CMPE 150/L : Introduction to Computer Networks

Chen Qian

Computer Engineering

UCSC Baskin Engineering

Lecture 12

Chapter 4: outline

Datagram forwarding table

routing algorithm

local forwarding table	
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

4 billion IP addresses, so rather than list individual destination address
list *range* of addresses (aggregate table entries)

IP destination address in arriving packet's header

1

3 2
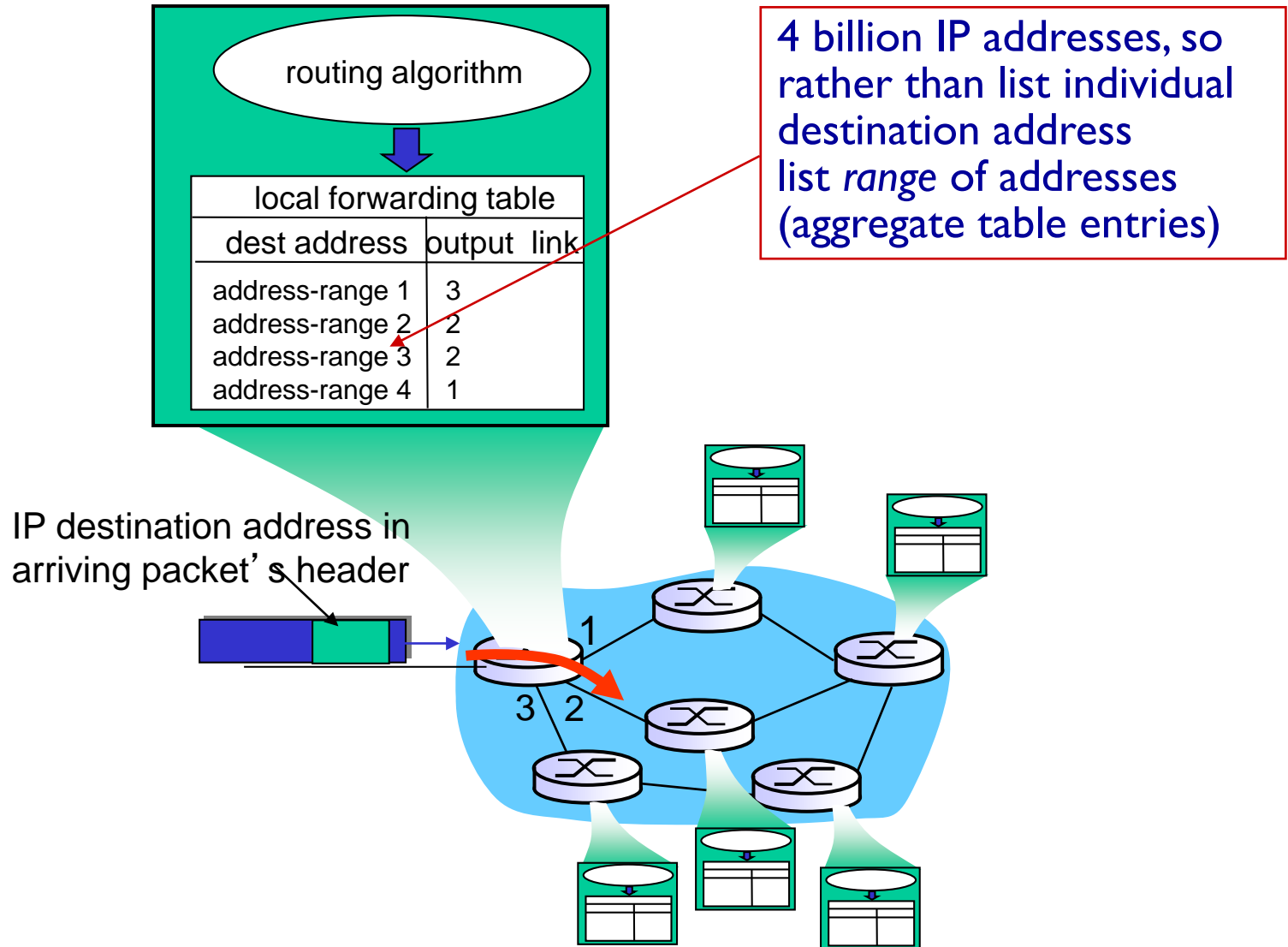
Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching
when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001   which interface?

DA: 11001000 00010111 00011000 10101010   which interface?

Datagram or VC network: why?

Internet (datagram)

- ❖ data exchange among computers
  - ▪ "elastic" service, no strict timing req.
- ❖ many link types
  - ▪ different characteristics
  - ▪ uniform service difficult
- ❖ "smart" end systems (computers)
  - ▪ can adapt, perform control, error recovery
  - ▪ *simple inside network, complexity at "edge"*

ATM (VC)

