CS 305: Computer Networks

Fall 2022

Lecture 9: Network Layer – The Data Plane

Ming Tang

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

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

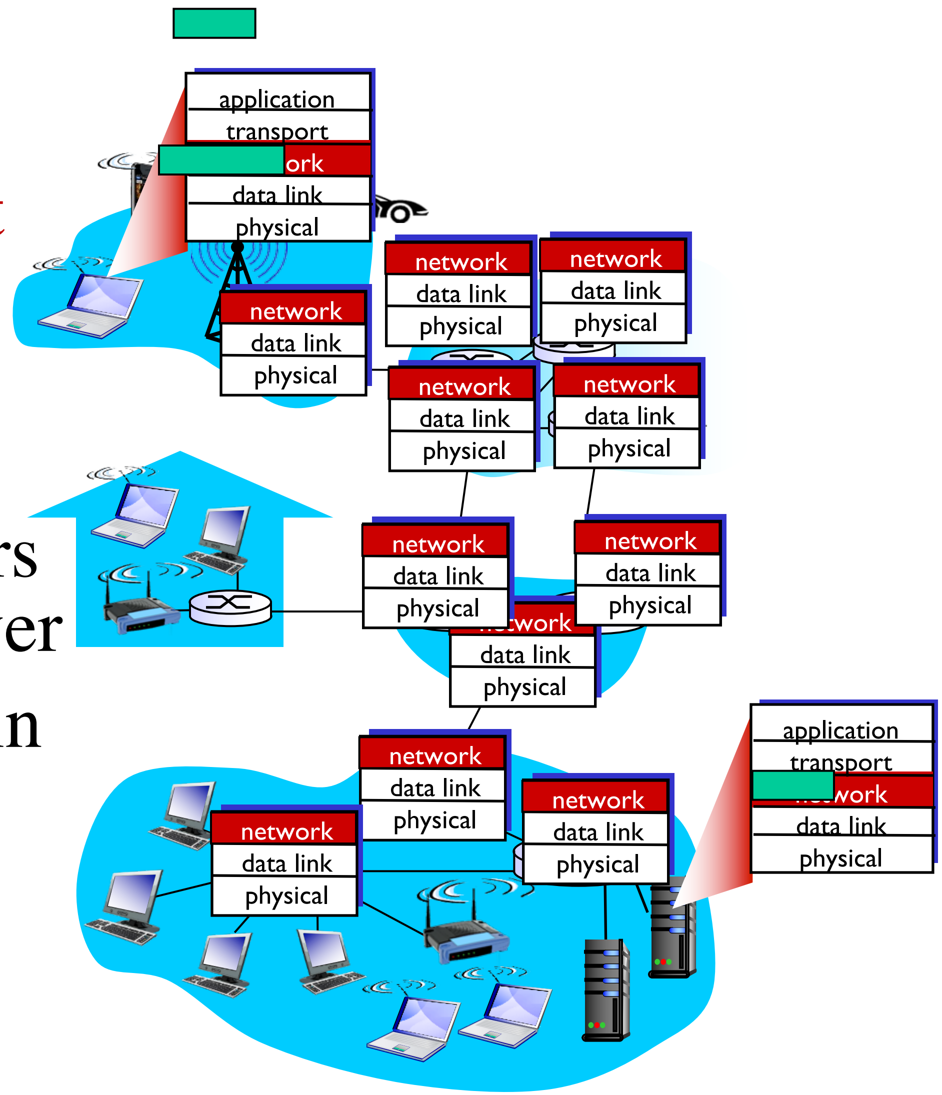
Chapter 4: network layer

Chapter goals:

- Understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding
- Instantiation, implementation in the Internet

Network layer

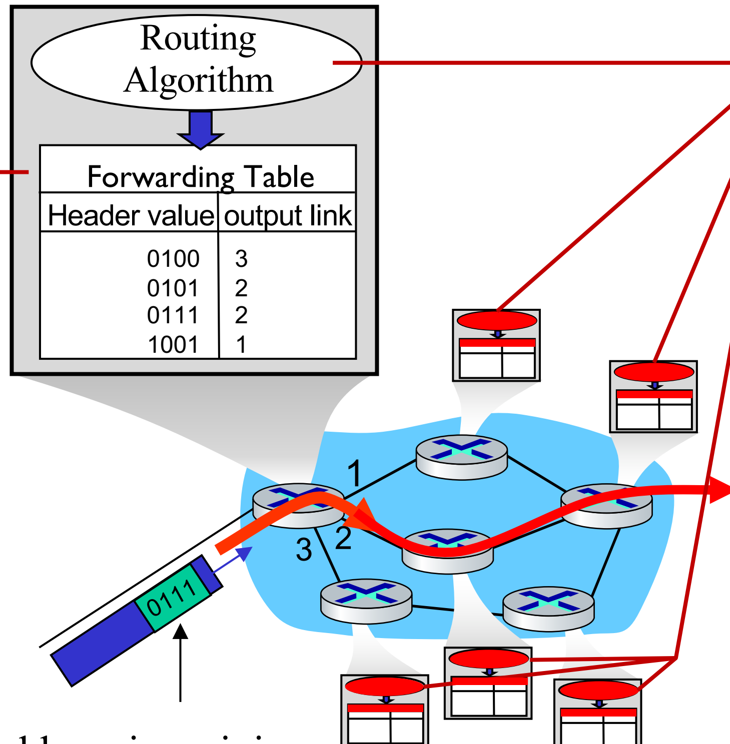
- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



Two key network-core functions

Forwarding:

local action: move arriving packets from router's input link to appropriate router output link



destination address in arriving packet's header

Routing:

- global action: determine source-destination paths taken by packets
- routing algorithms

Two key network-layer functions

Network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output
  - Data plane
- *routing:* determine route taken by packets from source to destination
  - routing algorithms
  - Control plane

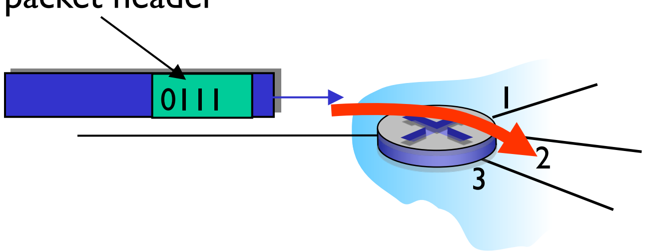
Analogy: taking a trip

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

Network layer: data plane, control plane

Data plane

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



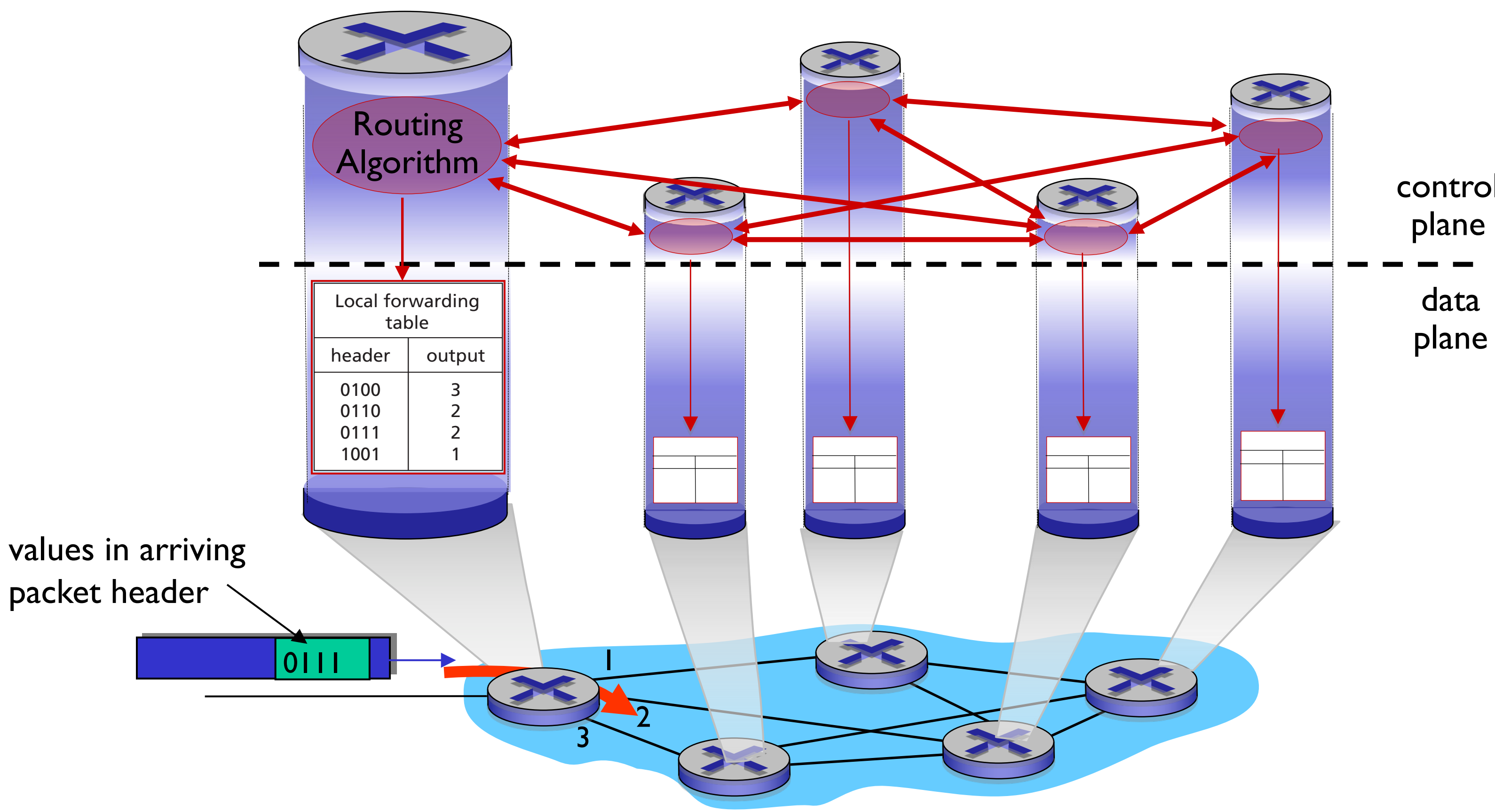
Control plane

- network-wide logic, softward
- 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

Control plane: Traditional Approach

Per-router control plane: Individual routing algorithm components *in each and every router* interact in the control plane

