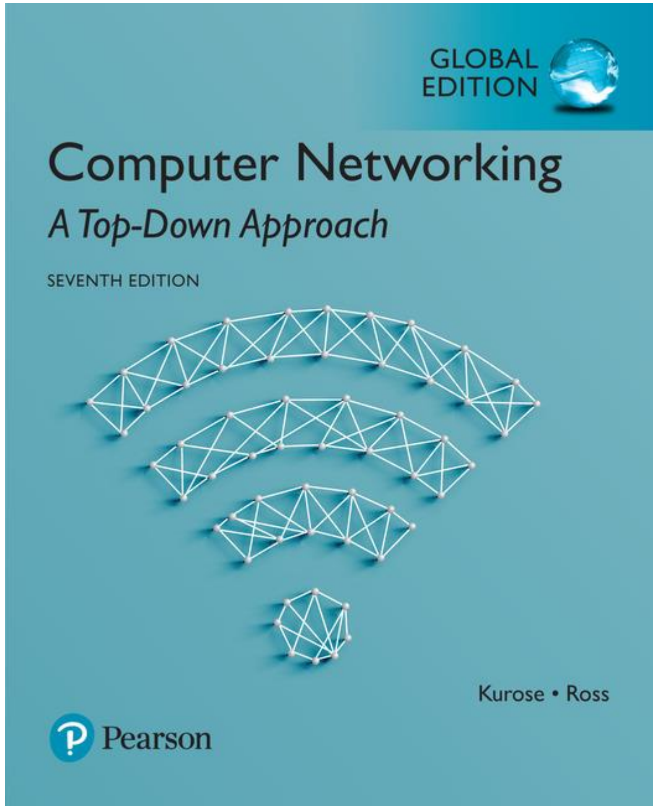
Chapter 4
Network Layer:
The Data Plane

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, Global Edition
Jim Kurose, Keith Ross
Pearson
April 2016

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

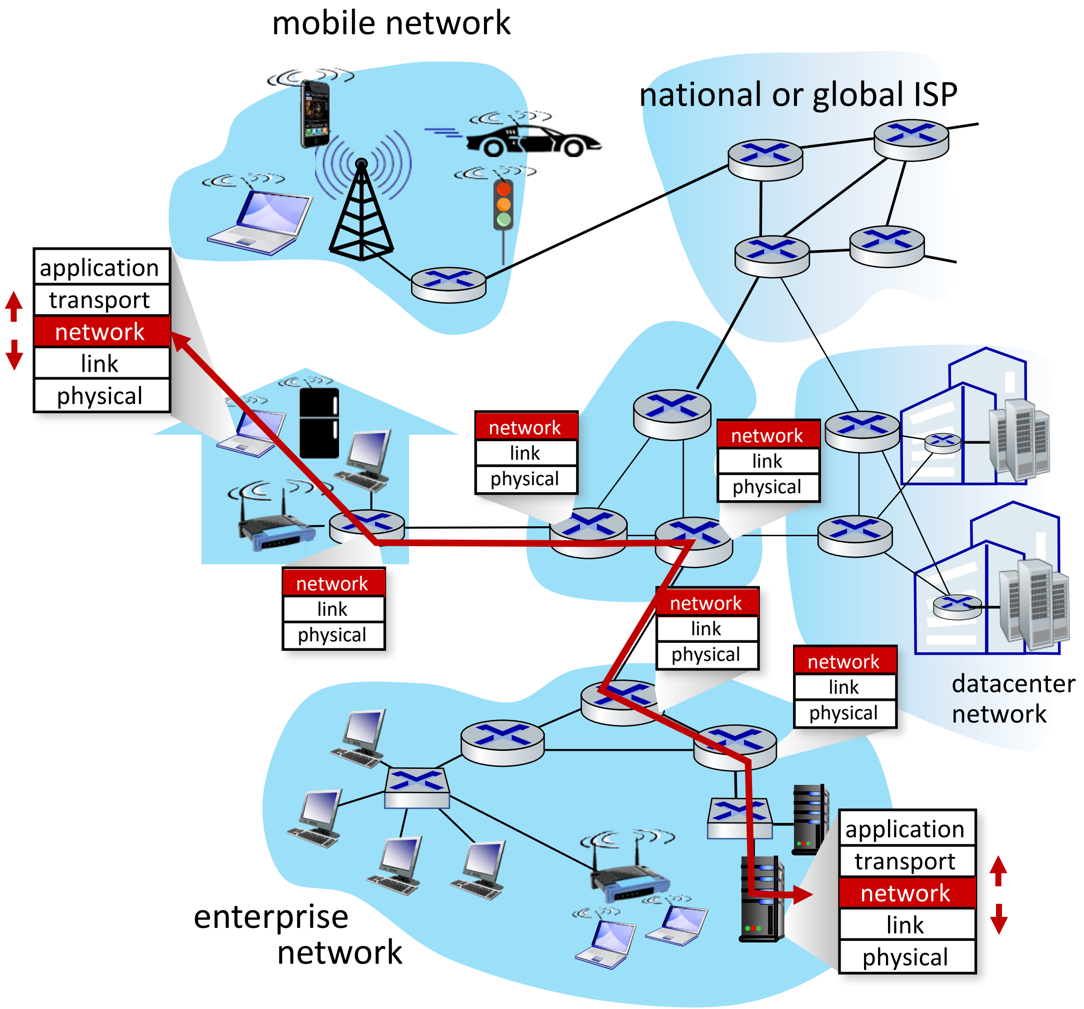
4.2 What’s inside a router

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

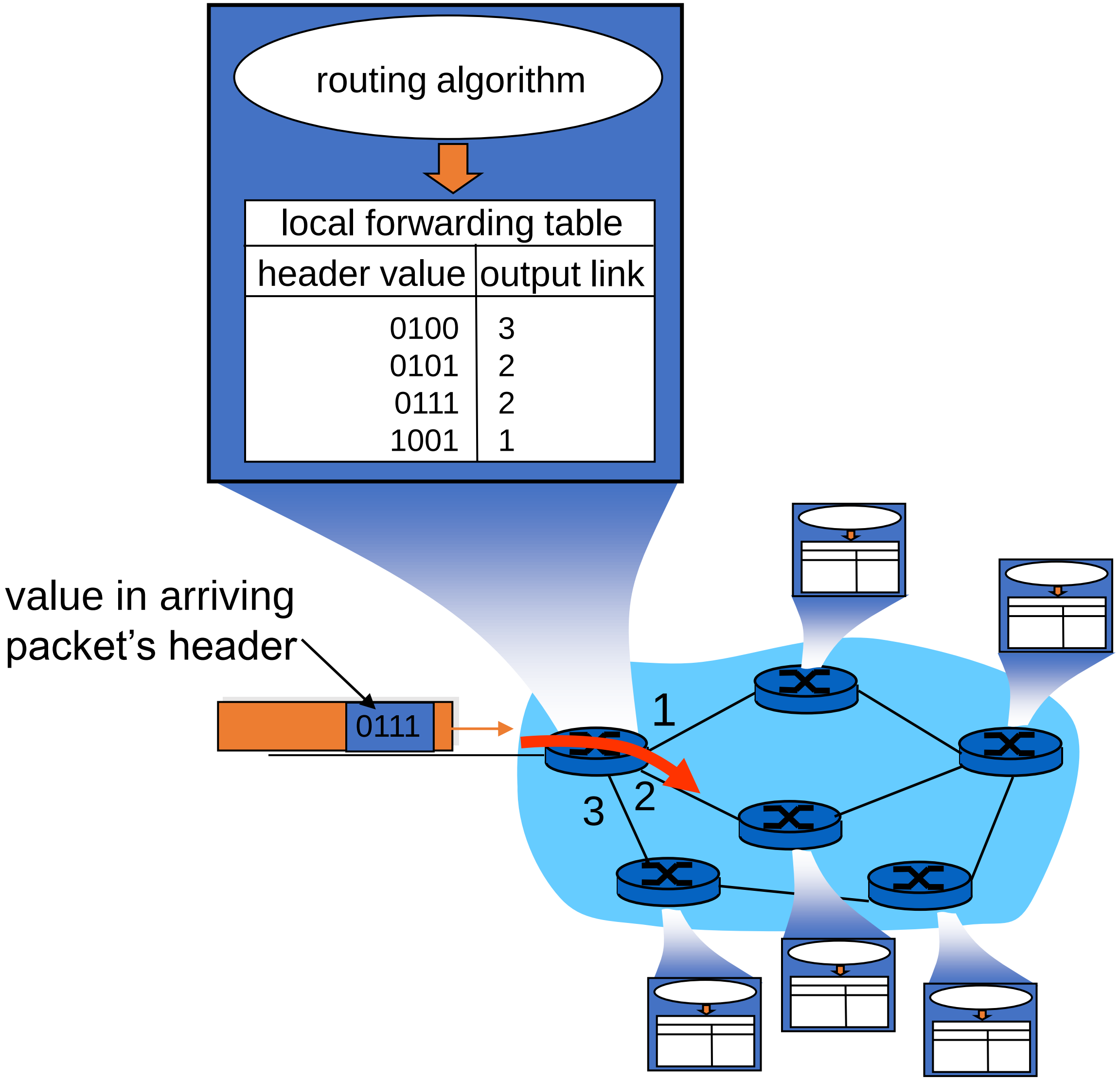
Network layer

- 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 and router
- Router examines header fields in all IP datagrams passing through it



Two Key Network-Layer Functions

- Forwarding: move packets from router’s input to appropriate router output
- Routing: determine the path taken by packets as they flow from a sender to a receiver
  - Routing algorithms – run at routers to determine “paths”;
  - Routers have a forwarding table
    - Destination address-based in Datagram networks
    - Virtual circuit number-based in VC Networks



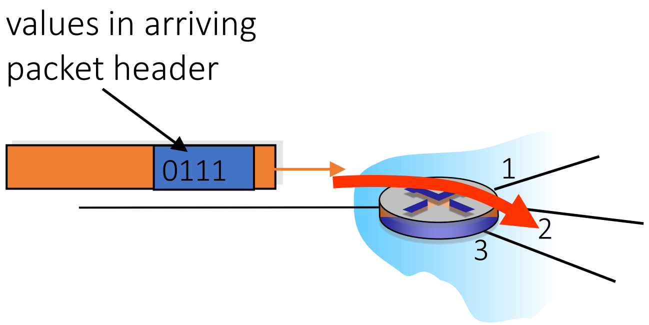
local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

Network Layer: Data Plane & Control Plane

Data plane

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

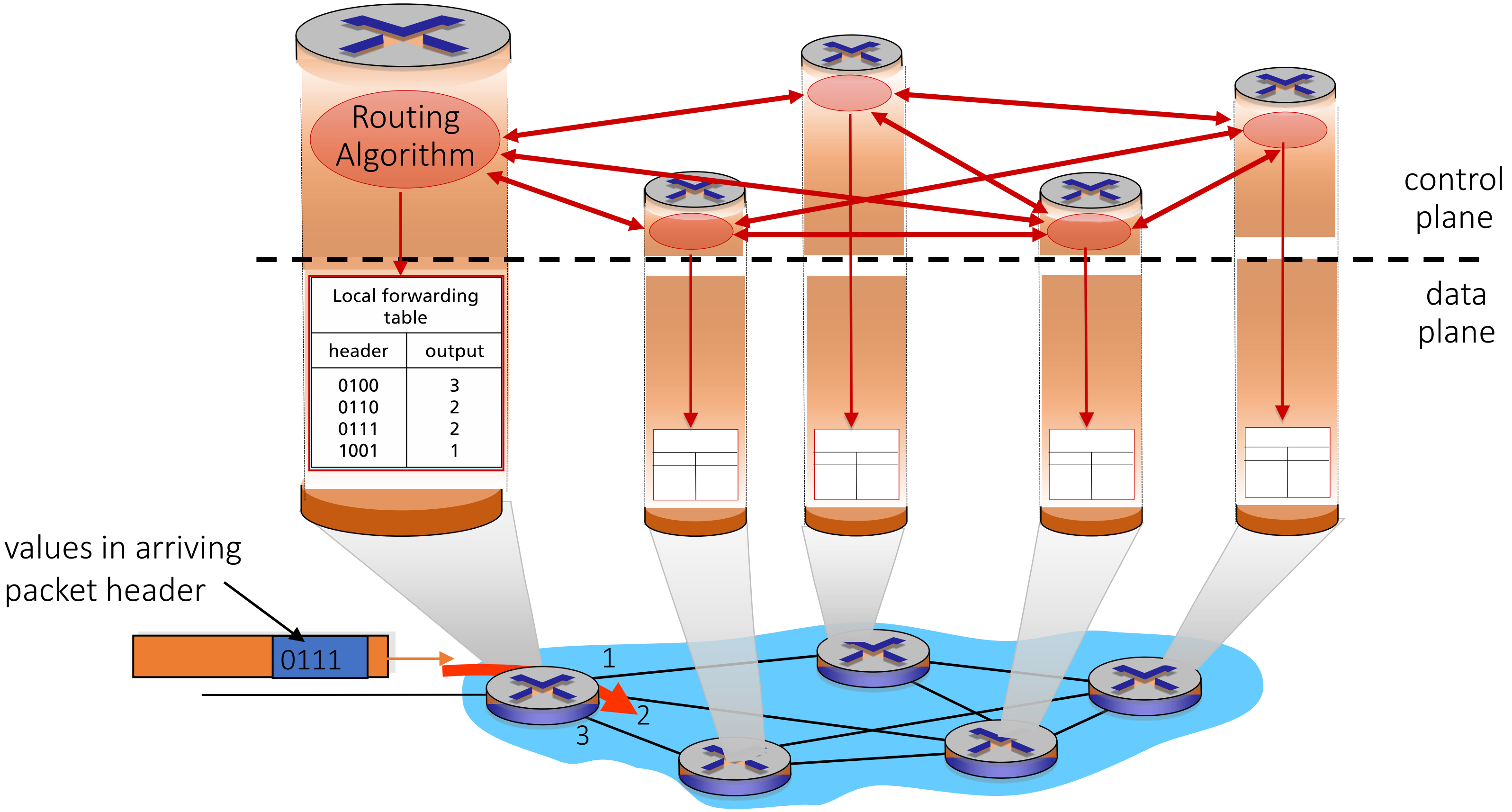


Control plane

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

(1) Per-Router Control Plane

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

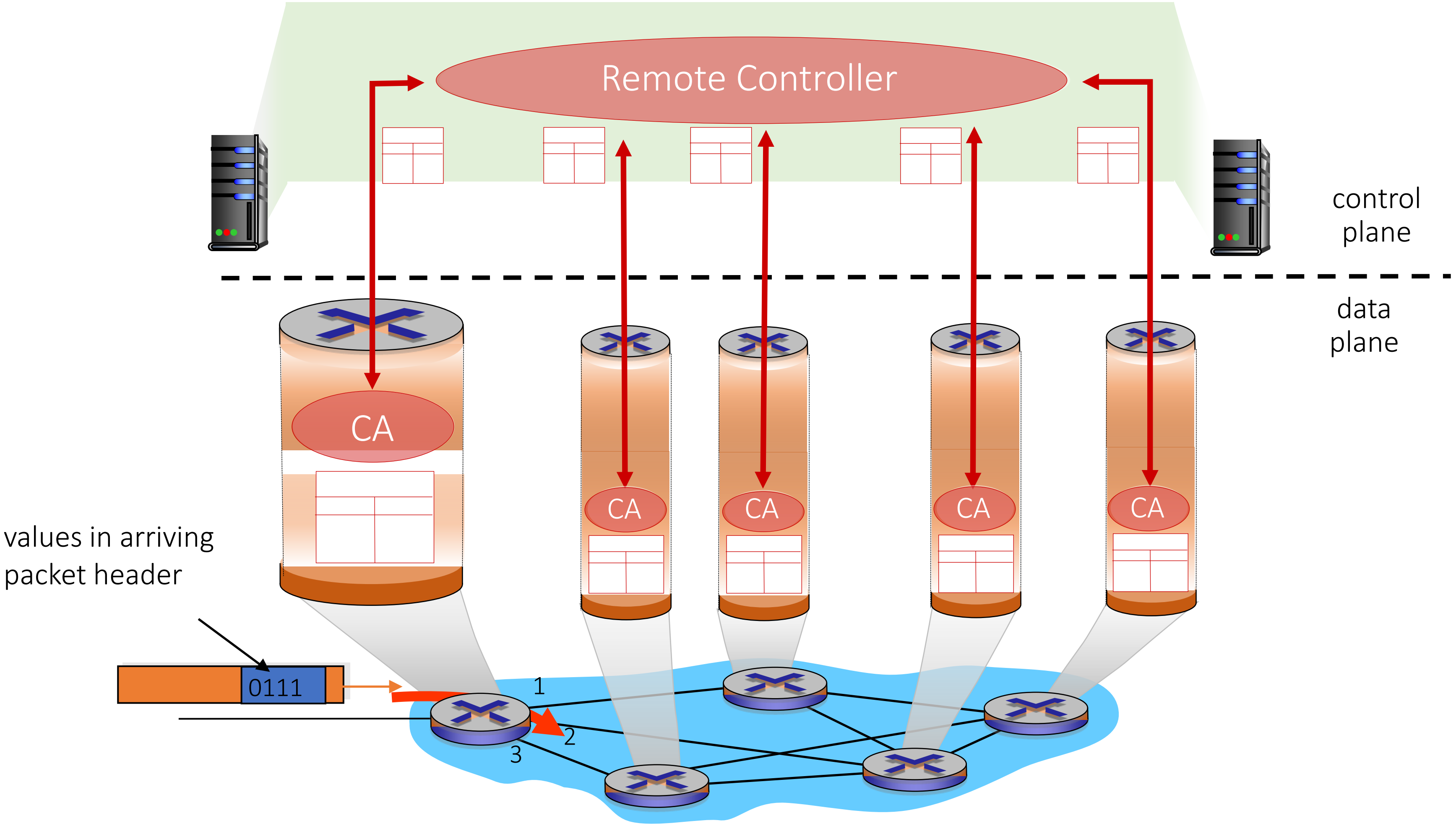


Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

(2) Software-Defined Networking (SDN) Control Plane

A distinct (typically remote) controller interacts with local control agents (CAs), 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 bounded delay e.g. less than 40 msec

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- Security by encryption

The Internet's network layer (IP) provides a single service, known as **best-effort** service.

