Chapter 4
Network Layer:
Data Plane

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

Network layer: our goals

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - addressing
  - generalized forwarding
  - Internet architecture

- instantiation, implementation in the Internet
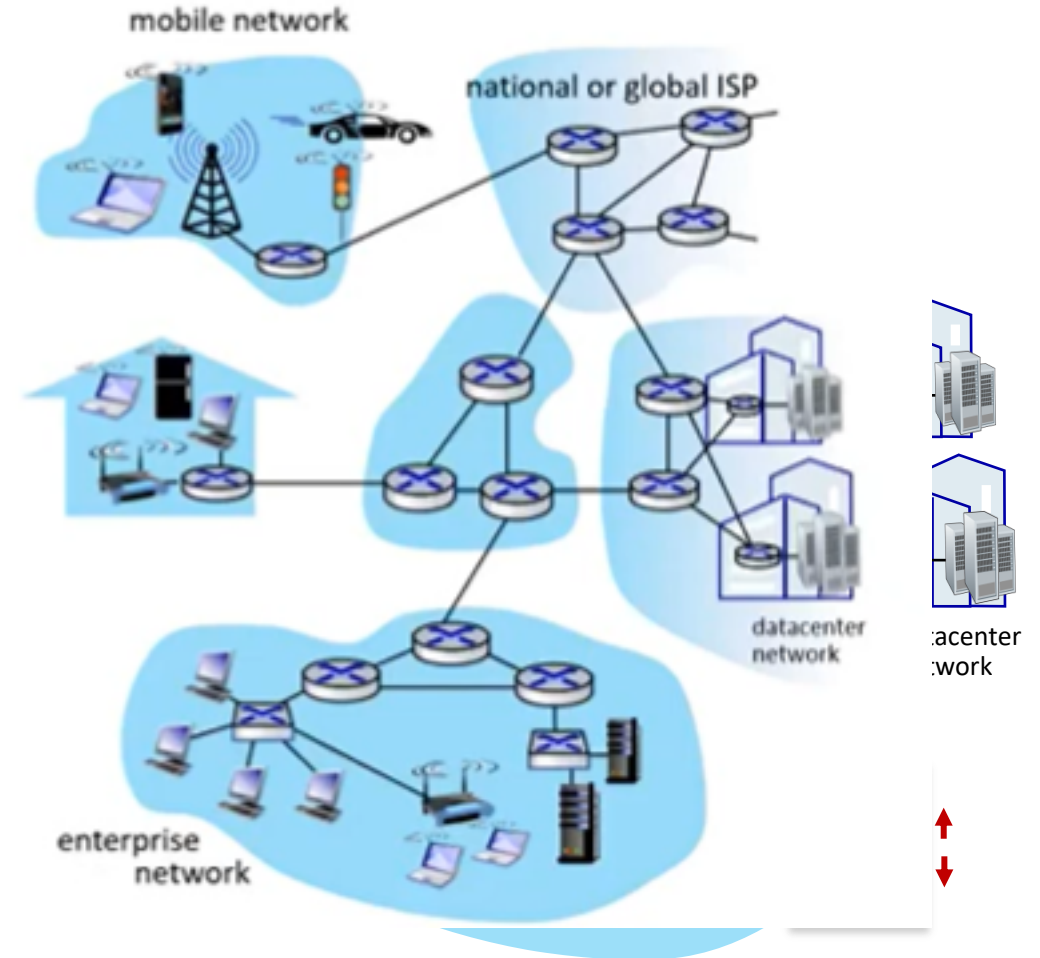 - IP protocol
  - NAT, middleboxes

Network layer: "data plane" roadmap

- **Network layer: overview**
  - data plane
  - control plane

- **What's inside a router**
  - input ports, switching, output ports
  - buffer management, scheduling

- **IP: the Internet Protocol**
  - datagram format
  - addressing
  - network address translation
  - IPv6

- **Generalized Forwarding, SDN**
  - Match+action
  - OpenFlow: match+action in action

- **Middleboxes**

Network-layer services and protocols

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

mobile network

national or global ISP

datacenter network

datacenter network

enterprise network

Two key network-layer functions

network-layer functions:

- *forwarding:* move packets from a router's input link to appropriate router output link

- *routing:* determine route taken by packets from source to destination

  - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange

- *routing:* process of planning trip from source to destination

forwarding

routing

Network layer: data plane, control plane

Data plane:

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

values in arriving
packet header

0111

1
2
3

Control plane

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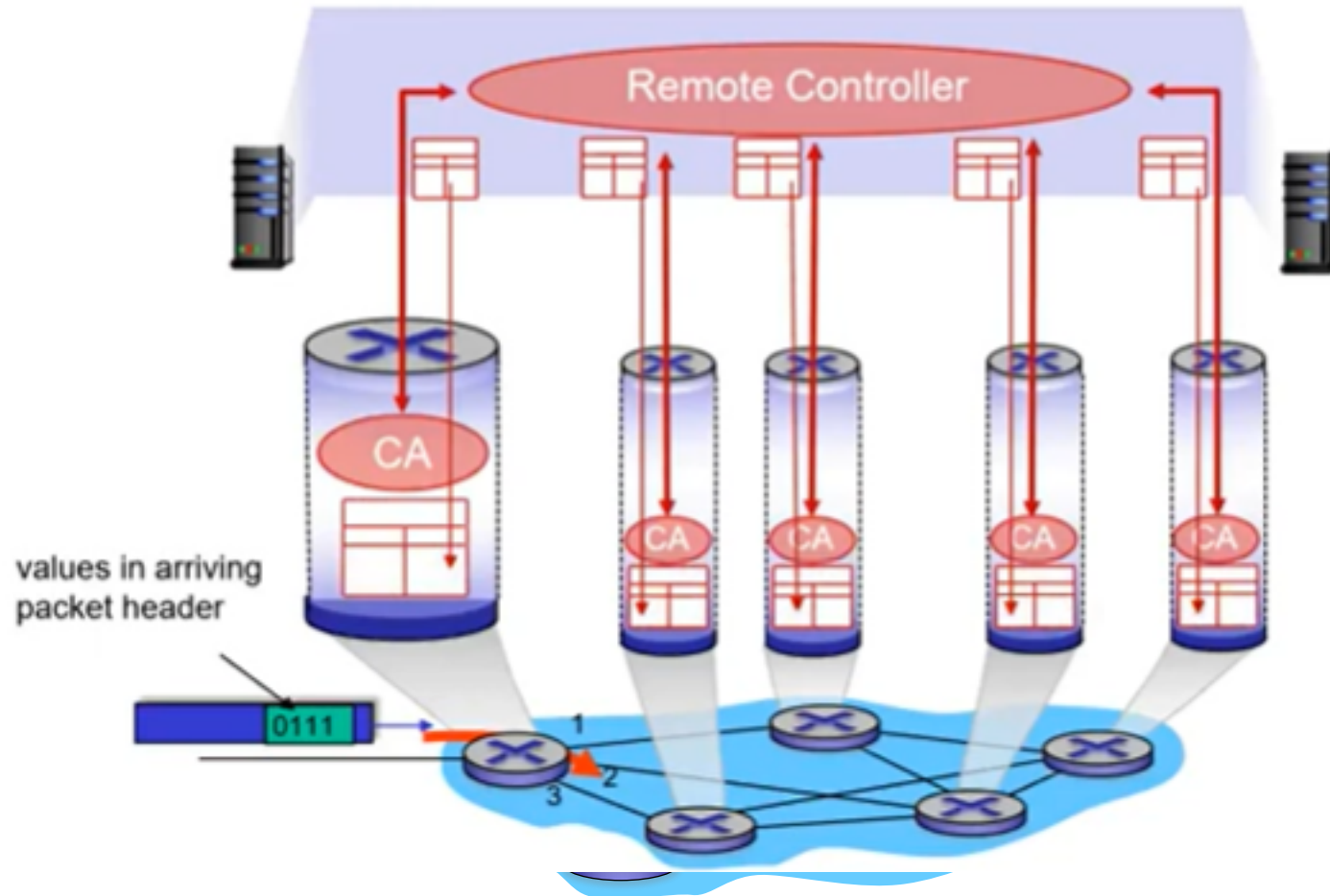 - *software-defined networking (SDN)*: implemented in (remote) servers

Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane
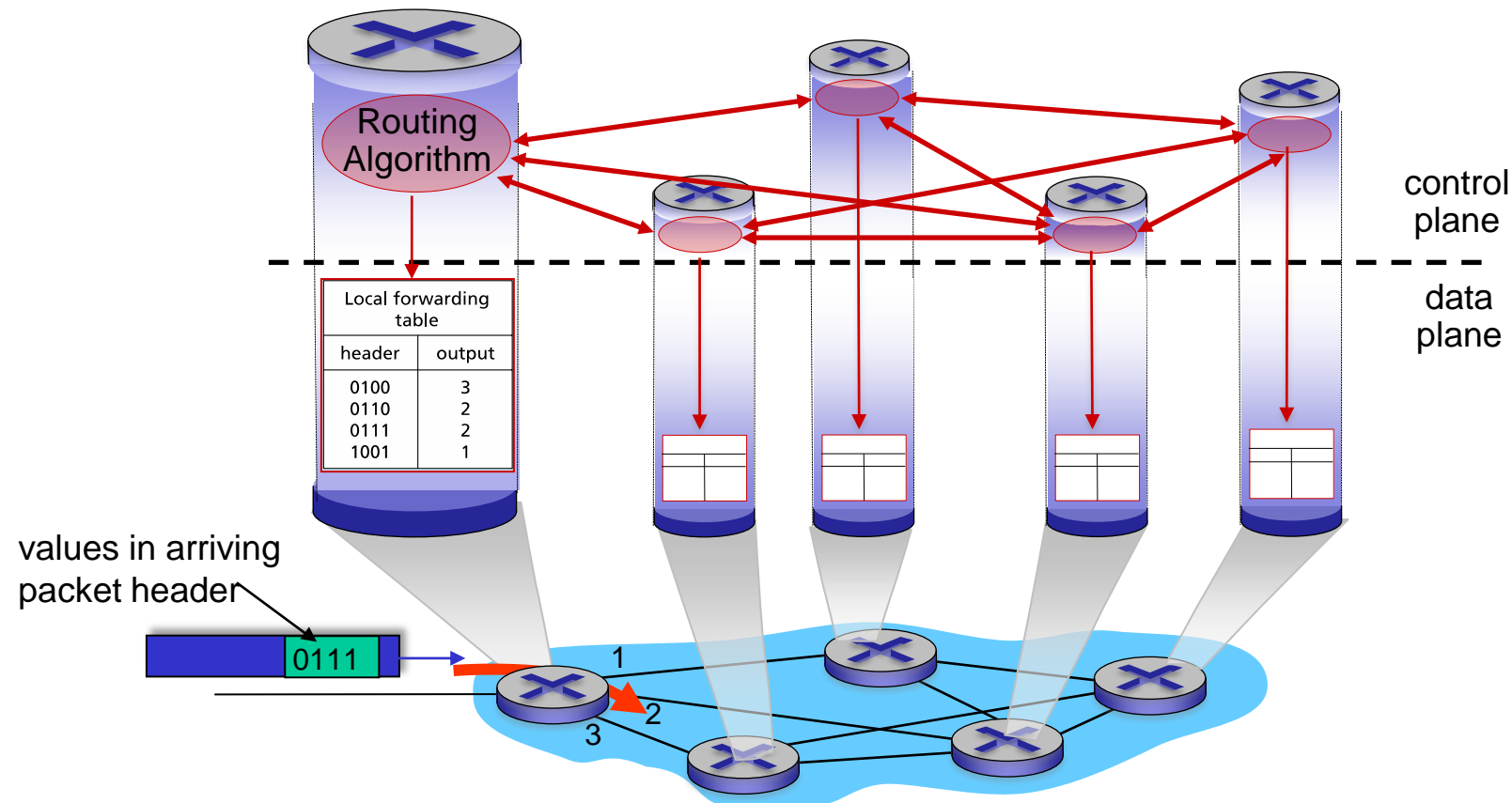
Software-Defined Networking (SDN) control plane
Remote controller computes, installs forwarding tables in routers

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for *individual* datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on*:*
- i. successful datagram delivery to destination
- ii. timing or order of delivery
- iii. bandwidth available to end-end flow

Network-layer service model

Quality of Service (QoS) Guarantees ?