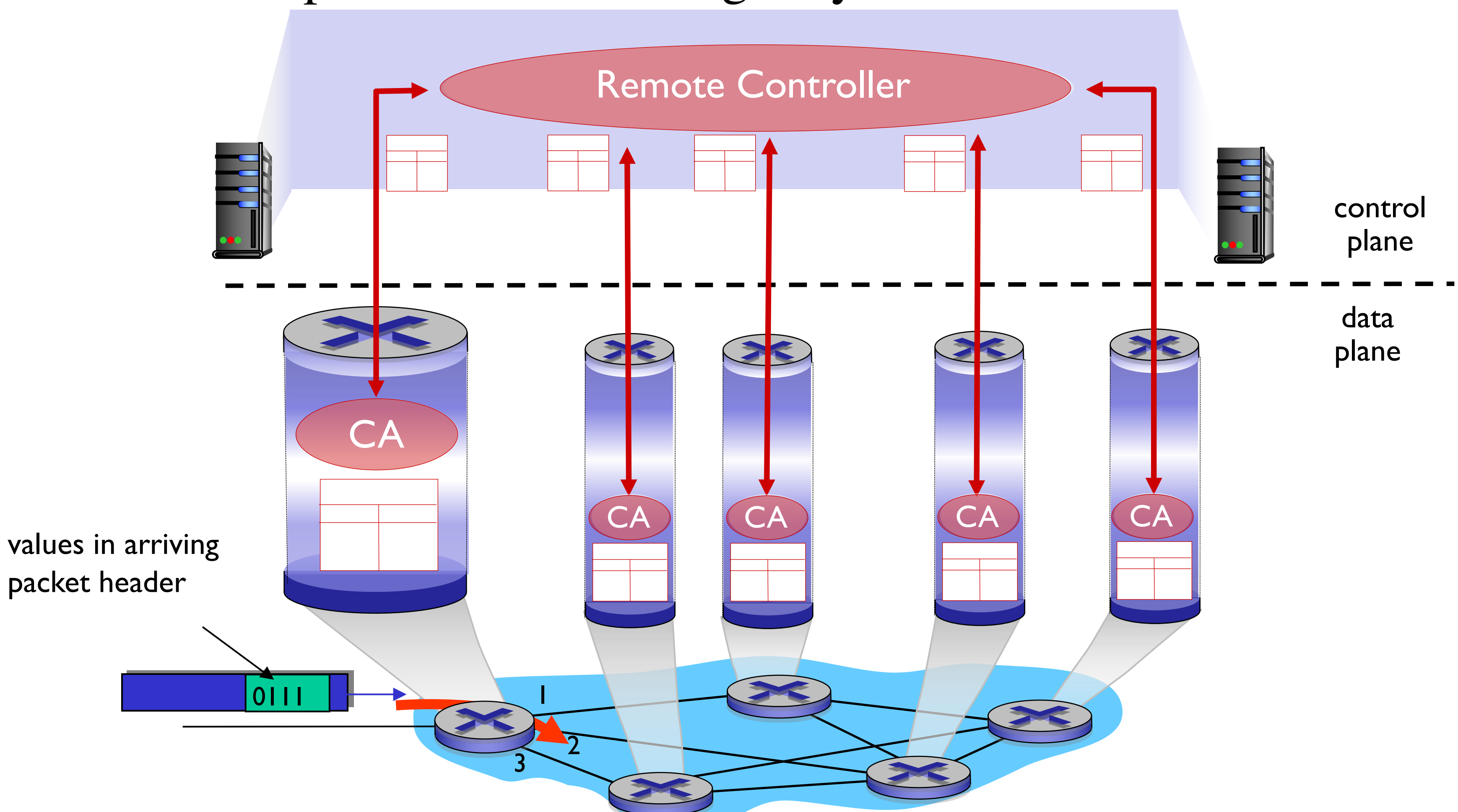
- Forwarding and routing functions are contained within a router



Control Plane: The SDN Approach

Logically centralized control plane: A distinct (typically remote) controller interacts with local control agents (CAs)

- Router performs forwarding only



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

other example services: security

Internet service model provide "best effort" service, no guarantee on bandwidth, loss, order or timing.

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

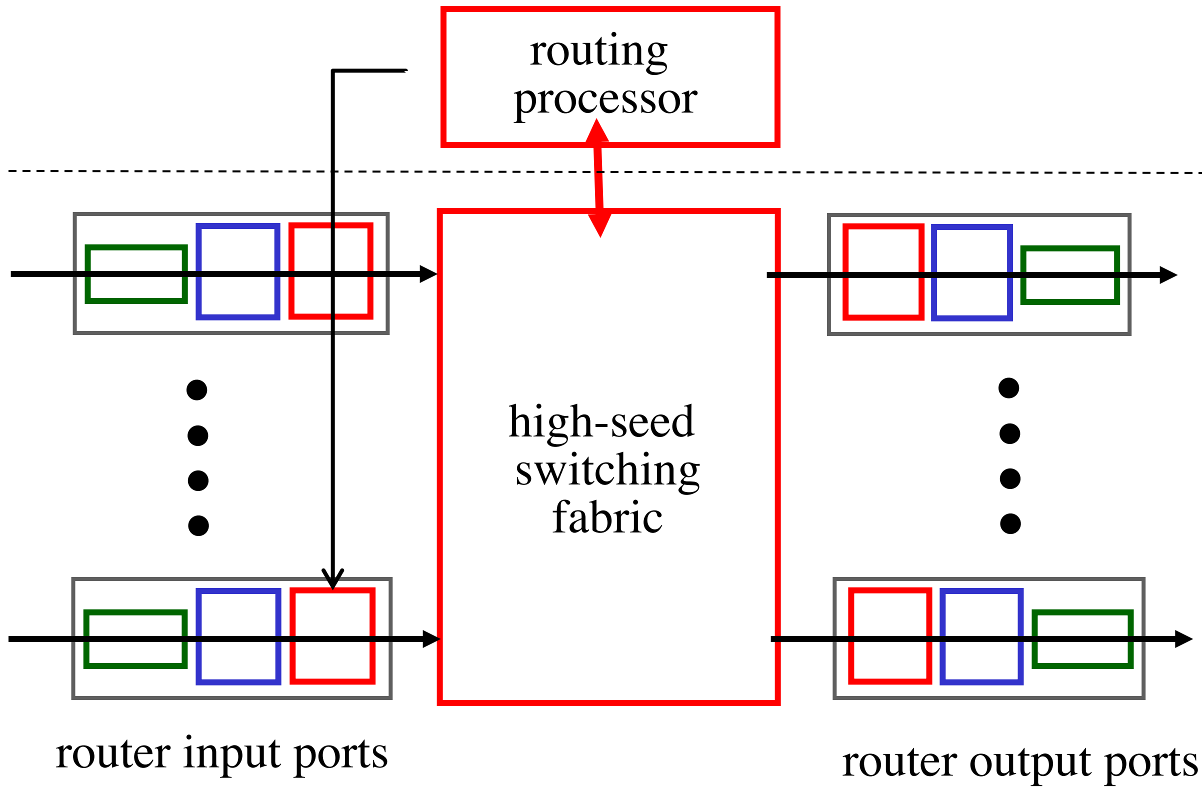
4.2 What's inside a router

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

Router architecture overview

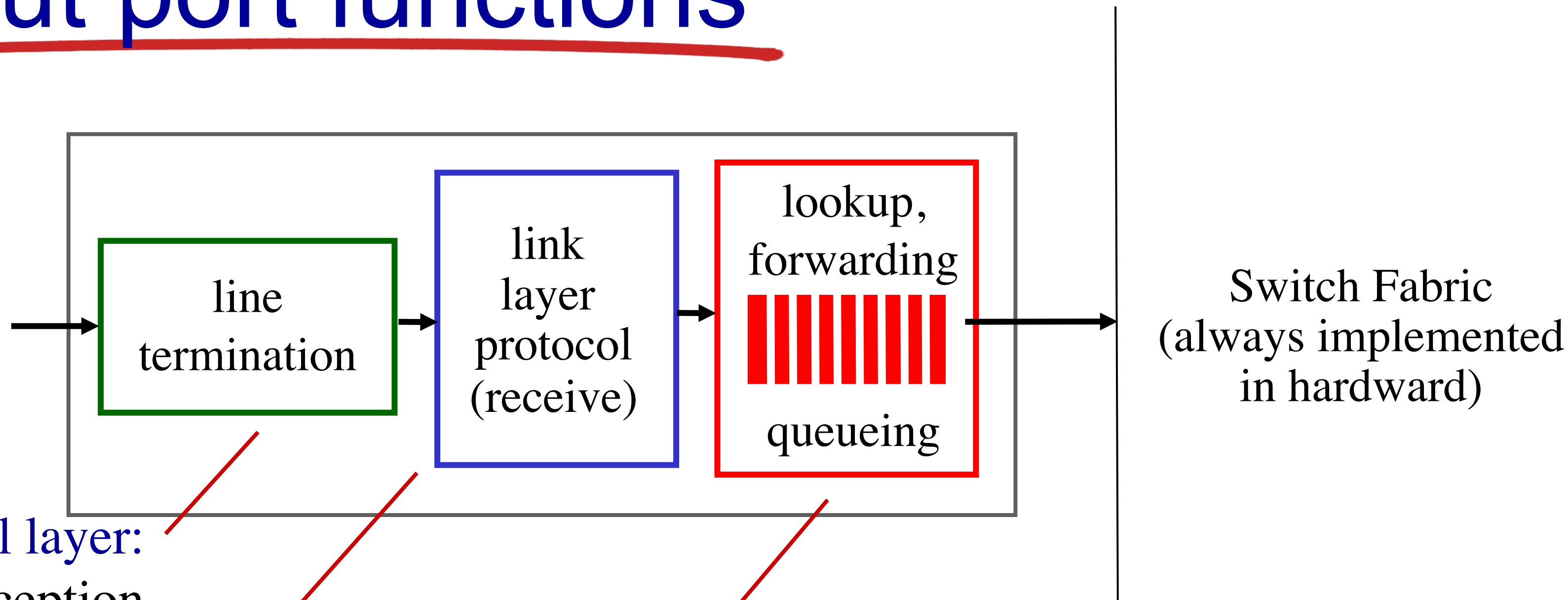
High-level view of generic router architecture:



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

forwarding data plane
(hardware) operates in
nanosecond timeframe

Input port functions



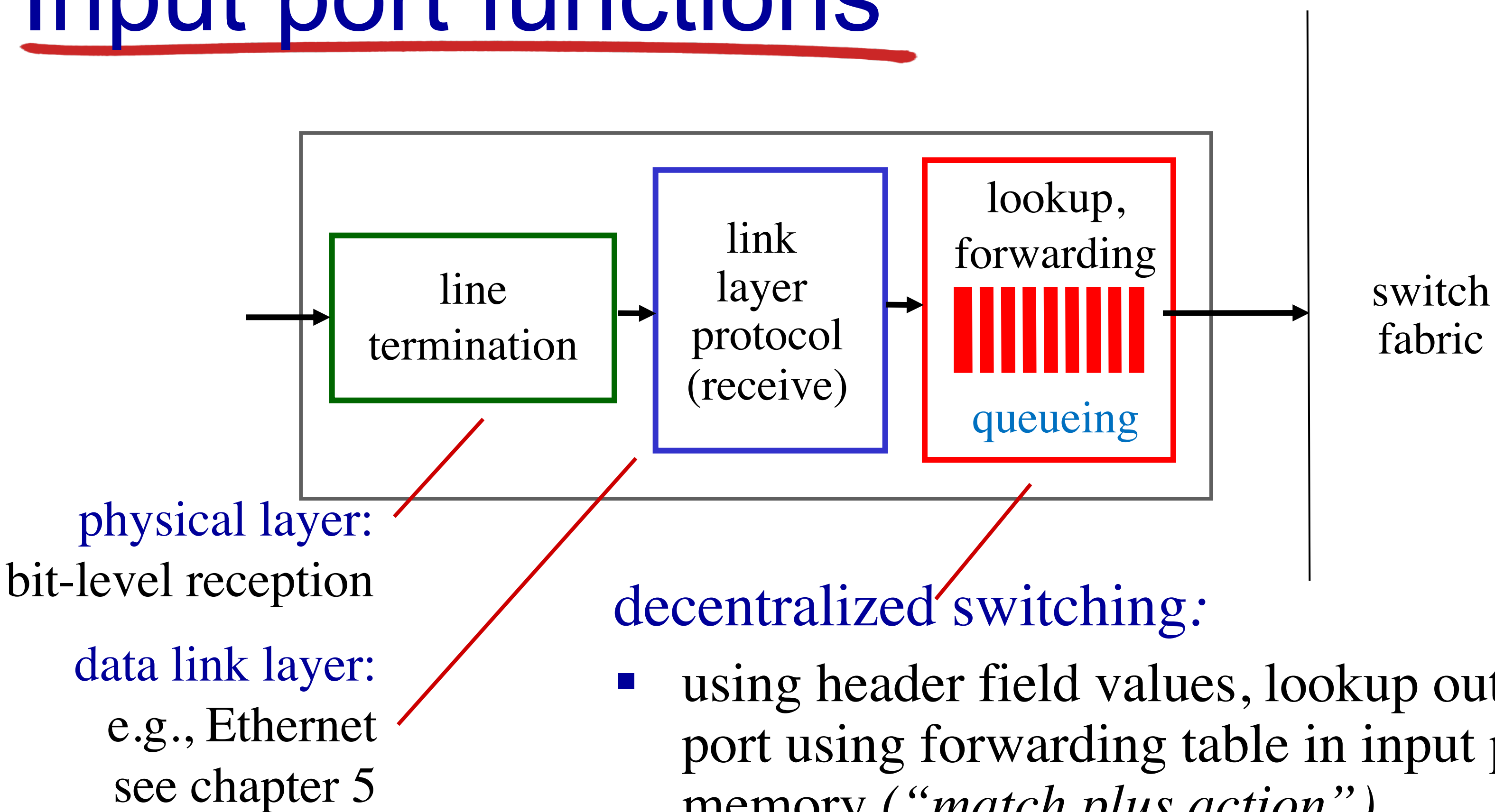
physical layer:
bit-level reception

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

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'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- *destination-based forwarding:* forward based only on destination IP address (traditional)
- *generalized forwarding:* forward based on any set of header field values

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 (i.e., overlap between entities)?

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

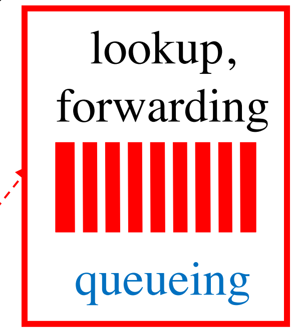
examples:

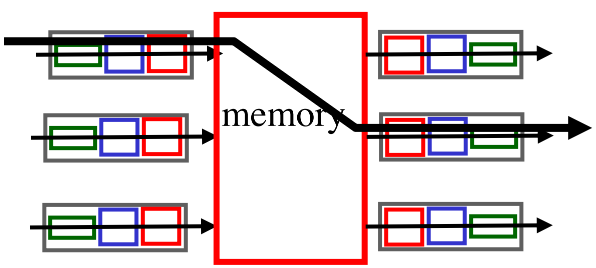
DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

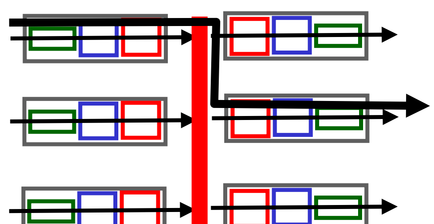
Switching fabrics

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

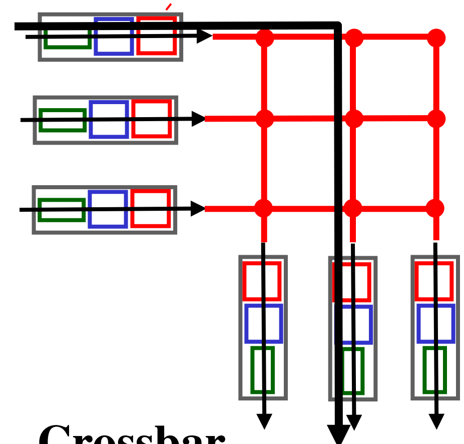




Switch via memory
- Interrupt; write and read
- Two packets cannot be forwarded at the same time

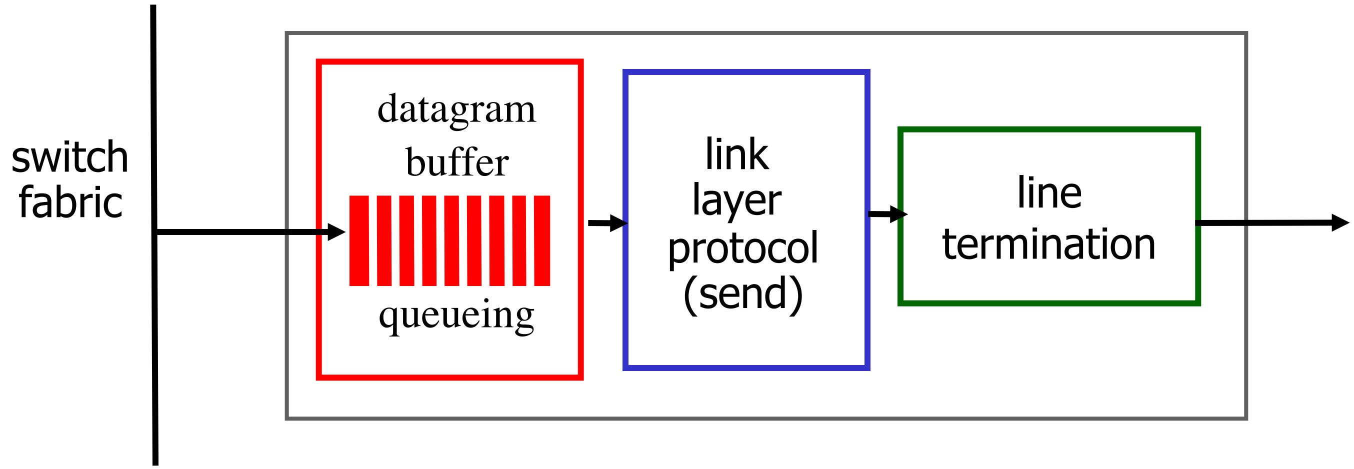


Switch via bus
- Broadcast; label
- One packet can cross at a time



Crossbar
- Multiple packets in parallel
- Non-blocking

Output ports



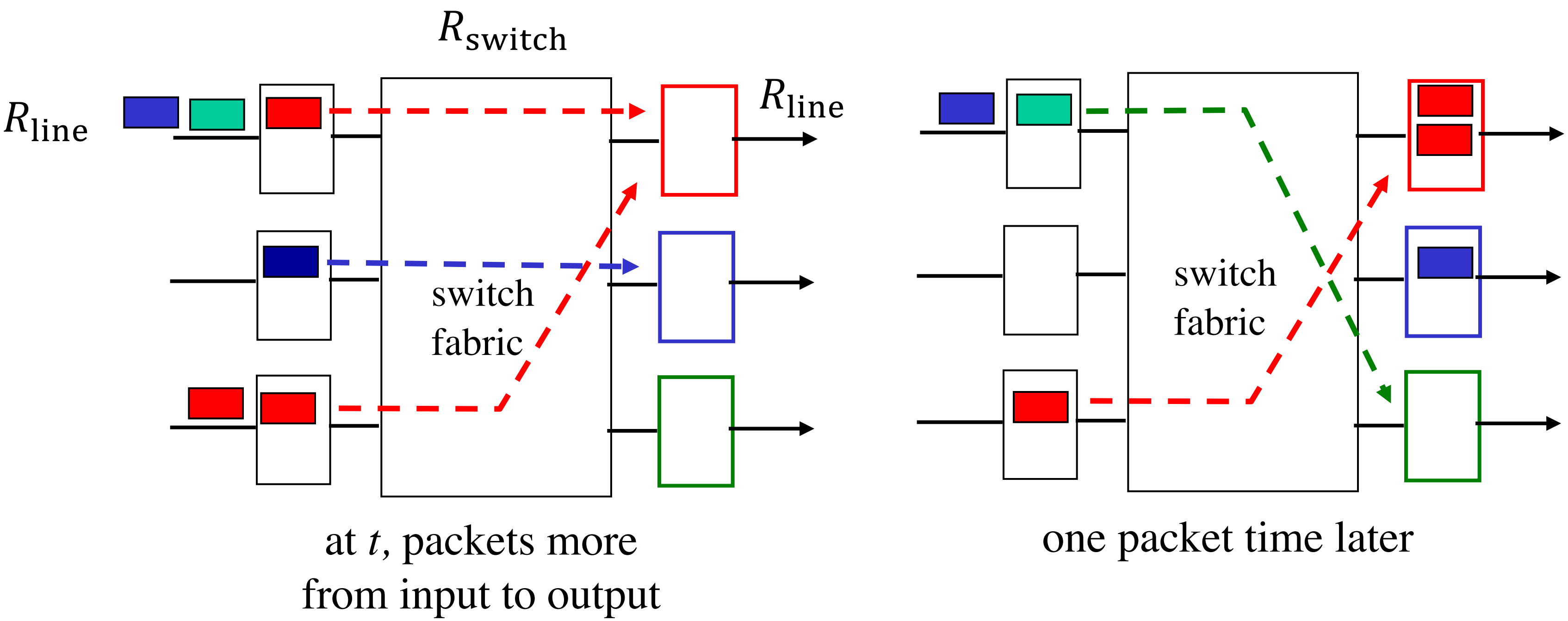
- *buffering* required when datagrams arrive from fabric faster than the transmission rate

