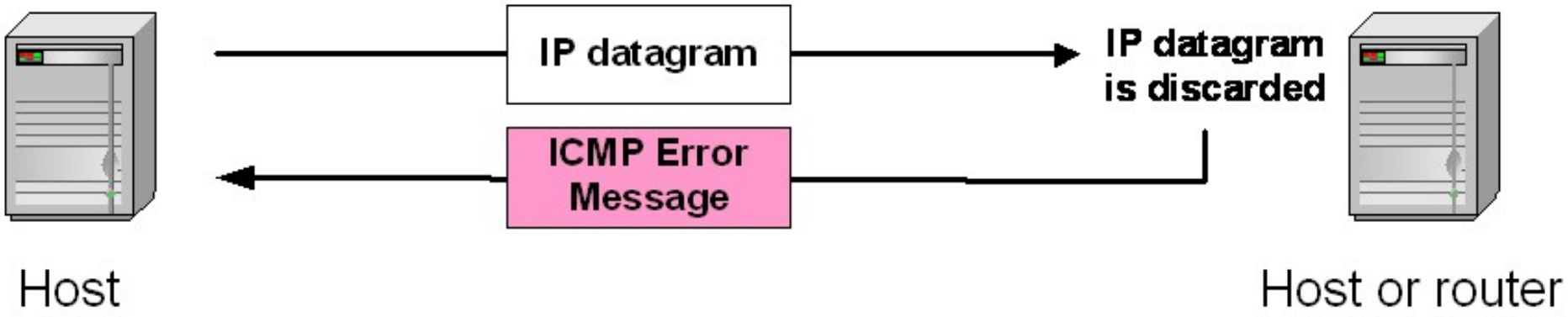


ICMP: Error Reporting



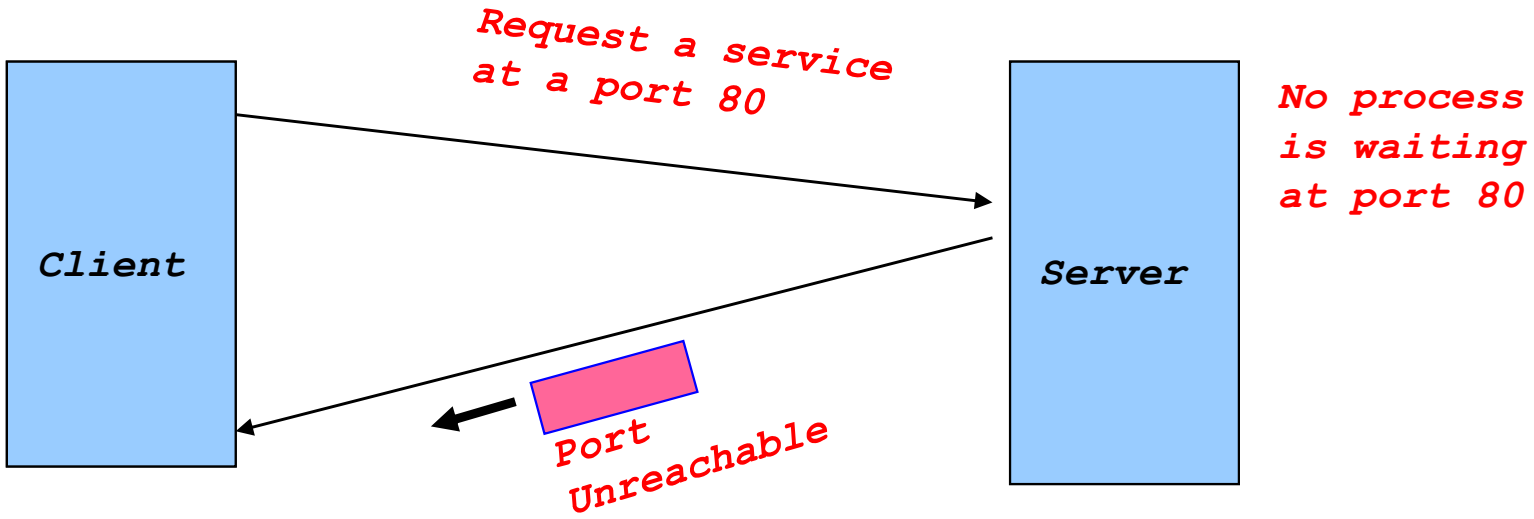
- ICMP error messages include the complete IP header and the first 8 bytes of the payload (typically: UDP, TCP)

ICMP: Error Reporting



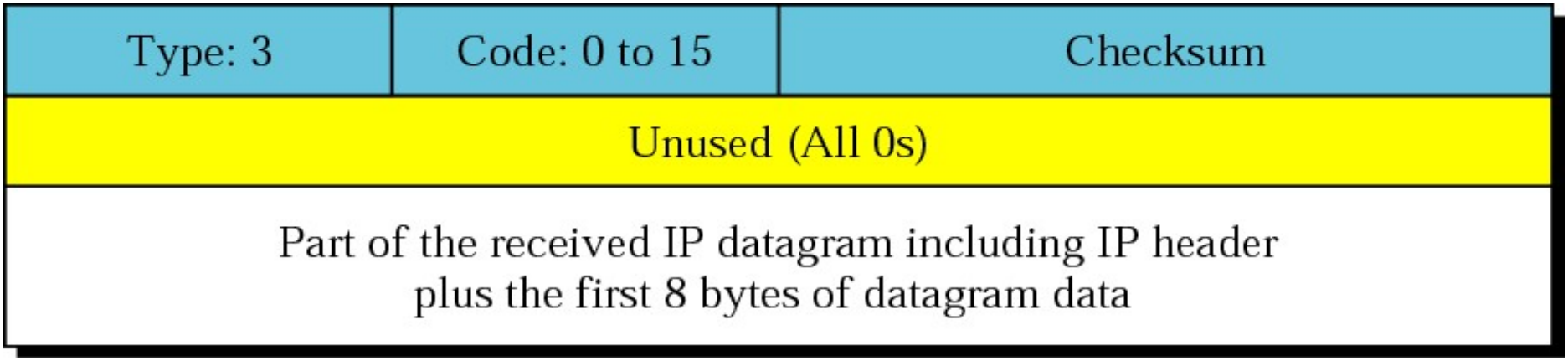
- ICMP error messages report error conditions
- Typically sent when a datagram is discarded
- Error message is often passed from ICMP to the application program

ICMP: Error Reporting



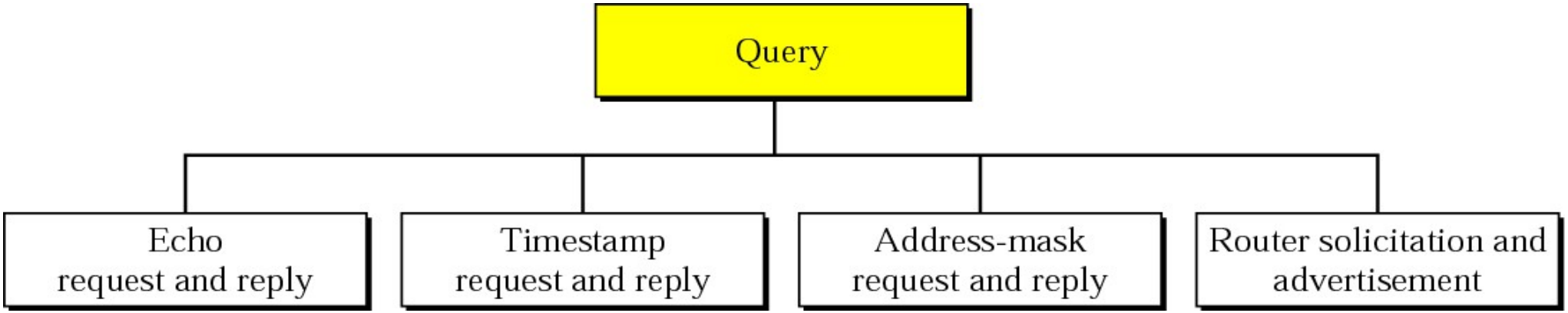
RFC 792: If, in the destination host, the IP module cannot deliver the datagram because the indicated protocol module or process port is not active, the destination host may send a destination unreachable message to the source host.

ICMP: Error Reporting

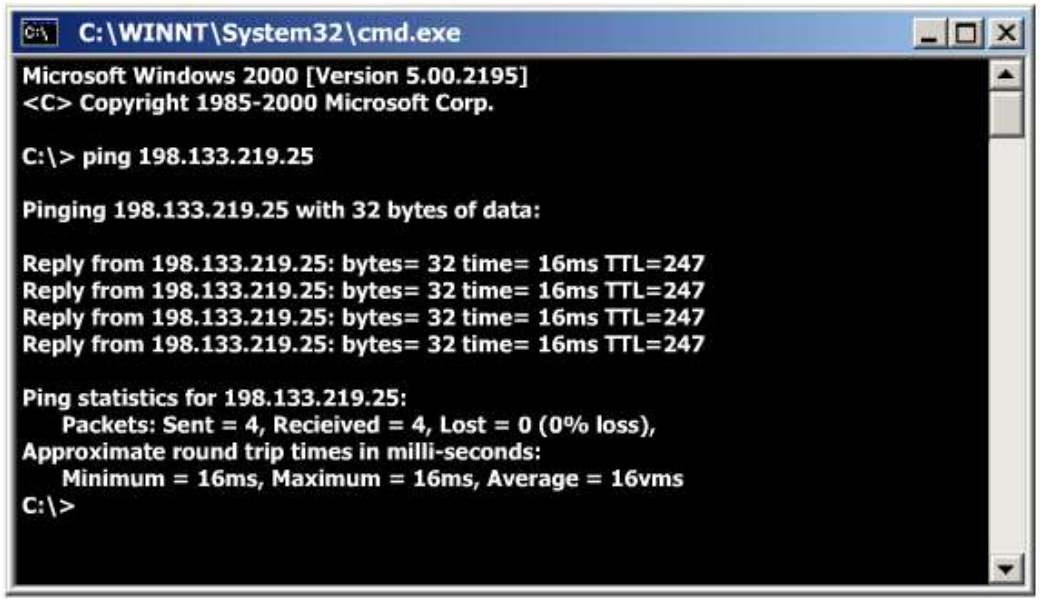


- Destination Unreachable, router cannot route a datagram.
  - 0 = network unreachable, 1 = host unreachable : hardware failure.
  - 2 = protocol unreachable : no select protocol install in destination.
  - 3 = port unreachable : program not running.
  - 4 = fragmentation is required but DF has been set.
  - 5 = source route cannot be accomplished.
  - 6 = destination network in unknown : router no information of network.
  - 7 = destination host is unknown

ICMP: Query


Query
Echo request and reply
Timestamp request and reply
Address-mask request and reply
Router solicitation and advertisement

ICMP: Query



8: Echo request
0: Echo reply

Type: 8 or 0	Code: 0	Checksum
Identifier		Sequence number
Optional data Sent by the request message; repeated by the reply message		

Echo = Type 8

Echo Reply = Type 0

Ethernet Header (Layer 2)			**IP Header (Layer 3)**	**ICMP Message (Layer 3)**						**Ether. Tr.**
Ethernet Destination Address (MAC)	Ethernet Source Address (MAC)	Frame Type	Source IP Add. Dest. IP Add. Protocol field	Type 0 or 8	Code 0	Check-sum	ID	Seq. Num.	Data	FCS

- IP Protocol Field = 1
- The echo request message is typically initiated using the ping command .

ICMP: Query

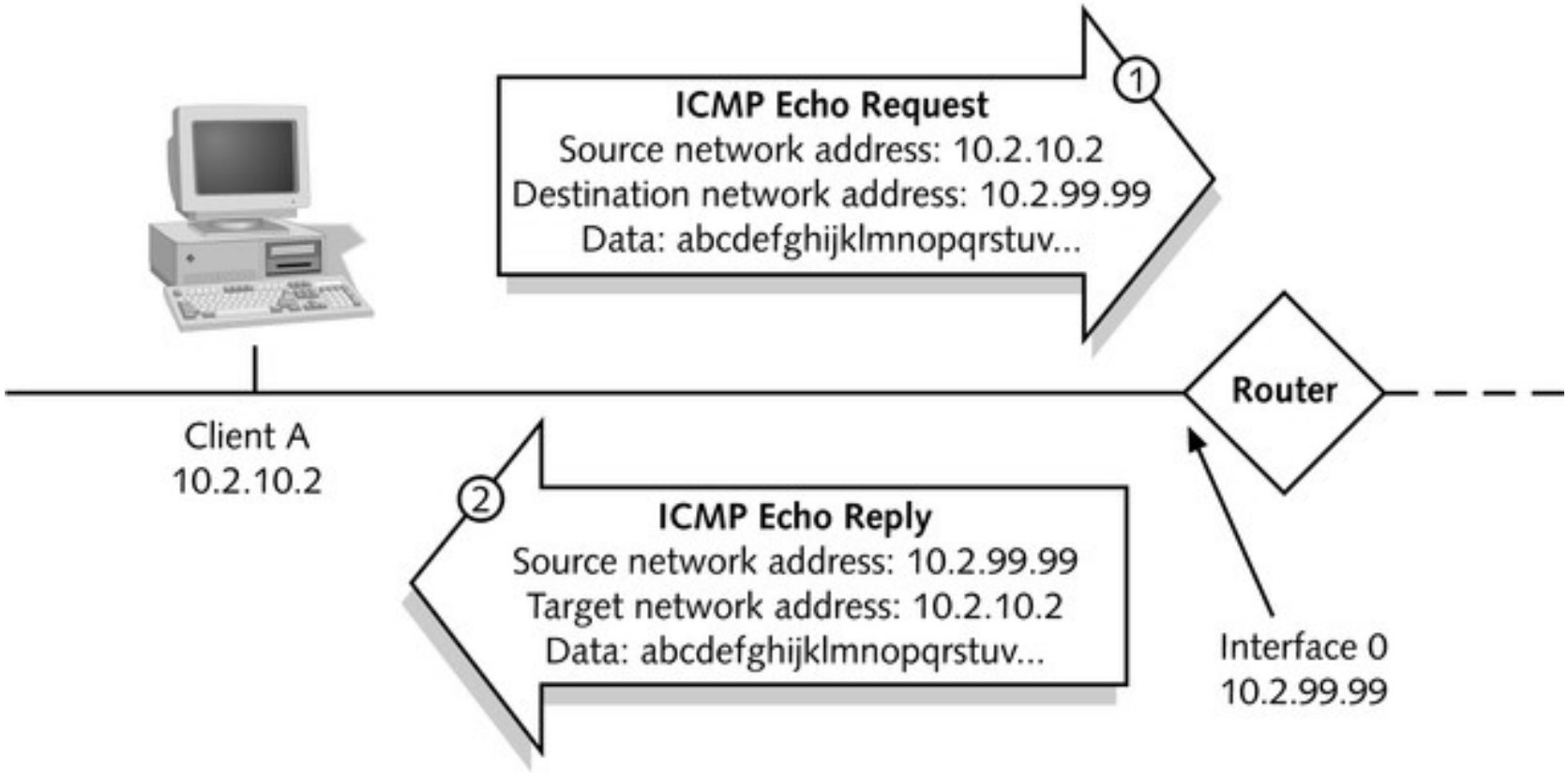


Figure 4-1 PING utility uses ICMP echo requests and replies

Exercise : Ping

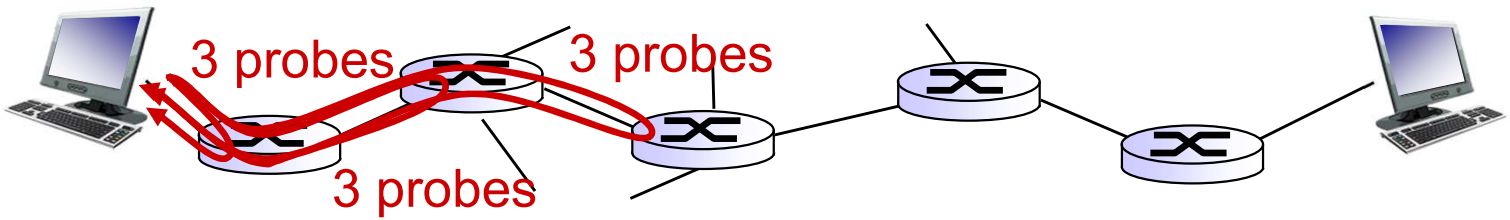
- เปิด wireshark ดักจับข้อมูล
- ใช้ command prompt แล้วใช้คำสั่ง ping ไปที่ router
- หยุด wireshark แล้ว filter ดูเฉพาะข้อมูล icmp

Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

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

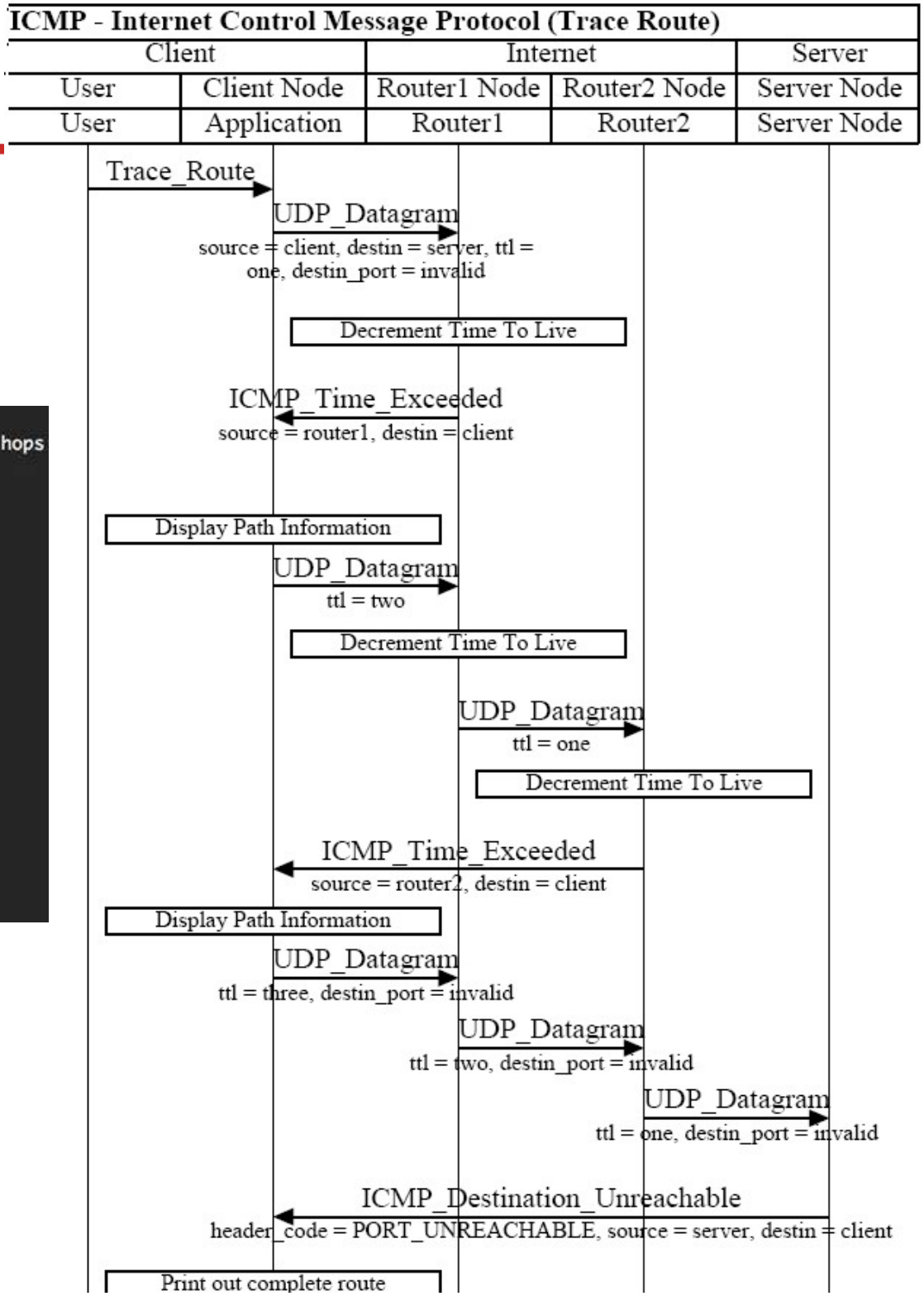


Traceroute

```
SG350X#traceroute ip software.cisco.com ttl 20
Tracing the route to software.cisco.com (184.26.111.212) from , 20 hops
max, 18 byte packets
Type Esc to abort.
 1  192.168.100.1 (192.168.100.1)  <10 ms  <10 ms  <10 ms
 2  124.6.177.113 (124.6.177.113)  <20 ms  <10 ms  <20 ms
 3  124.6.149.117 (124.6.149.117)  <20 ms  <30 ms  <30 ms
 4  120.28.0.61 (120.28.0.61)  <20 ms  <20 ms  <30 ms
 5  120.28.10.101 (120.28.10.101)  <40 ms  <30 ms  <30 ms
 6  120.28.9.158 (120.28.9.158)  <40 ms  <40 ms  <40 ms
 7   *  *  *
 8   *  *  *
 9  63.218.2.189 (63.218.2.189)  <50 ms  <50 ms  <50 ms
10  63.223.17.162 (63.223.17.162)  <60 ms  <50 ms  <50 ms
11  63.223.17.162 (63.223.17.162)  <50 ms  <50 ms  <50 ms
12  213.254.227.77 (213.254.227.77)  <50 ms  <60 ms  <50 ms
13   *  *  *
14  184.26.111.212 (184.26.111.212)  <190 ms  <200 ms  <200 ms

Trace complete.

SG350X#
```



Exercise : Traceroute

- ใช้ command prompt แล้วใช้คำสั่ง tracert 8.8.8.8

IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

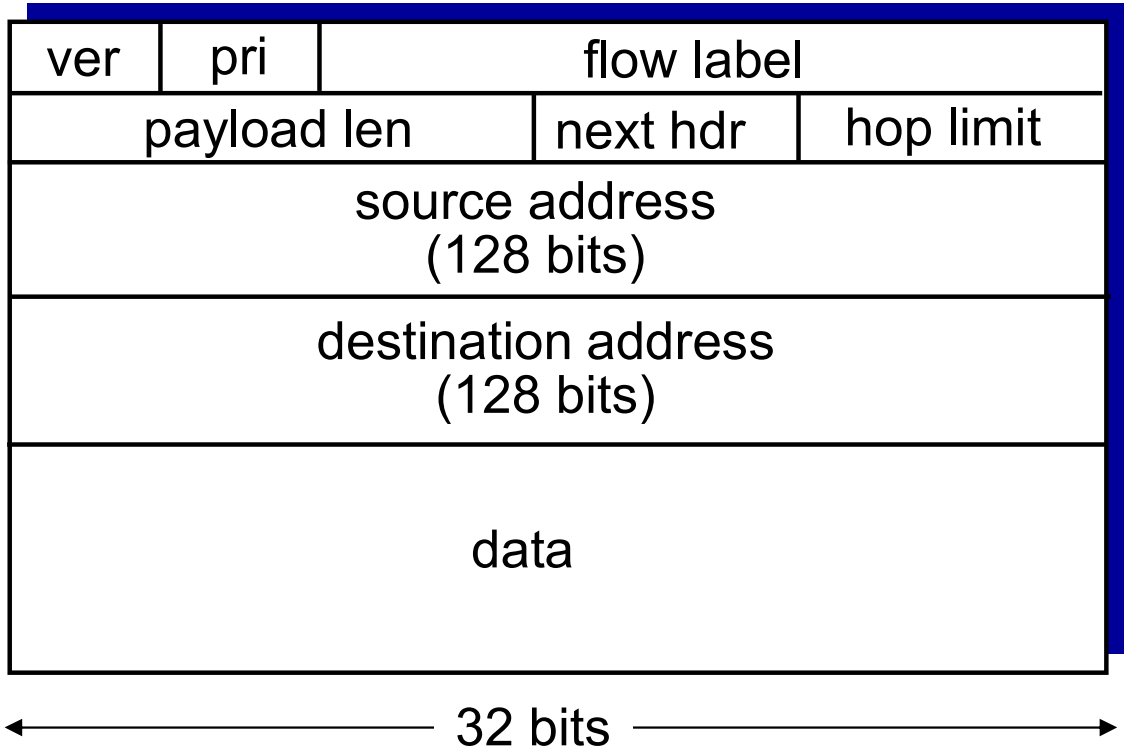
- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

priority: identify priority among datagrams in flow

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

next header: identify upper layer protocol for data

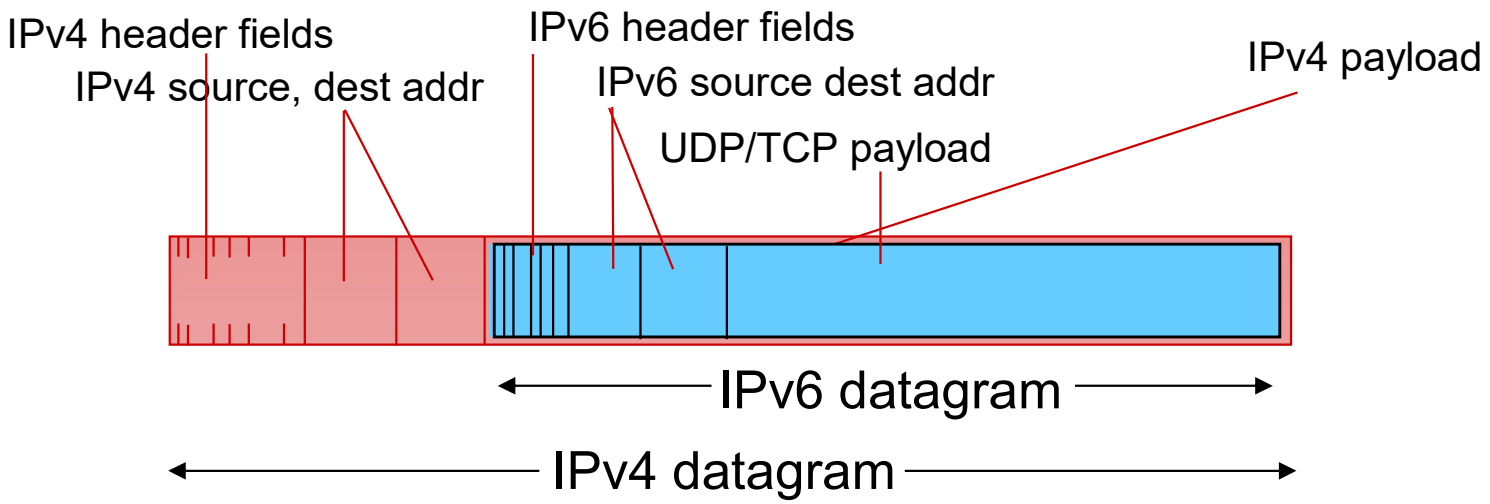


Other changes from IPv4

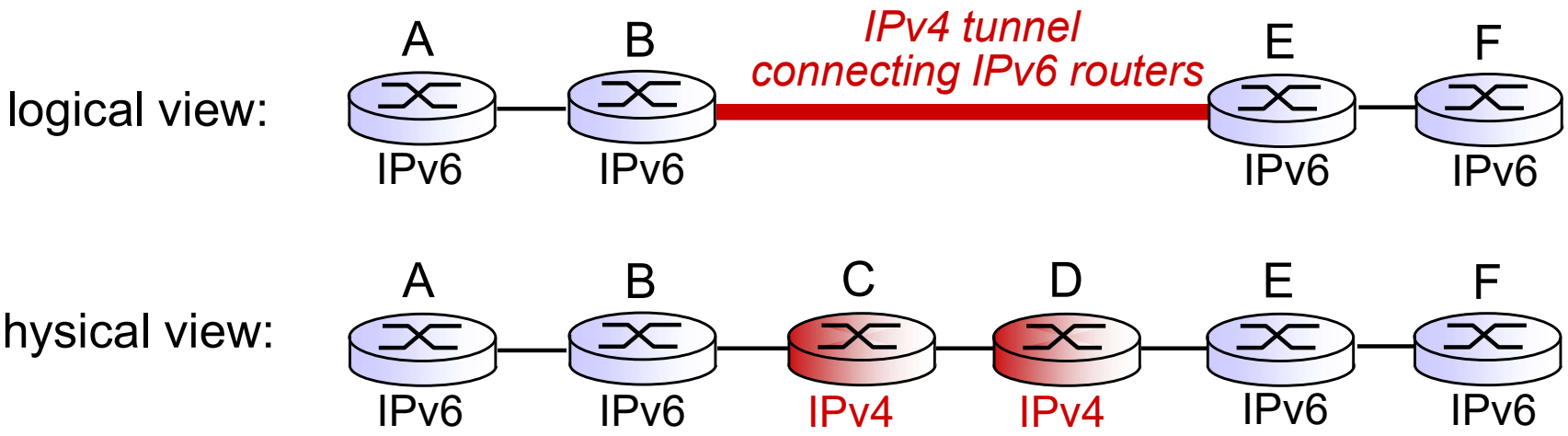
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

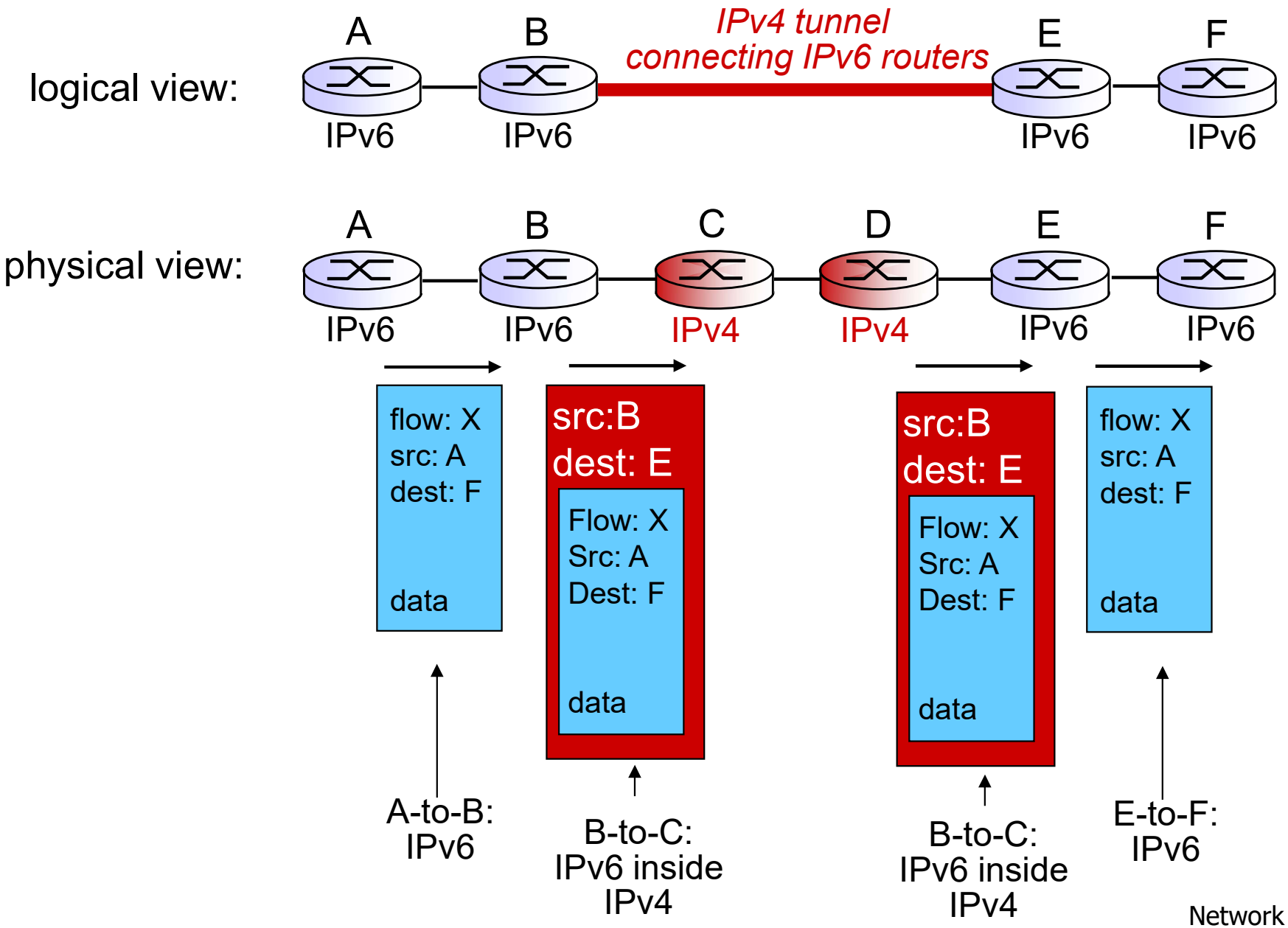
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
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
E-to-F:
IPv6

IPv6: adoption

- US National Institutes of Standards estimate [2013]:
  - ~3% of industry IP routers
  - ~11% of US gov't routers

- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, …
  - *Why?*

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

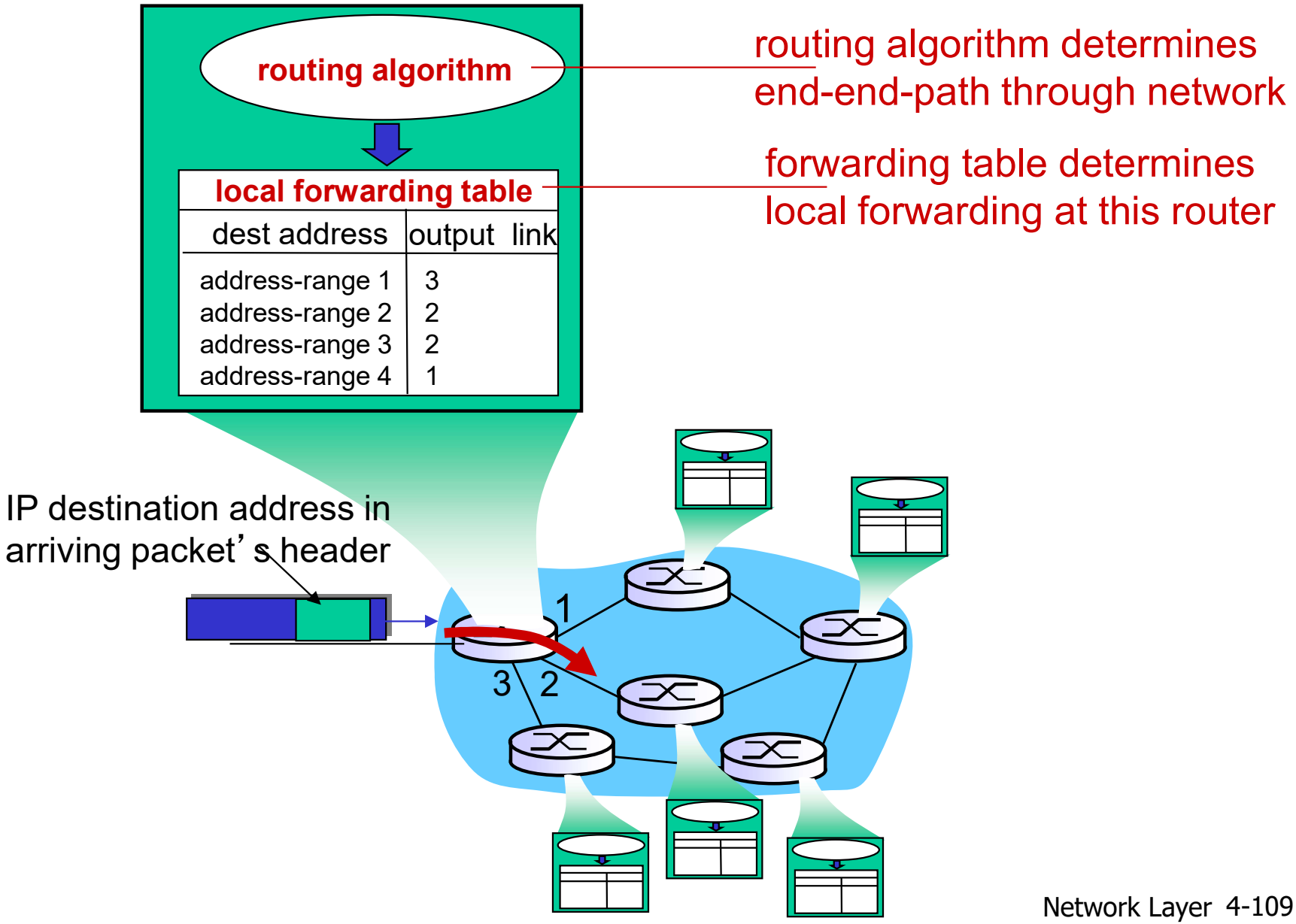
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