IP's best effort service model has arguably proven to be more than "good enough" to enable an amazing range of applications, including streaming video services such as Netflix and voice-and-video-over-IP, real-time conferencing applications such as Skype and Facetime.

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*:*

1. successful datagram delivery to destination
2. timing or order of delivery
3. bandwidth available to end-end flow

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		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

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

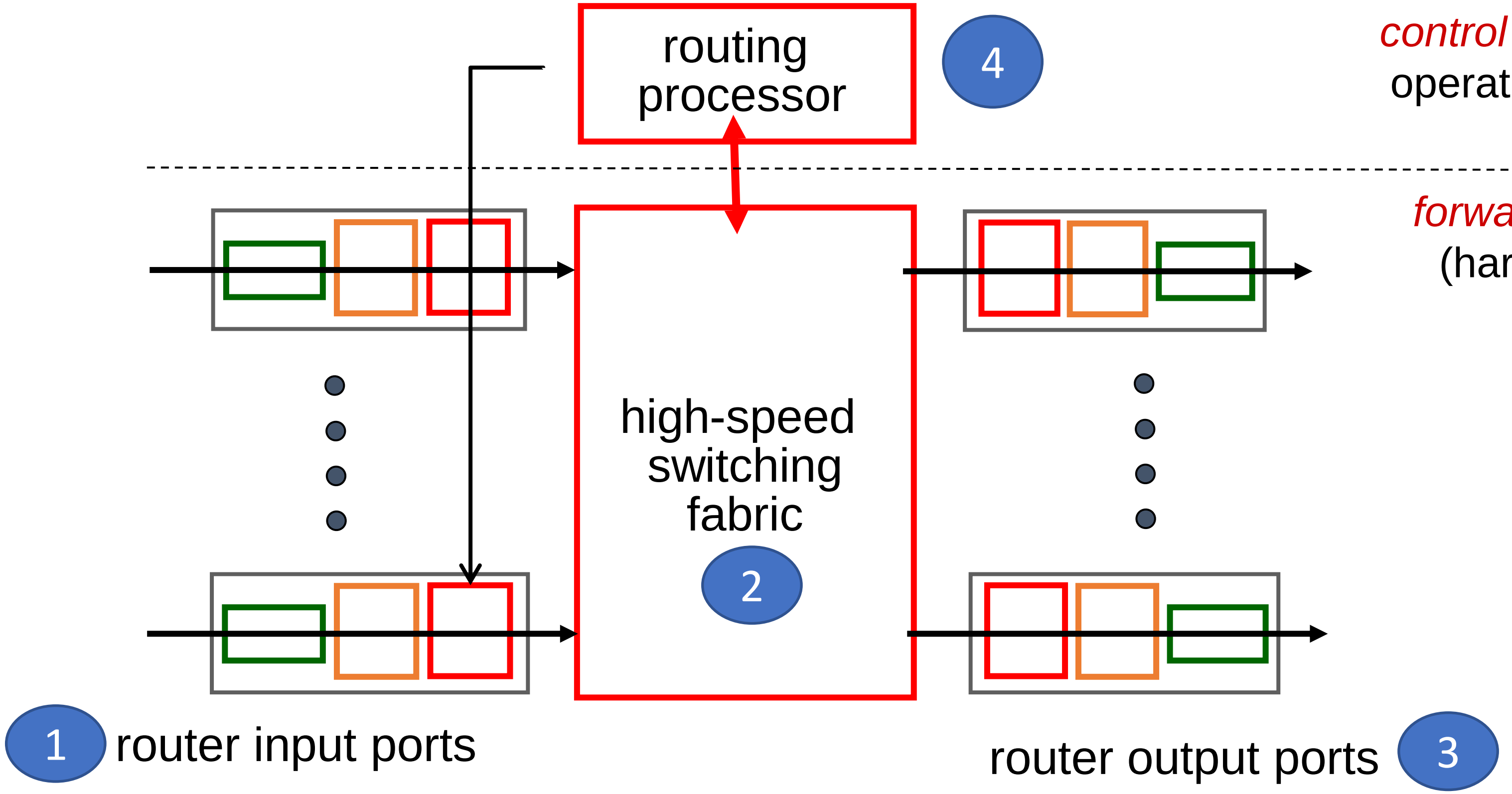
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

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

Router Architecture Overview

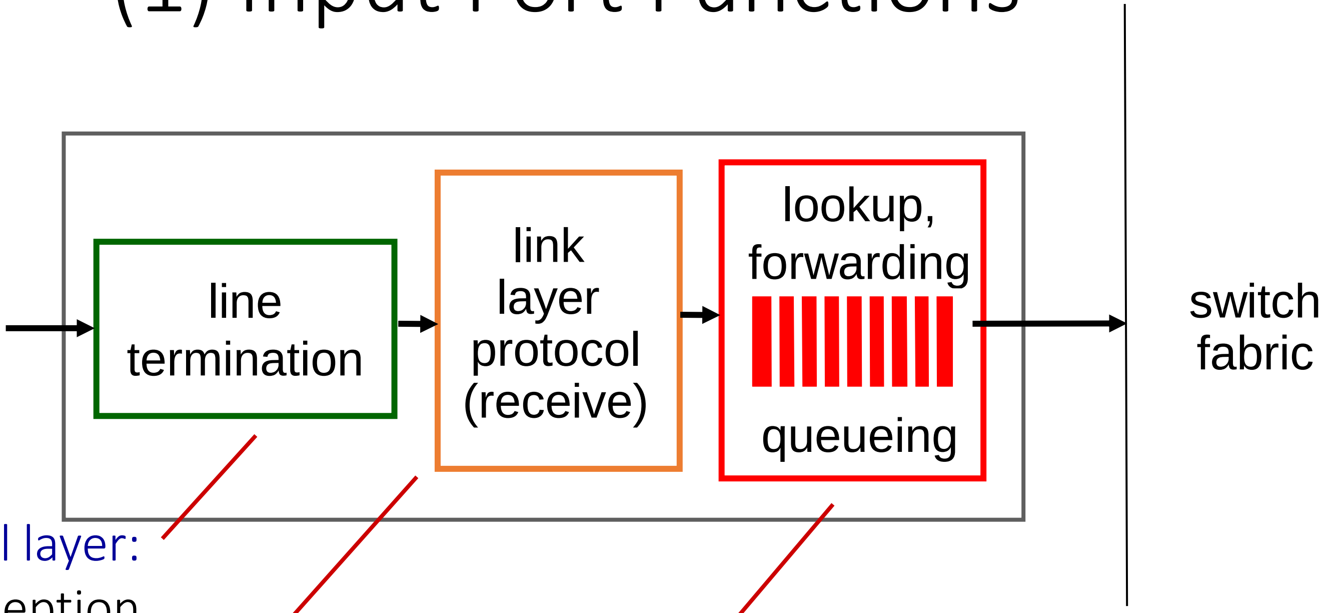
High-level view of generic router architecture (4 router components):



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

forwarding data plane (hardware) operates in nanosecond timeframe

(1) Input Port Functions



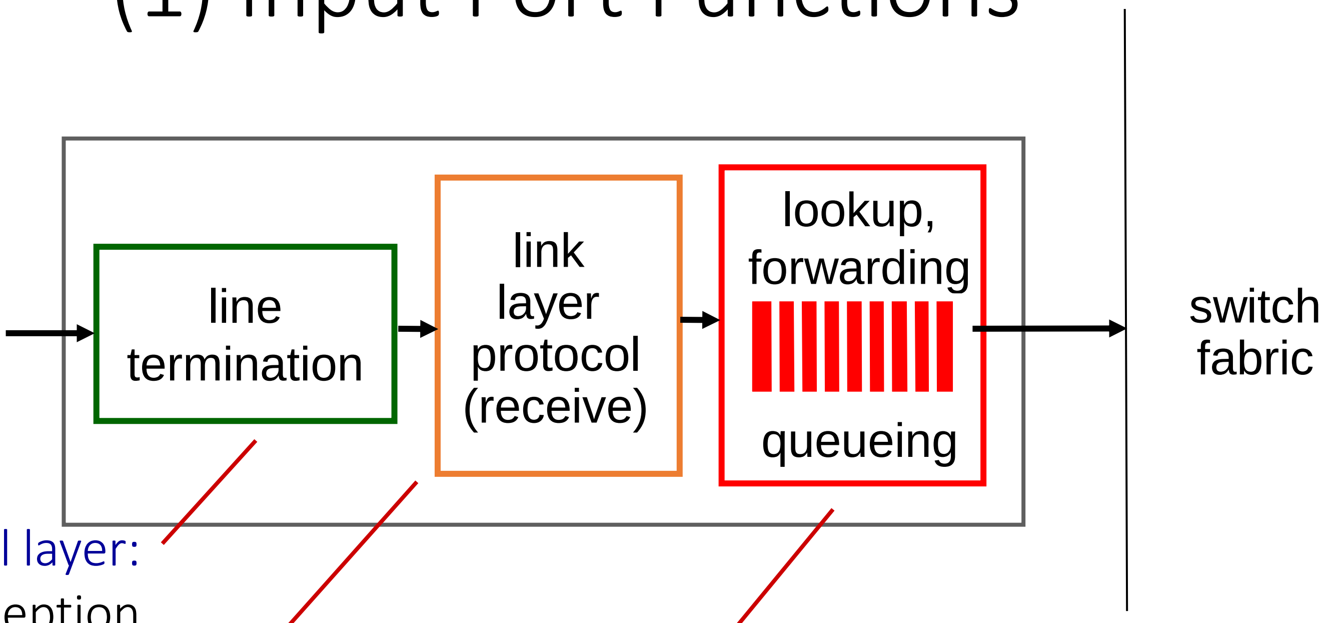
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see 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’
- queuing: if datagrams arrive faster than forwarding rate on switch fabric

(1) Input Port Functions



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 6

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	0
11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 11111111	
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:

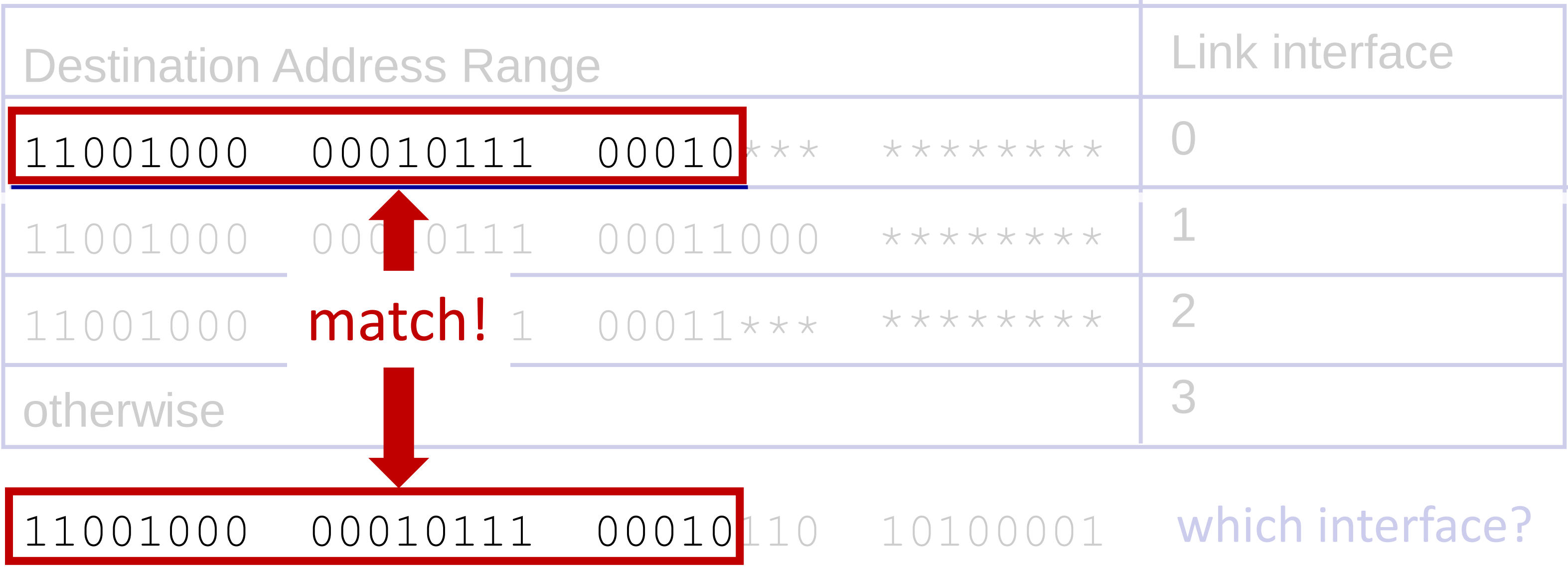
11001000 00010111 00010110 10100001 which interface?

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:

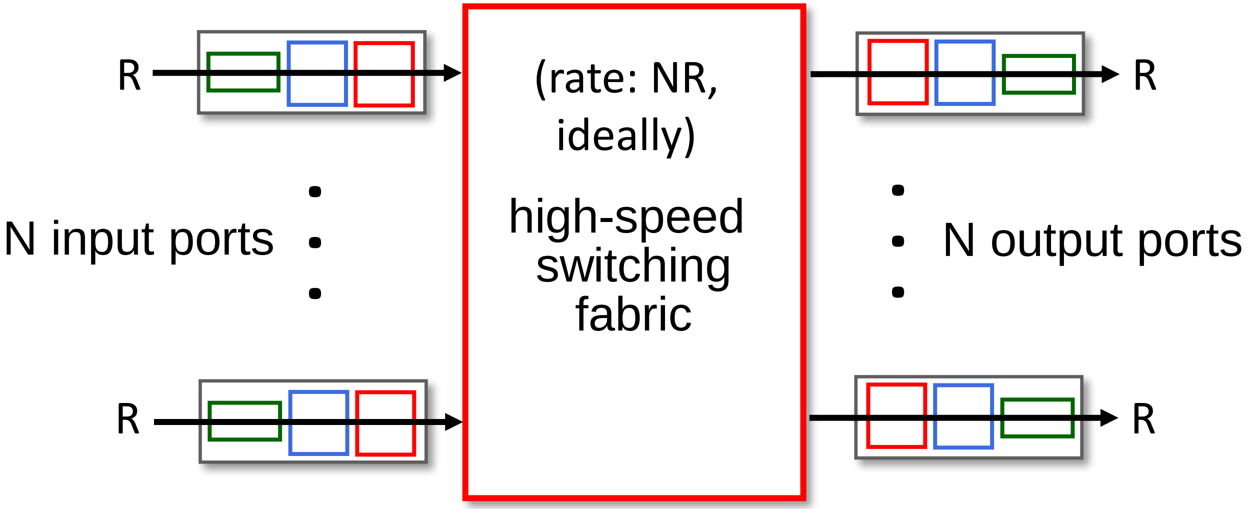
11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

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 support up to ~1M routing table entries in TCAM

