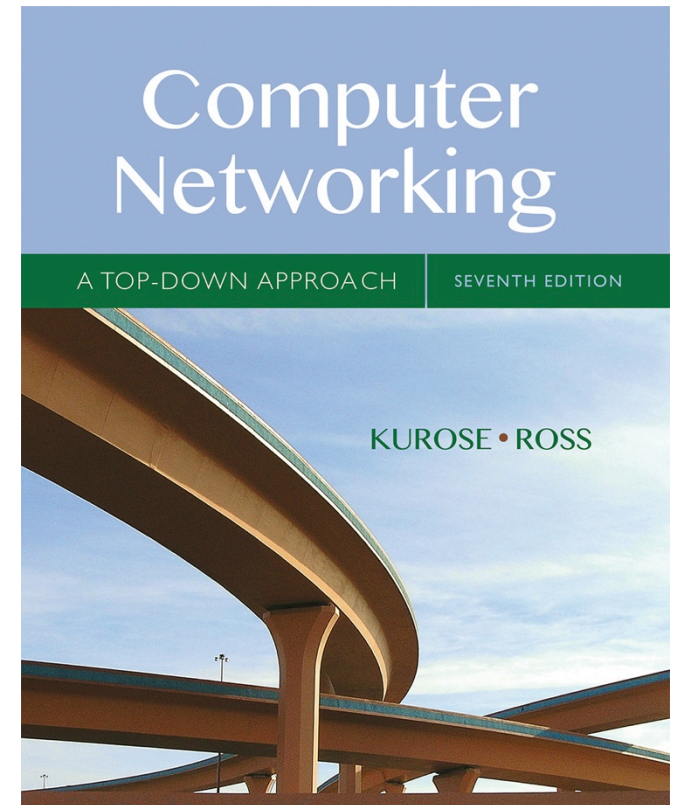


Chapter 4

Network Layer: The Data Plane

All material copyright 1996-2016
© J.F Kurose and K.W. Ross, All Rights Reserved



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 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.3 What's inside a router

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

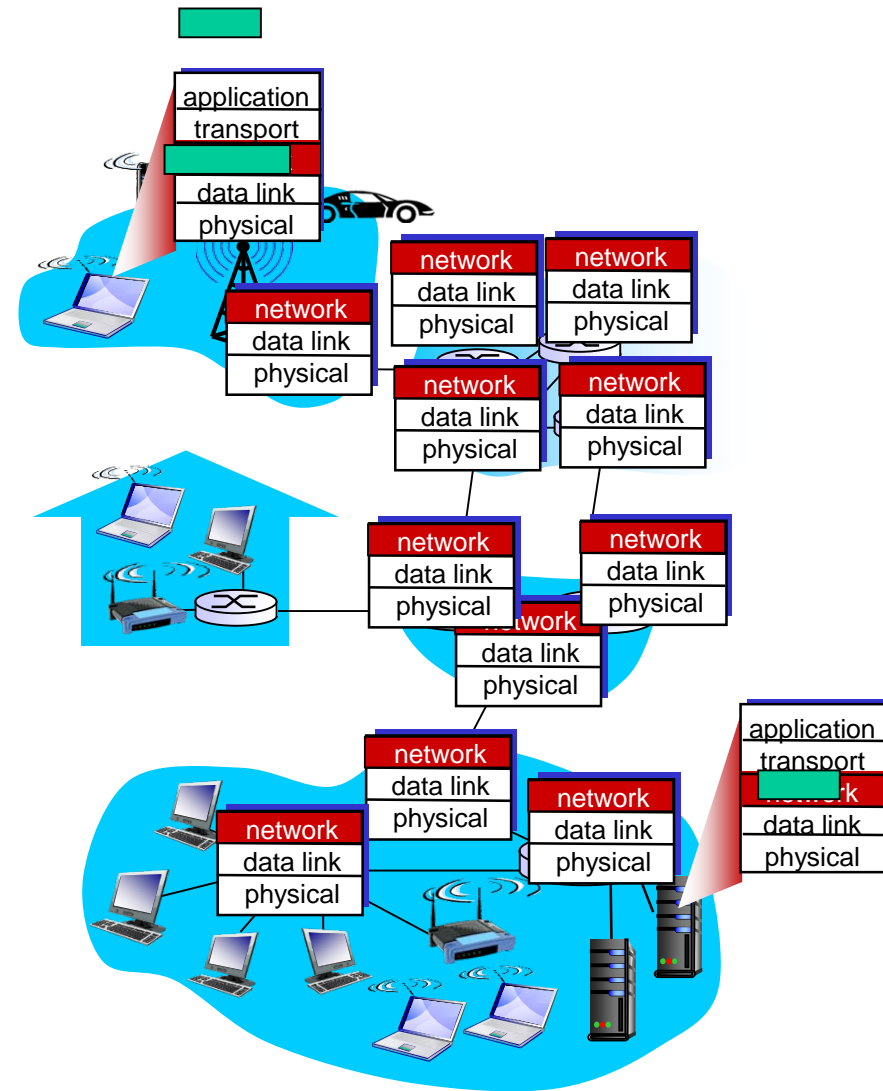
Chapter 4: network layer

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
 - IP Protocol
 - NAT

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



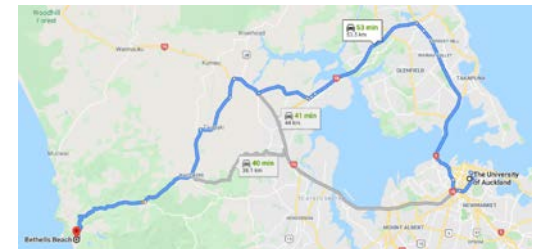
Two key network-layer functions

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

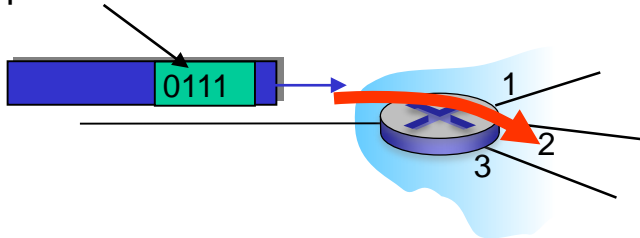


Network layer: data plane, control plane

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

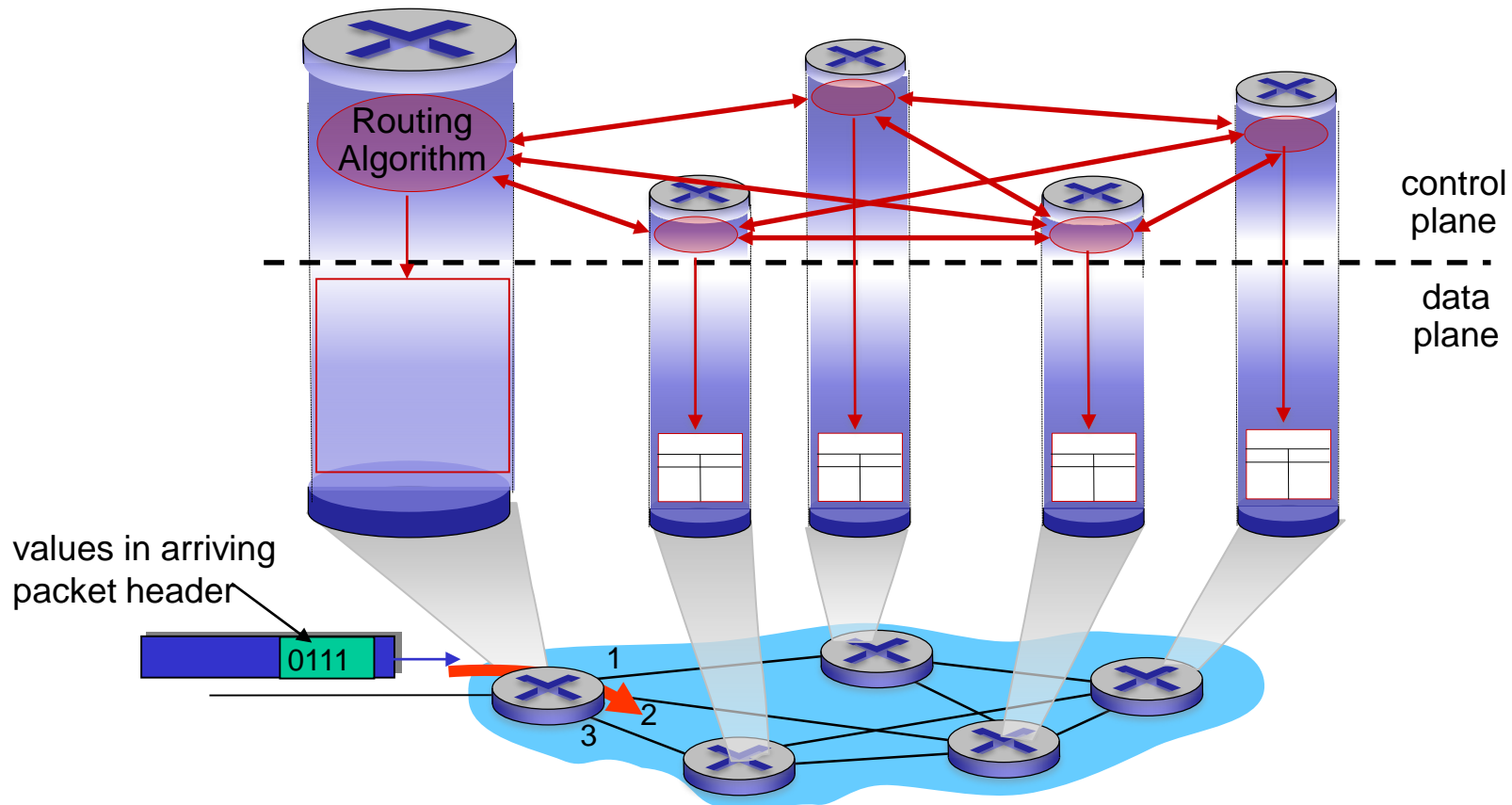


Control plane

- 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

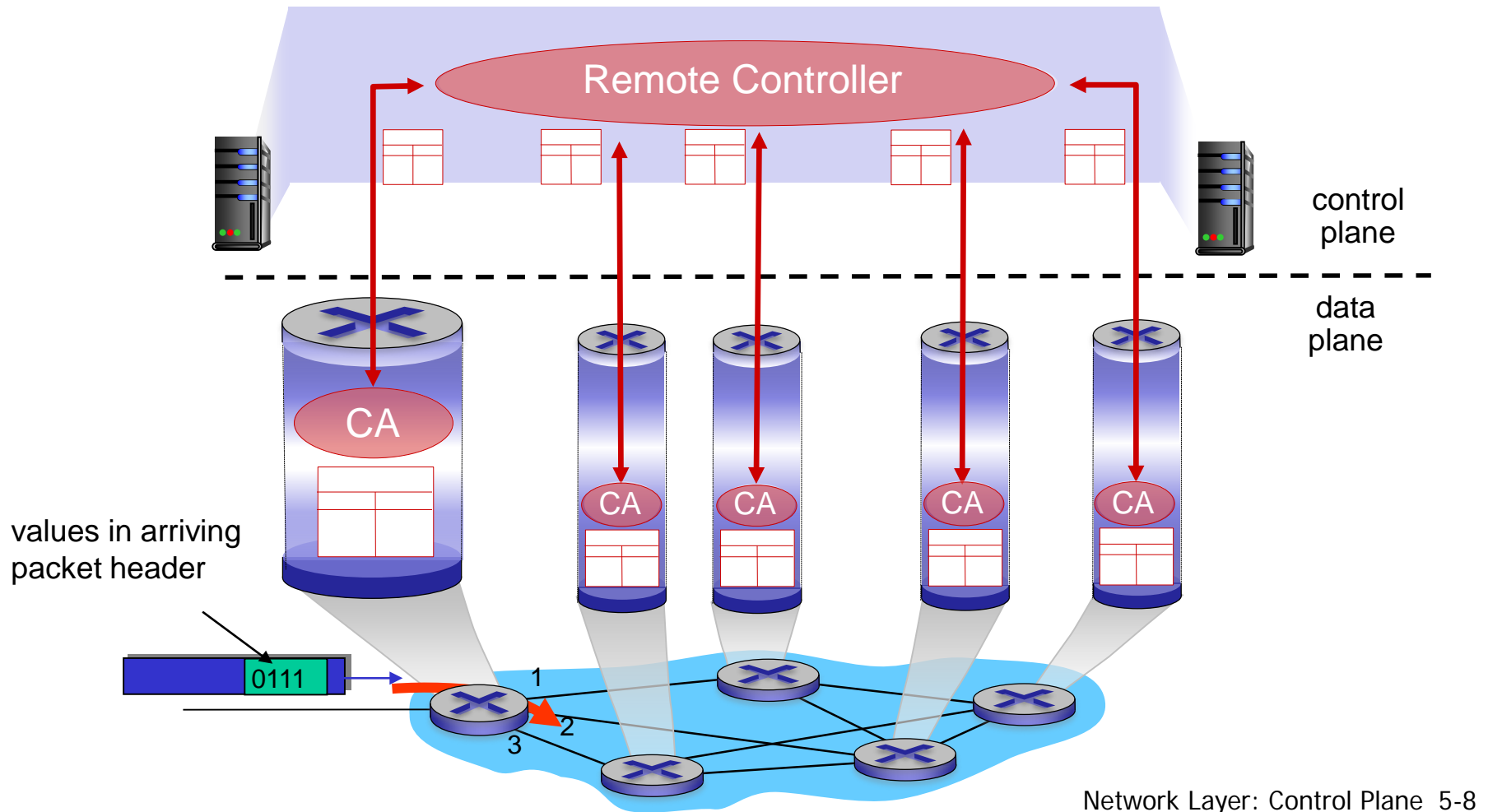
Per-router control plane

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



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



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 models:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