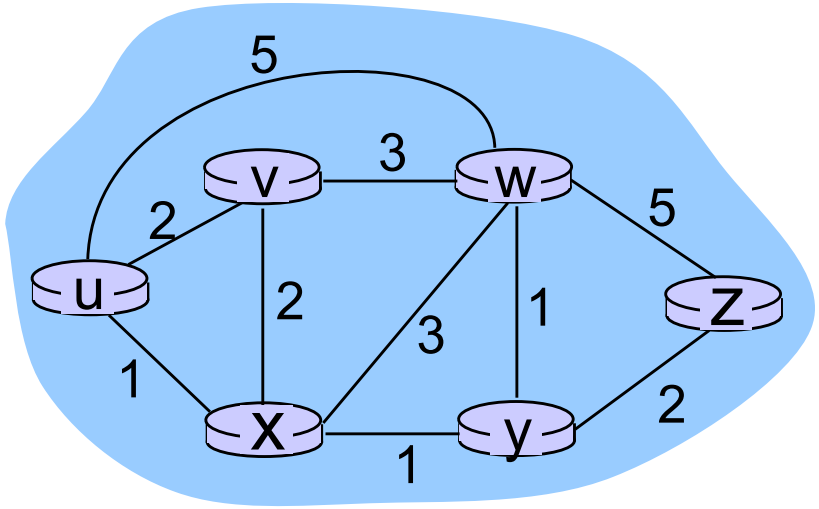
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding



Graph abstraction

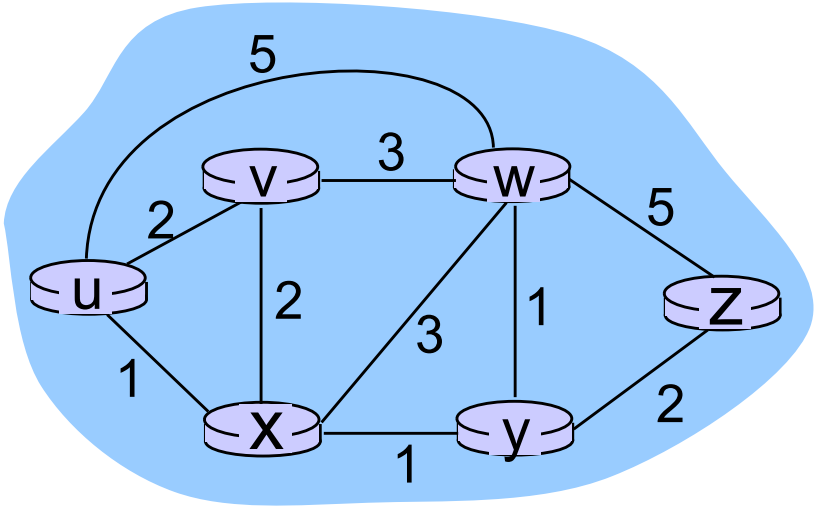


graph: G = (N,E)

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

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

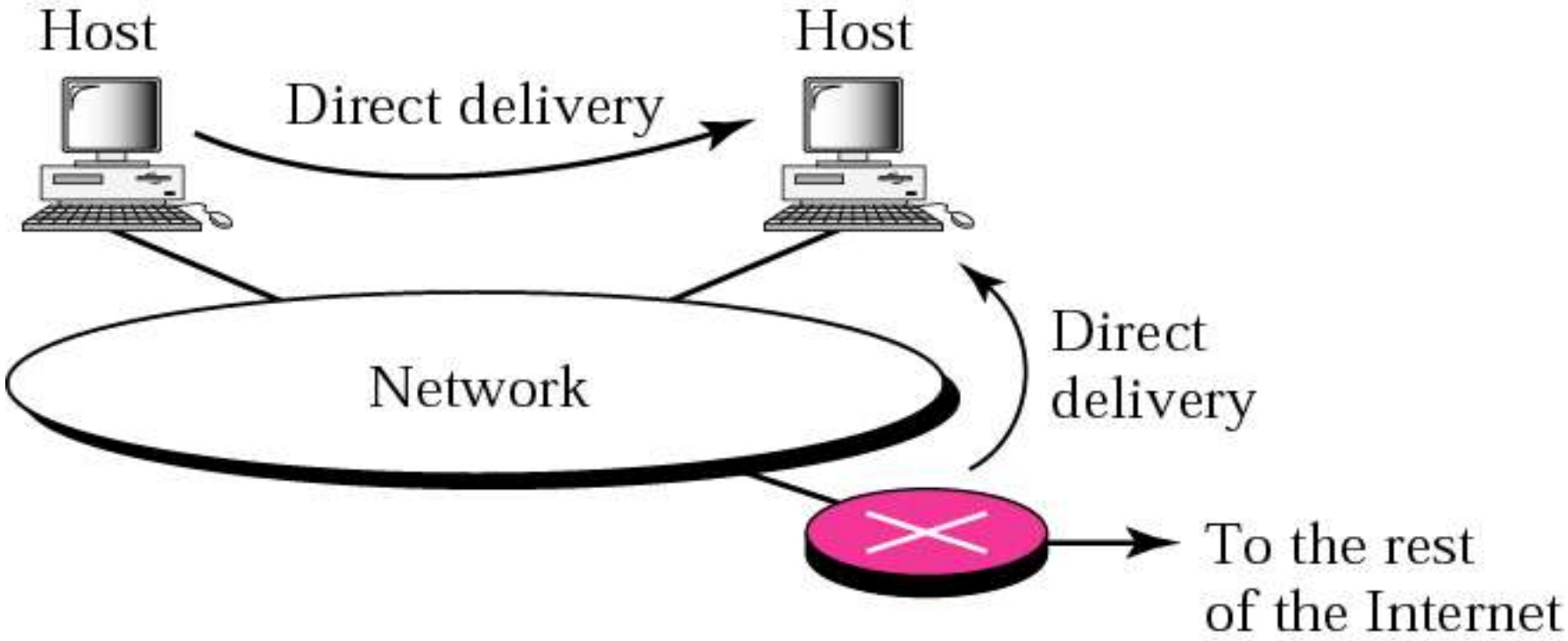
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

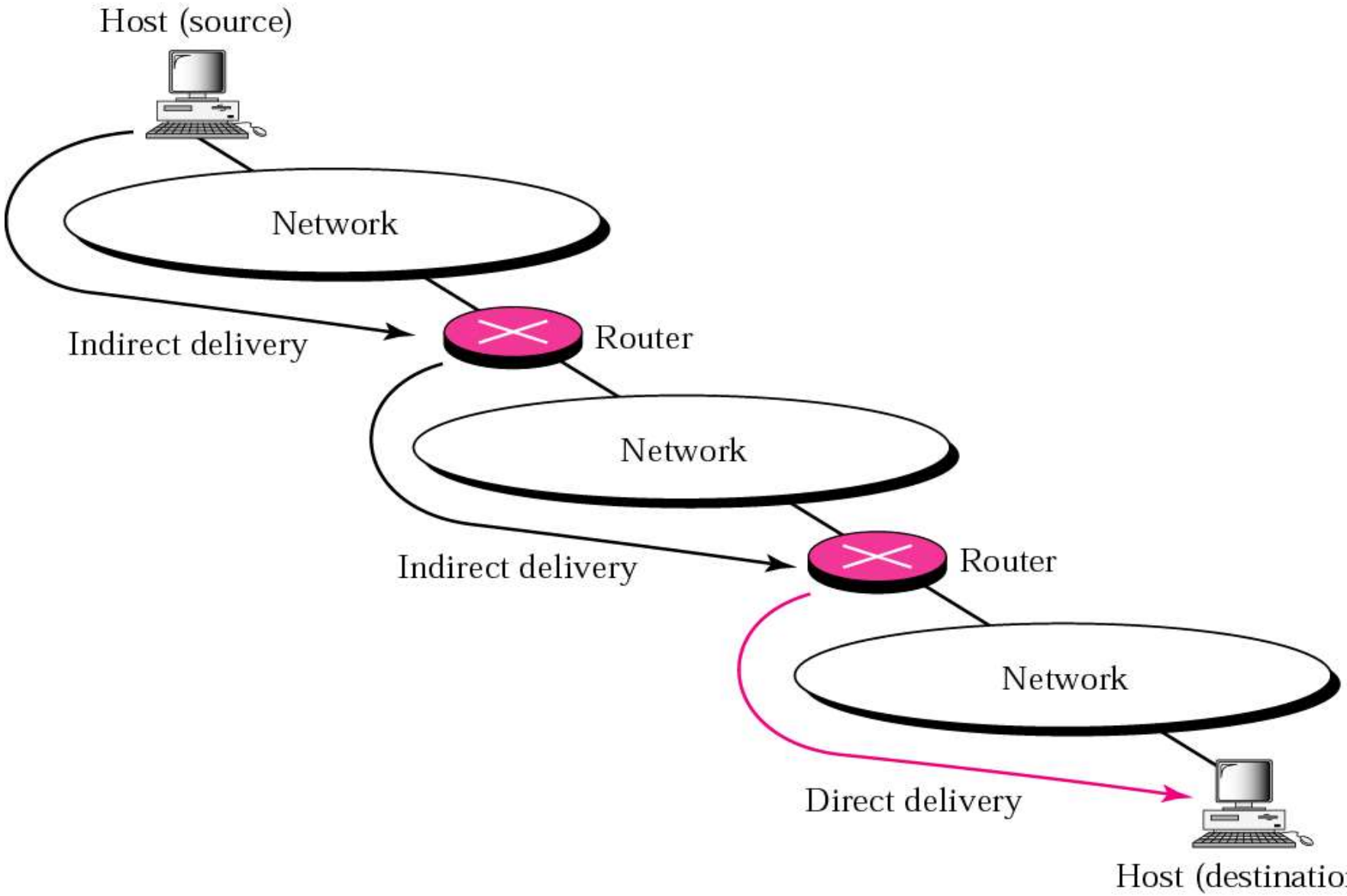
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Direct delivery



