Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

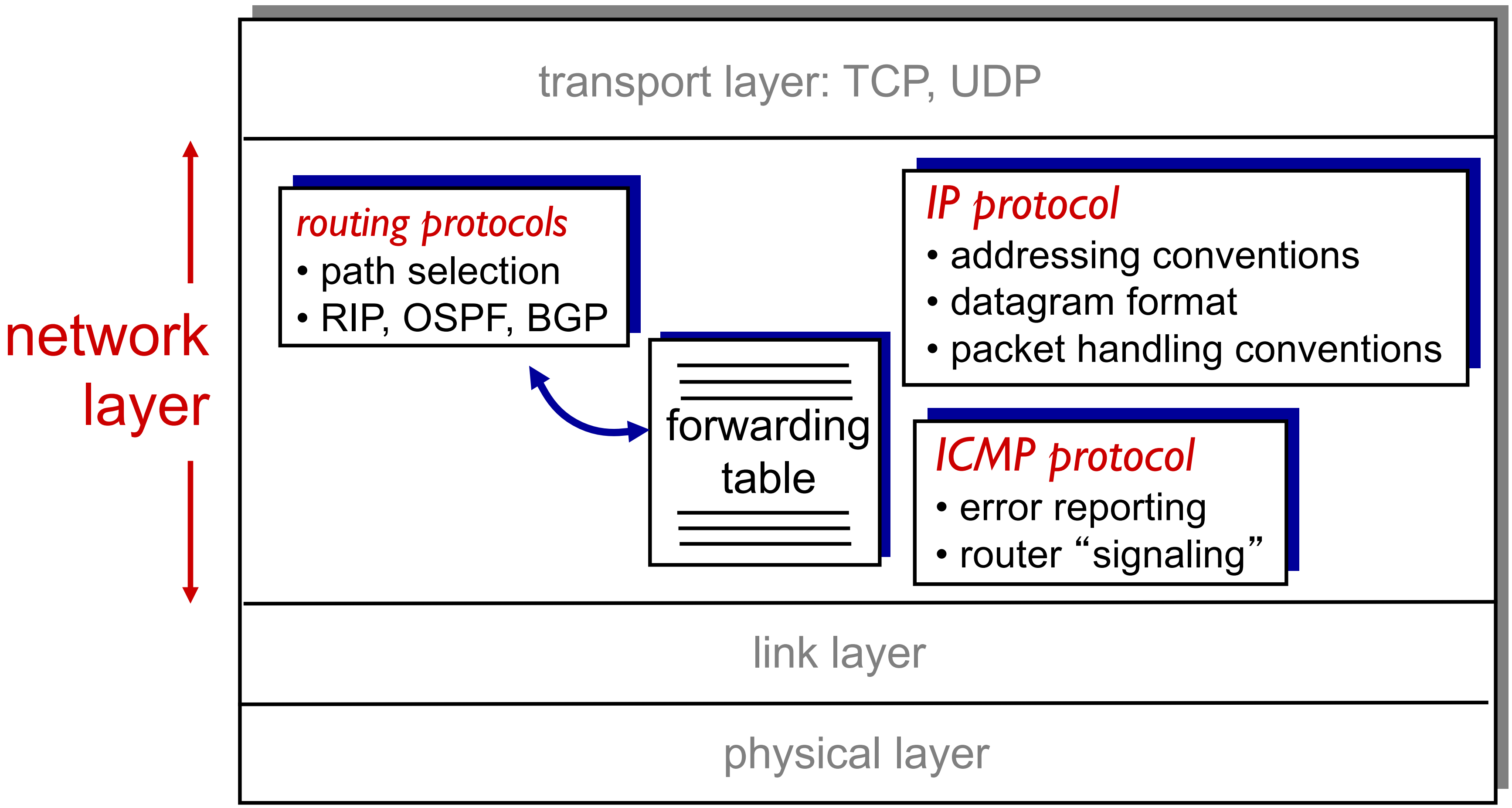
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

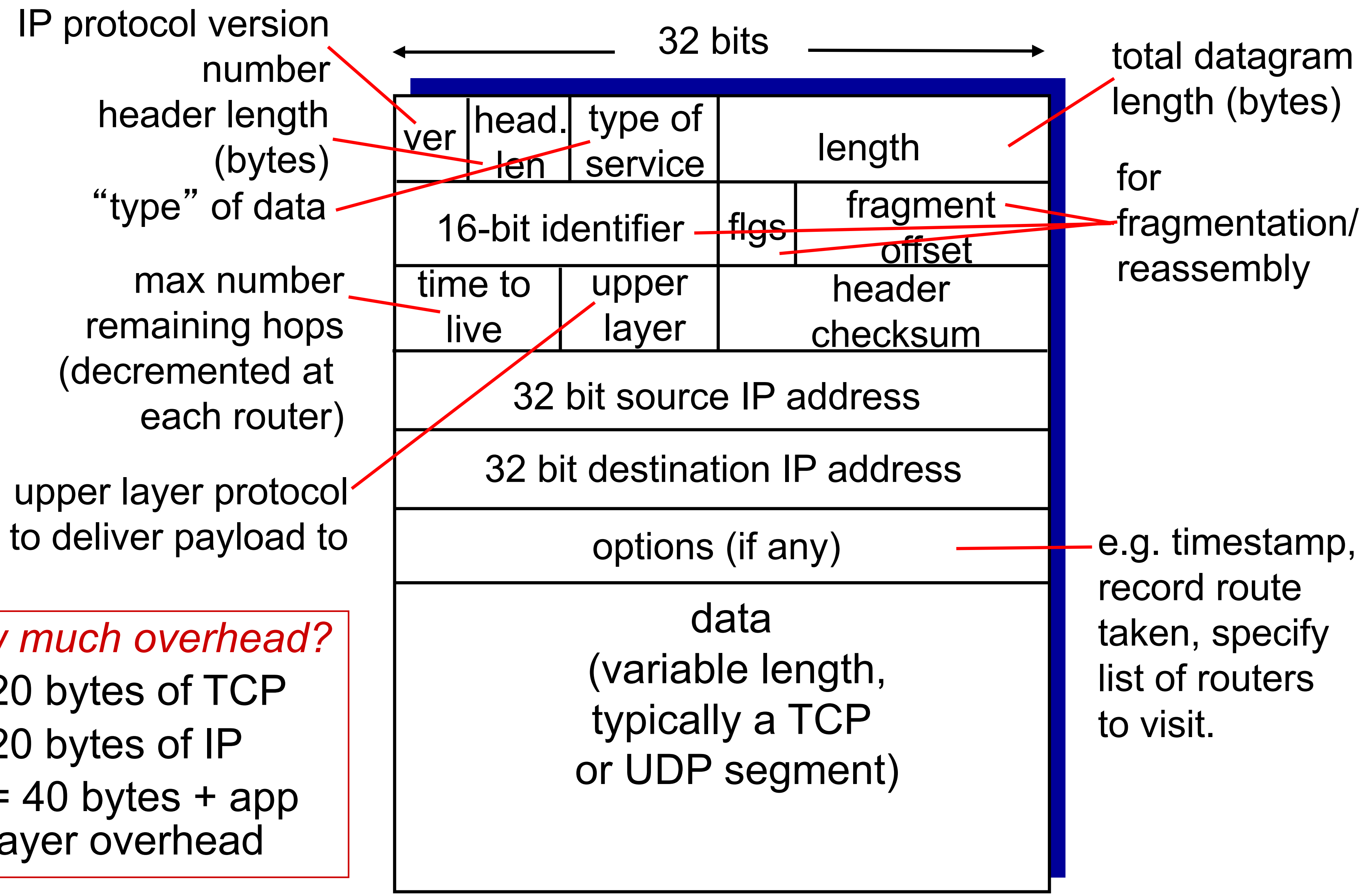
4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

The Internet network layer

host, router network layer functions:



IP datagram format

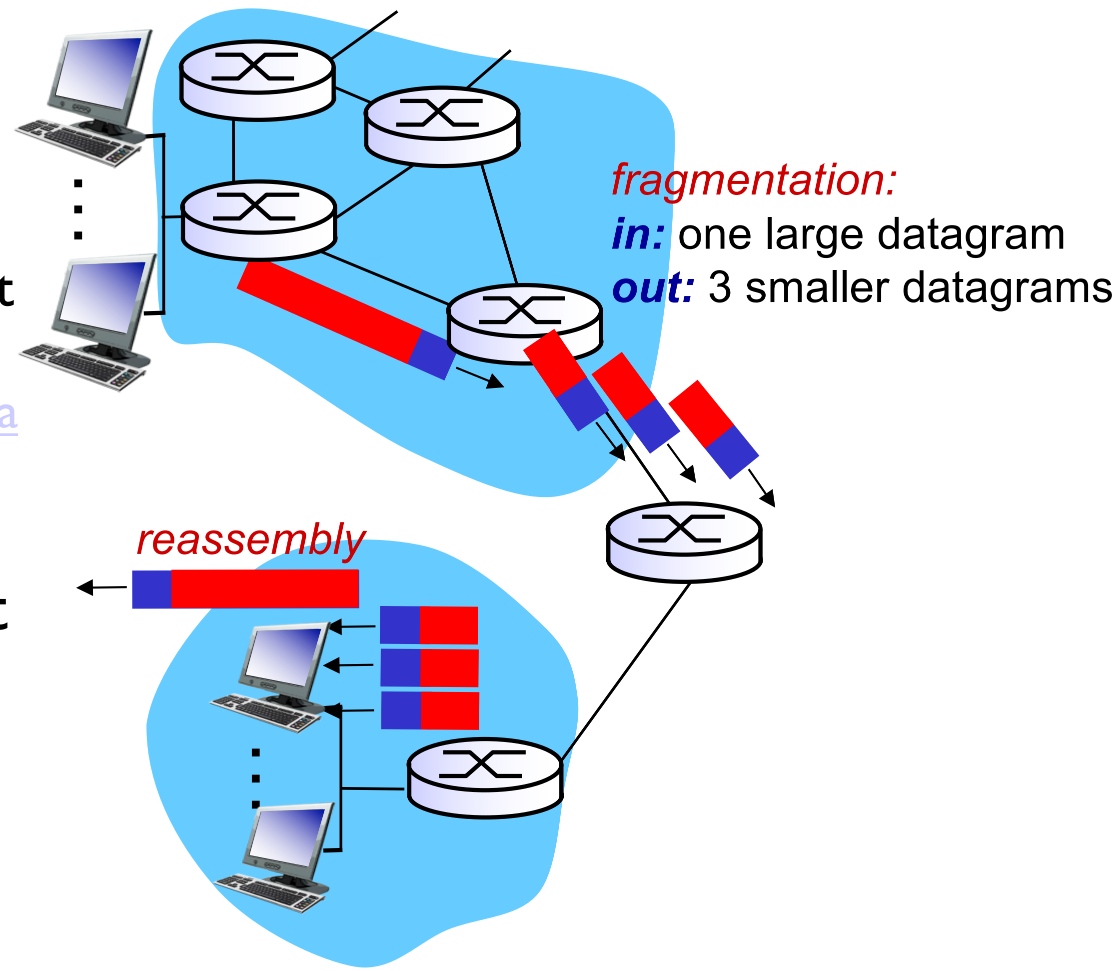


how much overhead?