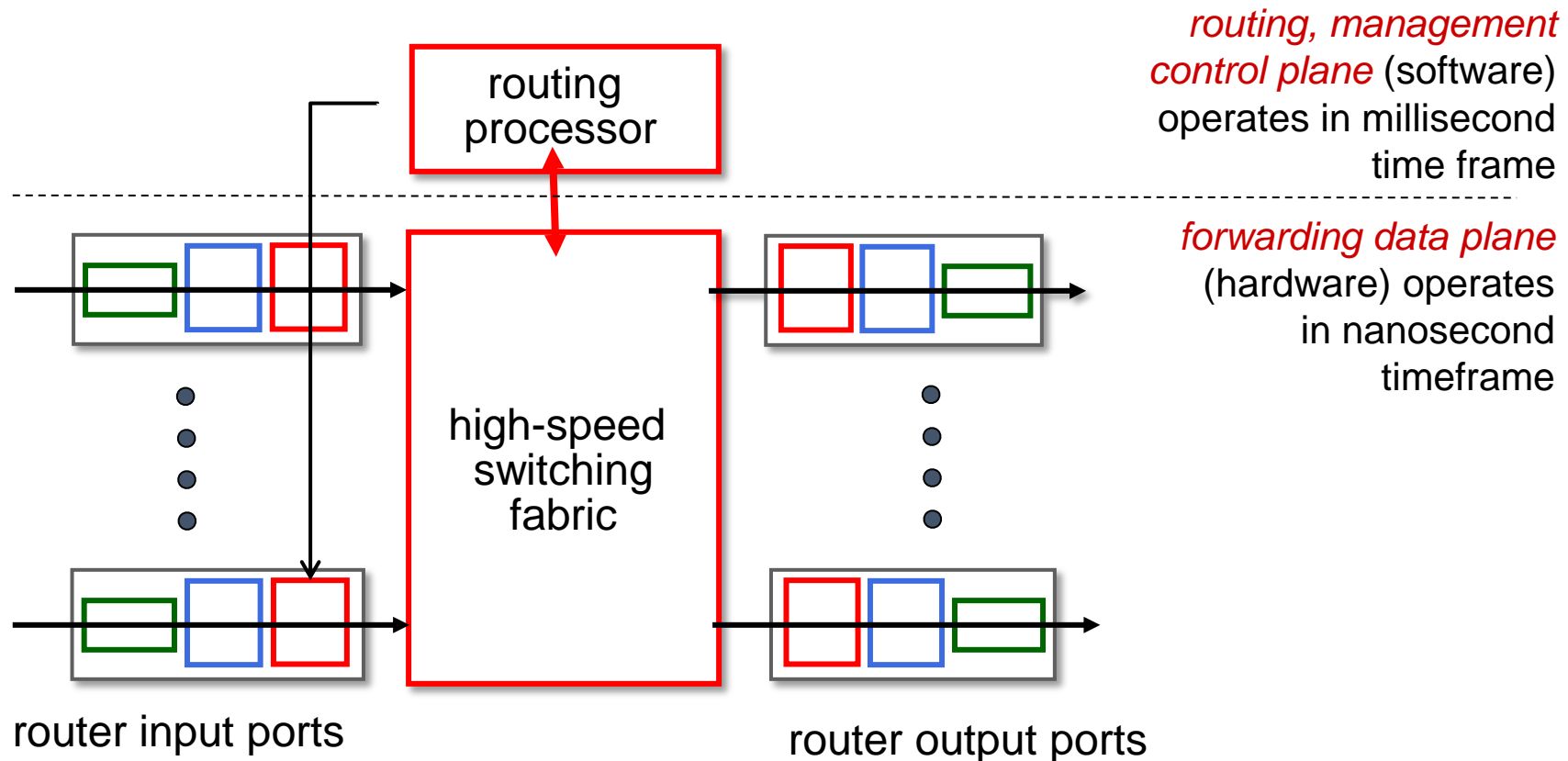| Network Architecture	Service Model	Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

Network layer: "data plane" roadmap

- Network layer: overview
  - data plane
  - control plane

- **What's inside a router**
  - input ports, switching, output ports
  - buffer management, scheduling

- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - Match+action
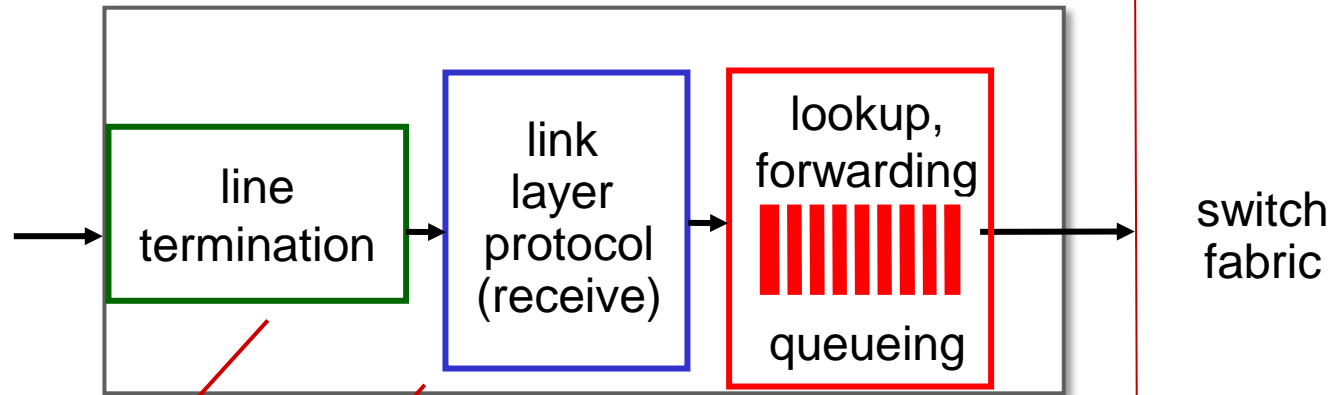 - OpenFlow: match+action in action

- Middleboxes

Router architecture overview

high-level view of generic router architecture:

routing processor

high-speed switching fabric

router input ports

router output ports

routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

Input port functions

physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- input port queuing: if datagrams arrive faster than forwarding rate into switch fabric
- destination-based forwarding: forward based only on destination IP address (traditional)

Destination-based forwarding

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 match

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:

11001000 00010111 00010110 10100001   which interface?

11001000 00010111 00011000 10101010   which interface?

Longest prefix matching

longest prefix match

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

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match ──────────────

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

match!

examples:

11001000   00010111   00010110   10100001   which interface?

11001000   00010111   00011000   10101010   which interface?

Longest prefix matching

longest prefix match

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

match!

examples:

11001000 00010111 00010110 10100001   which interface?

11001000 00010111 00011000 10101010   which interface?

Switching fabrics

- transfer packet from input link to appropriate output link
- 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
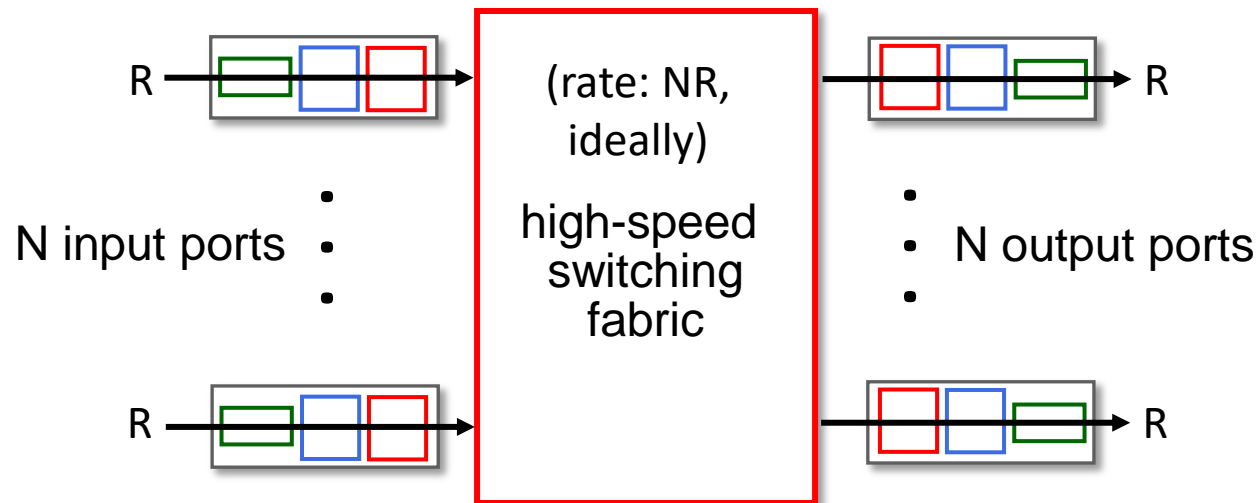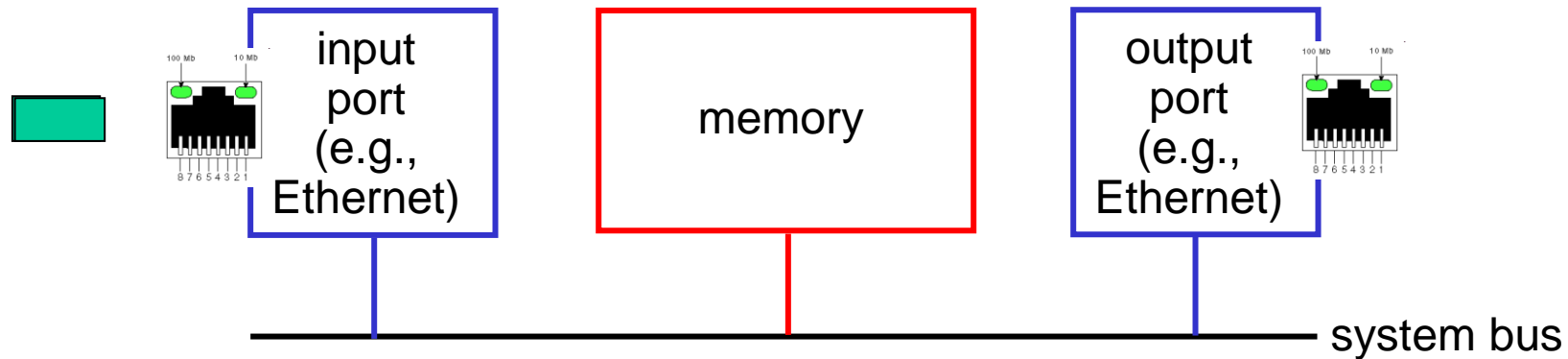
Switching fabrics