- ❖ evolved from telephony
- ❖ human conversation:
  - ▪ strict timing, reliability requirements
  - ▪ need for guaranteed service
- ❖ "dumb" end systems
  - ▪ telephones
  - ▪ *complexity inside 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
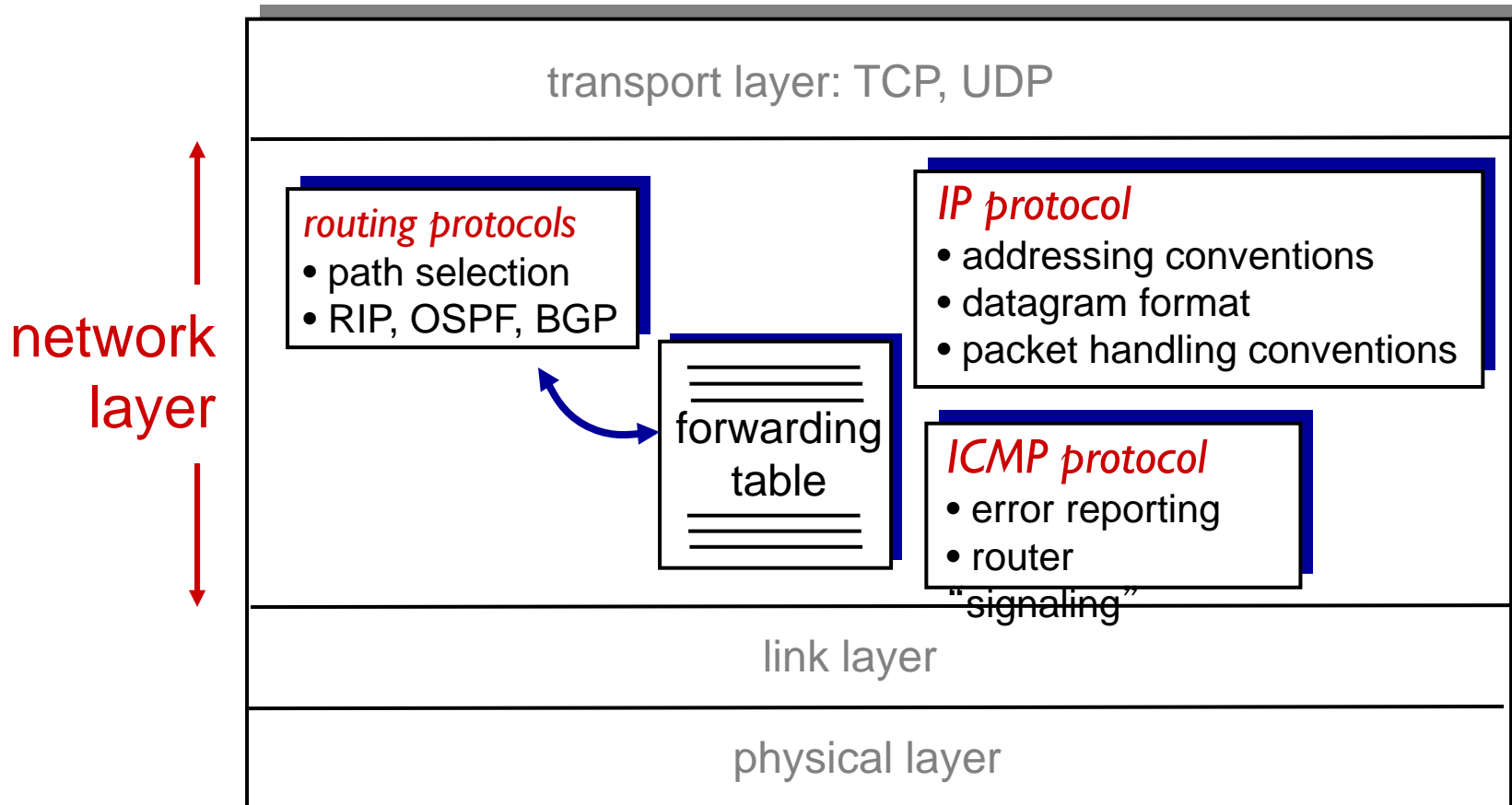   - hierarchical routing
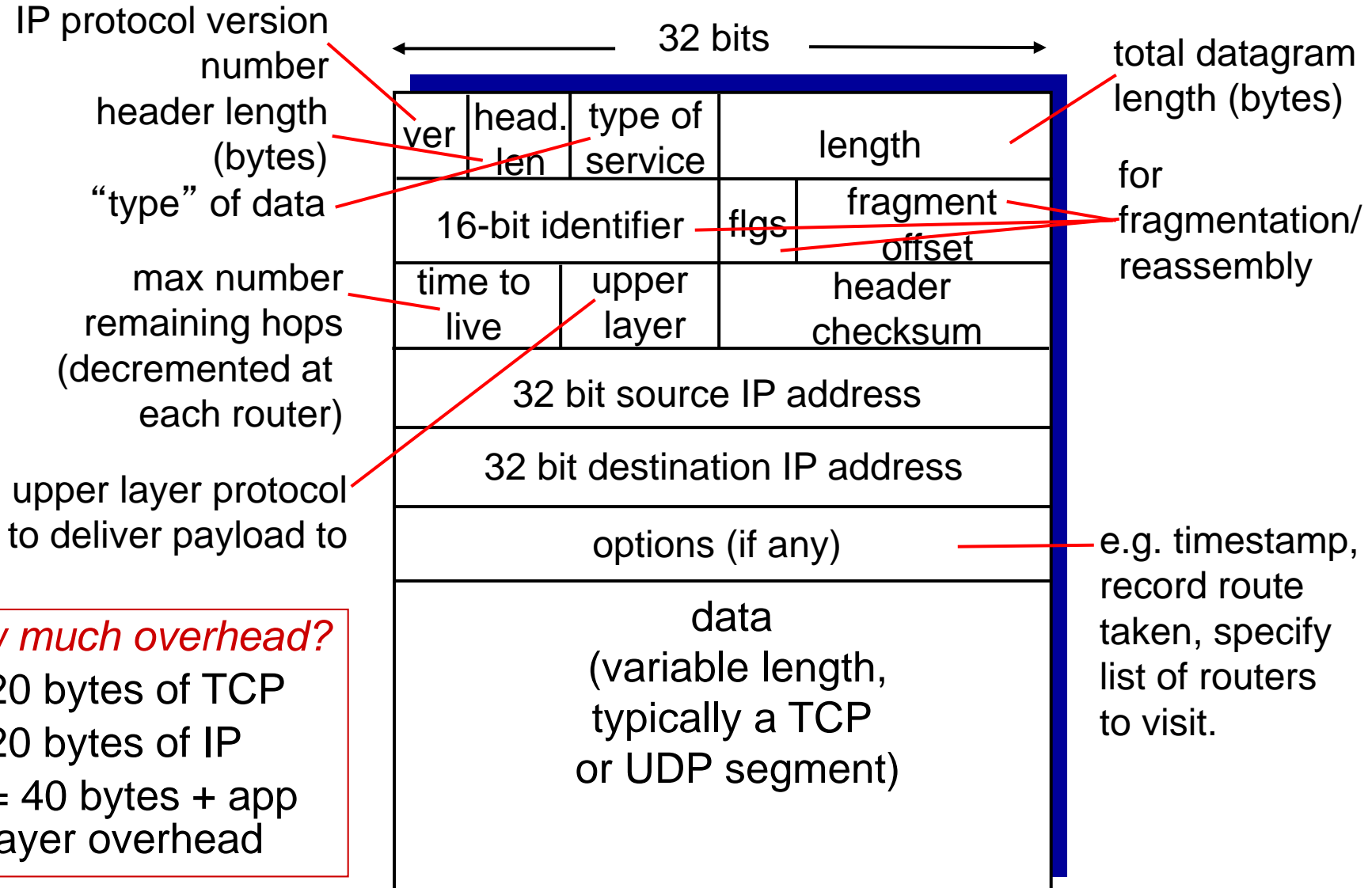
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP

4.7 broadcast and multicast
   routing

The Internet network layer

host, router network layer functions:

network layer (vertical label spanning transport to link layers)

transport layer: TCP, UDP

routing protocols
- path selection
- RIP, OSPF, BGP

forwarding table

IP protocol
- addressing conventions
- datagram format
- packet handling conventions

ICMP protocol
- error reporting
- router "signaling"

link layer

physical layer

IP datagram format

IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

32 bits

ver	head. len	type of service	length		
16-bit identifier			flgs	fragment offset	
time to live		upper layer	header checksum		
32 bit source IP address					
32 bit destination IP address					
options (if any)					
data (variable length, typically a TCP or UDP segment)					

total datagram length (bytes)

