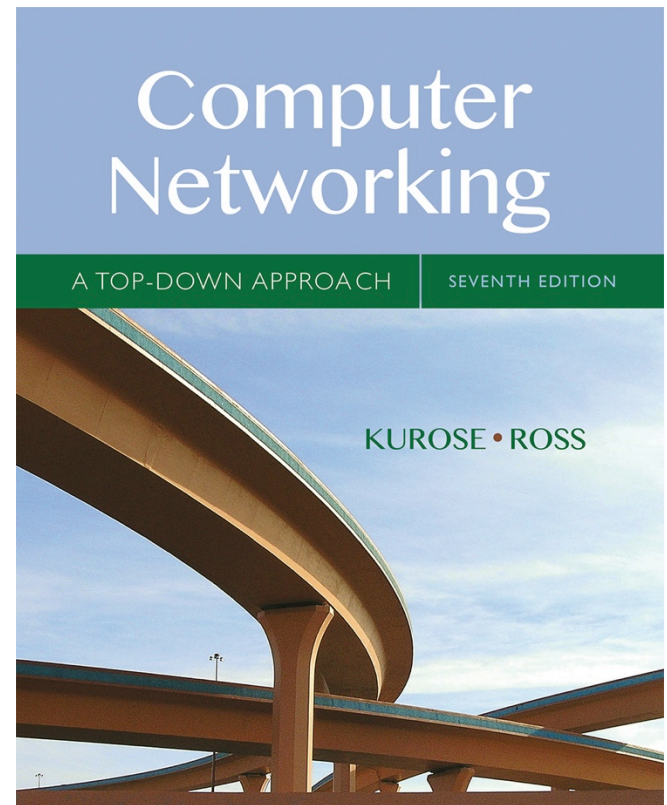# Chapter 4 (Review) Network Layer: The Data Plane

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.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

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
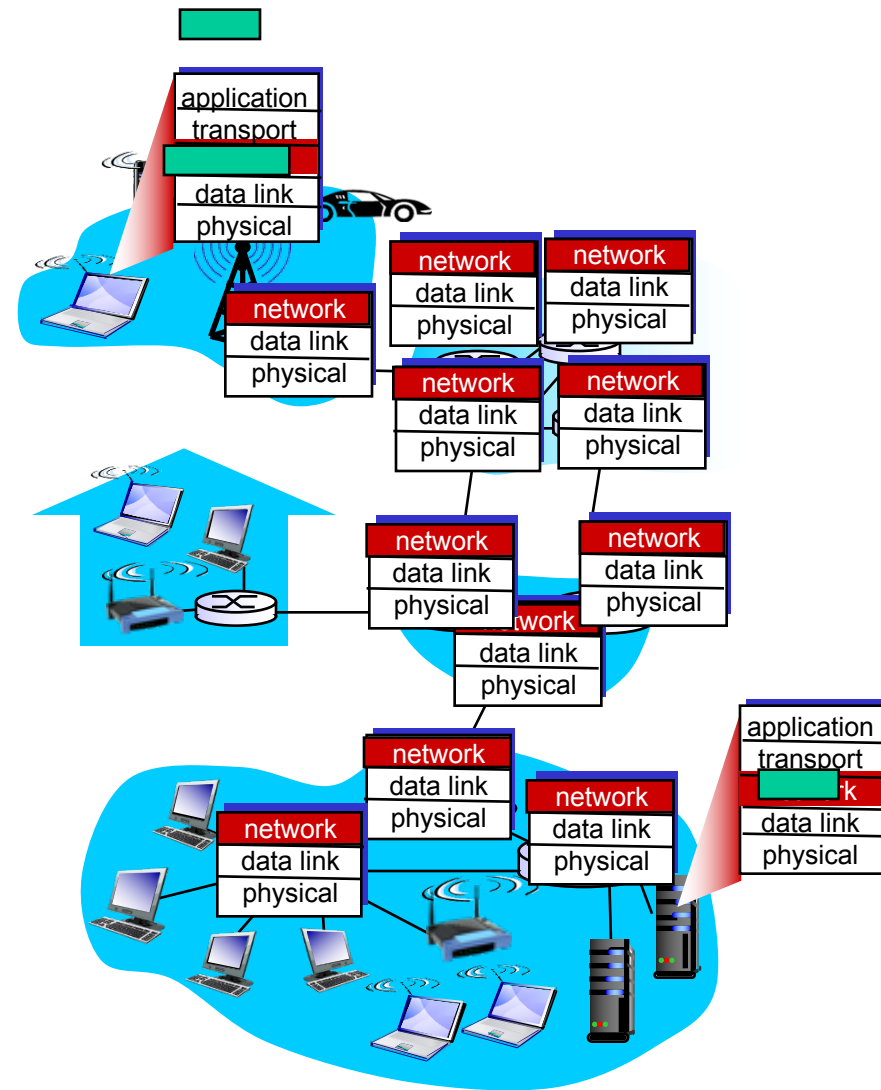
Two key network-layer functions

network-layer functions:
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
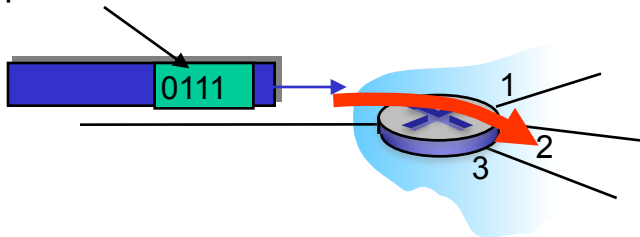 - *routing algorithms*

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

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
- forwarding function

values in arriving
packet header

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

Chapter 4: outline

Router architecture overview

- high-level view of generic router architecture:
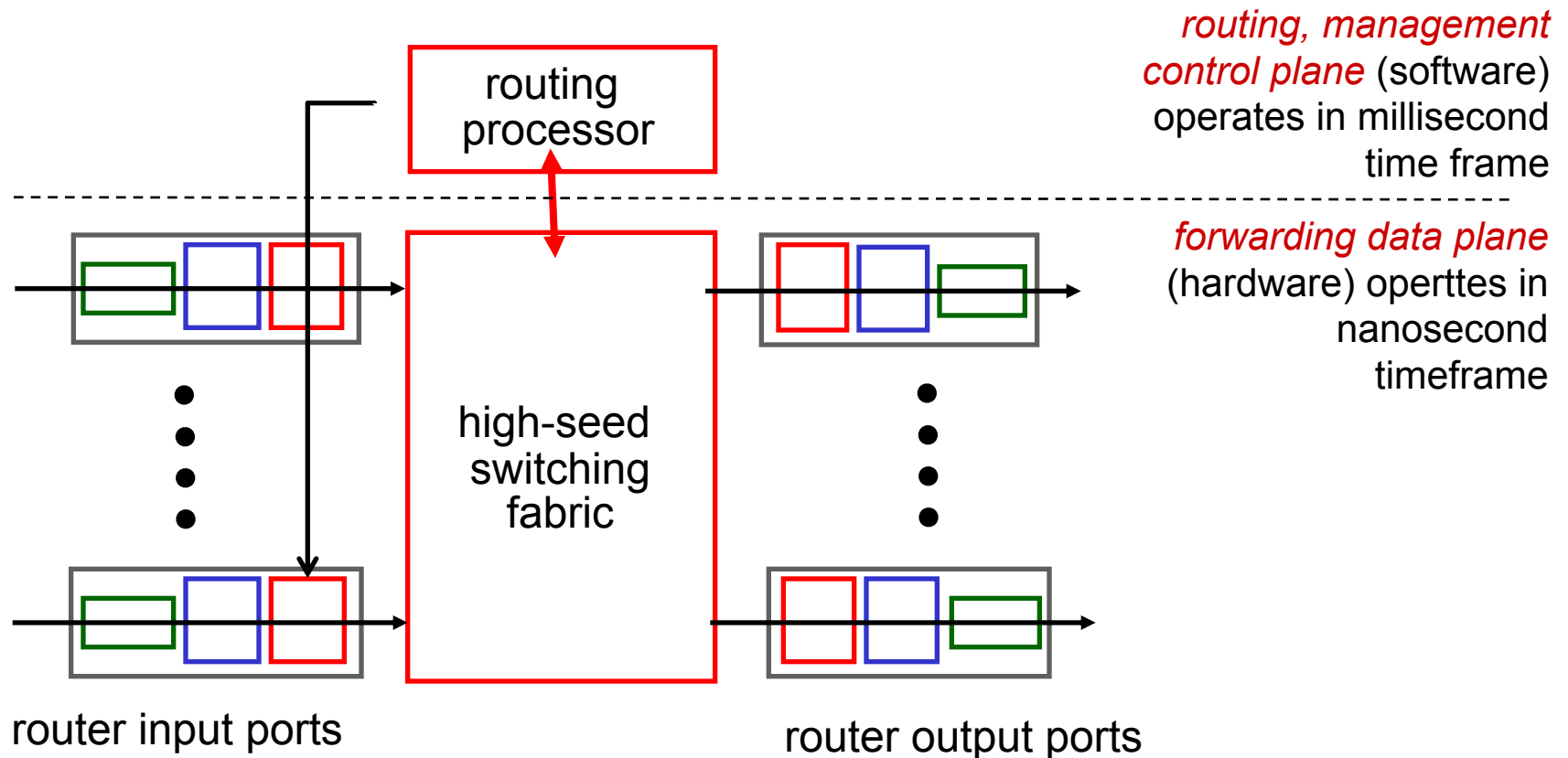
routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operttes in nanosecond timeframe