Datagram (packets) 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

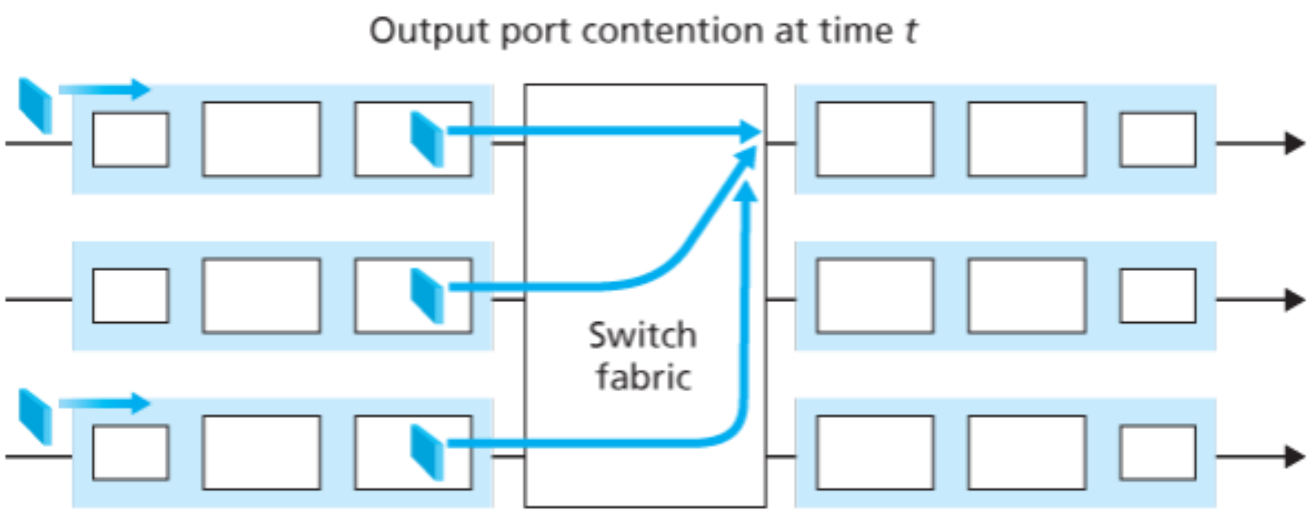
Input port queueing



Switch fabric is not fast enough (e.g., suppose $R_{\text{switch}} = R_{\text{line}}$)

- Packet queuing occur at input port
- Crossbar, and multiple packets must be transferred to the same port

Output port queueing



- If R_{switch} is N times faster than R_{line}, then negligible queuing at the input ports.
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

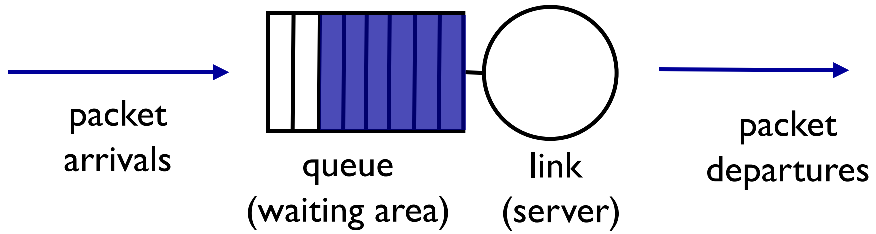
How much buffering?

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

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

Scheduling mechanisms

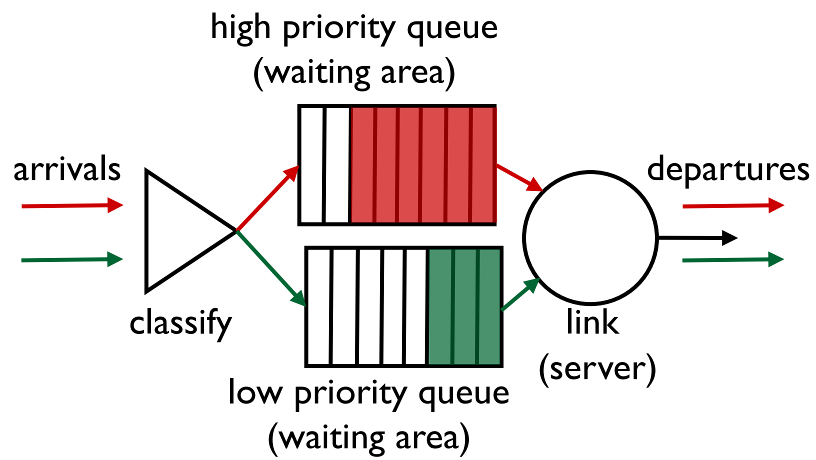
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) queuing:* send in order of arrival to queue
- *discard policy:* if packet arrives to full queue: who to discard?
  - *tail drop:* drop arriving packet
  - *priority:* drop/remove on priority basis
  - *random:* drop/remove randomly

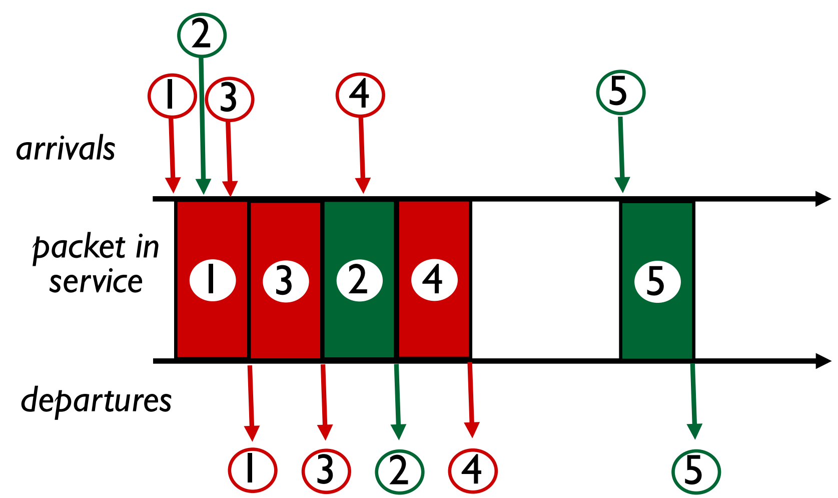


Scheduling policies: priority

priority queuing: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
- Transmit a packet from the highest priority class
- Non-preemptive priority queuing

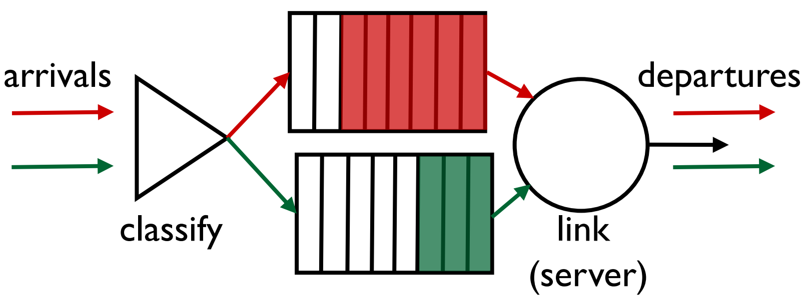


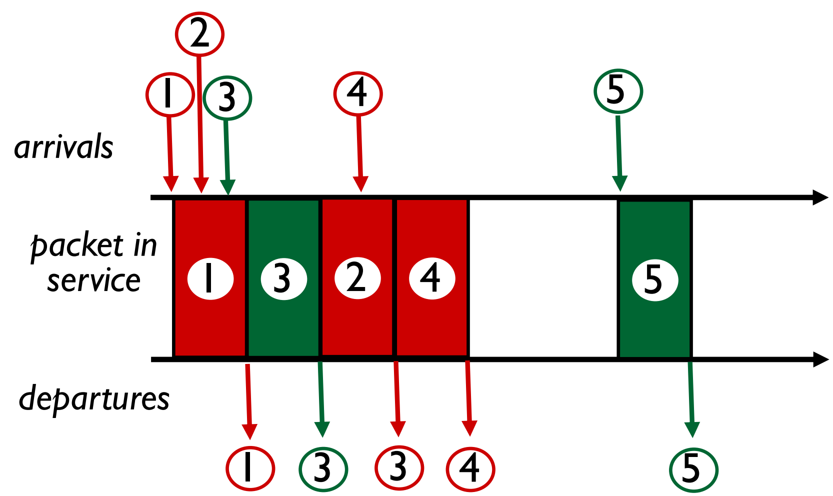


Scheduling policies: still more

Round Robin (RR) scheduling:

- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)

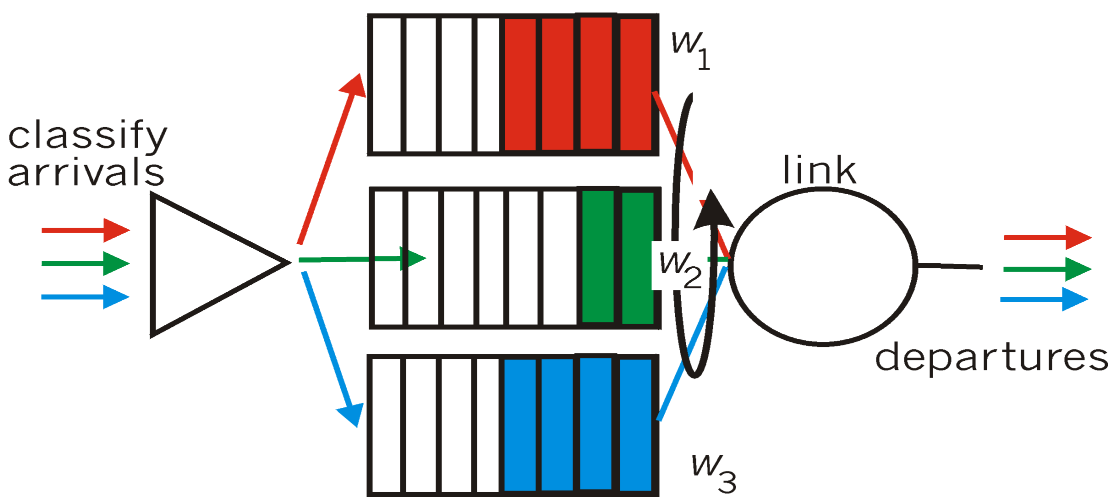




Scheduling policies

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle
  - $w_i / \sum_{j \in Q^{busy}} w_j$ of the bandwidth (thoughput)
  - Q^{busy}: all classes that have queued packets
  - Worst case: all queues have packets; $w_i / \sum_{j \in Q} w_j$



Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

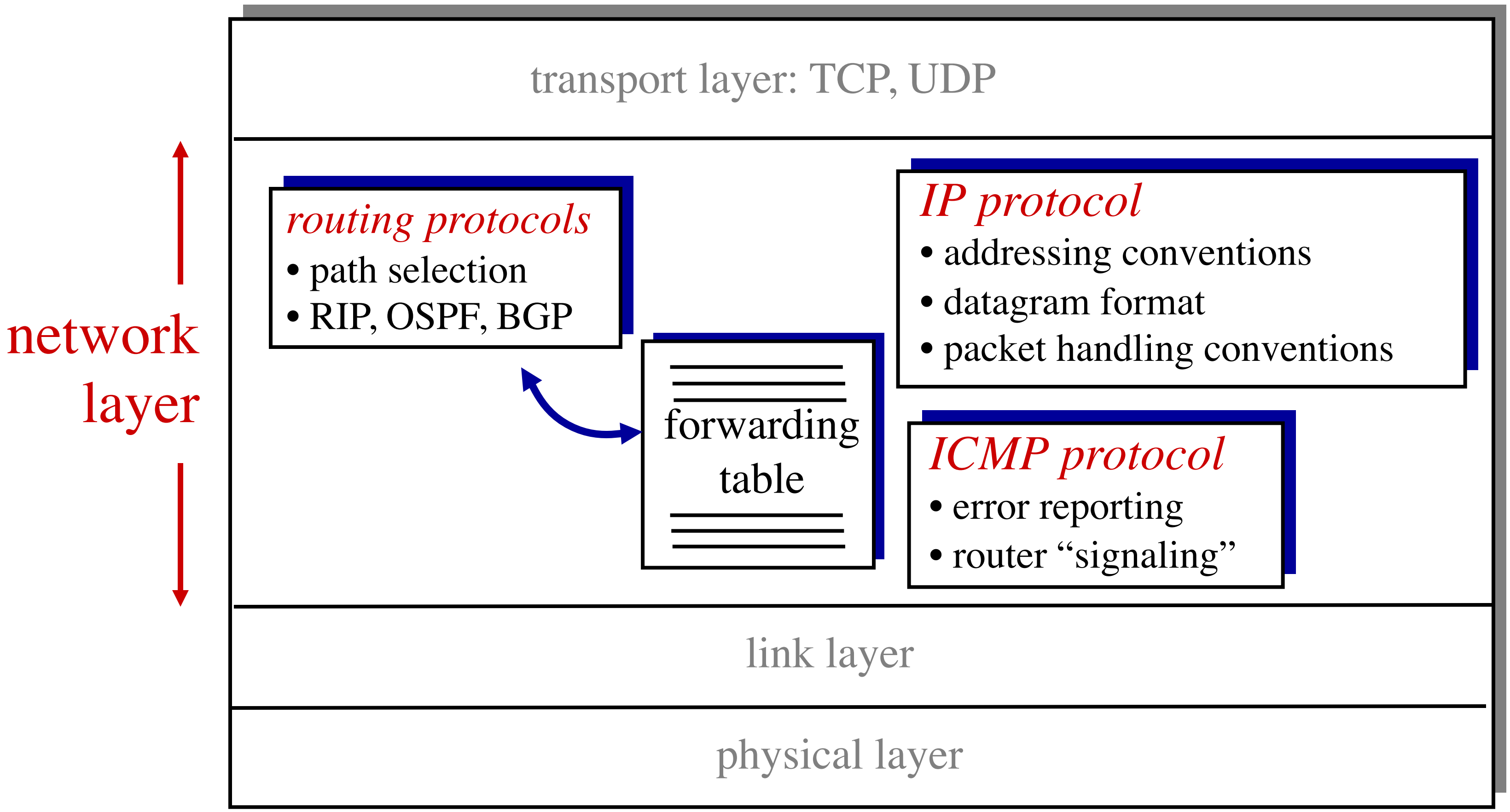
4.2 What's inside a router

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

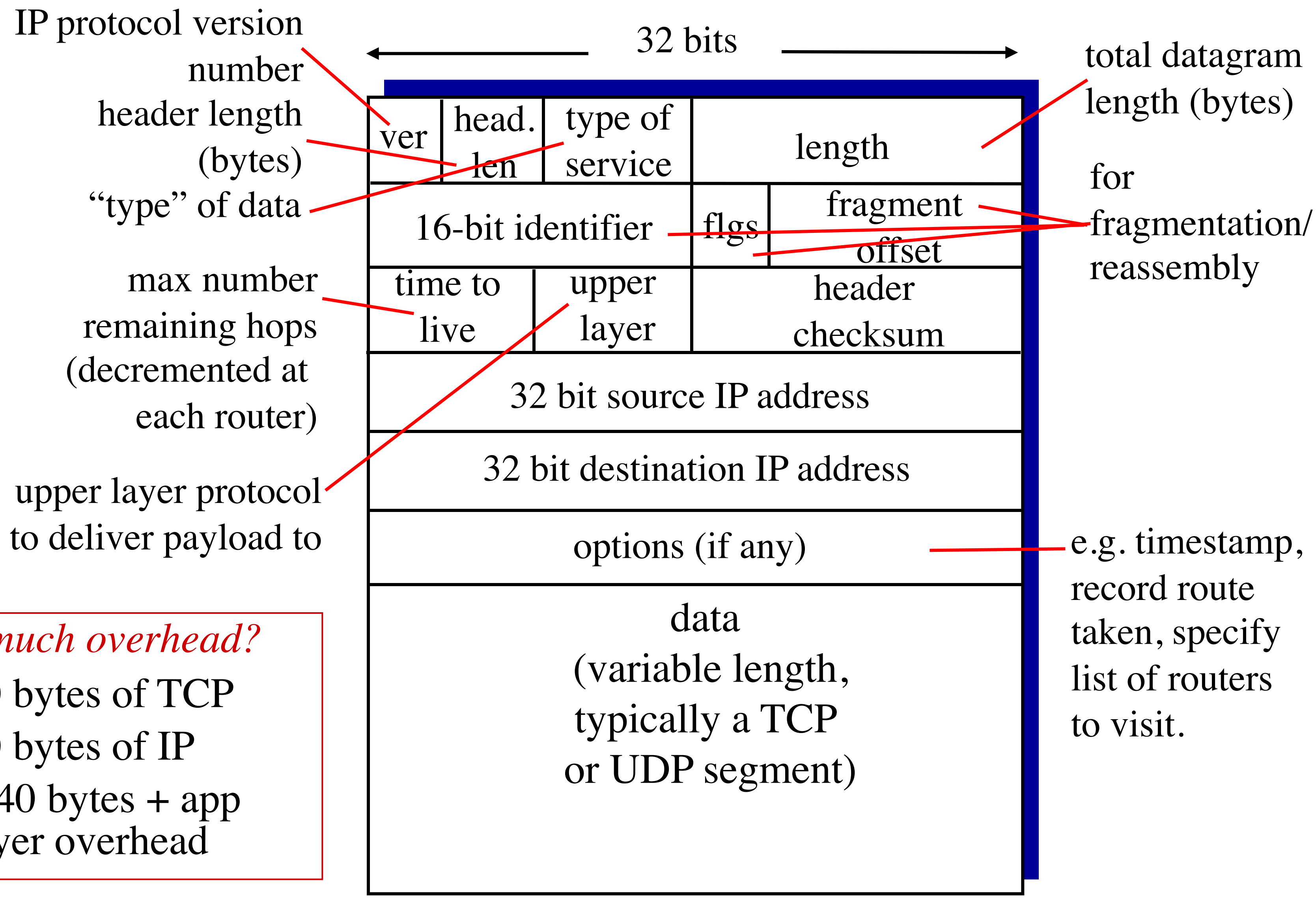
4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

The Internet network layer

host, router network layer functions:



IP datagram format

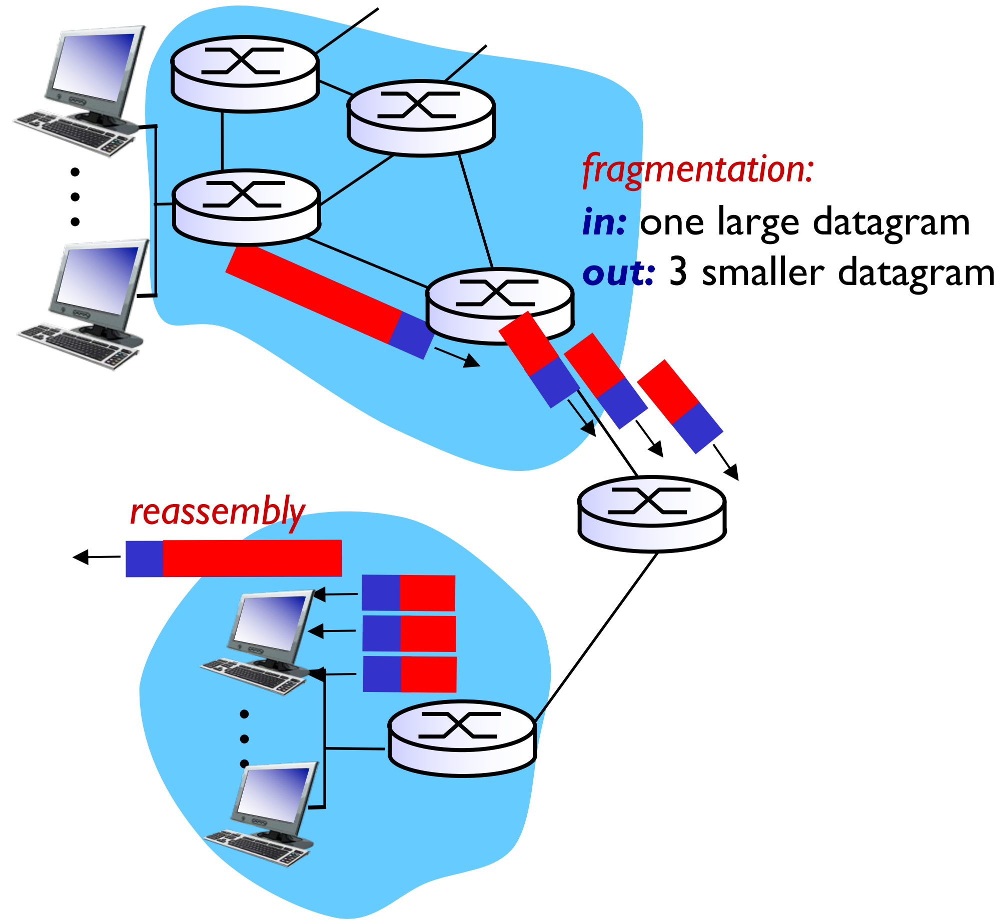


how much overhead?

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

IP fragmentation, reassembly

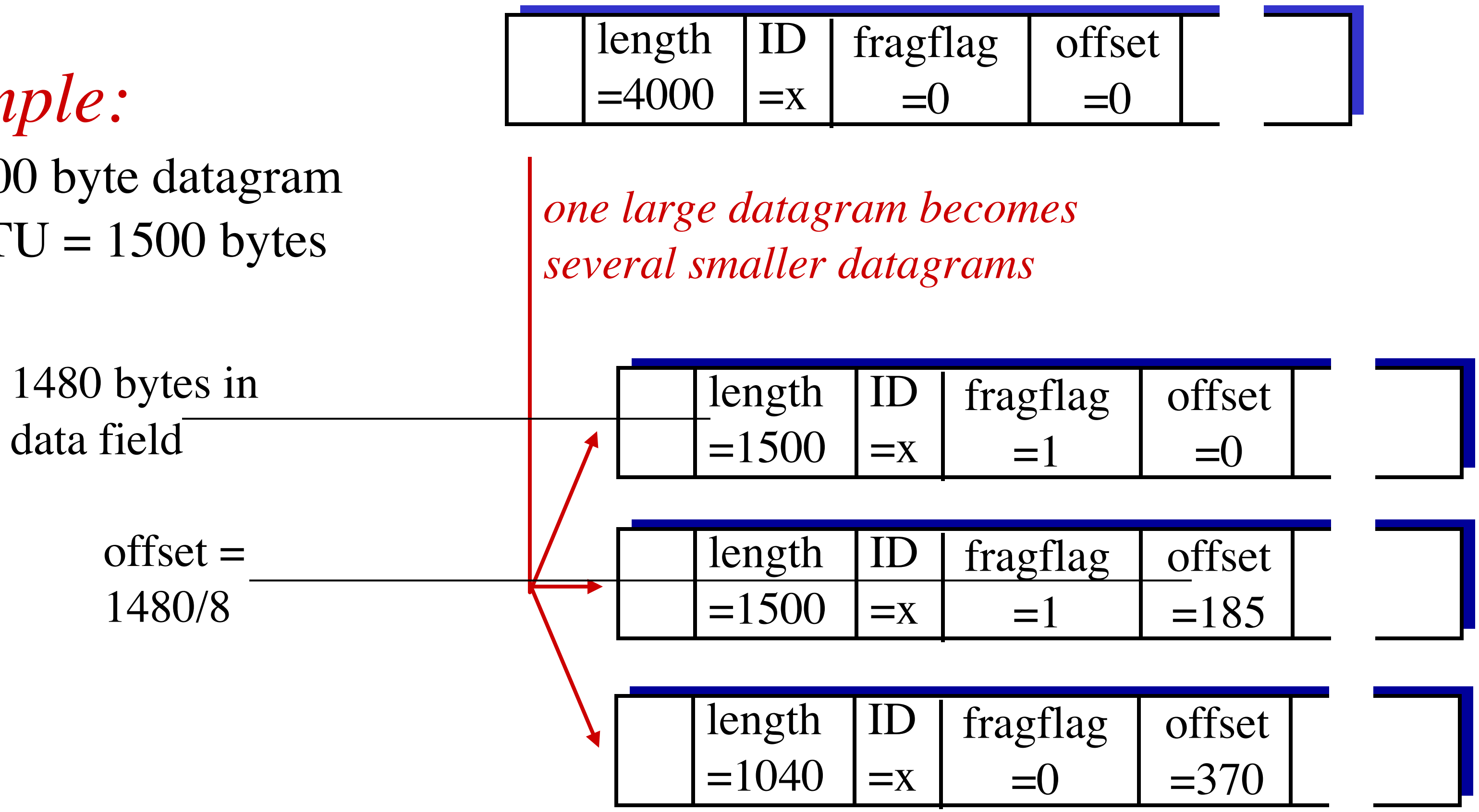
- network links have MTU (max. transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify,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



