

# CMPE 150/L : Introduction to Computer Networks

Chen Qian

Computer Engineering  
UCSC Baskin Engineering

Lecture 12

# No Lecture next Tuesday

- ❑ No Lecture next Tuesday 2/28
- ❑ Instructor gone for presentation on a conference.
- ❑ Labs are still available

# 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

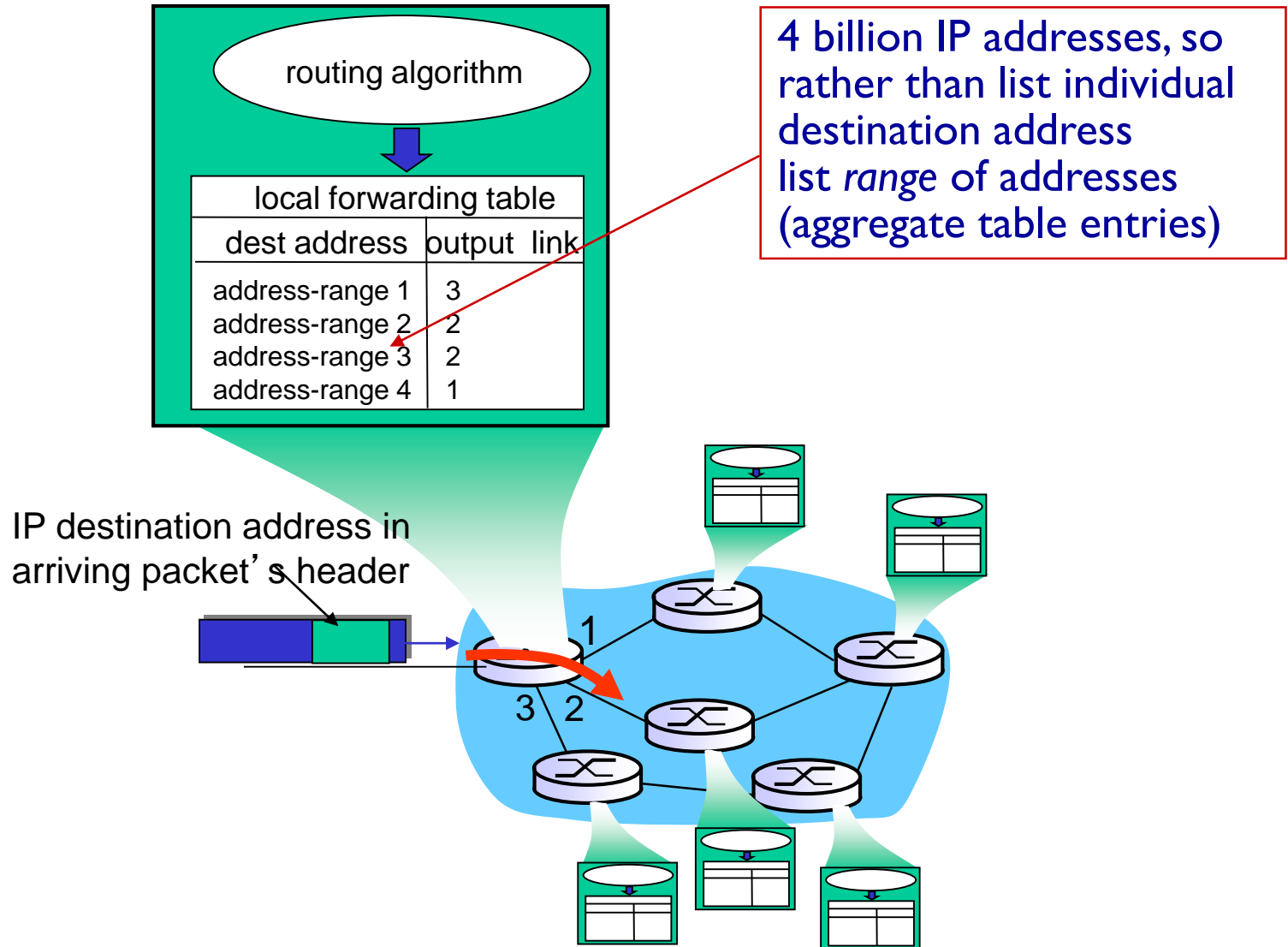
- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Datagram forwarding table



# 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

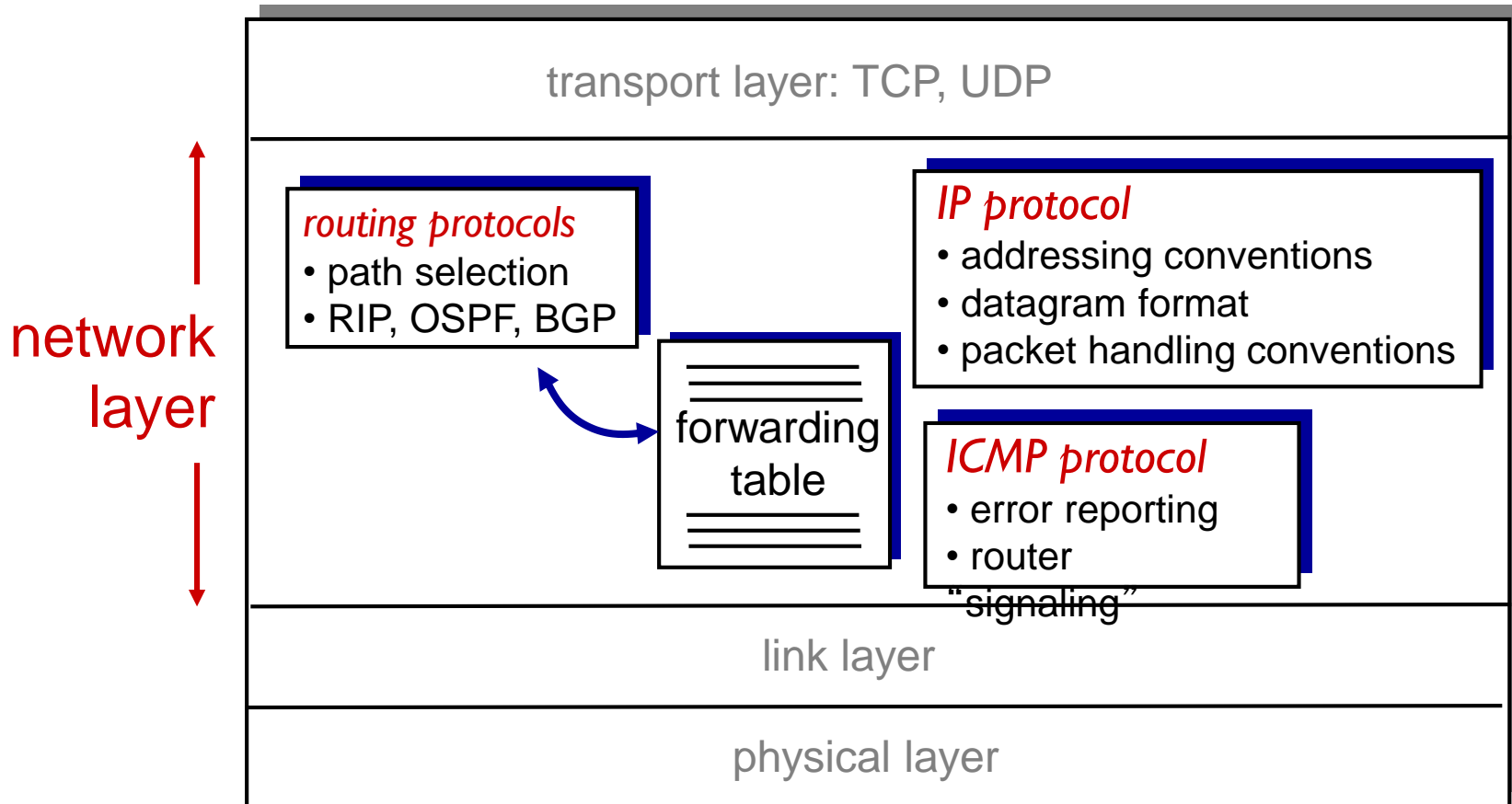
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