routing processor

high-seed switching fabric

router input ports

router output ports

Destination-based forwarding

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 最长前缀匹配

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?

Scheduling mechanisms

- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *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

packet
arrives

queue
(waiting area)

link
(server)
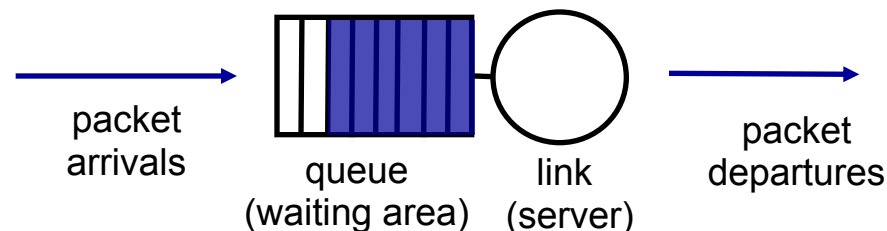
packet
departures

Scheduling policies: priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
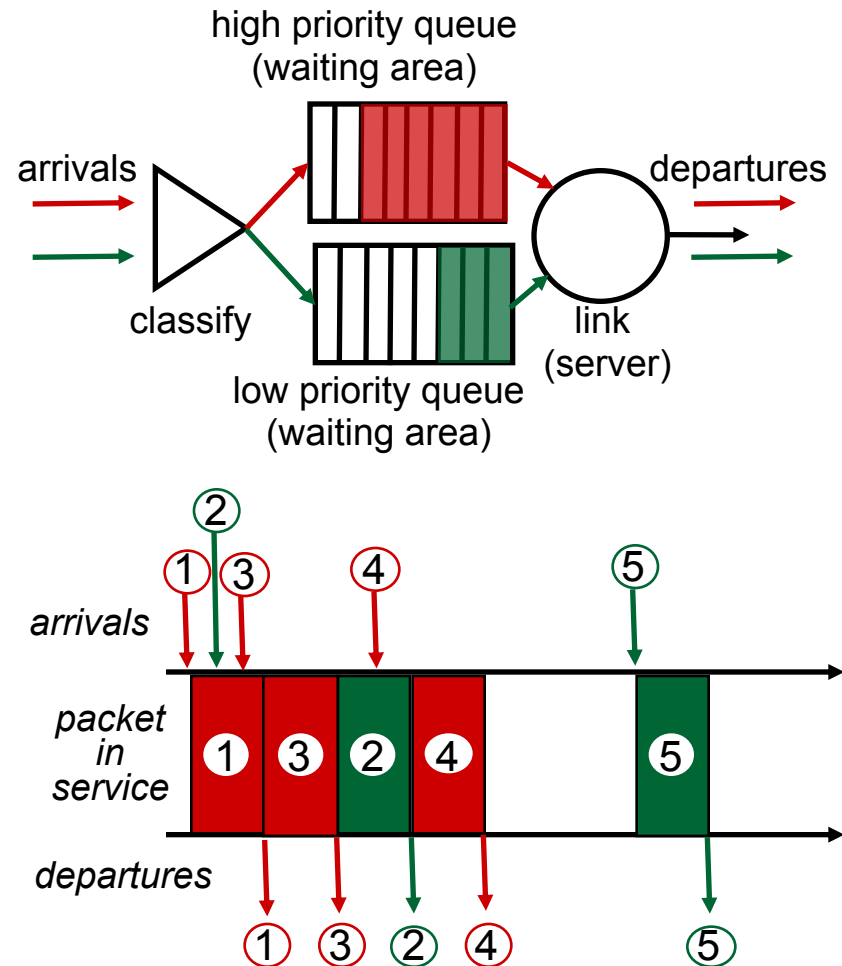 - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?

high priority queue (waiting area)

arrivals

classify

low priority queue (waiting area)

departures

link (server)

arrivals

packet in service

departures

Scheduling policies: still more

Round Robin (RR) scheduling:

- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?
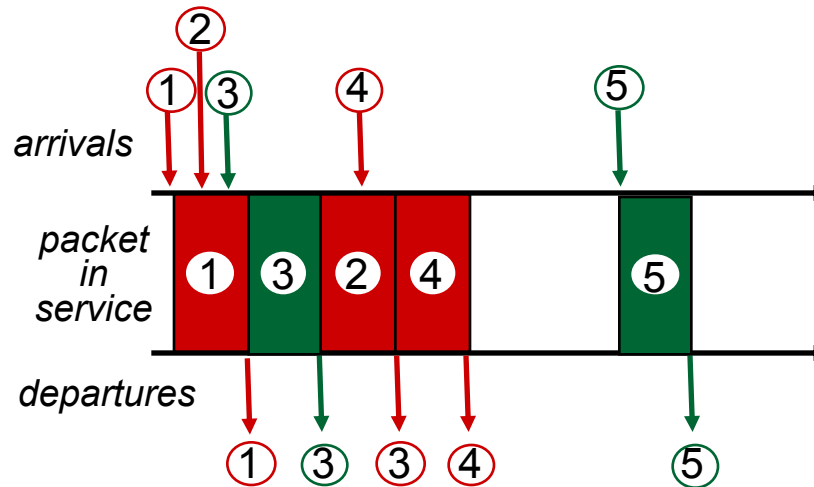
Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?
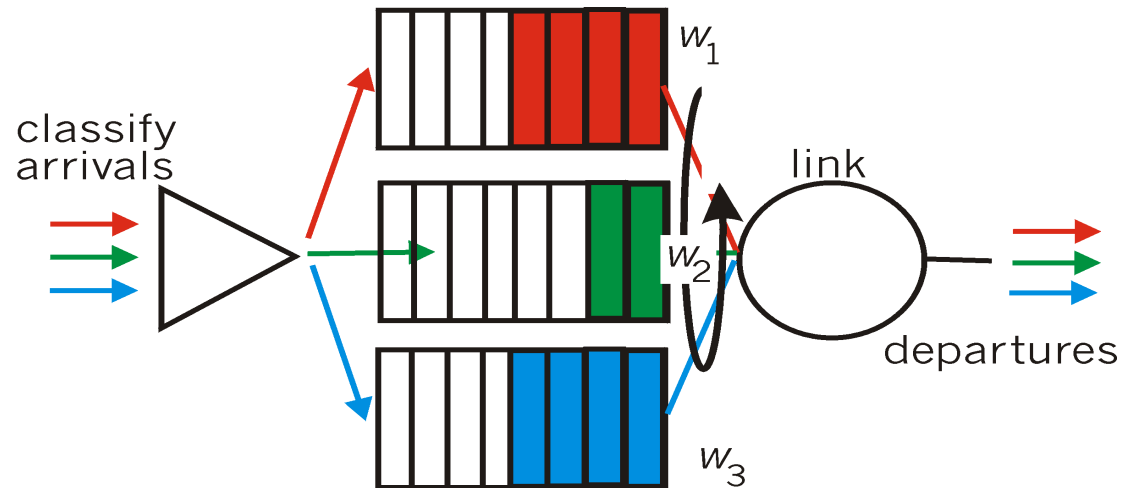
Chapter 4: outline