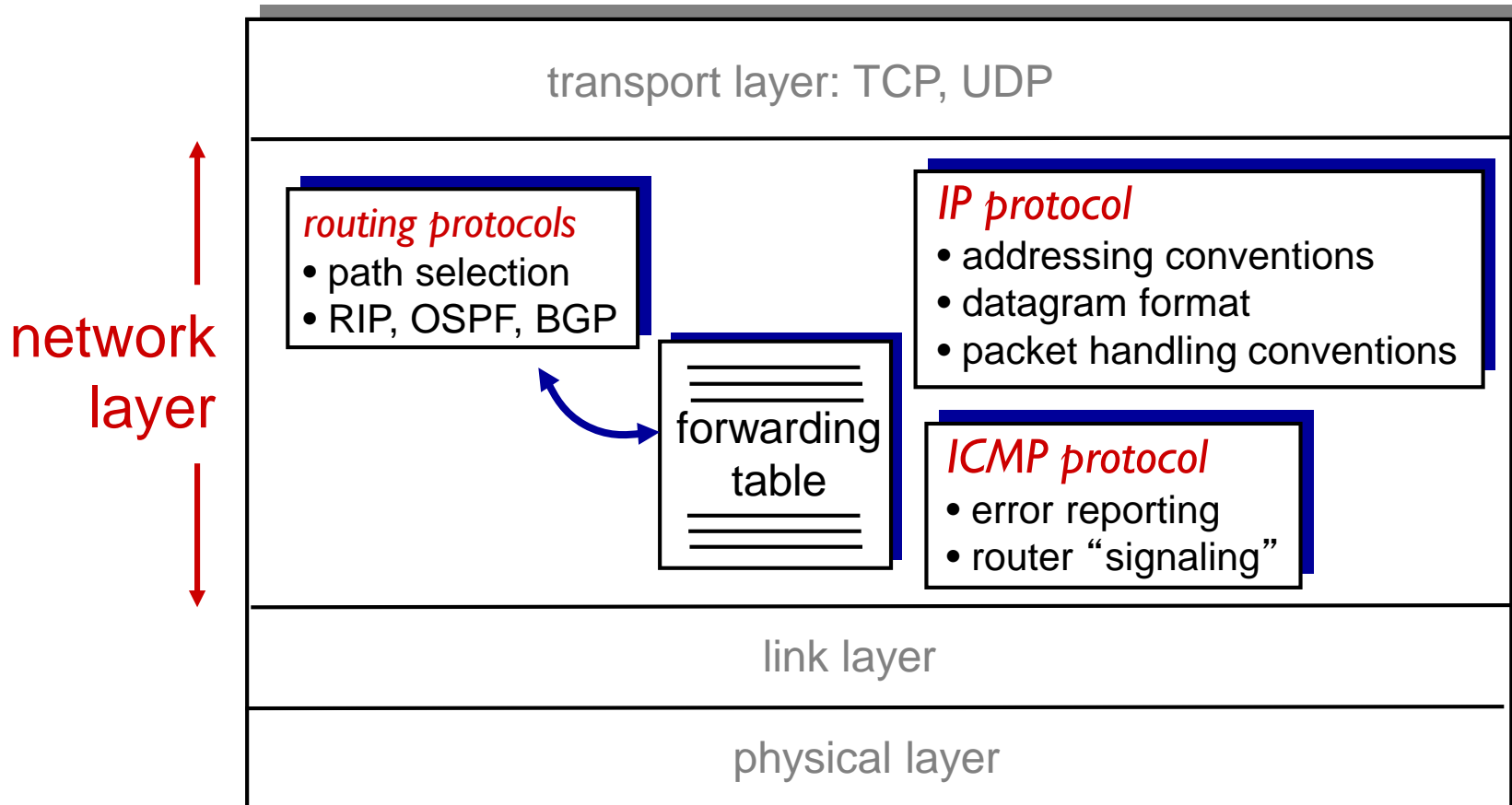
4.3 What's inside a router

4.4 Generalized Forward and SDN

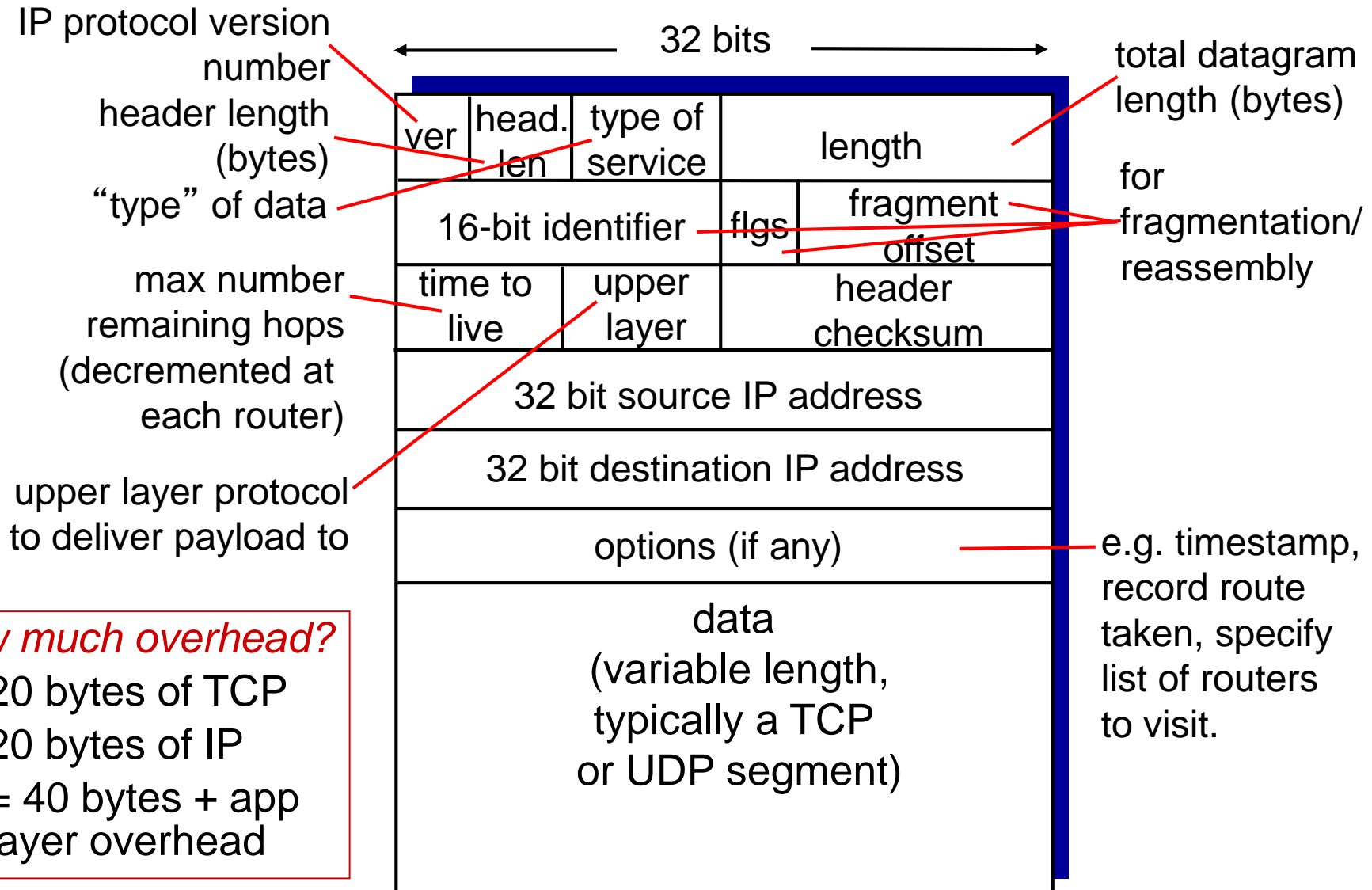
- match
- action
- OpenFlow examples of match-plus-action in action

The Internet network layer

host, router network layer functions:

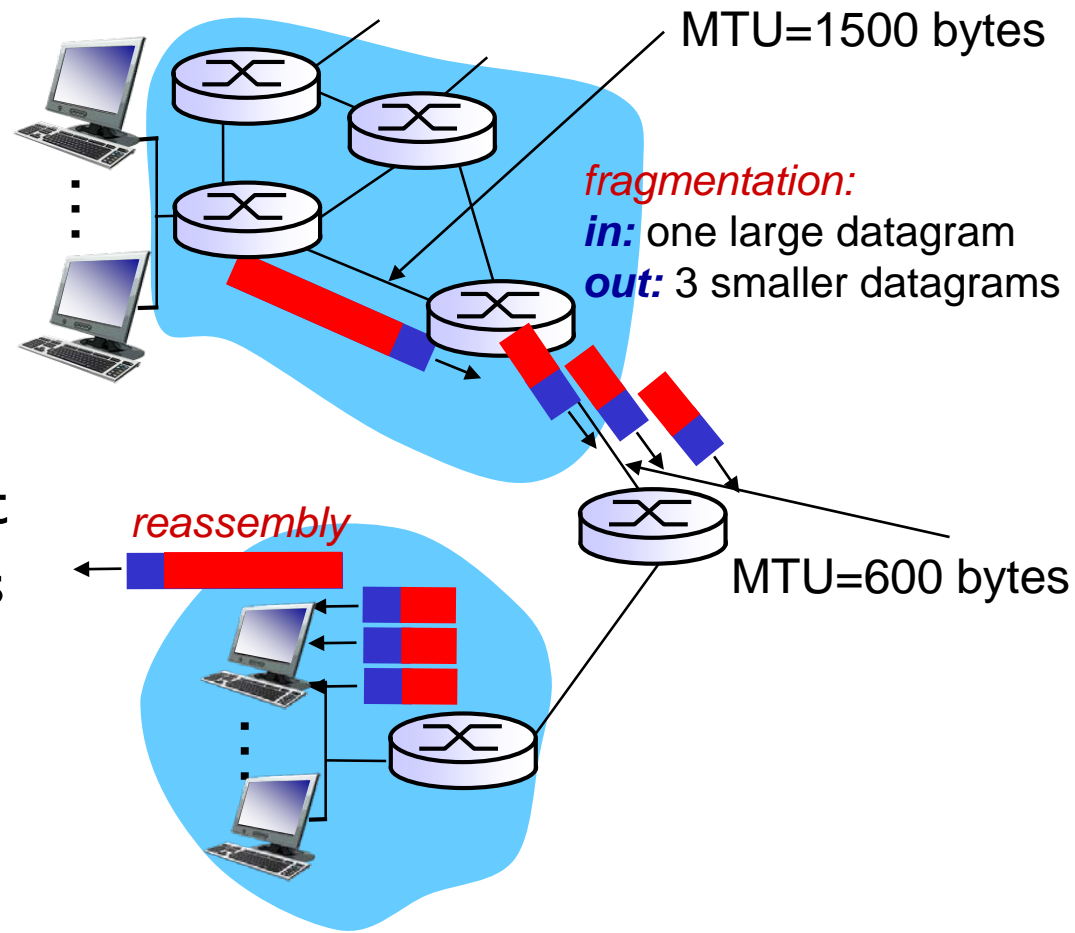


IP datagram format



IP fragmentation, reassembly

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



IP fragmentation, reassembly

example:

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

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

*one large datagram becomes
several smaller datagrams*

1480 bytes in
data field

offset =
 $1480/8$

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

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

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

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

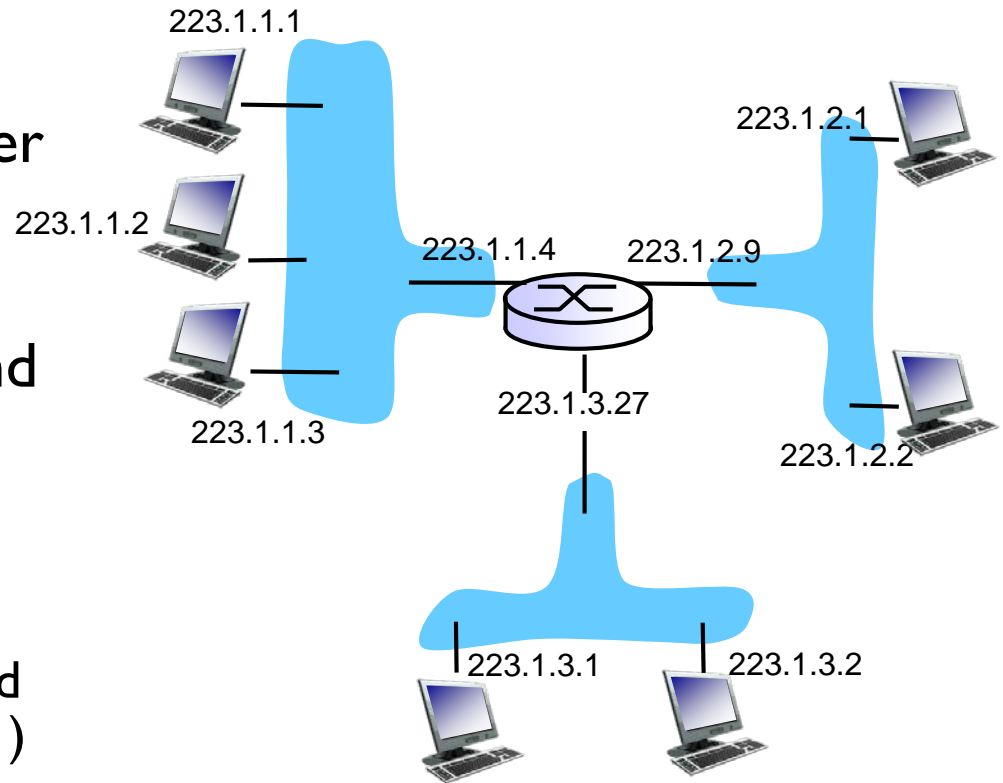
4.3 What's inside a router

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**



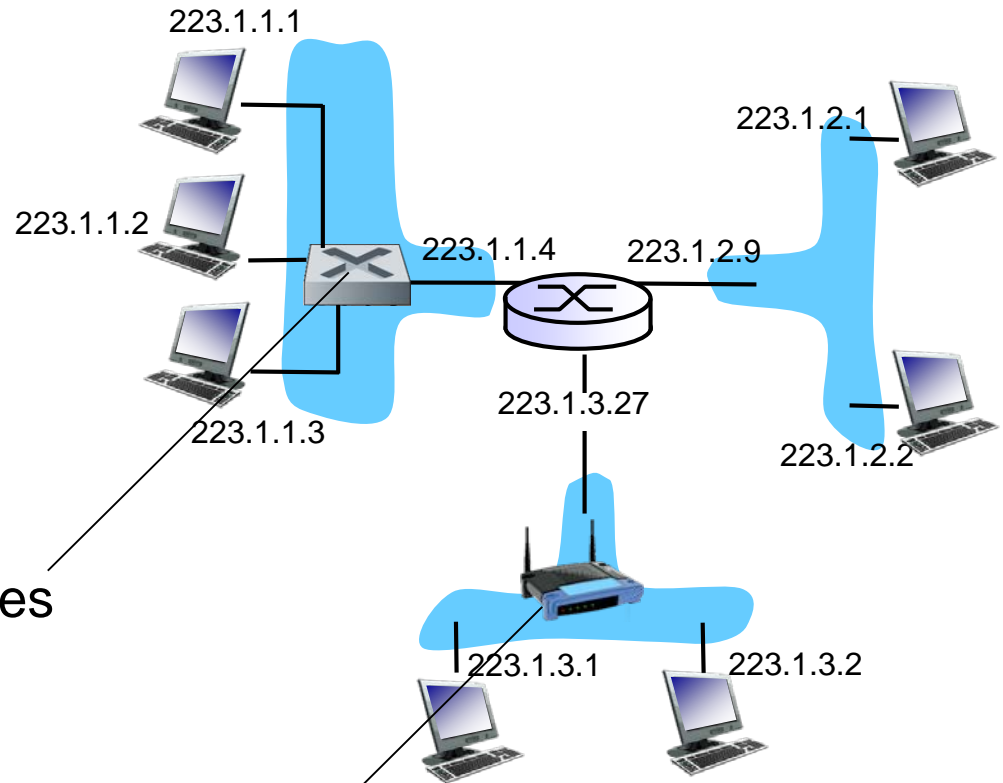
$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: 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) –will learn more later.



A: wireless WiFi interfaces connected by WiFi base station

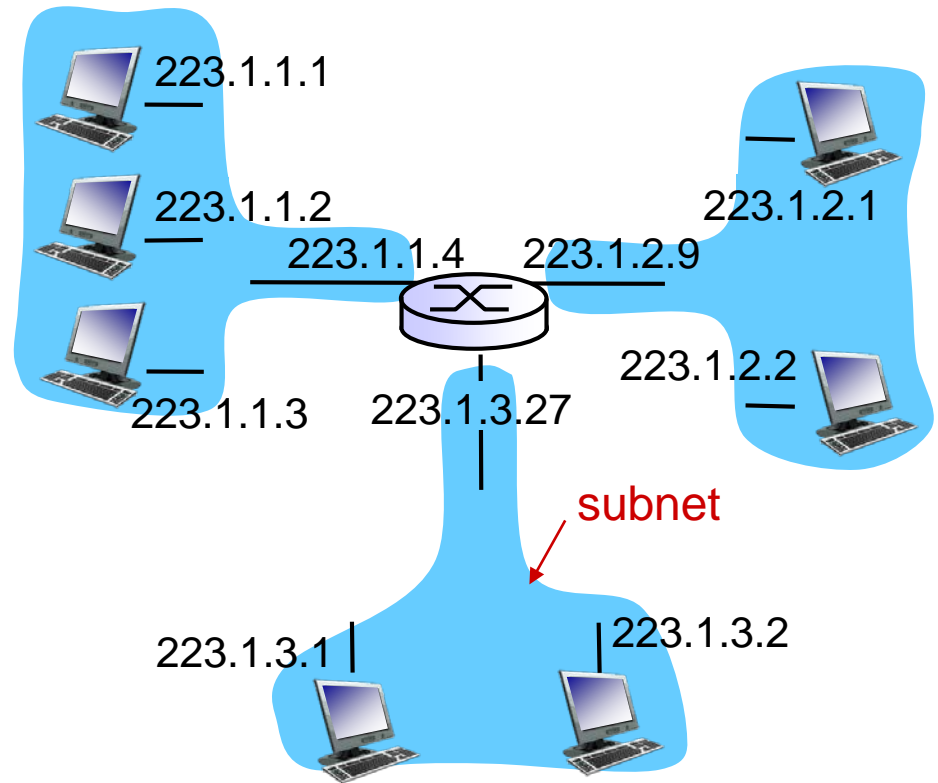
Subnets

■ *what 's a subnet ?*

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

■ IP address:

- subnet part - high order bits
- host part - low order bits

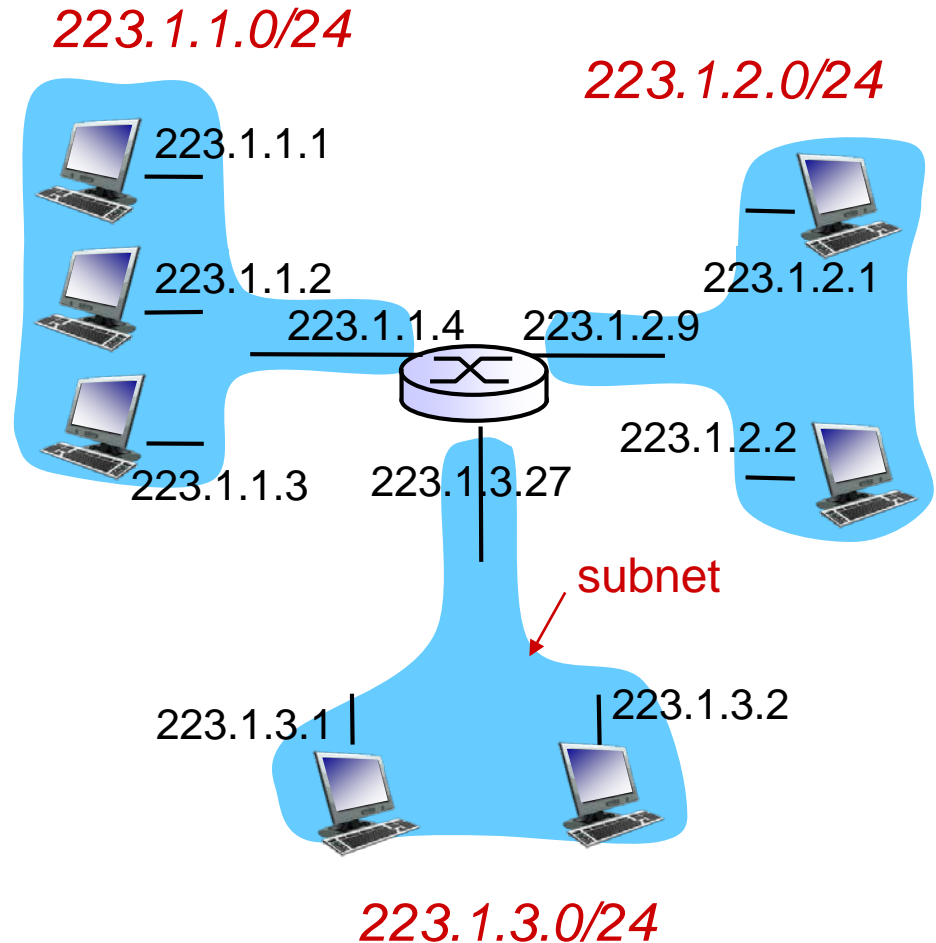


network consisting of 3 subnets

Subnets

Recipe for defining subnets

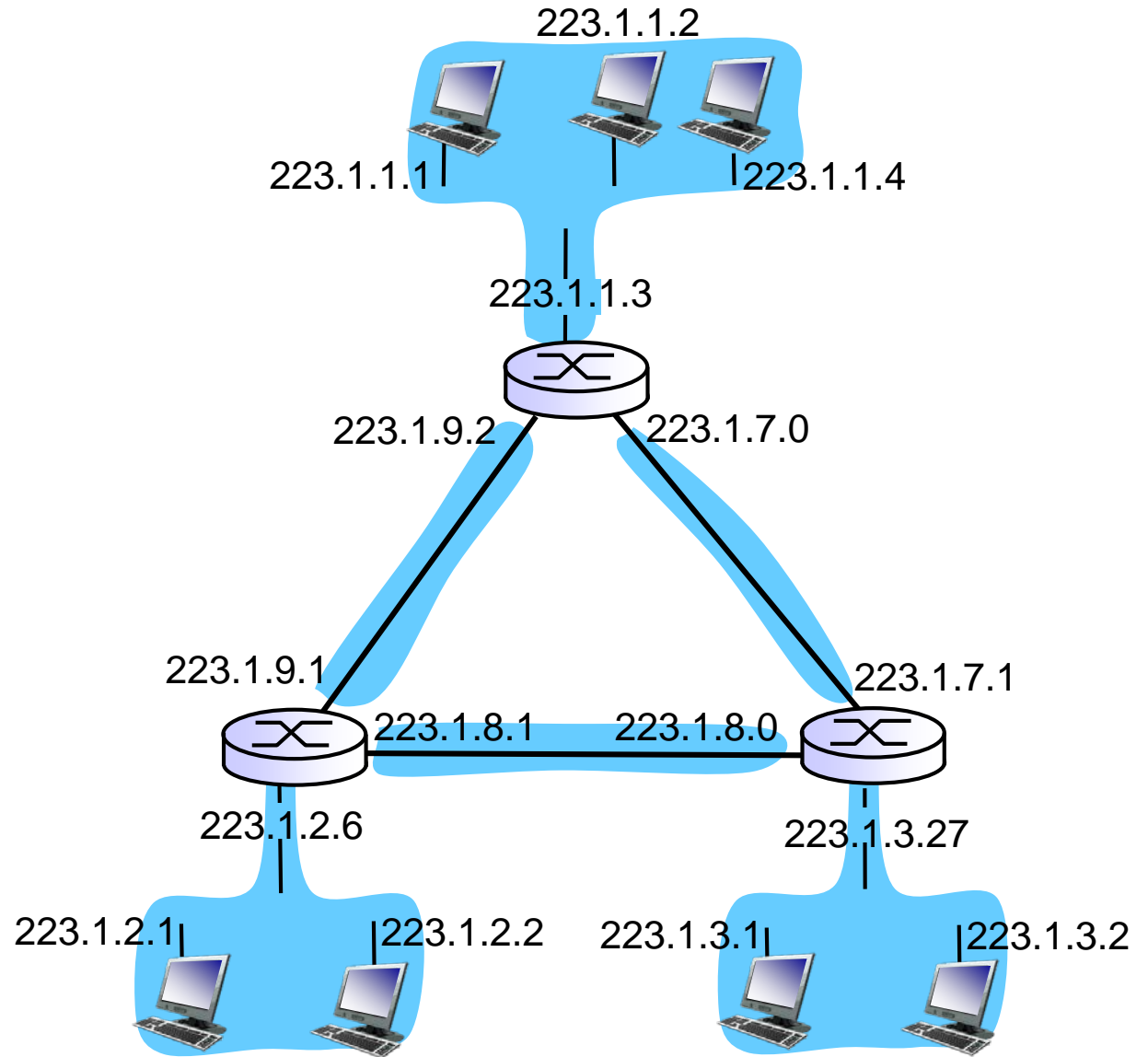
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

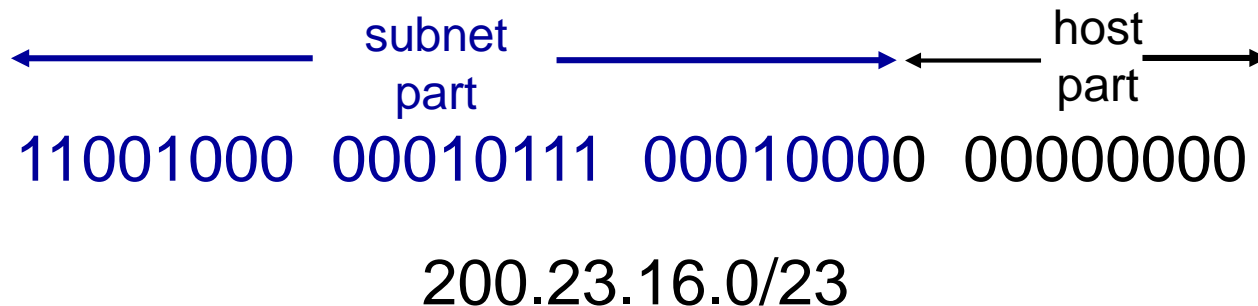
how many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



Address Masking

- Required with classless addressing
- Associated with a network
- Specifies division of addresses into network prefix and host suffix for that network
- 32-bit binary value
 - 1-bits correspond to prefix
 - 0-bits correspond to suffix
- For previous example
 - CIDR: 200.23.16.0/23
 - Mask in binary: 11111111.11111111.111111110.00000000
 - Mask in decimal notation 255.255.254.0

Special IPv4 Addresses

- Some address forms are reserved

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127/8	any	loopback	testing

- Loopback address (127.0.0.1) used for testing
 - Packets never leave the local host
- Addresses 240.0.0.0/ 8 and above are multicast

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 a *host* get IP address?
 - hard-coded by system admin in a file
 - UNIX: /etc/rc.config
 - **DHCP**: **D**ynamic **H**ost **C**onfiguration **P**rotocol:
dynamically get address from a DHCP server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