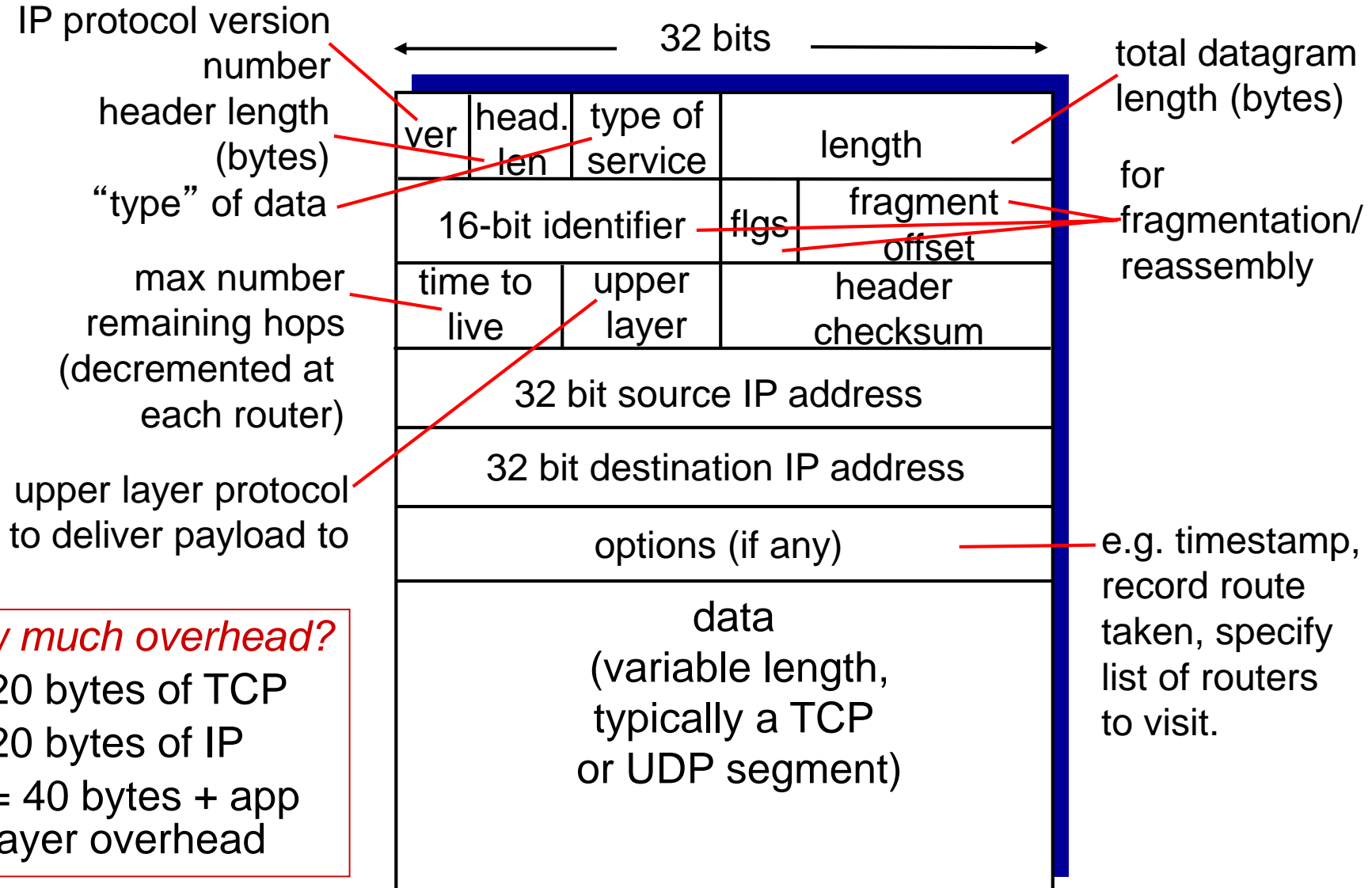


# The Internet network layer

host, router network layer functions:



# IP datagram format

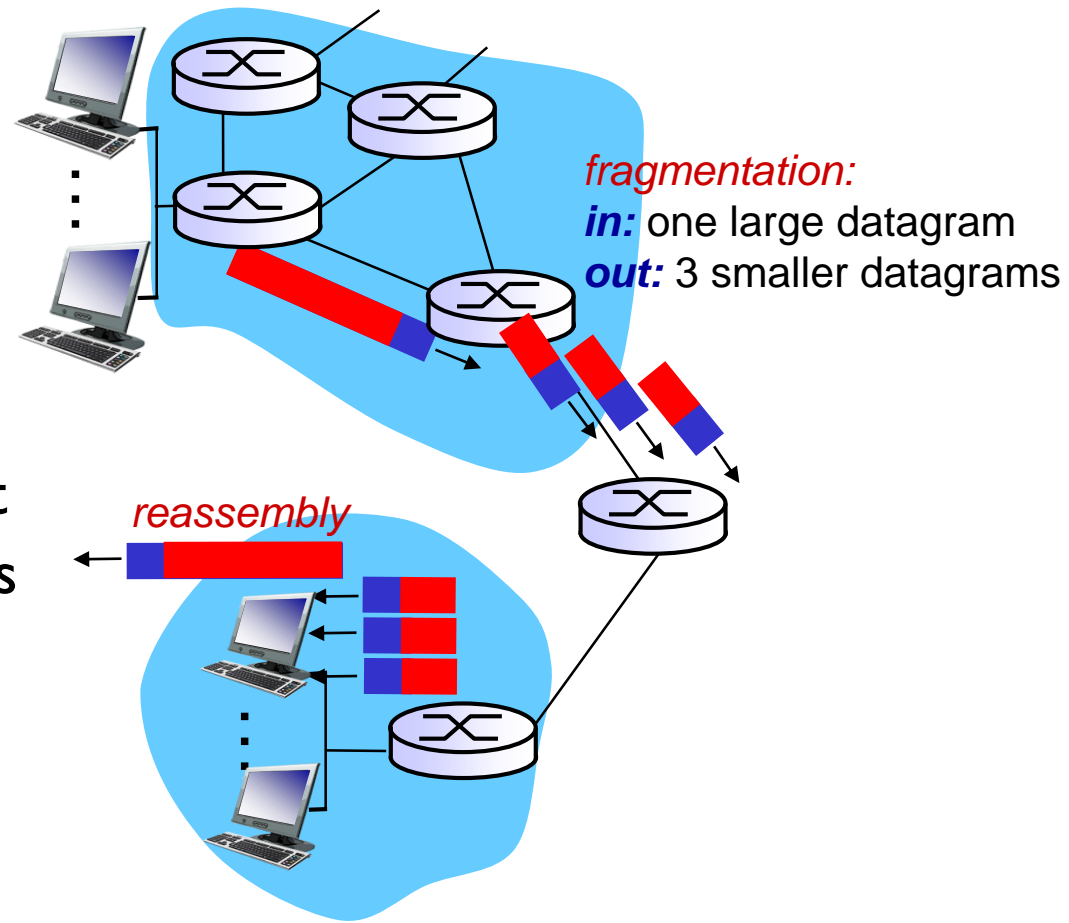


## how much overhead?

- ❖ 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

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

*one large datagram becomes several smaller datagrams*

1480 bytes in data field

offset =  
1480/8

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

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

	length	ID	fragflag	offset	
	=1040	=x	=0	=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

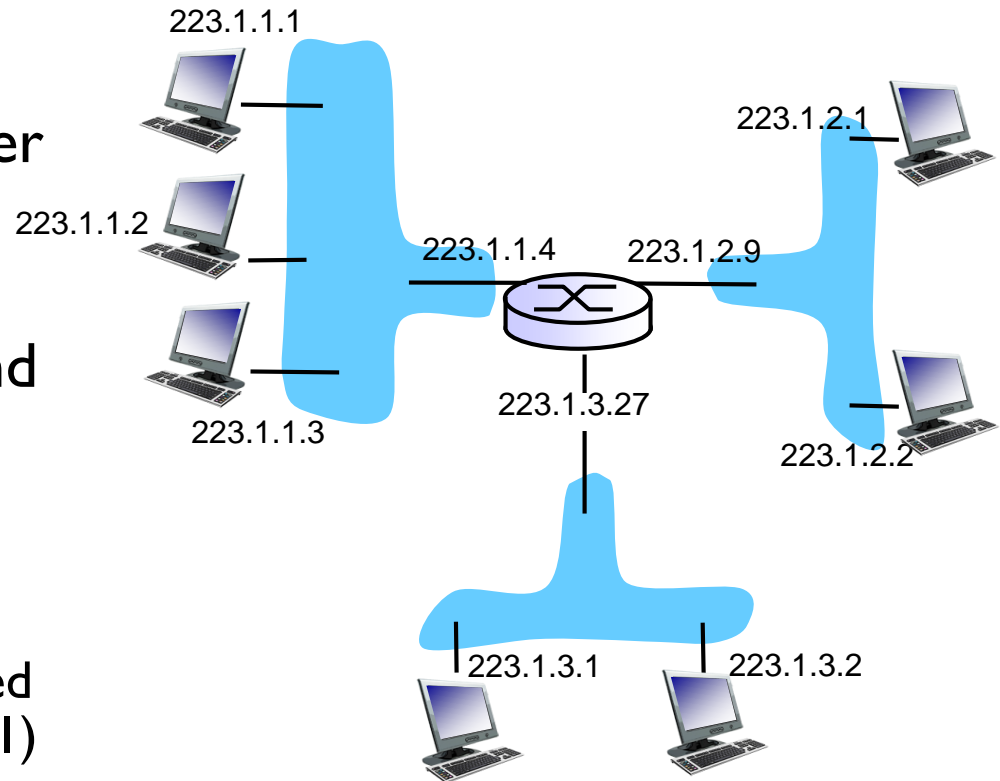
4.6 routing in the Internet

- 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**



$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{0000001}_{1} \underbrace{0000001}_{1} \underbrace{0000001}_{1}$$