for fragmentation/ reassembly

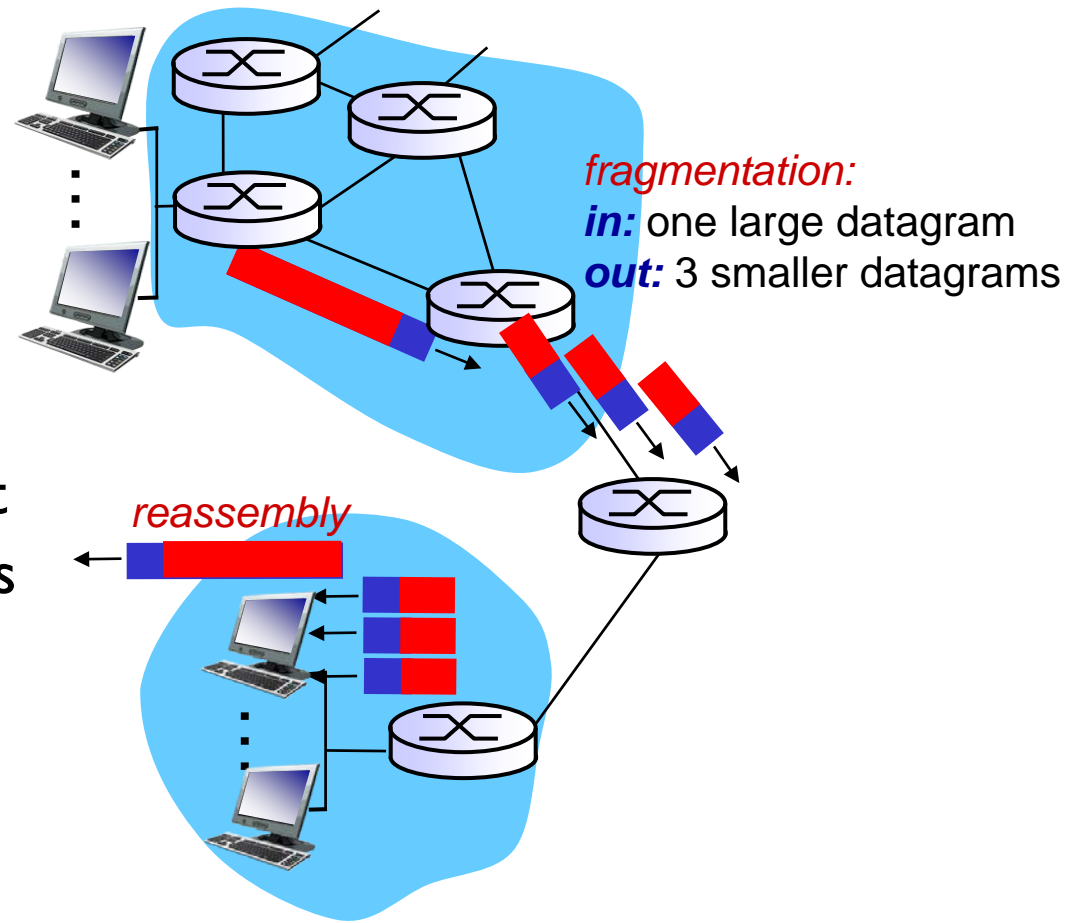e.g. timestamp, record route taken, specify list of routers to visit.

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

fragmentation:
in: one large datagram
out: 3 smaller datagrams

reassembly

IP fragmentation, reassembly

| | length =4000 | ID =x | fragflag =0 | offset =0 | |

example:

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

one large datagram becomes several smaller datagrams

1480 bytes in data field

| | length =1500 | ID =x | fragflag =1 | offset =0 | |

offset = 1480/8

| | length =1500 | ID =x | fragflag =1 | offset =185 | |

| | length =1040 | ID =x | fragflag =0 | offset =370 | |

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
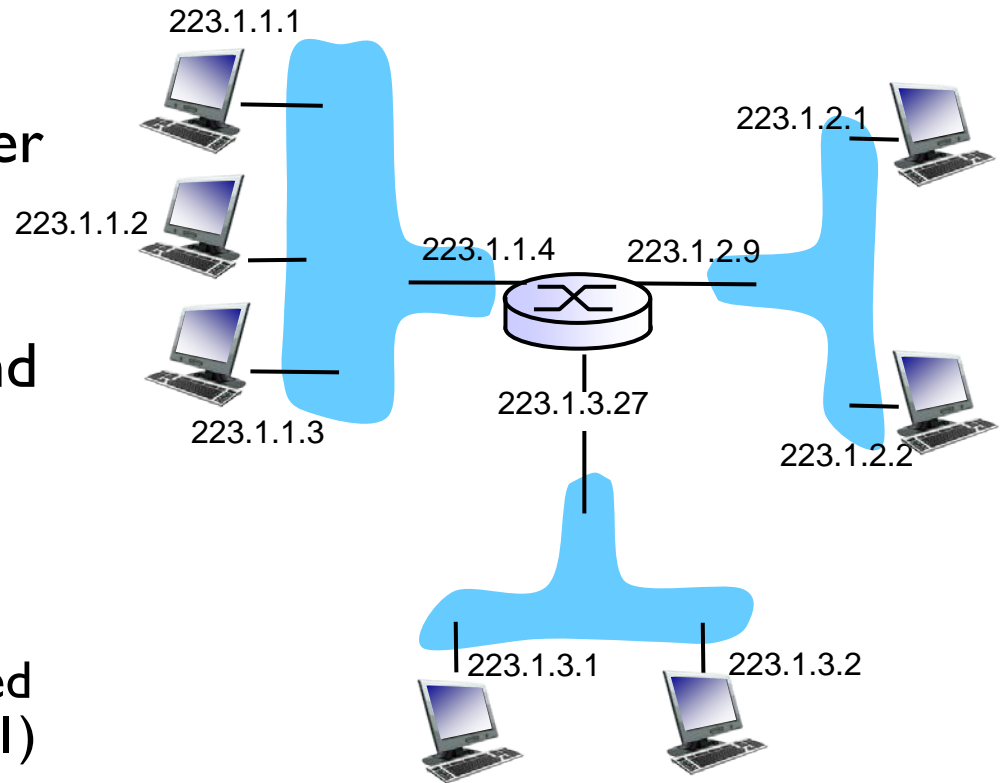- distance vector
- hierarchical routing

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

223.1.1.2

223.1.1.3

223.1.1.4

223.1.2.9

223.1.2.1

223.1.2.2

223.1.3.27

223.1.3.1

223.1.3.2

223.1.1.1 = 11011111 00000001 00000001 00000001

223   1   1   1

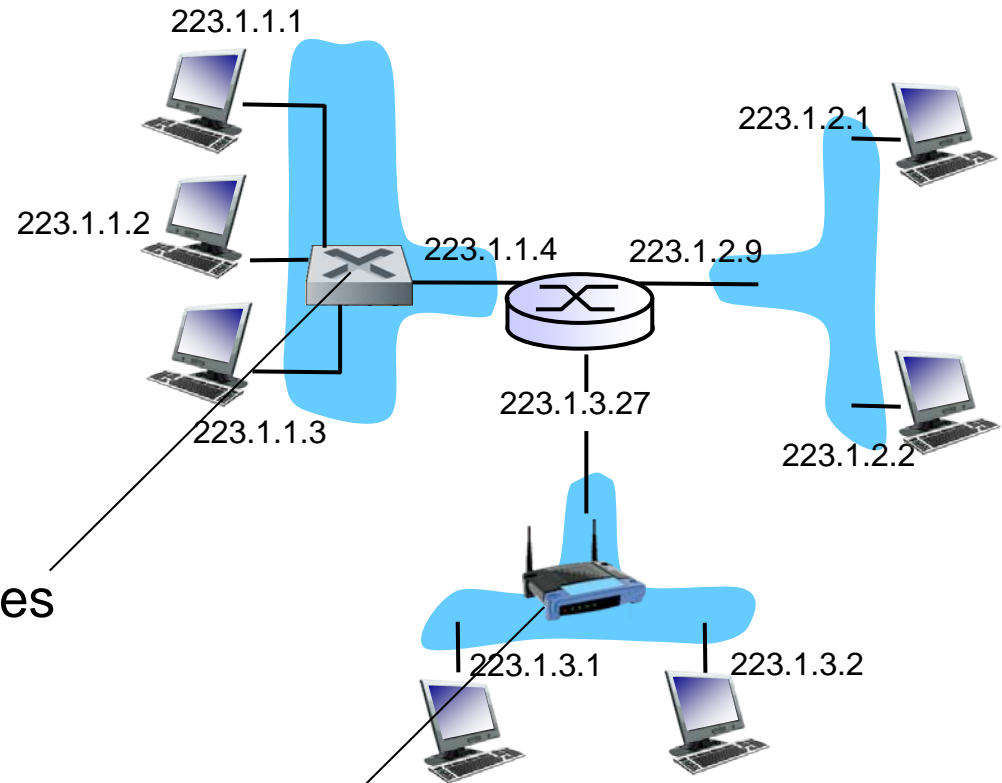# IP addressing: introduction

Q: how are interfaces actually connected?