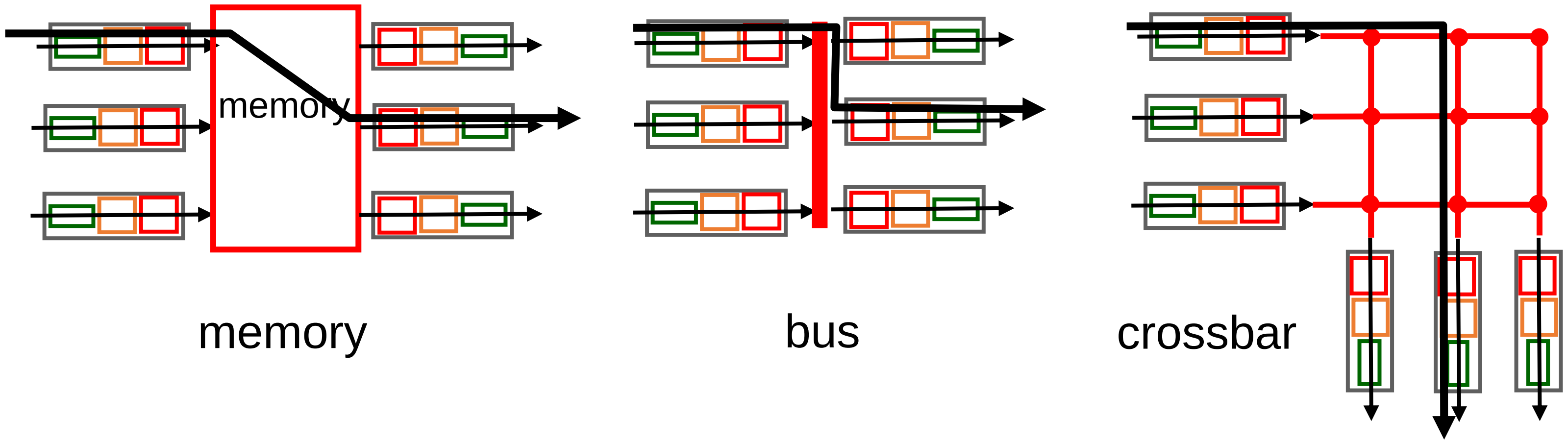
(2) Switching Fabrics

- 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 (rate: NR ideally)



(2) Switching Fabrics

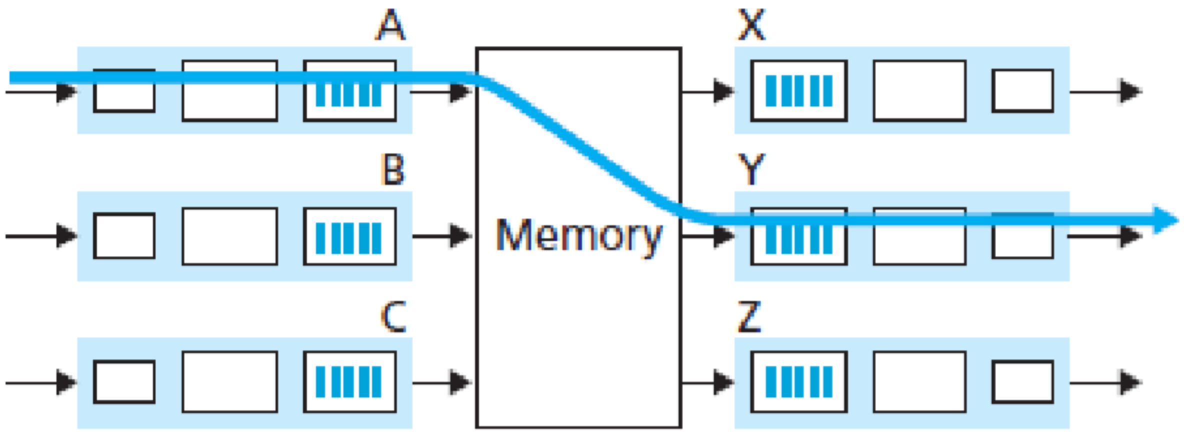
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable (rate: NR ideally)
- three types of switching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



Switching via Memory

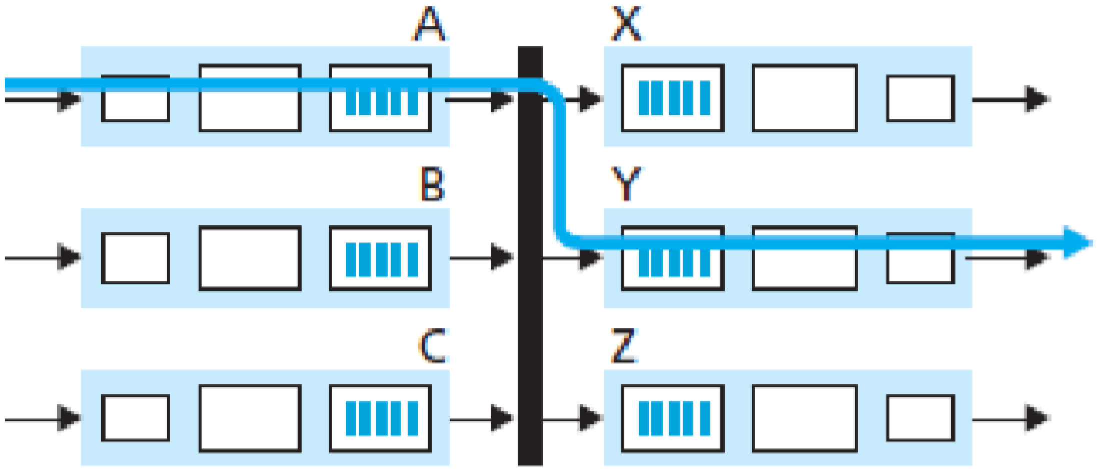
first generation routers:

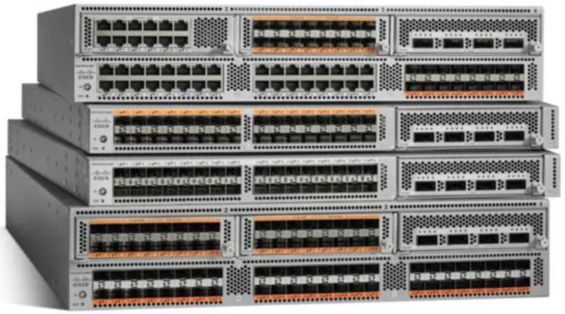
- traditional computers with switching under direct control of CPU
- packet copied to system’s memory
- speed limited by memory bandwidth



Switching via a Bus

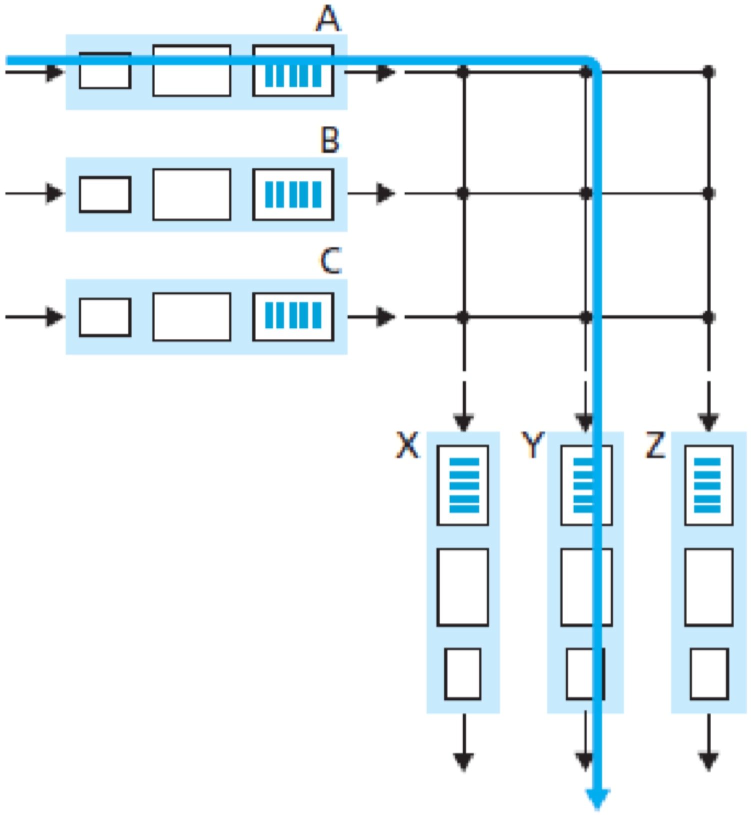
- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers





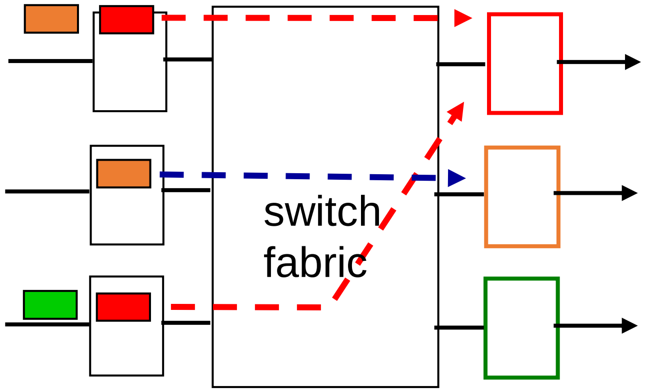
Switching via Interconnection Network

- overcome bus bandwidth limitations
- crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: (with multiple switching fabrics e.g. N fabrics) fragmenting datagram into fixed length cells, switch (spray) cells through the N fabrics, reassemble datagram at exit.
- Cisco 12000: switches 60 Gbps through the interconnection network

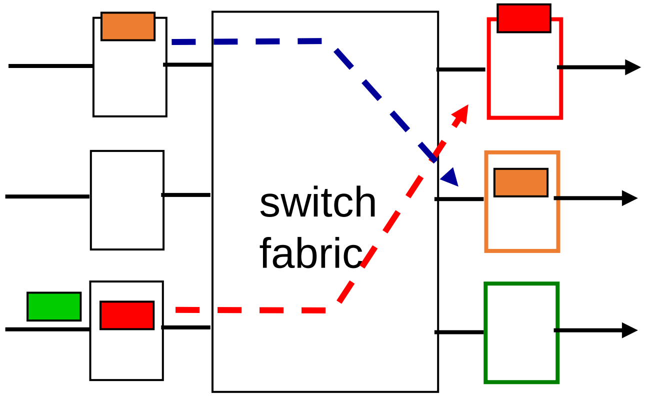


Input Port Queuing

- fabric slower than input ports combined -> queueing may occur at input queues
  - *queuing 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

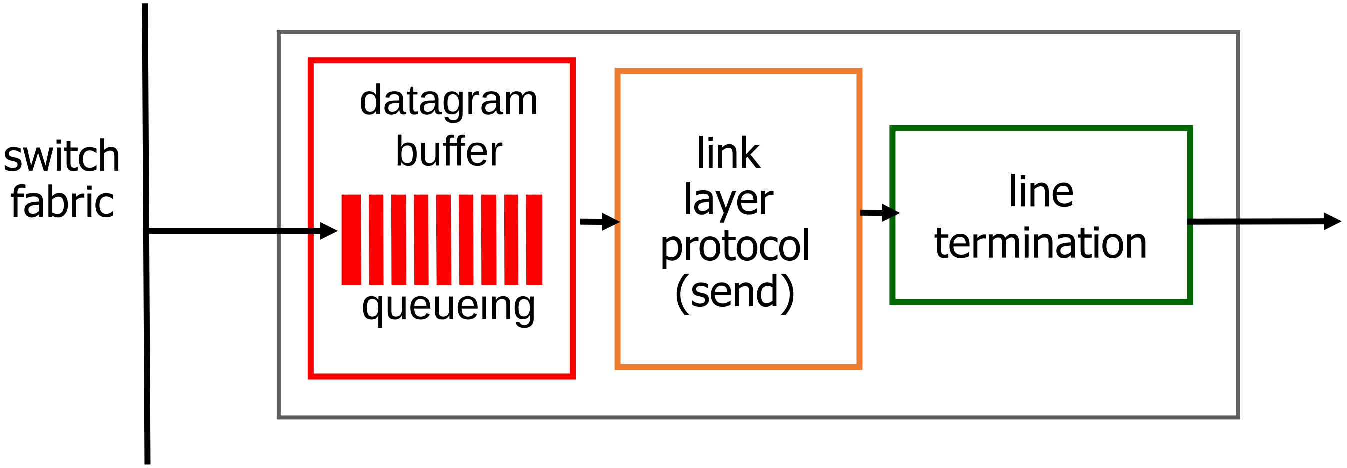


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

(3) Output Ports



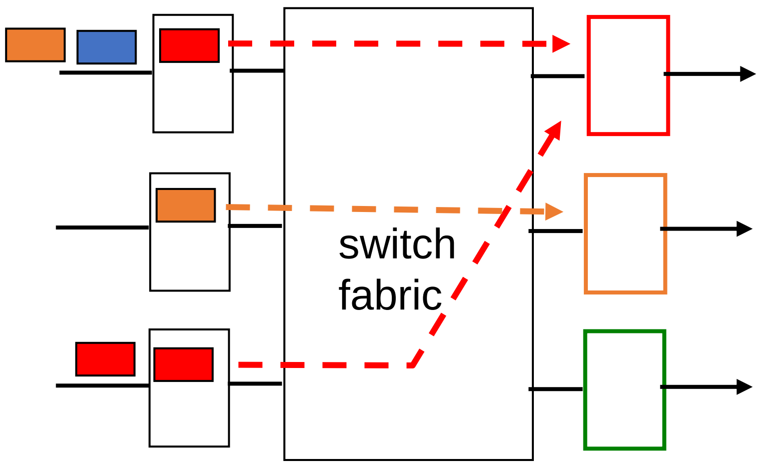
- *buffering* required when datagrams arrive from fabric faster than the transmission rate. Drop policy: which datagrams to drop if no free buffers?

- *scheduling discipline* chooses among queued datagrams for transmission

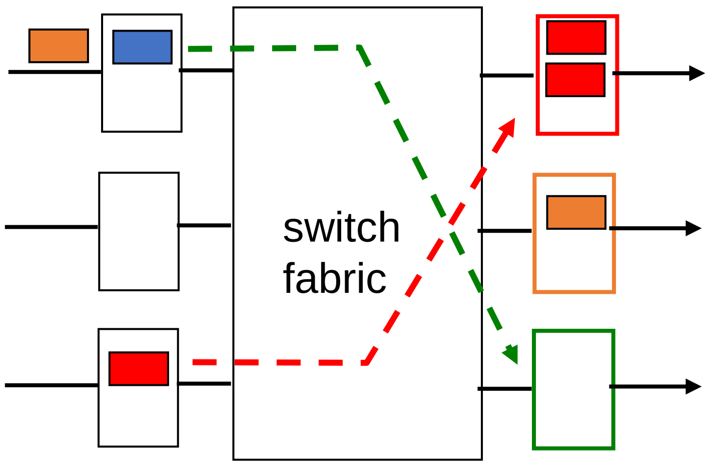
Datagram (packets) can be lost due to congestion, lack of buffers

Priority scheduling – who gets best performance

Output Port Queuing



at *t*, packets more
from input to output

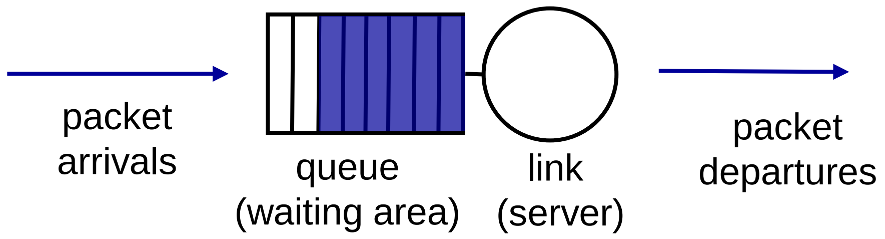


one packet time later

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

Scheduling Mechanisms

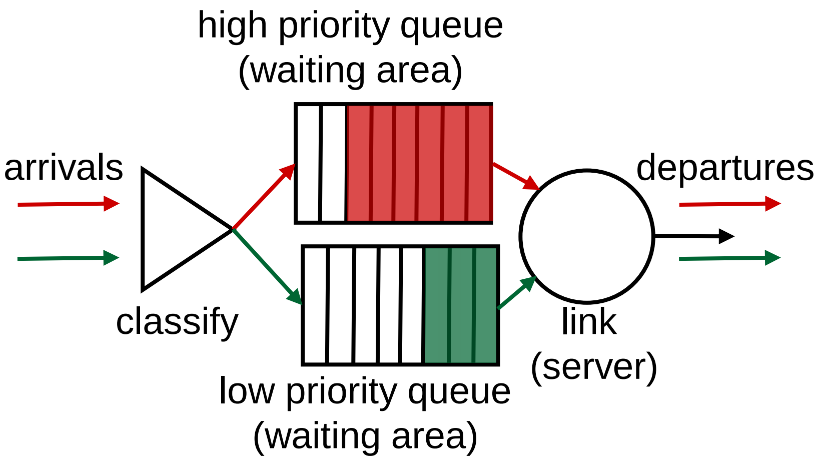
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority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

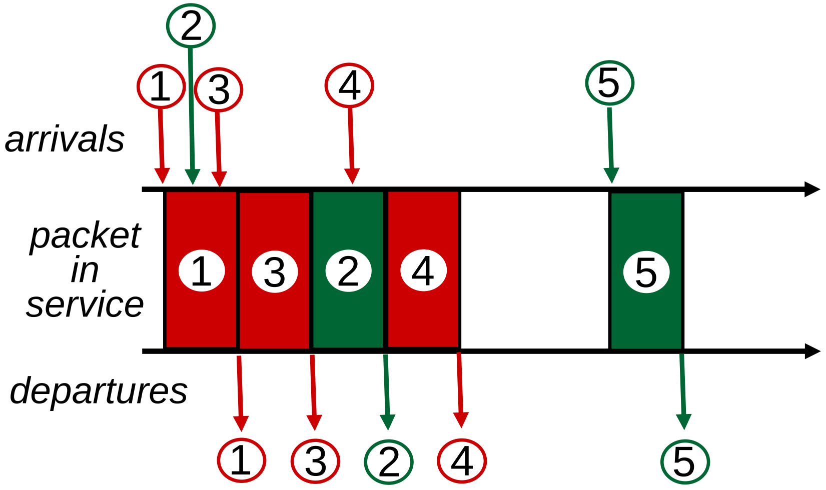


Scheduling Policies: Priority

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?