- three major types of switching fabrics:

memory

bus
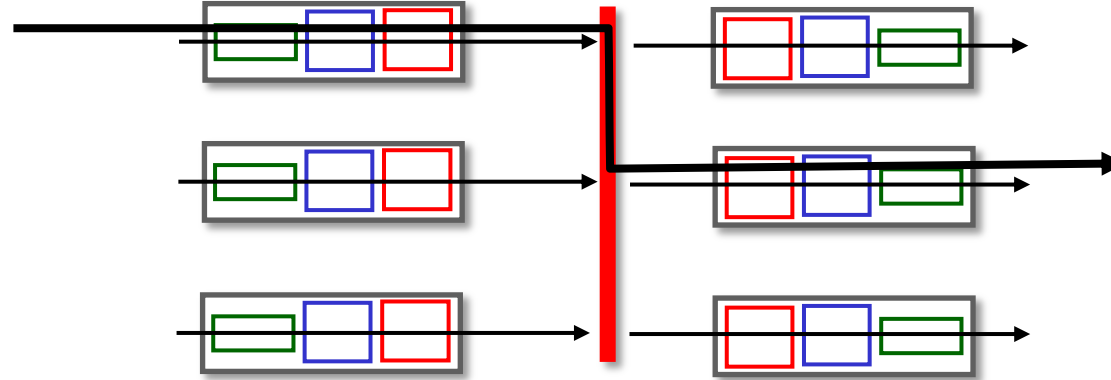
interconnection network
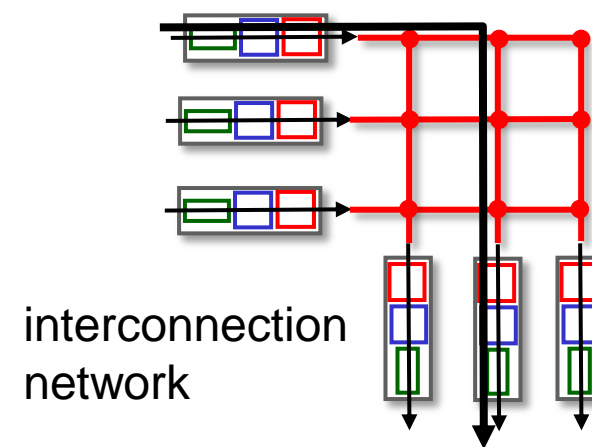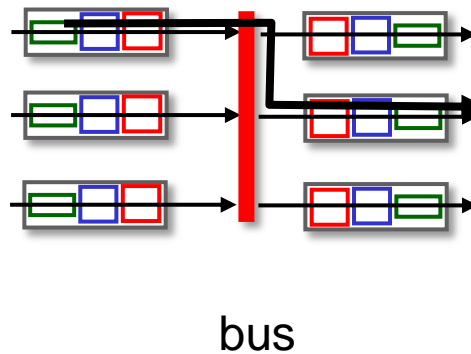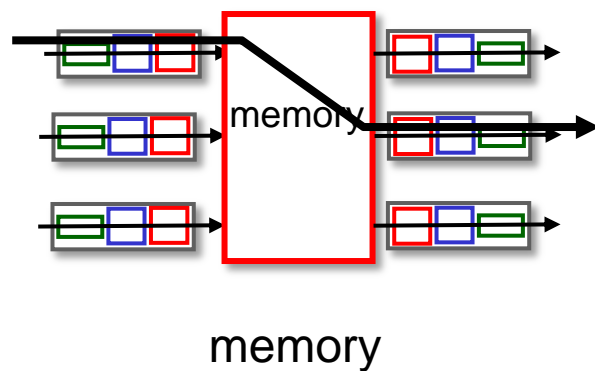
Switching via memory

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)

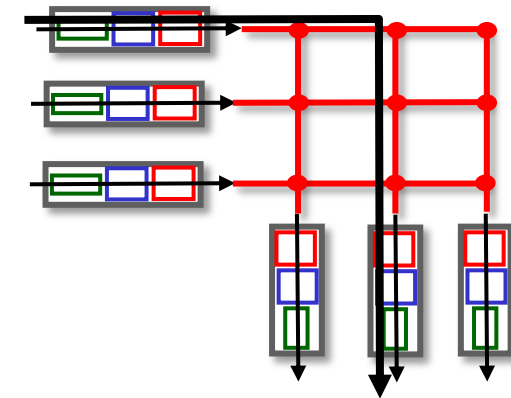# Switching via a bus

- datagram from input port memory to output port memory via a shared bus

- *bus contention:* switching speed limited by bus bandwidth
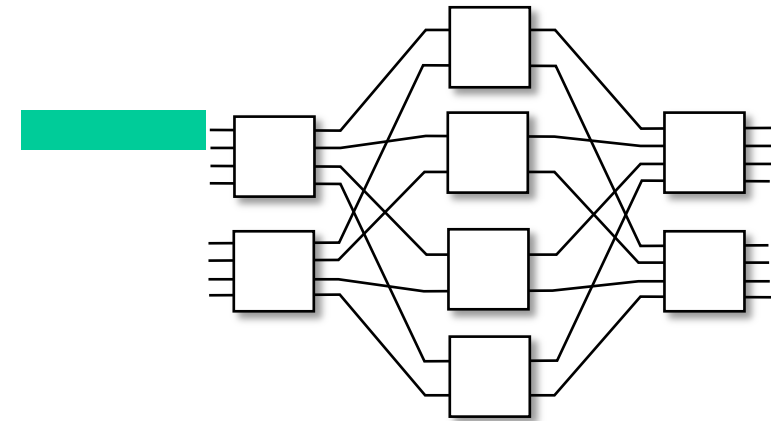
Switching via interconnection network

- Crossbar, Clos networks

- multistage switch: *nxn* switch from multiple stages of smaller switches

- exploiting parallelism:

3x3 crossbar

8x8 multistage switch
built from smaller-sized switches

Output port queuing

switch
fabric
(rate: NR)

| datagram buffer | link layer protocol (send) | line termination |

queueing

R
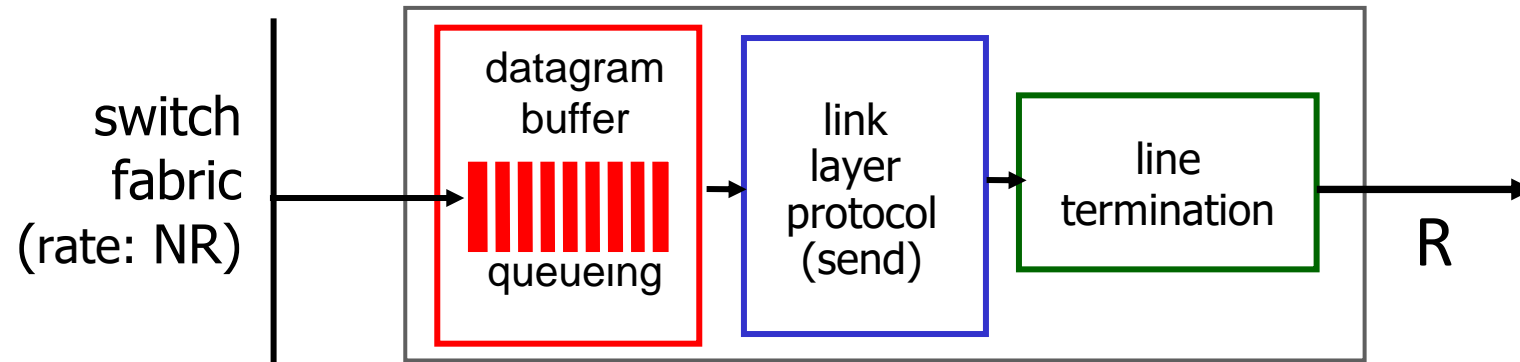
This is a really important slide

- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

Output port queuing

at *t,* packets more
from input to output

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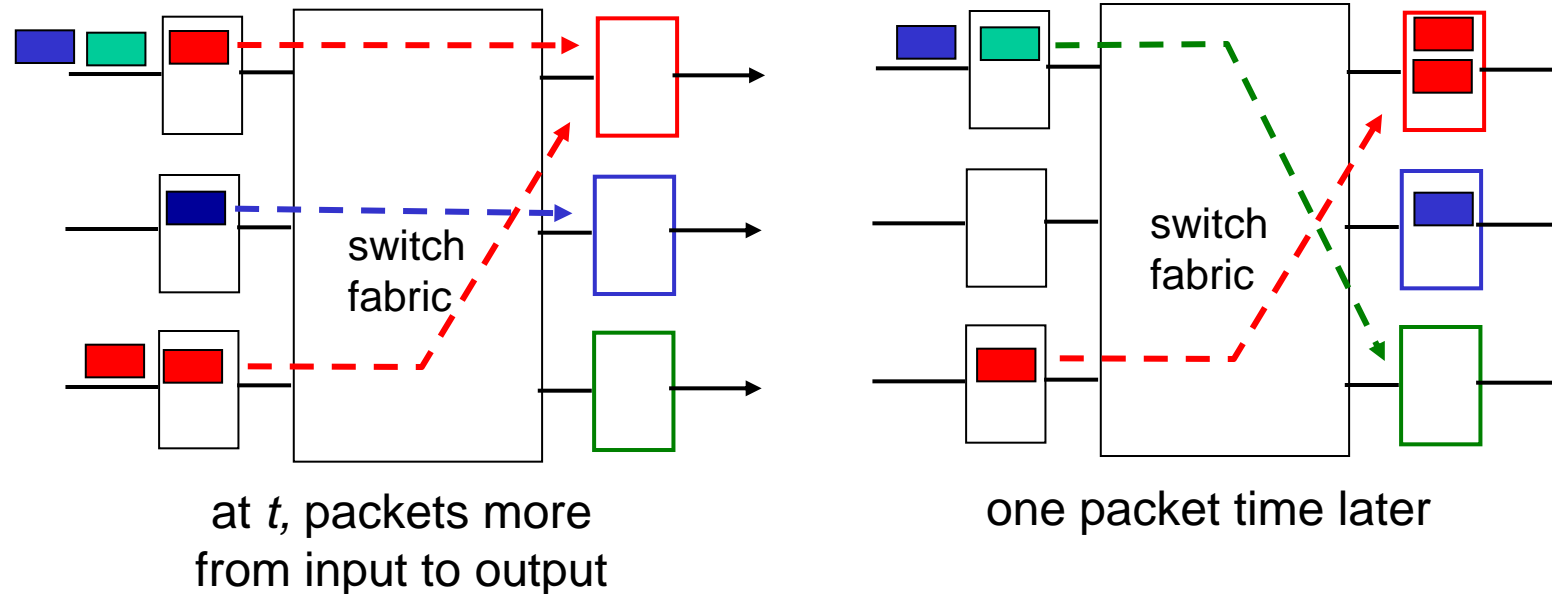
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Buffer Management

switch fabric → datagram buffer / queueing scheduling → link layer protocol (send) → line termination → R
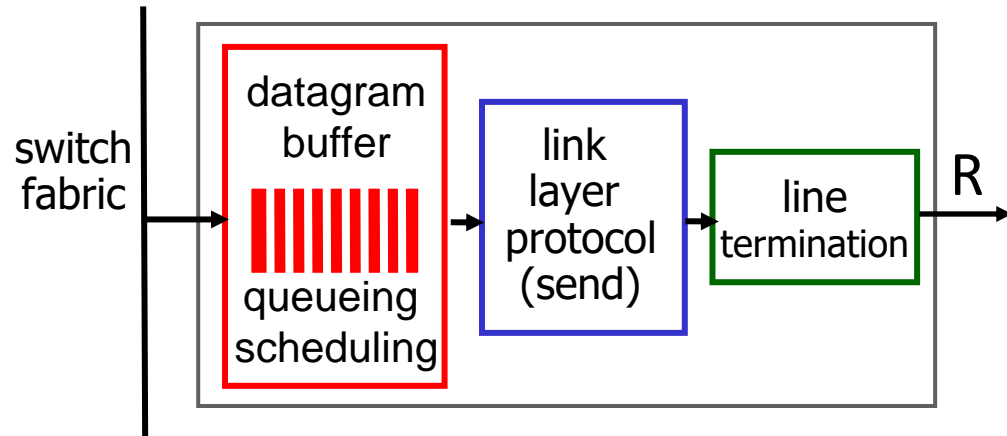
buffer management:

- **drop:** which packet to add, drop when buffers are full
  - **tail drop:** drop arriving packet
  - **priority:** drop/remove on priority basis

- **marking:** which packets to mark to signal congestion (ECN, RED)

Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link
- first come, first served
- priority
- round robin
- weighted fair queueing

FCFS: packets transmitted in order of arrival to output port
- also known as: First-in-first-out (FIFO)
- real world examples?