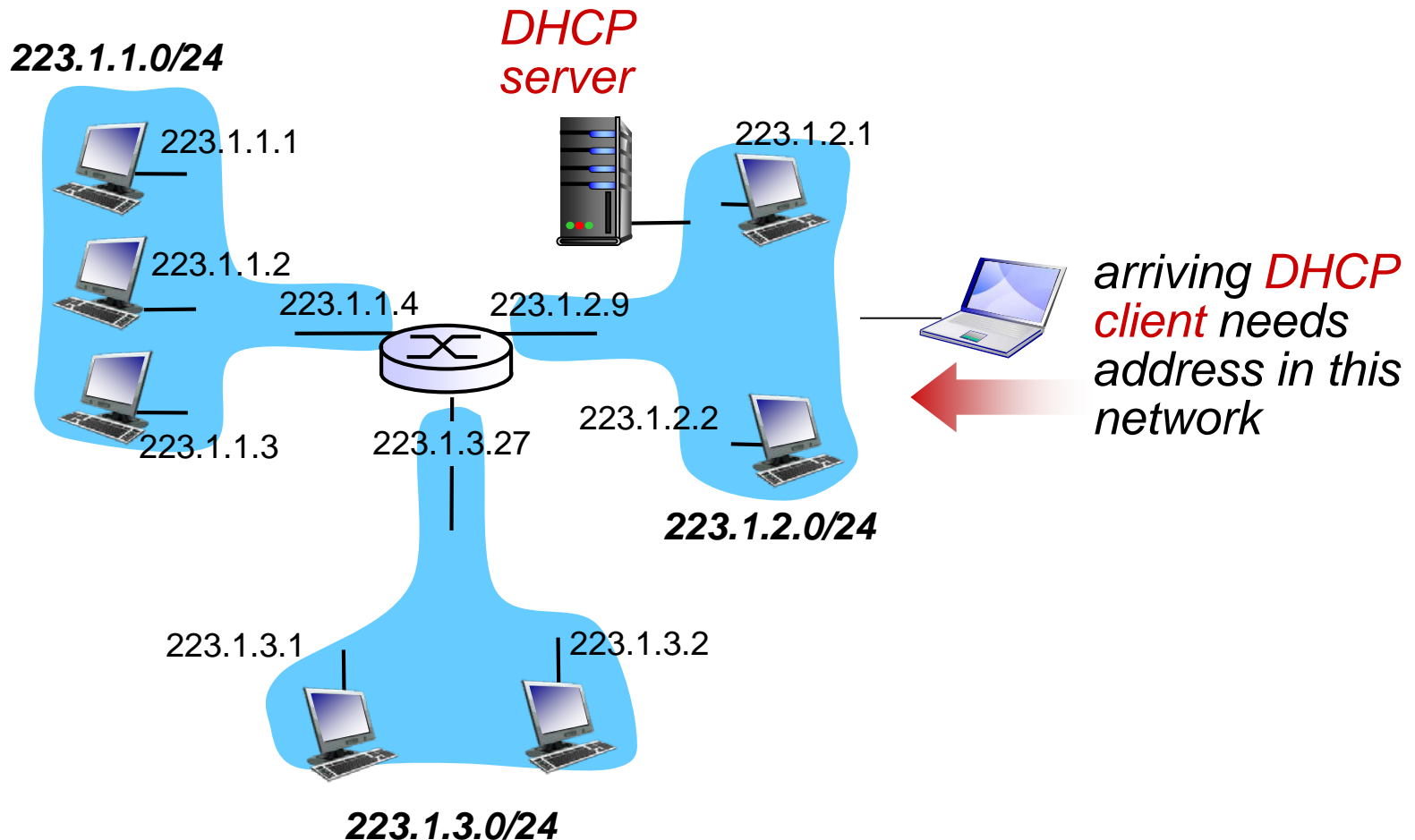
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

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

DHCP overview:

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

DHCP client-server scenario



DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

arriving
client



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'll take
that IP address!

DHCP ACK

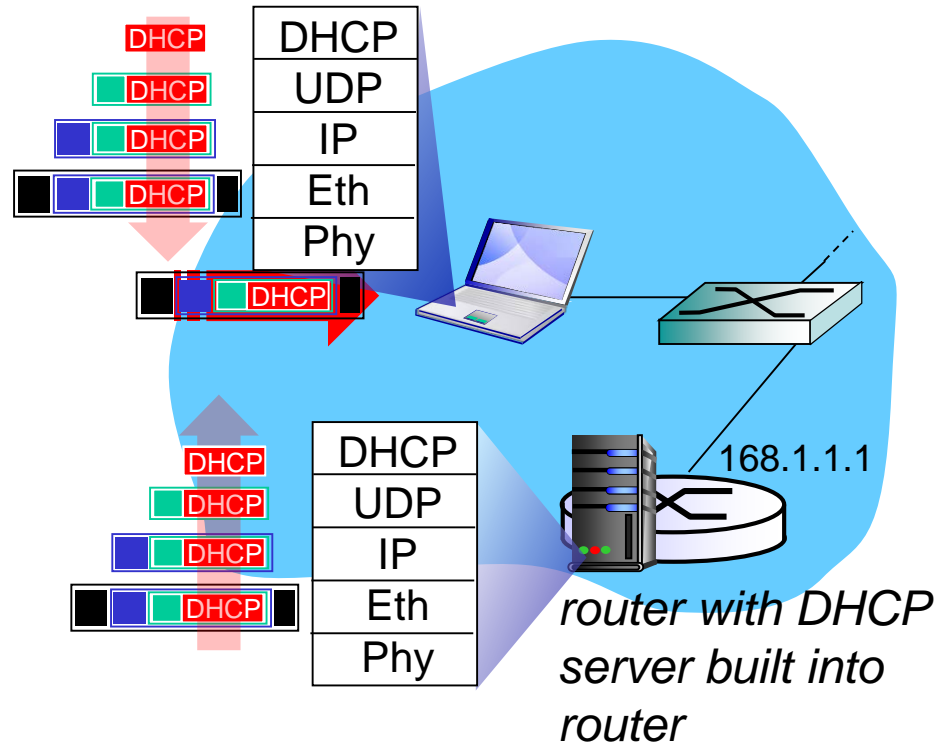
Broadcast: OK. You've
got that IP address!

DHCP: more than IP addresses

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

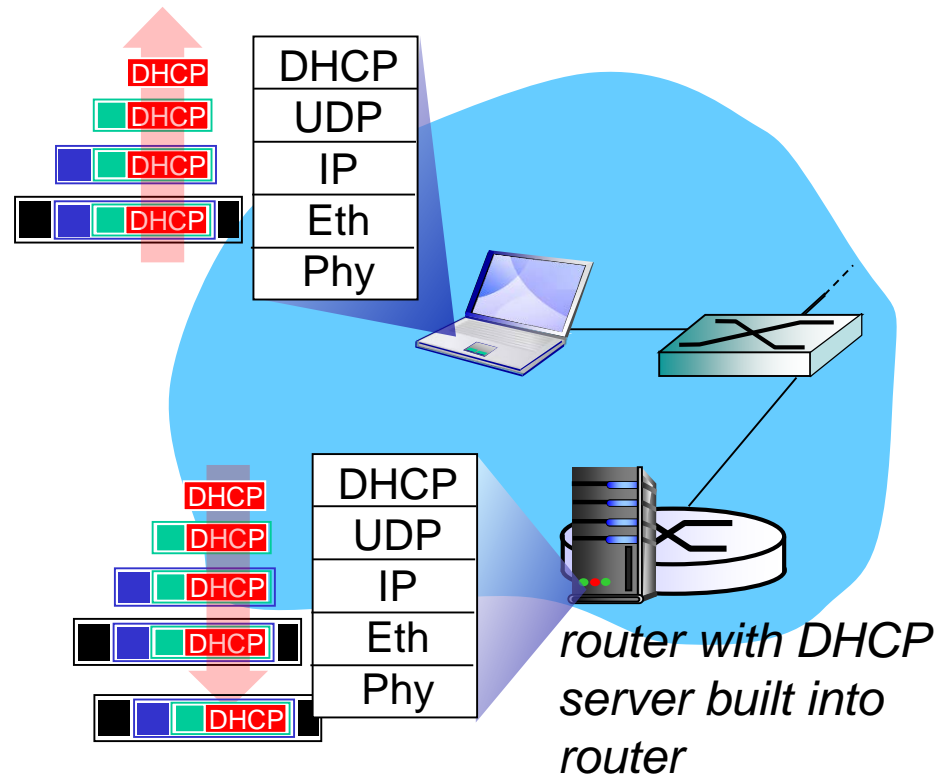
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP: Wireshark output (home LAN)

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

.....

request

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."

reply

IP addresses: how to get one?

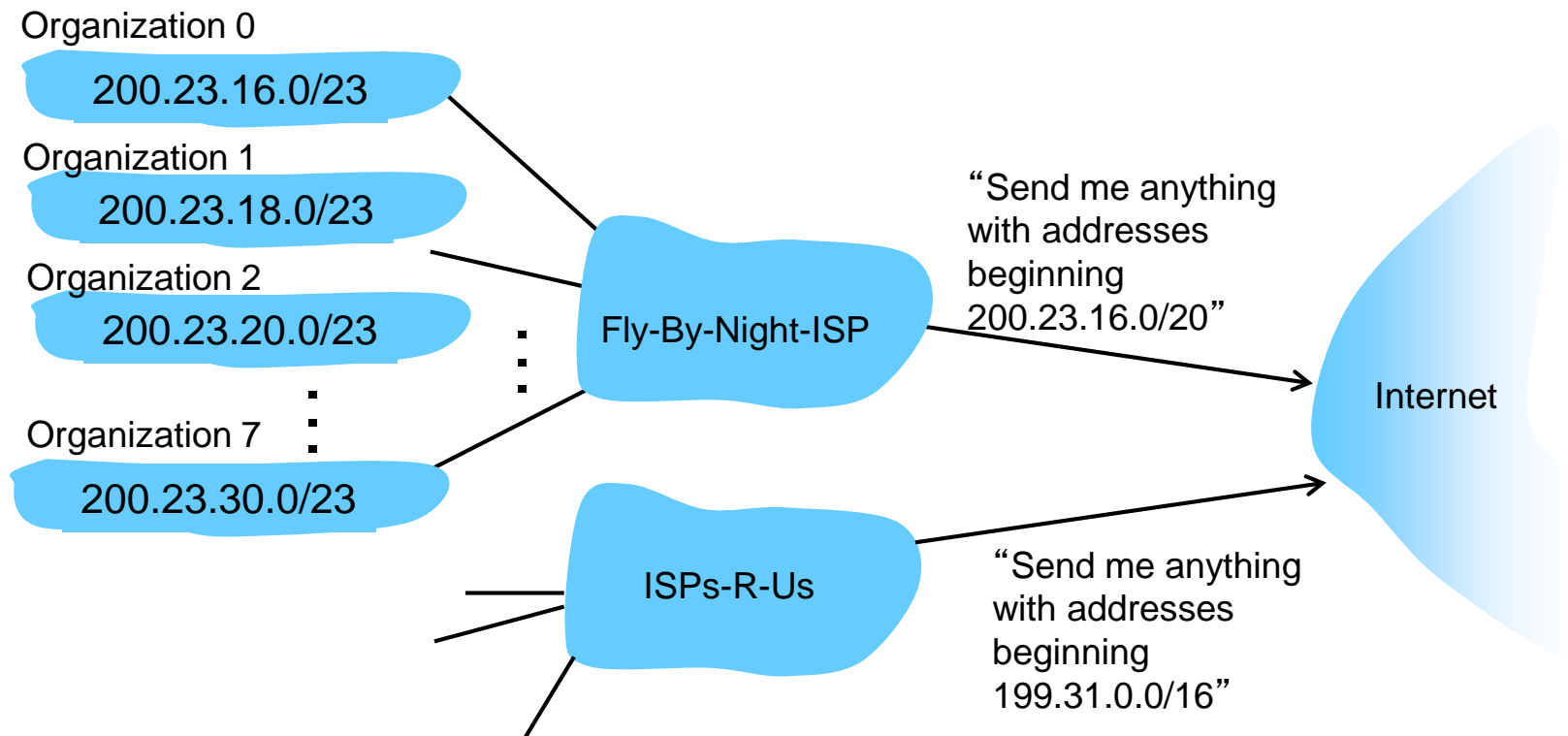
Q: how does *network* get subnet part of IP addr?