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:

network
layer

transport layer: TCP, UDP

routing protocols
• path selection
• RIP, OSPF, BGP

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

forwarding
table

ICMP protocol
• error reporting
• router
"signaling"

link layer

physical layer

IP datagram format

IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

32 bits

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

total datagram length (bytes)

for fragmentation/ reassembly

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

how much overhead?
- ❖ 20 bytes of TCP
- ❖ 20 bytes of IP
- ❖ = 40 bytes + app layer overhead

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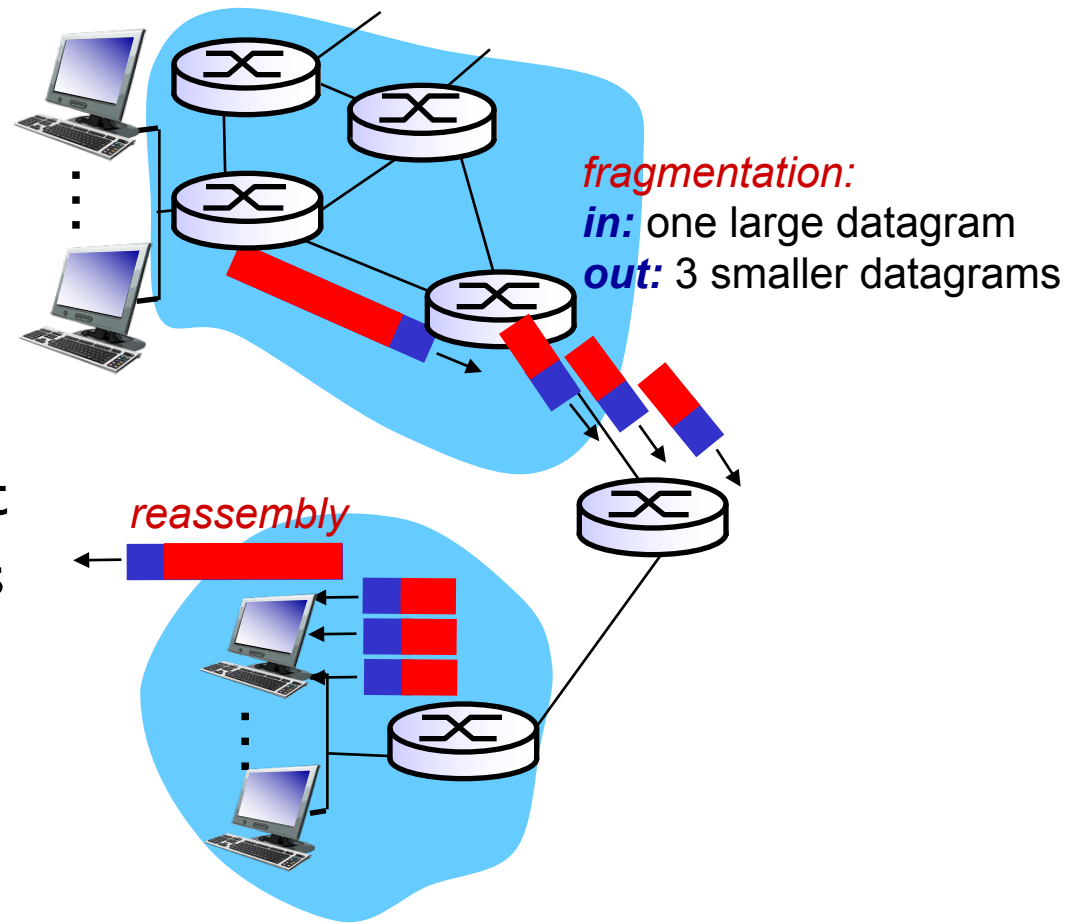
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order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

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

reassembly

IP fragmentation, reassembly

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

offset = 1480/8

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

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

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

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane
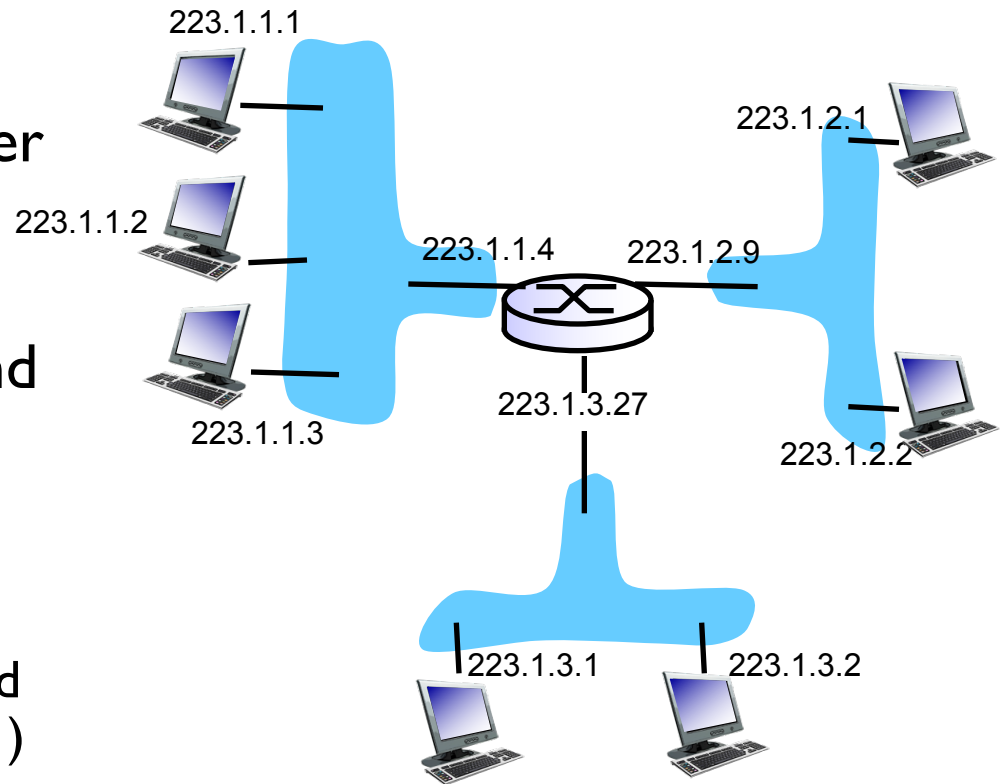
4.2 What's inside a router