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

223.1.1.1

223.1.2.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.3.27

223.1.1.3

223.1.2.2

A: wired Ethernet interfaces connected by Ethernet switches

223.1.3.1   223.1.3.2

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

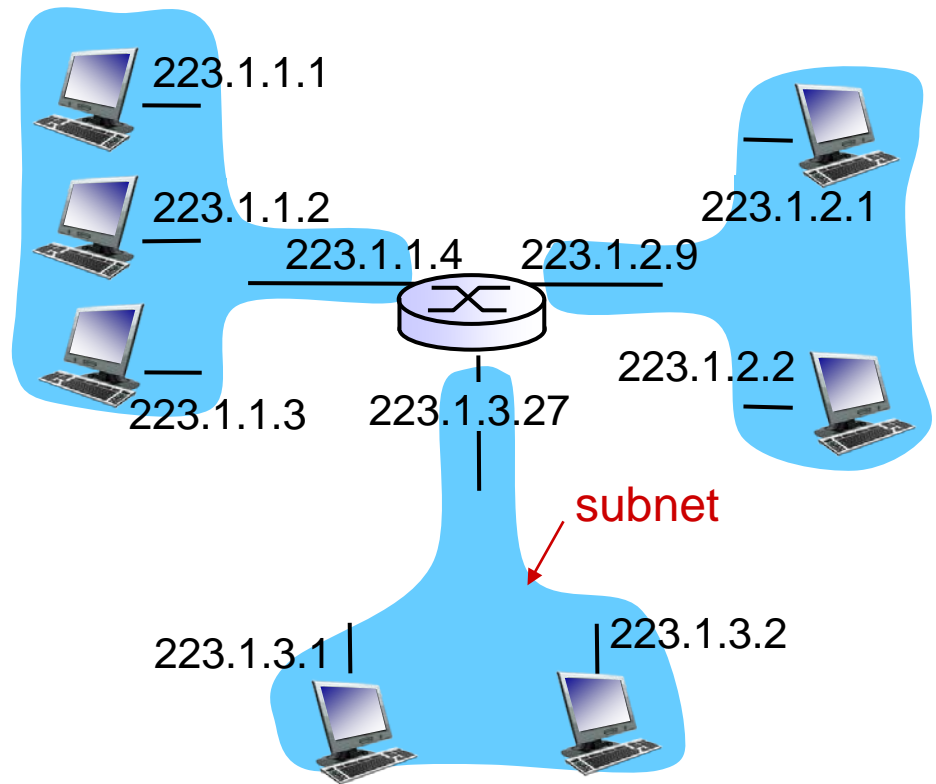*A:* wireless WiFi interfaces connected by WiFi base station

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
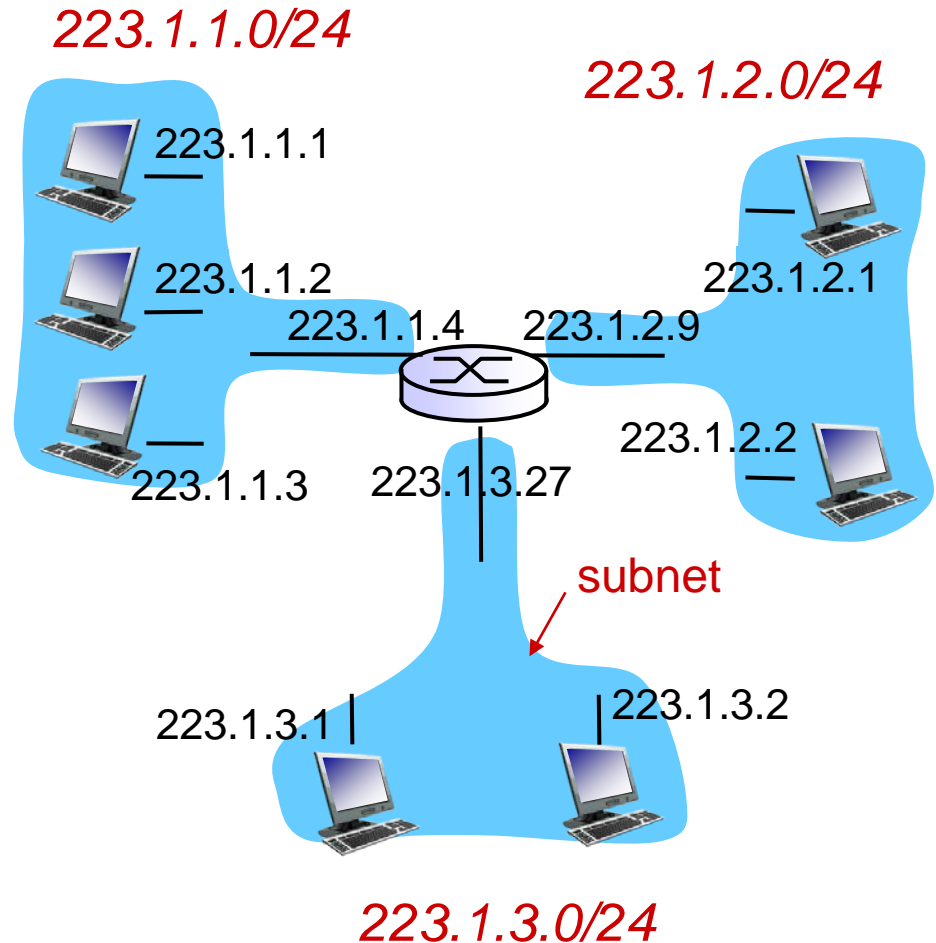 * can physically reach each other *without intervening router*