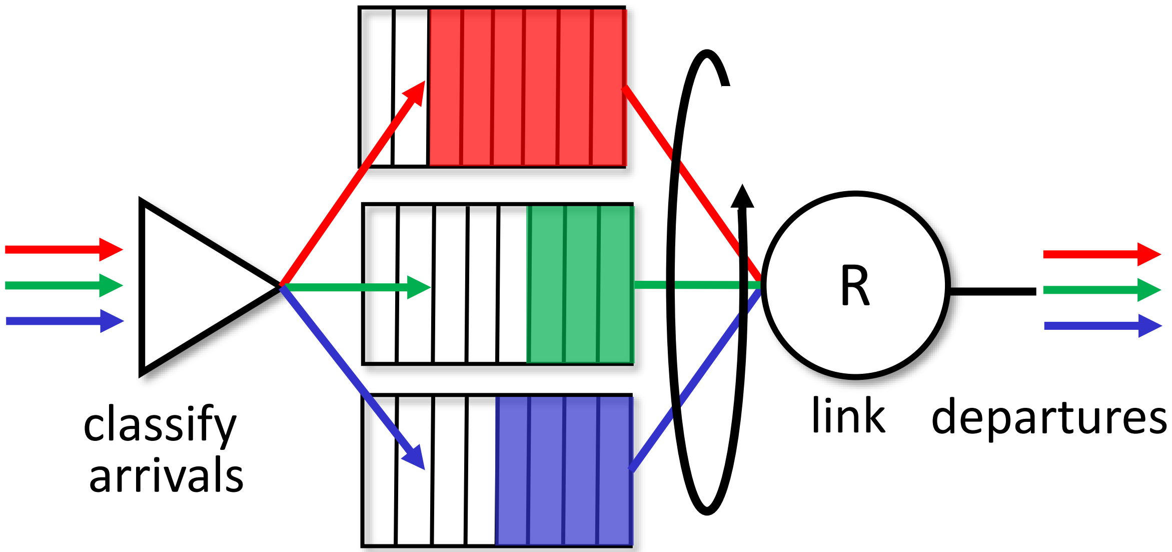


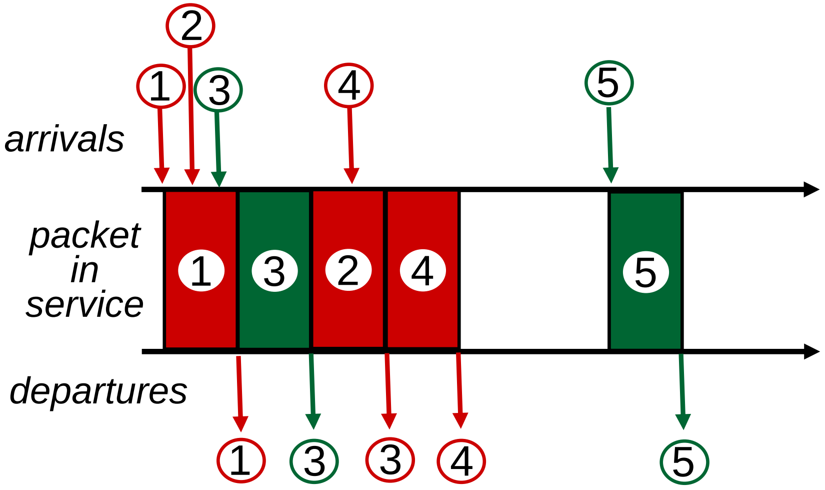


Scheduling Policies: still more

Round Robin (RR) scheduling:

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

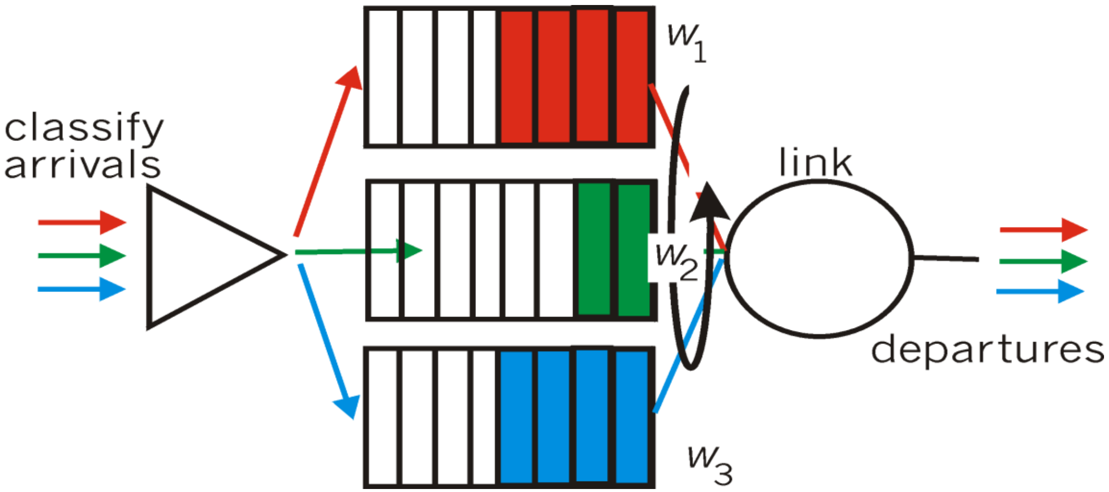




Scheduling Policies: still more

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

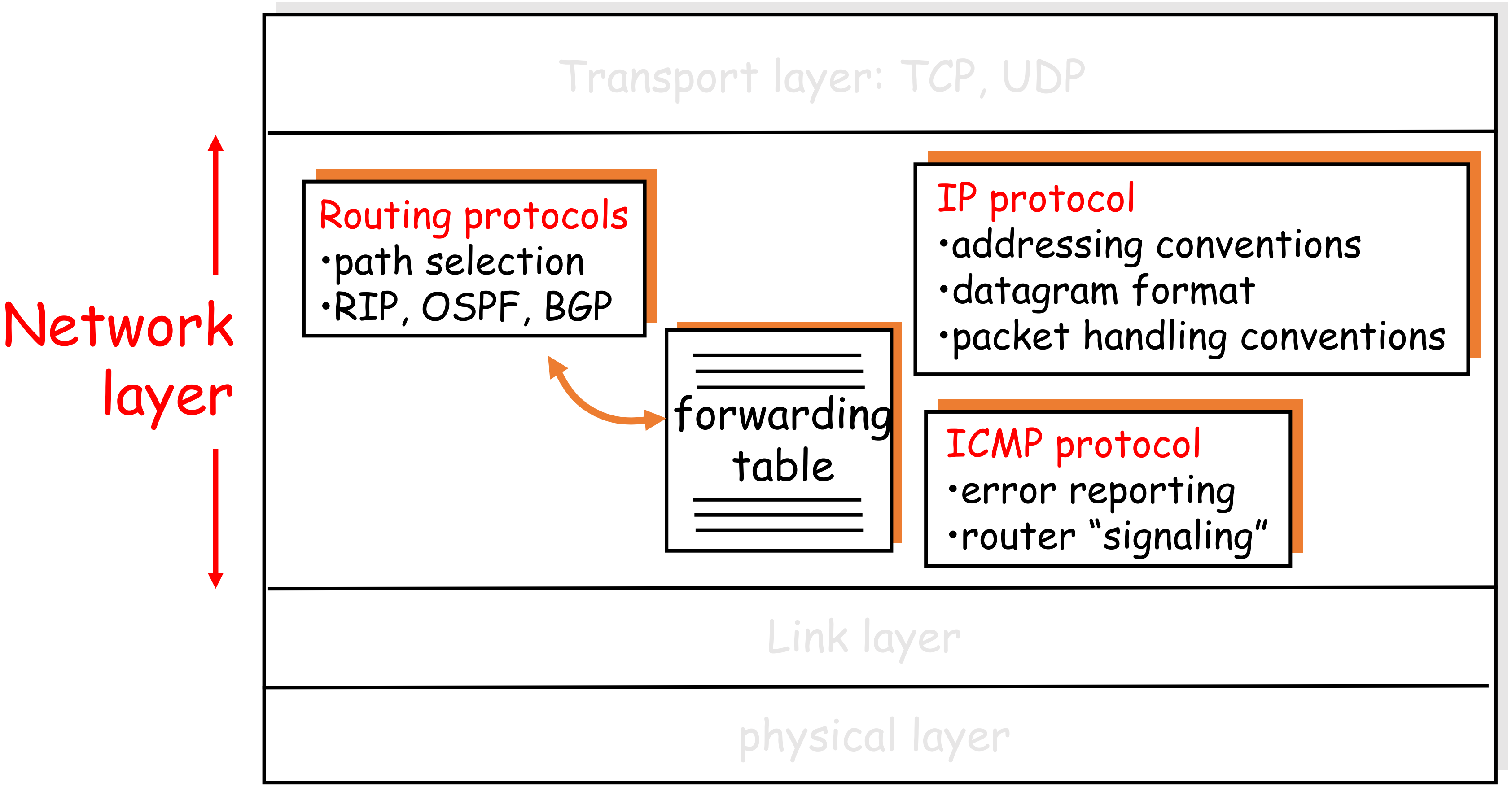
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

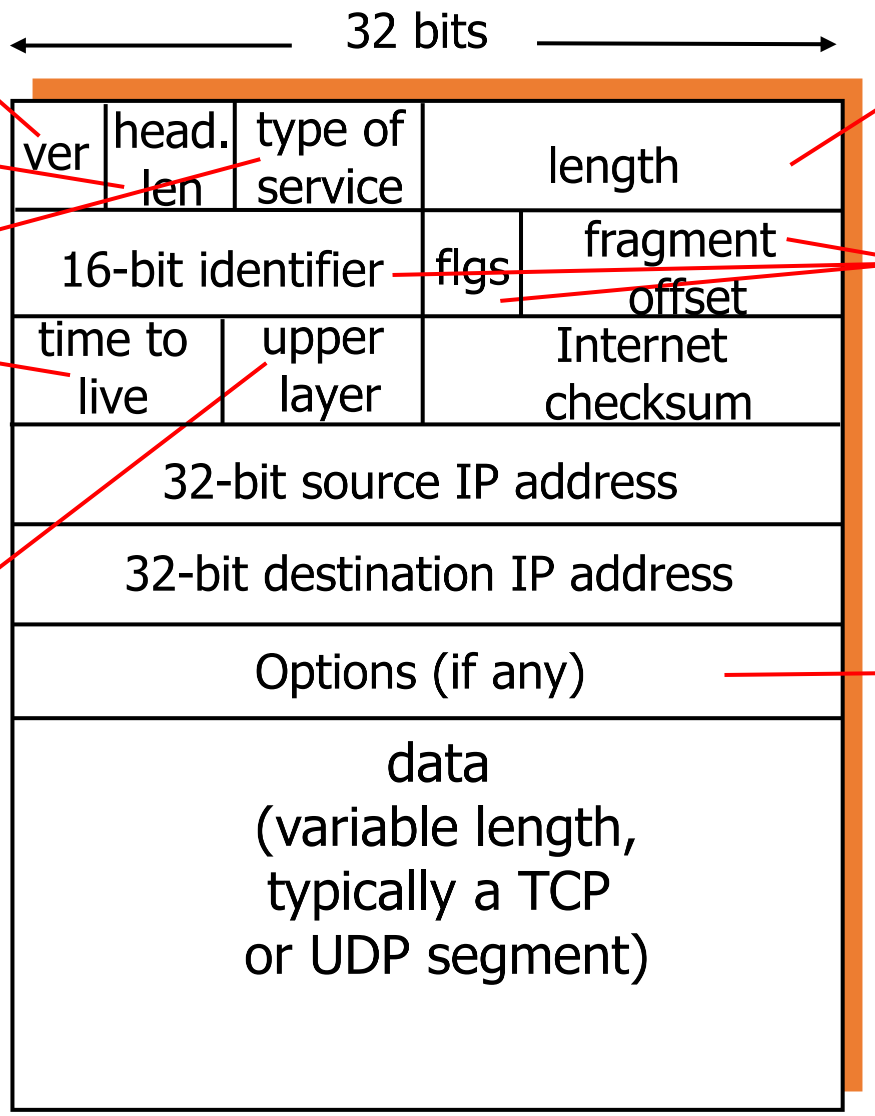
- match
- action
- OpenFlow examples of match-plus-action in action

What does the Network layer consist of?

Host, router network layer functions:



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

total datagram length (bytes)

for fragmentation/ reassembly

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

<u>how much overhead with TCP?</u>

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

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

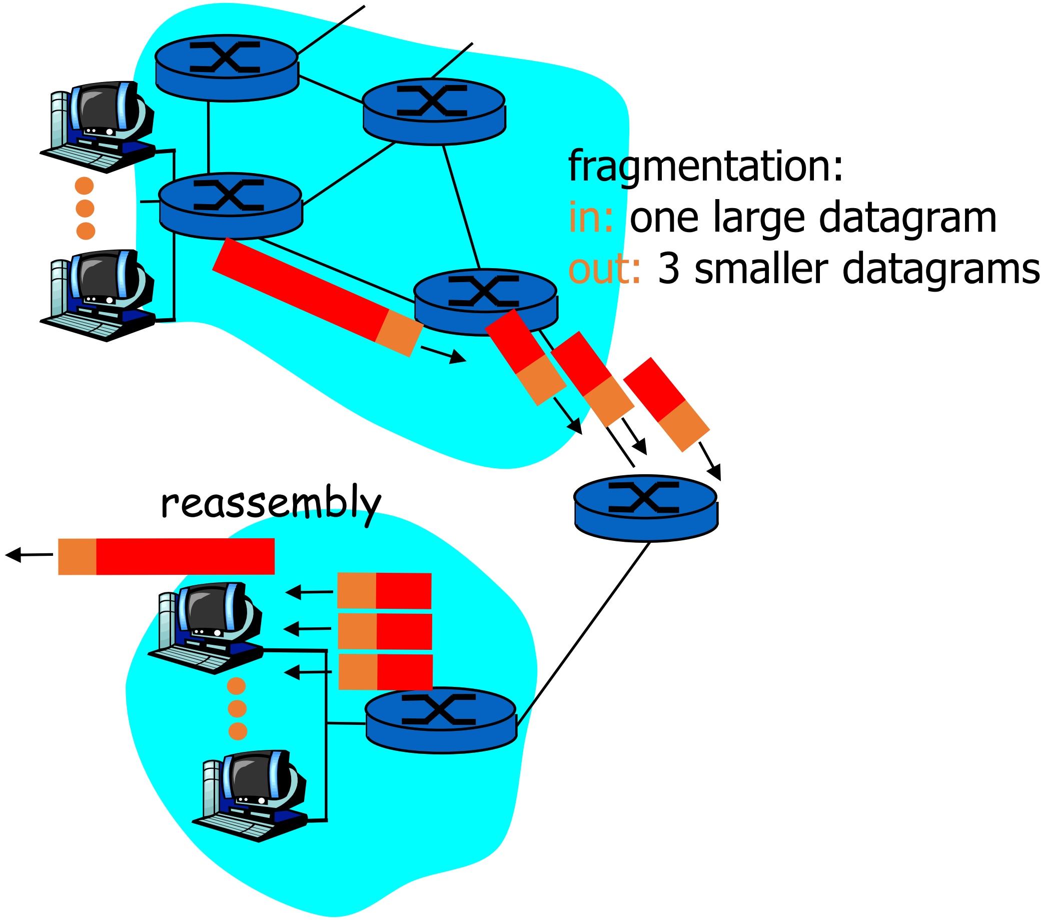
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