- 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
  - https://en.wikipedia.org/wiki/Maximum_transmission_unit
- 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

ID: unique per packet, incremented by sending host

offset: multiple of 8.

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

offset = 2*1480/8

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

Original datagram: 20+3980=4000
1st fragment: 20+1480= 1500 starts at byte 0
2nd fragment: 20+1480 = 1500 starts at byte 185*8=1480
3rd fragment: 20+1020 = 1040 starts at byte (185+185)*8=2960
Data: 1480+1480+1020=3980

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

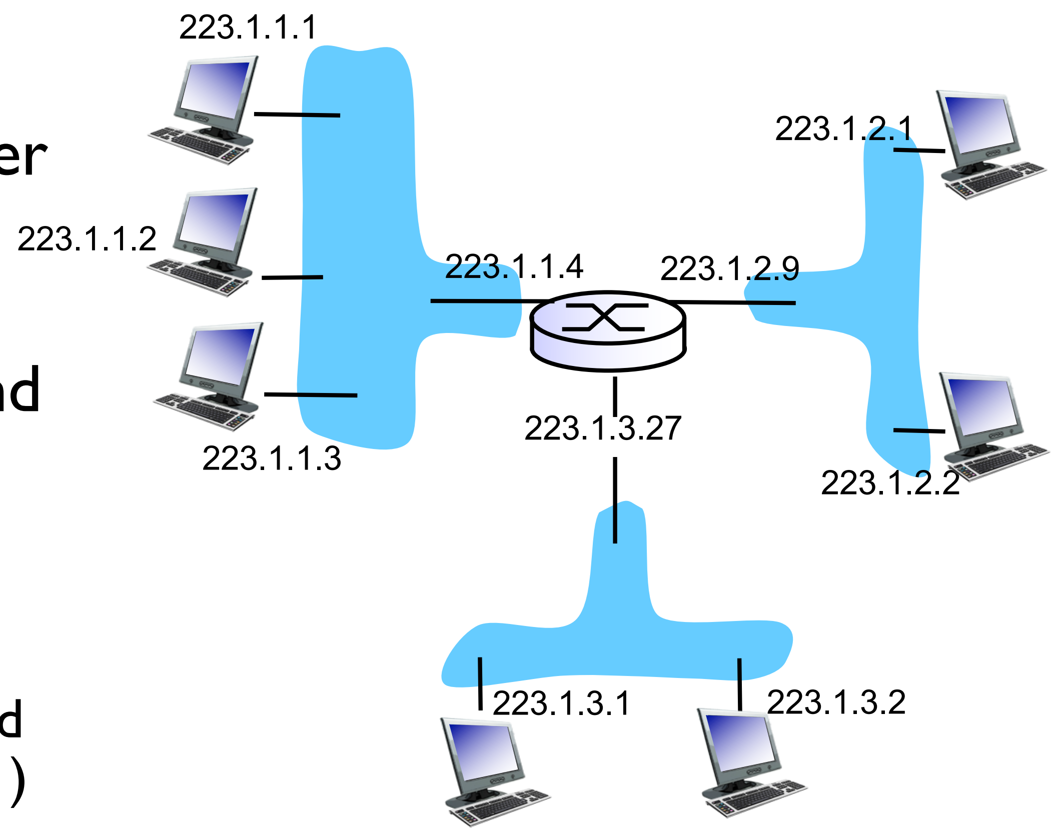
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

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*

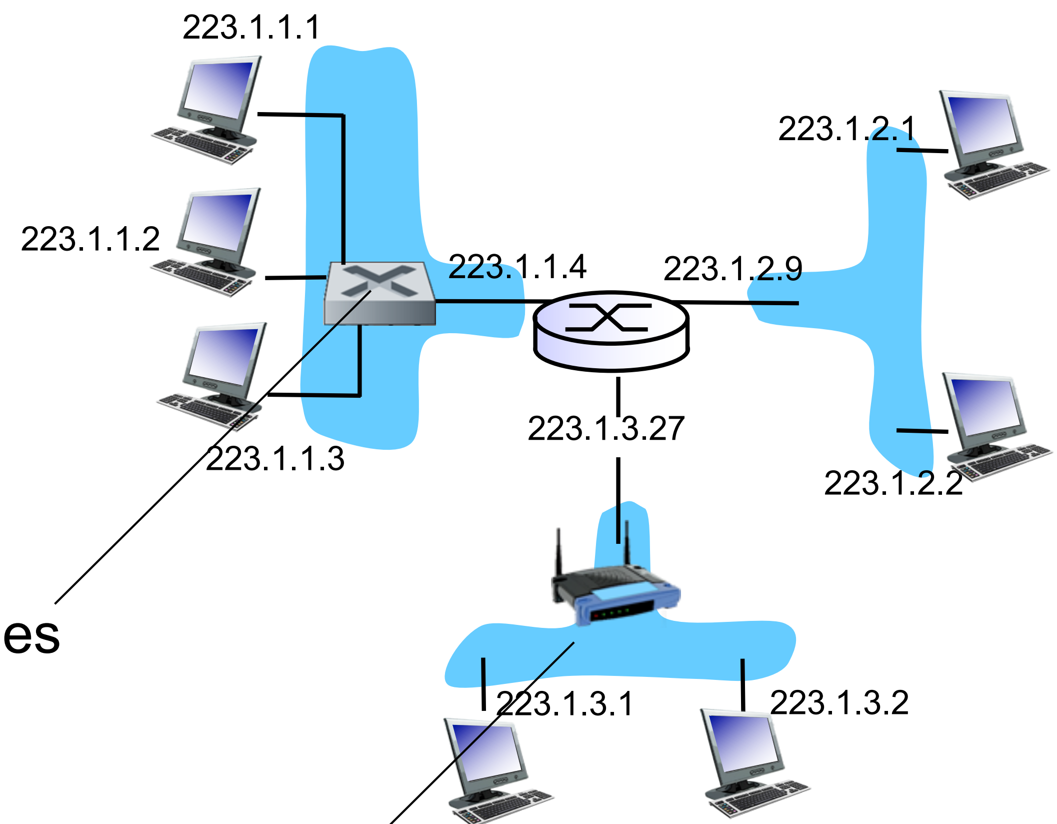


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

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



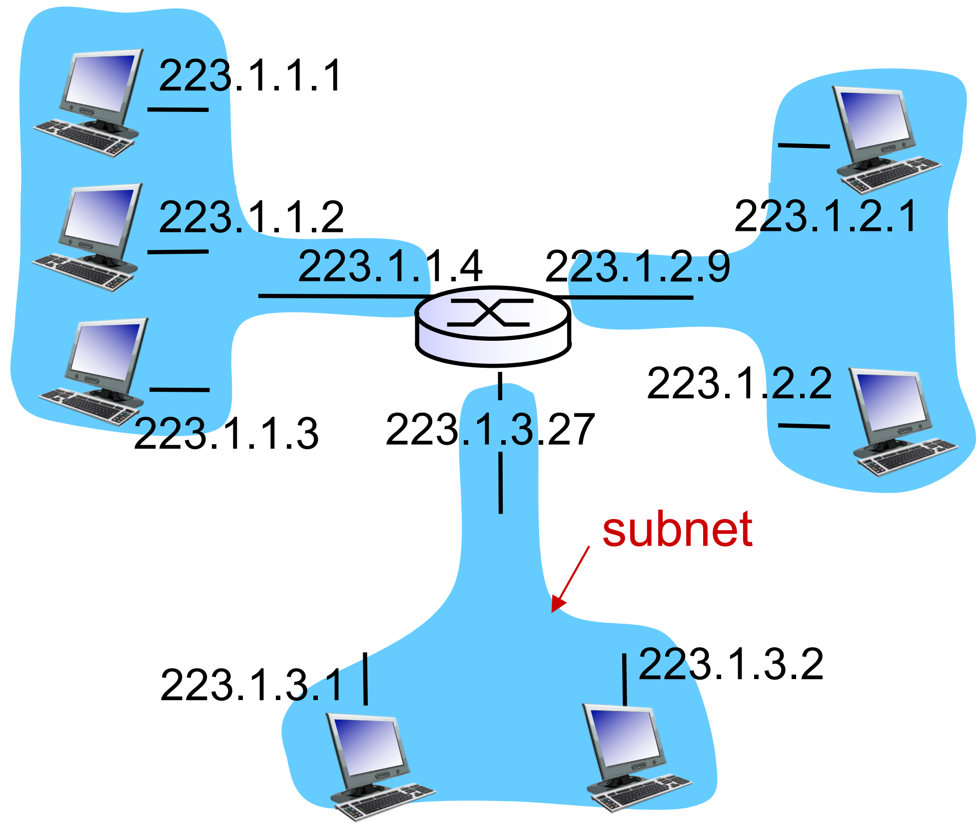
A: wired Ethernet interfaces connected by Ethernet switches

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 is a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