223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.2.1

223.1.2.2

223.1.1.3   223.1.3.27

subnet

223.1.3.1   223.1.3.2

network consisting of 3 subnets

Subnets

recipe
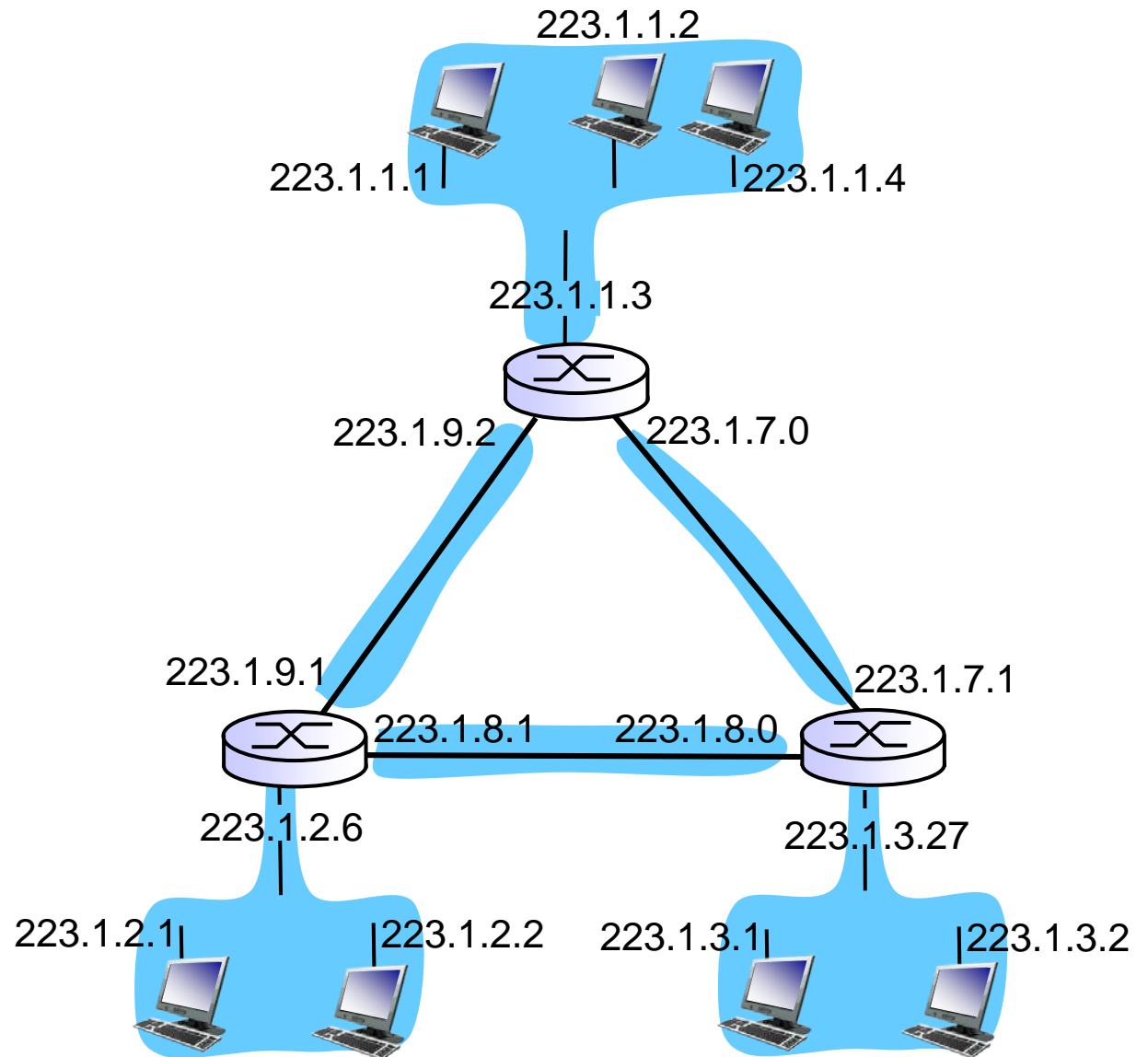
❖ to determine the subnets, detach each interface from its host or router, creating islands of isolated networks

❖ each isolated network is called a *subnet*

223.1.1.0/24

223.1.2.0/24

223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.2.1

223.1.2.2

223.1.1.3   223.1.3.27

subnet

223.1.3.1   223.1.3.2

223.1.3.0/24

subnet mask: /24

Subnets

how many?

223.1.1.2

223.1.1.1

223.1.1.4

223.1.1.3

223.1.9.2   223.1.7.0

223.1.9.1   223.1.7.1

223.1.8.1   223.1.8.0

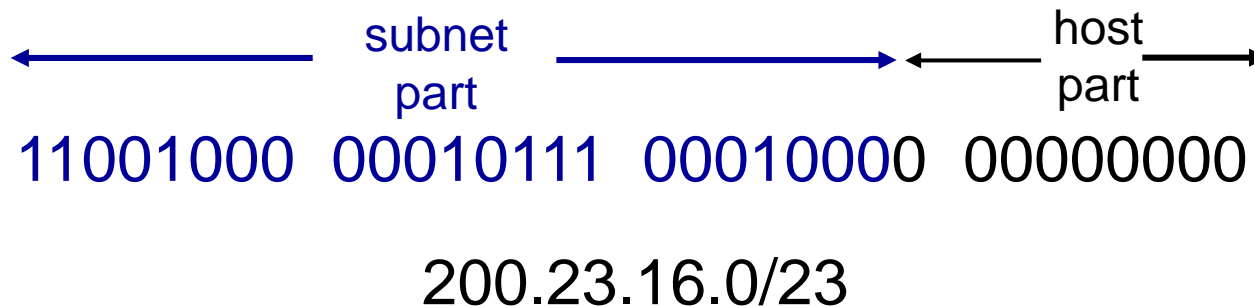223.1.2.6   223.1.3.27

223.1.2.1   223.1.2.2   223.1.3.1   223.1.3.2

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

```
      ←————————— subnet ————————→    ←— host —→
                  part                   part
   11001000  00010111  00010000  00000000
              200.23.16.0/23
```

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"