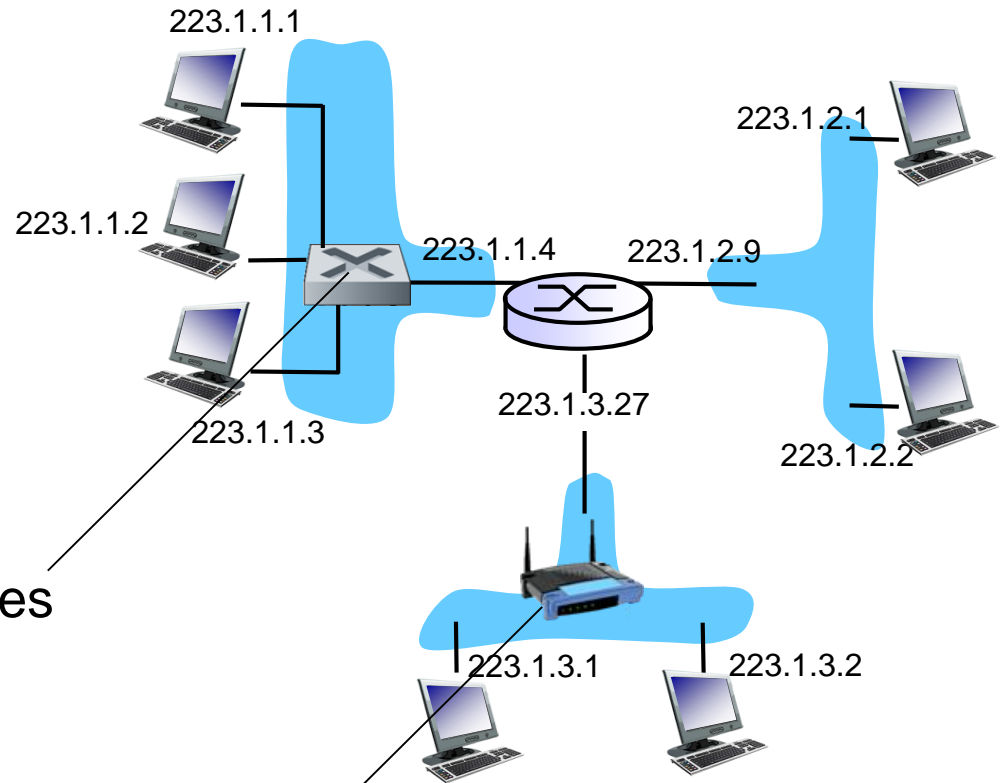
# IP addressing: introduction

**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

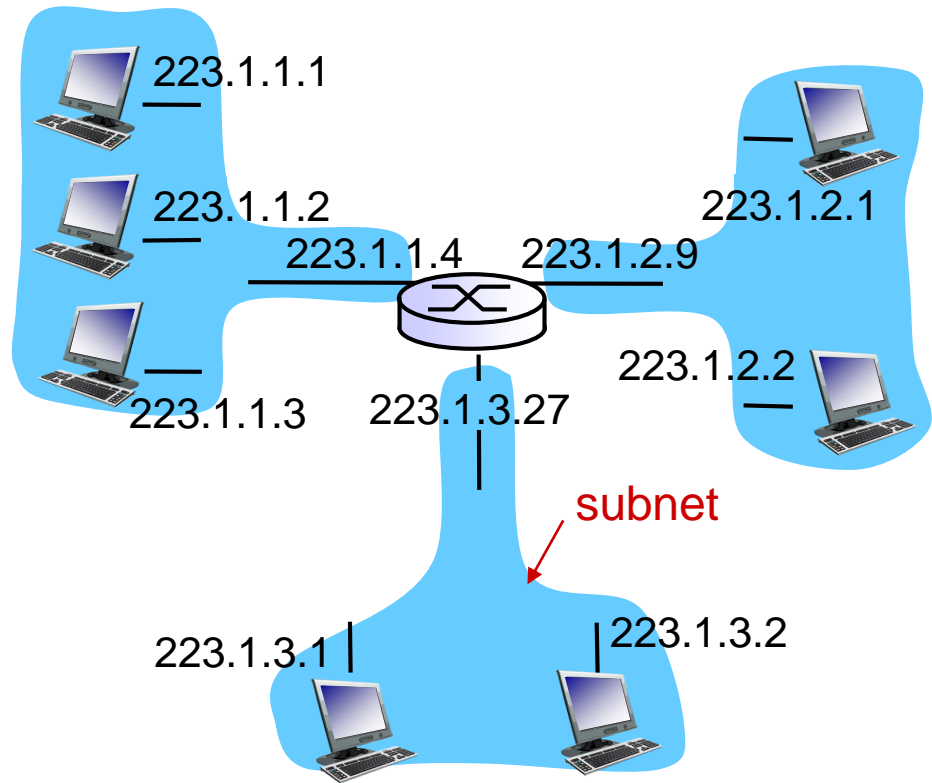
# Subnets

## ❖ 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*



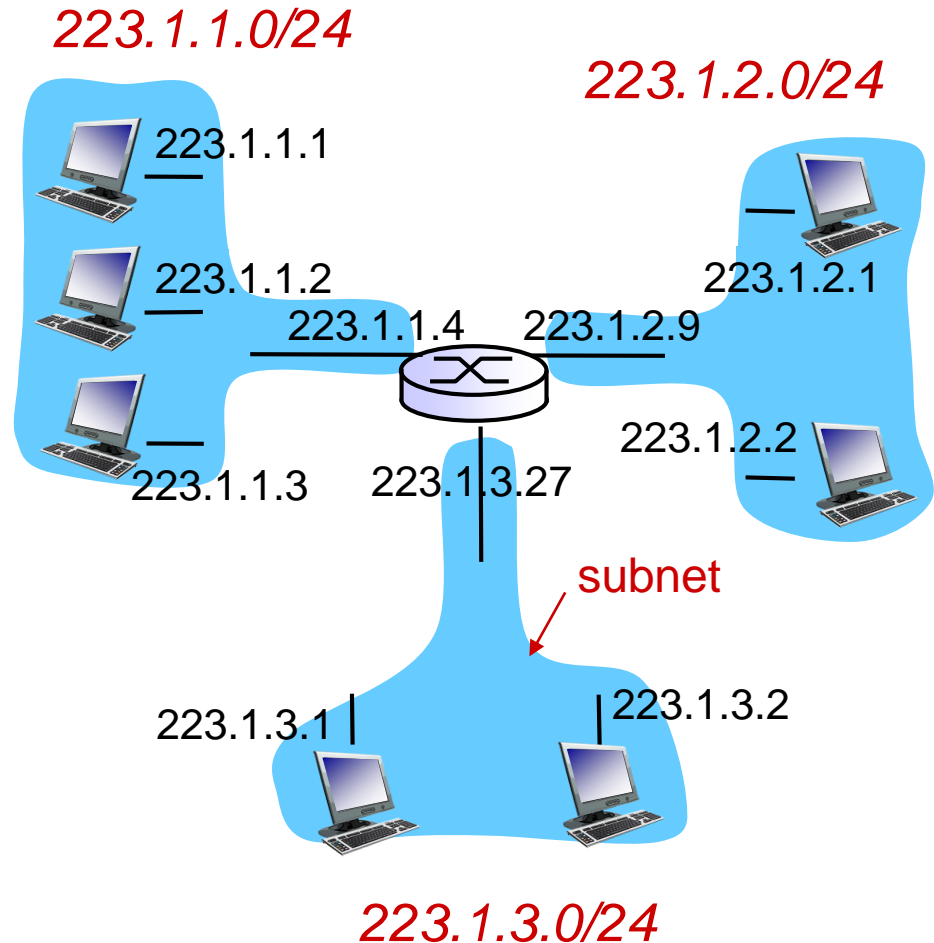
network consisting of 3 subnets



# Subnets

## *recipe*

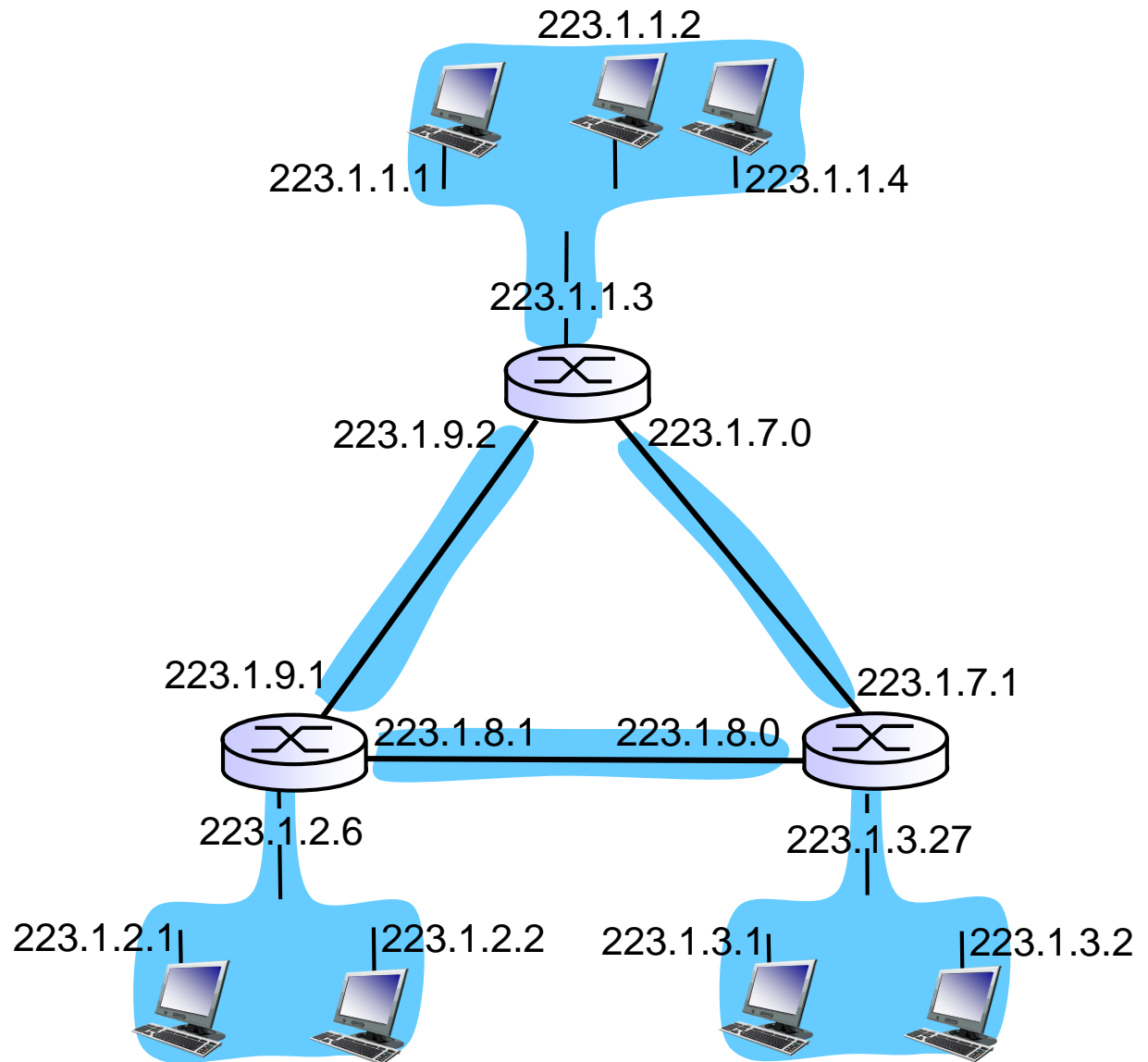