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

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

223.1.1.1 = 11011111 00000001 00000001 00000001
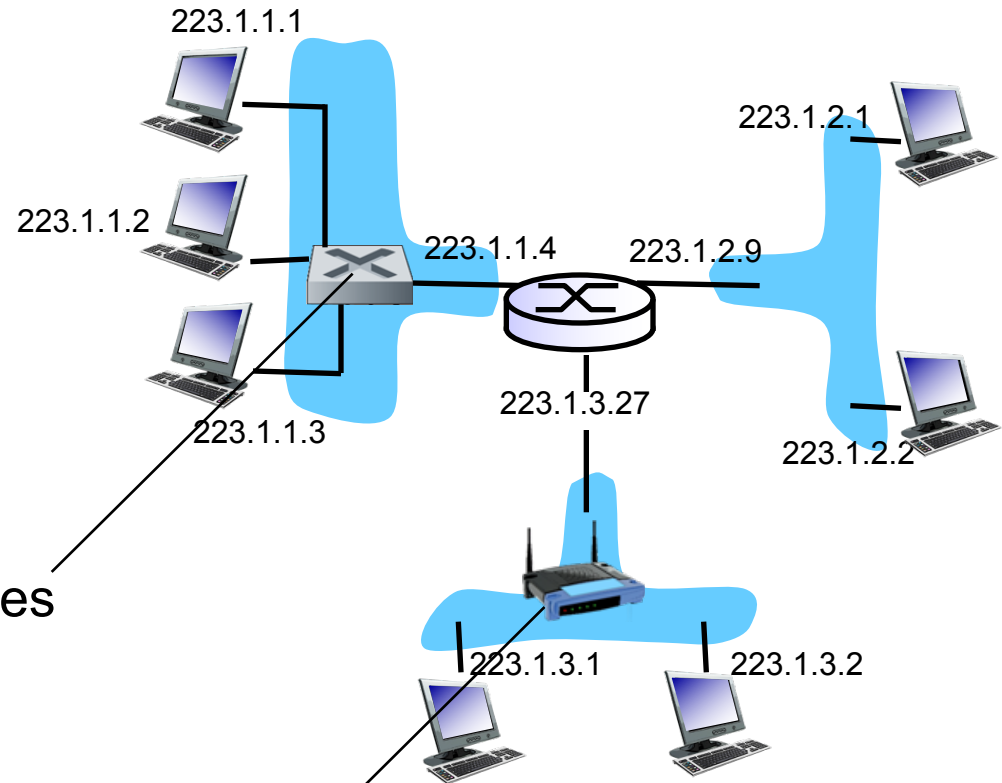
223   1   1   1

IP addressing: introduction

Q: how are interfaces actually connected?

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

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)

223.1.1.1

223.1.1.2

223.1.1.4

223.1.2.1

223.1.2.9

223.1.1.3

223.1.2.2

223.1.3.27

223.1.3.1

223.1.3.2

A: wireless WiFi interfaces connected by WiFi base station

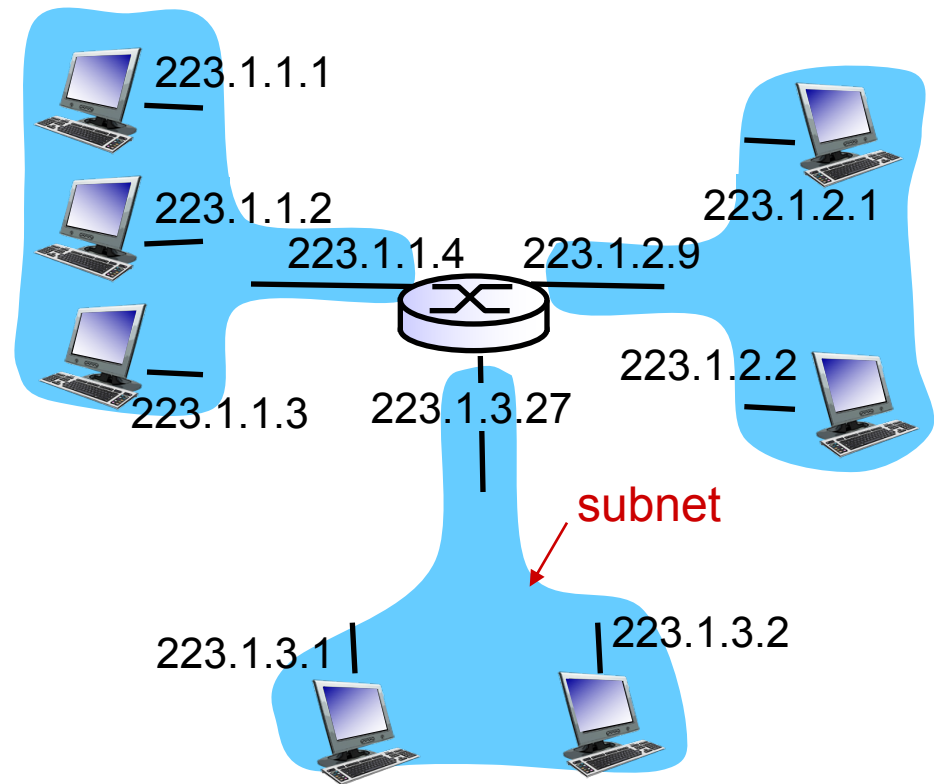# Subnets

- **IP address:**
  - subnet part - high order bits
  - host part - low order bits
- *what's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.2.1

223.1.2.2

223.1.1.3   223.1.3.27

subnet

223.1.3.1   223.1.3.2

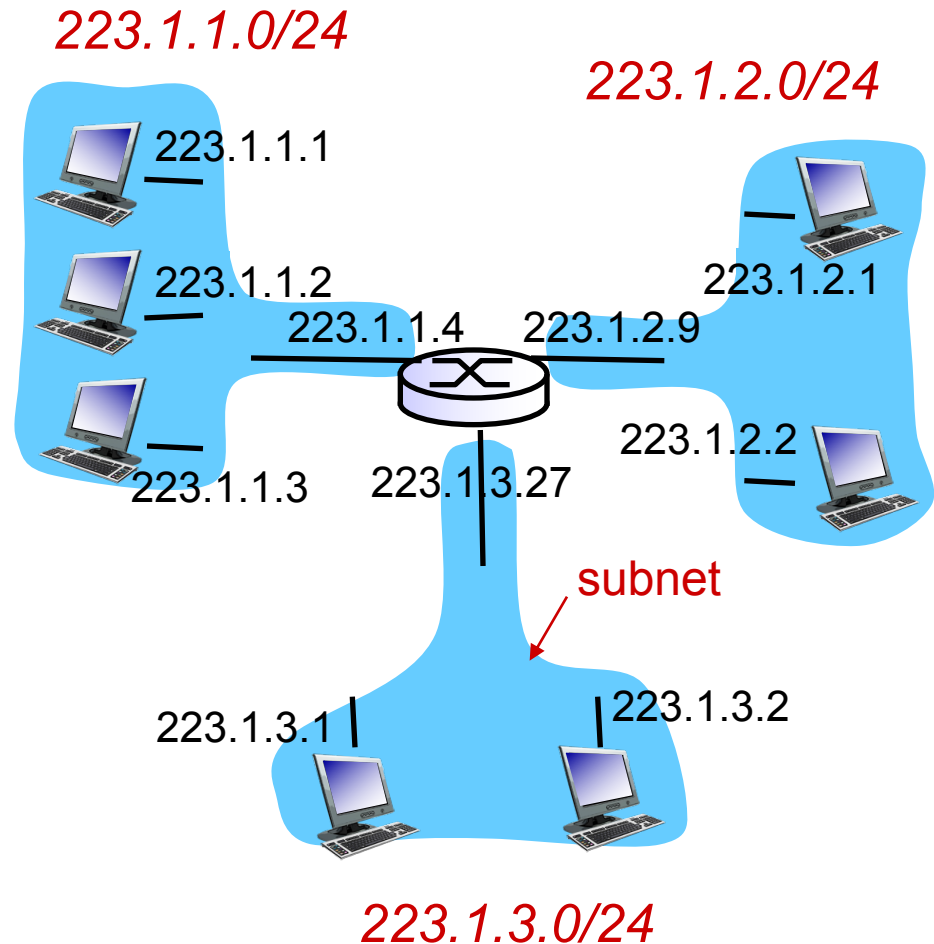network consisting of 3 subnets

Subnets

recipe

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

- each isolated network is called a *subnet*

223.1.1.0/24

223.1.2.0/24

223.1.1.1

223.1.1.2

223.1.1.4   223.1.2.9

223.1.2.1

223.1.2.2

223.1.1.3   223.1.3.27

subnet

223.1.3.1   223.1.3.2

223.1.3.0/24

subnet mask: /24

Subnets

how many?