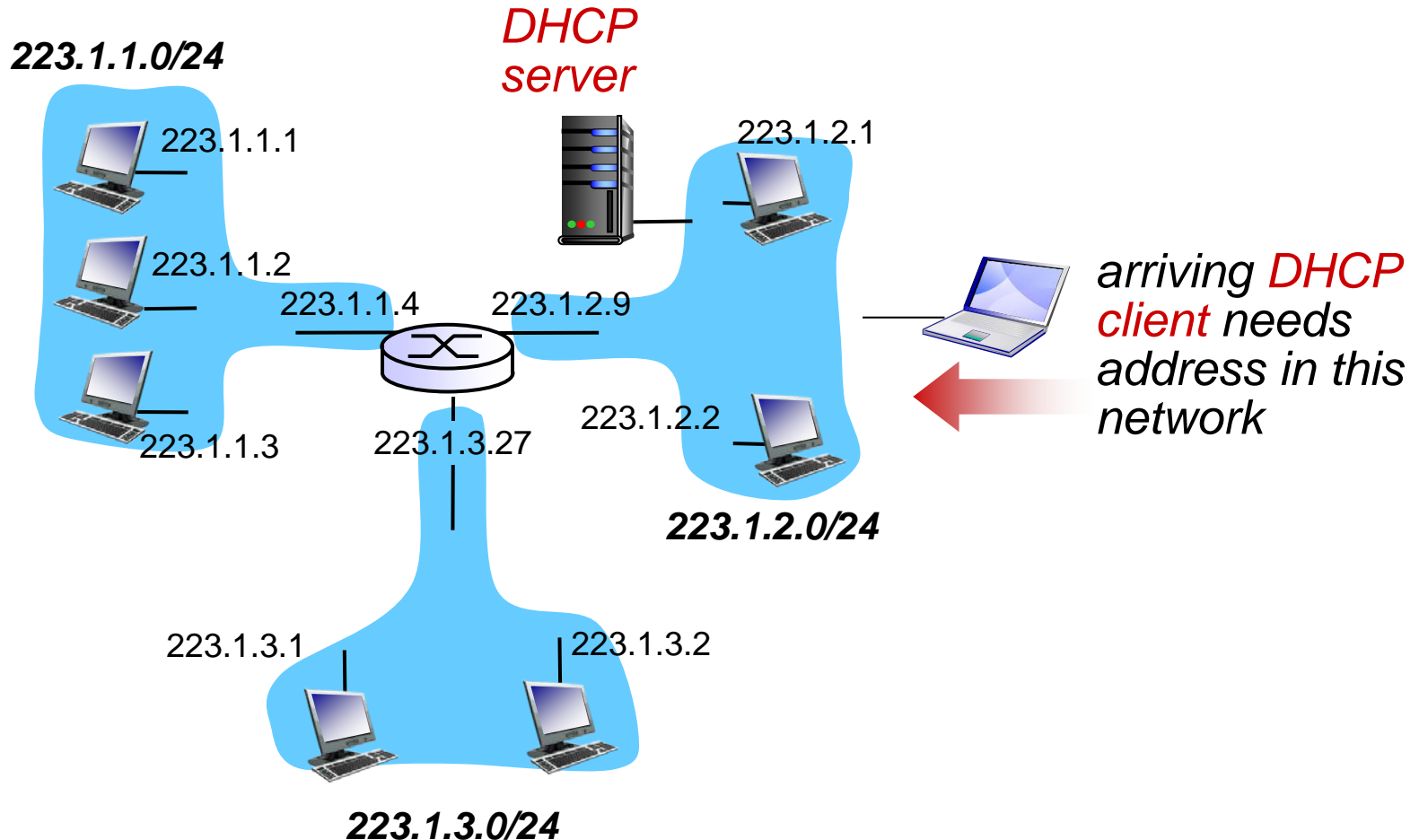# DHCP: Dynamic Host Configuration Protocol

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

DHCP client-server scenario

223.1.1.0/24

DHCP
server

223.1.1.1

223.1.2.1

223.1.1.2

223.1.1.4   223.1.2.9

arriving DHCP
client needs
address in this
network

223.1.1.3   223.1.3.27   223.1.2.2

223.1.2.0/24

223.1.3.1   223.1.3.2

223.1.3.0/24

DHCP client-server scenario

DHCP server: 223.1.2.5

arriving client

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

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

router with DHCP
server built into
router
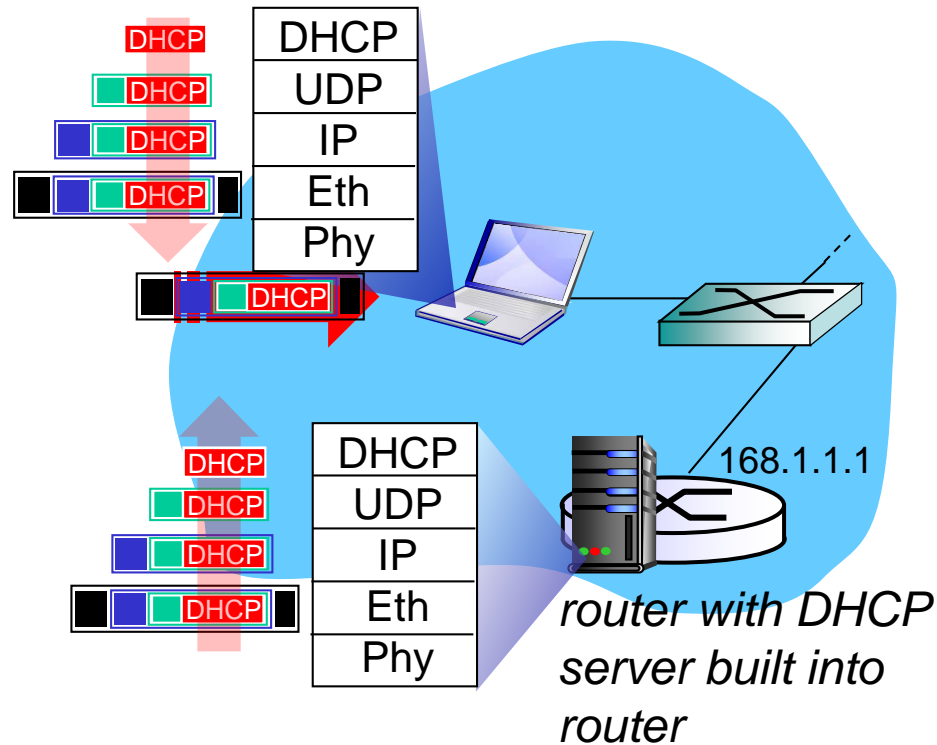
- ❖ connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP

- ❖ 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

router with DHCP
server built into
router
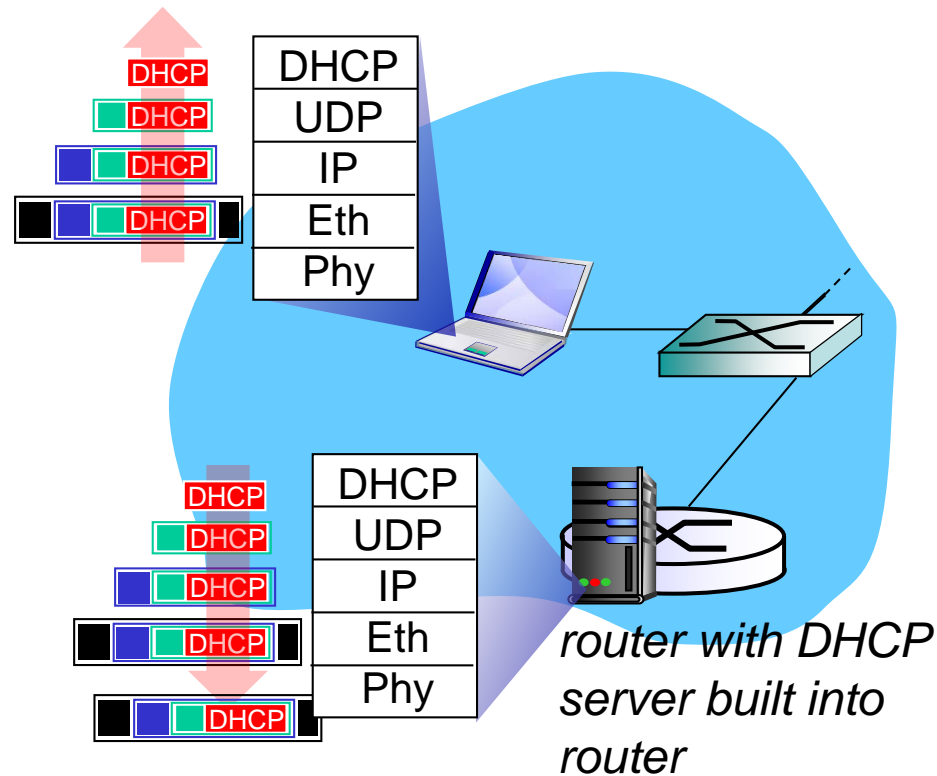
❖ DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