Scheduling policies: priority

Priority scheduling:

- **arriving traffic classified, queued by class**
  - any header fields can be used for classification

- **send packet from highest priority queue that has buffered packets**
  - FCFS within priority class
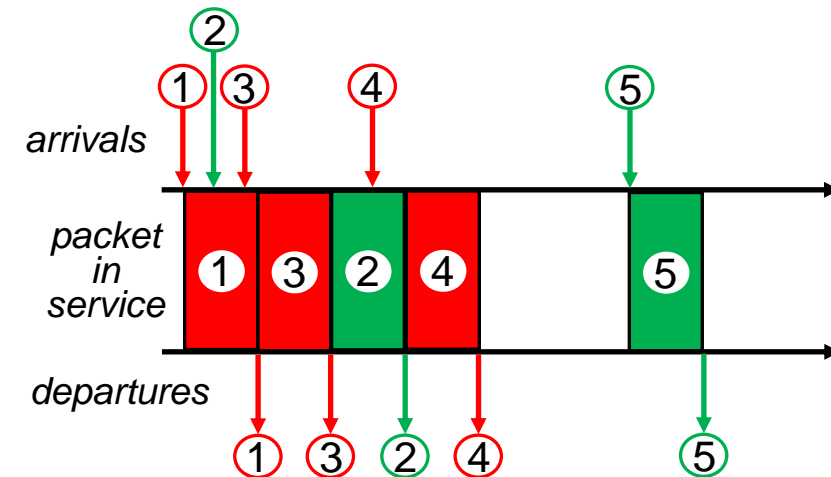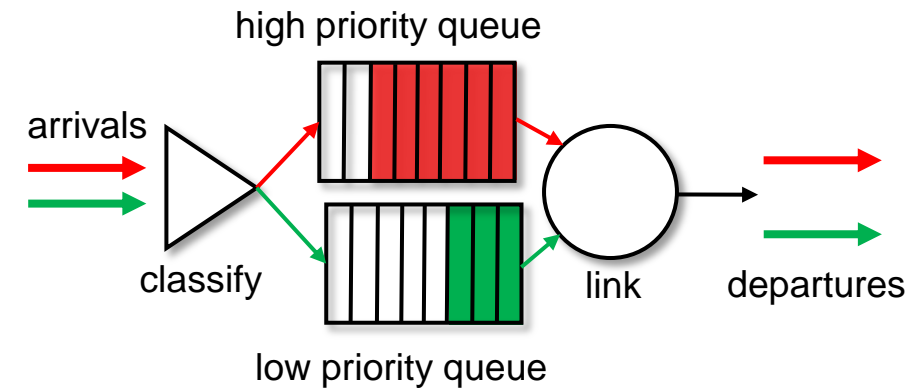
Scheduling policies: round robin

Round Robin (RR) scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification

- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn

classify arrivals

R

link   departures
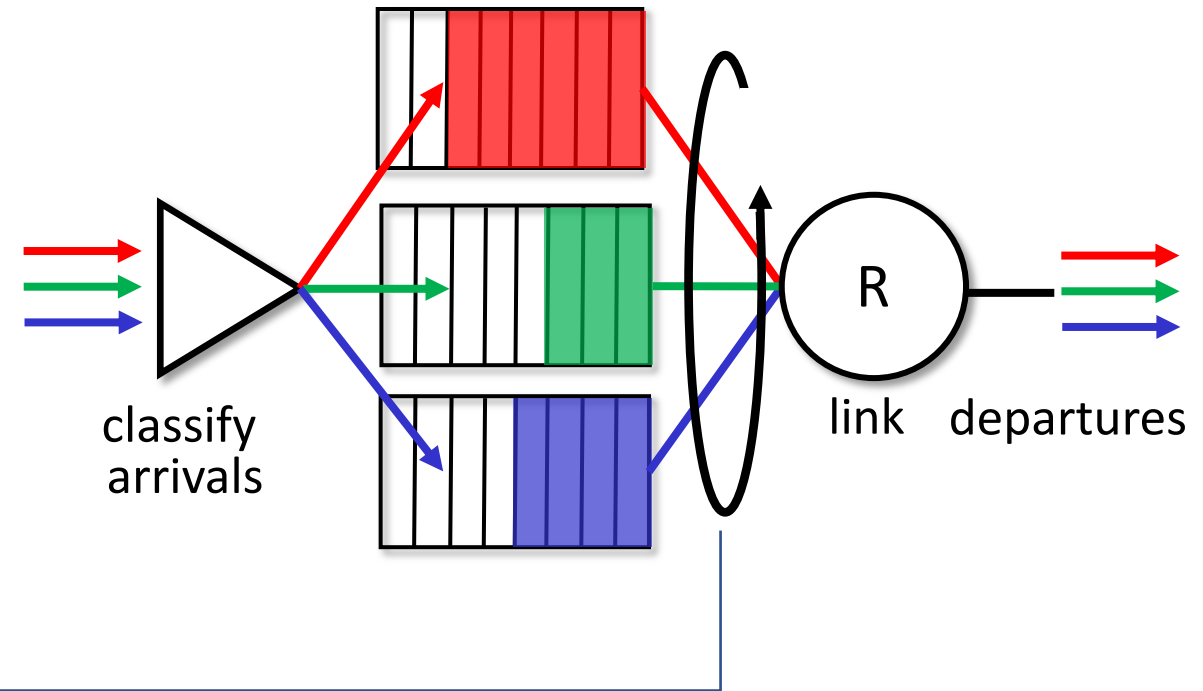
Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class, *i,* has weight, w_i, and gets weighted amount of service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$
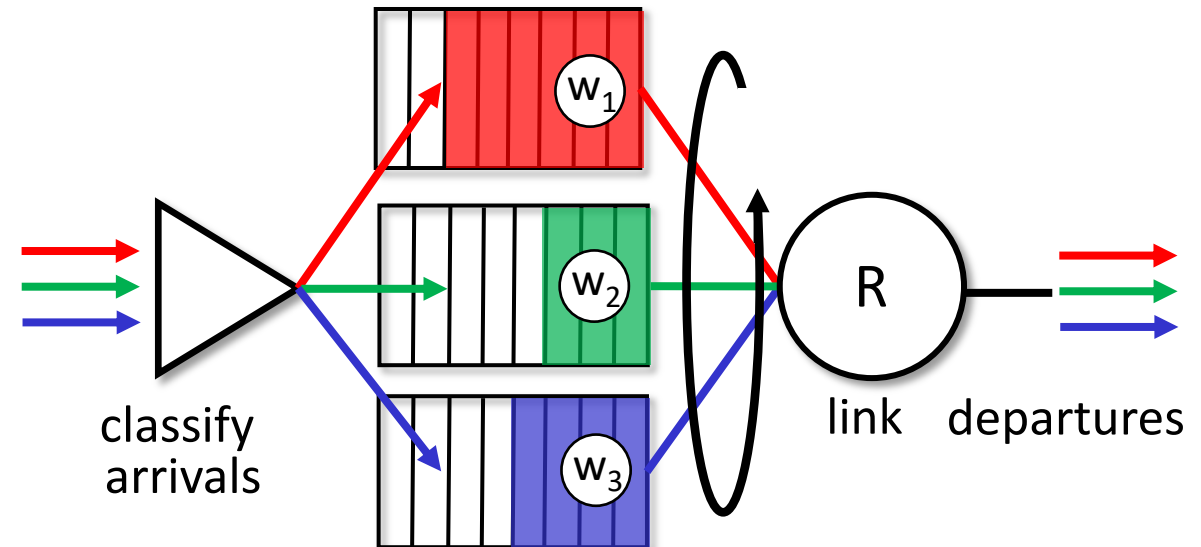
classify arrivals

w_1

w_2

w_3

R

link   departures

Network layer: "data plane" roadmap

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- **IP: the Internet Protocol**
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

Network Layer: Internet

host, router network layer functions:

network layer

transport layer: TCP, UDP

Path-selection algorithms:
implemented in
- routing protocols (OSPF, BGP)
- SDN controller

forwarding table

IP protocol
- datagram format
- addressing
- packet handling conventions

ICMP protocol
- error reporting
- router "signaling"

link layer

physical layer

IP Datagram format

32 bits

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

IP protocol version number

header length(bytes)

"type" of service:
- diffserv (0:5)
- ECN (6:7)

TTL: remaining max hops (decremented at each router)

upper layer protocol (e.g., TCP or UDP)

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

total datagram length (bytes)

fragmentation/ reassembly

header checksum

32-bit source IP address

Maximum length: 64K bytes
Typically: 1500 bytes or less

e.g., timestamp, record route taken

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 *destination*
  - IP header bits used to identify, order related fragments

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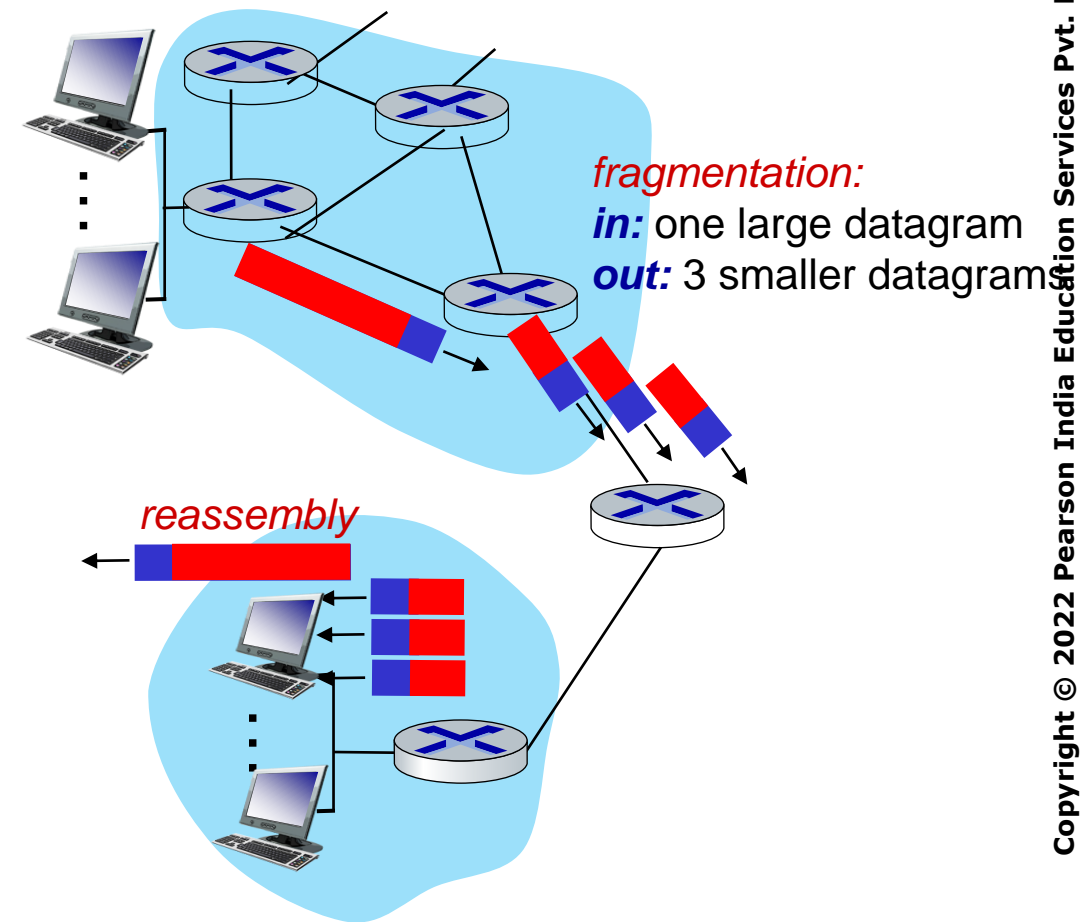
reassembly

IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

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

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 | |

IP addressing: introduction

- **IP address:** 32-bit identifier associated with each host or 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)

dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223   1   1   1

IP addressing: introduction

- **IP address:** 32-bit identifier associated with each host or 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)