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

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

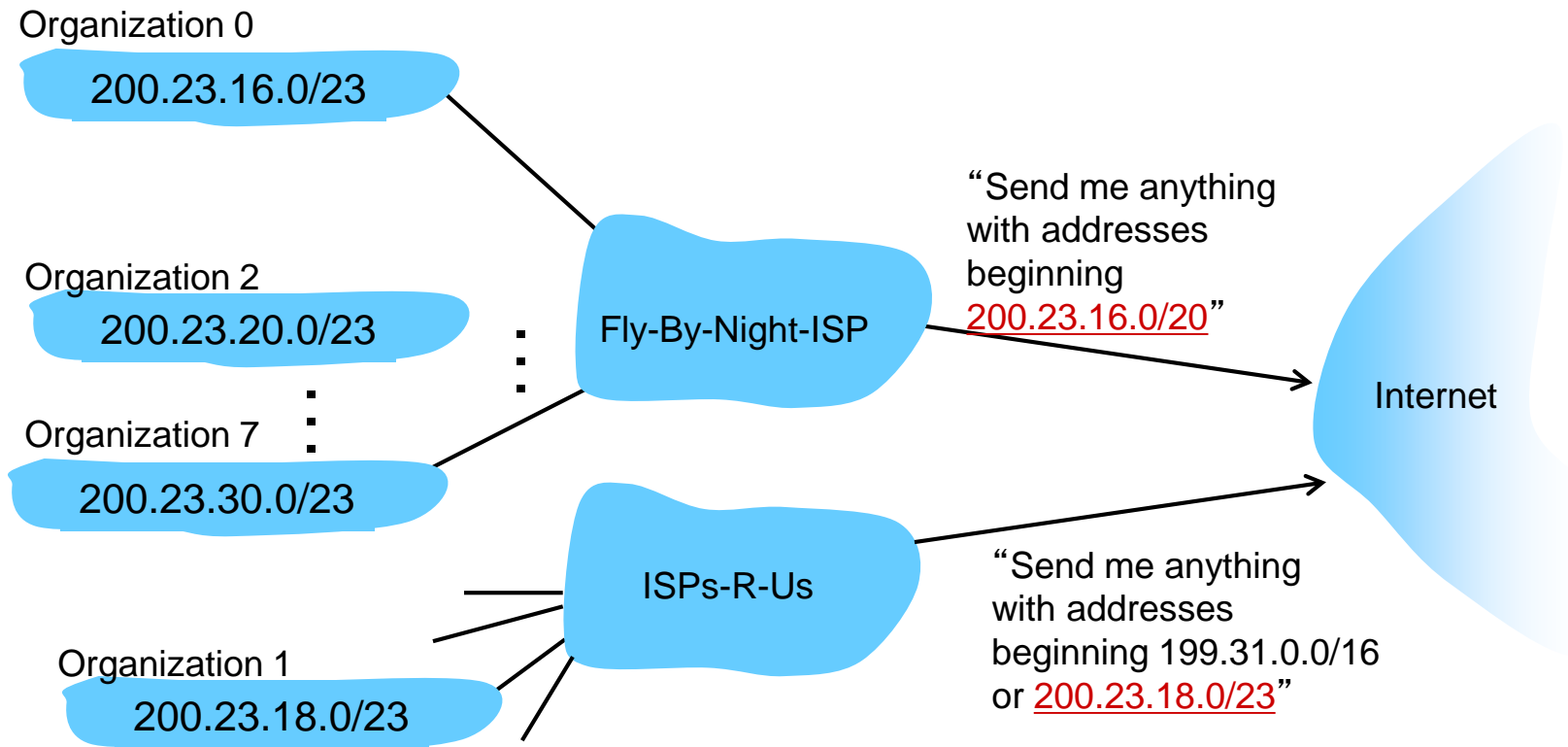
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addressing: the last word...

Q: how does an ISP get block of addresses?

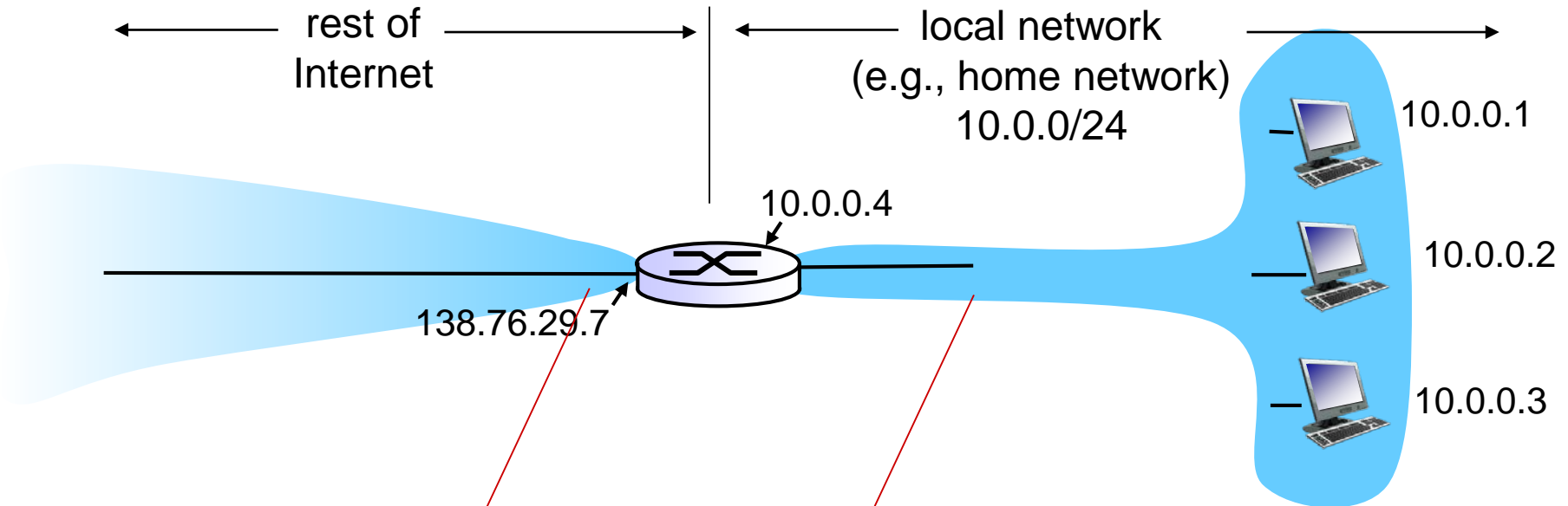
A: **ICANN:** Internet Corporation for Assigned Names and Numbers <http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs)
 - regional registries may then allocate to local registries
- manages DNS
- assigns domain names, resolves disputes

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

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:

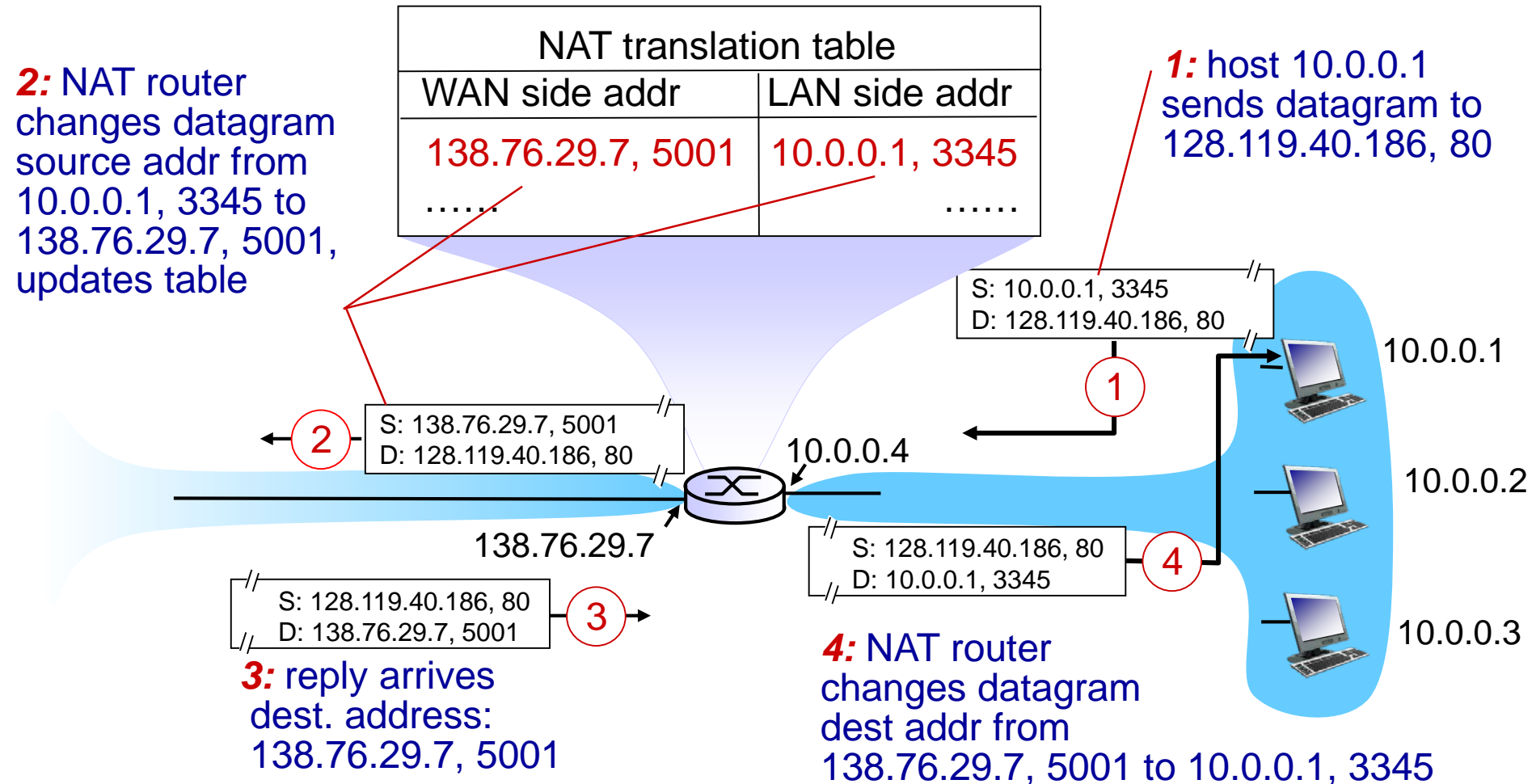
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside 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



* 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

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.3 What's inside a router

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: 40-byte fixed length header
 - header changes to facilitate quality of service (flows)
- IPv6 initiated mid-1990's
 - Slow transition: 25 years later: 30% Google clients, use IPv6, 25% of NZ domain are IPv6-enabled.(check out <https://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption> and <https://www.apnic.net/community/ipv6/ipv6-in-your-region/?cc=NZ>)

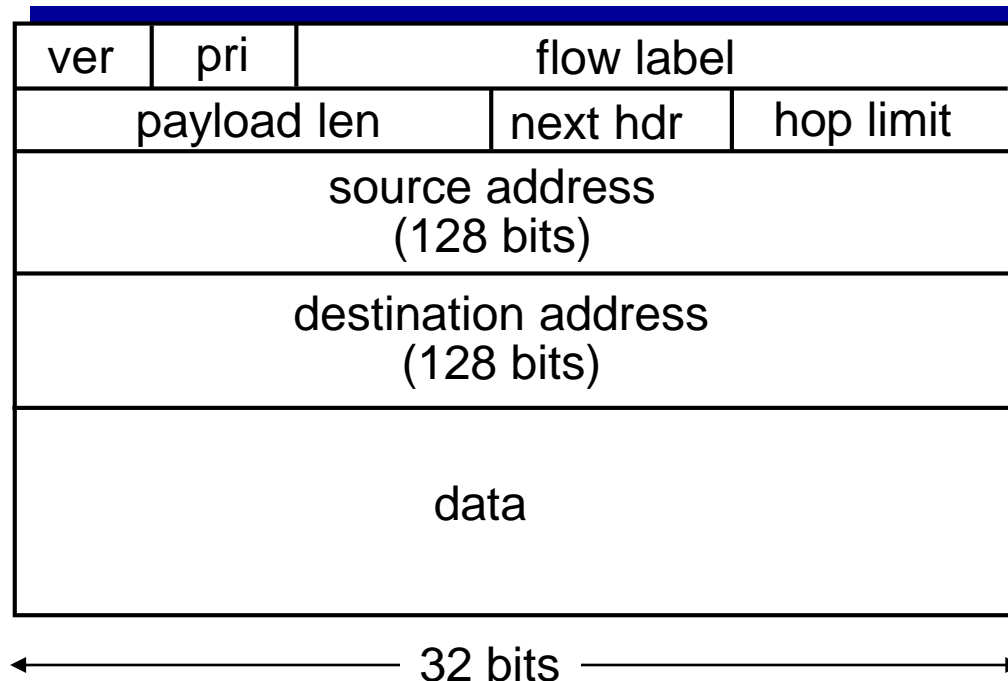
IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label: identify datagrams in same “flow.”

(concept of “flow” not well defined).

next header: identify upper layer protocol for data

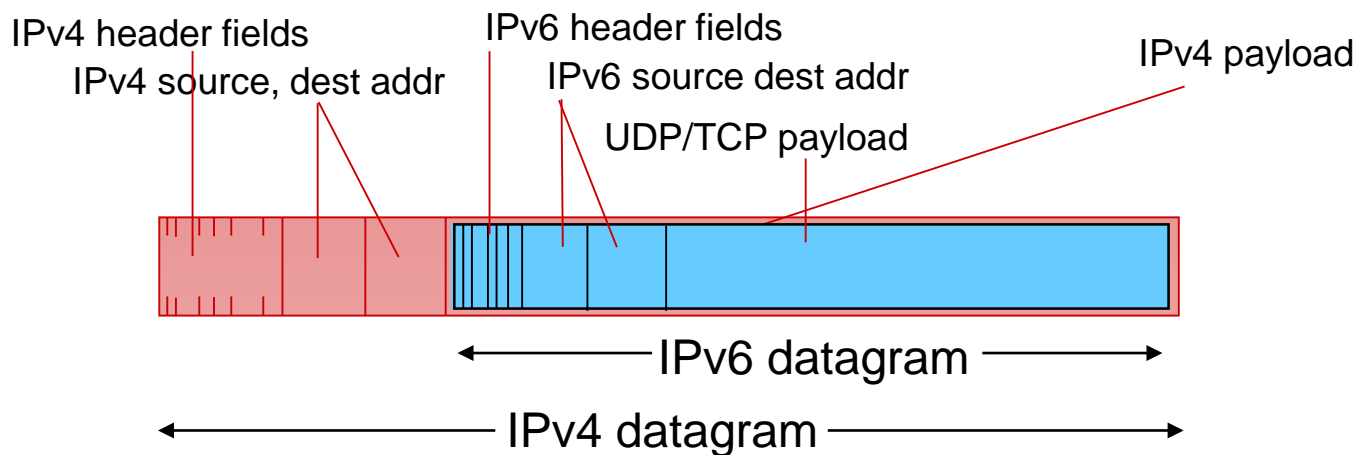


Other changes from IPv4

- *checksum*: removed entirely to reduce processing time at each hop
- *options*: allowed, but outside of header, indicated by “Next Header” field
- *ICMPv6*: new version of ICMP
 - additional message types, e.g. “Packet Too Big”
 - multicast group management functions
- No fragmentation by routers, done by sending host instead.
- Addresses are 128-bit Hexidecimal numbers eg FE80:CD00:0000:0000:0000:0000:211E:729C