❖ 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 DSN server, IP address of its first-hop router
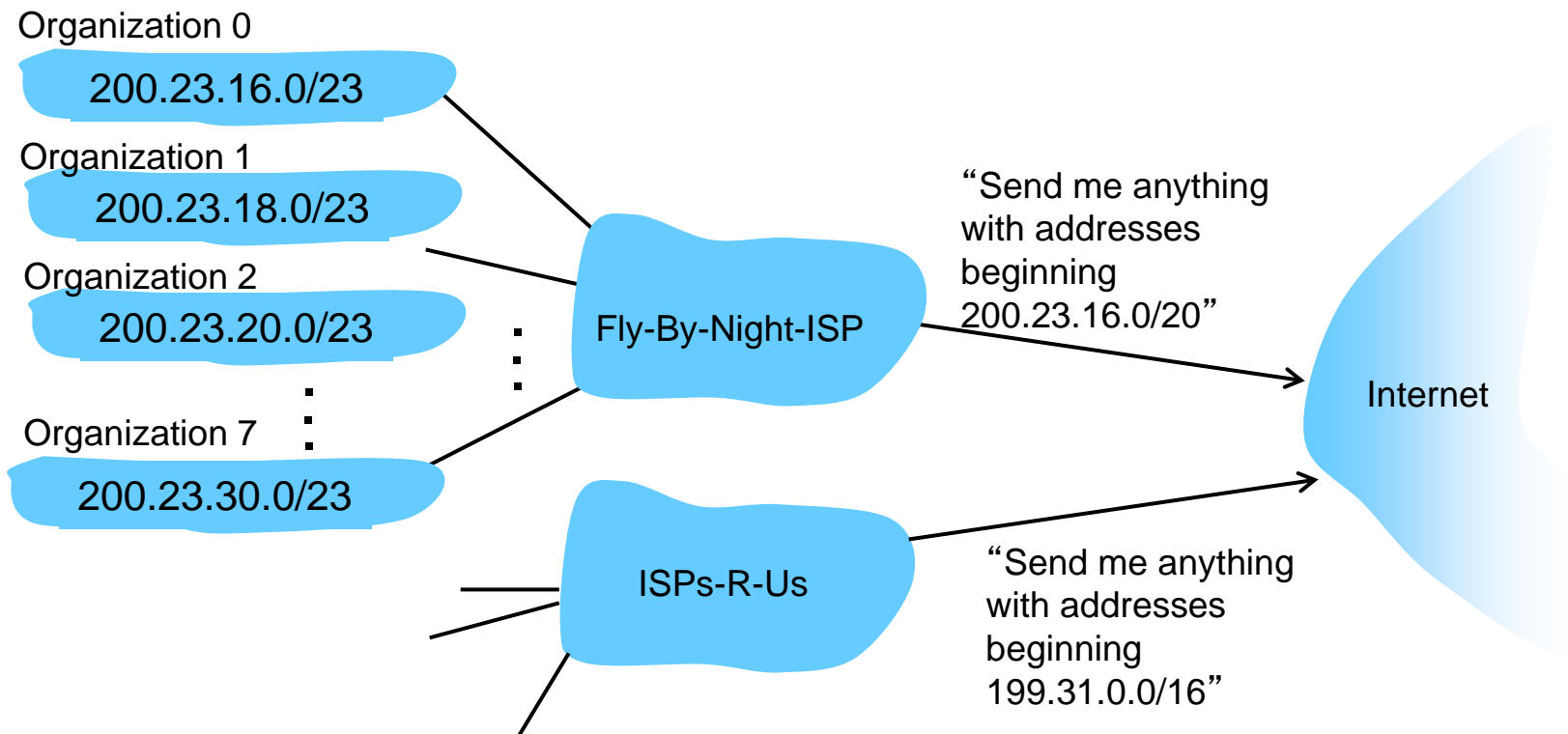
IP addresses: how to get one?

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

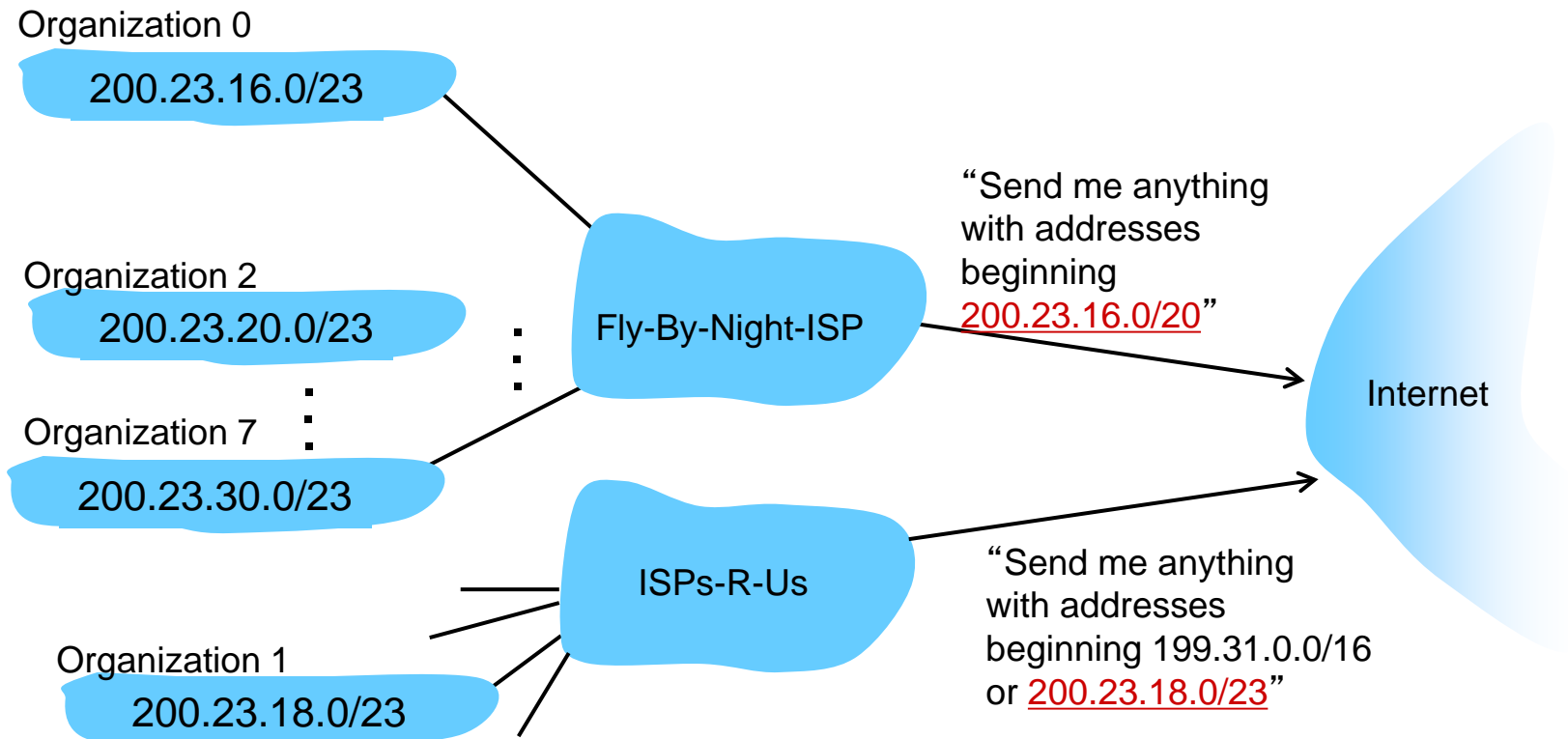# Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:

Organization 0
200.23.16.0/23

Organization 1
200.23.18.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Fly-By-Night-ISP

ISPs-R-Us

"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

Internet

Hierarchical addressing: more specific routes

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

Organization 0
200.23.16.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Organization 1
200.23.18.0/23

Fly-By-Night-ISP

ISPs-R-Us

"Send me anything
with addresses
beginning
200.23.16.0/20"

"Send me anything
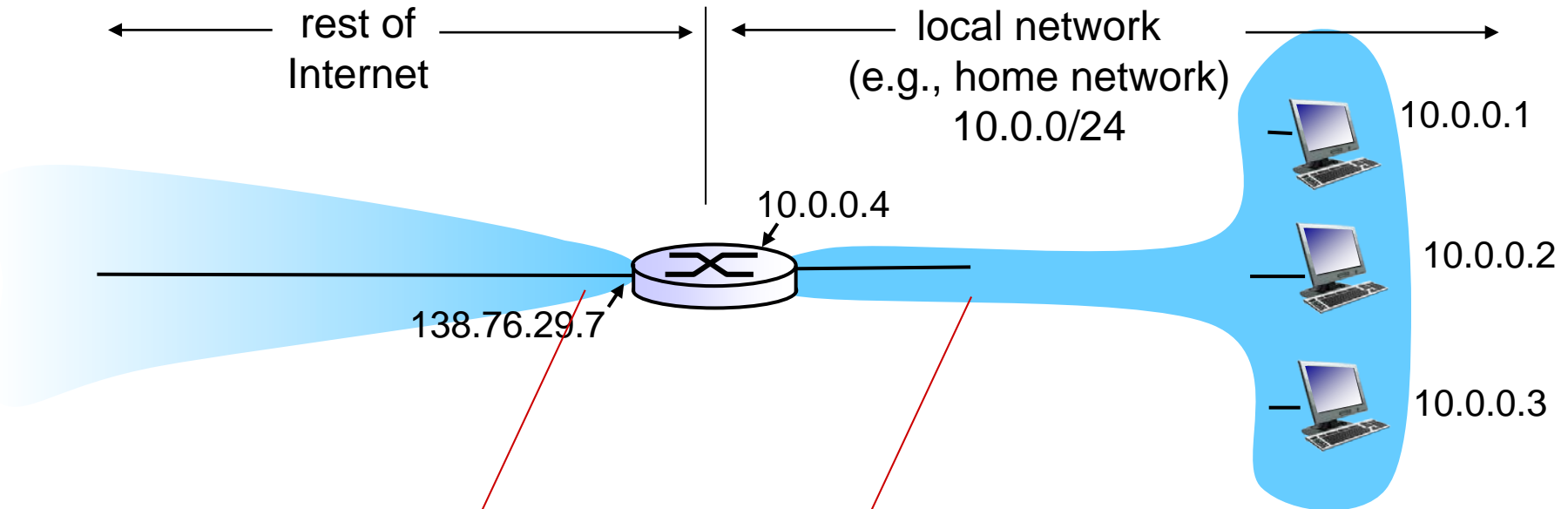with addresses
beginning 199.31.0.0/16
or 200.23.18.0/23"

Internet

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

rest of Internet ← →   ← local network (e.g., home network) 10.0.0/24 →

10.0.0.4

138.76.29.7

10.0.0.1

10.0.0.2

10.0.0.3

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

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

NAT: network address translation

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

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

NAT: network address translation
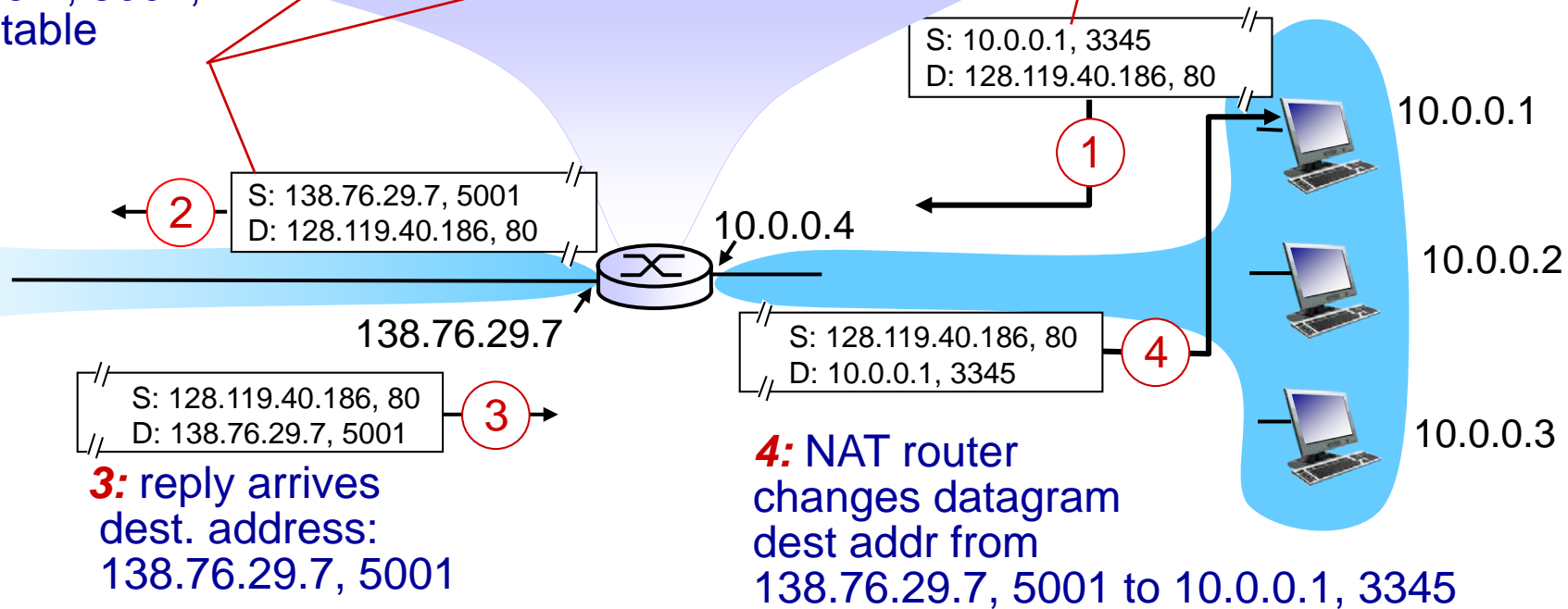
implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) 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

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

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

S: 10.0.0.1, 3345
D: 128.119.40.186, 80

①

S: 138.76.29.7, 5001
D: 128.119.40.186, 80

②

10.0.0.4

138.76.29.7

S: 128.119.40.186, 80
D: 10.0.0.1, 3345

④

10.0.0.1

10.0.0.2

10.0.0.3

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

③

3: reply arrives dest. address: 138.76.29.7, 5001

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

Next class

- ❖ Please read Chapter 4.4-4.5 of your textbook BEFORE Class