- ❖ to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- ❖ each isolated network is called a *subnet*



subnet mask: /24

# Subnets

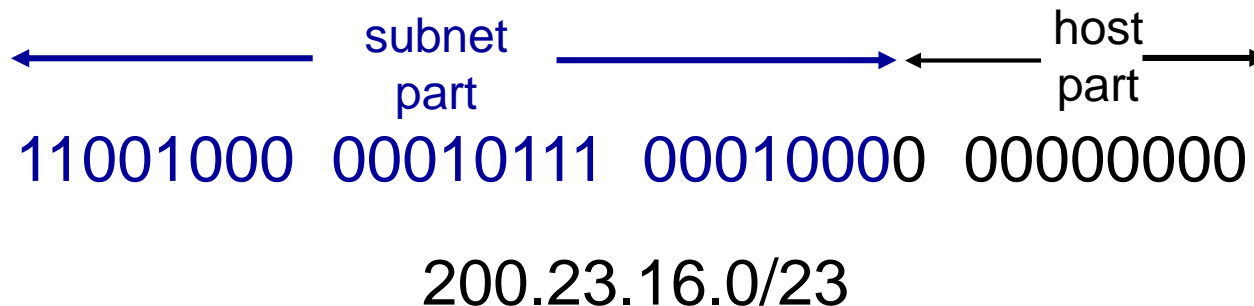
how many?



# 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



# 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

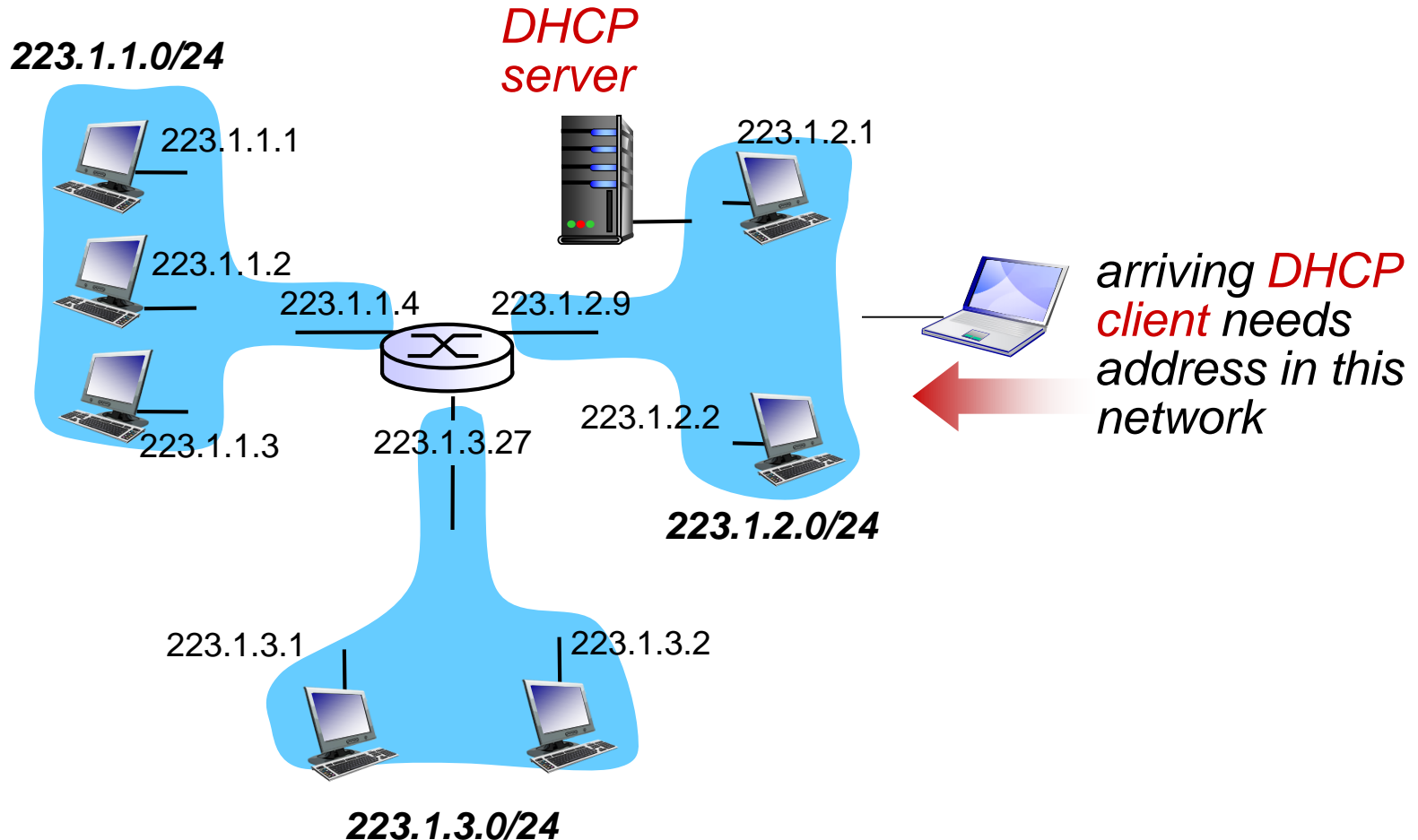
*goal:* allow host to *dynamically* obtain its IP address from network server when it joins network