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

IP Fragmentation & Reassembly

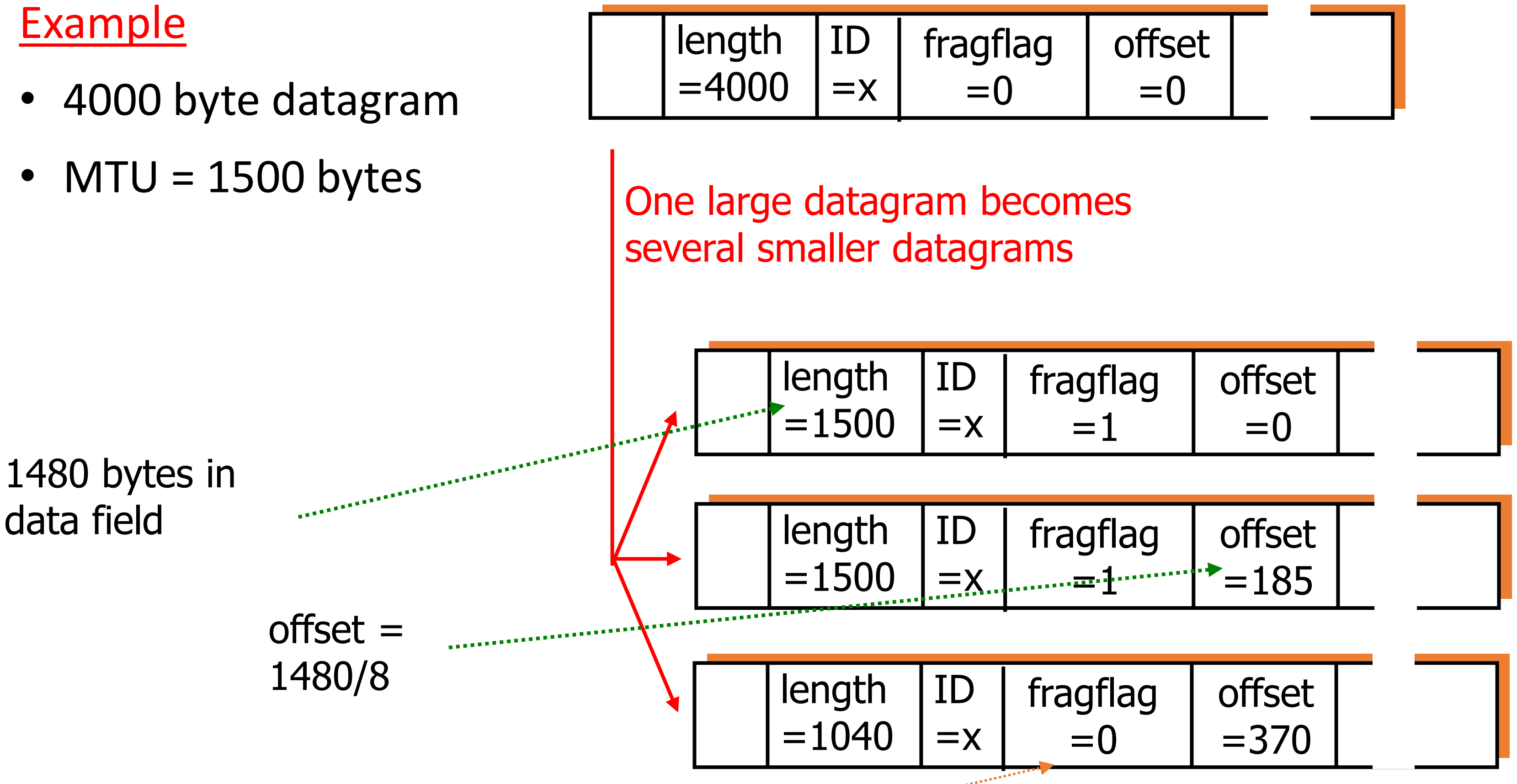
- network links have MTU (max. transmission unit) - largest possible link-level frame.
    - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
    - one datagram becomes several datagrams
    - "reassembled" only at final destination
    - IP header bits used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



IP Fragmentation & Reassembly

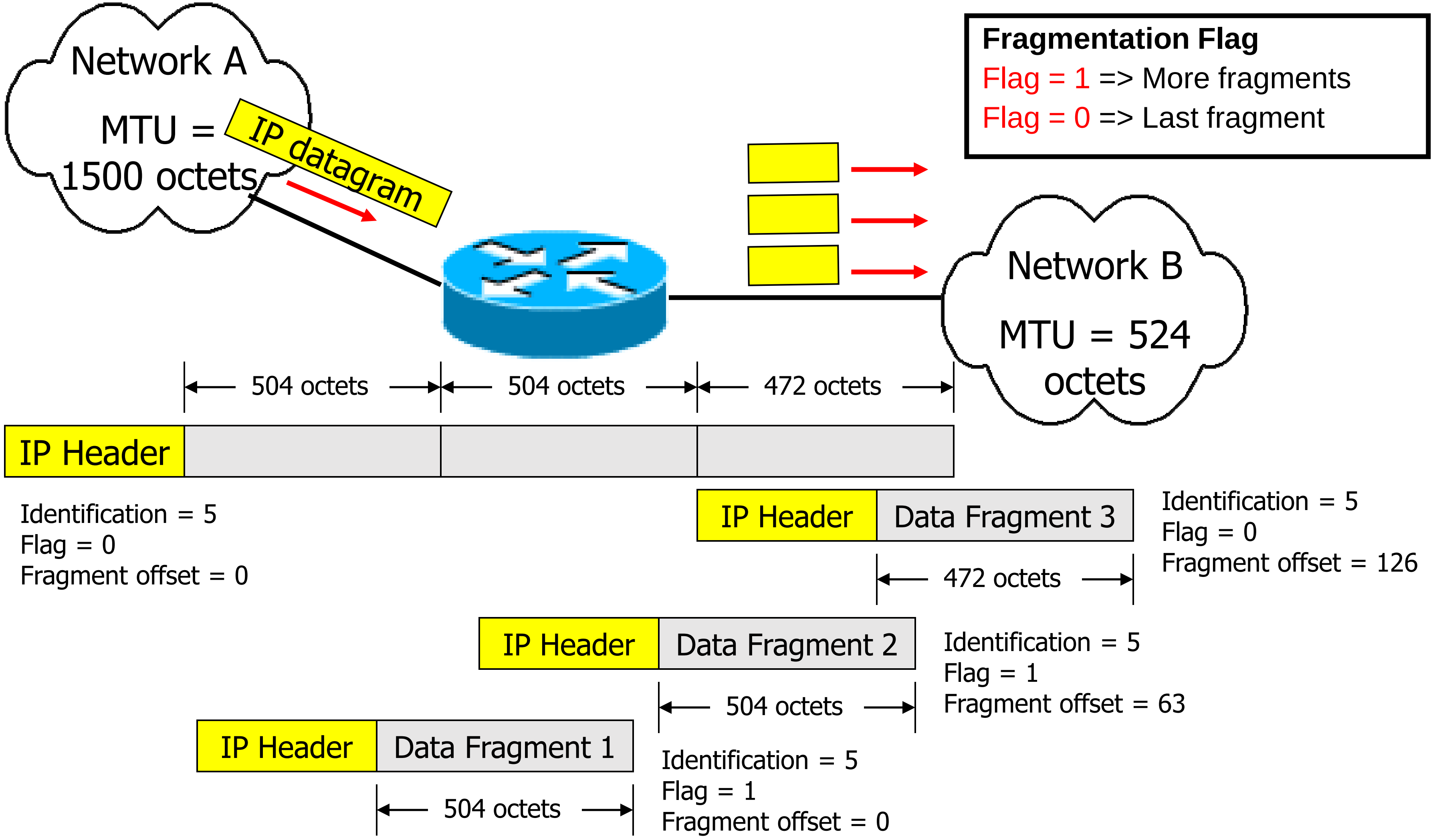
Example

- 4000 byte datagram
- MTU = 1500 bytes



Fragmentation Flag
Flag = 1 => More fragments
Flag = 0 => Last fragment

Fragmentation Flags & Fragment Offset



Network A

MTU = 1500 octets

IP datagram

Fragmentation Flag

Flag = 1 => More fragments

Flag = 0 => Last fragment

Network B

MTU = 524 octets

504 octets | 504 octets | 472 octets

IP Header

Identification = 5
Flag = 0
Fragment offset = 0

IP Header | Data Fragment 3

472 octets

Identification = 5
Flag = 0
Fragment offset = 126

IP Header | Data Fragment 2

504 octets

Identification = 5
Flag = 1
Fragment offset = 63

IP Header | Data Fragment 1

504 octets

Identification = 5
Flag = 1
Fragment offset = 0

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

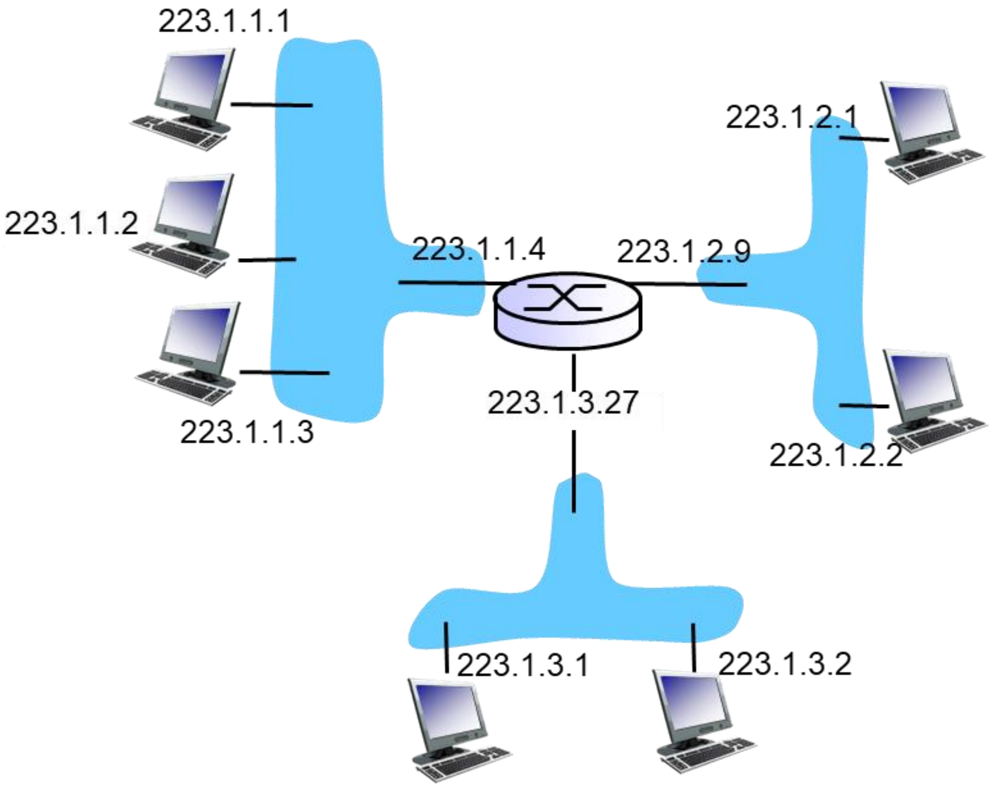
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

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 may have multiple interfaces
  - IP addresses associated with each interface

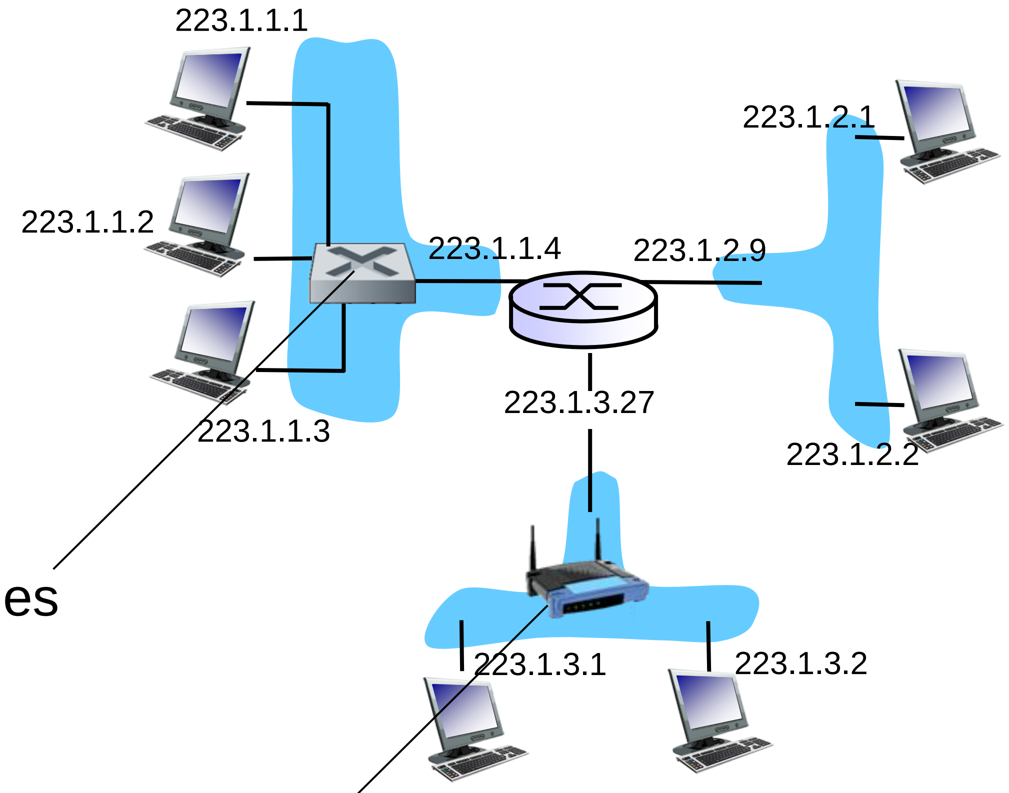


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP Addressing: Introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapters 6, 7.



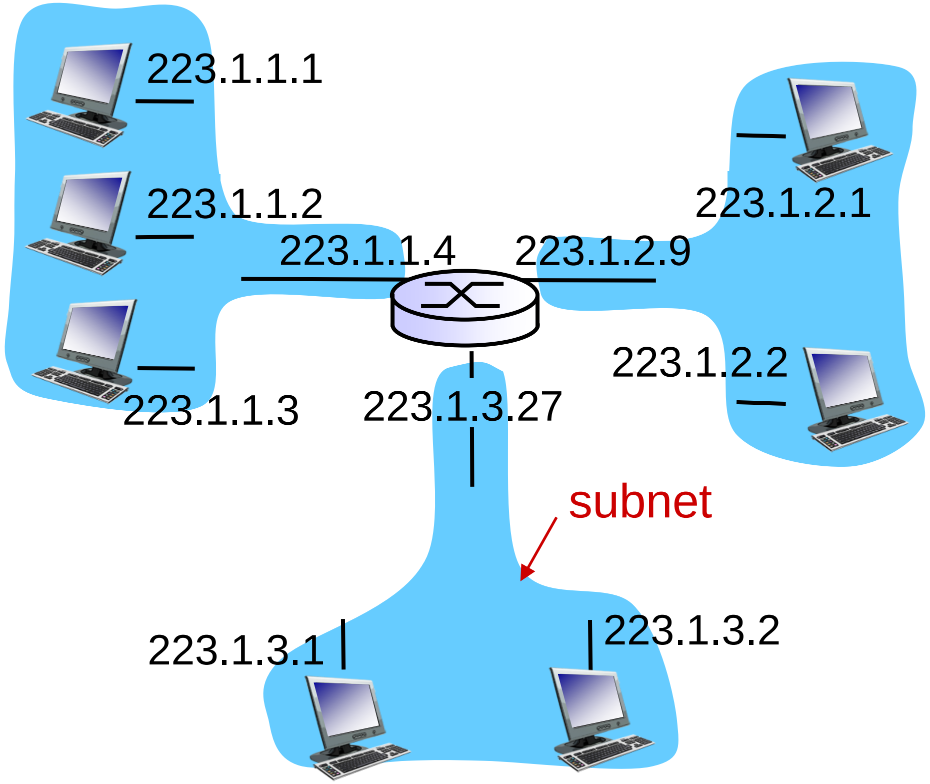
A: wired Ethernet interfaces connected by Ethernet switches