223.1.1.2

223.1.1.1

223.1.1.4

223.1.1.3

223.1.9.2   223.1.7.0

223.1.9.1   223.1.7.1

223.1.8.1   223.1.8.0

223.1.2.6   223.1.3.27

223.1.2.1   223.1.2.2   223.1.3.1   223.1.3.2
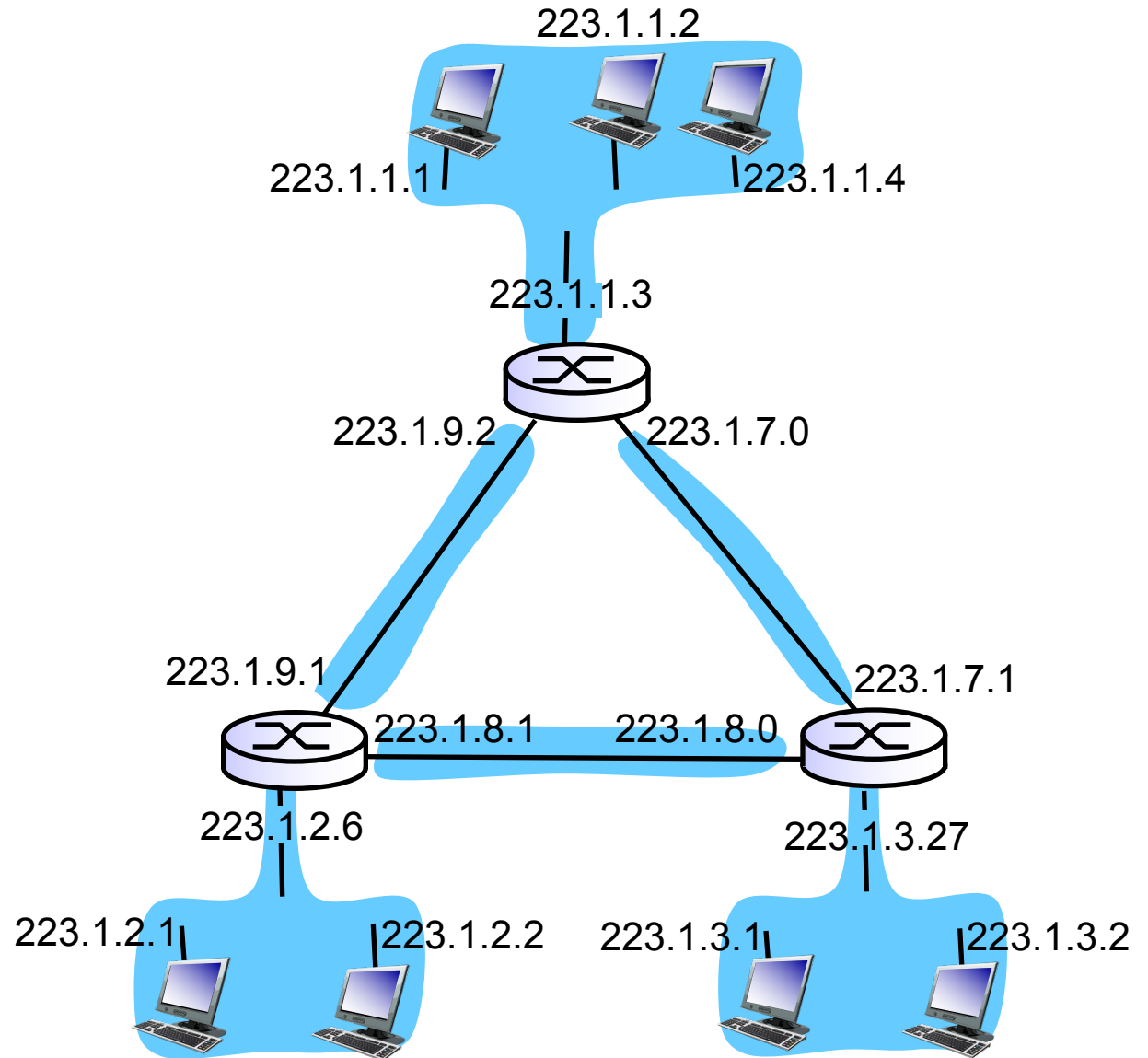
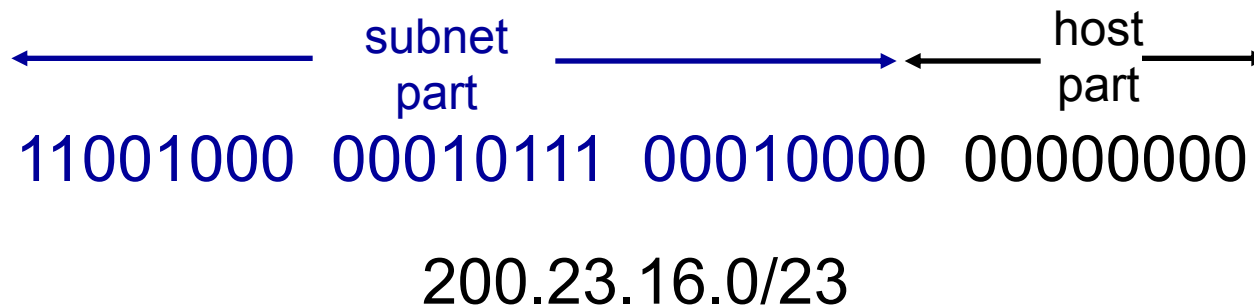# IP addressing: CIDR

CIDR: **C**lassless **I**nter**D**omain **R**outing
- 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

IP addresses: how to get one?