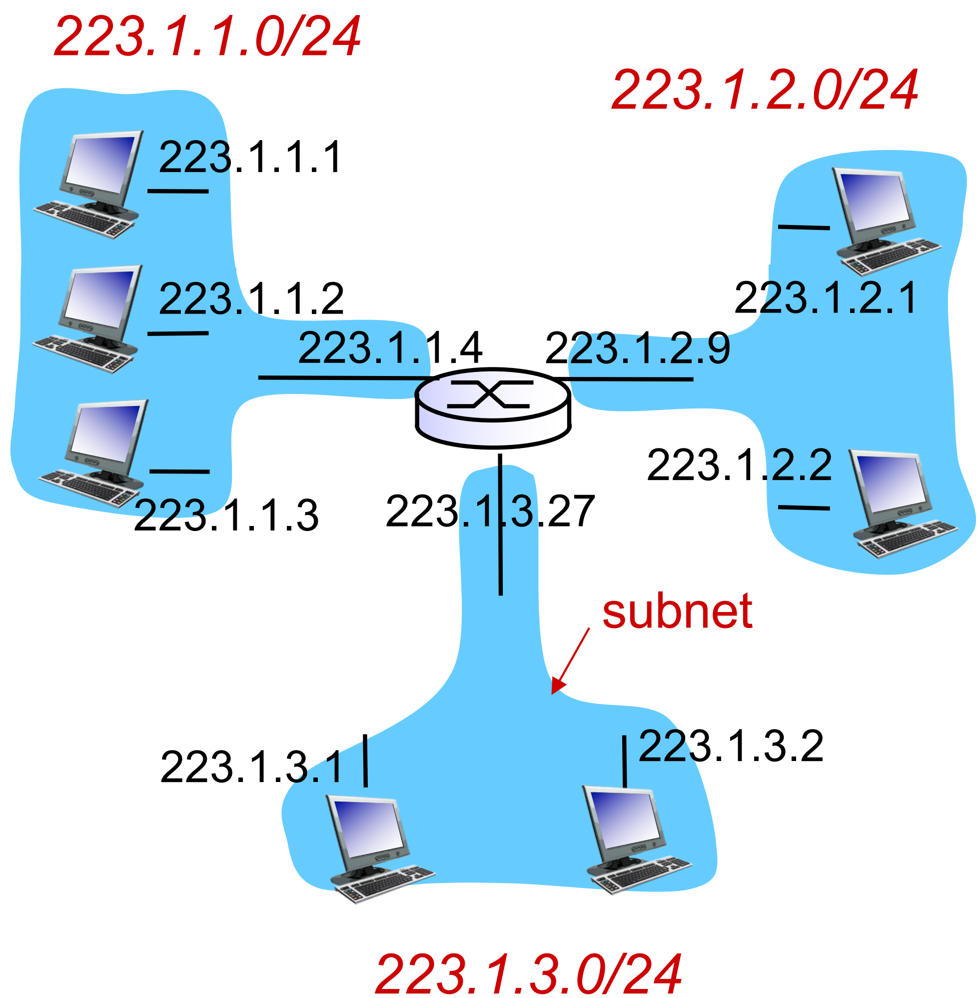


network consisting of 3 subnets

Subnets

recipe

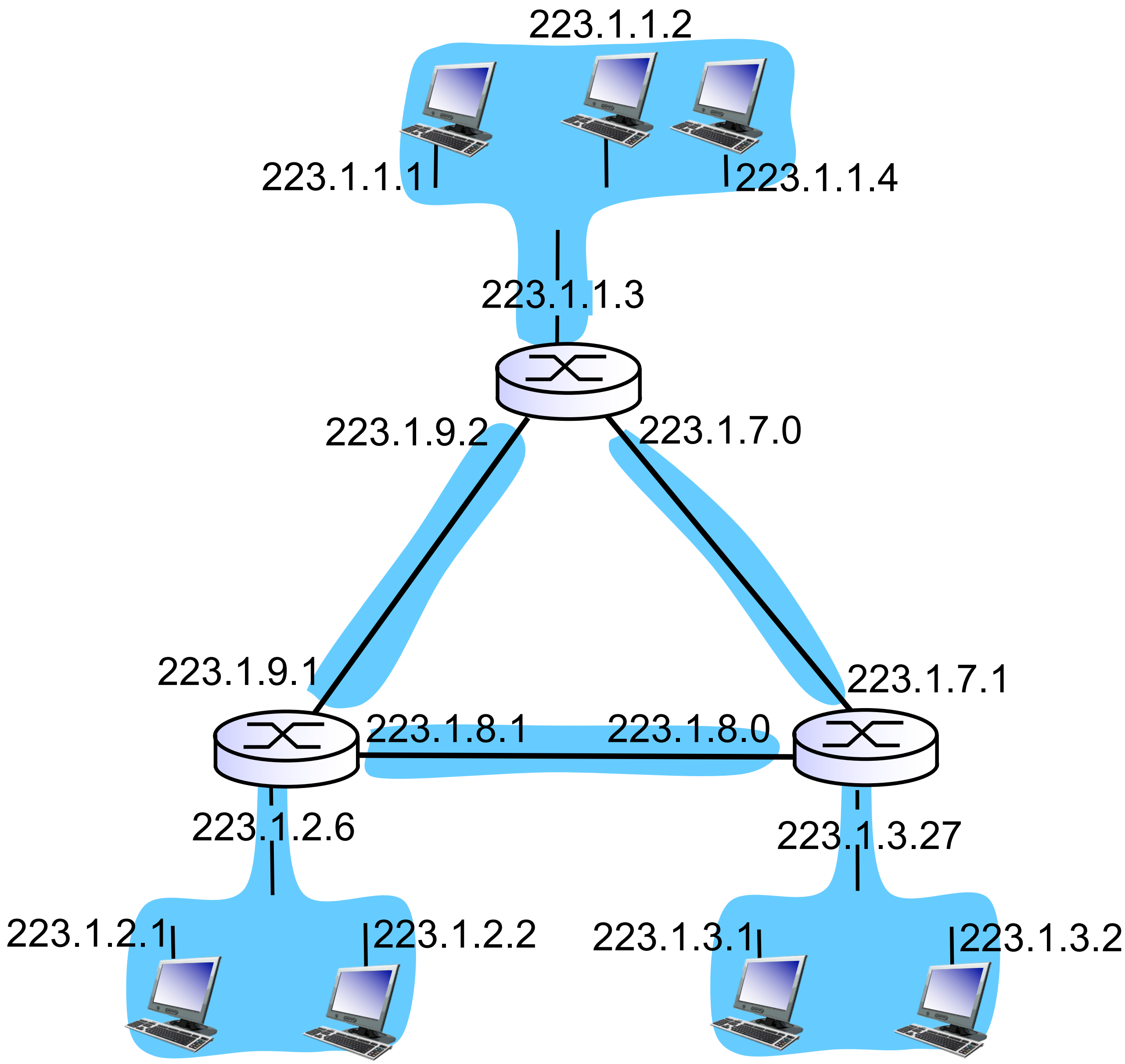
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

how many?



Historic Classful Network Architecture:

Class	Starting with (bits)	Range of first byte (decimal)	Network id format	Host id format	Number of networks	Numnbe r of hosts
A	0	0-127	a	b.c.d	2^7 = 128	2^{24} = 16777216
B	10	128-191	a.b	c.d	2^{14} = 16384	2^{16} = 65536
C	110	192-223	a.b.c	d	2^{21} = 2097152	2^8 = 256

subnet part ←→ host part

11001000 00010111 00010000 00000000

Class C network ("/24"): 200.23.16.0

Example IP in that network: 200.23.16.1

(Since 1993) 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 part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Q: what did we gain?

A: more efficient use of the IP address space

Some special IP addresses

- All 1's: means "all hosts on this subnet"

subnet part | host part

11001000 00010111 00010001 11111111

200.23.17.255/23

- All 0's: means "this subnet"

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

More IP addresses

- Reserved IP addresses for special purposes
  - https://en.wikipedia.org/wiki/Reserved_IP_addresses#IPv4
  - 127.0.0.1: local host
  - Multicast
    - 224.0.0.0/8–239.0.0.0/8
  - Private networks:
    - not routed, typically used through NATs.
    - 24- bit block, /8 prefix, 1xA: 10.0.0.0-10.255.255.255
    - 20-bit block, /12 prefix, 16xB: 172.16.0.0- 172.31.255.255
    - 16-bit block, /16 prefix, 256xC: 192.168.0.0-192.168.255.255
  - Assigned to special institutions

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
  - Mac: /etc/resolv.conf
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

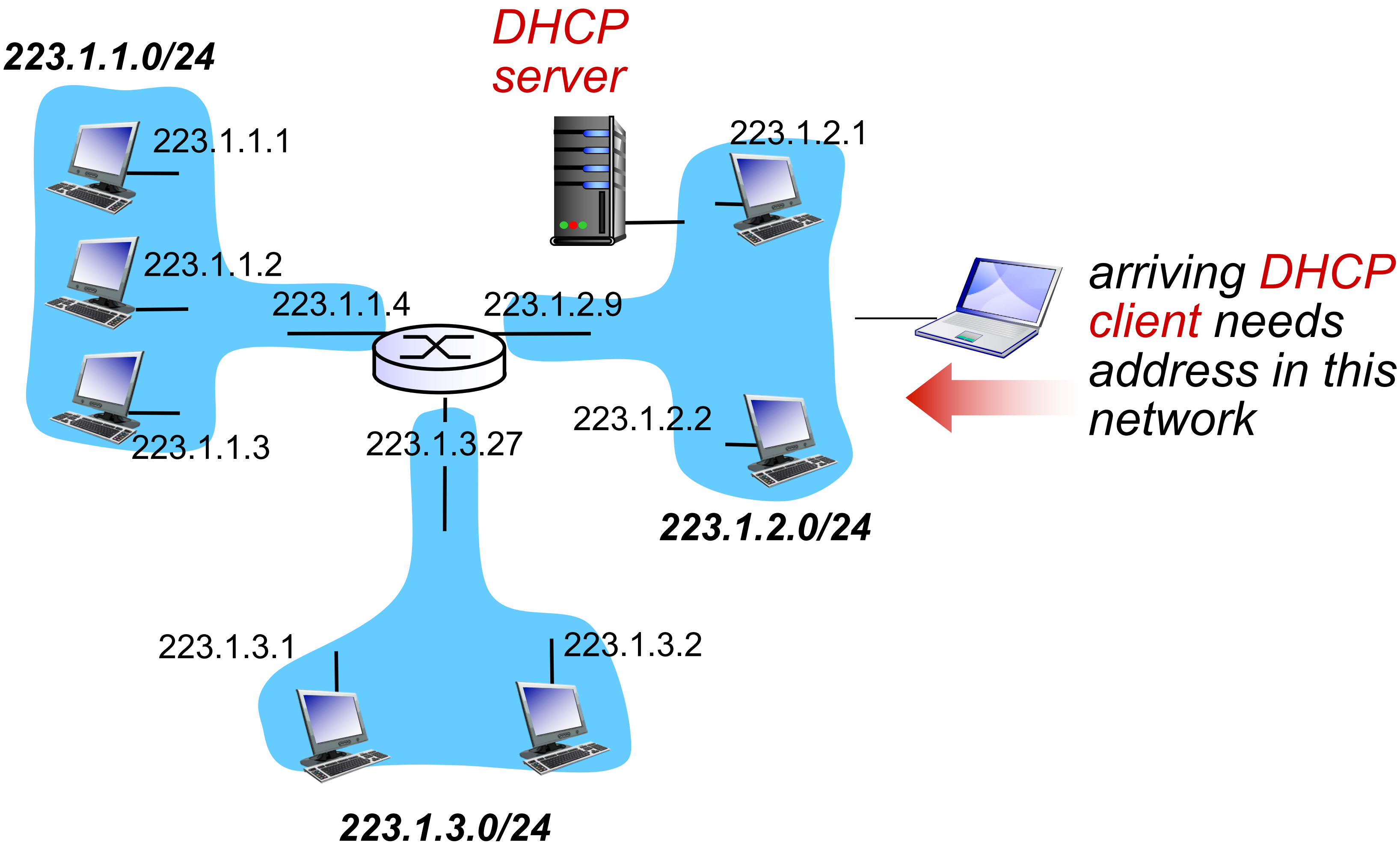
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- 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)