Indirect delivery



A Link-State Routing Algorithm

Dijkstra's algorithm

- net 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 dest.'s

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

Dijsktra's Algorithm

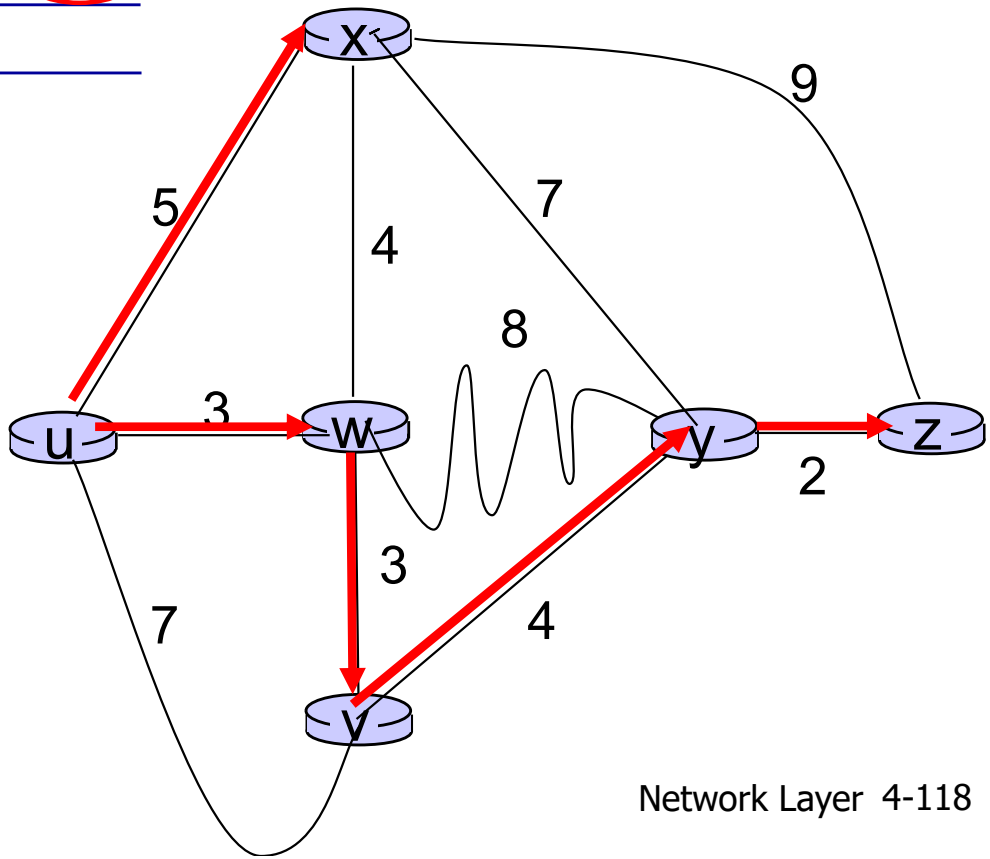
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + 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'
```

Dijkstra's algorithm: 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	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

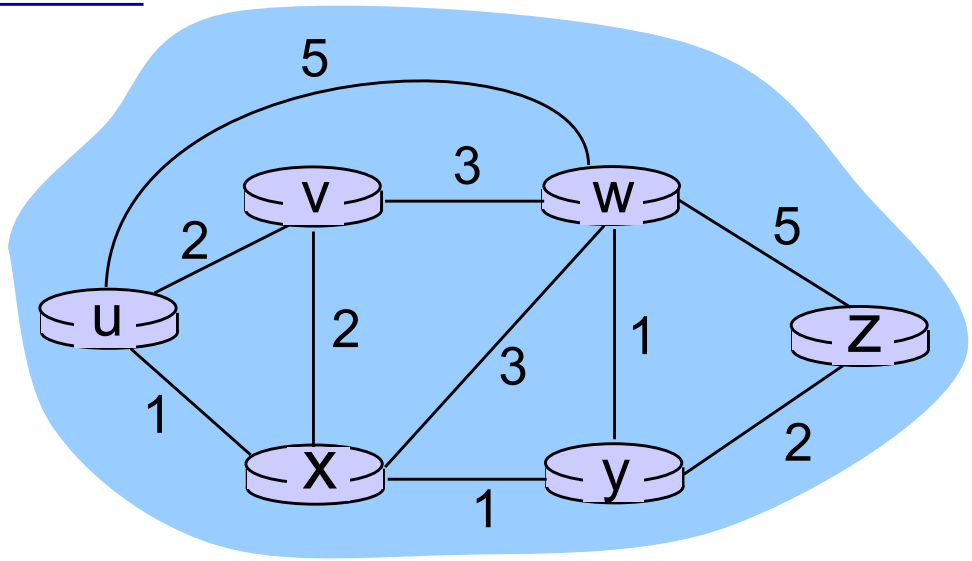
notes:

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



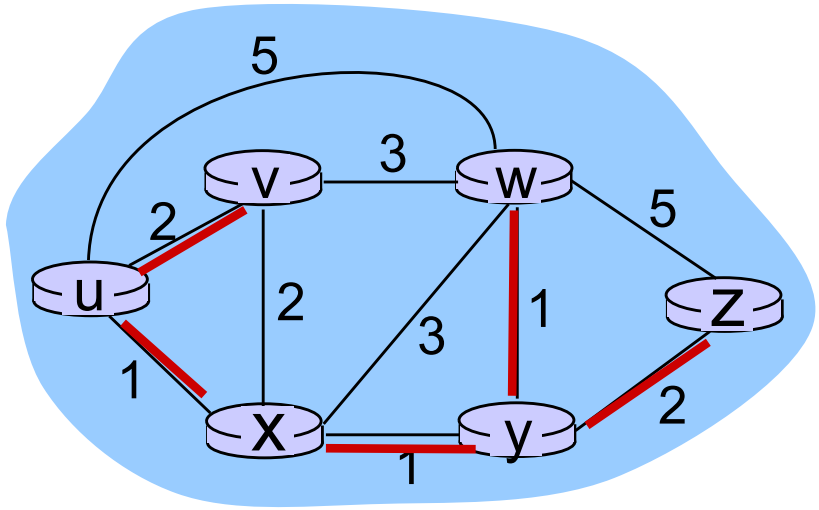
Exercise : Dijkstra's algorithm

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



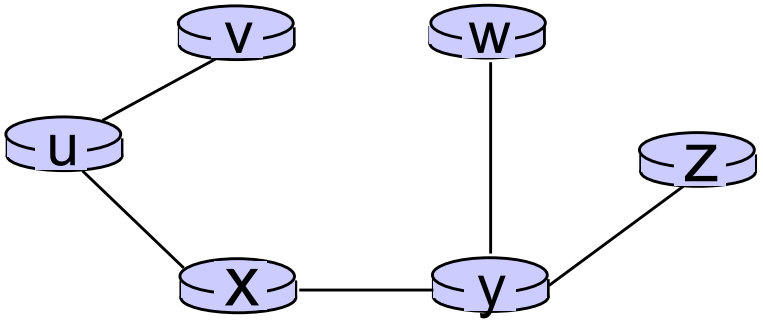
Dijkstra's algorithm: another example

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



Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

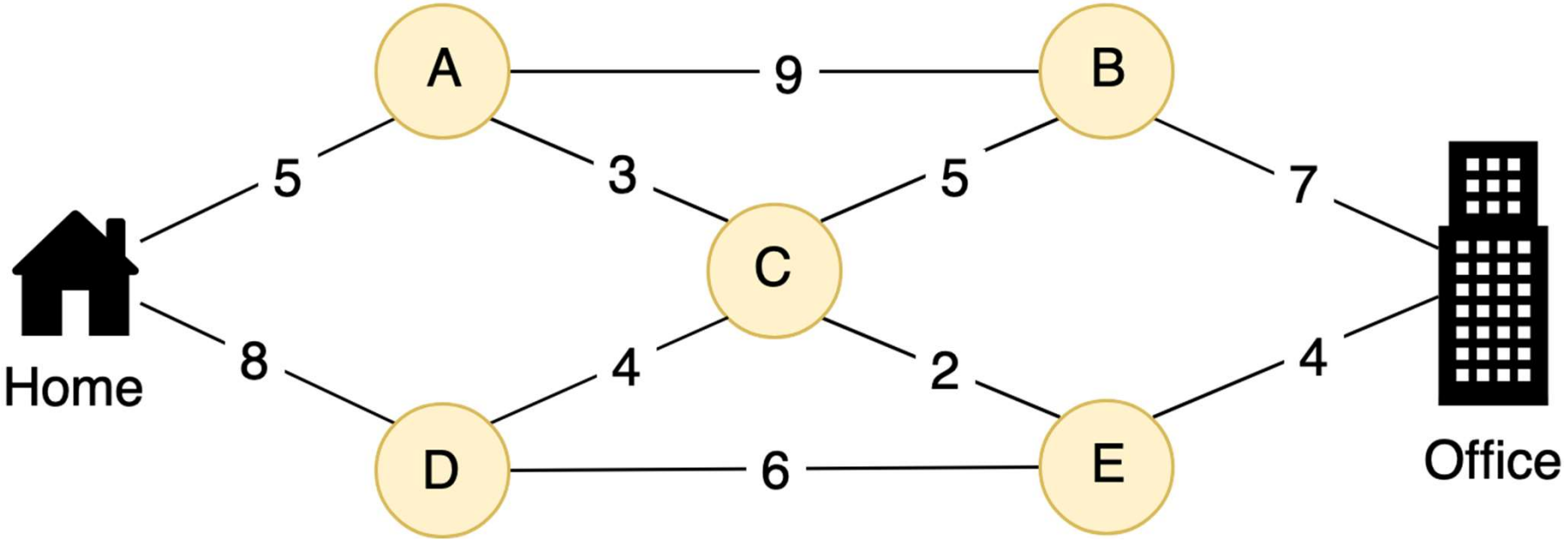
destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Exercise : Dijkstra's algorithm

- ไปที่ https://www.101computing.net/dijkstras-shortest-path-algorithm/
- ดูตัวอย่างเพิ่มเติม https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/

Exercise : Dijkstra's algorithm

- เครือข่ายจาก Home ไป Office ผ่าน Router ดังรูป จงหาเส้นทางที่สั้นที่สุดโดยใช้ Dijkstra's Algorithm และแสดง Forwarding Table ของ Router ที่ข้อมูลผ่าน



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

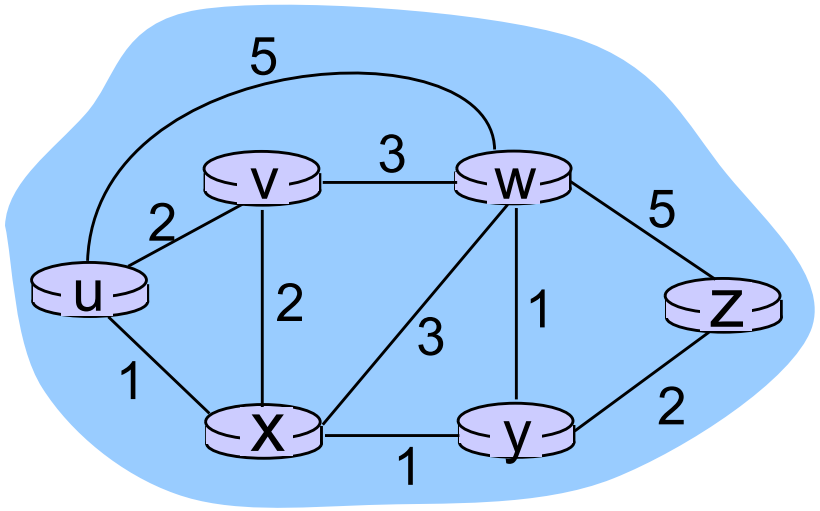
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min 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:

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

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

Distance vector algorithm

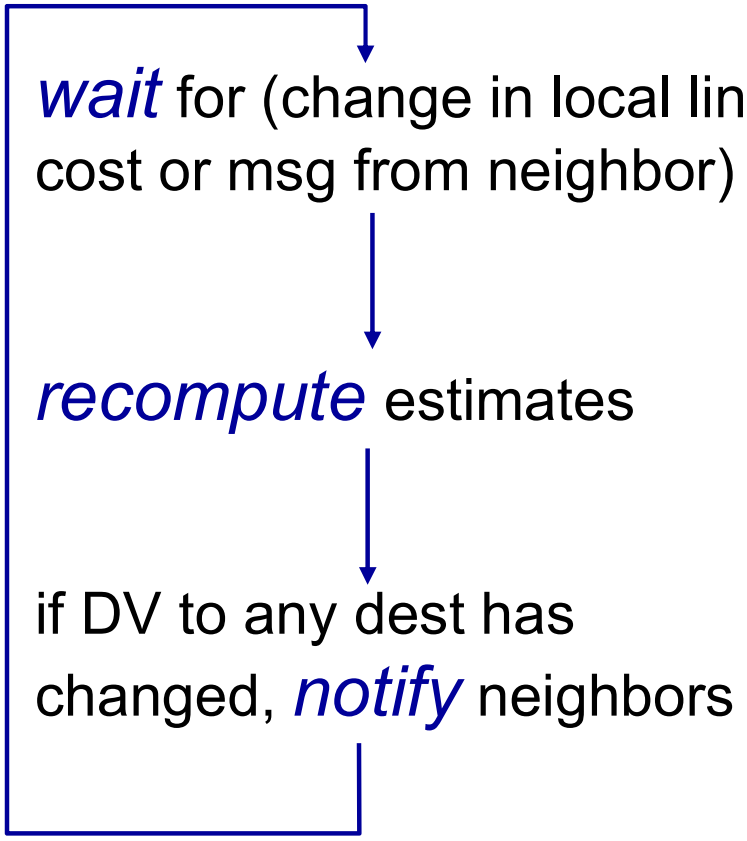
iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - 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	cost to y	cost to z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

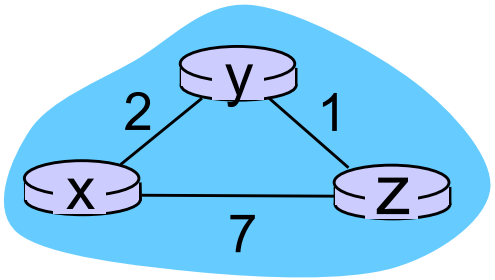
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

from	cost to x	cost to y	cost to 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	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

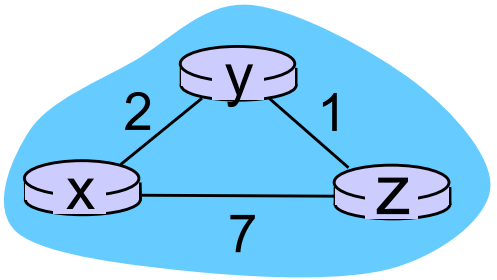
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

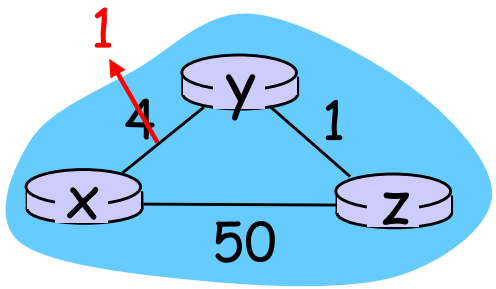


time

Distance vector: link cost changes

link cost changes:

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



"good news travels fast"

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