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

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"
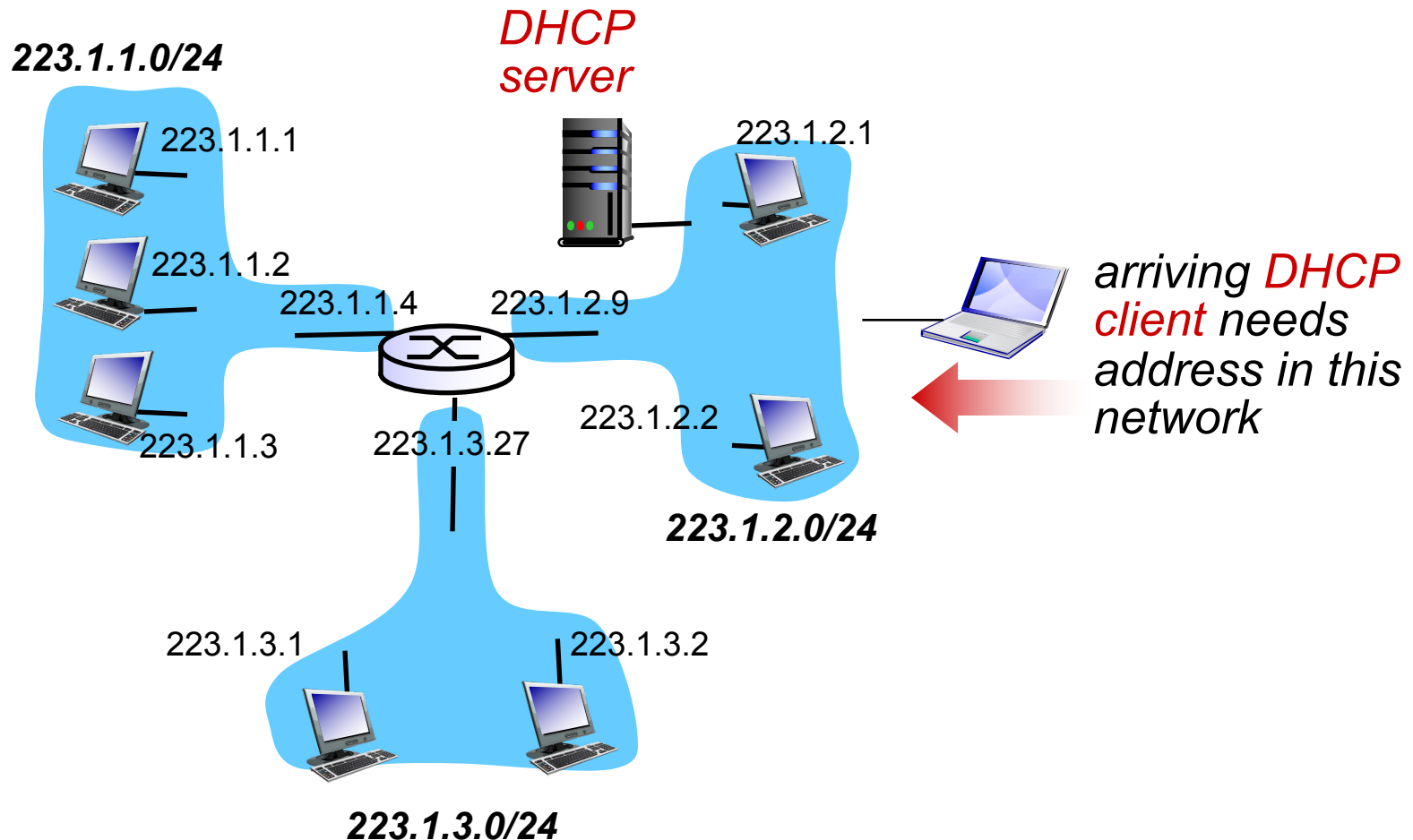
DHCP: Dynamic Host Configuration Protocol

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

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

DHCP overview:

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

DHCP client-server scenario

223.1.1.0/24

DHCP server

223.1.1.1

223.1.2.1

223.1.1.2

223.1.1.4   223.1.2.9

arriving DHCP client needs address in this network

223.1.1.3   223.1.3.27   223.1.2.2

223.1.2.0/24

223.1.3.1   223.1.3.2

223.1.3.0/24

DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

Broadcast: is there a DHCP server out there?

arriving client

DHCP offer

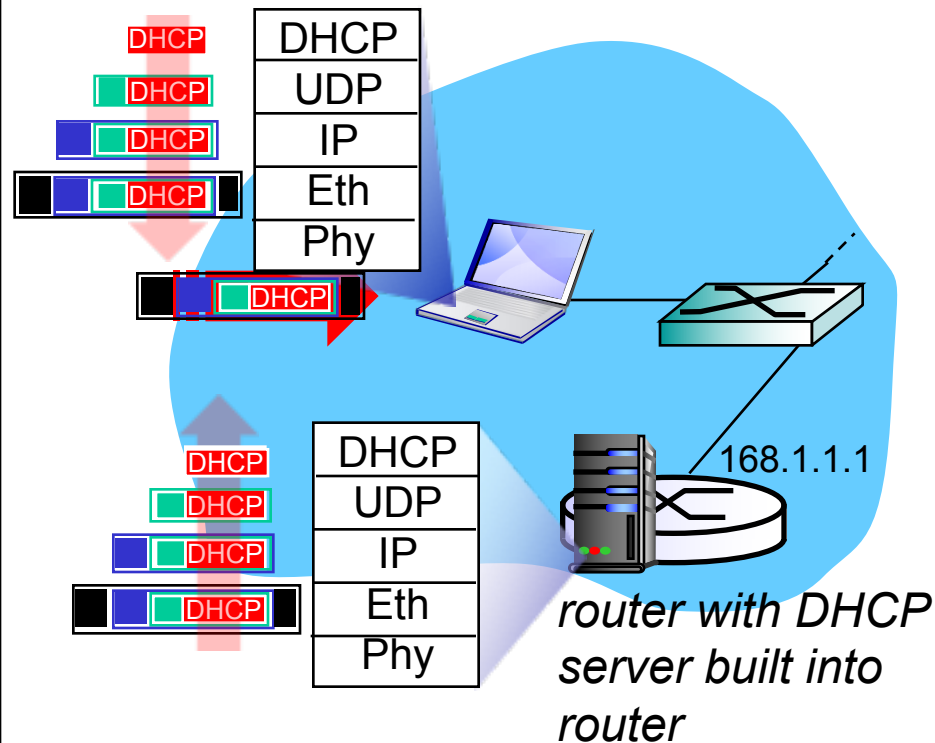Broadcast: I'm a DHCP server! Here's an IP address you can use

DHCP request

Broadcast: OK. I'll take that IP address!

DHCP ACK

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

DHCP: example

router with DHCP server built into router

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

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet

- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example

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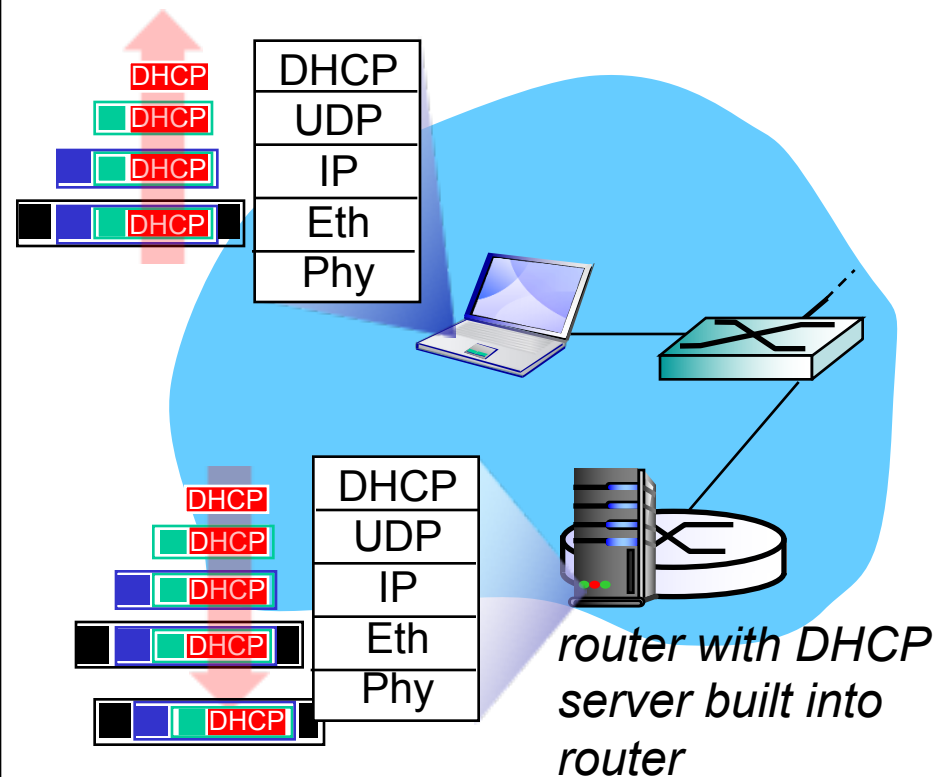
- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client

- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router
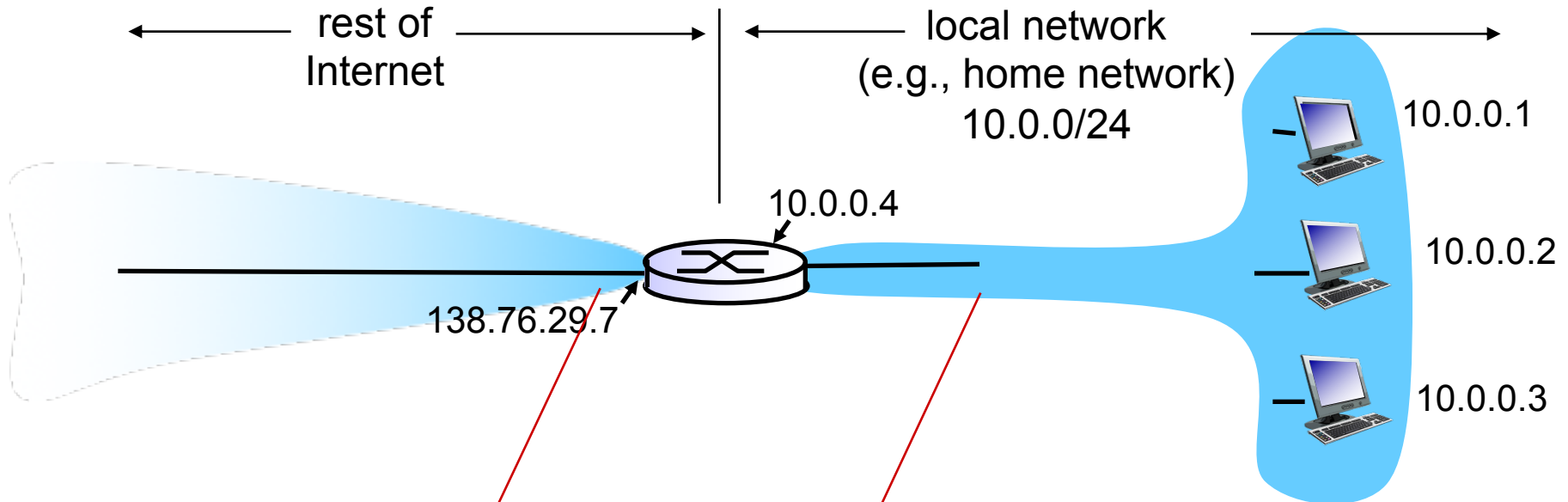
IP addressing: the last word...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned

Names and Numbers http://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

NAT: network address translation

rest of
Internet

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

10.0.0.1

10.0.0.2

10.0.0.3

10.0.0.4

138.76.29.7

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)

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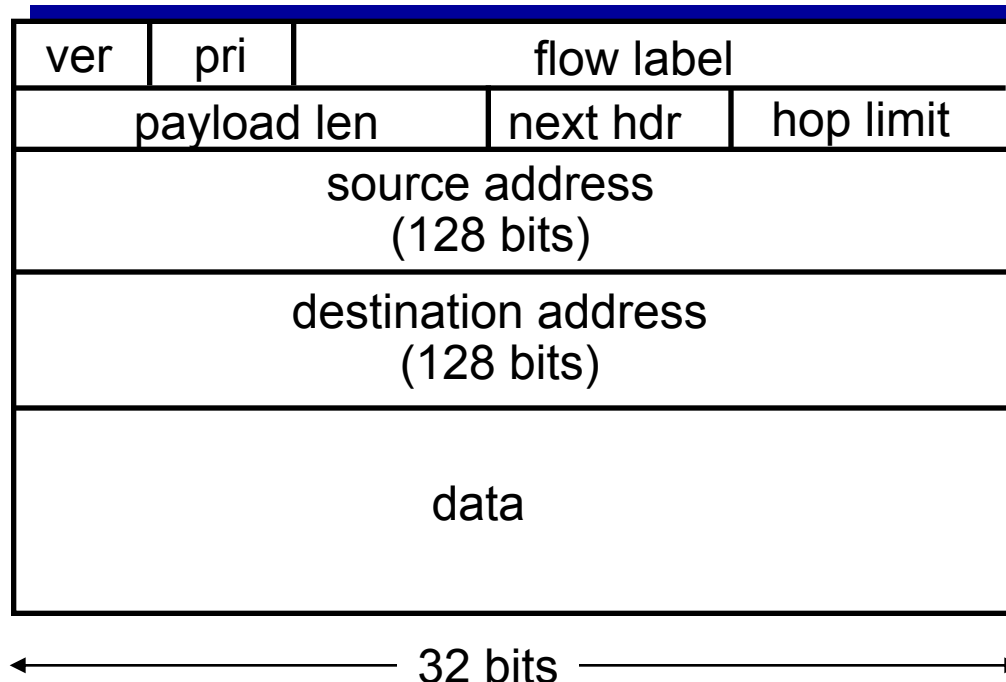
ver	pri	flow label		
payload len			next hdr	hop limit
source address (128 bits)				
destination address (128 bits)				
data				

←——————— 32 bits ———————→

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane
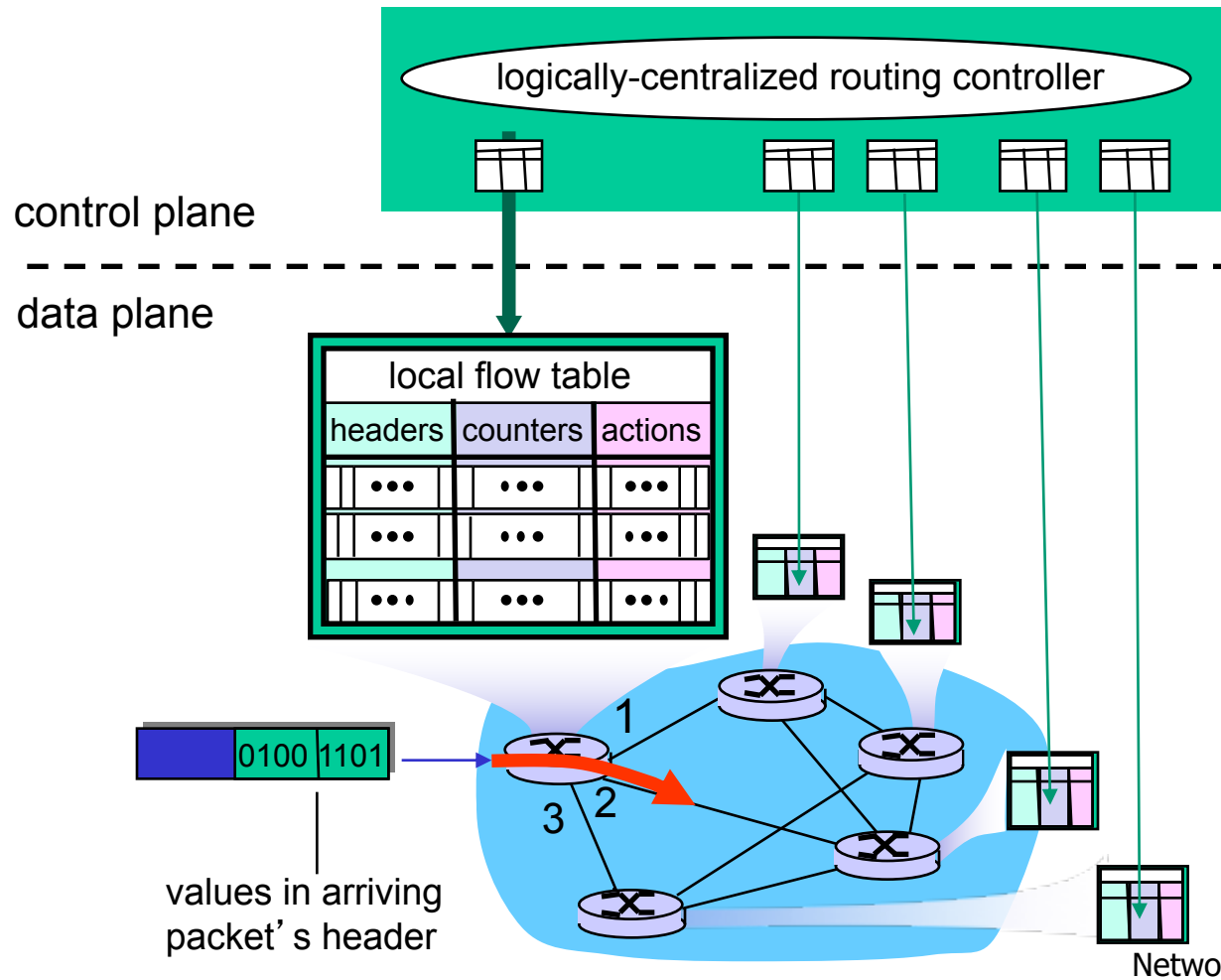
4.2 What's inside a router