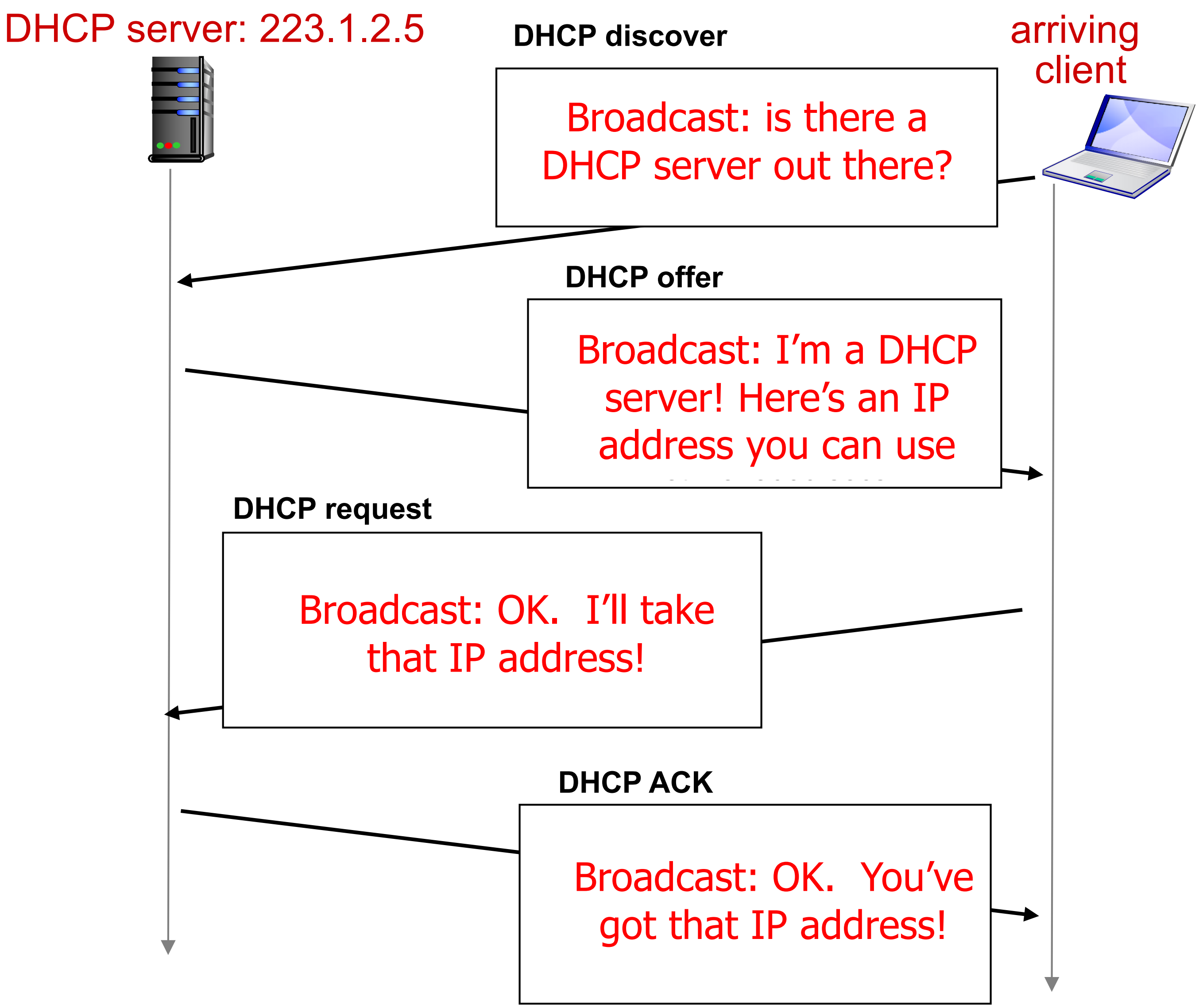
DHCP overview:

- 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

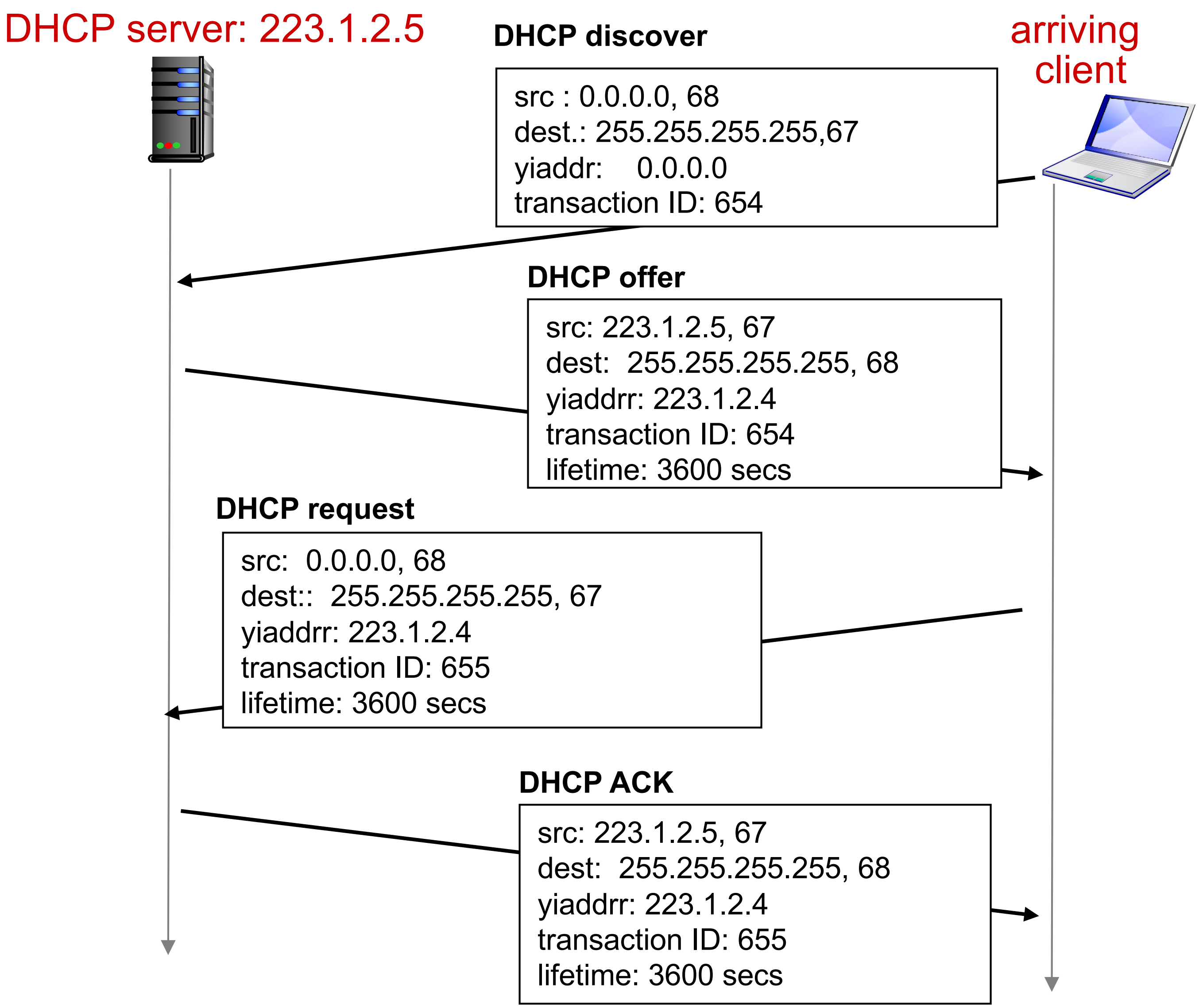
DHCP client-server scenario



DHCP client-server scenario



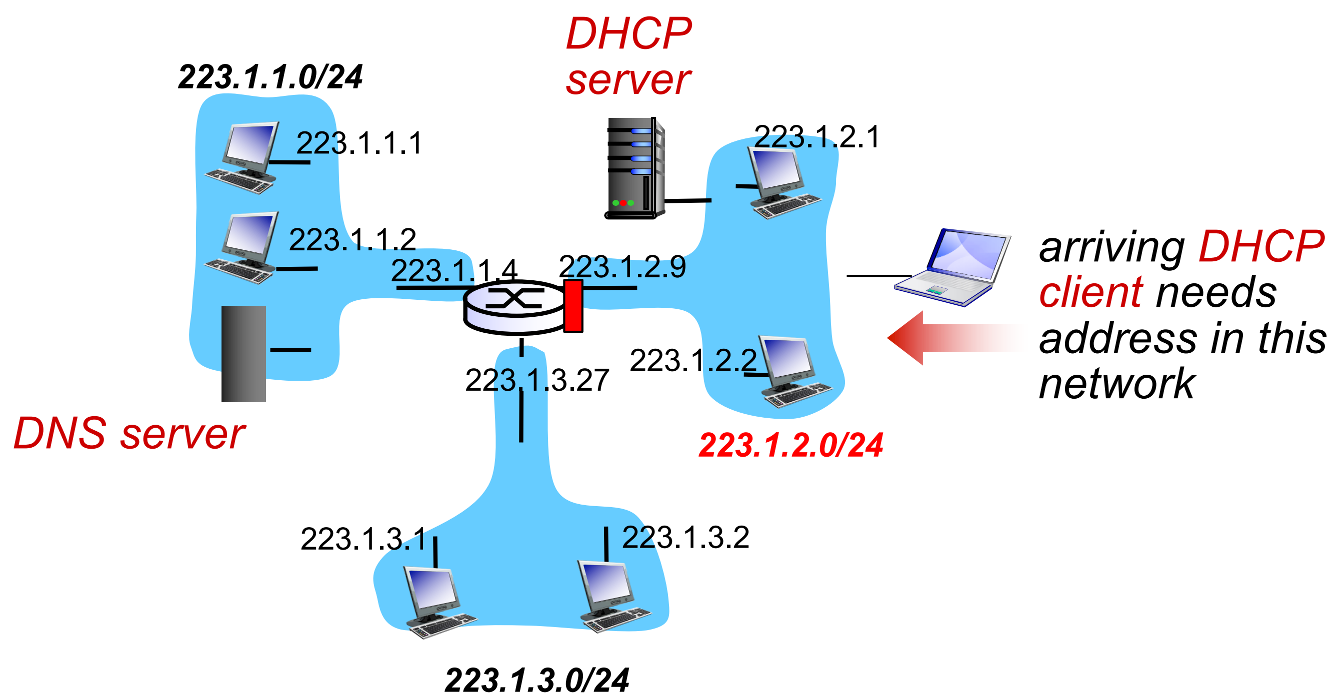
DHCP client-server scenario



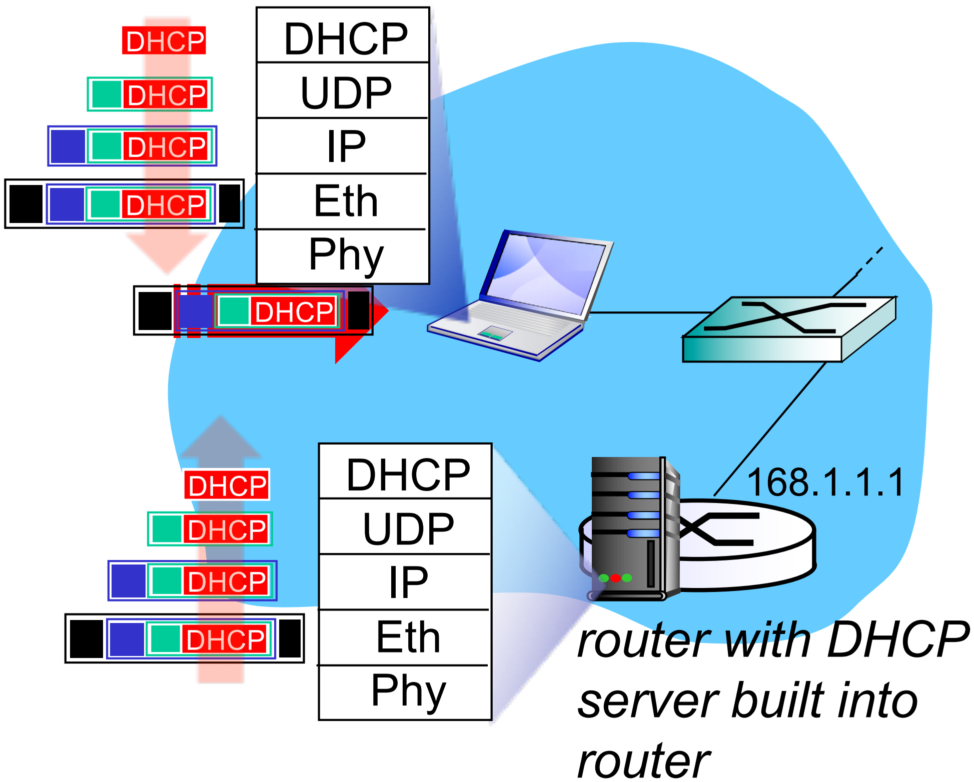
DHCP: more than IP addresses

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

- address of first-hop router for client (223.1.2.1)
- name and IP address of DNS server
- network mask (indicating network versus host portion of address) /24

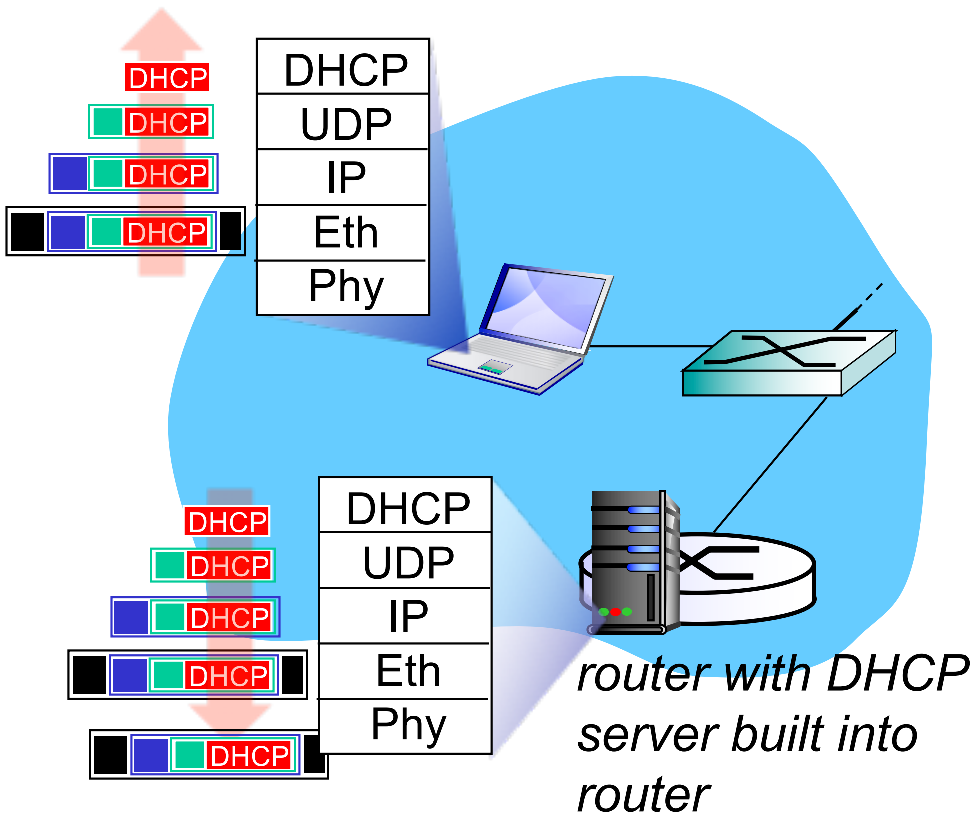


DHCP: example



- 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



- 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

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."

Recap: how does an end-host gets an IP address

Q: How does a *host* get an 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?

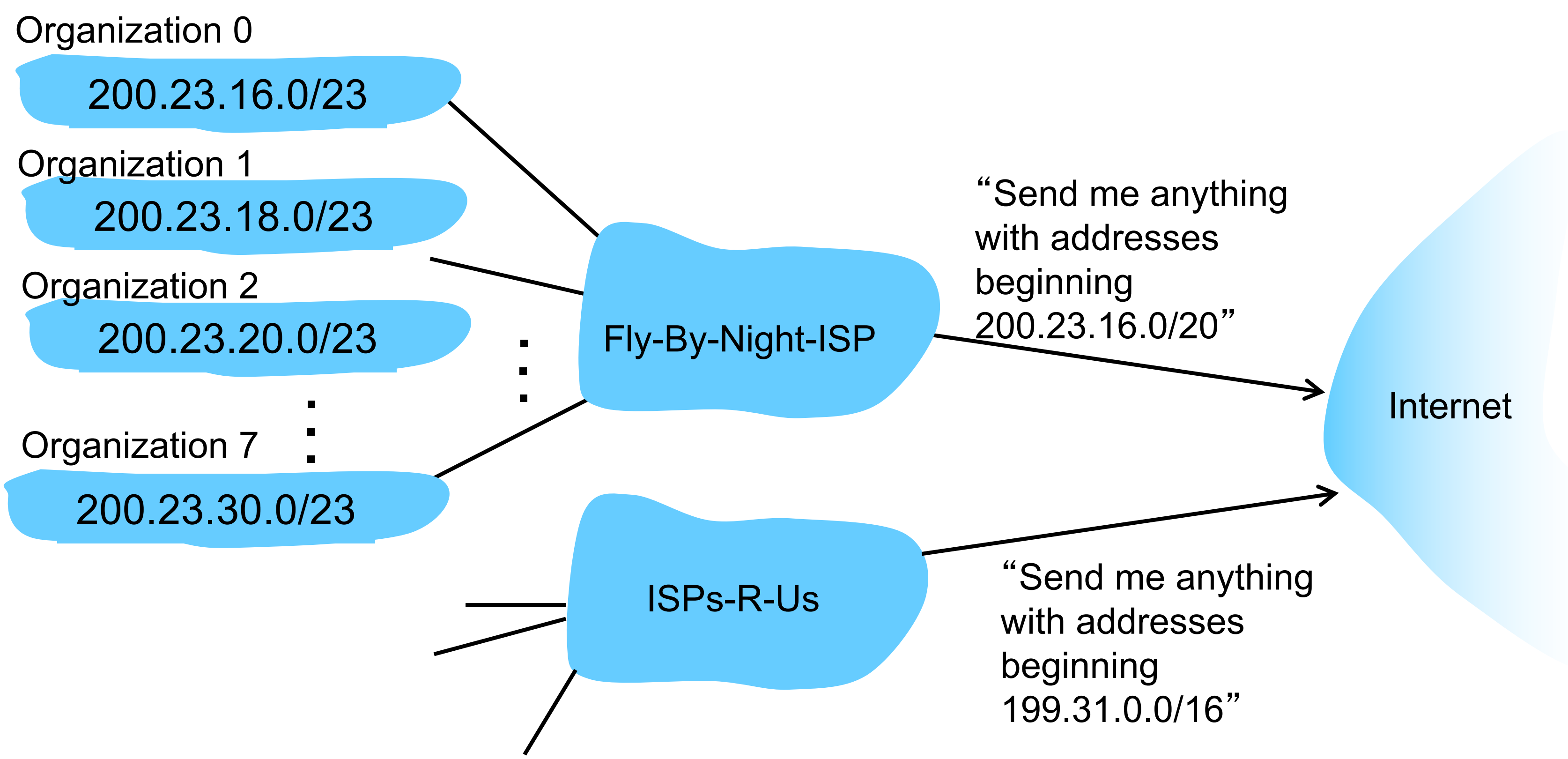
Q: how does a *network* get subnet part of IP addr?