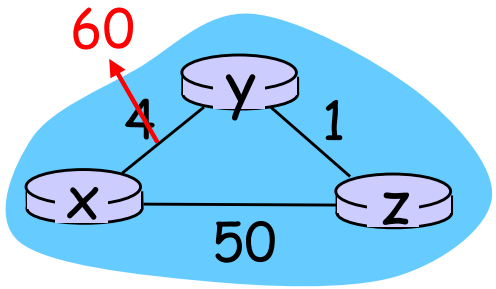
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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!
- 44 iterations before algorithm stabilizes: see text



poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Hierarchical routing

our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 600 million destinations:

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

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

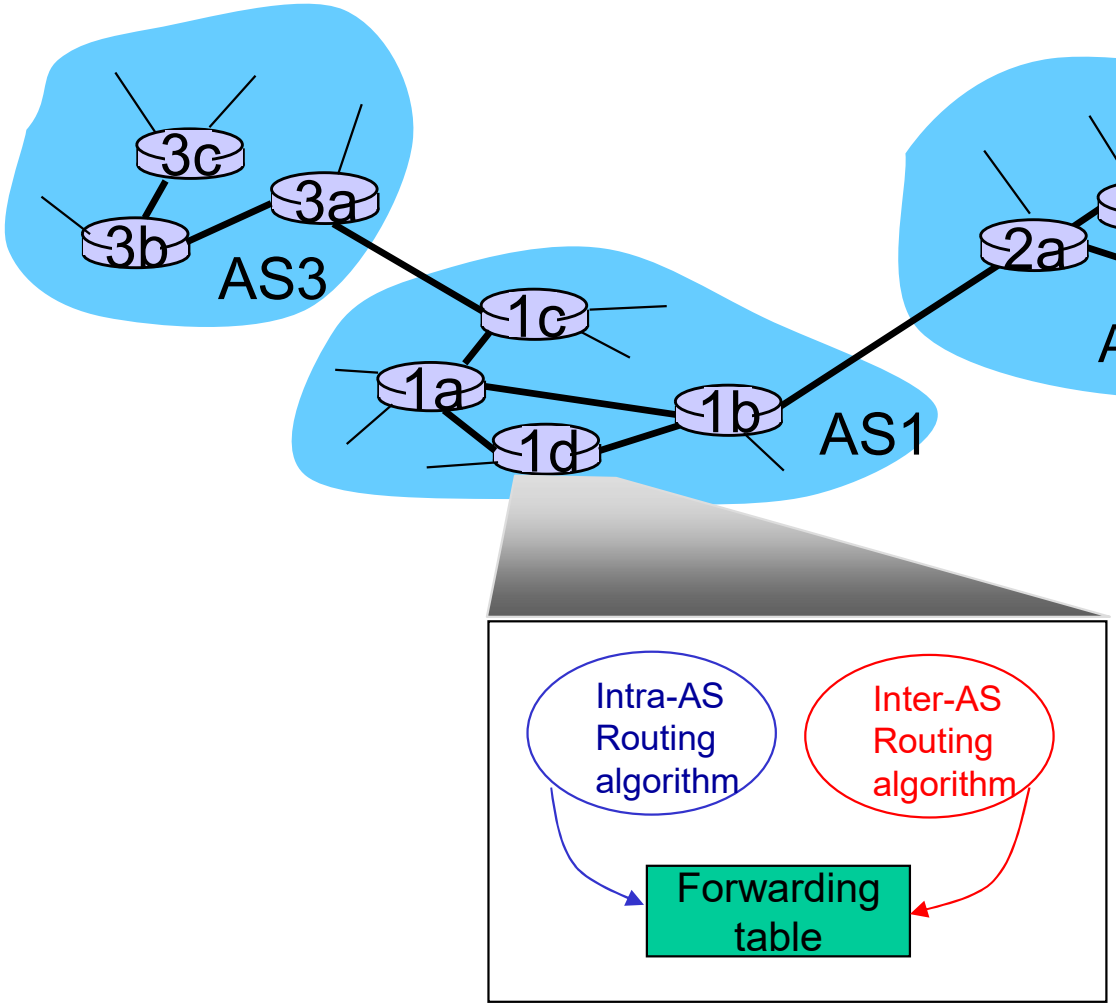
Hierarchical routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway router:

- at "edge" of its own AS
- has link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

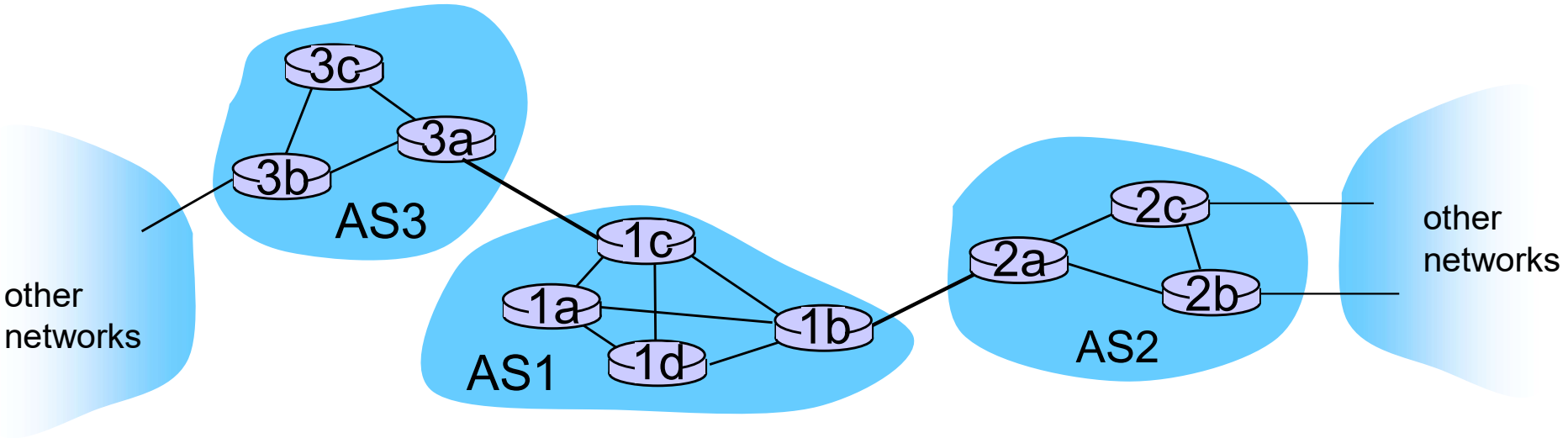
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

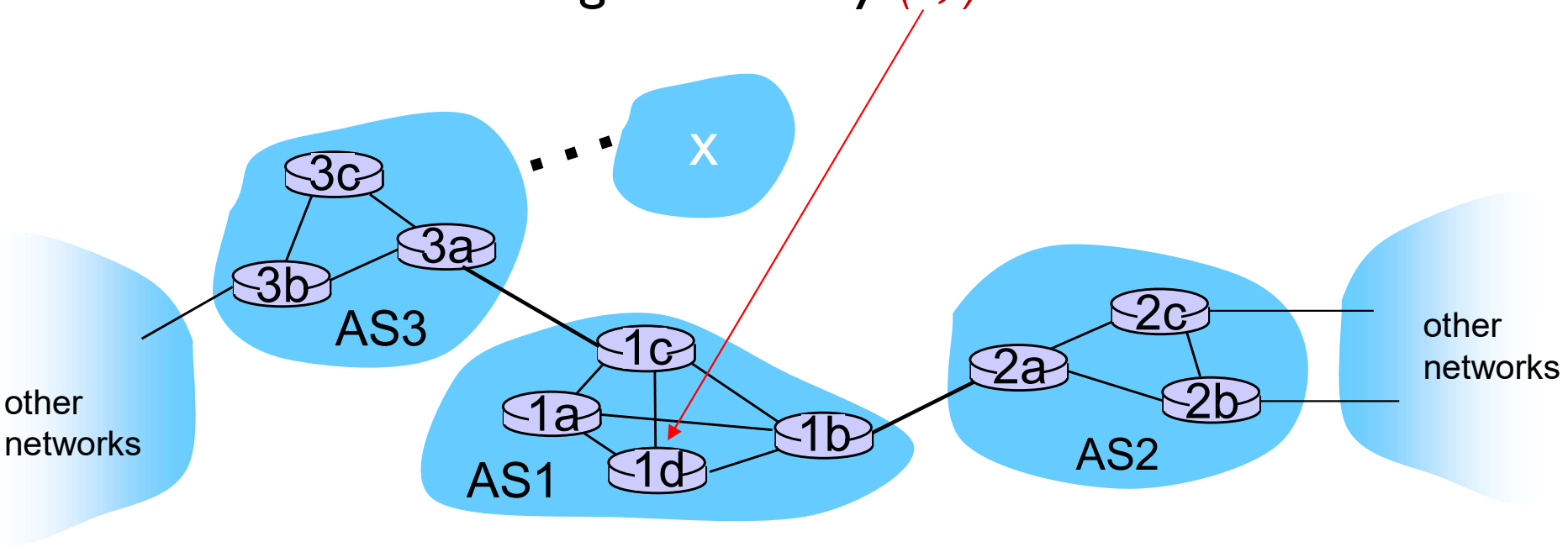
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



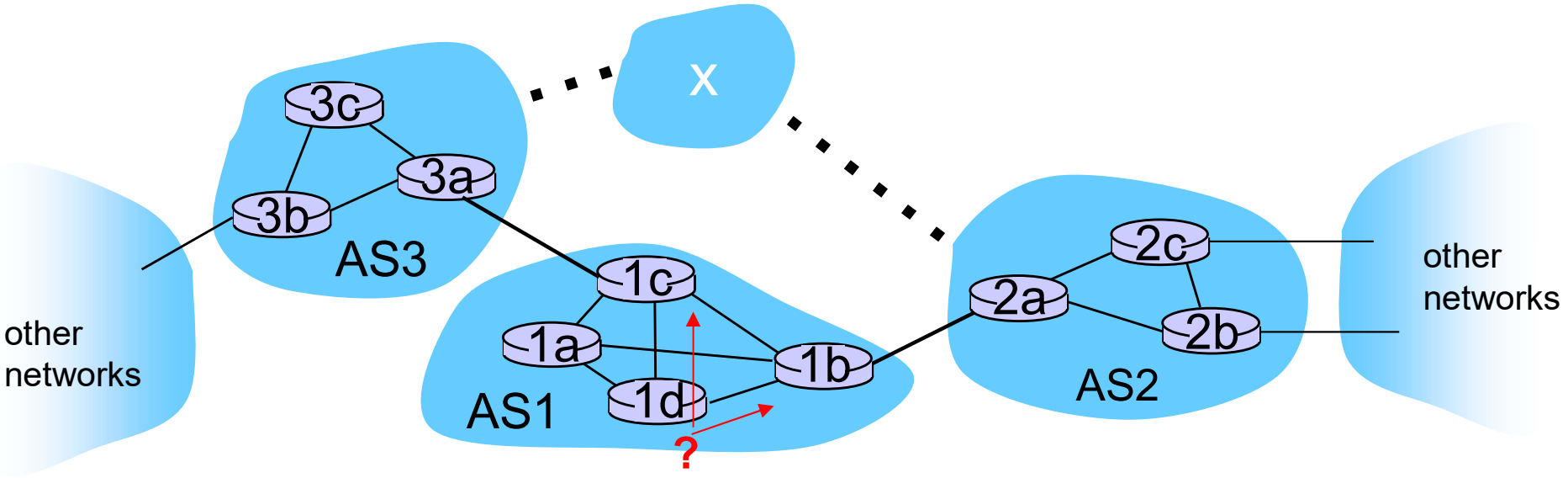
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



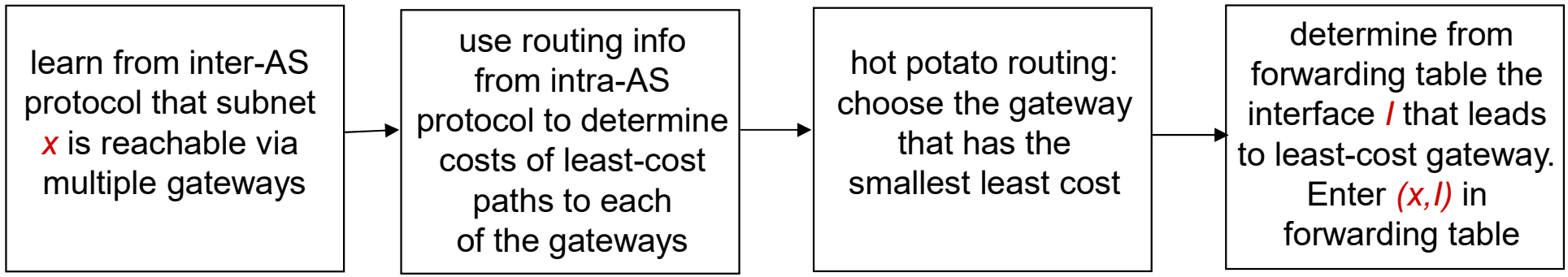
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol!



Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!
- *hot potato routing: send* packet towards closest of two routers.



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

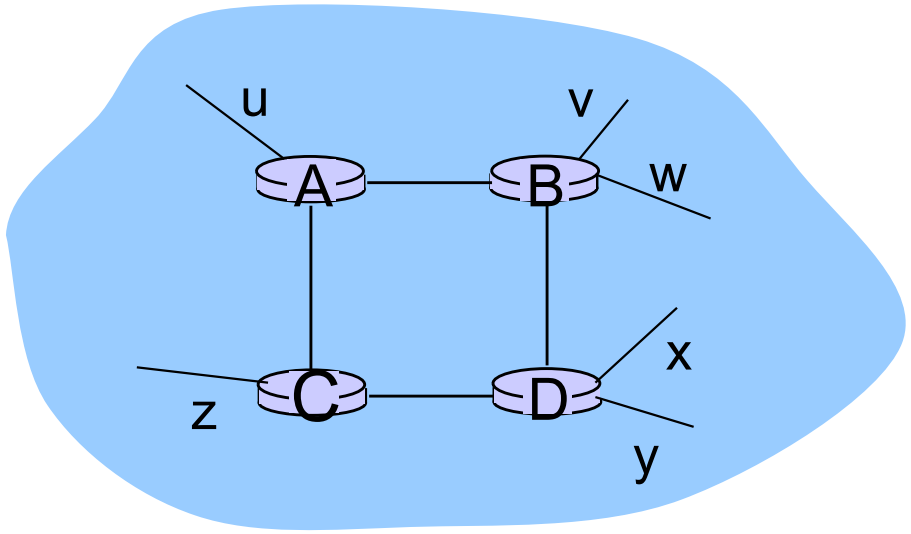
4.7 broadcast and multicast routing

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

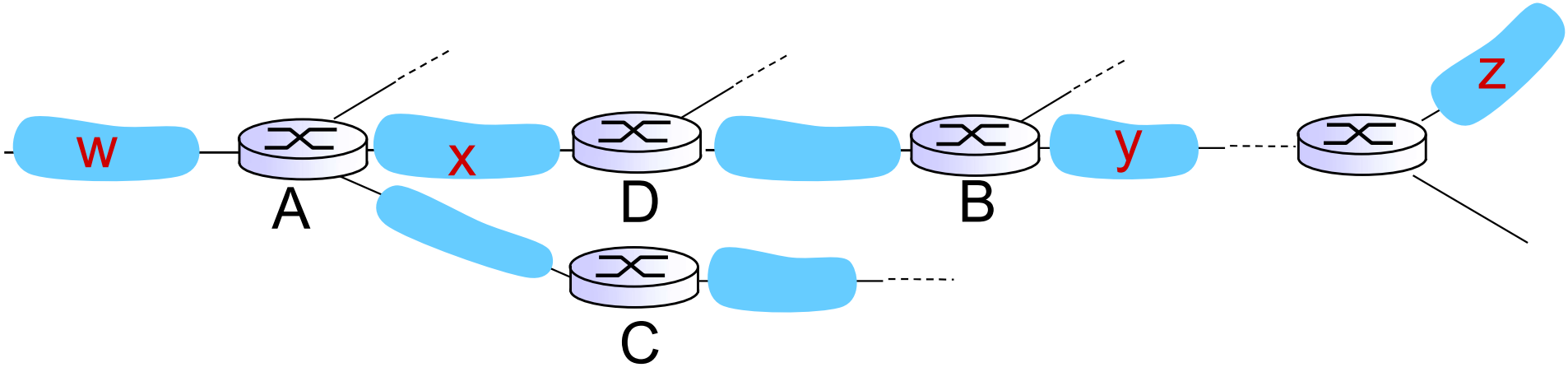
- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets (in IP addressing sense)*



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP: example



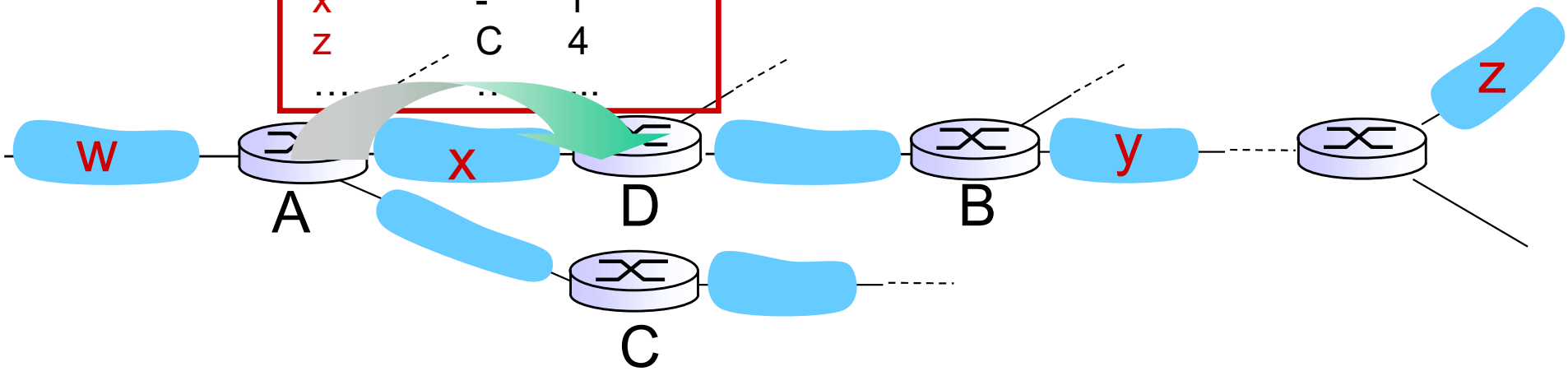
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

RIP: example

A-to-D advertisement

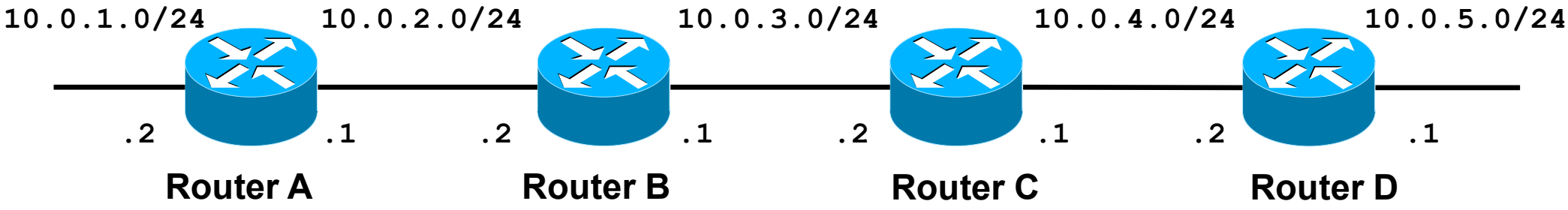
dest	next	hops
w	-	1
x	-	1
z	C	4
....	...	...



routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B A	7 5
x	--	1
....	....	....

RIP: example



Router A

Net	via	cost
t=0:		
10.0.1.0	-	0
10.0.2.0	-	0
t=1:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
t=2:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2

Router B

Net	via	cost
t=0:		
10.0.2.0	-	0
10.0.3.0	-	0
t=1:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1

Router C

Net	via	cost
t=0:		
10.0.3.0	-	0
10.0.4.0	-	0
t=1:		
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1
t=2:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1

Router D

Net	via	cost
t=0:		
10.0.4.0	-	0
10.0.5.0	-	0
t=1:		
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0
t=2:		