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

Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller

logically-centralized routing controller

control plane

data plane

local flow table

headers	counters	actions
...	...	...
...	...	...
...	...	...

1

0100 1101

3 2

values in arriving
packet's header

OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets
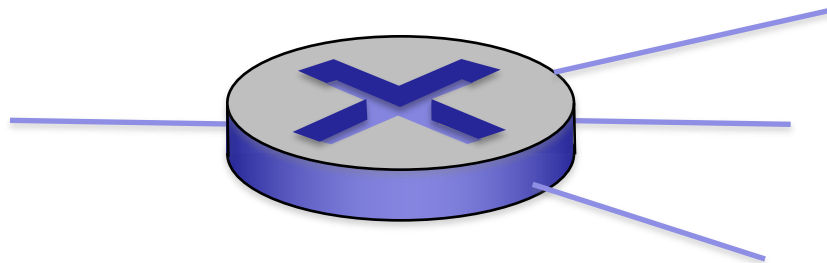
Flow table in a router (computed and distributed by controller) define router's match+action rules
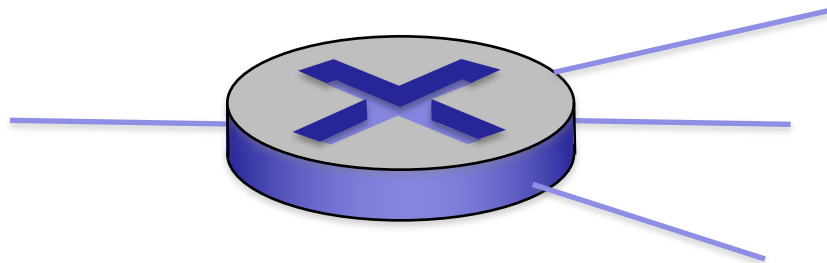
OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets

* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries

Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Link layer   Network layer   Transport layer
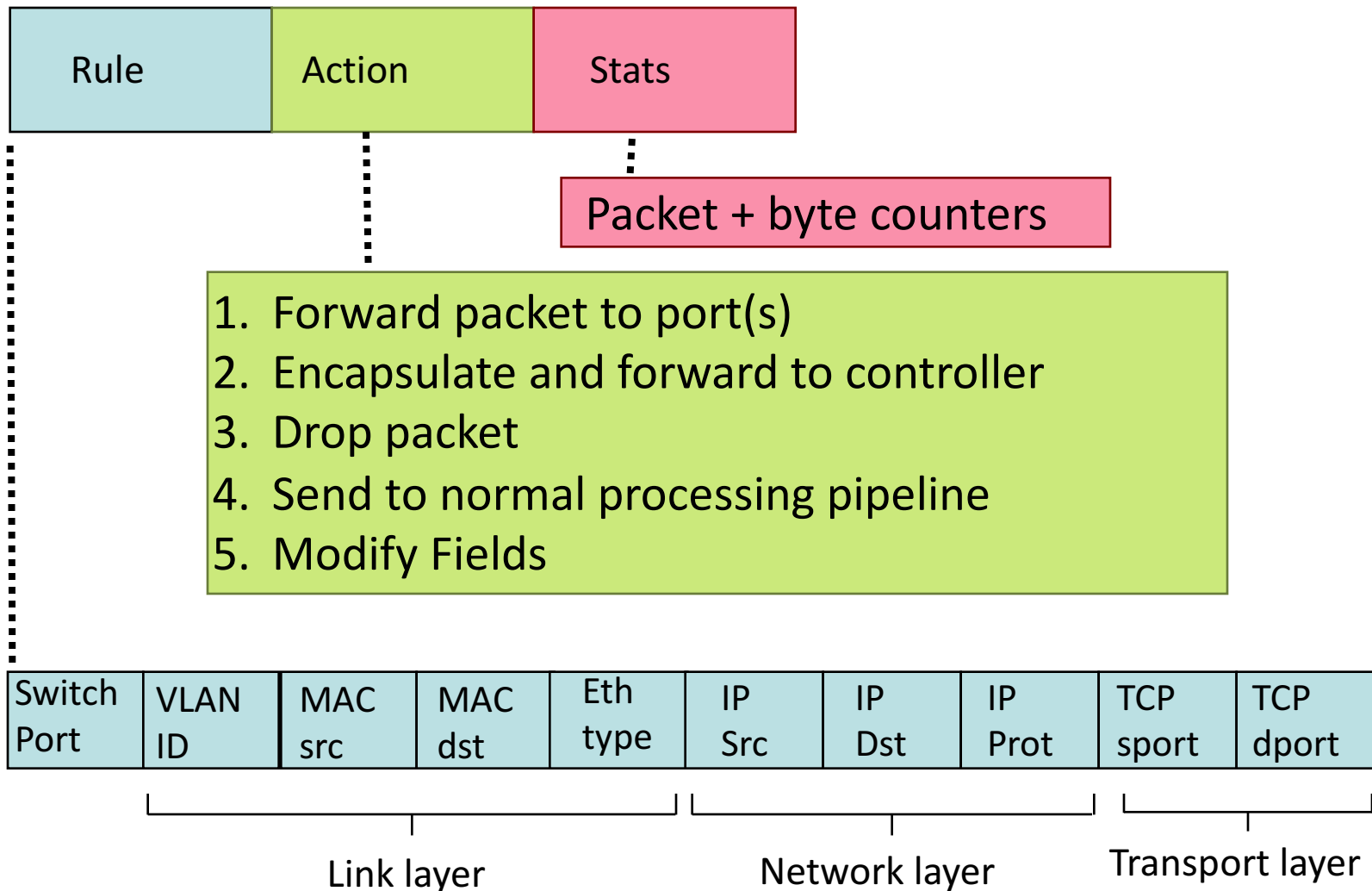
Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02
should be forwarded to output port 6

OpenFlow abstraction

- *match+action:* unifies different kinds of devices

- Router
  - *match:* longest destination IP prefix
  - *action:* forward out a link
- Switch
  - *match:* destination MAC address
  - *action:* forward or flood

- Firewall
  - *match*: IP addresses and TCP/UDP port numbers
  - *action:* permit or deny
- NAT
  - *match:* IP address and port
  - *action:* rewrite address and port

Chapter 4
end