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

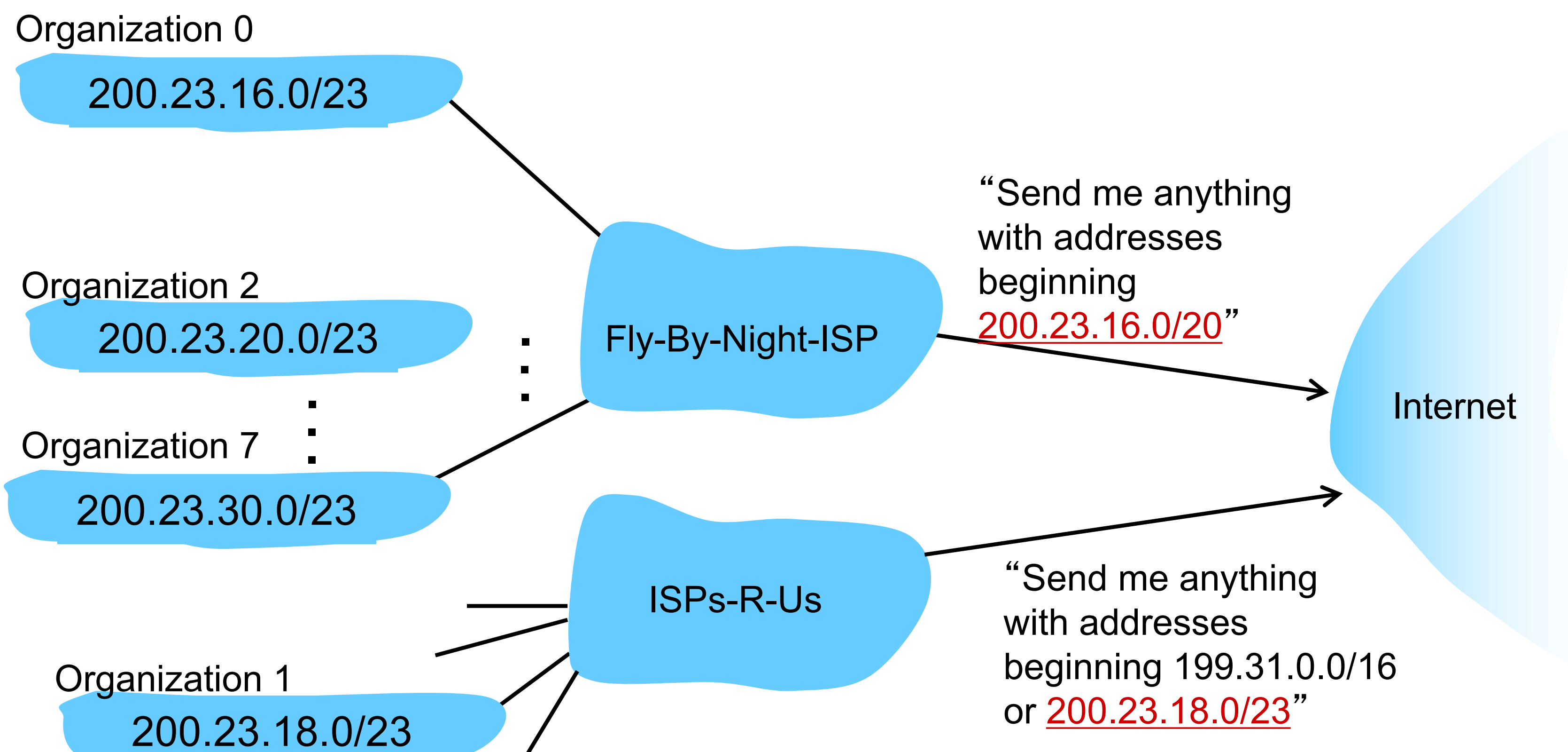
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

- ISPs-R-Us has a more specific route to Organization 1
- Longest Prefix Match preferred



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
  - http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml
- manages DNS
- assigns domain names, resolves disputes
- delegates to regional registries (RIRs)

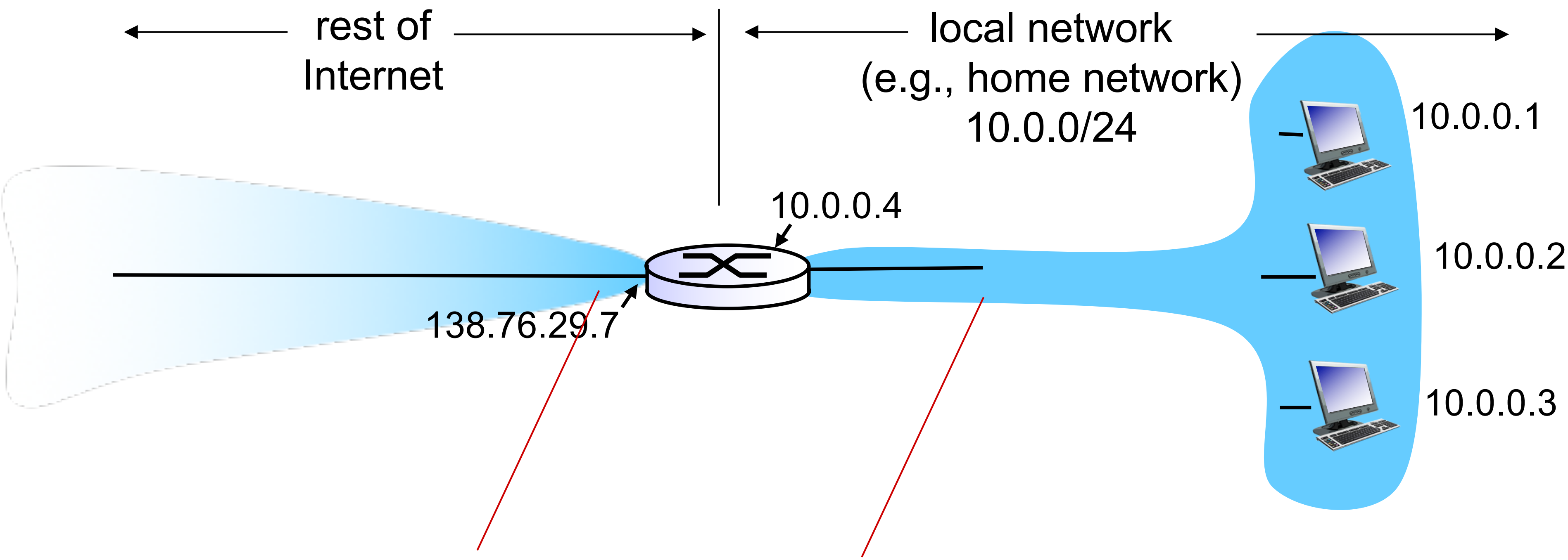
IP addresses: a scarce resource!

- More users than IP addresses
  - In the 1970's: 2^{32} was plenty of addresses
  - ~2010: 1B computers, 5B phones
- Bad utilization
  - Some prefixes are full
  - Some are reserved and unused!
- How are they allocated today?
  - http://en.wikipedia.org/wiki/List_of_assigned_/8_IPv4_address_blocks
  - ARIN: https://en.wikipedia.org/wiki/American_Registry_for_Internet_Numbers
  - http://www.caida.org/research/id-consumption/whois-map/

IP addresses: a scarce resource!

- More users than IP addresses:
  - In the 1970's: 2^{32} was plenty of addresses
  - Today: >1B computers, >7B phones
- Bad utilization of existing IP addresses
  - Some prefixes are full
  - Some are reserved and unused!
  - CAIDA's map
- Solutions:
  - DHCP: get an IP dynamically, when you use it
  - NAT: entire subnet uses one public IP externally
  - IPv6: increase the IP address from 32 to 64 bits.

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

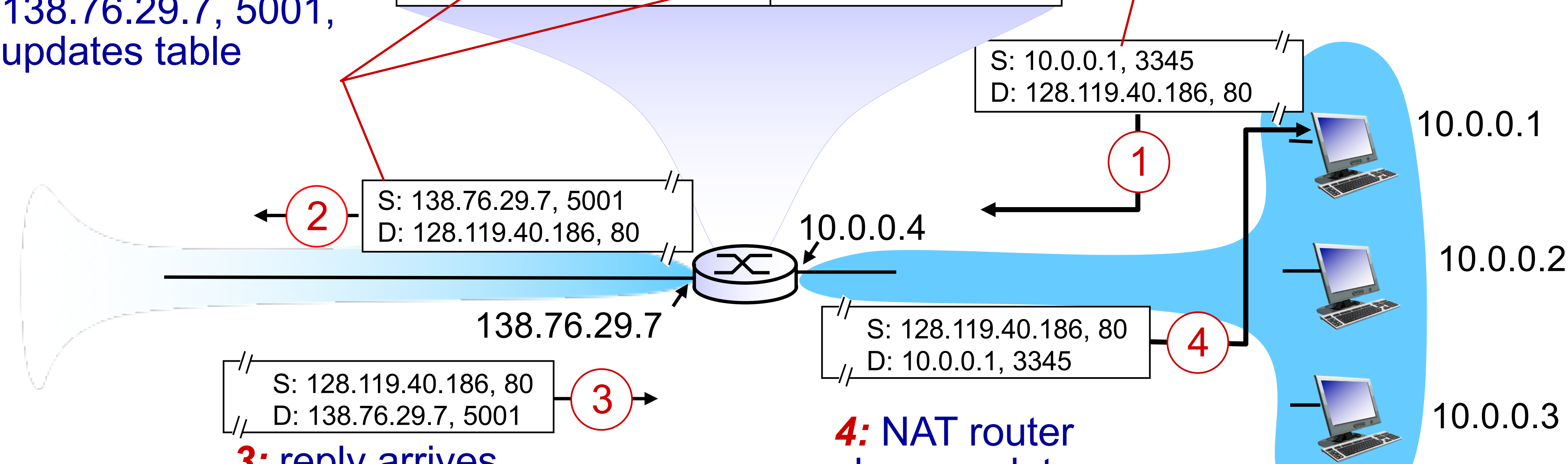
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