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

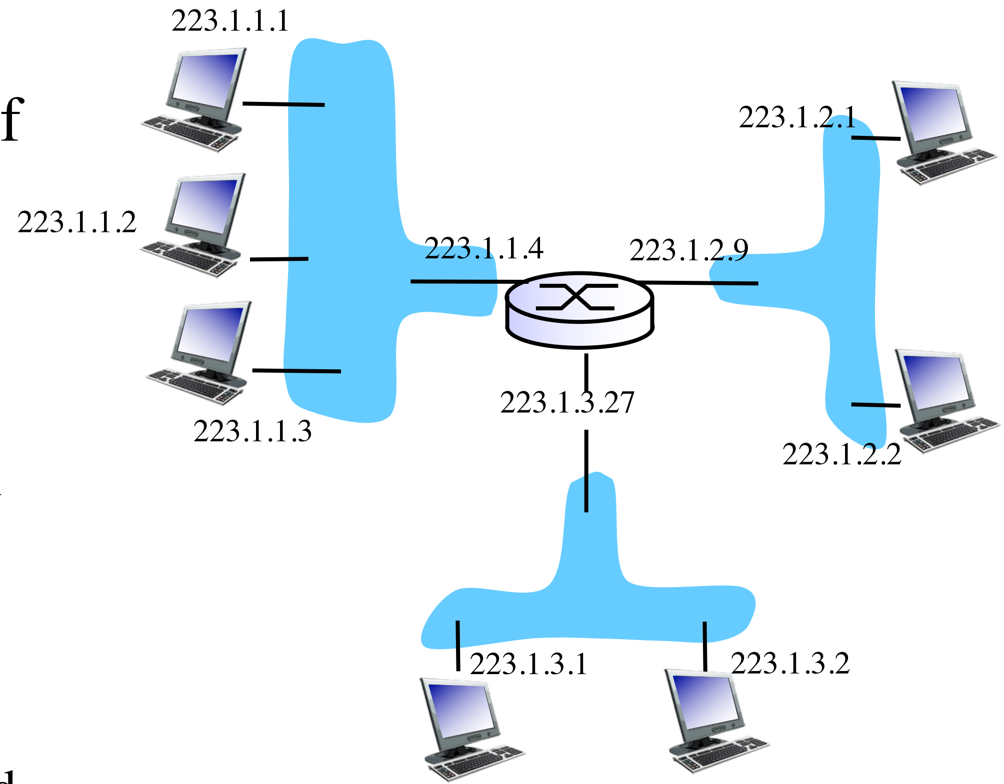
4.2 What's inside a router

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 interface of hosts and routers
- *interface:* (network interface card) 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*



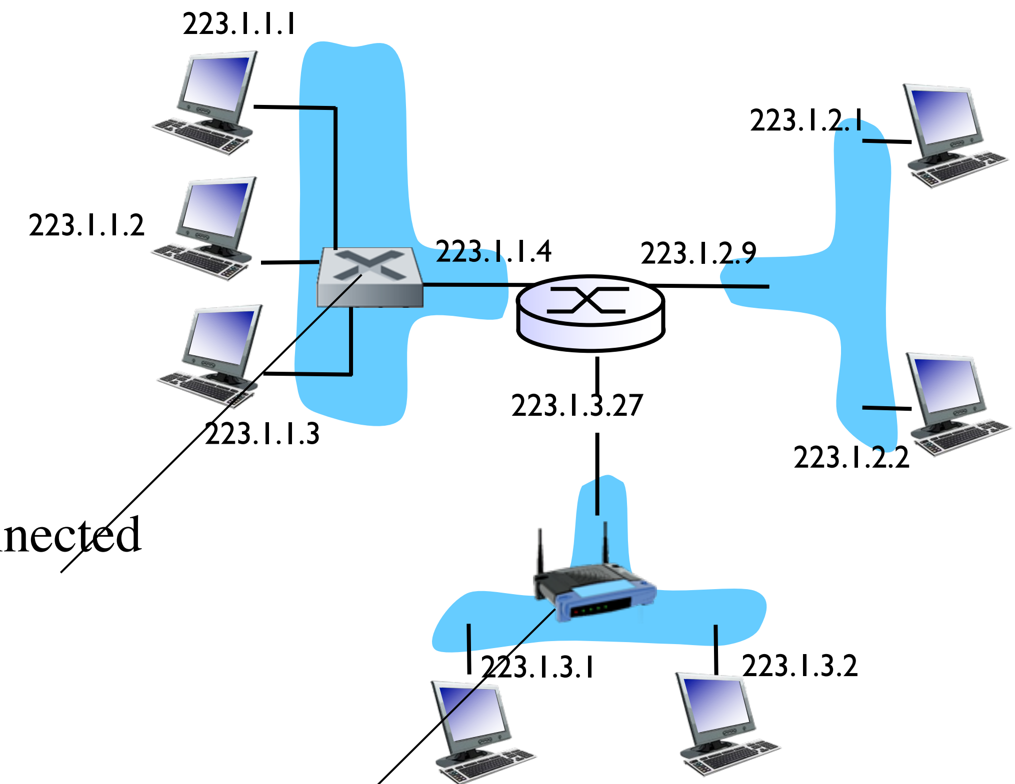
$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

- *Each interface in the global Internet must have an IP address that is globally unique (except for interfaces behind NATs)*

IP addressing: introduction

Q: how are interfaces actually connected?

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



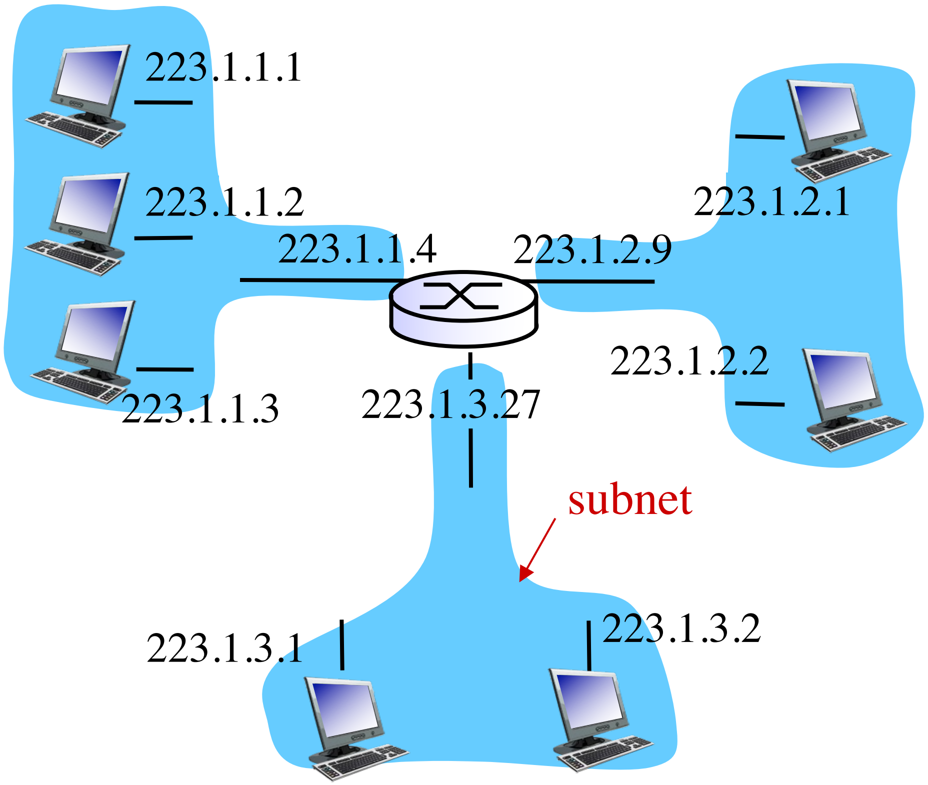
A: wired Ethernet interfaces connected by Ethernet switches