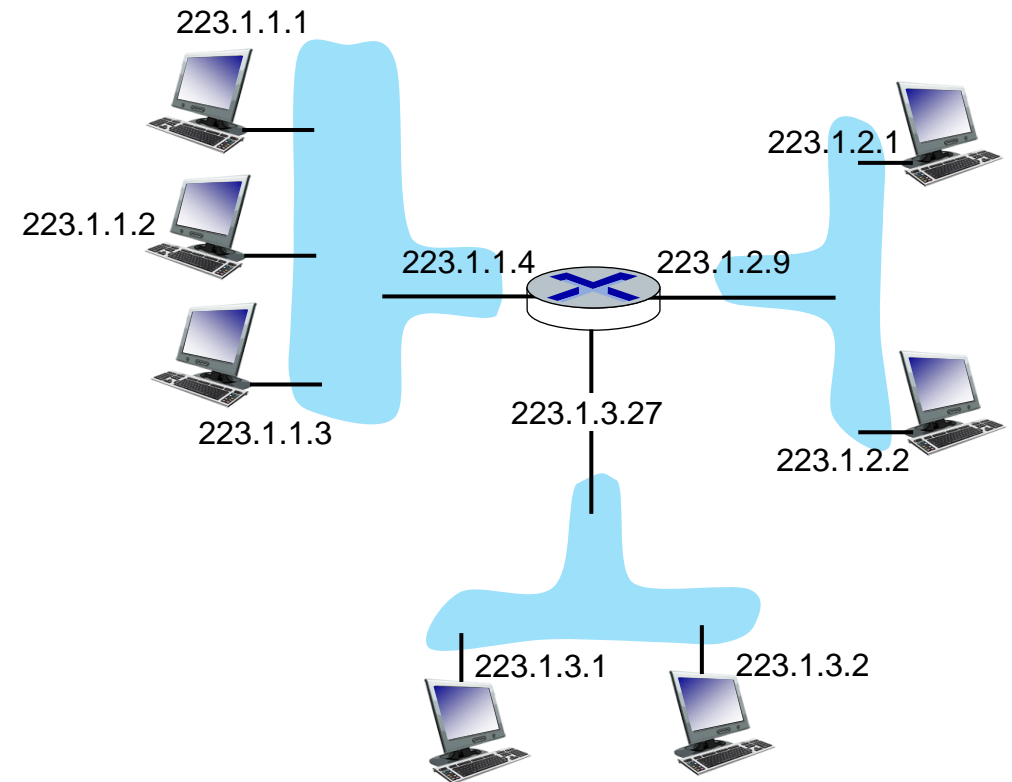223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.1.3

223.1.3.27

223.1.2.1

223.1.2.2

223.1.3.1   223.1.3.2

dotted-decimal IP address notation:

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 chapters 6, 7

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

223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.1.3

223.1.2.1

223.1.2.2

223.1.3.27

223.1.3.1   223.1.3.2

A: wired Ethernet interfaces connected by Ethernet switches
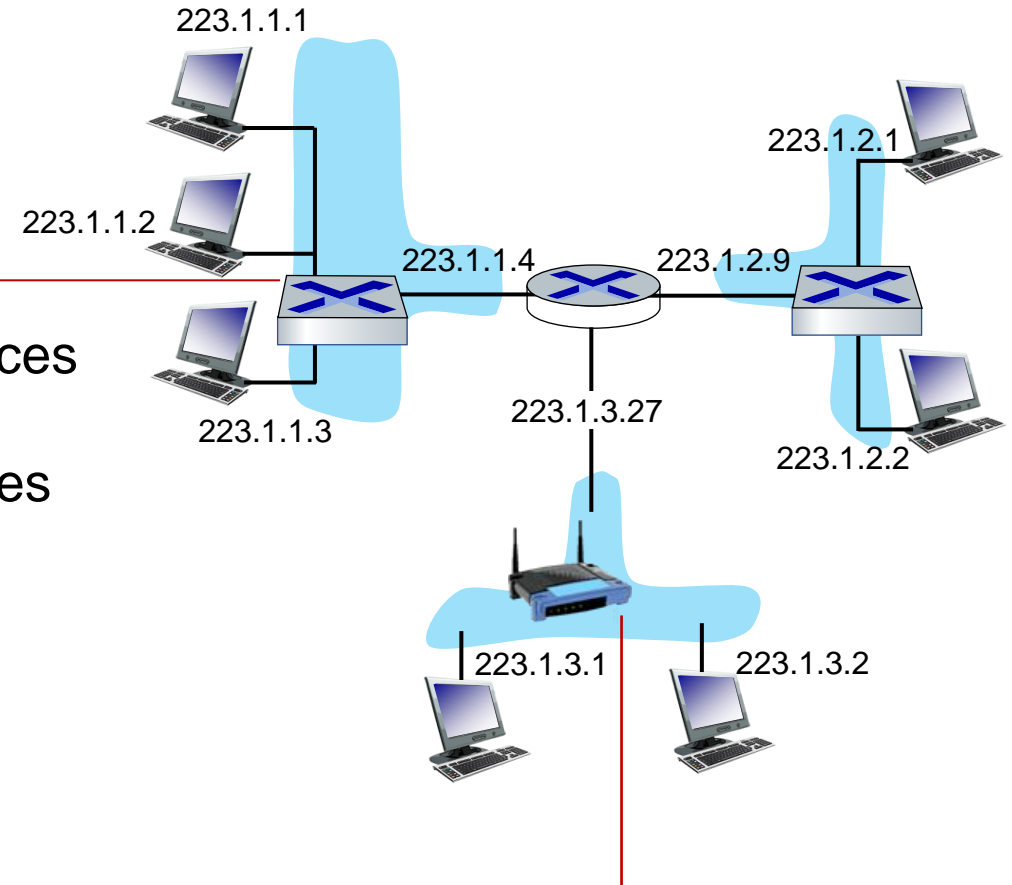
A: wireless WiFi interfaces connected by WiFi base station

Subnets

▪ *What's a subnet ?*

- device interfaces that can physically reach each other without passing through an intervening router

▪ IP addresses have structure:

- subnet part: devices in same subnet have common high order bits

- host part: remaining low order bits

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.1
223.1.2.9
223.1.2.2
223.1.3.27
223.1.3.1
223.1.3.2
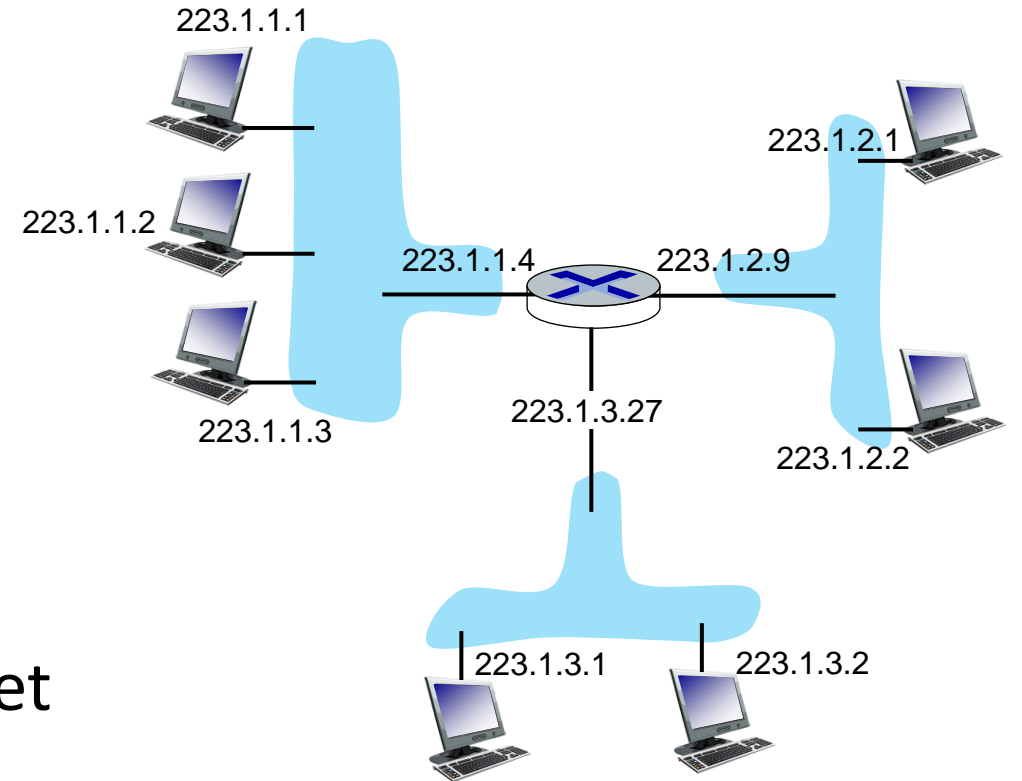
network consisting of 3 subnets

Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating "islands" of isolated networks

- each isolated network is called a *subnet*

subnet 223.1.1.0/24

223.1.1.1

223.1.1.2

223.1.1.3

subnet 223.1.2.0/24

223.1.2.1

223.1.1.4   223.1.2.9

223.1.3.27

223.1.2.2

subnet 223.1.3.0/24

223.1.3.1   223.1.3.2
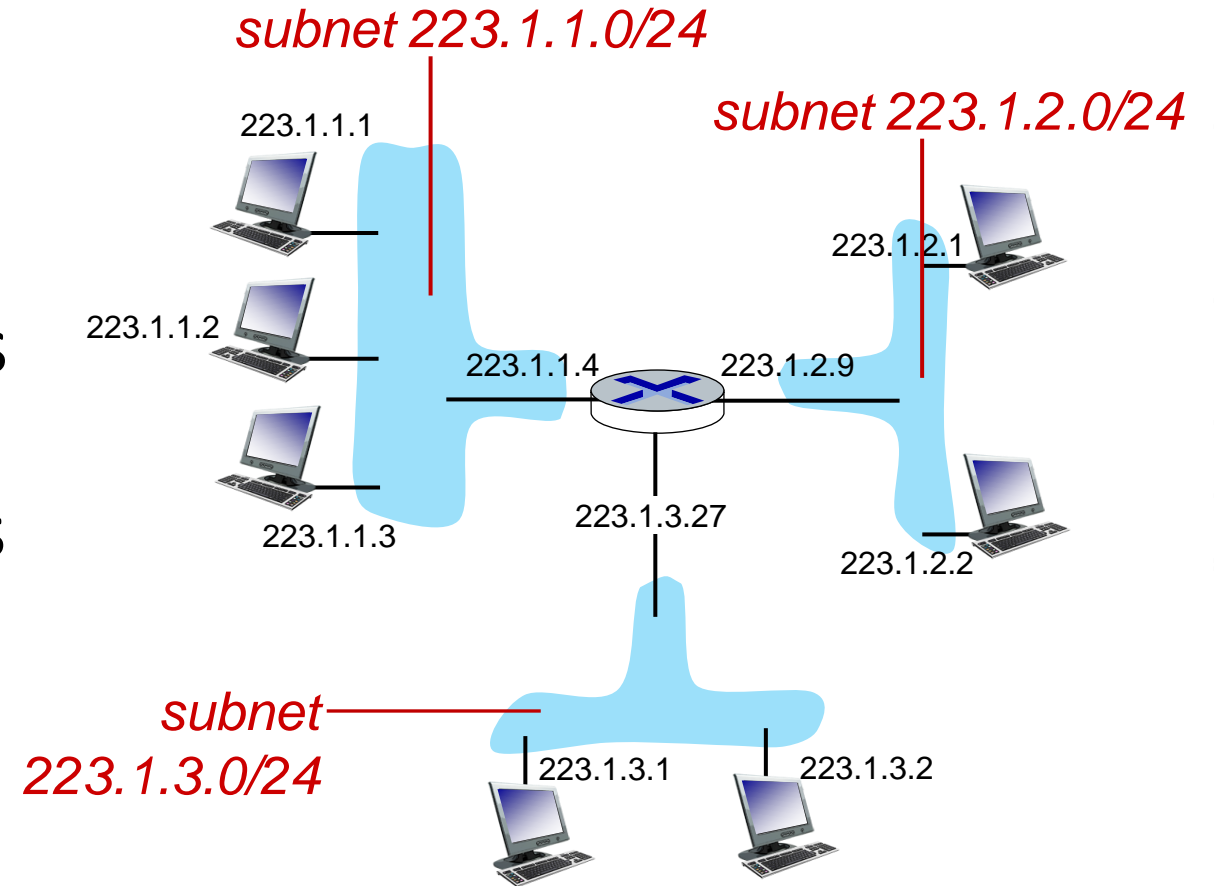
subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