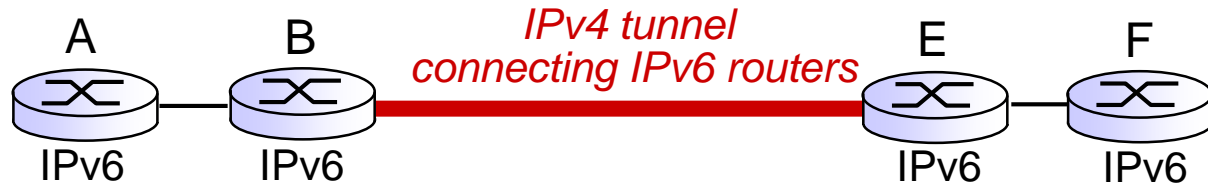
Transition from IPv4 to IPv6

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

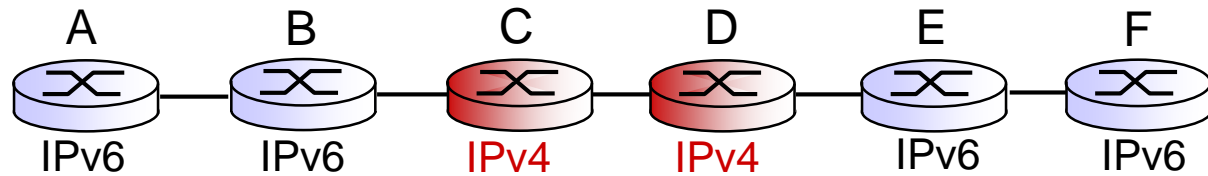


Tunneling

logical view:

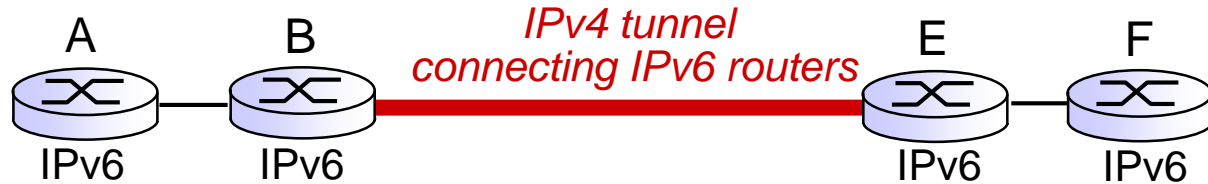


physical view:

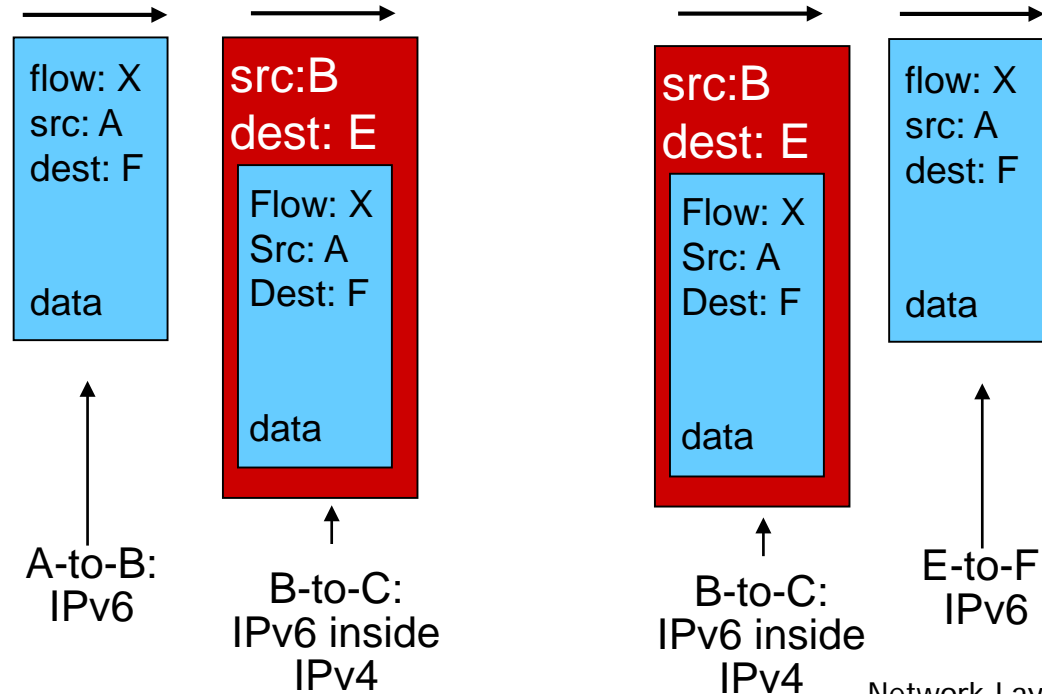
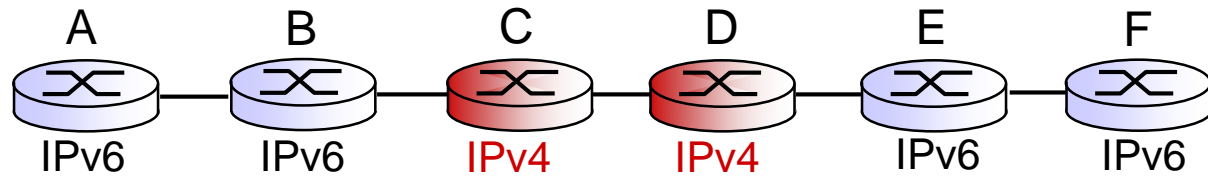


Tunneling

logical view:



physical view:



IPv6: adoption

- Google: 30% of clients access services via IPv6.
- the '.govt.nz' domain has been designed to be dual-stack capable, which means it supports both IPv4 and IPv6, allowing any user to access IPv6 '.govt.nz' websites natively (from *).
- *Long (long!) time for deployment, use*
 - 25 years and counting!
 - think of application-level changes in last 25 years: WWW, Facebook, streaming media, Skype, ...
 - *Why?*

*https://snapshot.ict.govt.nz/resources/digital-ict-archive/static/localhost_8000/guidance-and-resources/architecture/internet-protocol-version-6-for-new-zealand-government/index.html#Dual

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

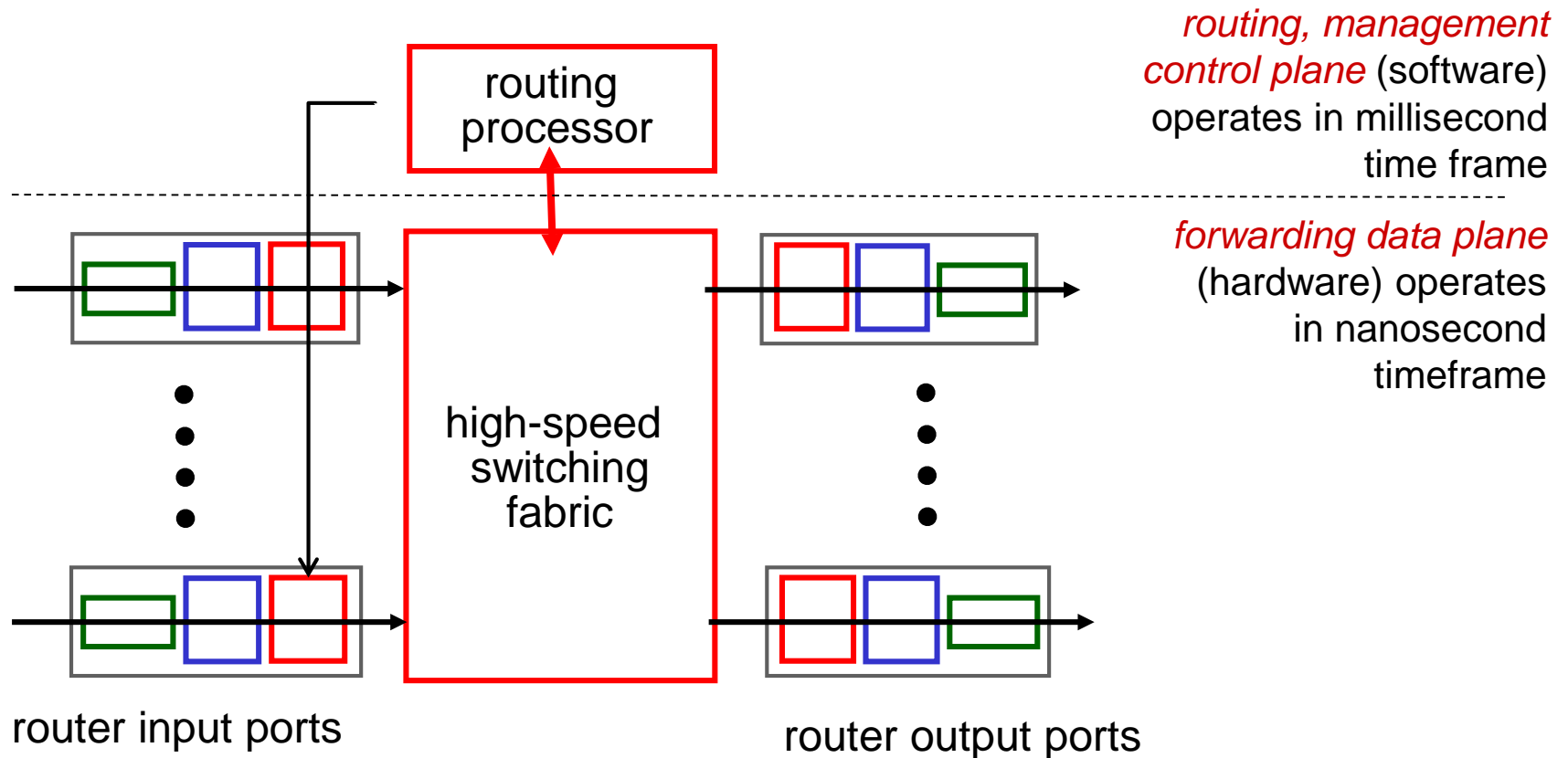
4.3 What's inside a router

4.4 Generalized Forward and SDN

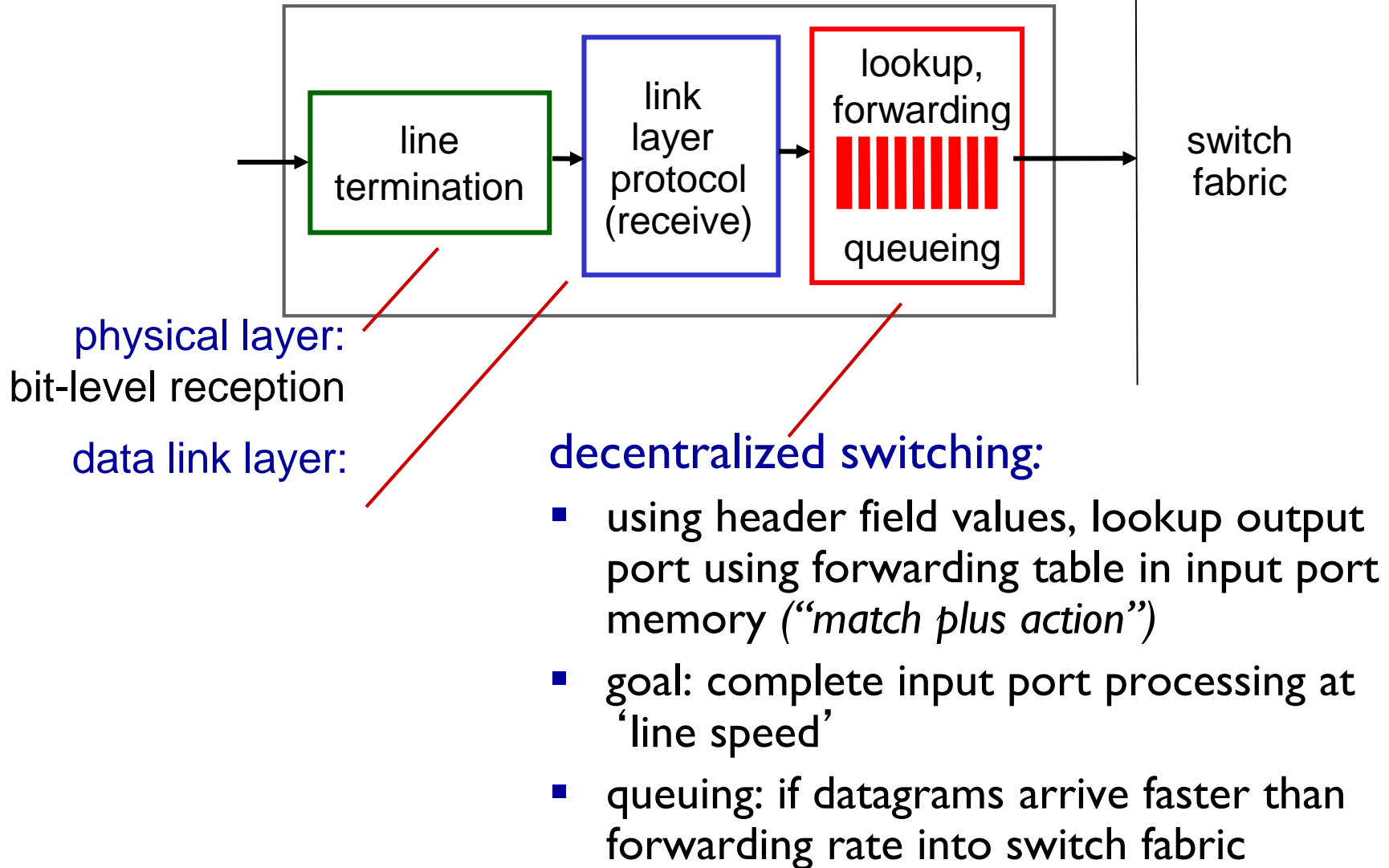
- match
- action
- OpenFlow examples of match-plus-action in action