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

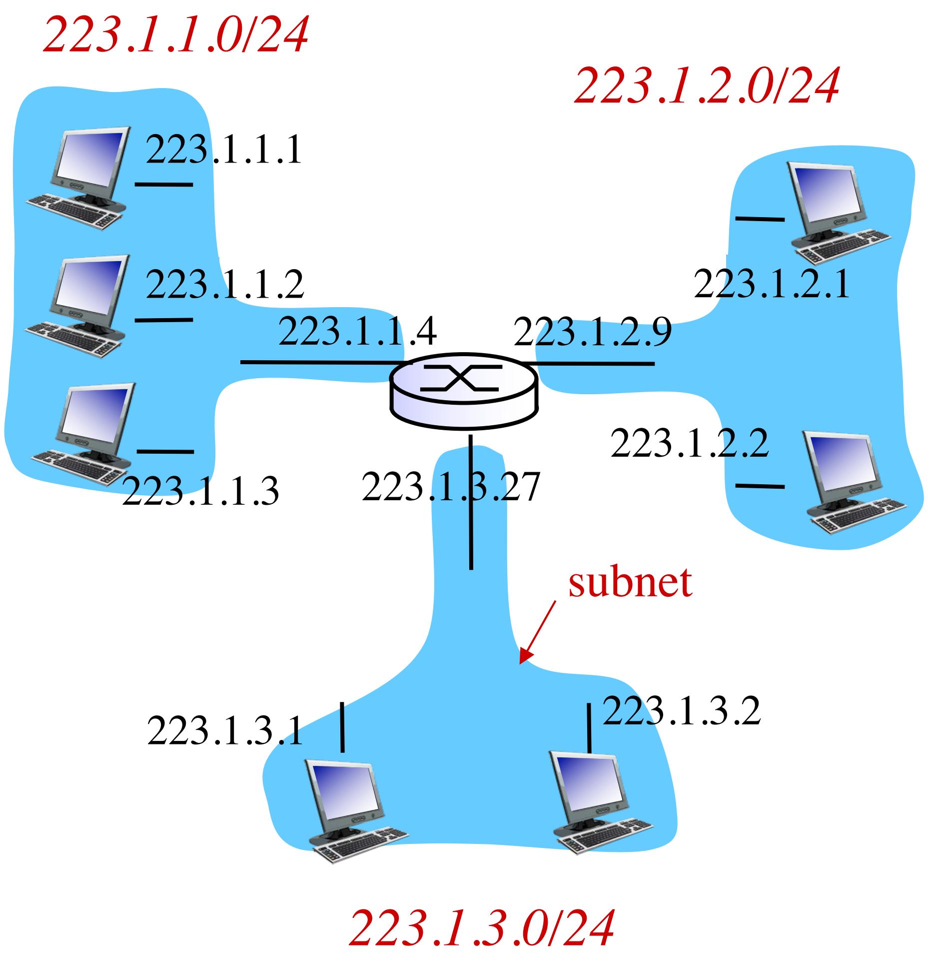


network consisting of 3 subnets

Subnets

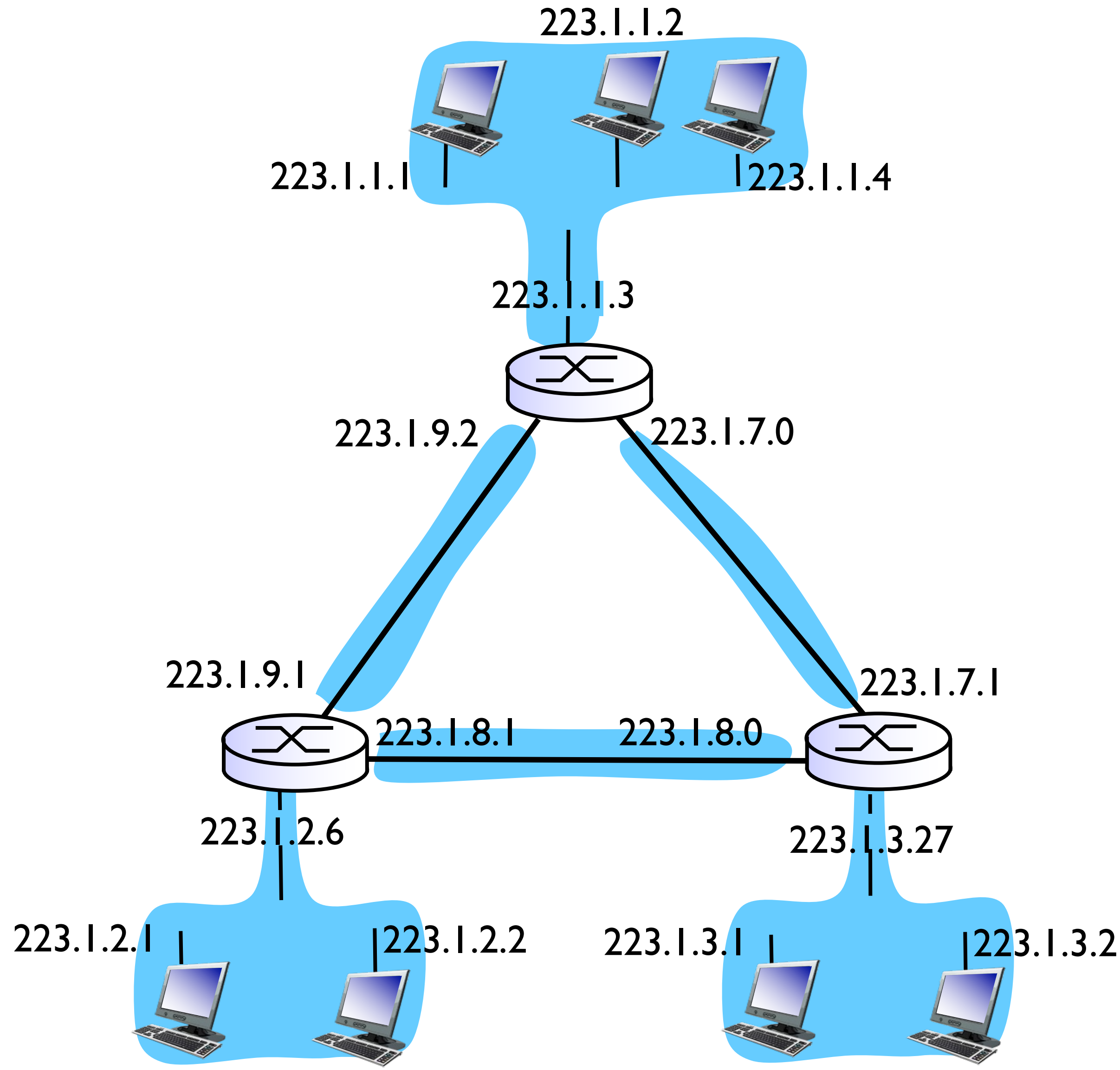
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*
- Address assigned to a subnet: *223.1.1.0/24*

subnet mask: /24
255.255.255.0



Subnets

how many?



IP addressing: CIDR

CIDR: Classless Inter Domain 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

Subnet mask: 255.255.254.0

Obtaining a Block of Address

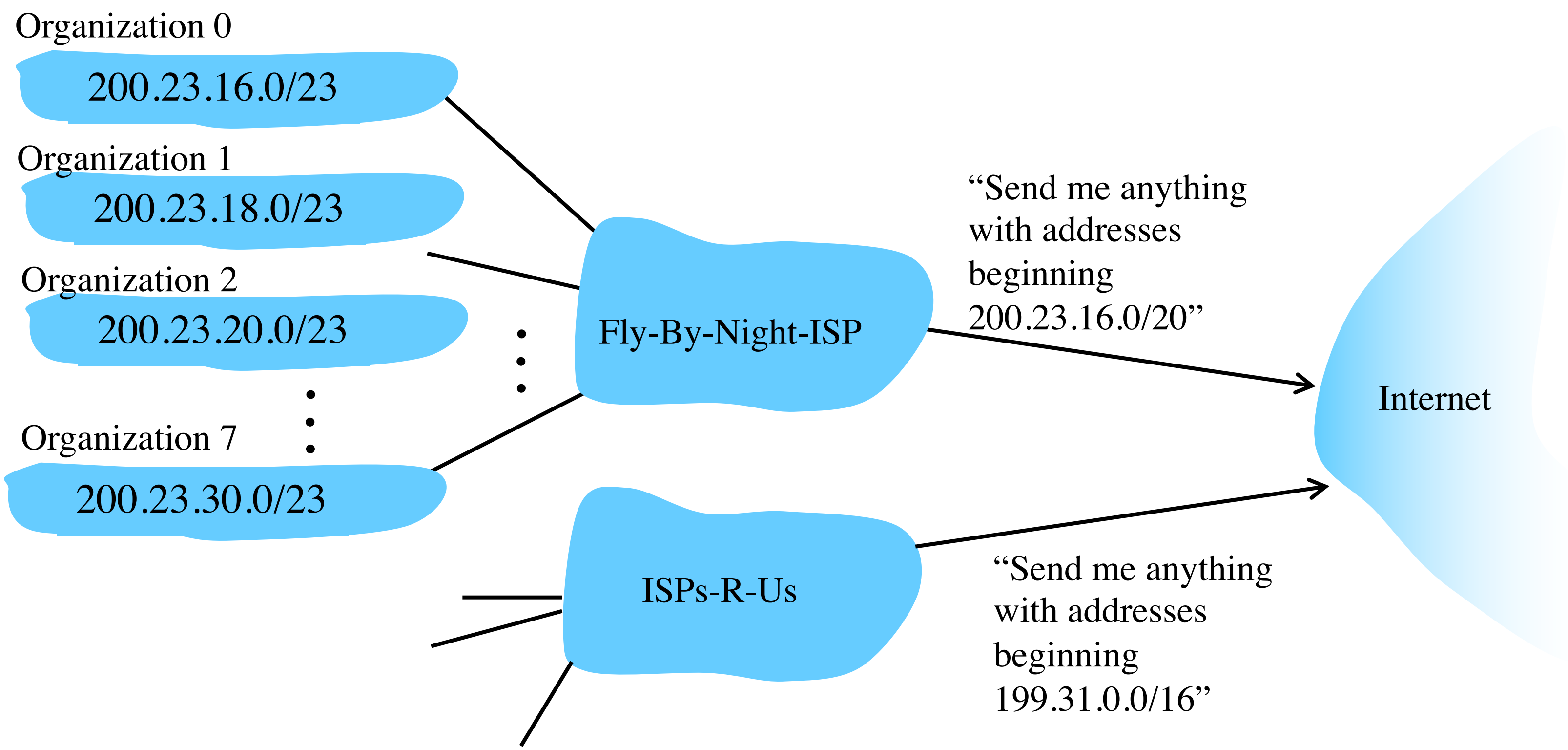
Q: how does *network* get subnet part of IP addr?

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:



Question 1

Suppose an ISP owns the block of addresses of the form 192.168.56.32/27. Suppose it wants to create four subnets from this block, with each block having the same number of IP addresses.

What are the prefixes (of form a.b.c.d/x) for the four subnets?

11000000 10101000 00111000 00100000

11000000 10101000 00111000 00100000

11000000 10101000 00111000 00101000

11000000 10101000 00111000 00110000

11000000 10101000 00111000 00111000

192.168.56.32/29
192.168.56.40/29
192.168.56.48/29
192.168.56.56/29

Question 2

Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17.0/24.