- 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



# DHCP client-server scenario

DHCP server: 223.1.2.5



**DHCP discover**

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

arriving  
client



**DHCP offer**

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

**DHCP request**

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

**DHCP ACK**

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

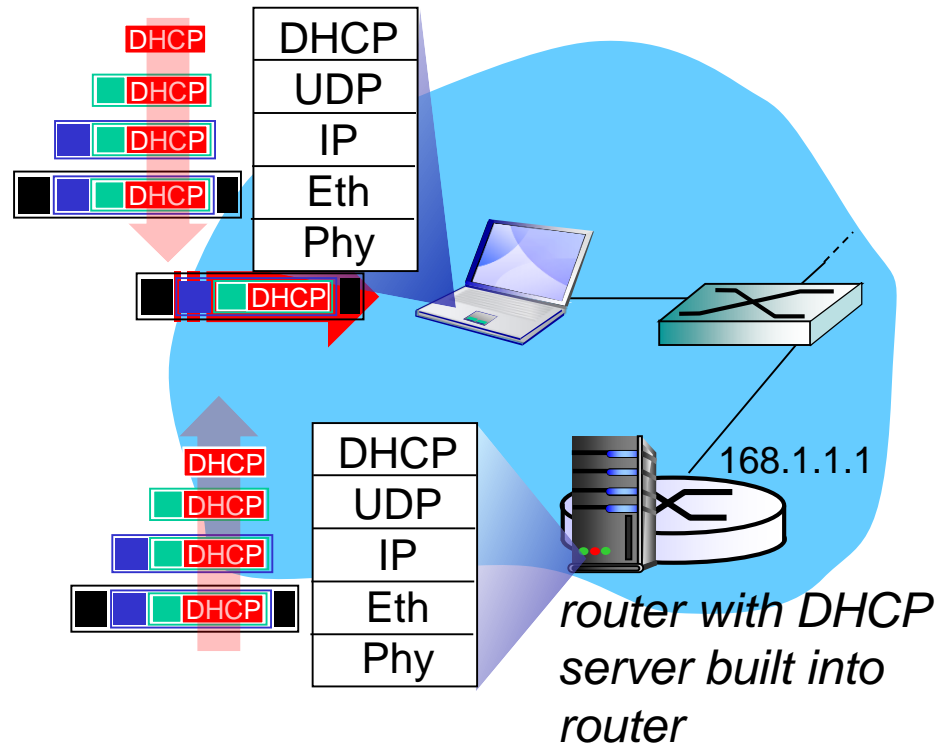
# DHCP: more than IP addresses

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)

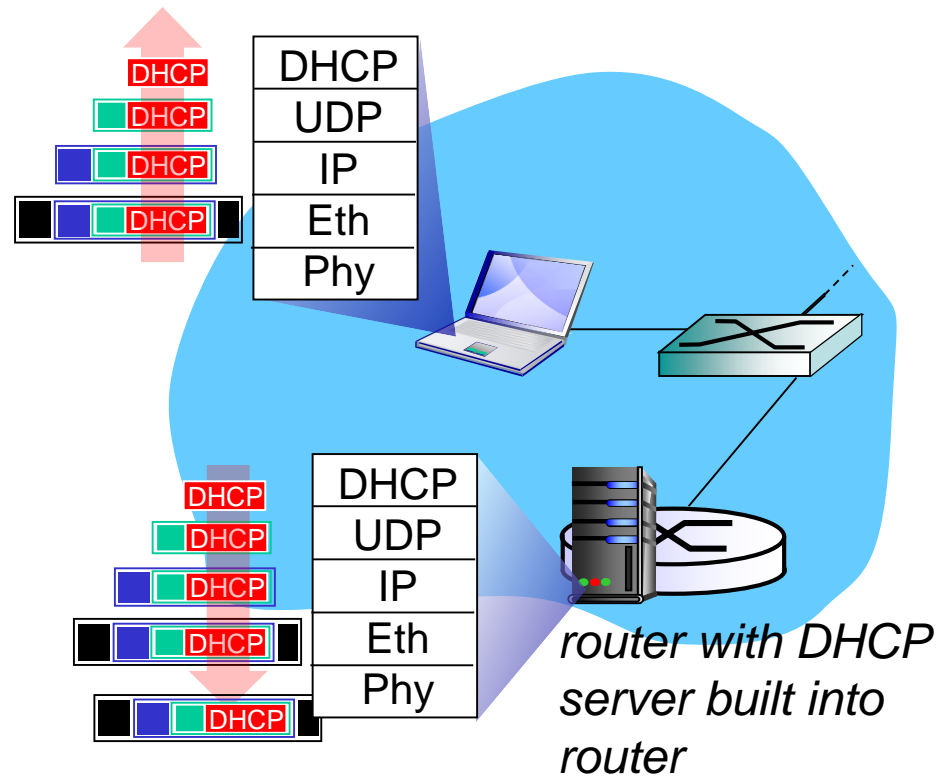


# DHCP: example



- ❖ 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: FFFFFFFFFF) 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

# IP addresses: how to get one?

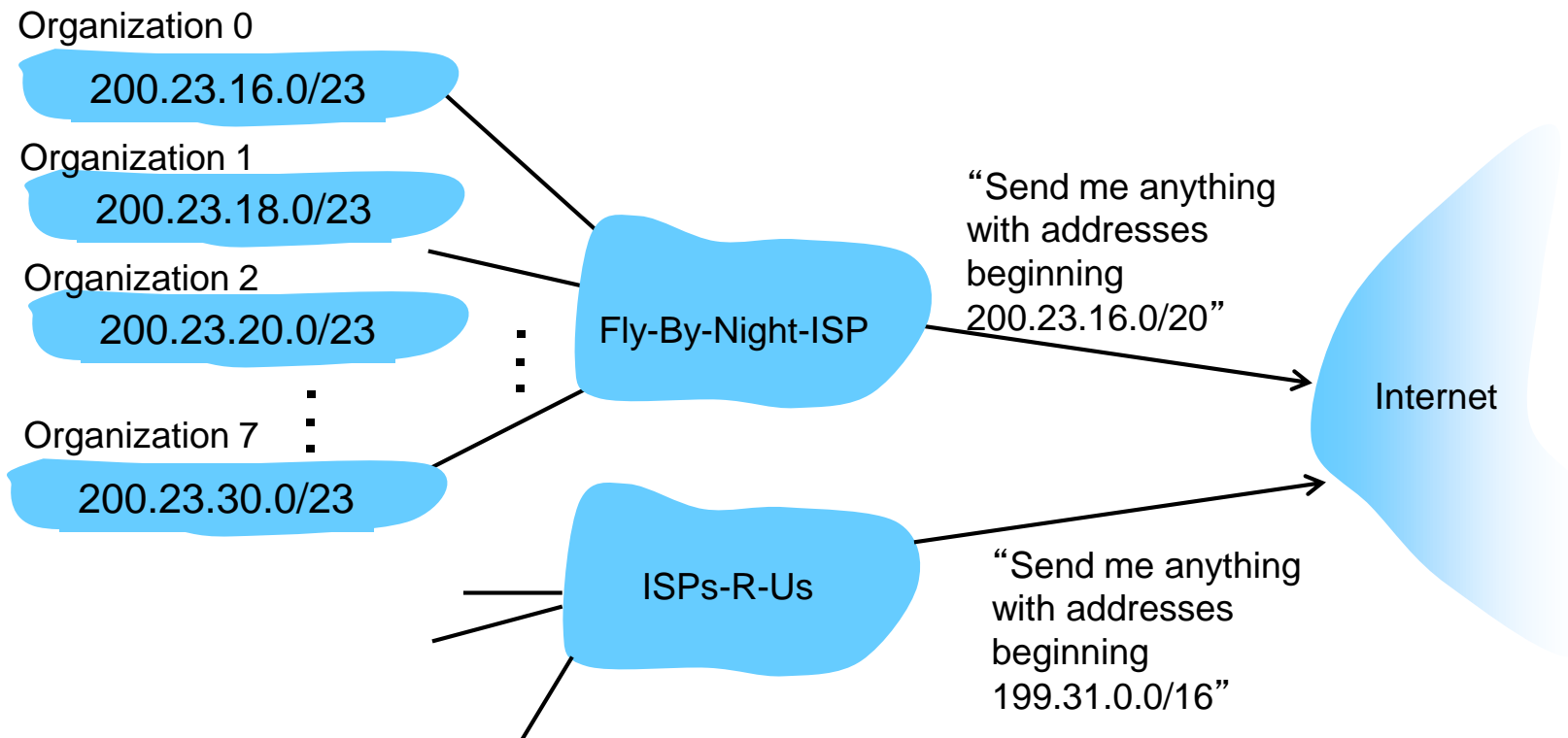
**Q:** how does *network* get subnet part of IP addr?