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0

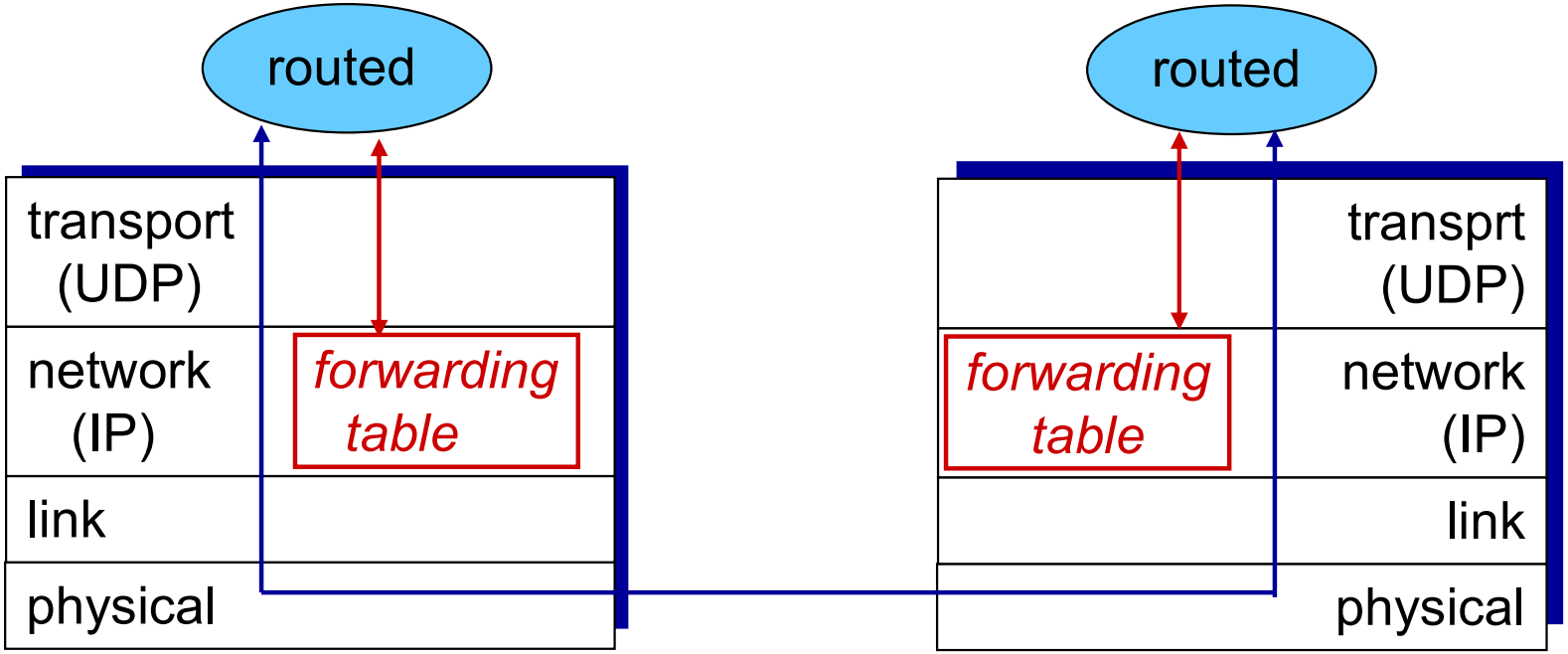
RIP: link failure, recovery

if no advertisement heard after 180 sec -->
neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP table processing

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



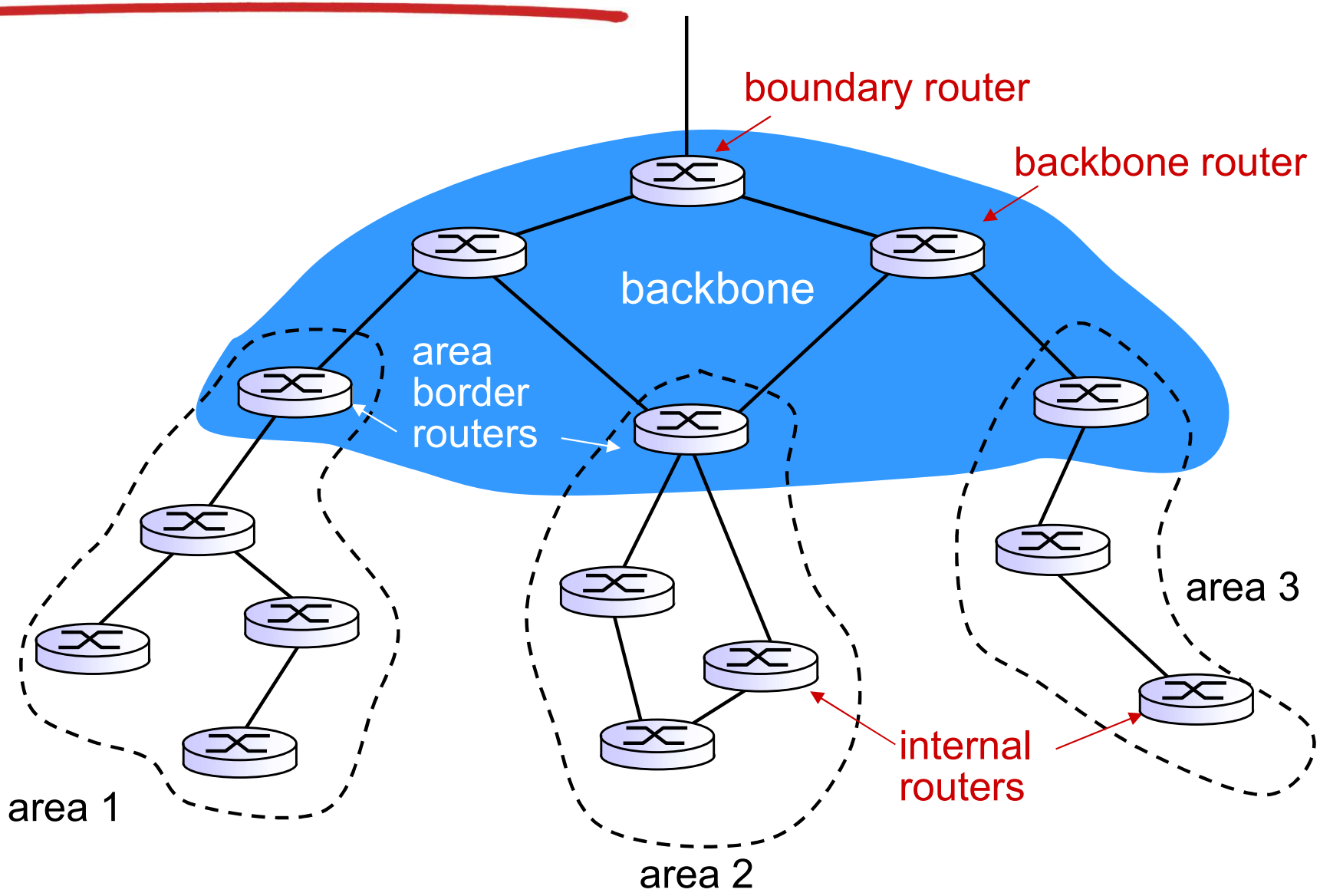
OSPF (Open Shortest Path First)

- "open": publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF "advanced" features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

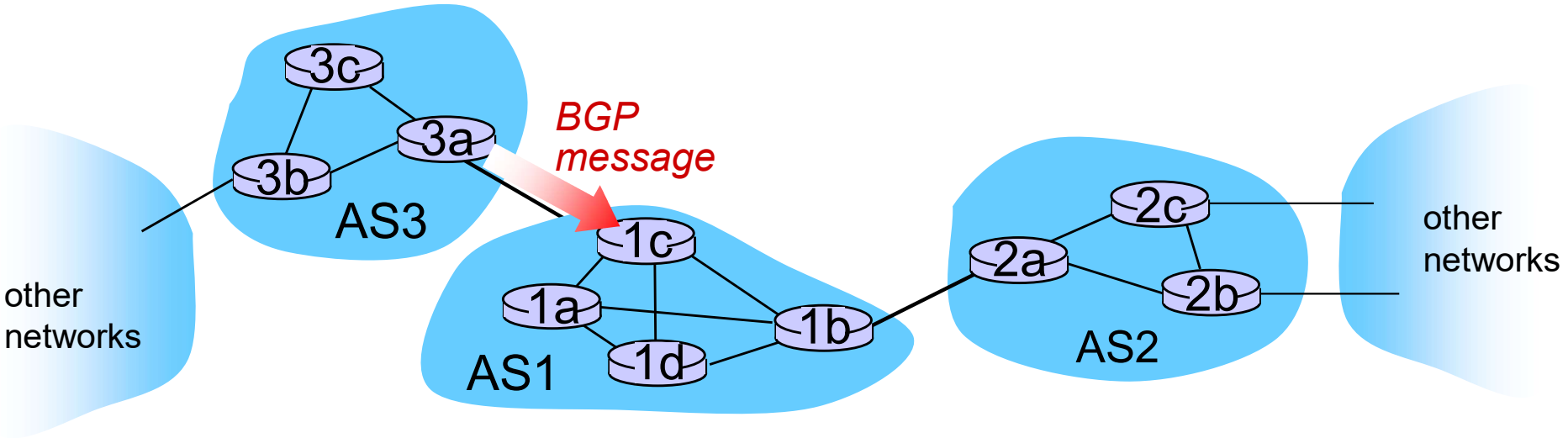
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

BGP basics

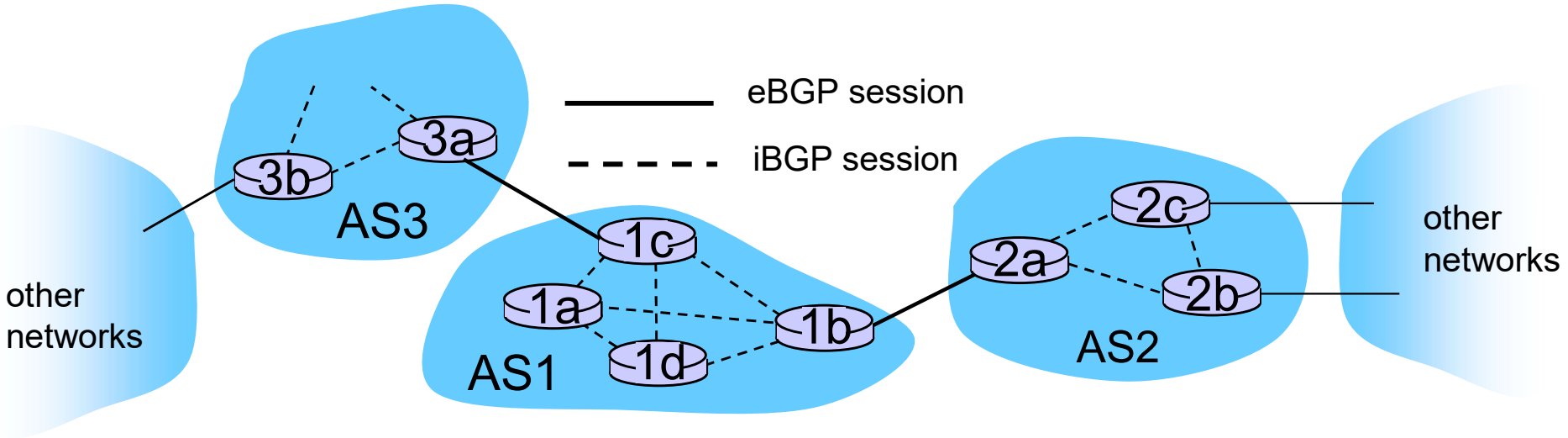
- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections

- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- gateway router receiving route advertisement uses import policy to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing

BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

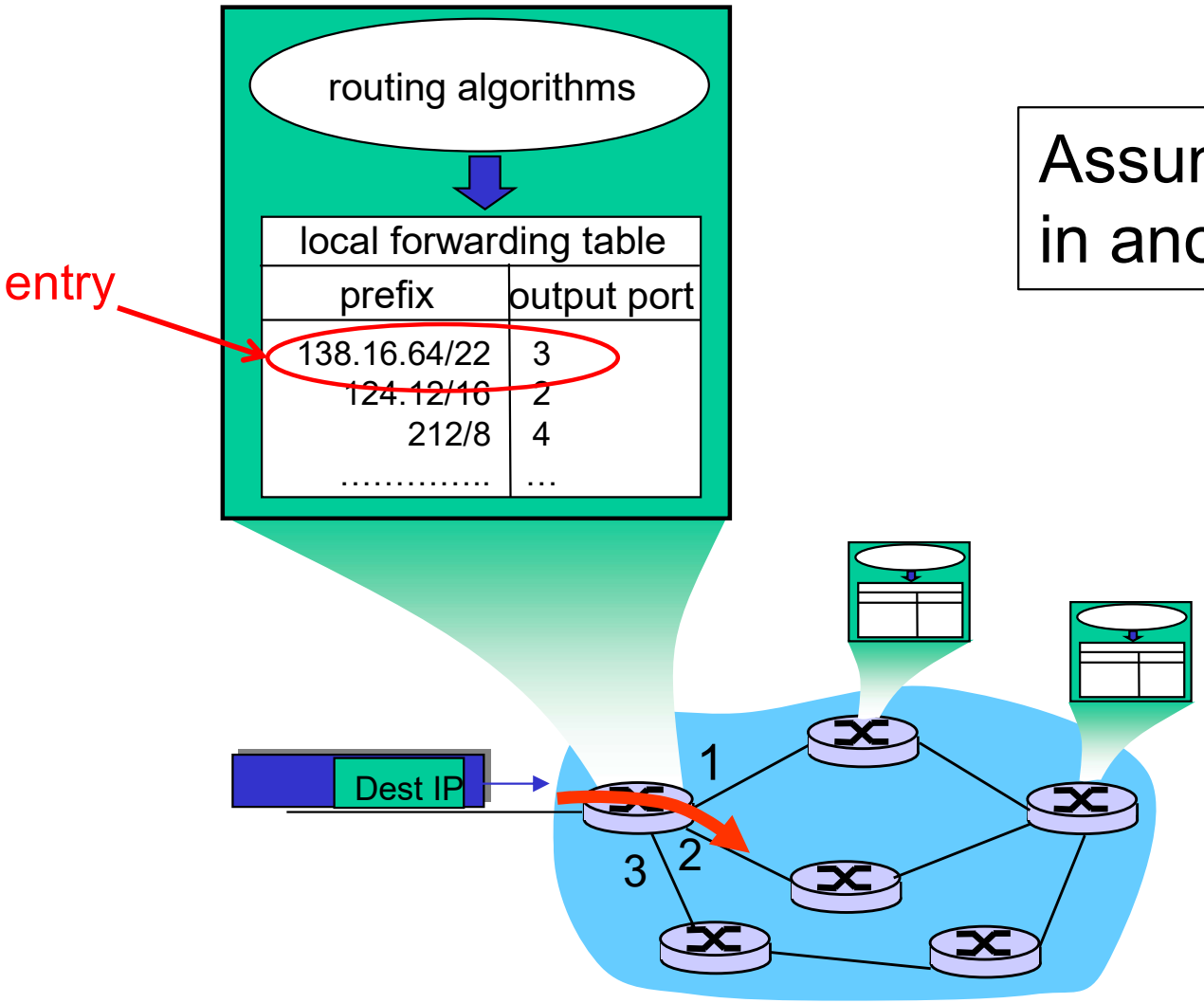
BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

Putting it Altogether: *How Does an Entry Get Into a Router's Forwarding Table?*

- Answer is complicated!
- Ties together hierarchical routing (Section 4.5.3) with BGP (4.6.3) and OSPF (4.6.2).
- Provides nice overview of BGP!

How does entry get in forwarding table?



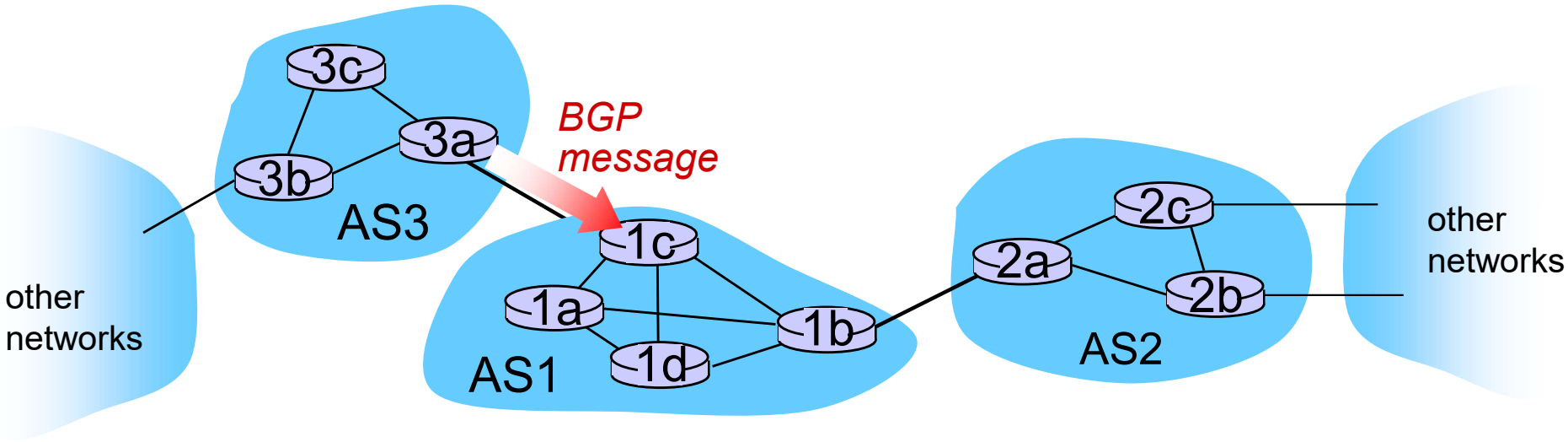
Assume prefix is in another AS.

How does entry get in forwarding table?

High-level overview

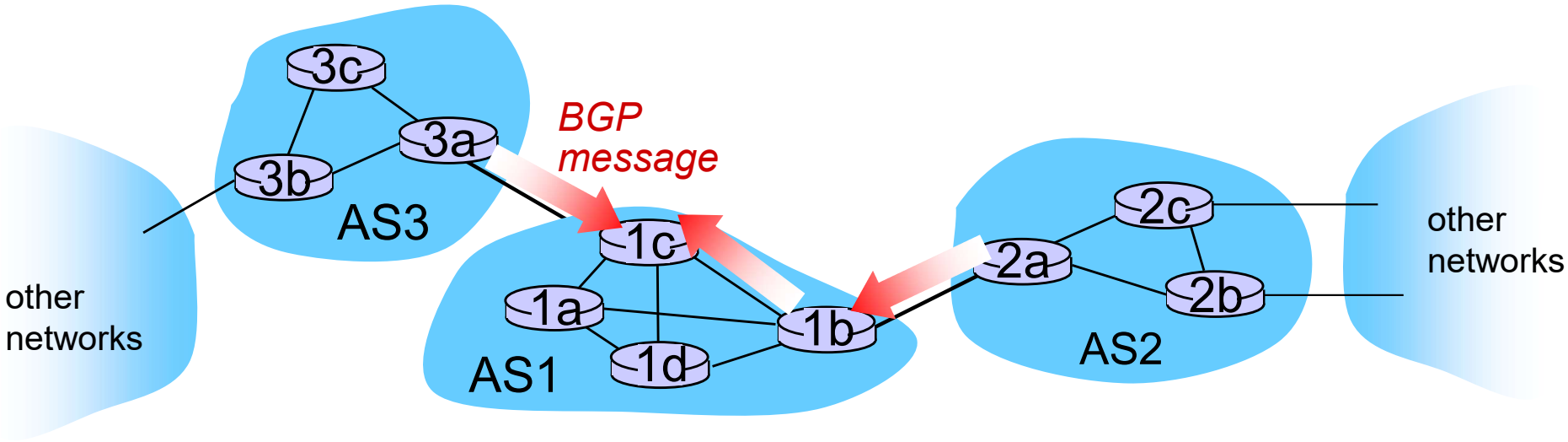
1. Router becomes aware of prefix
2. Router determines output port for prefix
3. Router enters prefix-port in forwarding table

Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

Router may receive multiple routes



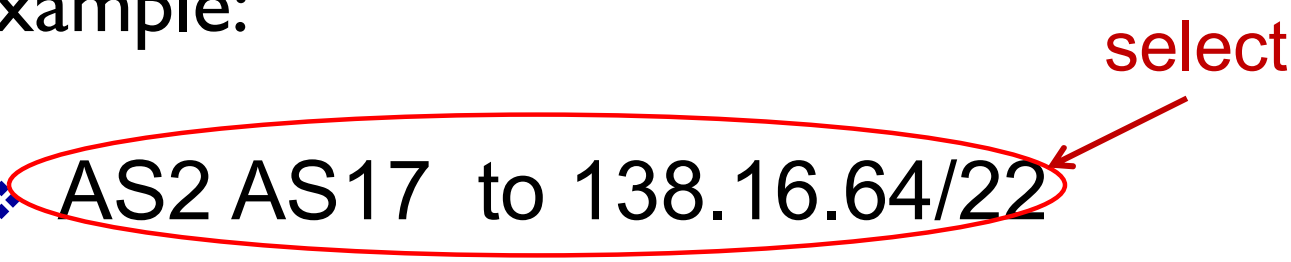
- Router may receive multiple routes for same prefix
- Has to select one route

Select best BGP route to prefix

- Router selects route based on shortest AS-PATH

- Example:

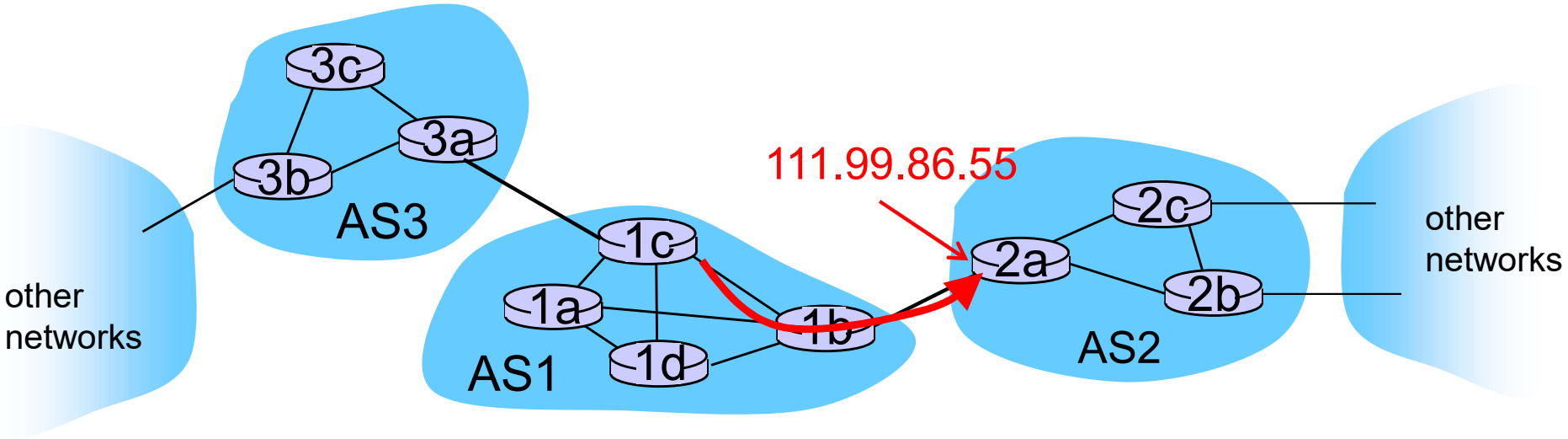
  - AS2 AS17 to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22



- What if there is a tie? We'll come back to that!

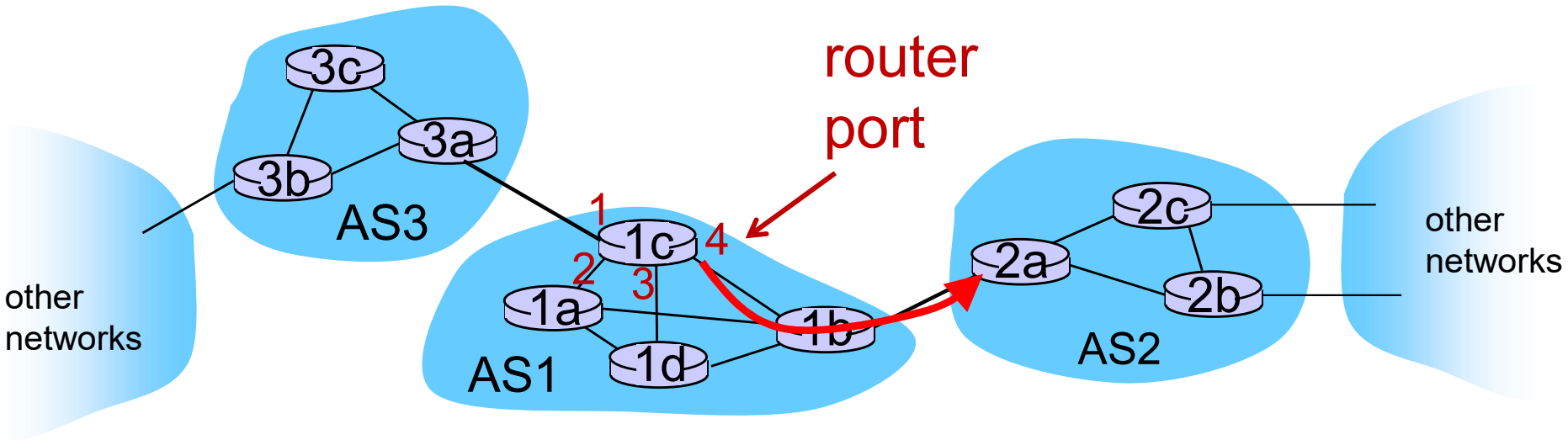
Find best intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



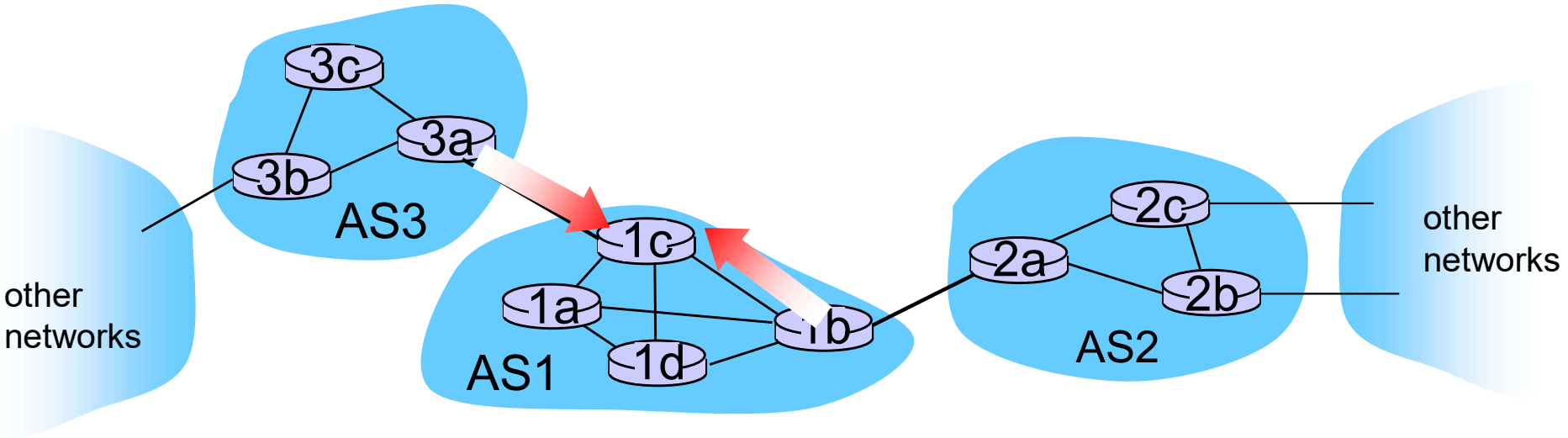
Router identifies port for route

- Identifies port along the OSPF shortest path
- Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



Hot Potato Routing

- Suppose there two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS201 since it is closer

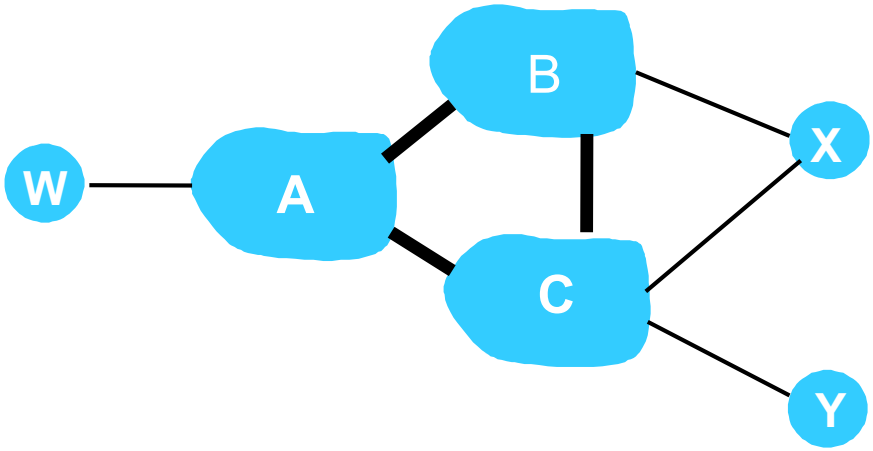