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

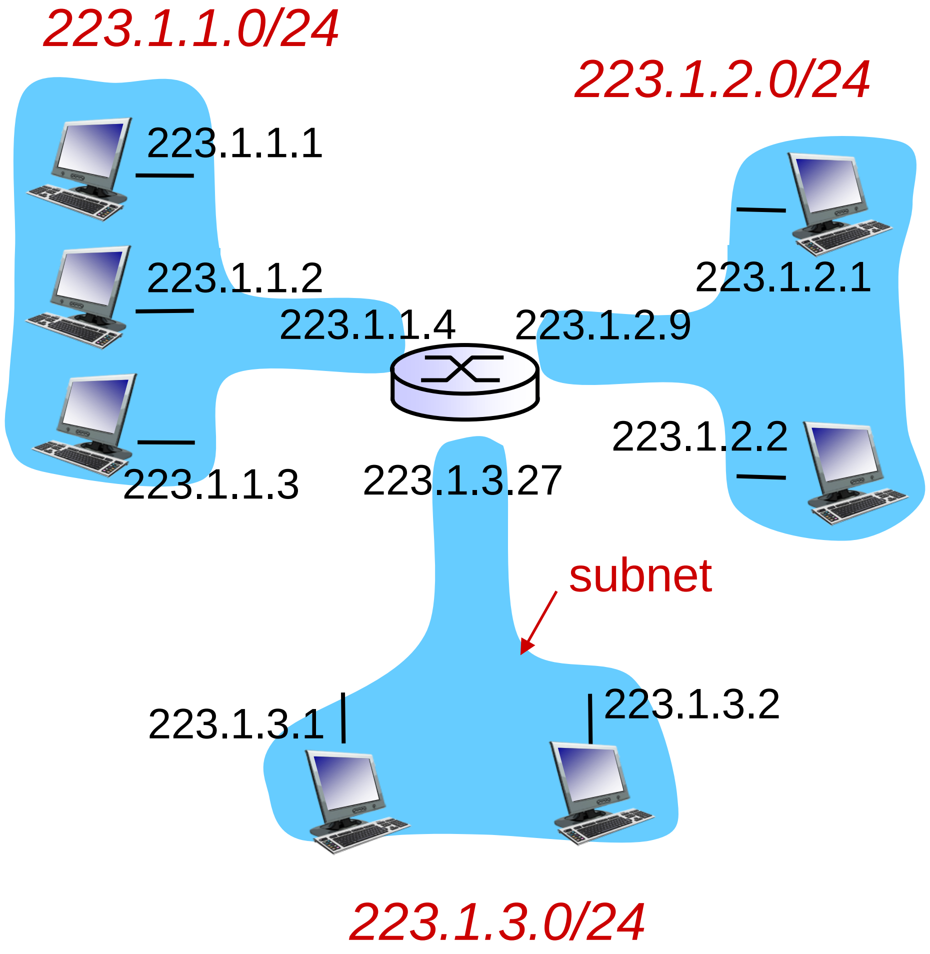


network consisting of 3 subnets

Subnets

Recipe

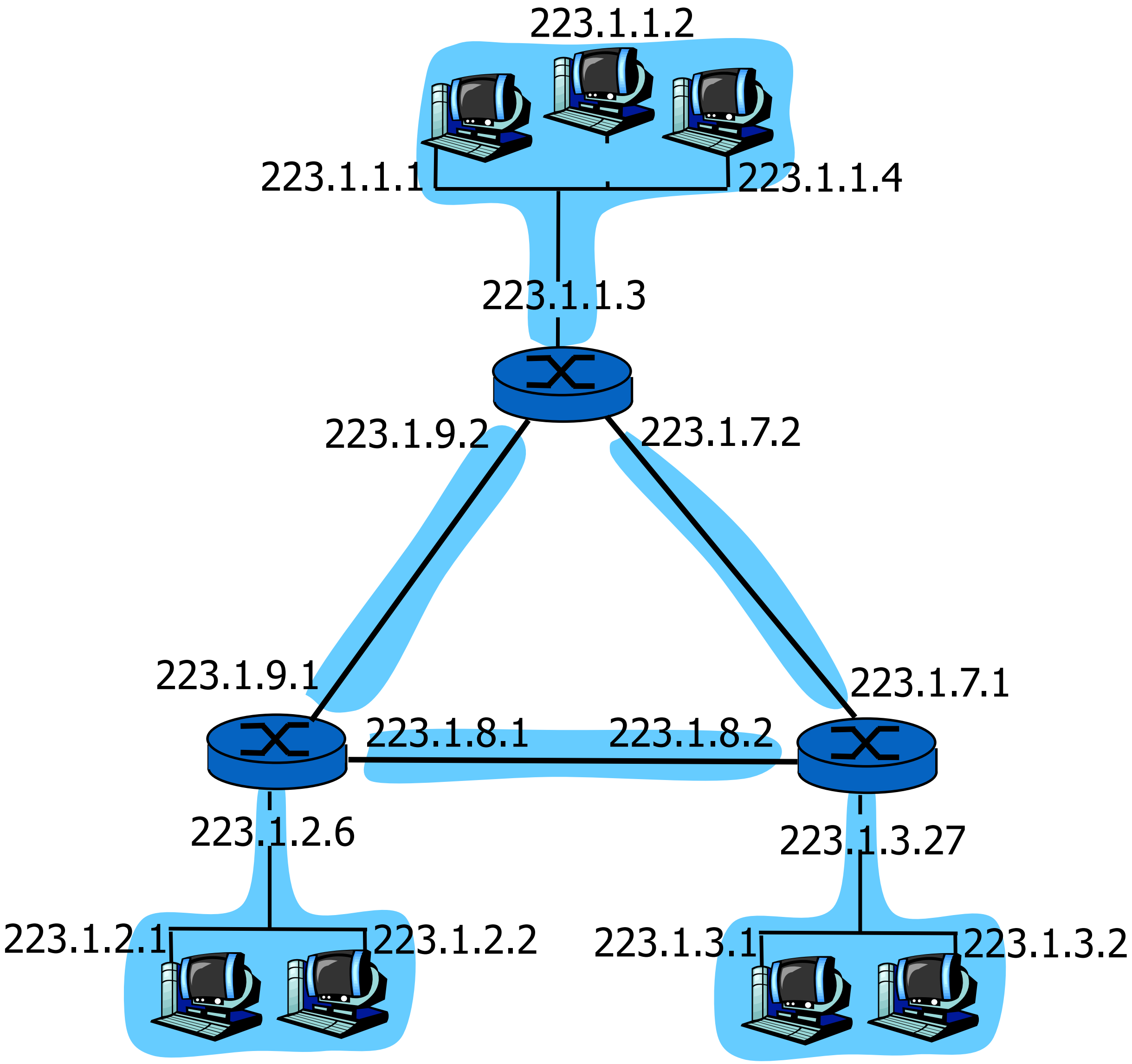
To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



subnet mask: /24

Subnets

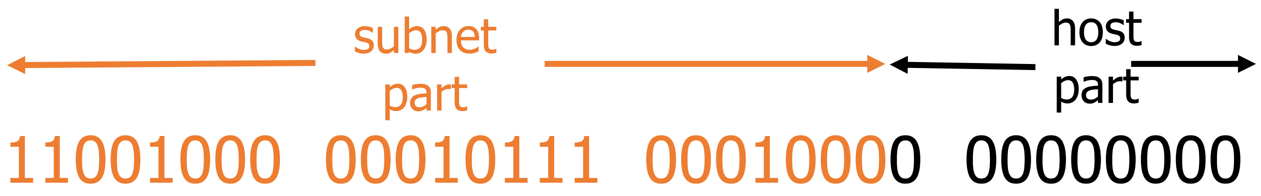
How many subnets?



Addressing in the Internet

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
- Before CIDR, Internet used a class-based addressing scheme where x could be 8, 16, or 24 bits. These correspond to classes A, B, and C respectively.



subnet part

host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP addresses: how to get one?

Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Win: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config

- DHCP: Dynamic Host Configuration Protocol: dynamically get address from a server
  - this is becoming very popular

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)

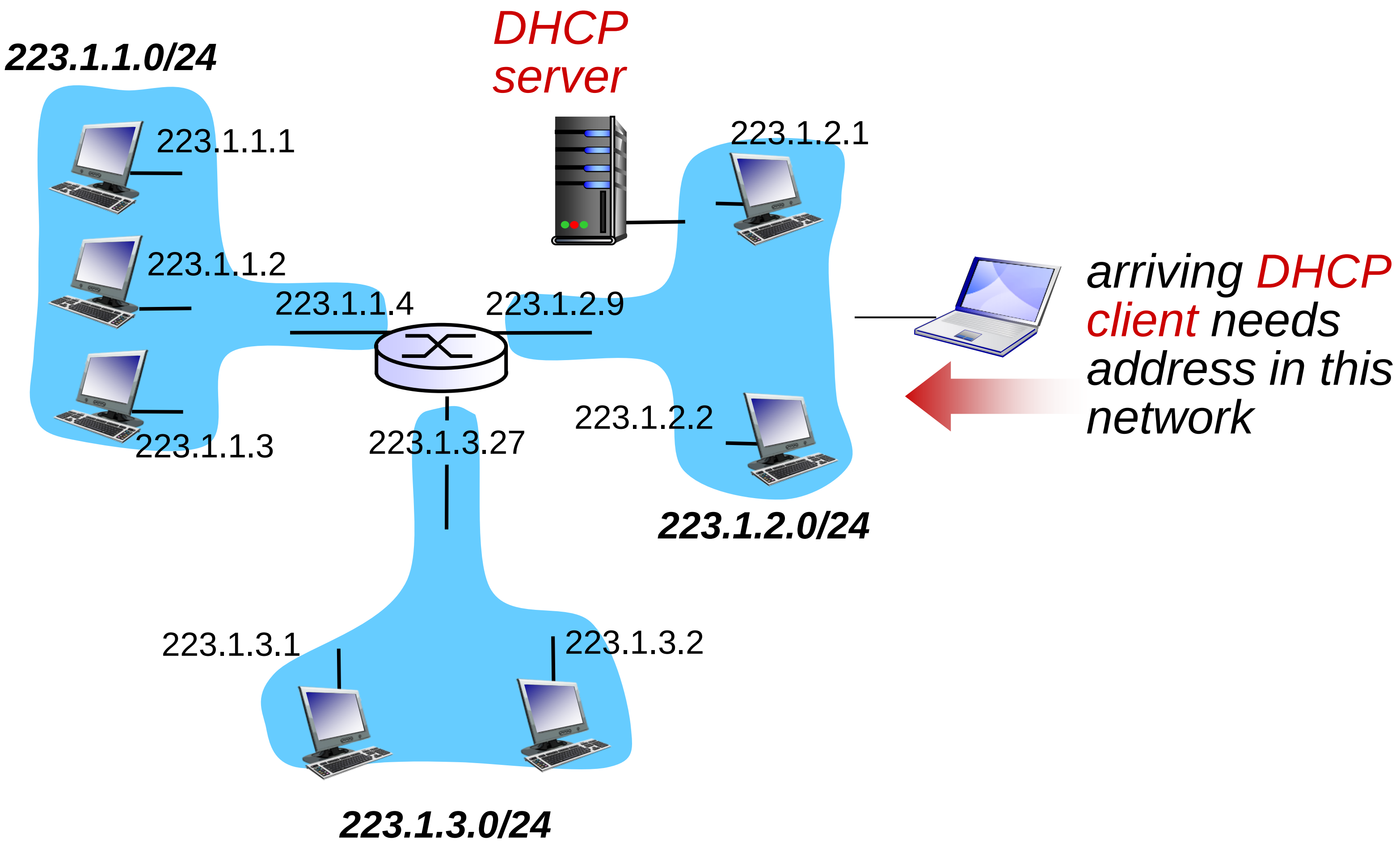
DHCP overview:

- 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

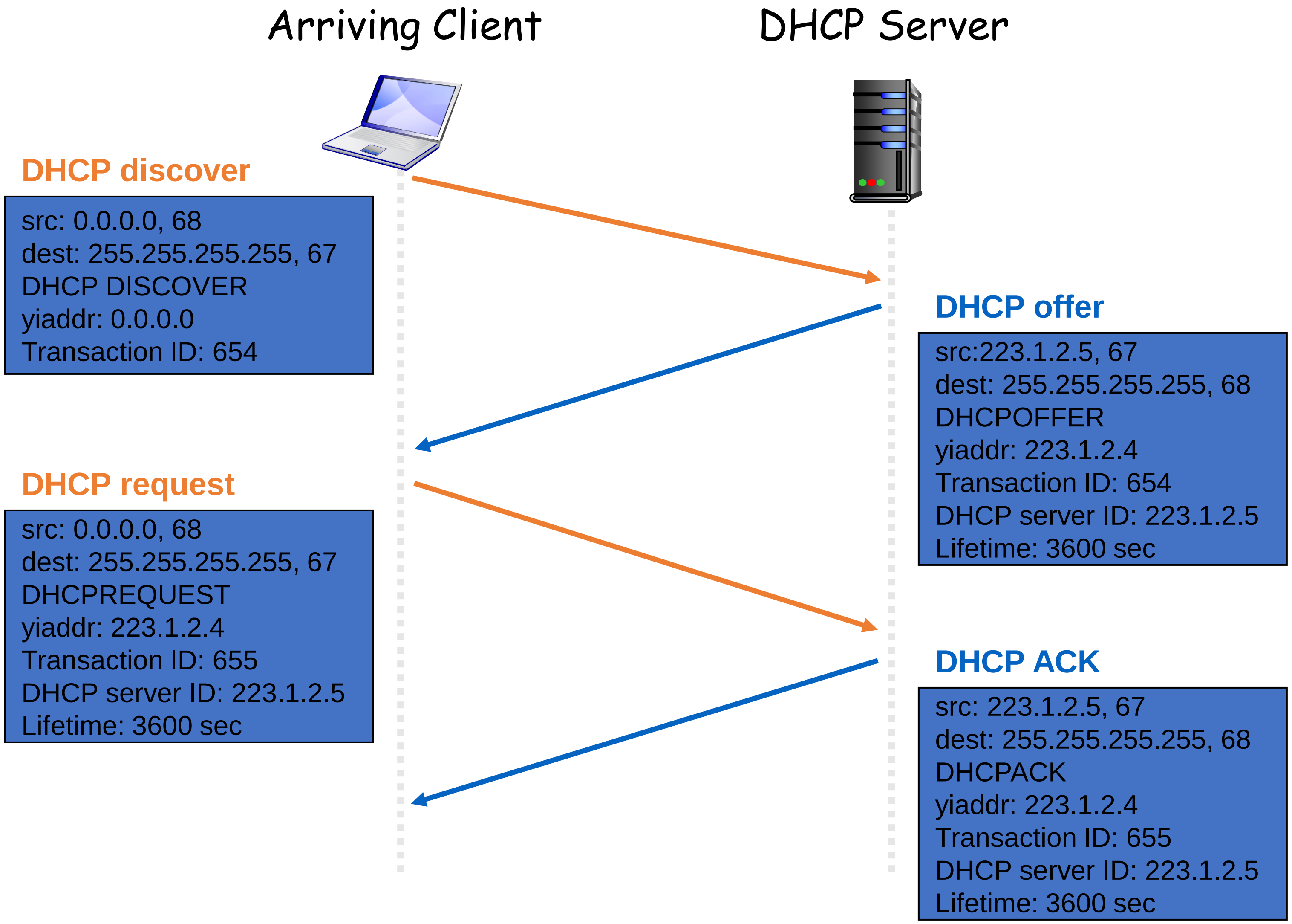
DHCP: Dynamic Host Configuration Protocol

- Used to assign IP address to hosts dynamically.
- Client-server protocol.
- Clients : obtain network configuration information e.g. IP address, subnet mask, default gateway, DNS
- Server : allocates configuration information
    - If no server is present on the network, a DHCP relay agent (in a router that knows the address of DHCP server for that network) is need.

DHCP Client-Server Scenario



arriving DHCP client needs address in this network



Arriving Client

DHCP Server

DHCP discover

src: 0.0.0.0, 68
dest: 255.255.255.255, 67
DHCP DISCOVER
yiaddr: 0.0.0.0
Transaction ID: 654

DHCP offer

src:223.1.2.5, 67
dest: 255.255.255.255, 68
DHCPOFFER
yiaddr: 223.1.2.4
Transaction ID: 654
DHCP server ID: 223.1.2.5
Lifetime: 3600 sec

DHCP request

src: 0.0.0.0, 68
dest: 255.255.255.255, 67
DHCPREQUEST
yiaddr: 223.1.2.4
Transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 sec

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
DHCPACK
yiaddr: 223.1.2.4
Transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 sec

DHCP: more than IP addresses

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

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

IP addresses: how to get one?