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

223.1.1.2

subnet 223.1.1/24

223.1.1.1

223.1.1.4

223.1.1.3

223.1.9.2

223.1.7.0

subnet 223.1.7/24

subnet 223.1.9/24

223.1.9.1

223.1.8.1

223.1.8.0

223.1.7.1

223.1.2.6

subnet 223.1.8/24

223.1.3.27

subnet 223.1.2/24

subnet 223.1.3/24

223.1.2.1

223.1.2.2

223.1.3.1

223.1.3.2
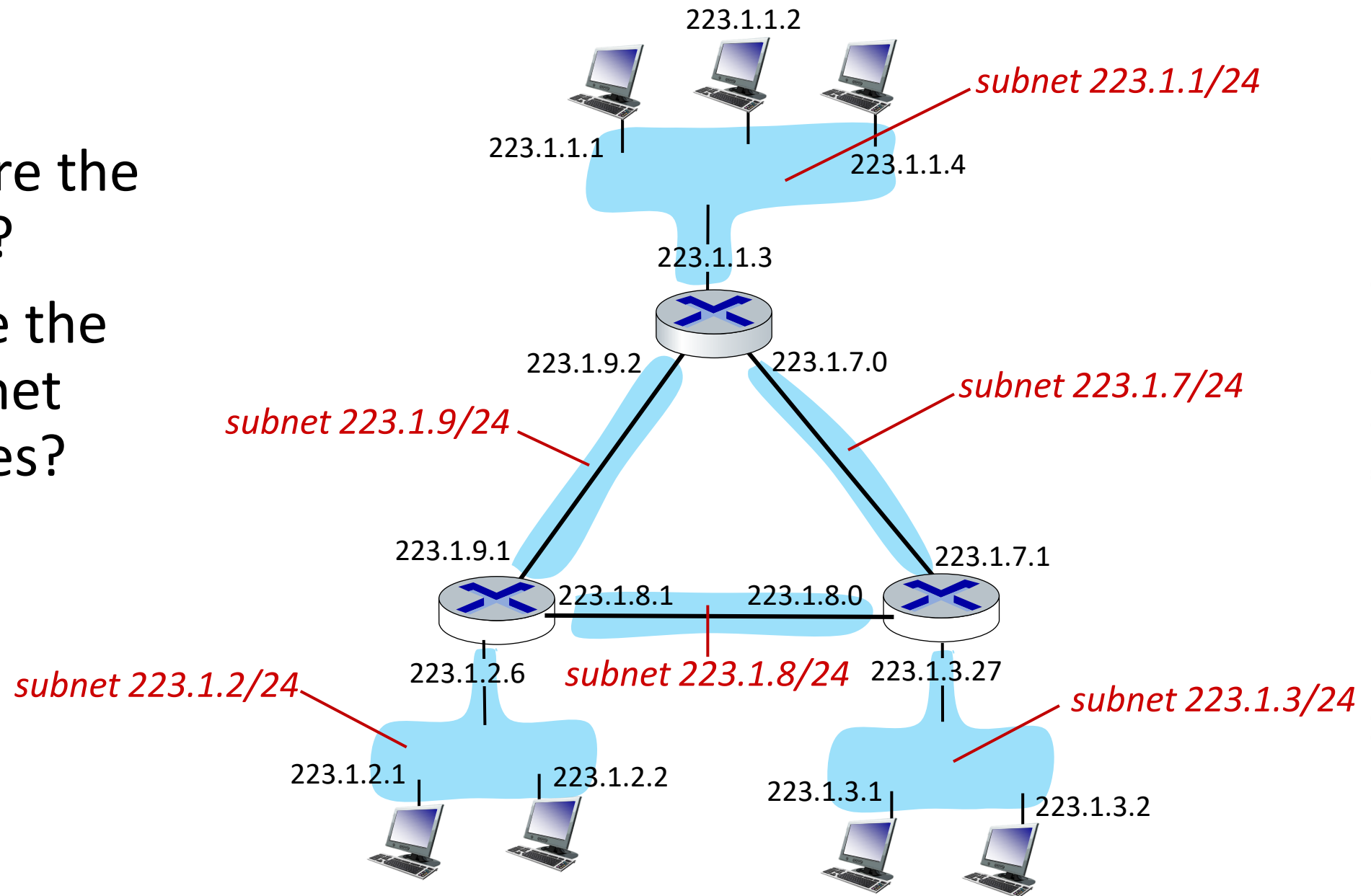
IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced "cider")
- 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

Pearson
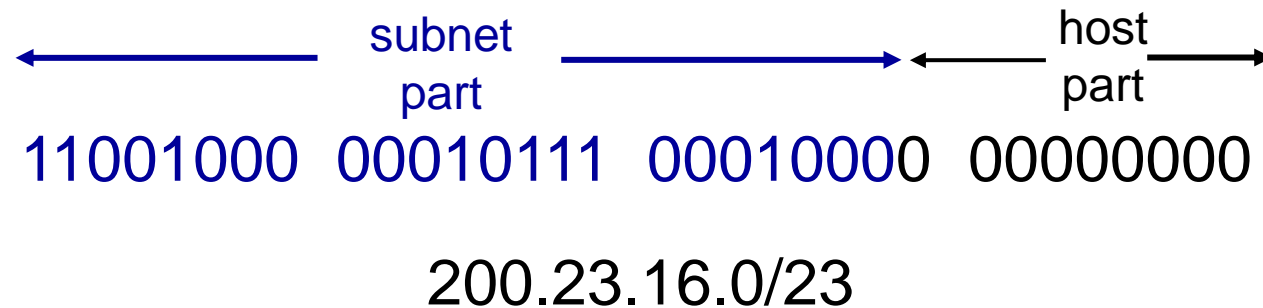
IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?

2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

goal: host *dynamically* obtains IP address from network server when it "joins" network
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:
- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

DHCP client-server scenario
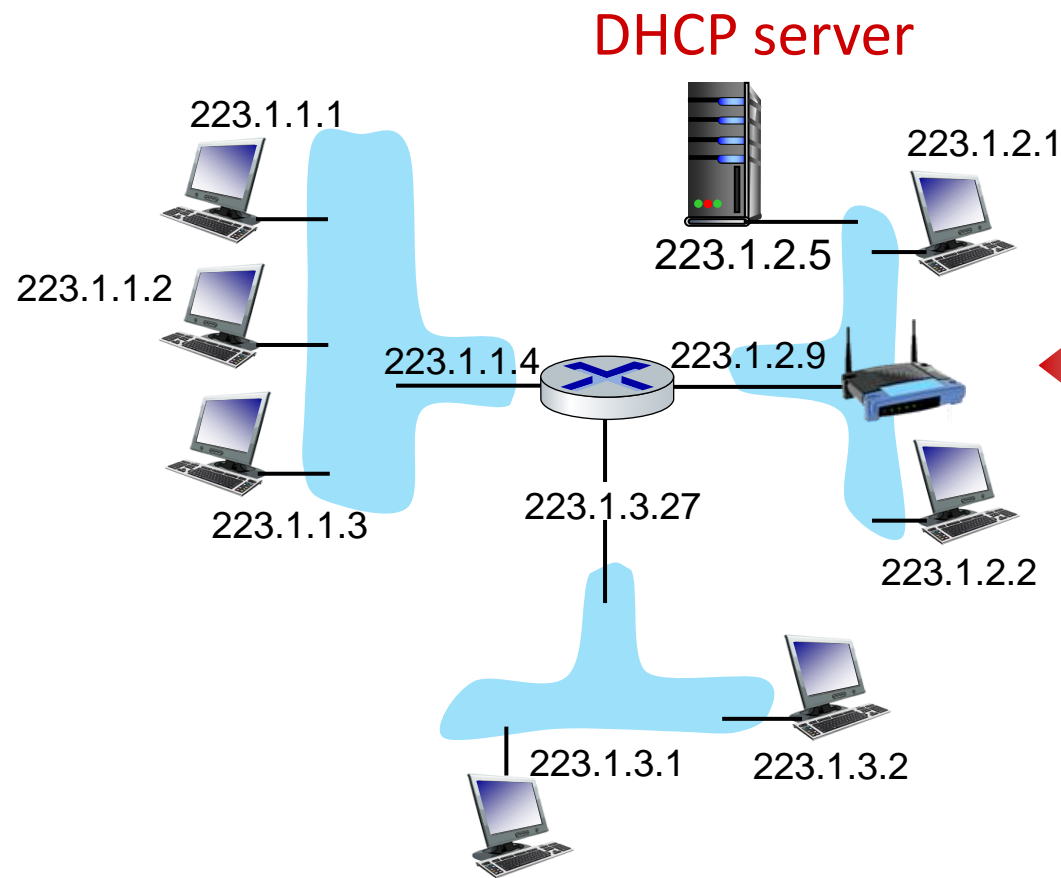
Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

DHCP server

223.1.1.1

223.1.2.1

223.1.2.5

223.1.1.2

223.1.1.4

223.1.2.9

223.1.1.3

223.1.3.27

223.1.2.2

arriving DHCP client needs address in this network

223.1.3.1

223.1.3.2

DHCP client-server scenario

DHCP server: 223.1.2.5

Arriving client

DHCP discover

Broadcast: is there a DHCP server out there?

DHCP offer

Broadcast: I'm a DHCP server! Here's an IP address you can use

DHCP request

Broadcast: OK. I would like to use this IP address!

DHCP ACK

Broadcast: OK. You've got that IP address!

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

DHCP: more than IP addresses

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

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

IP addresses: how to get one?

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

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

ISP's block   <u>11001000 00010111 0001</u>0000 00000000   200.23.16.0/20

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

Organization 0   <u>11001000 00010111 0001000</u>0 00000000   200.23.16.0/23
Organization 1   <u>11001000 00010111 0001001</u>0 00000000   200.23.18.0/23
Organization 2   <u>11001000 00010111 0001010</u>0 00000000   200.23.20.0/23
    ...   …..   ….   ….
Organization 7   <u>11001000 00010111 0001111</u>0 00000000   200.23.30.0/23

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:
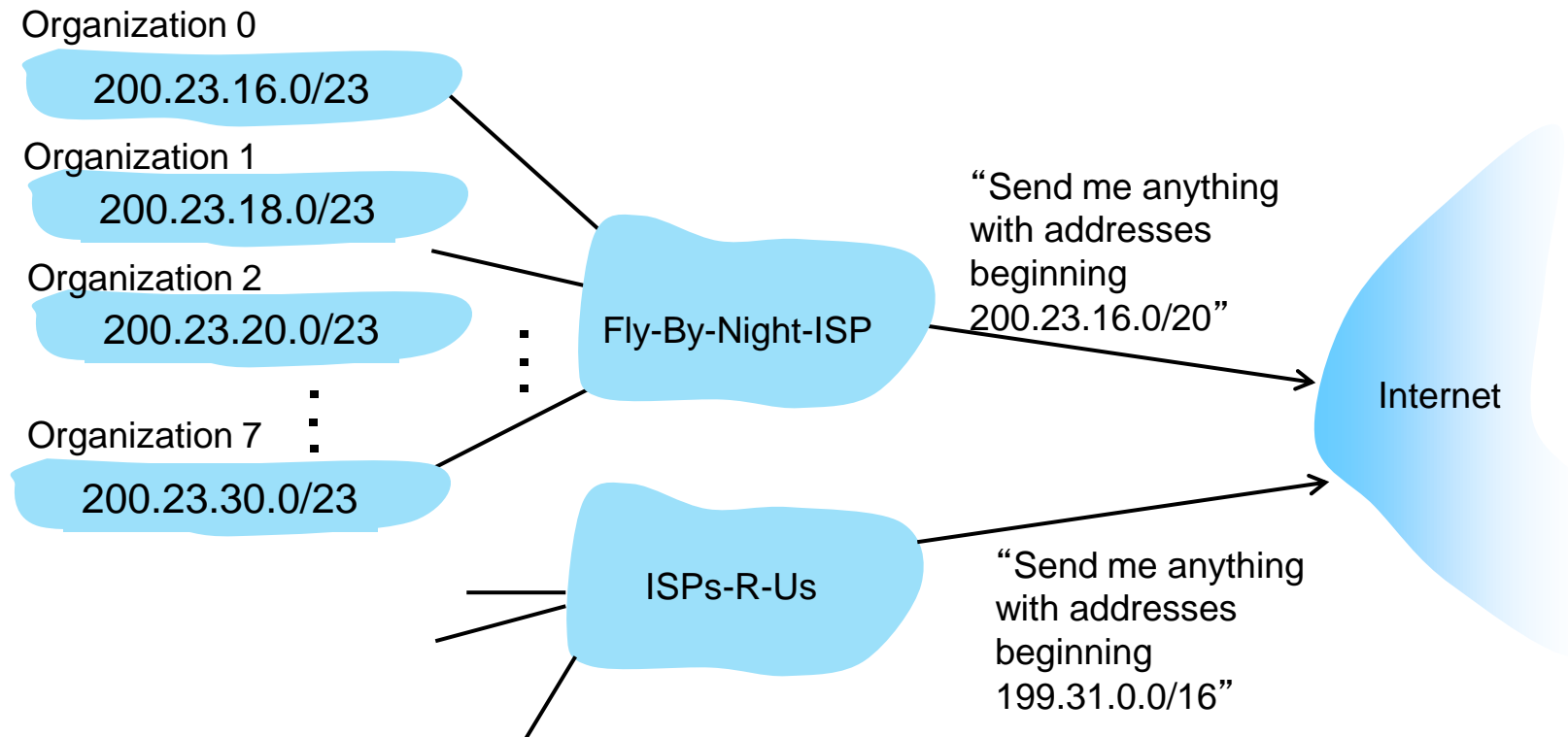
IP addressing: last words …

Q: how does an ISP get block of addresses?