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



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

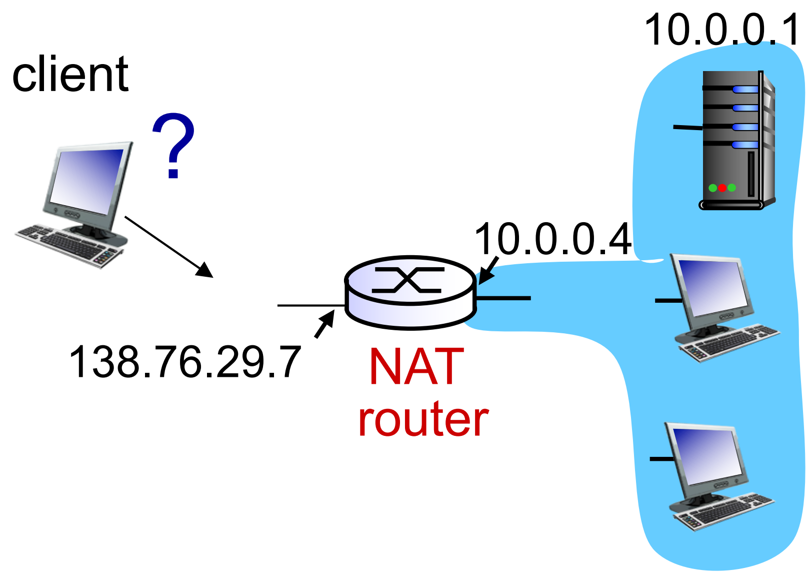
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

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
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

NAT traversal problem

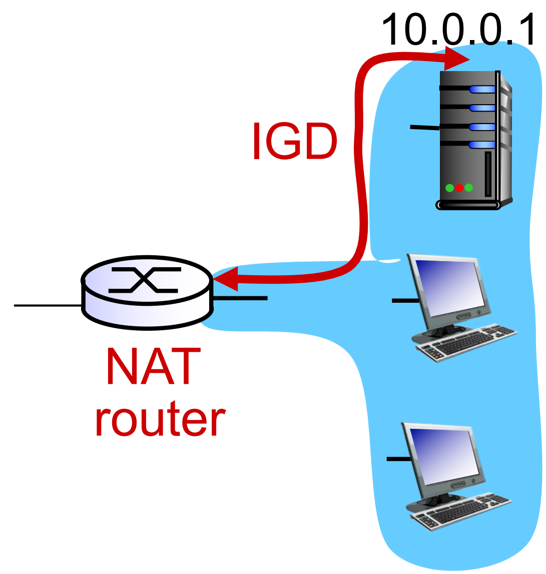
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

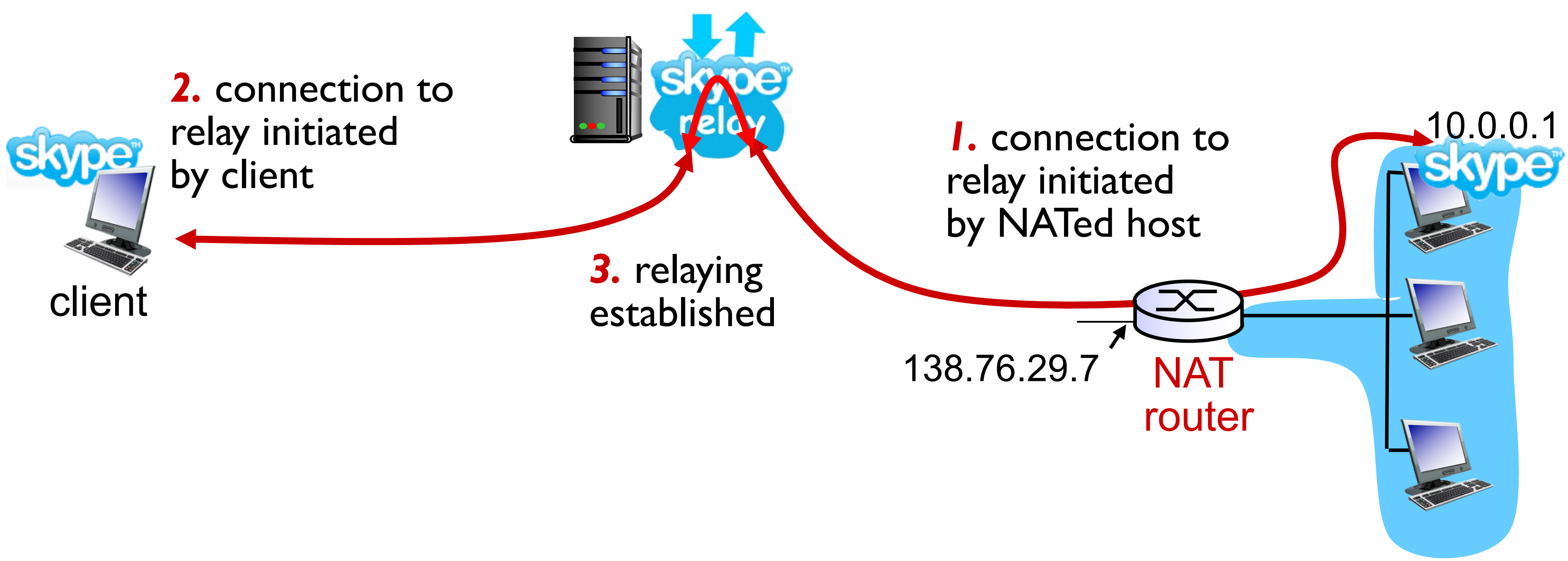
- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  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 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IPv6: motivation

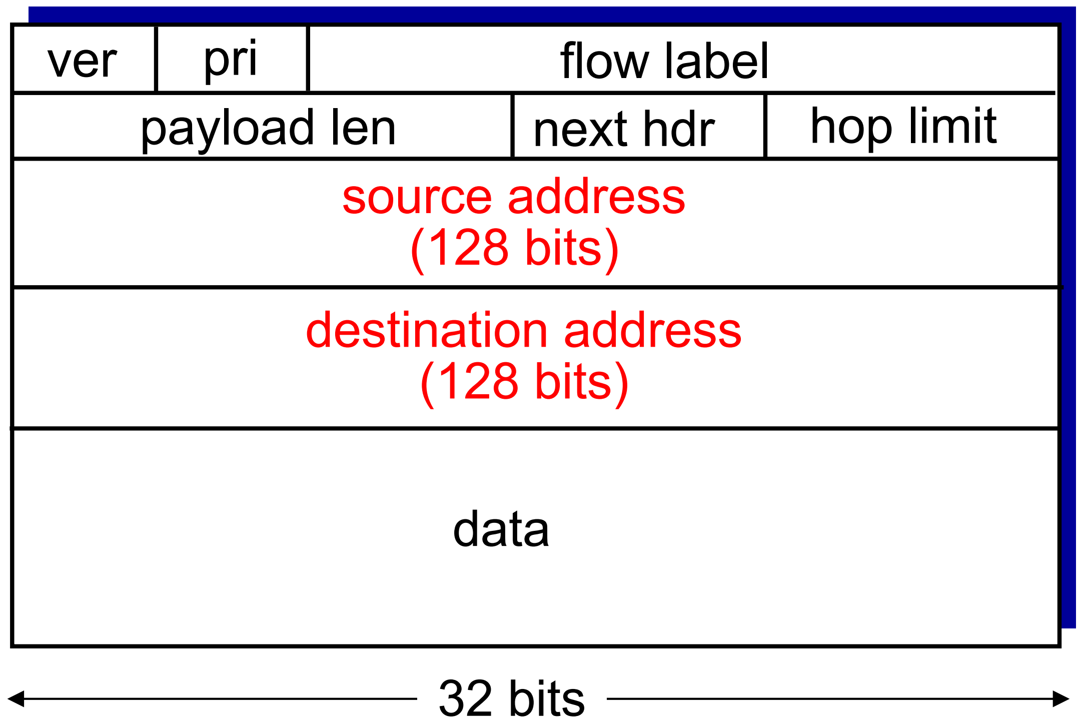
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

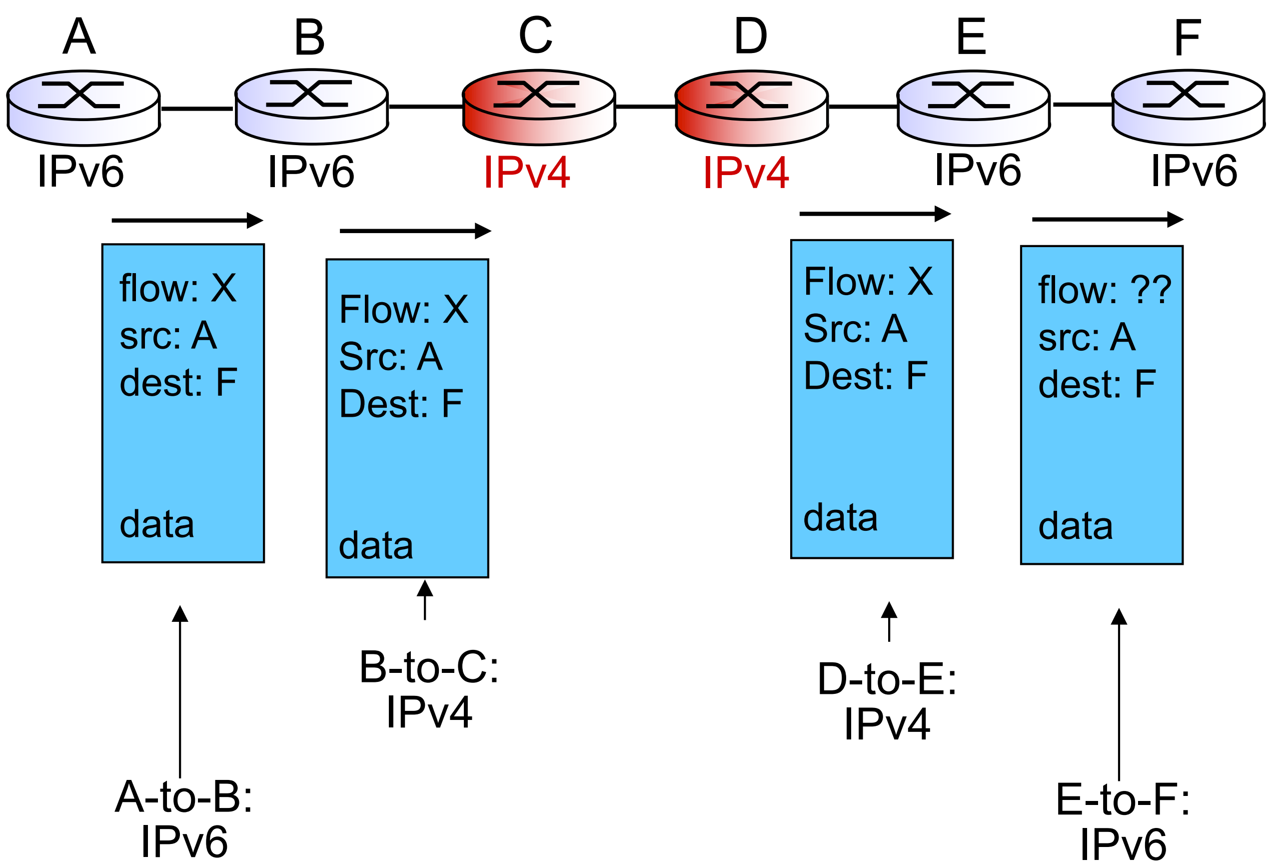
- *priority:* identify priority among datagrams in flow
- *flow Label:* identify datagrams in same "flow." (concept of"flow" not well defined).
- *next header:* identify upper layer protocol for data
- *checksum*: removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new ICMP(additional message types, e.g. "Packet Too Big", multicast group management)