- Subnet 1 is required to support at least 60 interfaces $60 \leq 64 = 2^6$
- Subnet 2 is to support at least 90 interfaces $90 \leq 128 = 2^7$
- Subnet 3 is to support at least 12 interfaces $12 \leq 16 = 2^4$

Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints

11011111 00000001 00010001 00000000

Subnet 2: 11011111 00000001 00010001 00000000 223.1.17.0/25

Subnet 1: 11011111 00000001 00010001 10000000 223.1.17.128/26

Subnet 3: 11011111 00000001 00010001 11000000 223.1.17.192/28

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 root servers
- assigns domain names, resolves disputes

Obtaining a Host Address

Once obtained a block of addresses:

- manually configure the IP addresses into the router

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

- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"
  - Same IP each time, or temporary IP addresses

DHCP: Dynamic Host Configuration Protocol

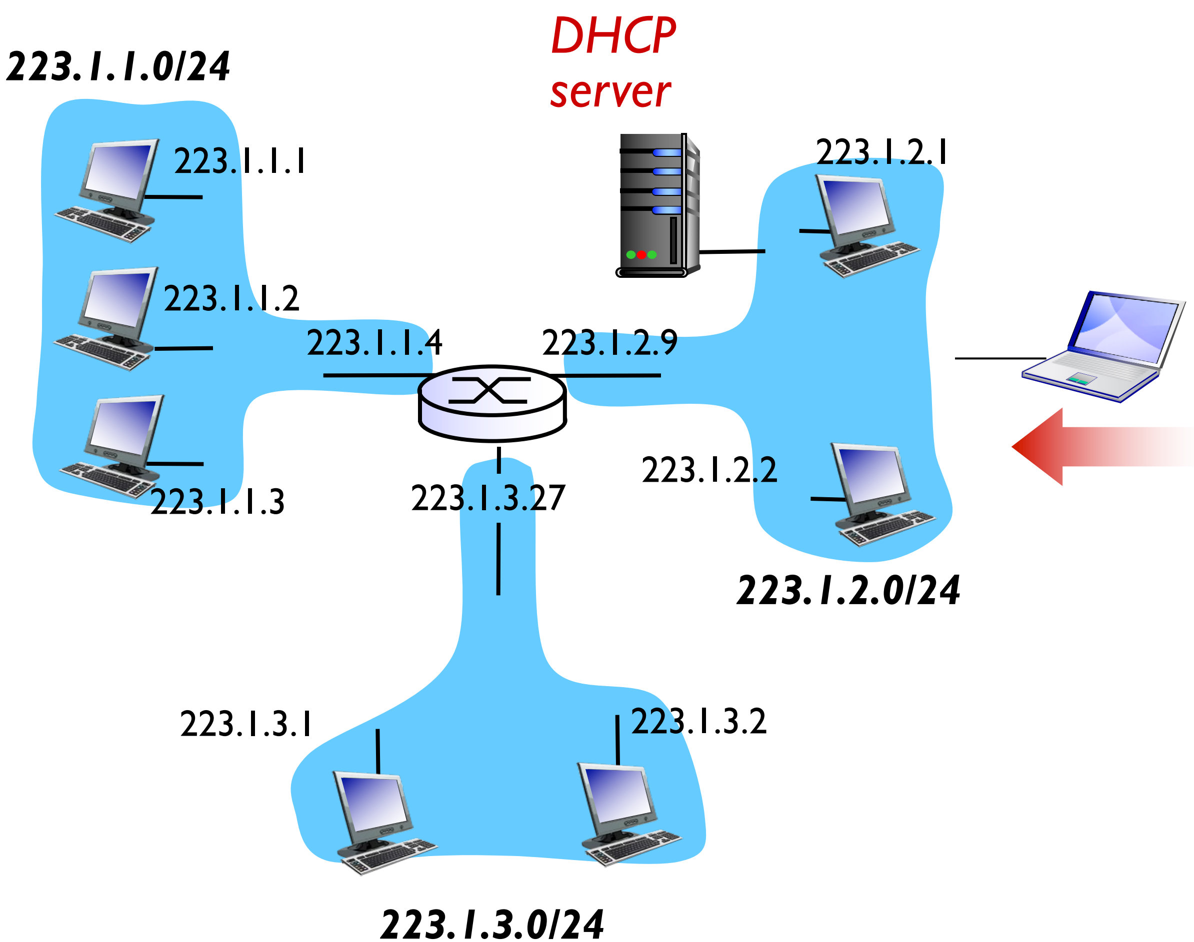
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

DHCP overview (A Client-Server Protocol):

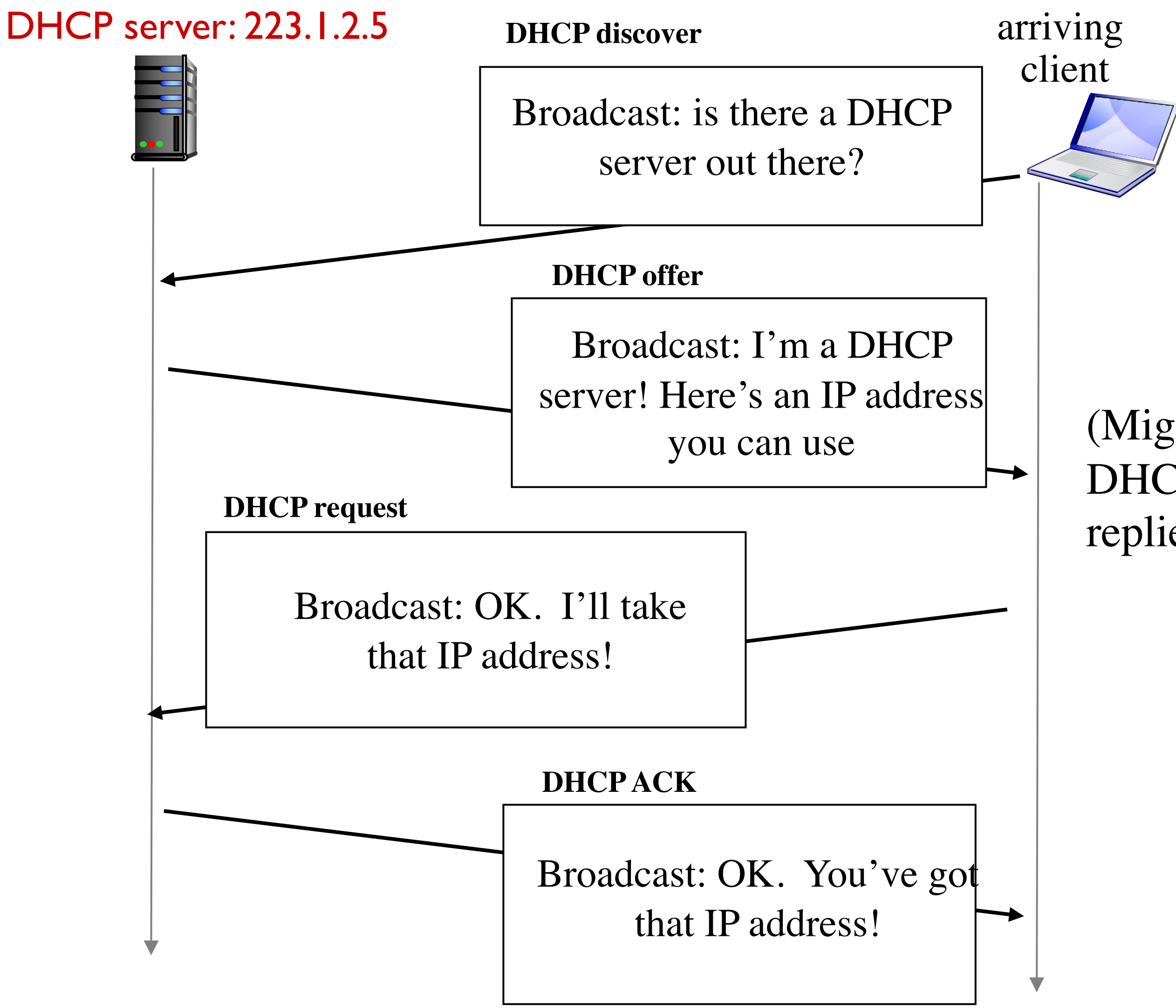
- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



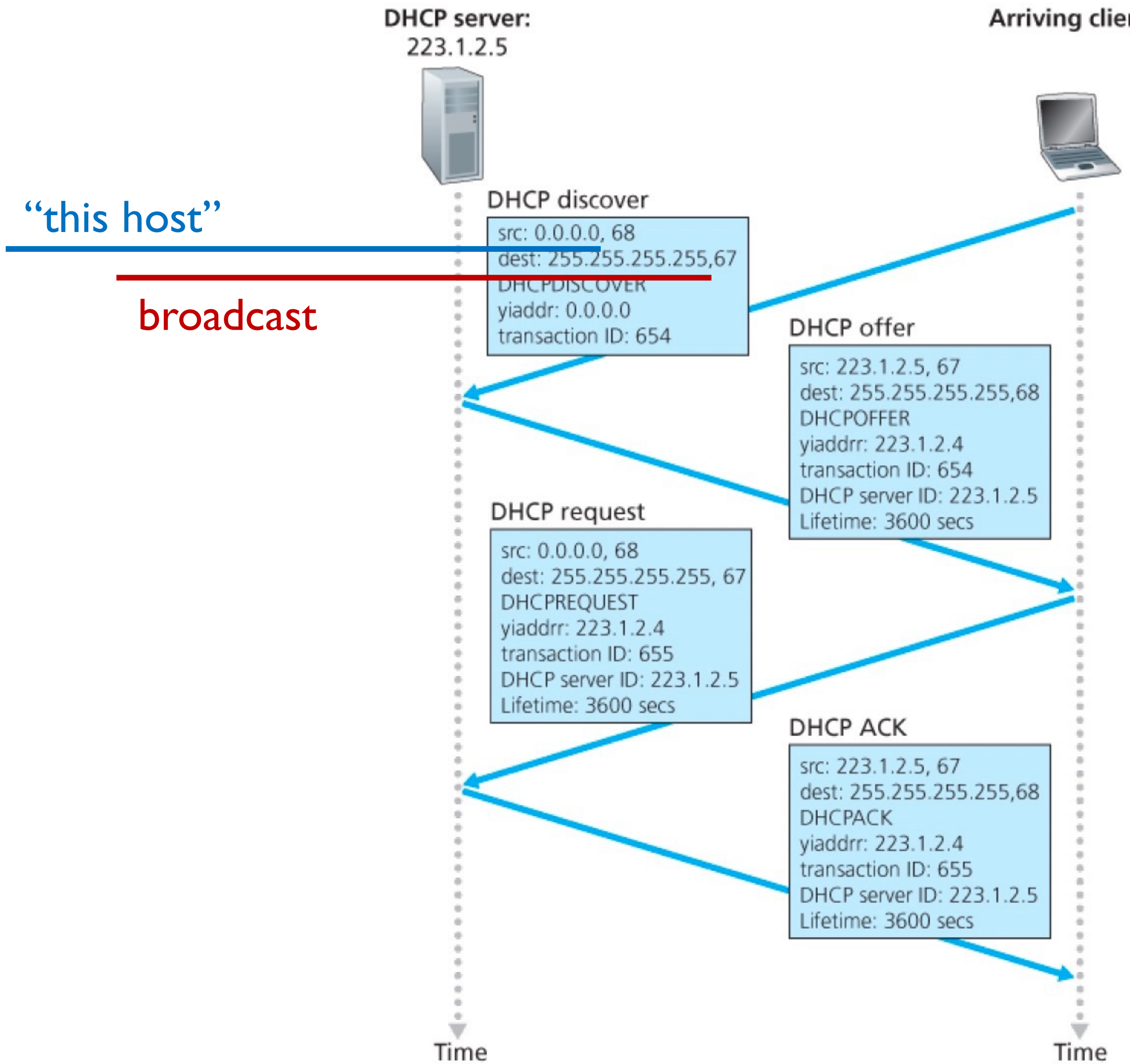
arriving DHCP client needs address in this network

DHCP client-server scenario



(Might be multiple DHCP servers who replies)

DHCP client-server scenario



UDP
Broadcast
Port number: 67

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 drawback:

- A new IP address is obtained each time a node connects to a new subnet
- Mobile devices

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