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

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

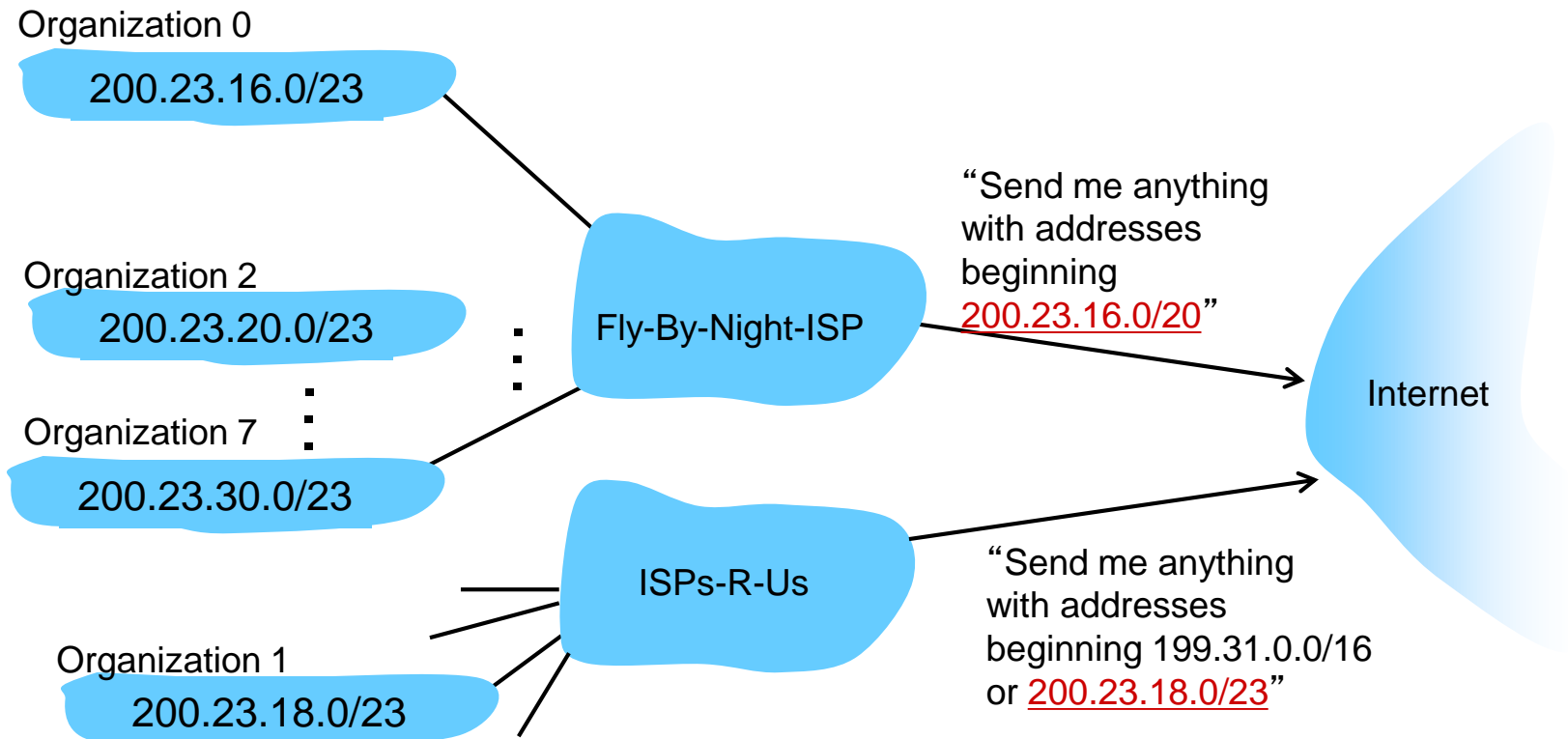
# Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



# Hierarchical addressing: more specific routes

ISPs-R-U's has a more specific route to Organization 1



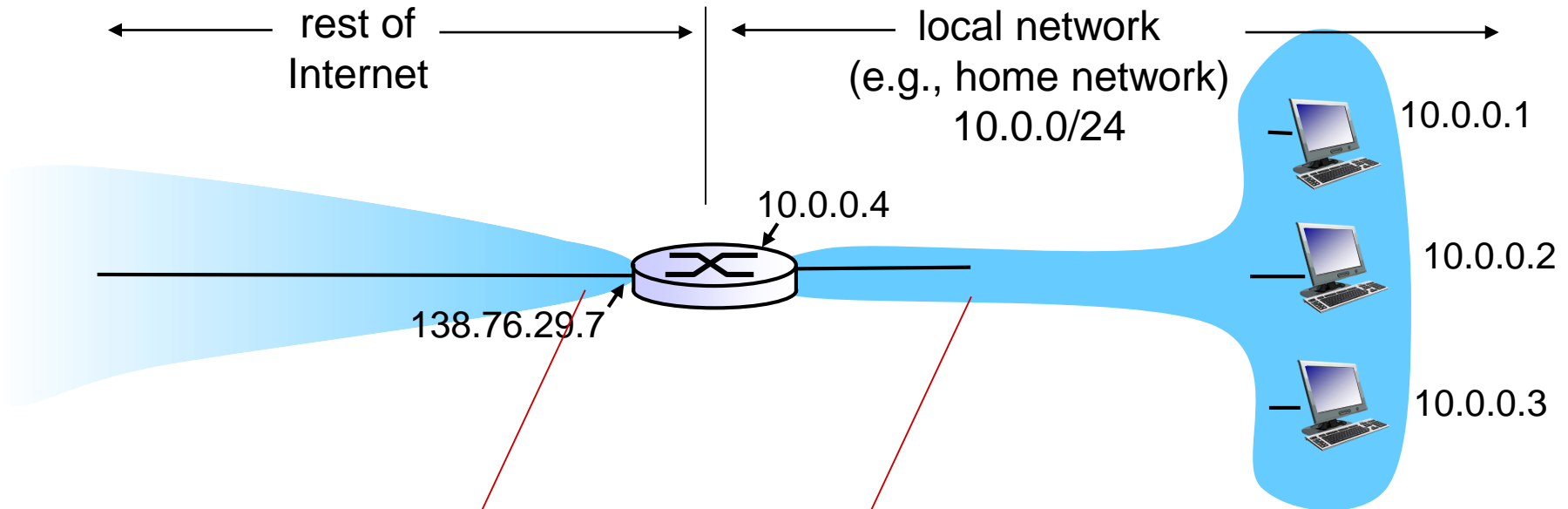
# IP addressing: the last word...

**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

*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:

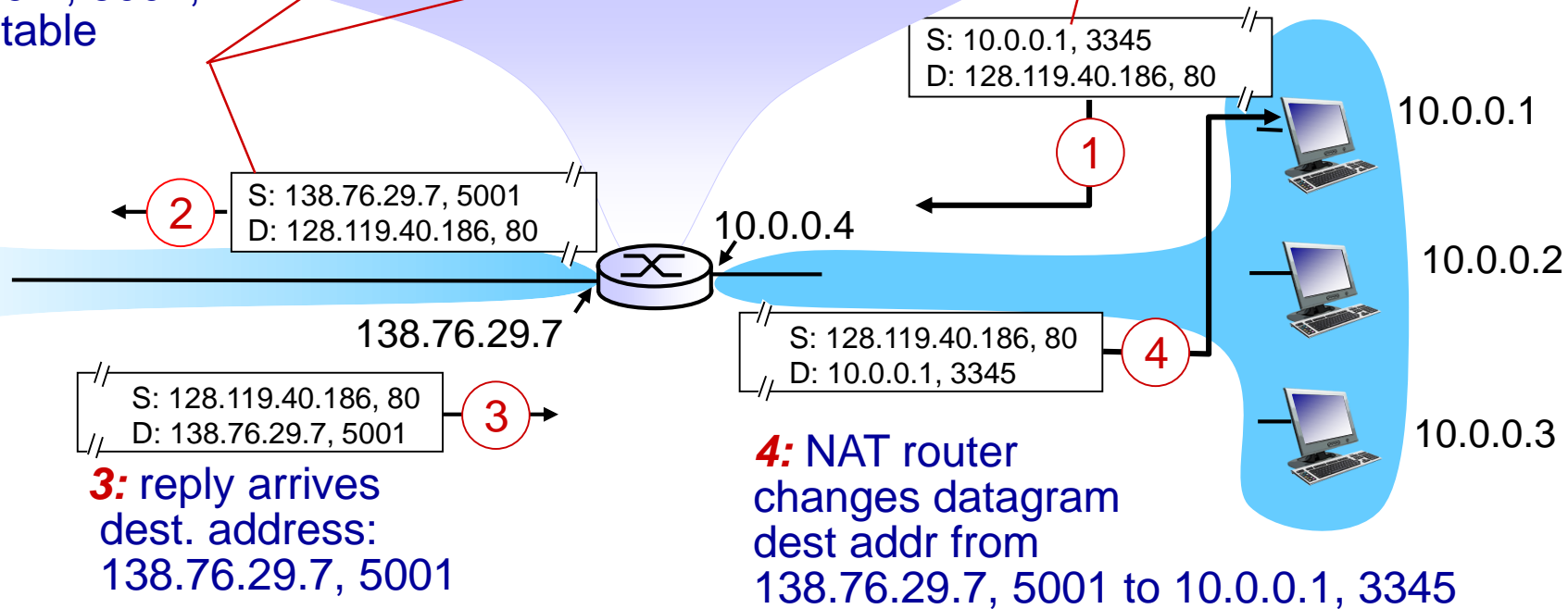
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)  
... remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP 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