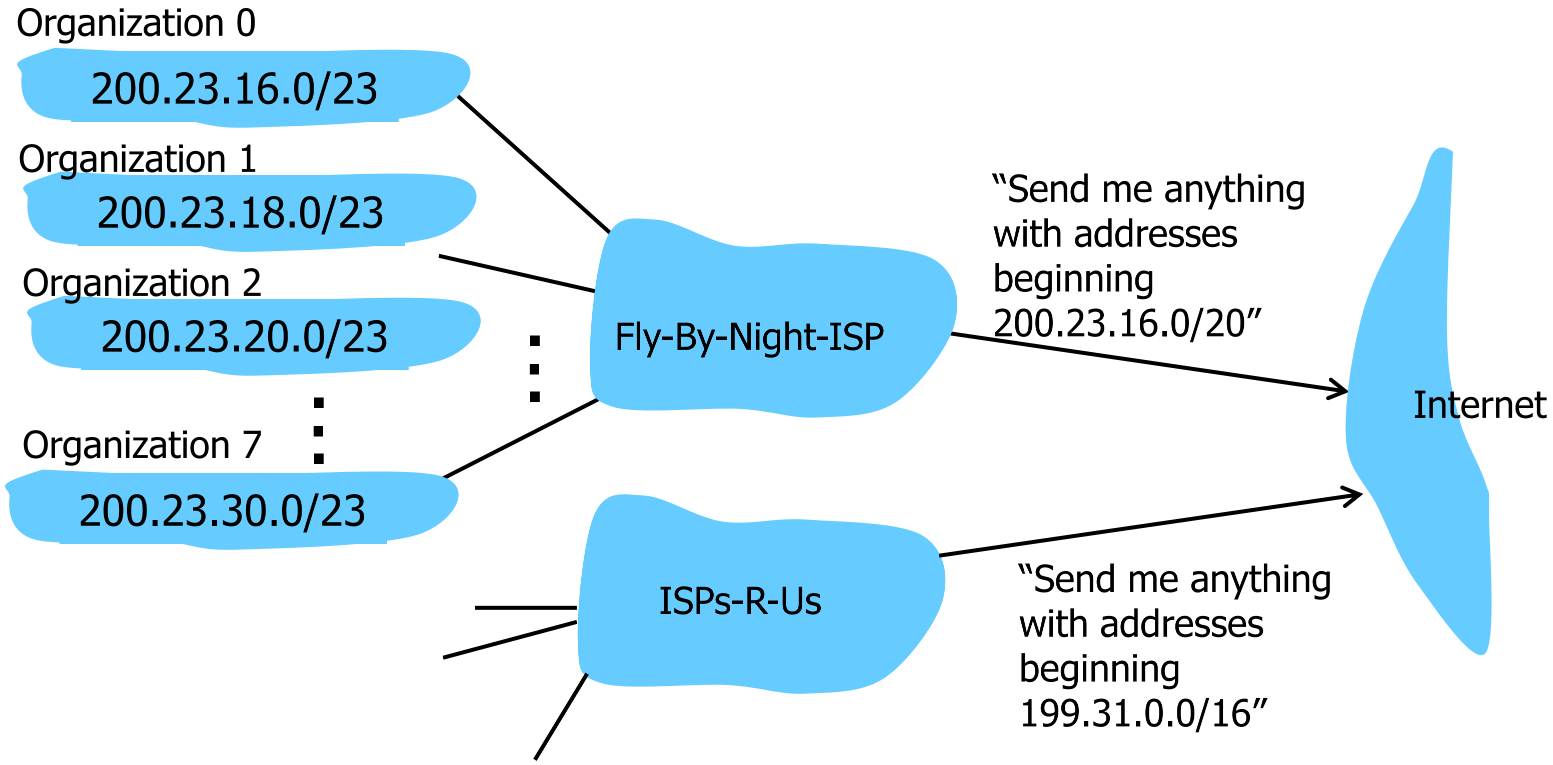
Q: How does *network* get subnet part of IP address?

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

ISP's block	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

- ISP has an address block; it can further divide this block into sub-blocks and assign them to subscriber organizations.
- Hierarchical addressing allows efficient advertisement of routing information:

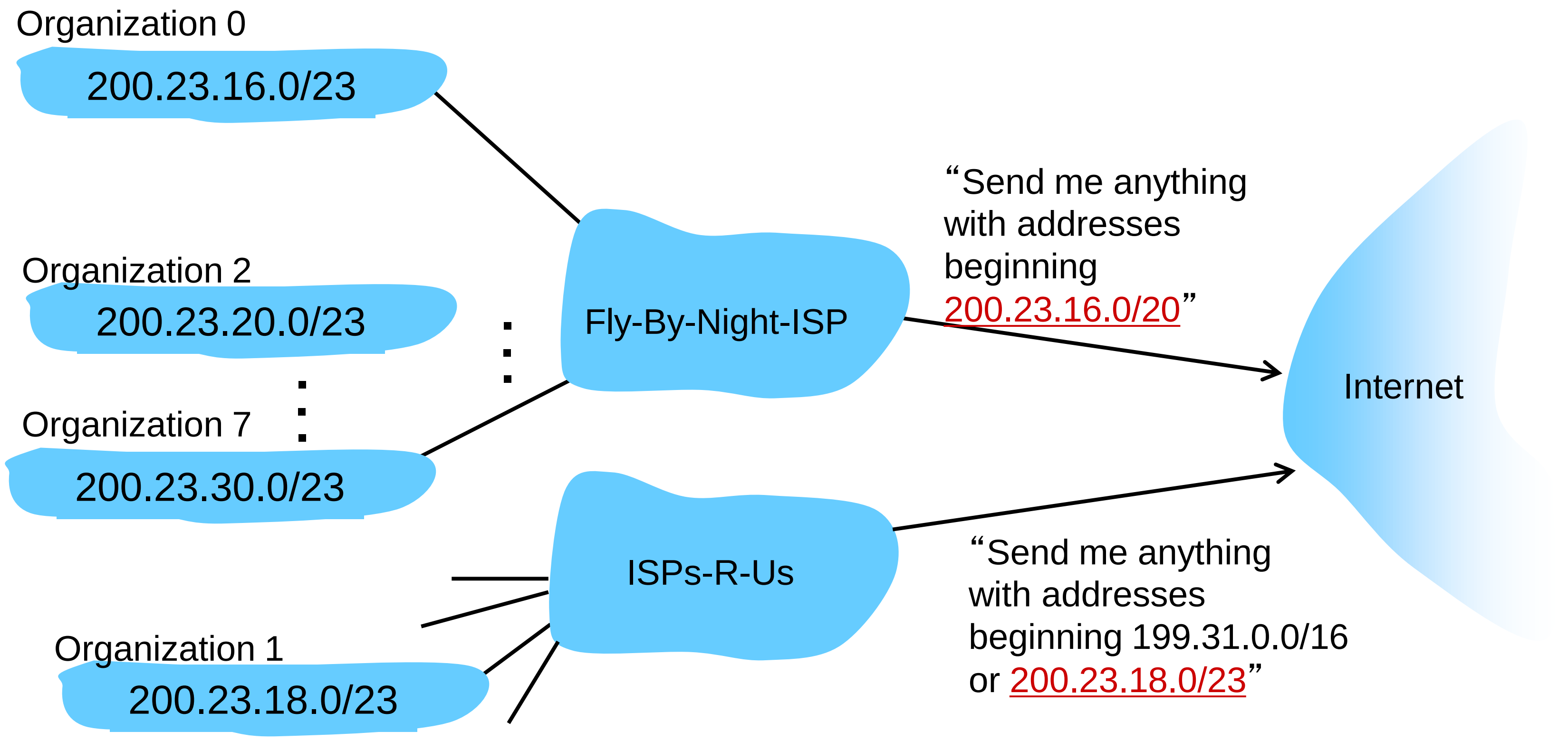


"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

Hierarchical Addressing: more specific routes

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



“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23”

Forwarding: Longest prefix matching

Prefix Match	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?**

IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

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

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What’s inside a router

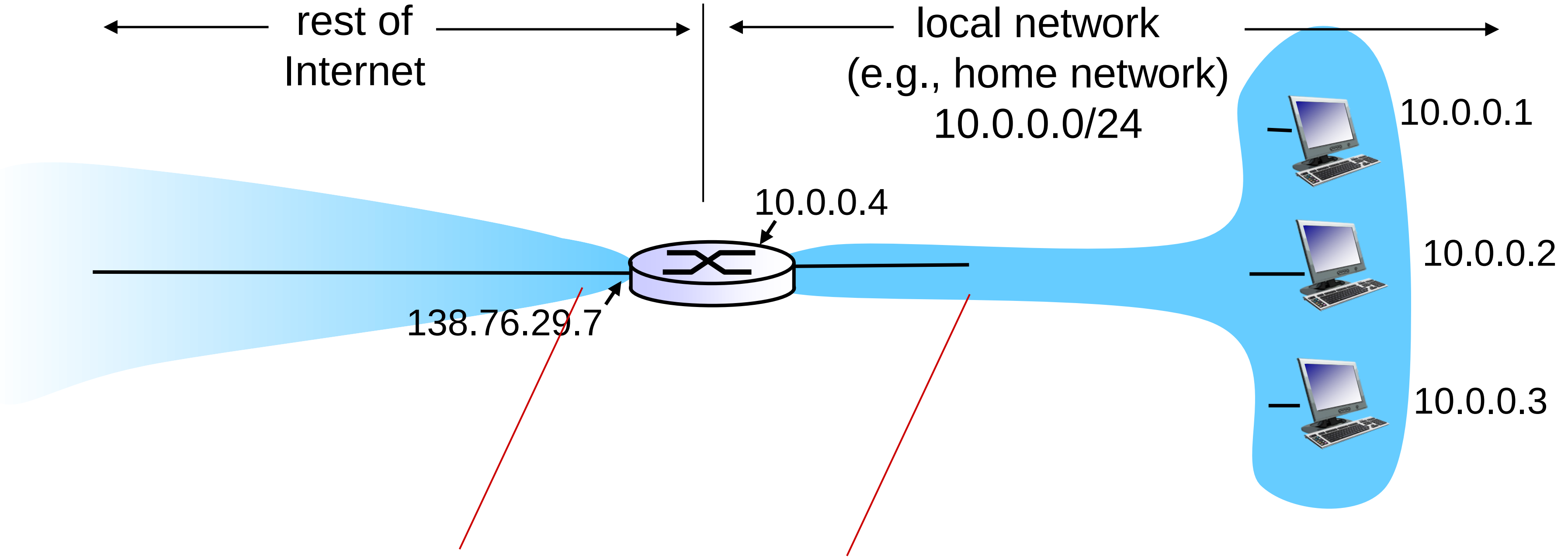
4.3 IP: Internet Protocol

- 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

NAT: network address translation



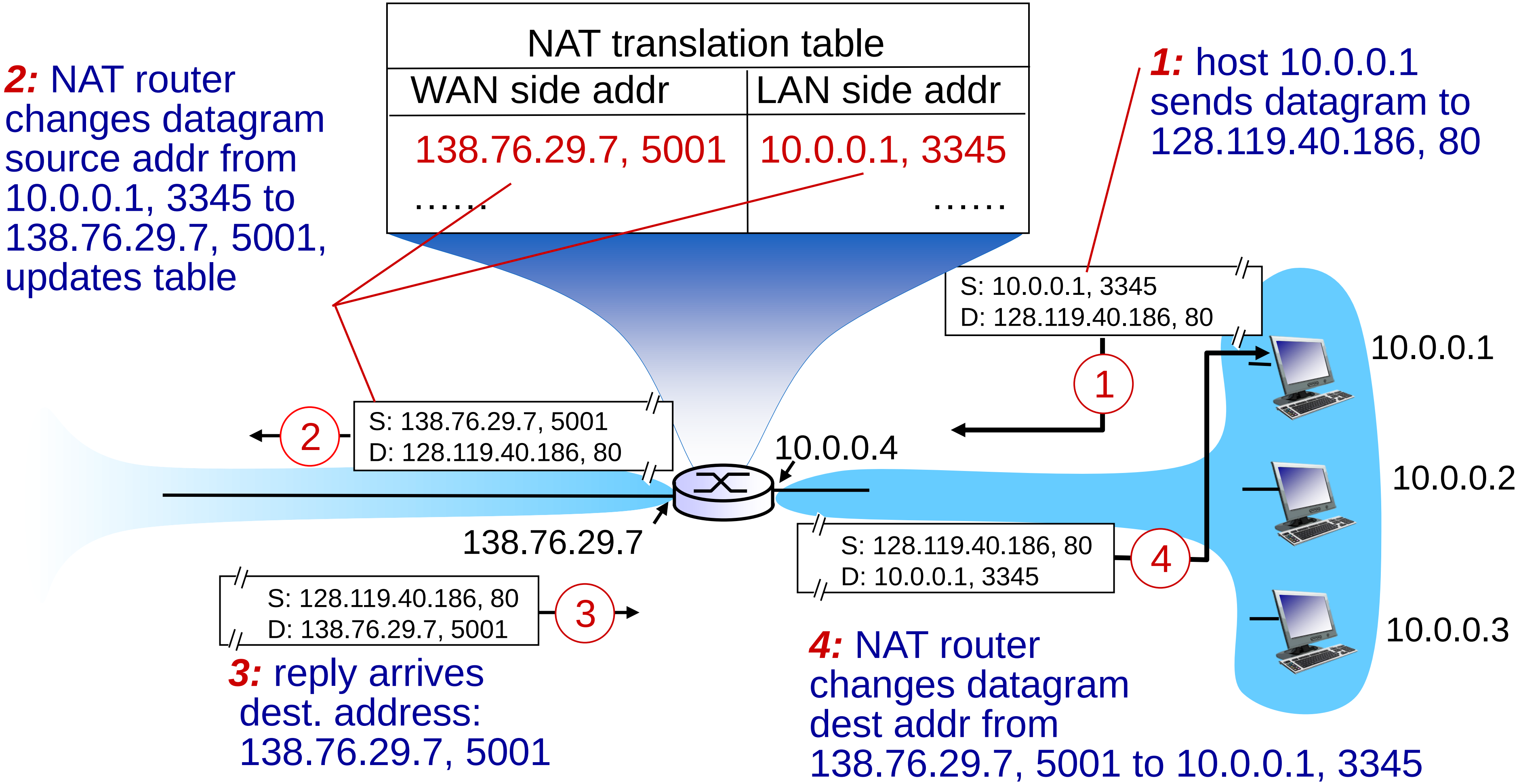
all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside word is concerned:
  - no need to be allocated range of addresses from ISP: just one IP address is used 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 and invisible by outside world (a security plus)

NAT: network address translation



NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

S: 10.0.0.1, 3345
D: 128.119.40.186, 80

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

S: 138.76.29.7, 5001
D: 128.119.40.186, 80

3: reply arrives dest. address: 138.76.29.7, 5001

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

S: 128.119.40.186, 80
D: 10.0.0.1, 3345

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

NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single WAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

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. (actually, all already allocated)
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

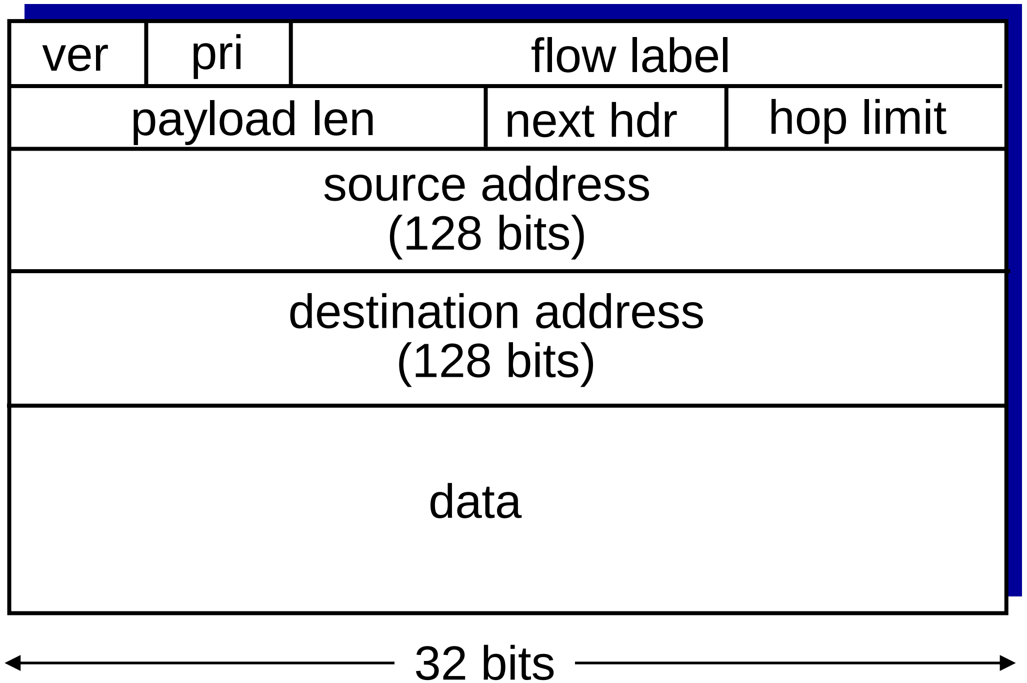
- fixed-length 40 byte header
- no fragmentation allowed