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

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)

- manages DNS root zone, including delegation of individual TLD (.com, .edu , …) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011

- NAT (next) helps IPv4 address space exhaustion

- IPv6 has 128-bit address space

# Network layer: "data plane" roadmap

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- **IP: the Internet Protocol**
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

NAT: network address translation

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

rest of Internet

local network (e.g., home network) 10.0.0/24

138.76.29.7

10.0.0.4

10.0.0.1
10.0.0.2
10.0.0.3
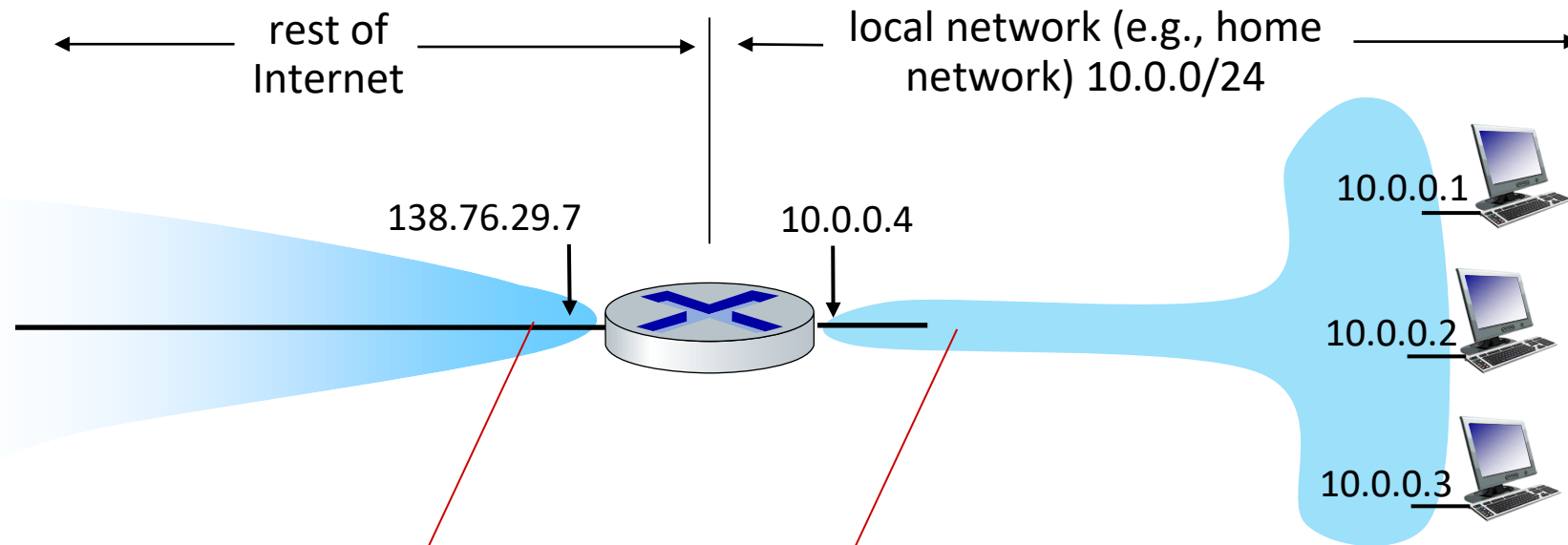
all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

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

NAT: network address translation

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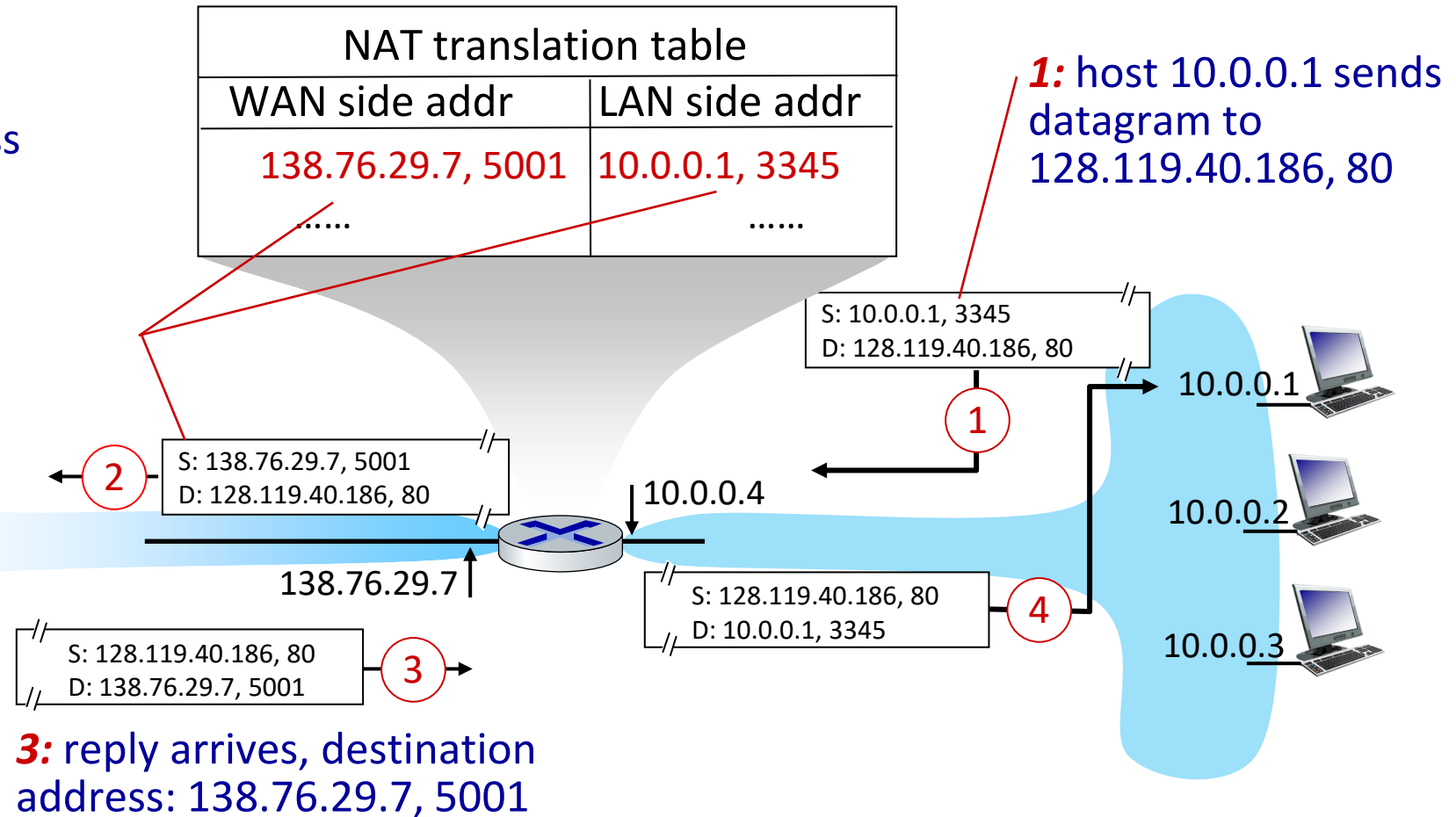
NAT: network address translation

implementation: NAT router must (transparently):

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

- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

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

NAT: network address translation

2: NAT router changes datagram source address 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

10.0.0.1

S: 138.76.29.7, 5001
D: 128.119.40.186, 80

②

10.0.0.4

10.0.0.2

138.76.29.7

S: 128.119.40.186, 80
D: 10.0.0.1, 3345

④

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

③

10.0.0.3

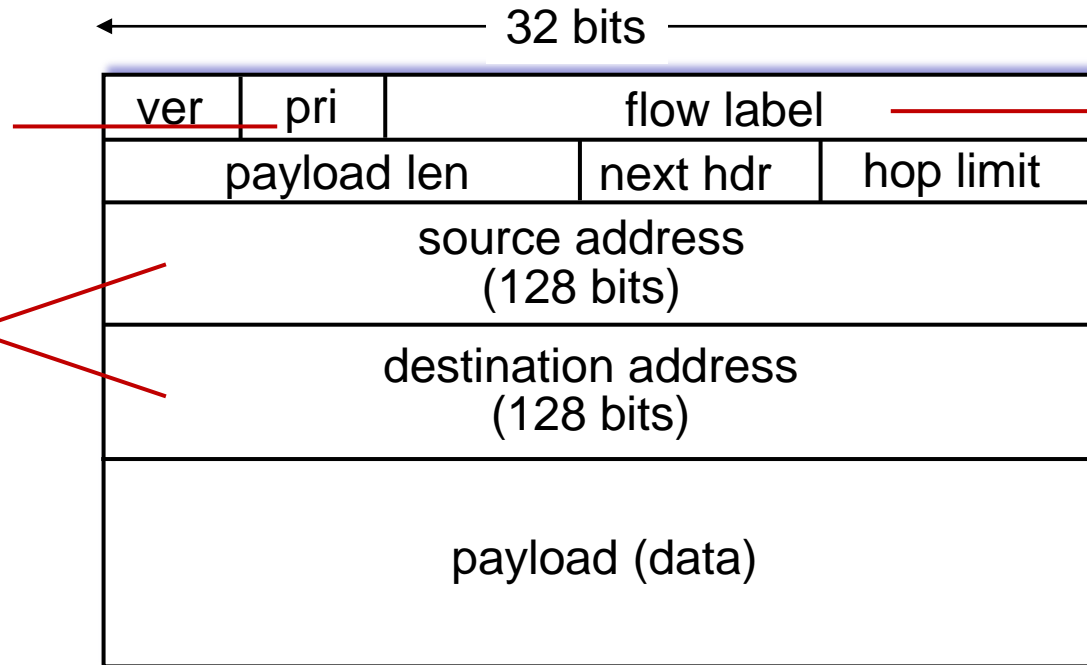**3:** reply arrives, destination address: 138.76.29.7, 5001

IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated

- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of "flows"

IPv6 datagram format

priority: identify priority among datagrams in flow
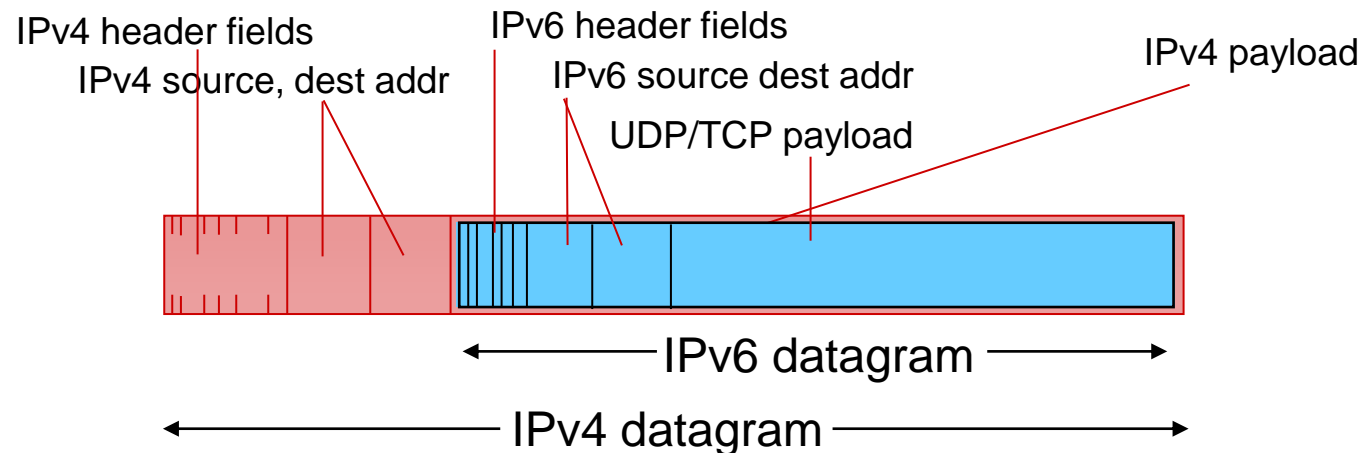
128-bit IPv6 addresses