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

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

**1:** host 10.0.0.1 sends datagram to 128.119.40.186, 80



**3:** reply arrives  
dest. address:  
138.76.29.7, 5001

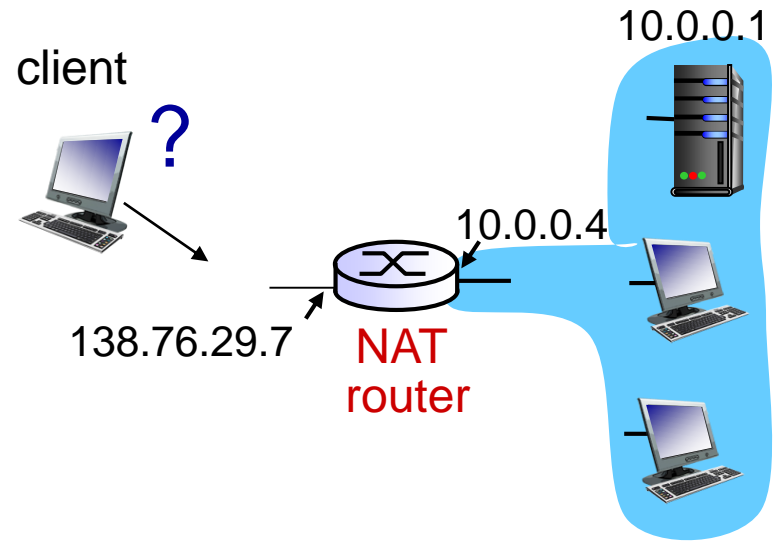
**4:** NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

# NAT: network address translation

- ❖ 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- ❖ NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

# 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., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

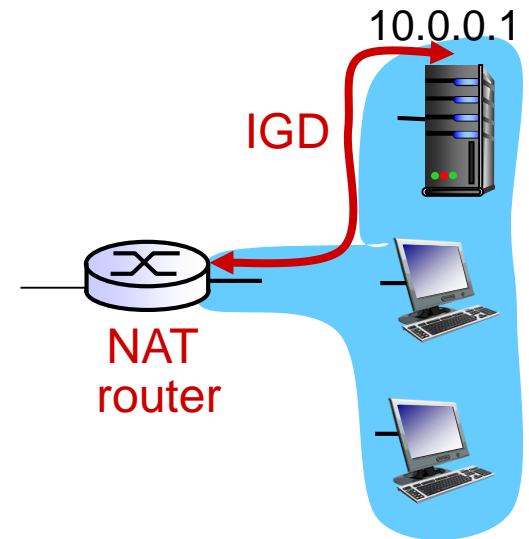


# NAT traversal problem

❖ **solution 2:** Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:

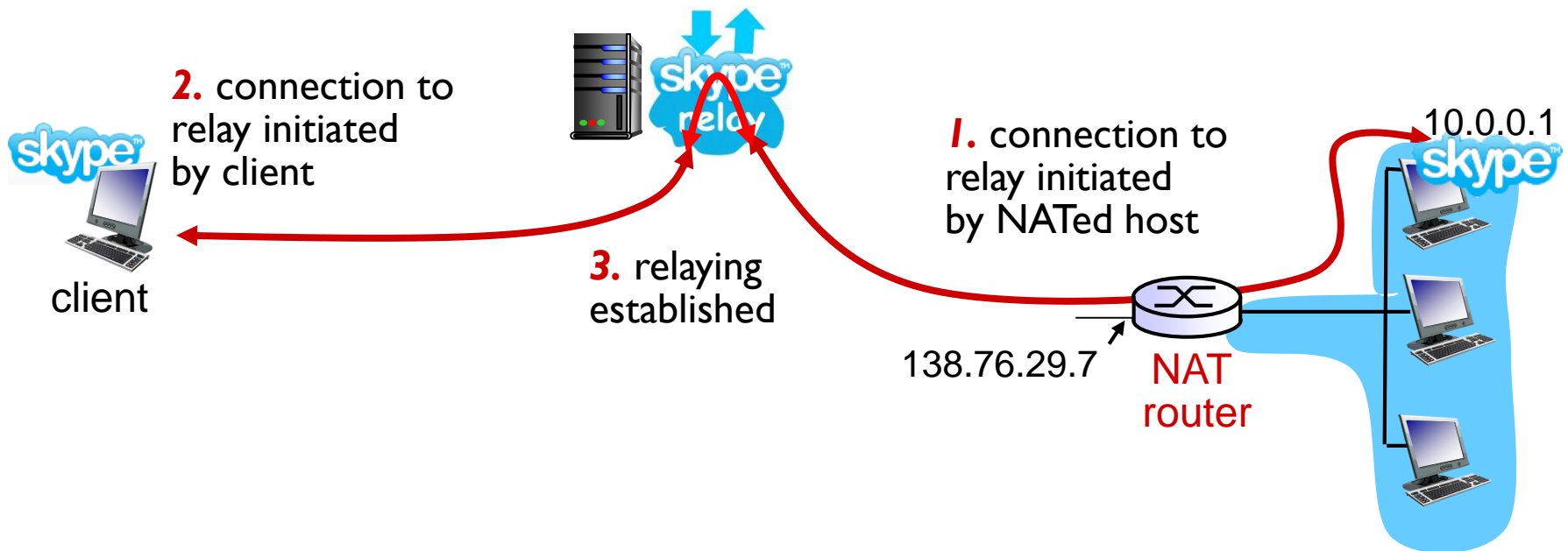
- ❖ learn public IP address (138.76.29.7)
- ❖ add/remove port mappings (with lease times)

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

- ❖ used by hosts & routers to communicate network-level information

- error reporting: unreachable host, network, port, protocol
- echo request/reply (used by ping)

- ❖ network-layer “above” IP:

- ICMP msgs carried in IP datagrams

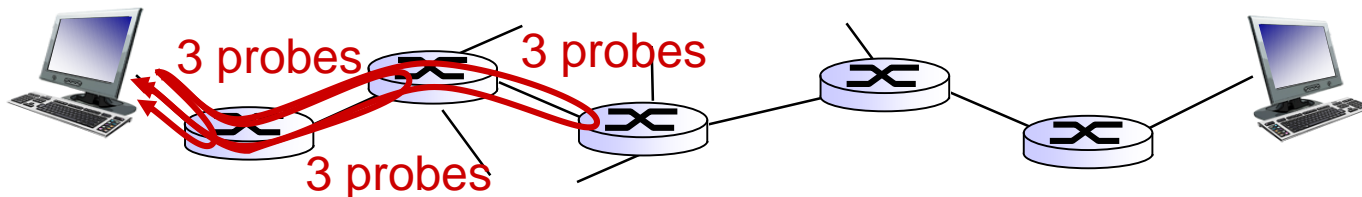
- ❖ **ICMP message:** type, code plus first 8 bytes of IP datagram causing error