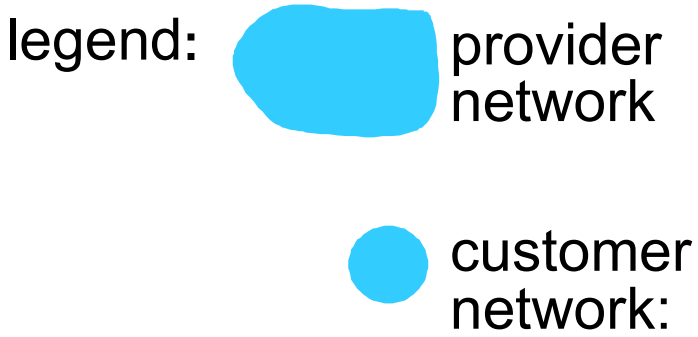
How does entry get in forwarding table?

Summary

1. Router becomes aware of prefix
   - via BGP route advertisements from other routers
2. Determine router output port for prefix
   - Use BGP route selection to find best inter-AS route
   - Use OSPF to find best intra-AS route leading to best inter-AS route
   - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

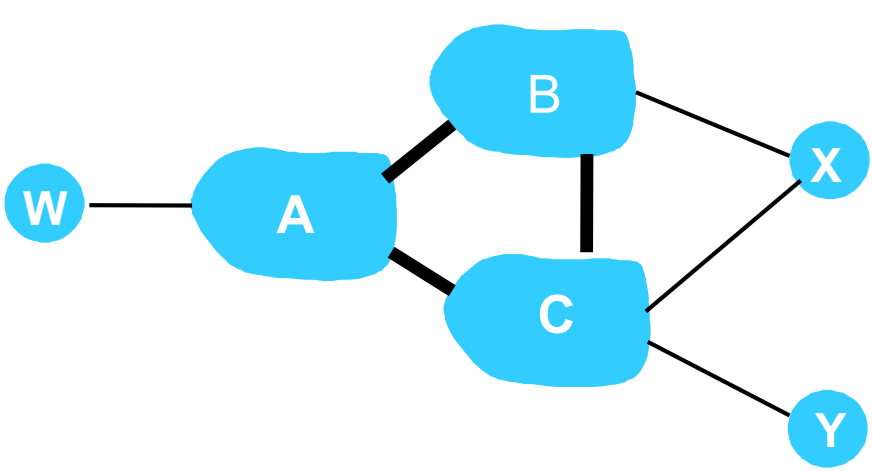
BGP routing policy

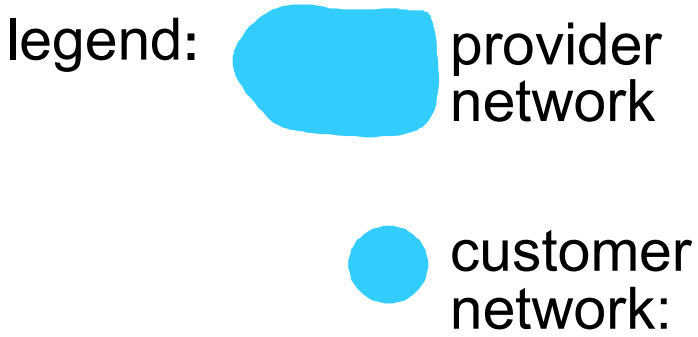




- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)





- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

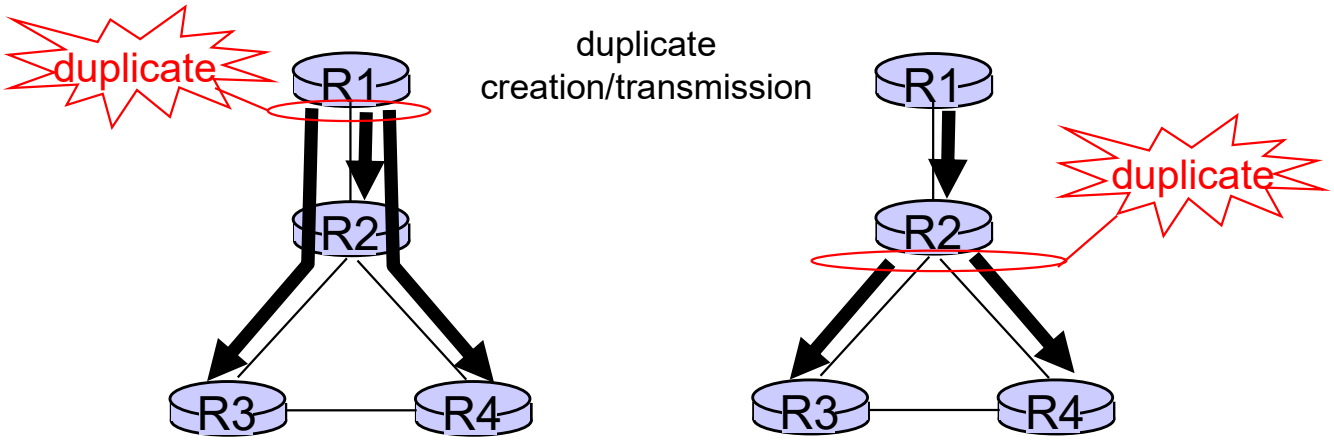
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Broadcast routing

- deliver packets from source to all other nodes
- source duplication is inefficient:



source duplication

in-network duplication

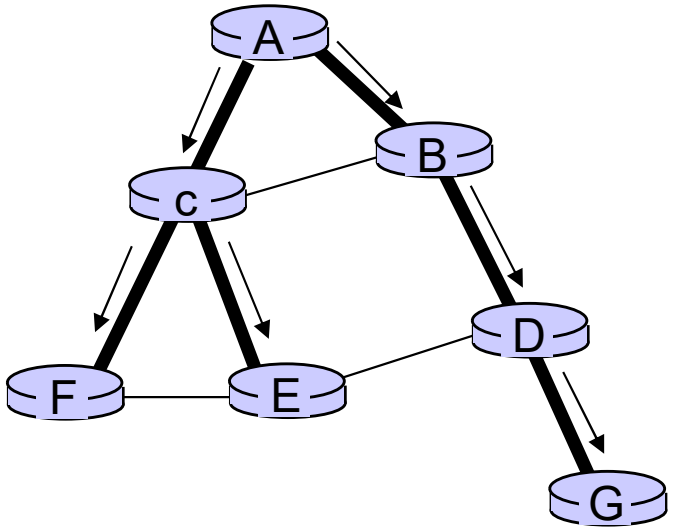
- source duplication: how does source determine recipient addresses?

In-network duplication

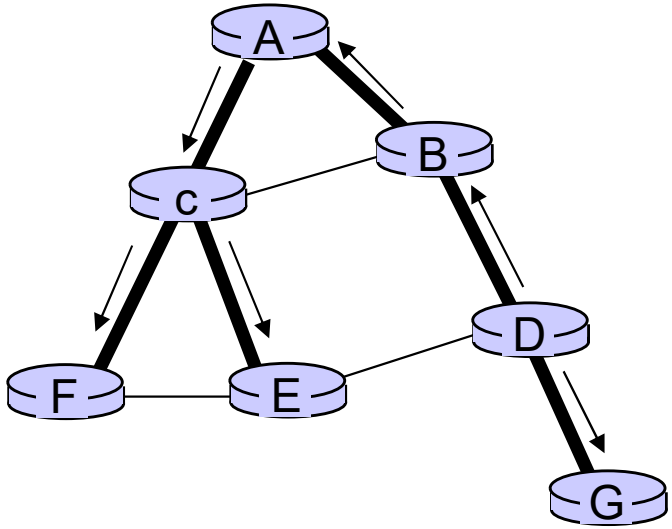
- *flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadacsted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

Spanning tree

- first construct a spanning tree
- nodes then forward/make copies only along spanning tree



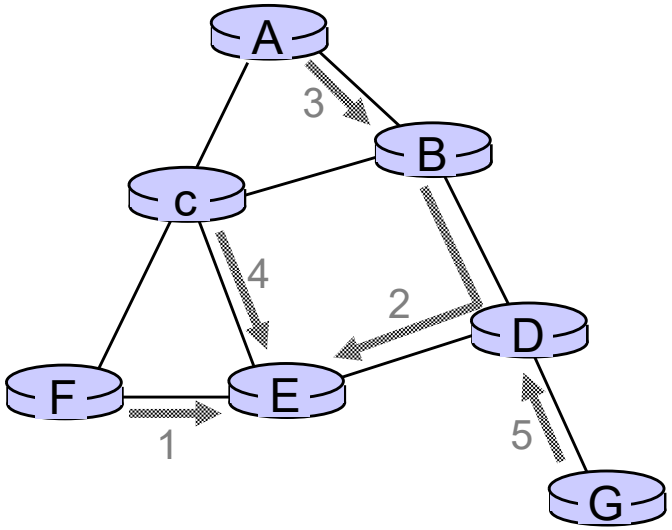
(a) broadcast initiated at A



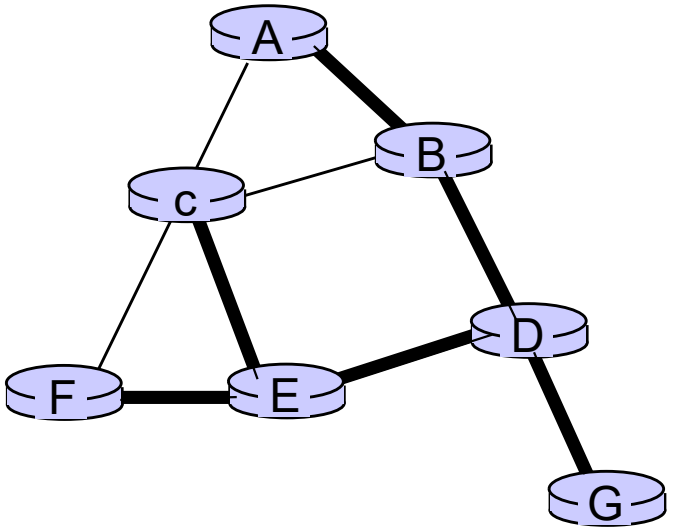
(b) broadcast initiated at D

Spanning tree: creation

- center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree



(a) stepwise construction of spanning tree (center: E)

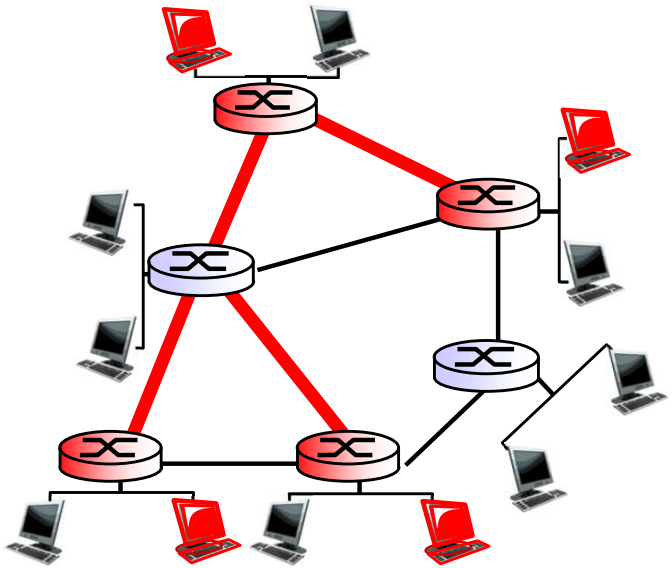


(b) constructed spanning tree

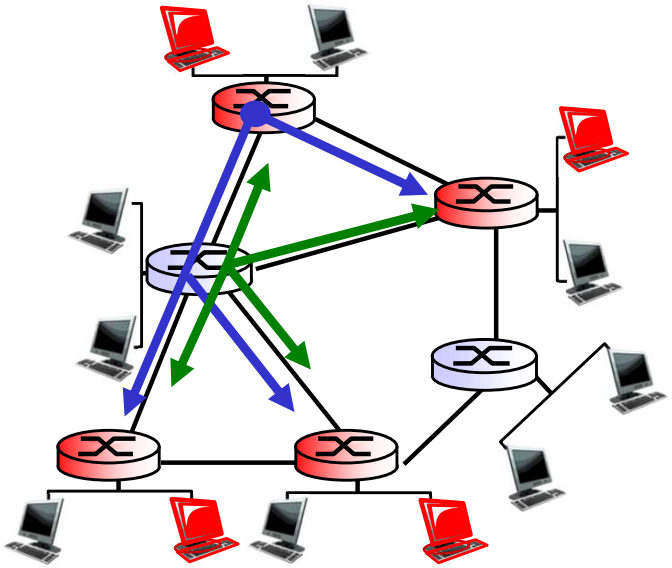
Multicast routing: problem statement

goal: find a tree (or trees) connecting routers having local mcast group members

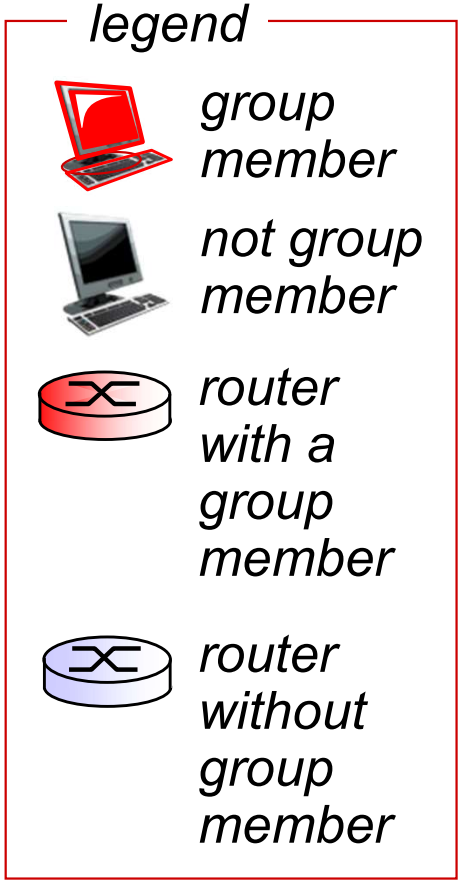
- *tree:* not all paths between routers used
- *shared-tree:* same tree used by all group members
- *source-based:* different tree from each sender to rcvrs



shared tree



source-based trees



Approaches for building mcast trees

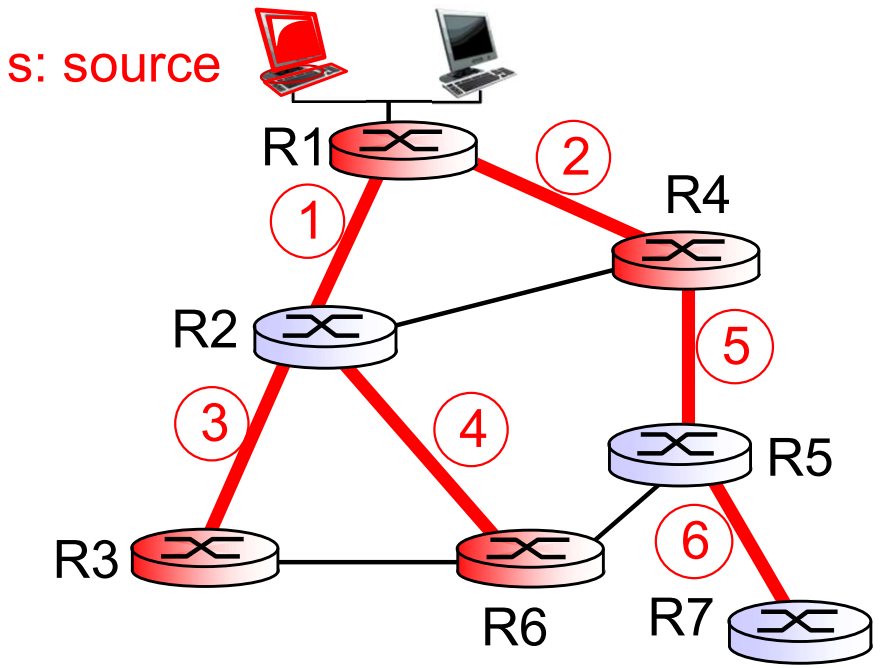
approaches:

- *source-based tree:* one tree per source
  - shortest path trees
  - reverse path forwarding
- *group-shared tree:* group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest path tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm



LEGEND

router with attached group member

router with no attached group member

i — link used for forwarding, i indicates order link added by algorithm