Router architecture overview

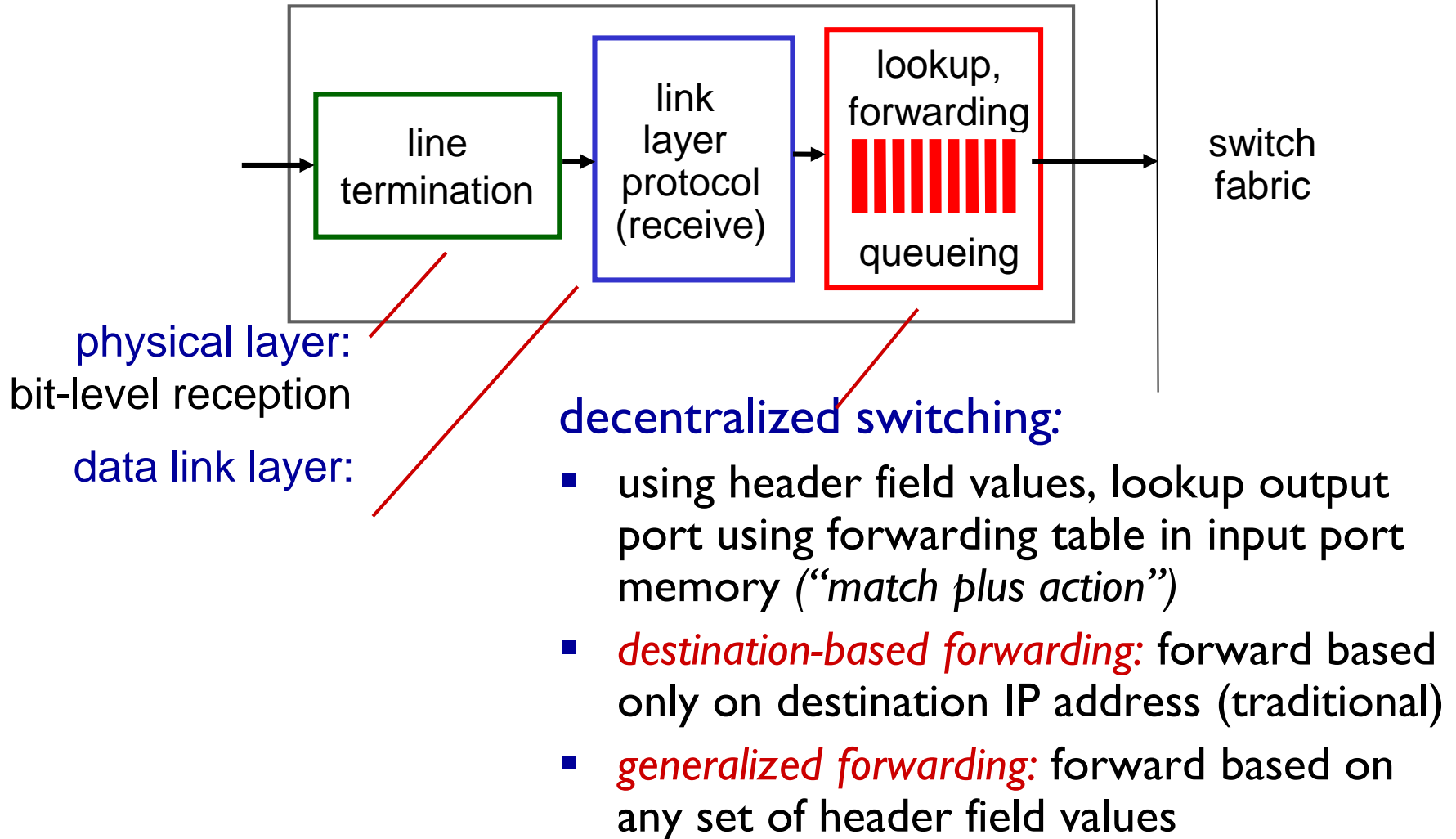
- high-level view of generic router architecture:



Input port functions



Input port functions



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 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

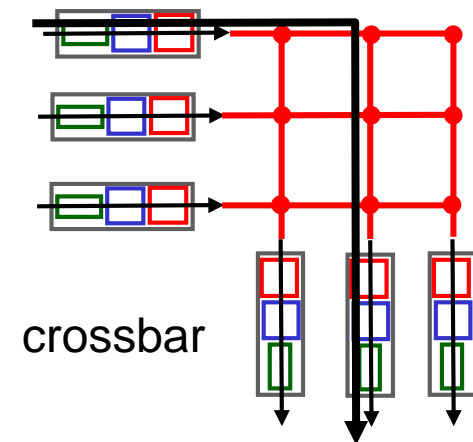
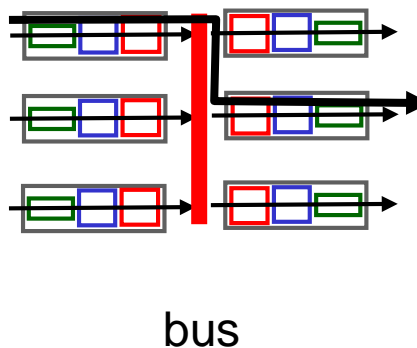
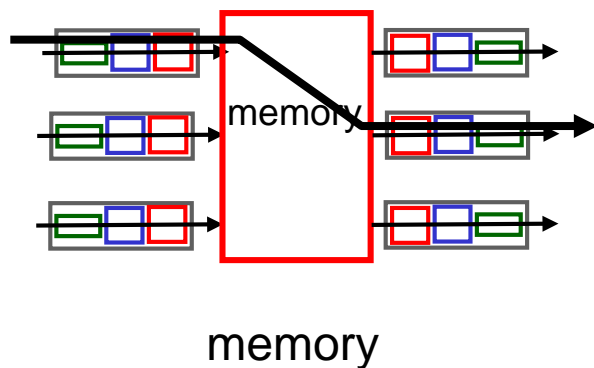
which interface?

Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
 - *content addressable*: present address to TCAM: retrieve address in one clock cycle, regardless of table size
 - Cisco Catalyst: can up ~1M routing table entries in TCAM

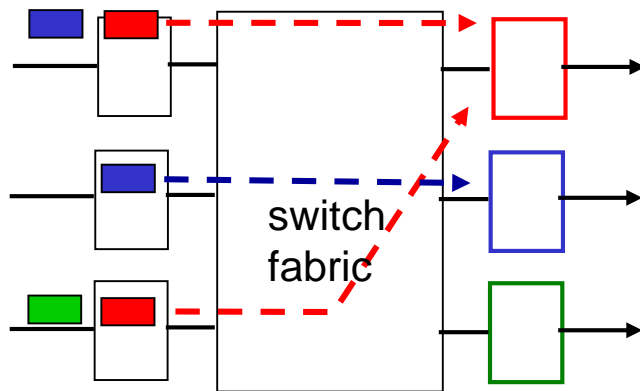
Switching fabrics

- transfer packet from input buffer to appropriate output buffer
- 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
- three types of switching fabrics

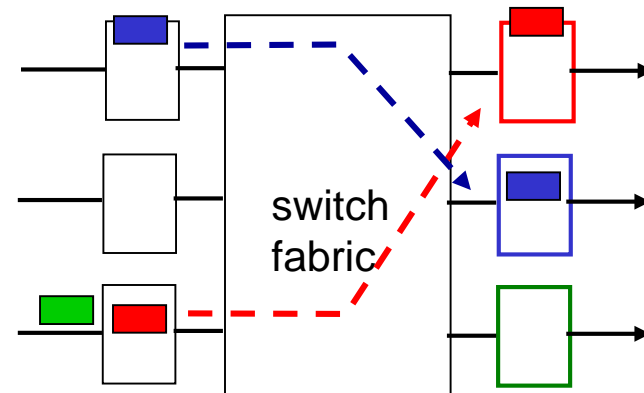


Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
 - *queueing delay and loss due to input buffer overflow!*
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward

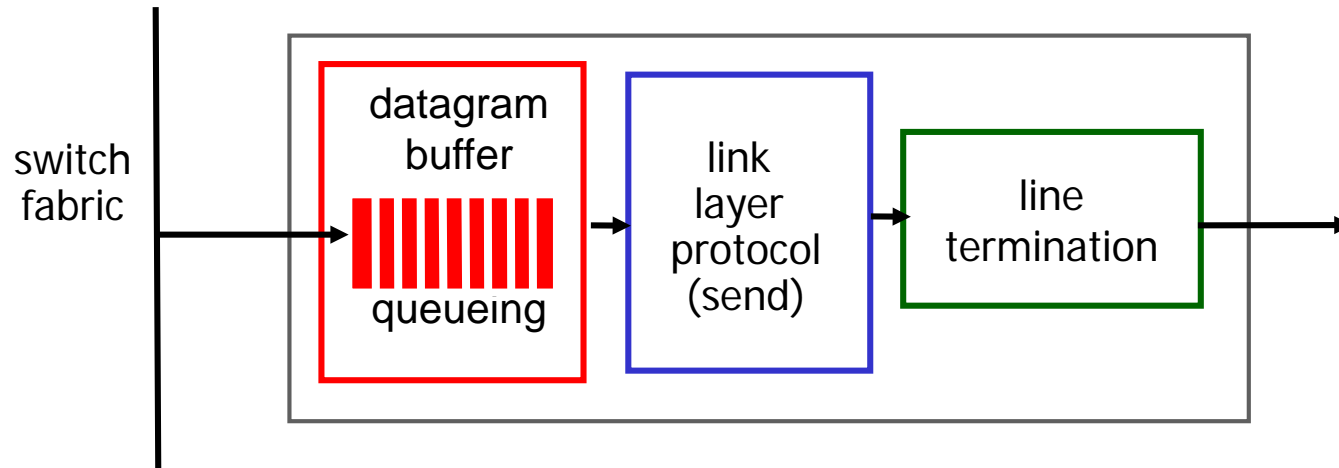


output port contention:
only one red datagram can be
transferred.
lower red packet is blocked



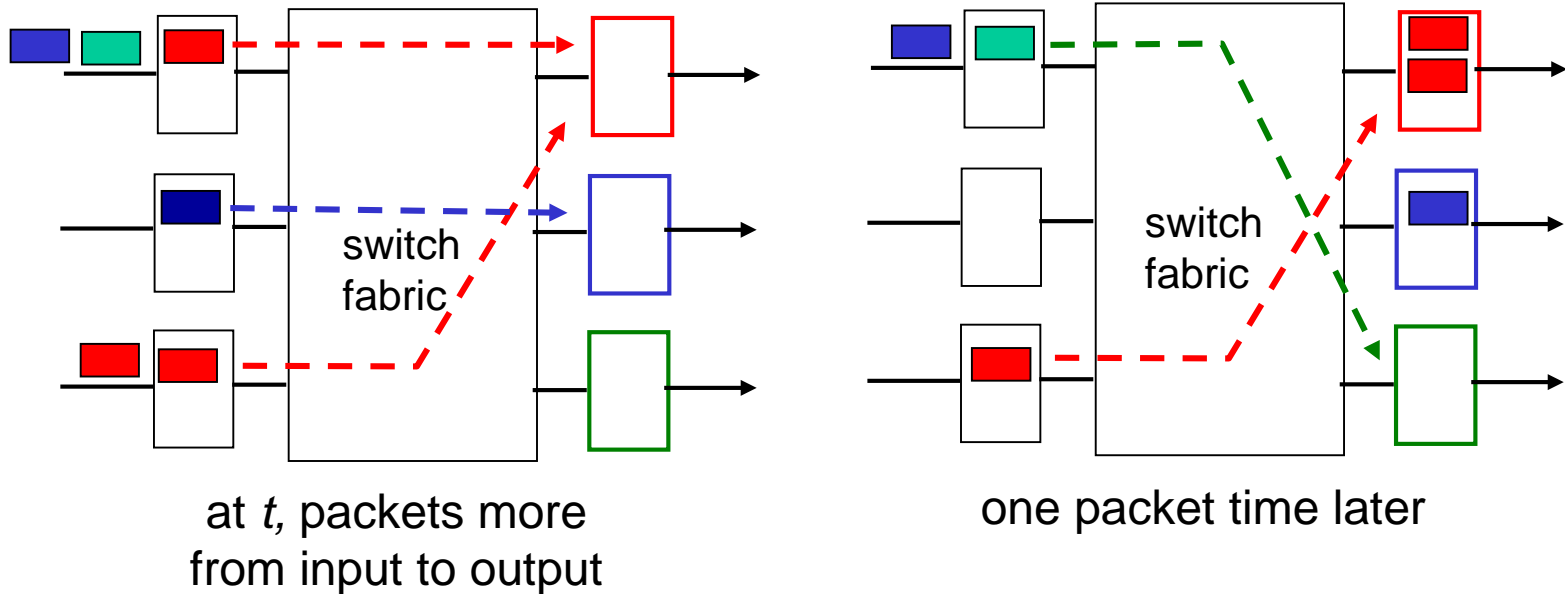
one packet time later:
green packet
experiences HOL
blocking

Output ports



- *buffering* required when the switch fabric faster than the line
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

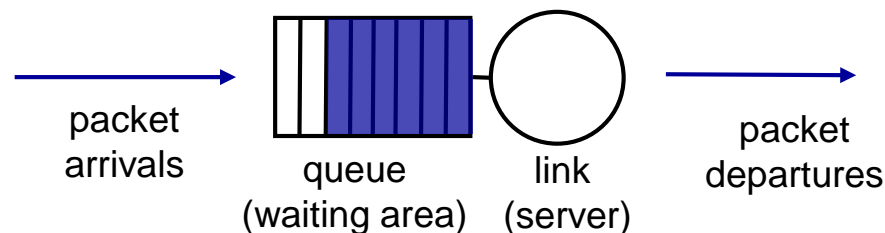
Output port queueing



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

Scheduling mechanisms

- *scheduling*: choose next packet to send on link
- *FIFO (first in first out) scheduling*: 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



Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

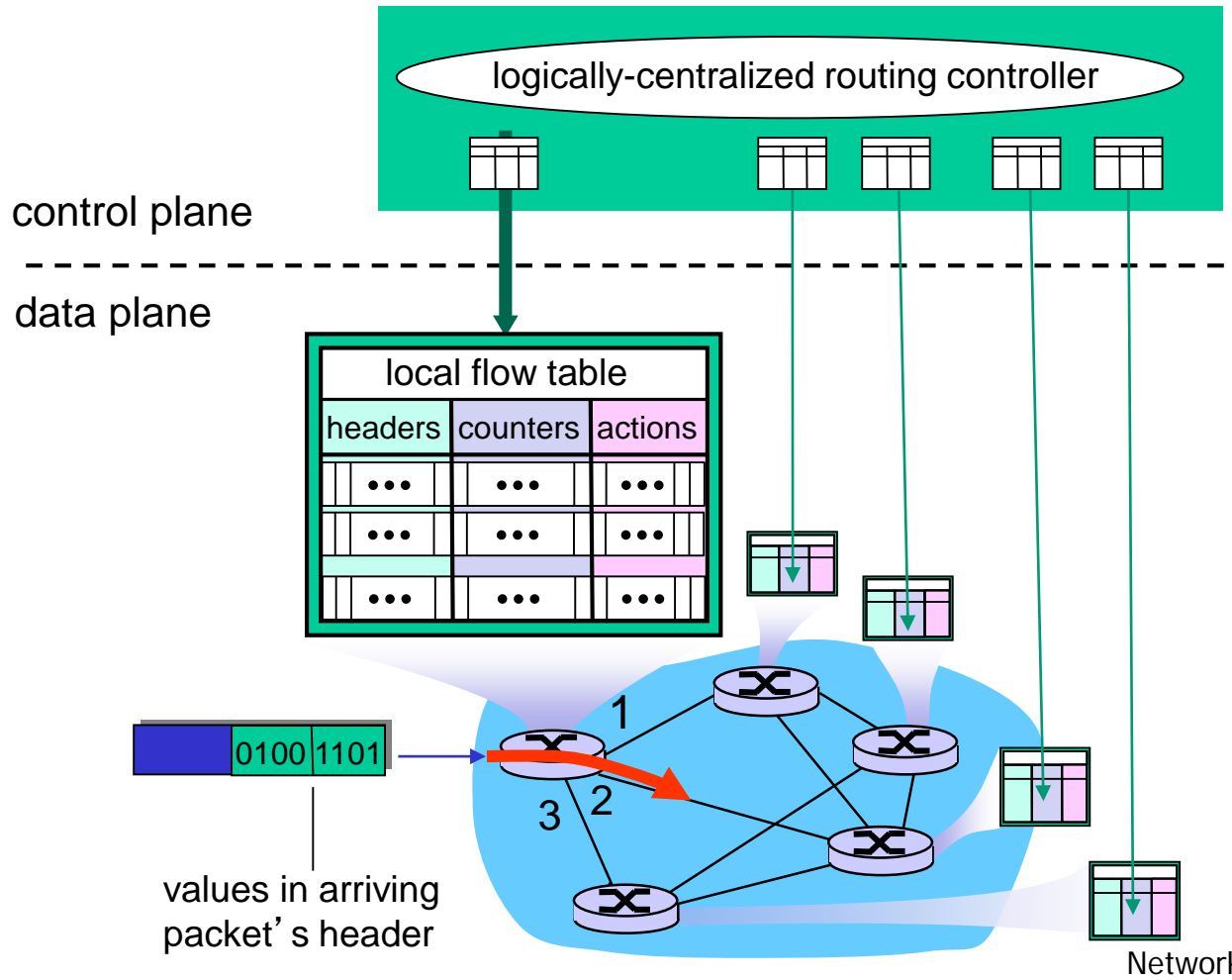
4.3 What's inside a router

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

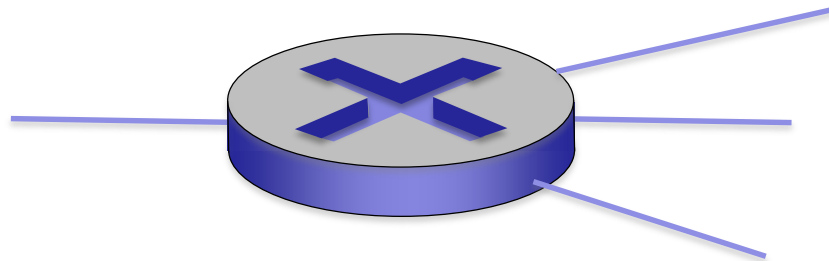
Generalized Forwarding and SDN

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



OpenFlow data plane abstraction

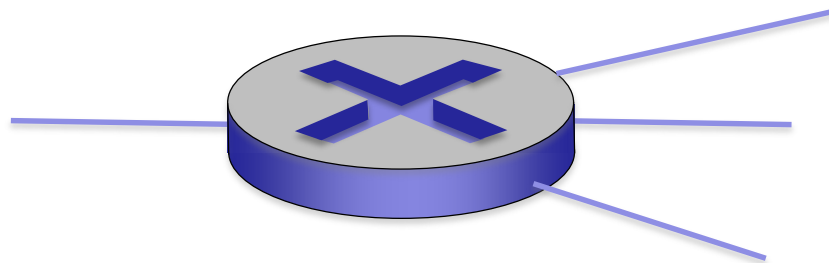
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
 - *Pattern*: match values in packet header fields
 - *Actions: for matched packet*: drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: disambiguate overlapping patterns
 -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



Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

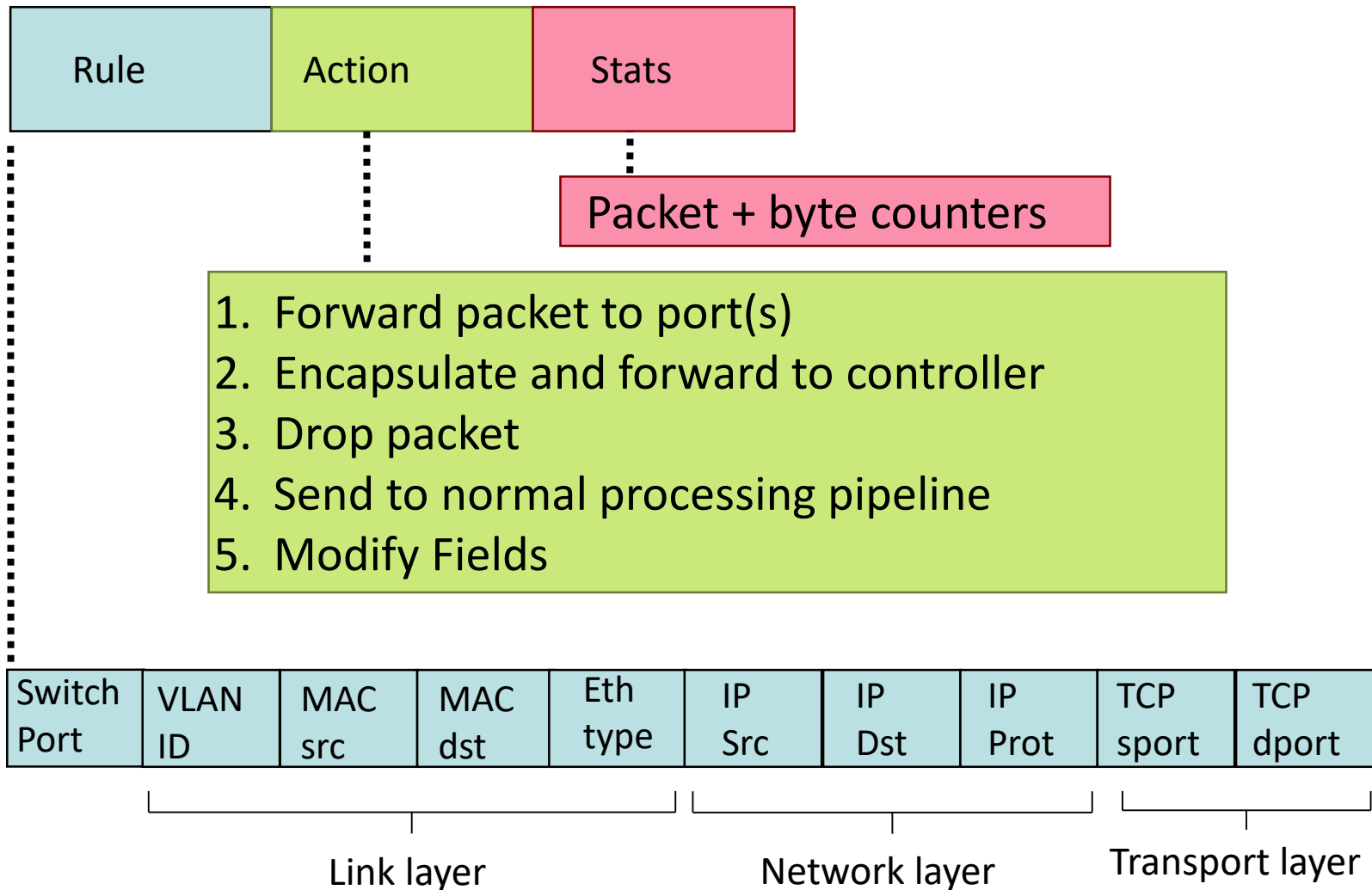
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
 - *Pattern*: match values in packet header fields
 - *Actions: for matched packet*: drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: disambiguate overlapping patterns
 -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



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	Action
*	*	*	*	*	*	*	*	*	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	Action
*	*	*	*	*	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*

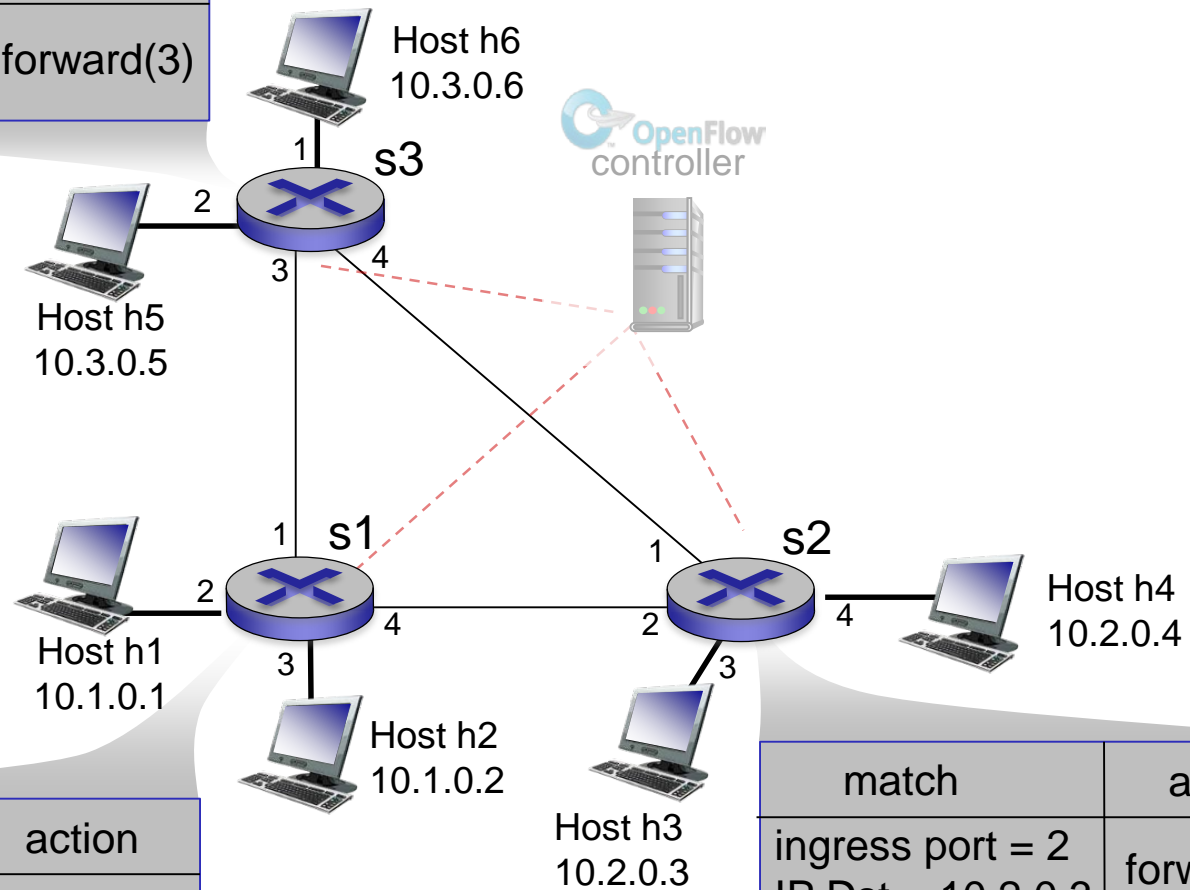
OpenFlow abstraction

- *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

OpenFlow example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Chapter 4: done!

4.1 Overview of Network layer: data plane and control plane

4.2 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.3 What's inside a router

4.4 Generalized Forward and SDN

- match plus action
- OpenFlow example

Question: how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)

References

- Most slides are supplied with Computer Networking, A top down approach by Kurose and Ross. Adaptions have been made by Catherine Watson
- Slides 23 and 24 are from Computer networking and Internets by Douglass E. Comer.