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header



# 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  $n$ th 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



# 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

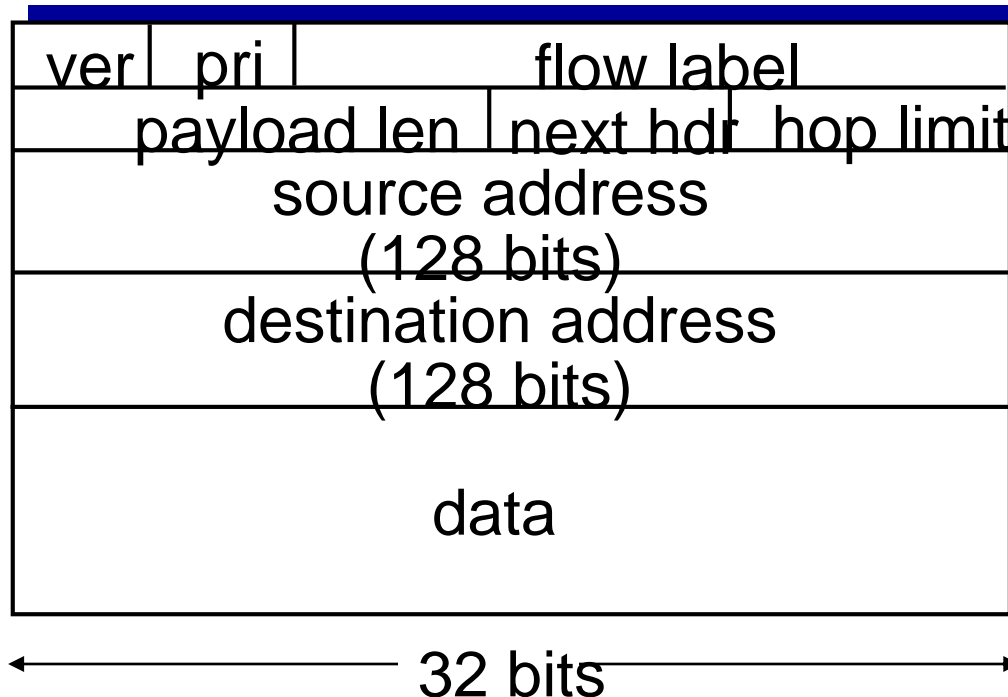
# IPv6 datagram format

*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

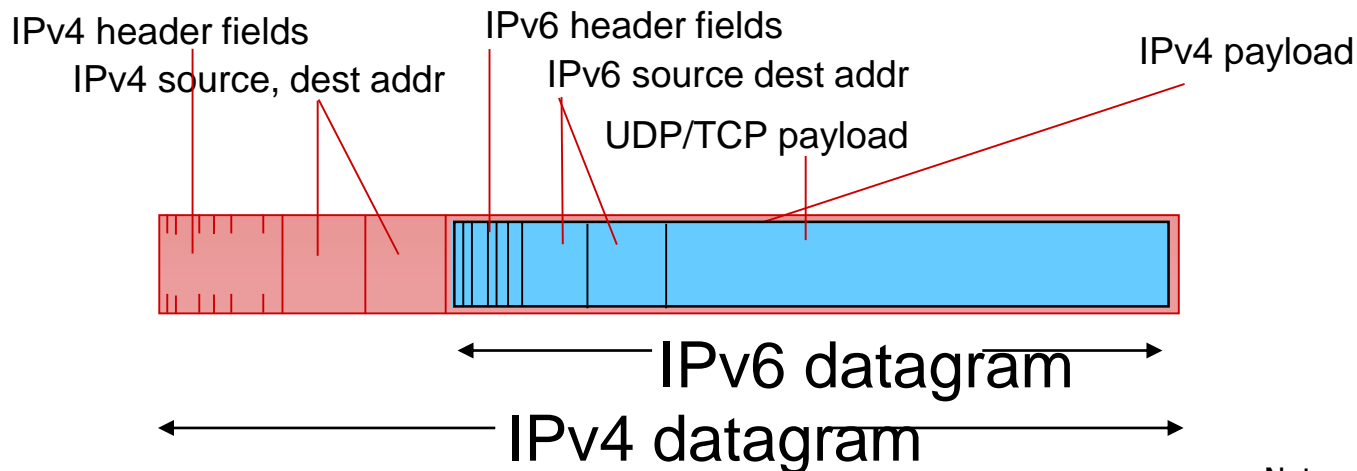


# Other changes from IPv4

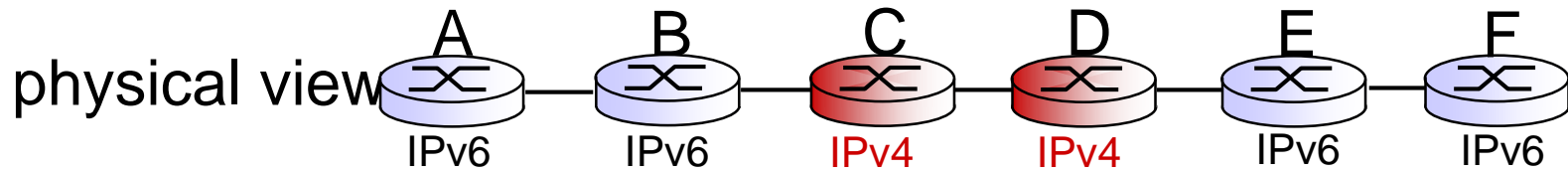
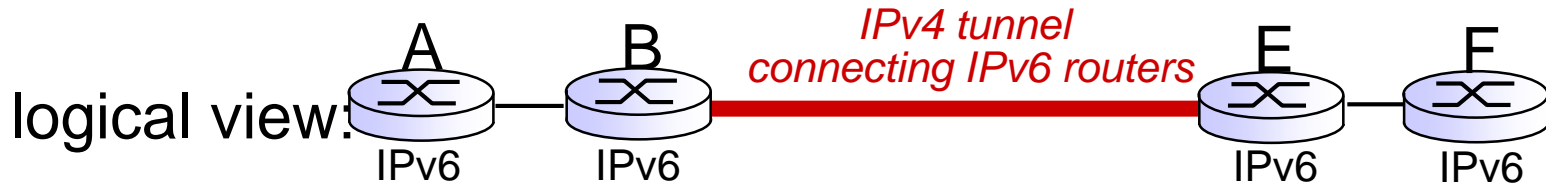
- ❖ *checksum*: removed entirely to reduce processing time at each hop
- ❖ *options*: allowed, but outside of header, indicated by “Next Header” field
- ❖ *ICMPv6*: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

# Transition from IPv4 to IPv6

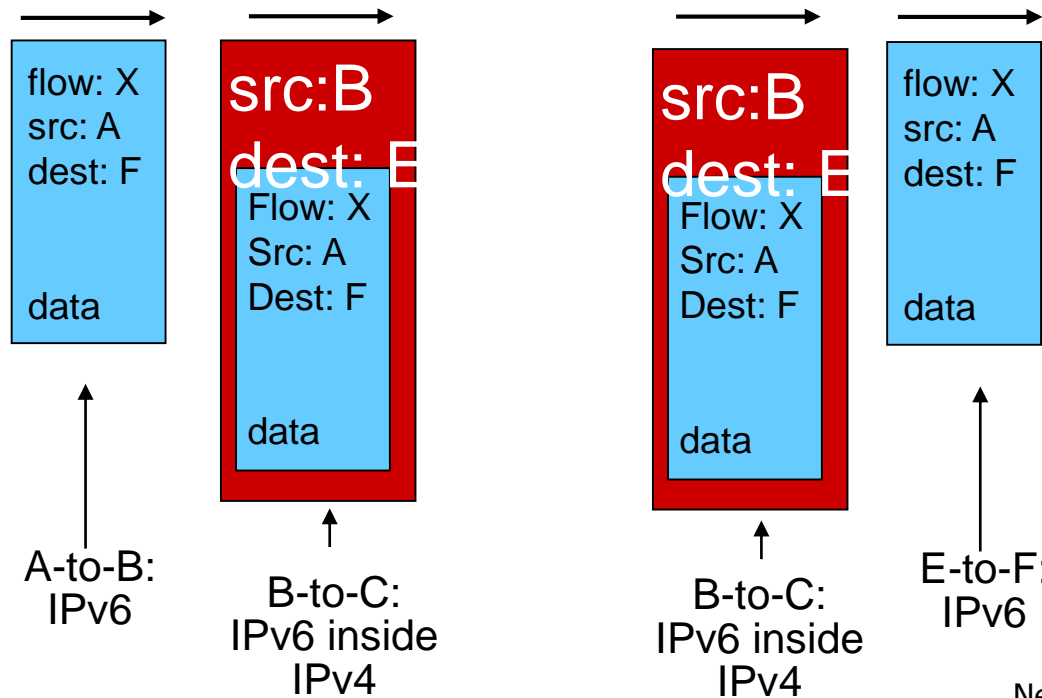
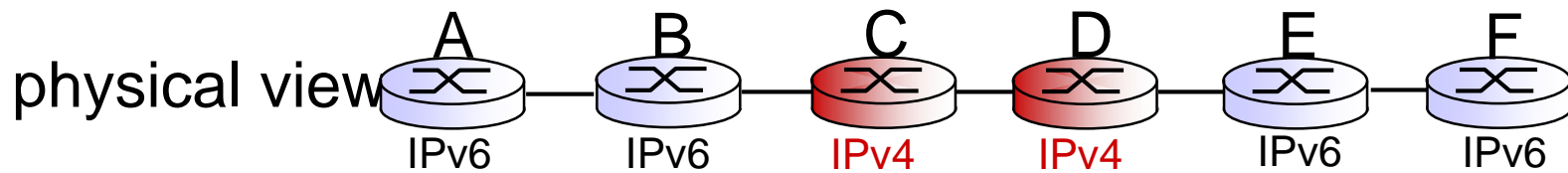
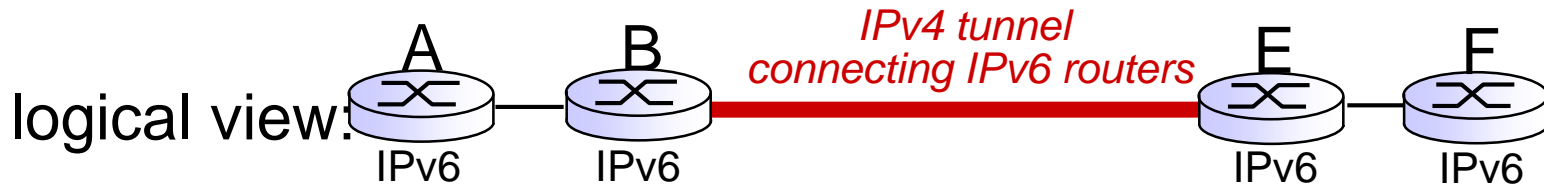
- ❖ not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?
- ❖ **tunneling**: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



# Tunneling



# Tunneling



# Next class

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