Transition from IPv4 to IPv6

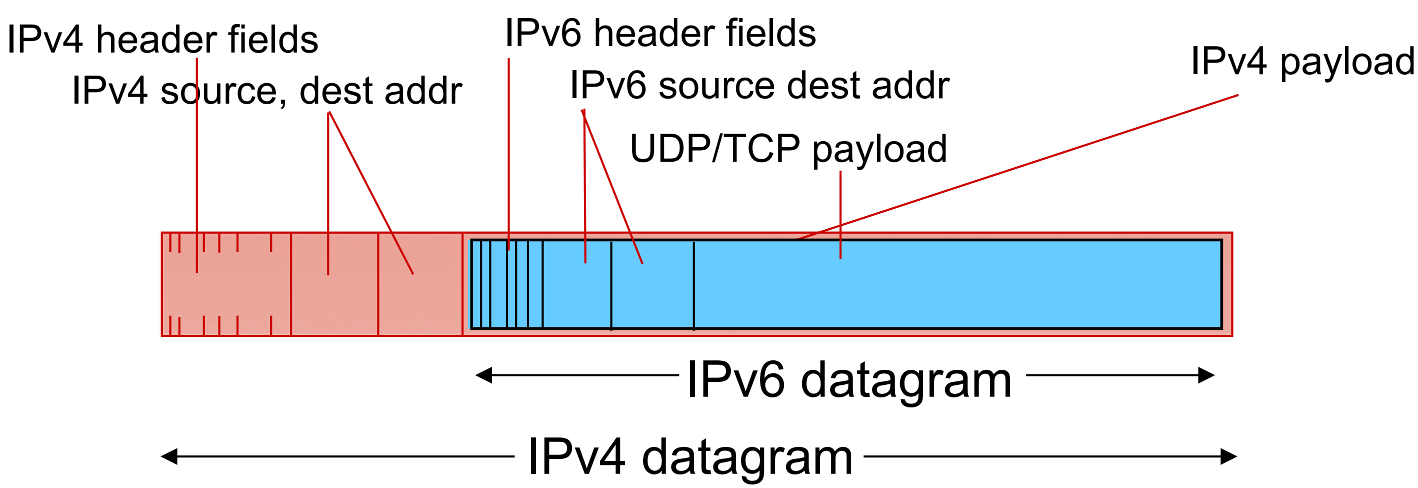
- Not all routers can be upgraded simultaneously
  - no "flag days": Could we shut down the Internet on Friday on IPv4, upgrade all routers to IPv6, and restart on Monday?
- how will network operate with mixed IPv4 and IPv6 routers?
  - *Dual stack: IPv4/IPv6 [RFC4213]*
    - *Have both ipv4 (which defeats the purpose) and ipv6 addresses*
    - *Talk ipv4 to ipv4 routers and ipv6 to ipv6 routers*
    - *Determine if other routers are ipv4 or ipv6 – capable through DNS: DNS can return different addresses based on the capabilities of requesting node*
    - *If ipv4-capable talks to ipv6-capable: convert ipv6 → ipv4,*
      - *Ipv4→ipv6 will not recover some of the IPv6 fields s*

Dual Stack

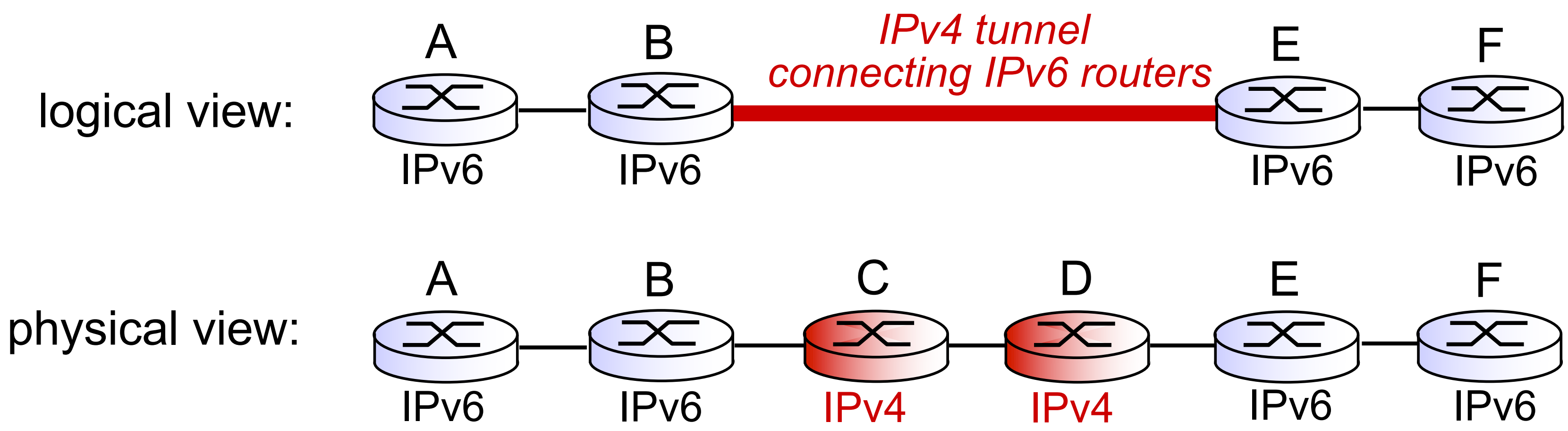

A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
flow: ??
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv4
D-to-E:
IPv4
E-to-F:
IPv6

Transition from IPv4 to IPv6

- Not all routers can be upgraded simultaneously
  - no "flag days": Could we shut down the Internet on Friday on IPv4, upgrade all routers to IPv6, and restart on Monday?
- how will network operate with mixed IPv4 and IPv6 routers?
  - *Dual stack: IPv4/IPv6 [RFC4213]*
  - *Tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

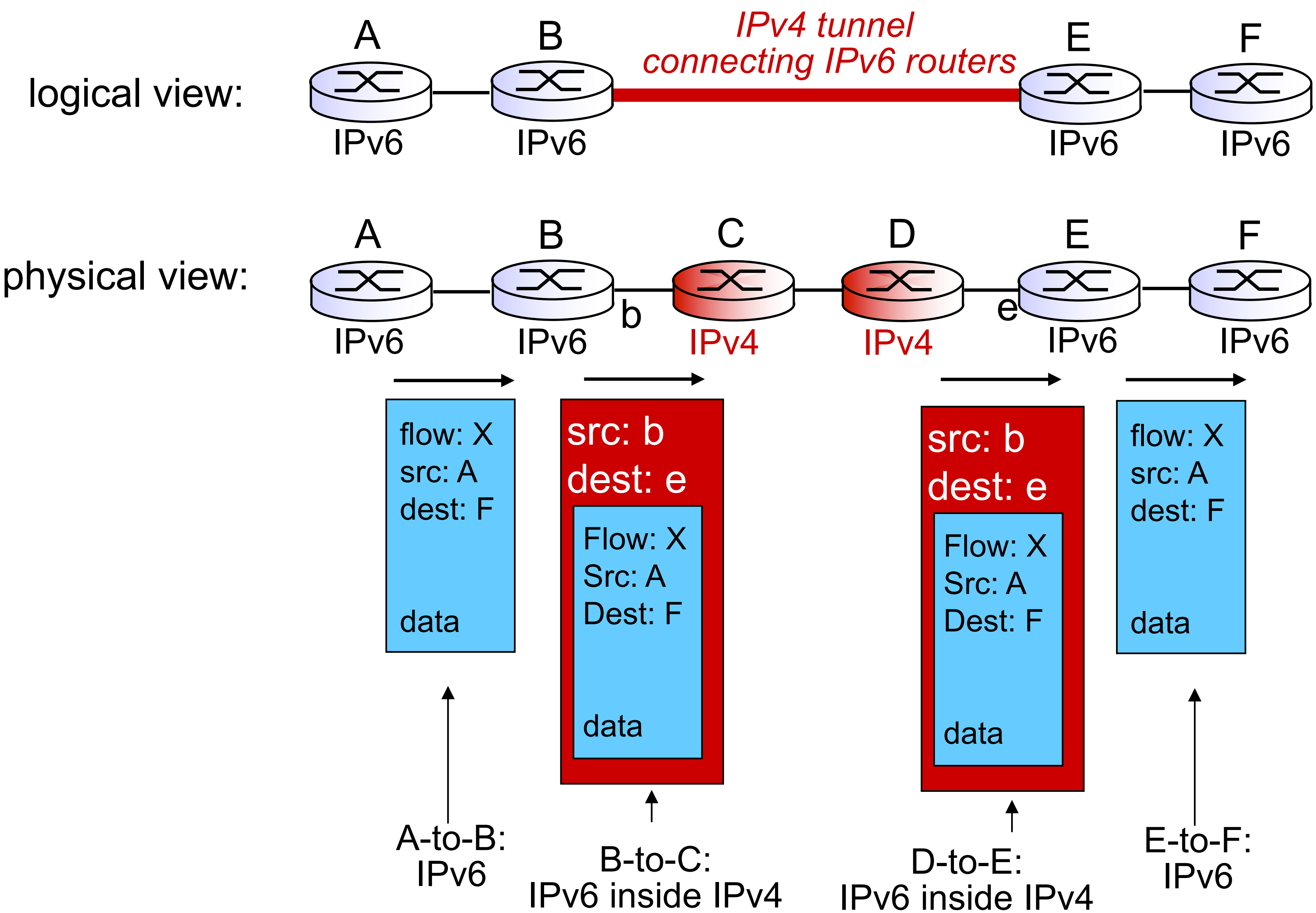


Tunneling



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


IPv4 tunnel
connecting IPv6 routers
logical view:
A
B
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
b
e
flow: X
src: A
dest: F
data
src: b
dest: e
Flow: X
Src: A
Dest: F
data
src: b
dest: e
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside IPv4
D-to-E:
IPv6 inside IPv4
E-to-F:
IPv6

IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Mobile, IoT:

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