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

32 bits				
ver	pri	flow label		
payload len		next hdr		hop limit
source address (128 bits)				
destination address (128 bits)				
payload (data)				

What's missing (compared with IPv4):
- no checksum (to speed processing at routers)
- no fragmentation/reassembly
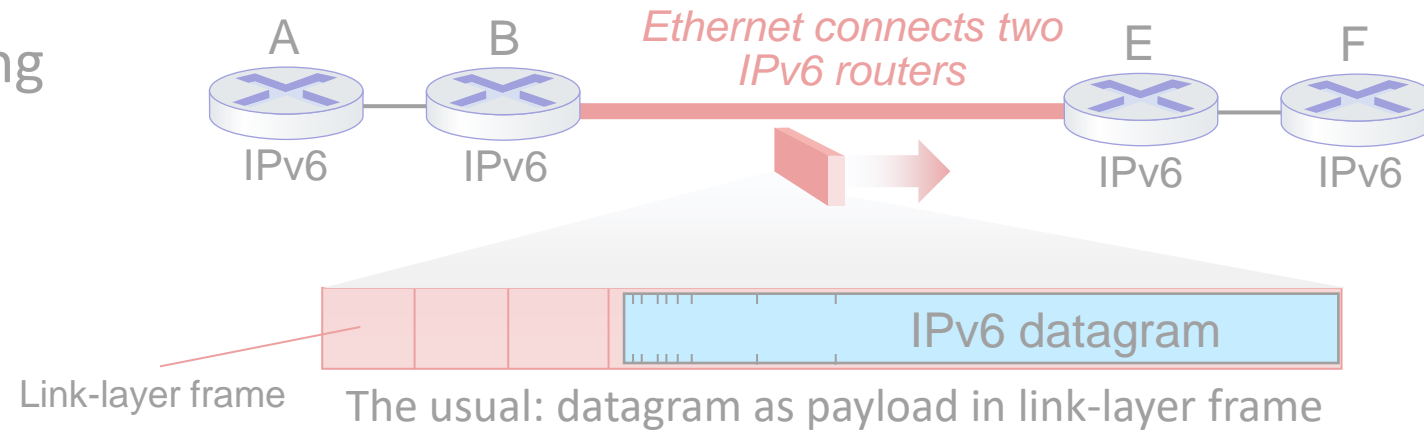- no options (available as upper-layer, next-header protocol at router)

Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?

- tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers ("packet within a packet")
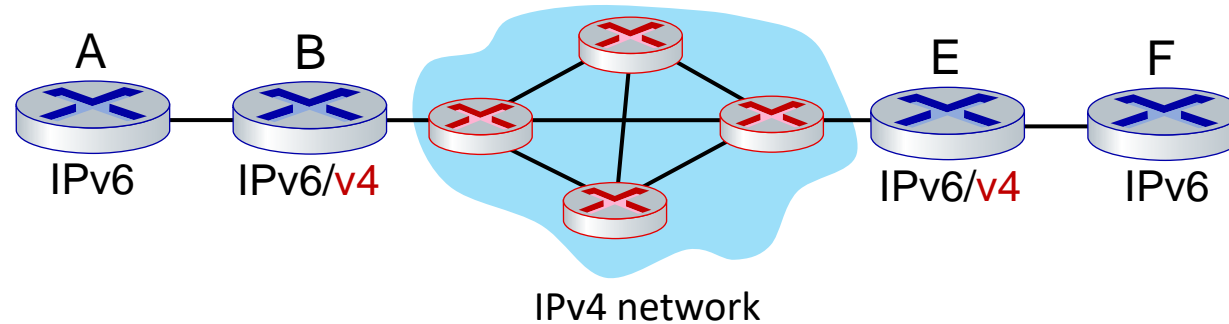 - tunneling used extensively in other contexts (4G/5G)

IPv4 header fields
IPv4 source, dest addr

IPv6 header fields
IPv6 source dest addr
UDP/TCP payload

IPv4 payload

IPv6 datagram

IPv4 datagram

Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

A — IPv6   B — IPv6   *Ethernet connects two IPv6 routers*   E — IPv6   F — IPv6

Link-layer frame   IPv6 datagram

The usual: datagram as payload in link-layer frame

IPv4 network connecting two IPv6 routers

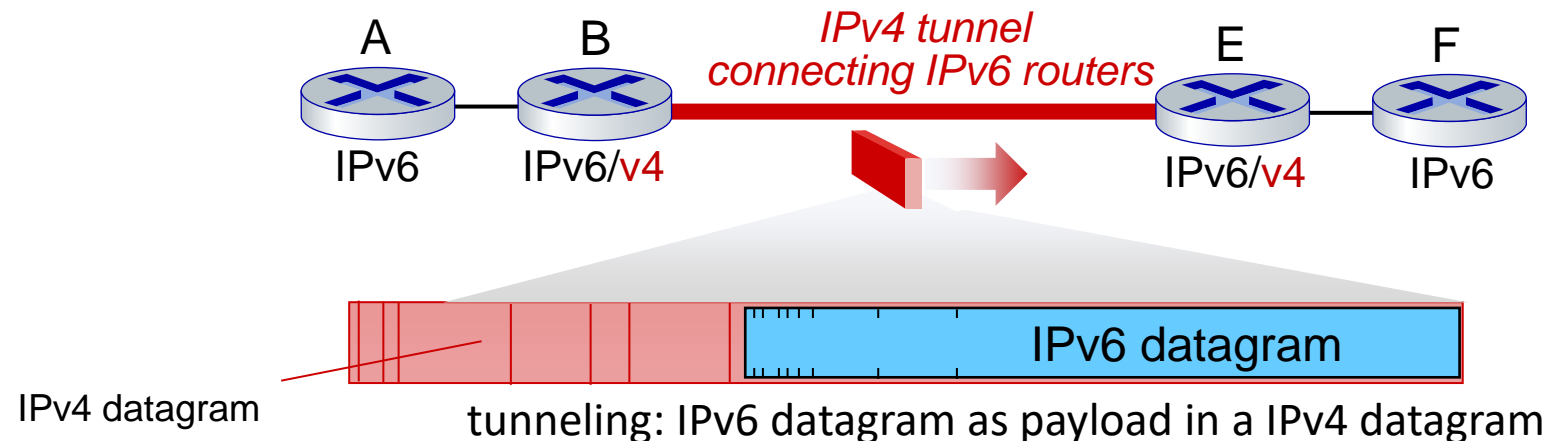A — IPv6   B — IPv6/v4   E — IPv6/v4   F — IPv6

IPv4 network

Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

Ethernet connects two IPv6 routers

A
IPv6
B
IPv6
E
IPv6
F
IPv6

Link-layer frame

IPv6 datagram

The usual: datagram as payload in link-layer frame

IPv4 tunnel connecting two IPv6 routers

A
IPv6
B
IPv6/v4
E
IPv6/v4
F
IPv6

IPv4 tunnel connecting IPv6 routers

IPv4 datagram

IPv6 datagram

tunneling: IPv6 datagram as payload in a IPv4 datagram

Tunneling

logical view:

A   B   *IPv4 tunnel*   E   F
connecting IPv6 routers

IPv6   IPv6/v4   IPv6/v4   IPv6

physical view:

A   B   C   D   E   F

IPv6   IPv6/v4   IPv4   IPv4   IPv6/v4   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

src:B
dest: E

Flow: X
Src: A
Dest: F

data

flow: X
src: A
dest: F

data

Note source and destination addresses!

A-to-B: IPv6

B-to-C: IPv6 inside IPv4

B-to-C: IPv6 inside IPv4

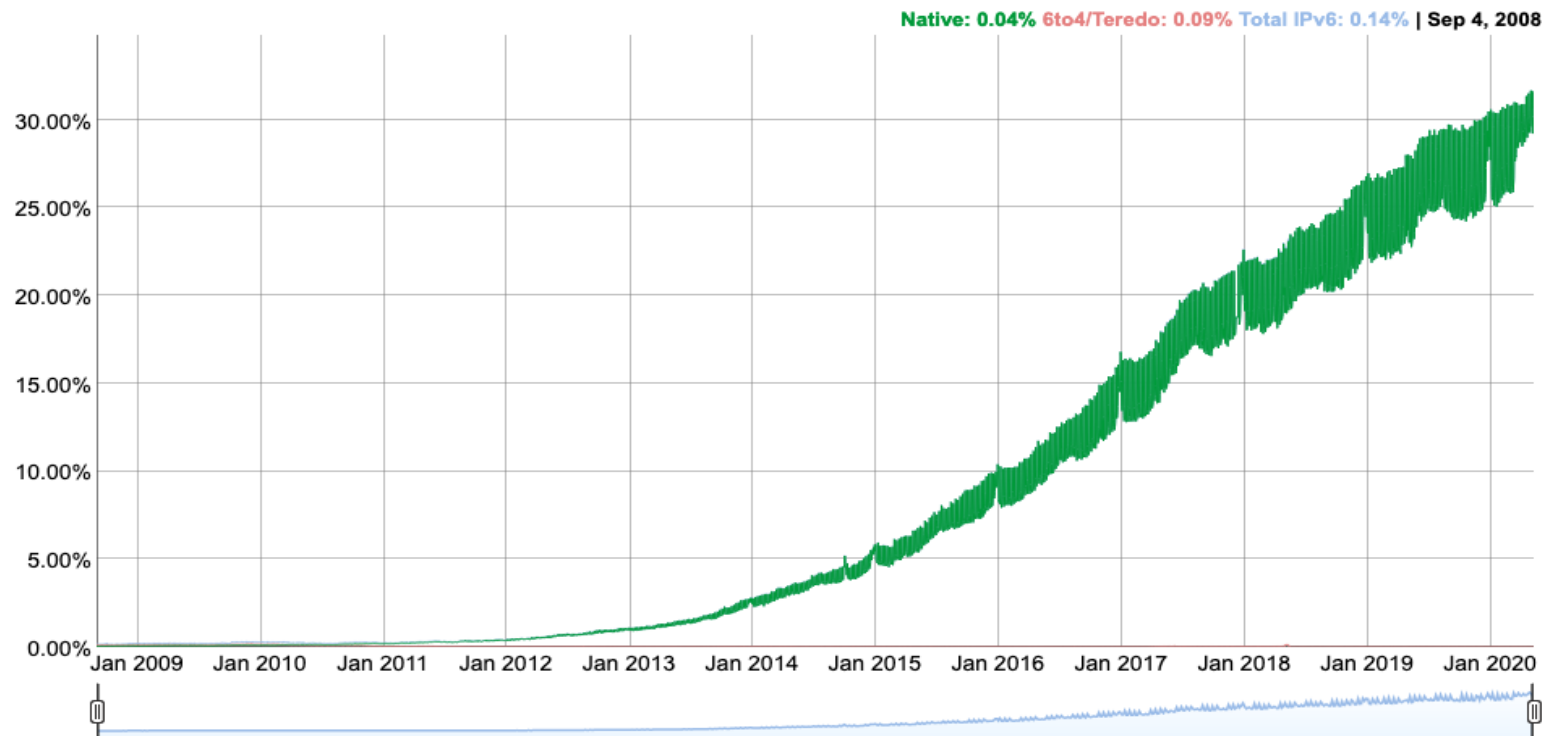B-to-C: IPv6 inside IPv4

E-to-F: IPv6

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

Native: 0.04% 6to4/Teredo: 0.09% Total IPv6: 0.14% | Sep 4, 2008

1

https://www.google.com/intl/en/ipv6/statistics.html