Reverse path forwarding

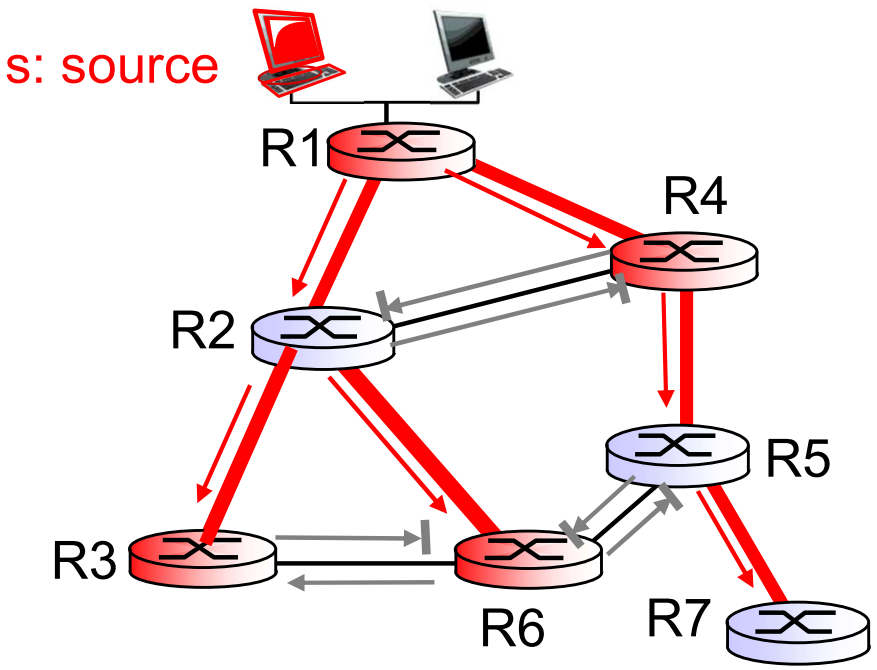
- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)

then flood datagram onto all outgoing links

else ignore datagram

Reverse path forwarding: example



LEGEND

router with attached group member

router with no attached group member

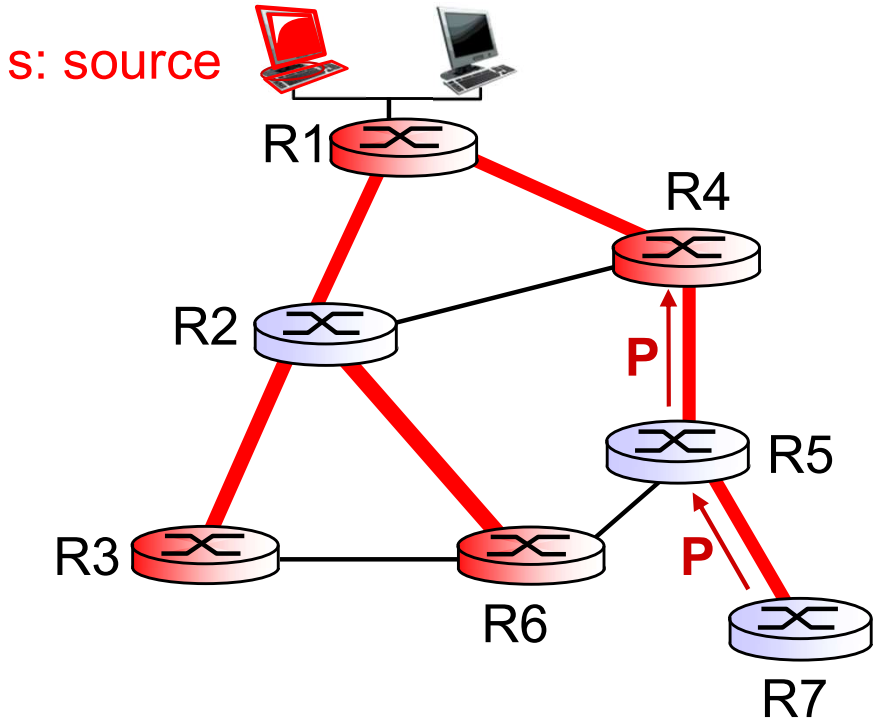
datagram will be forwarded

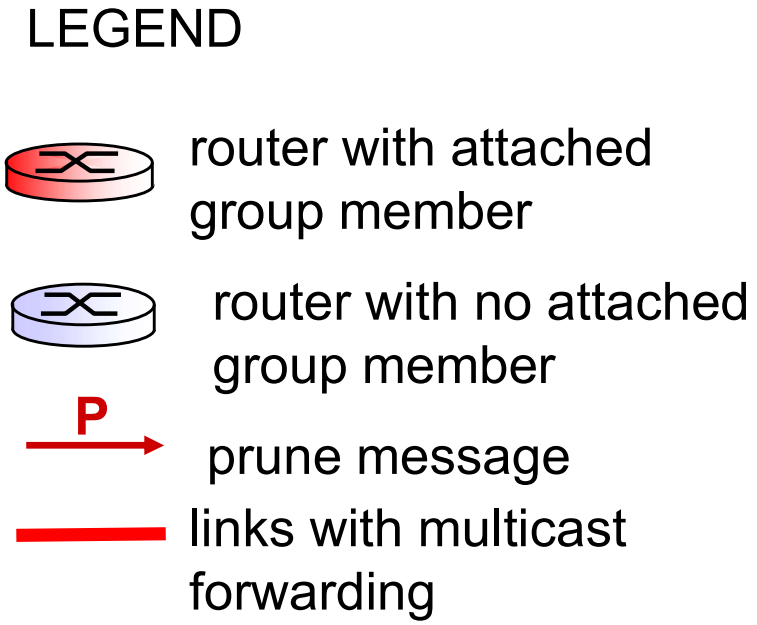
datagram will not be forwarded

- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members





Shared-tree: steiner tree

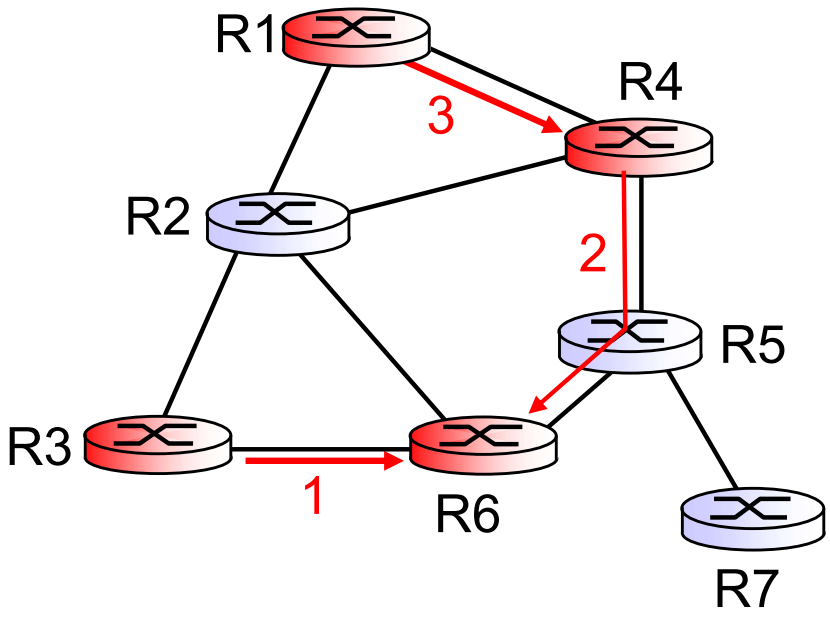
- *steiner tree:* minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: example

suppose R6 chosen as center:



LEGEND

router with attached group member

router with no attached group member

1 path order in which join messages generated

Internet Multicasting Routing: DVMRP

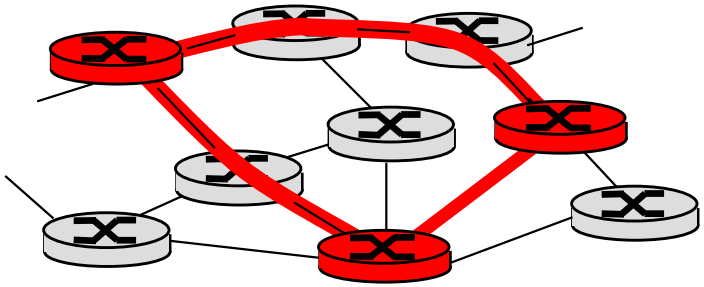
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

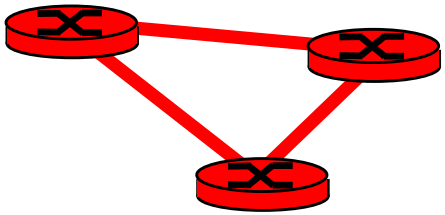
- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial router

Tunneling

Q: how to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router (recall IPv6 inside IPv4 tunneling)
- receiving mcast router unencapsulates to get mcast datagram

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios :

dense:

- group members densely packed, in "close" proximity.
- bandwidth more plentiful

sparse:

- # networks with group members small wrt # interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of sparse-dense dichotomy:

dense

- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

sparse:

- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

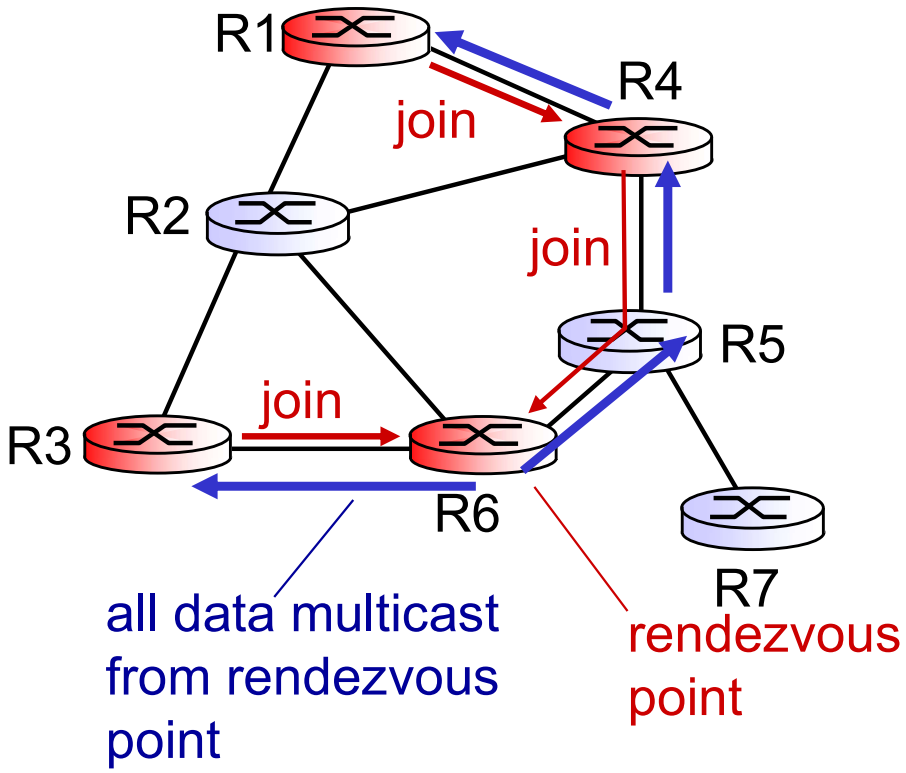
PIM- dense mode

flood-and-prune RPF: similar to DVMRP but…

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM - sparse mode

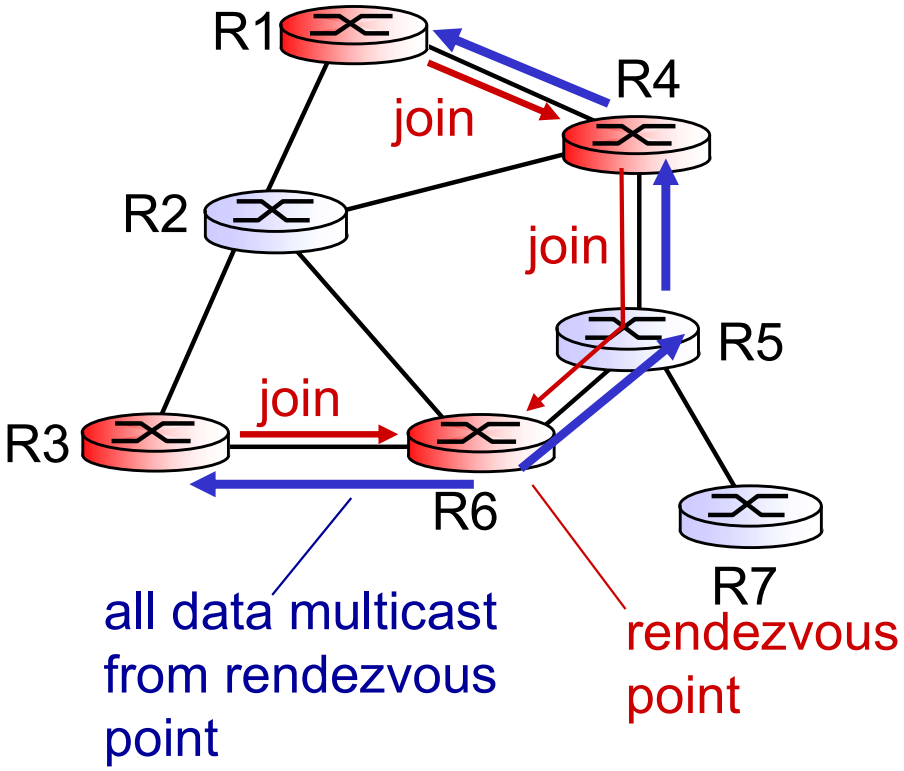
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM - sparse mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers
  - "no one is listening!"



Chapter 4: *done!*

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format, IPv4 addressing, ICMP, IPv6

4.5 routing algorithms

- link state, distance vector, hierarchical routing

4.6 routing in the Internet

- RIP, OSPF, BGP

4.7 broadcast and multicast routing

- understand principles behind network layer services:
  - network layer service models, forwarding versus routing how a router works, routing (path selection), broadcast, multicast
- instantiation, implementation in the Internet