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



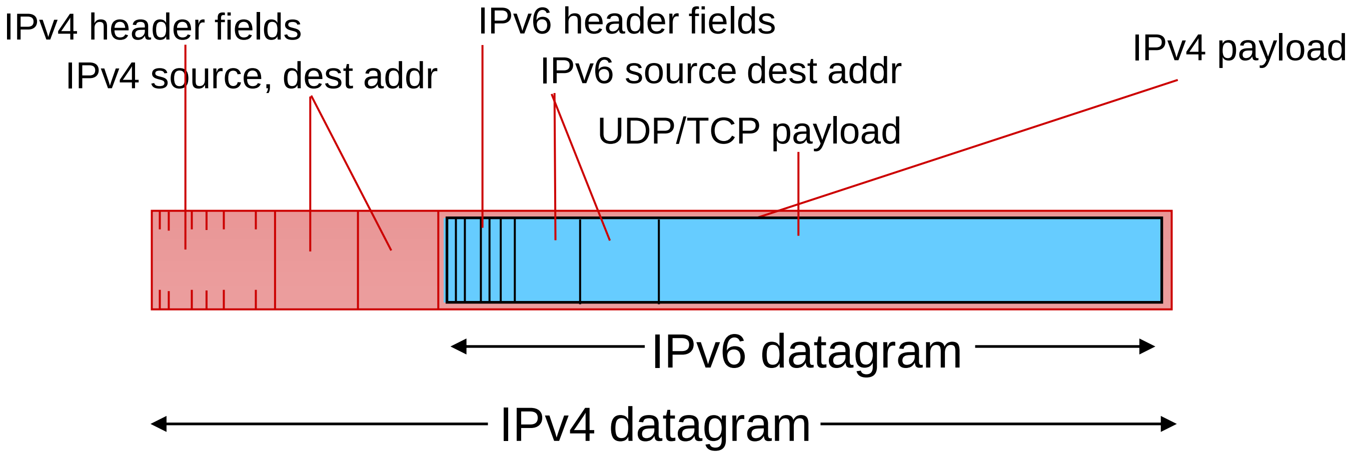
ver
pri
flow label
payload len
next hdr
hop limit
source address
(128 bits)
destination address
(128 bits)
data
32 bits

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

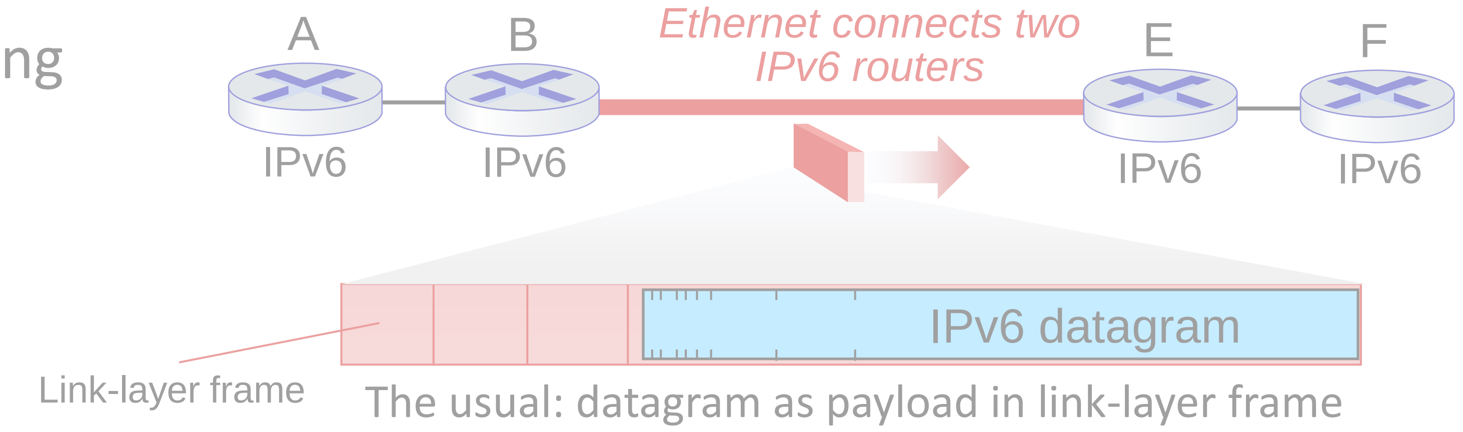
Transition from IPv4 to IPv6

- 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



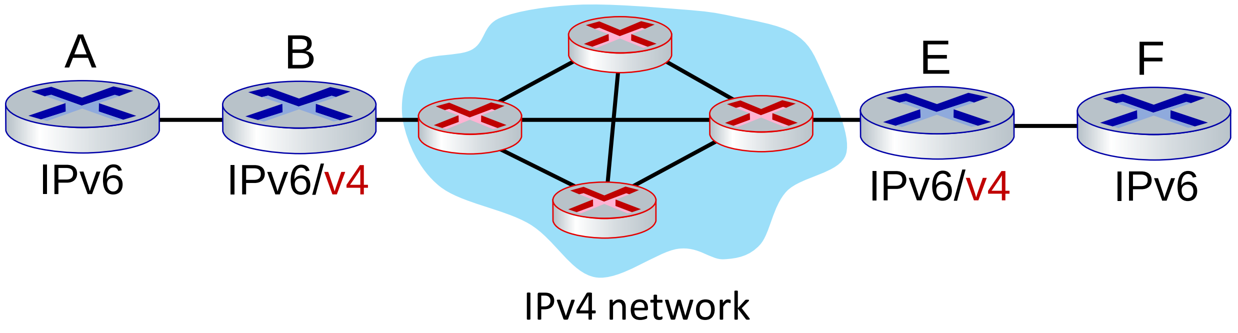
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

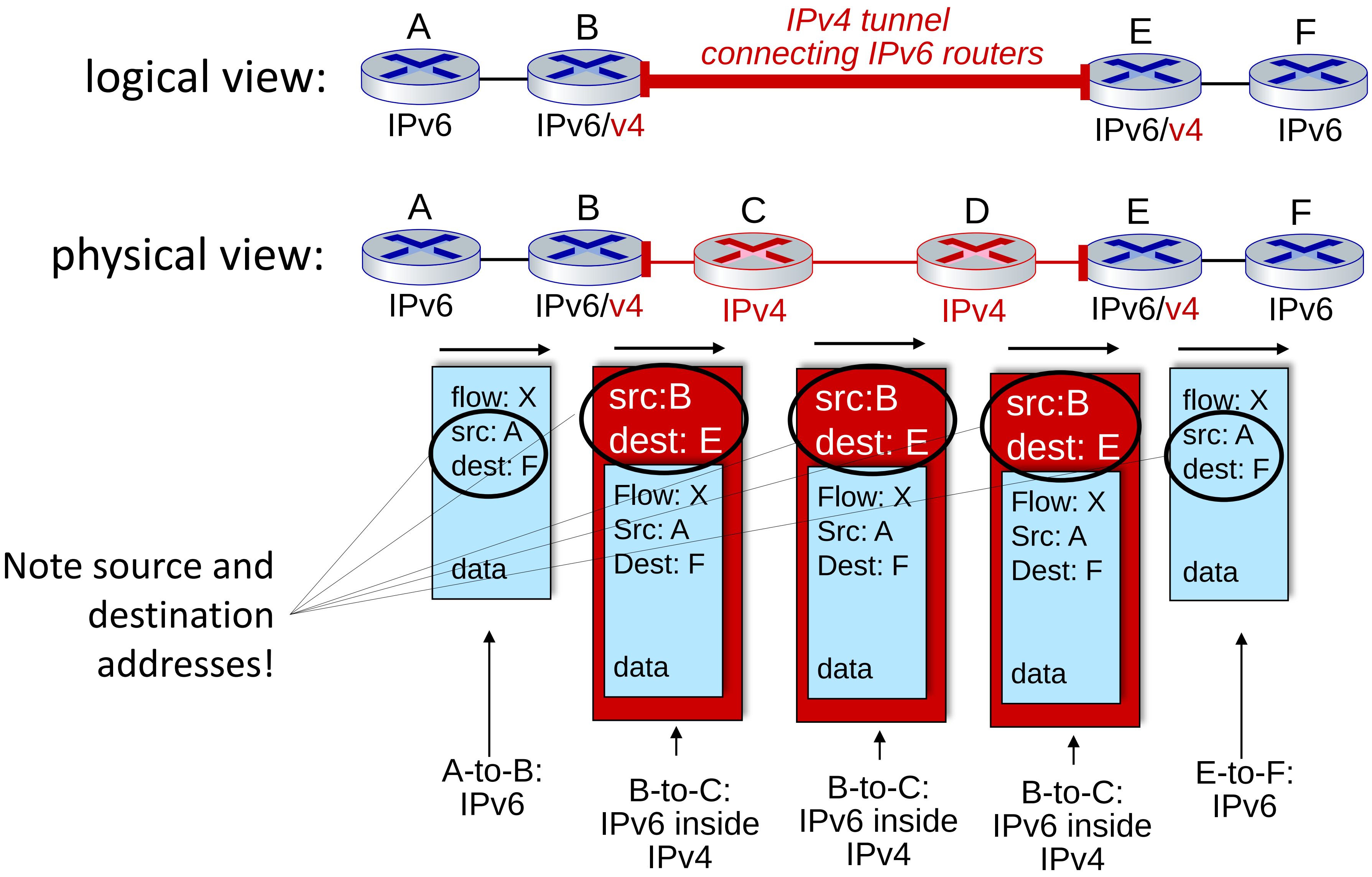


The usual: datagram as payload in link-layer frame

IPv4 network connecting two IPv6 routers



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
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

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

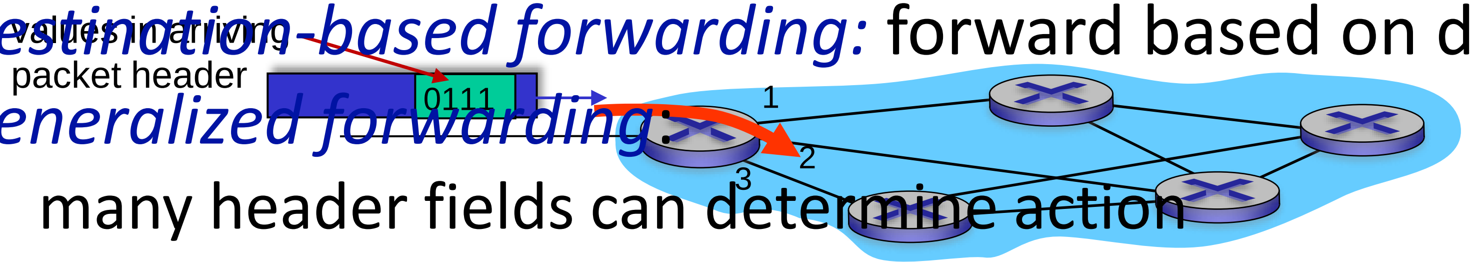
4.4 Generalized Forward and SDN

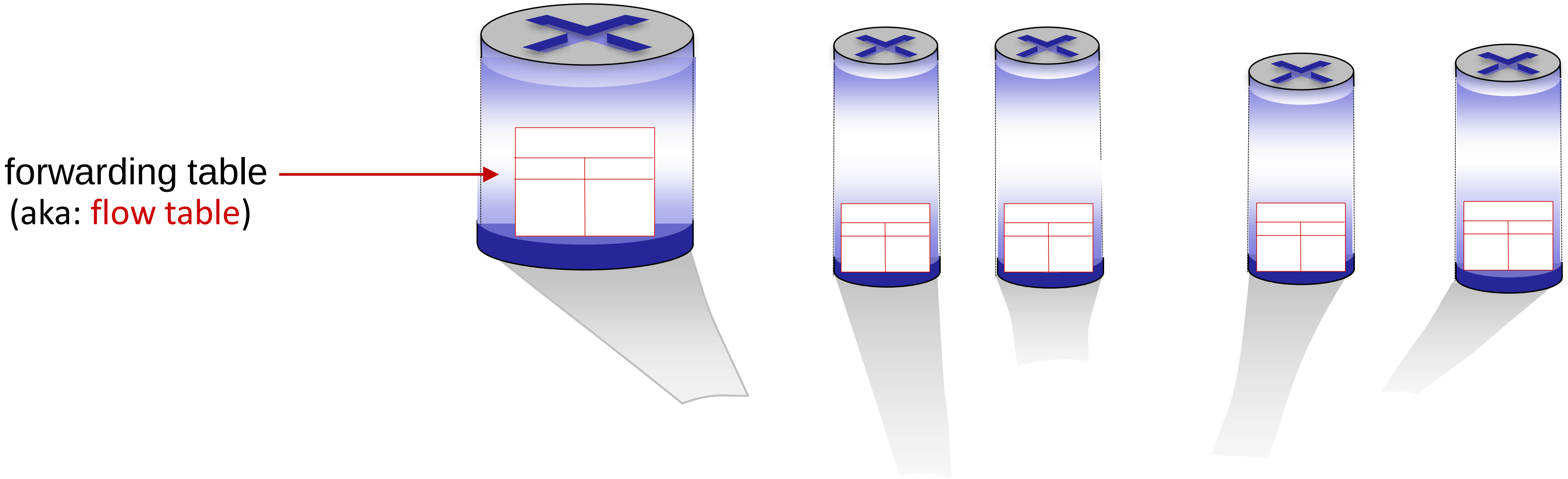
- match
- action
- OpenFlow examples of match-plus-action in action

Generalized forwarding: match plus action

Review: each router contains a forwarding table (aka: flow table)

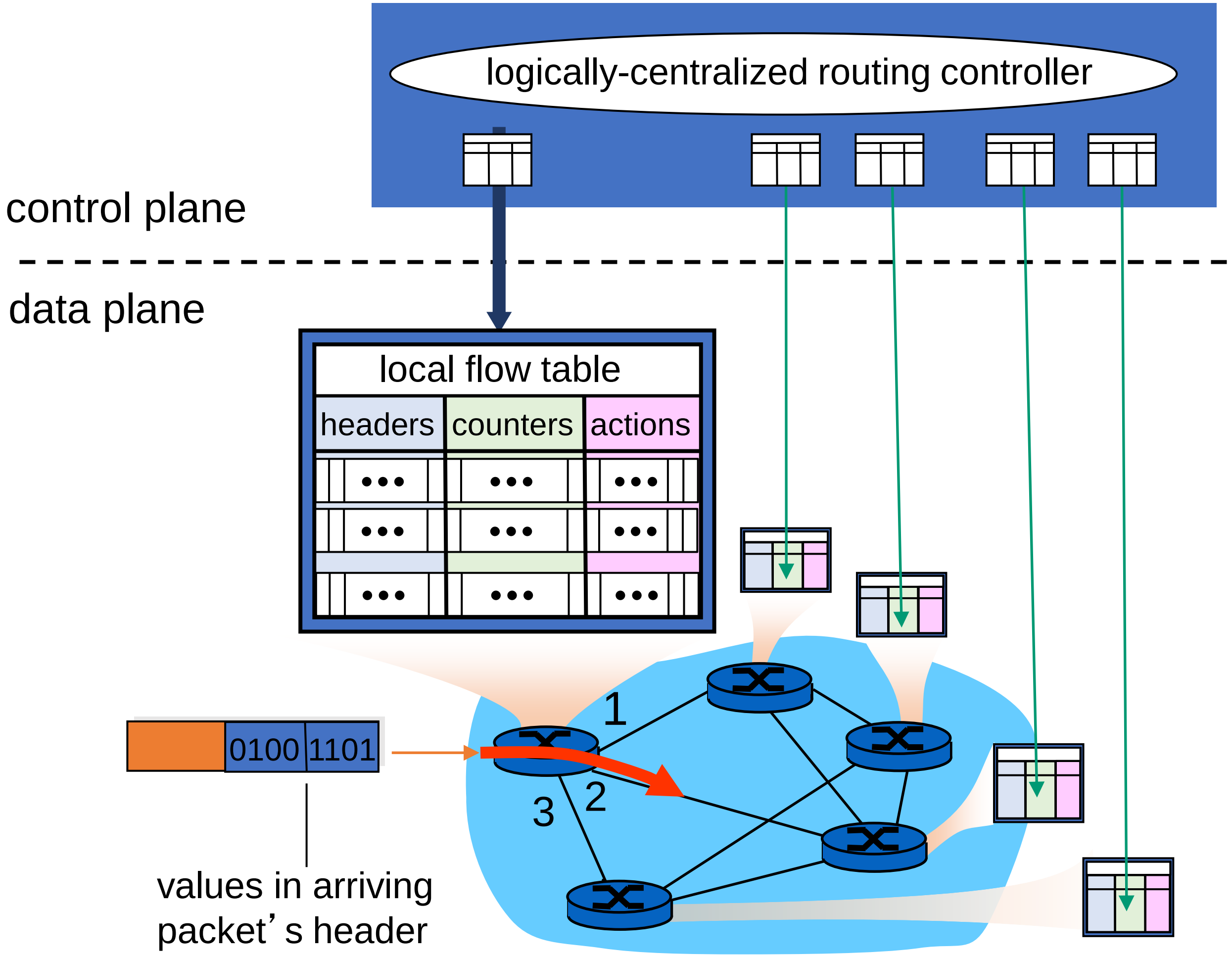
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet





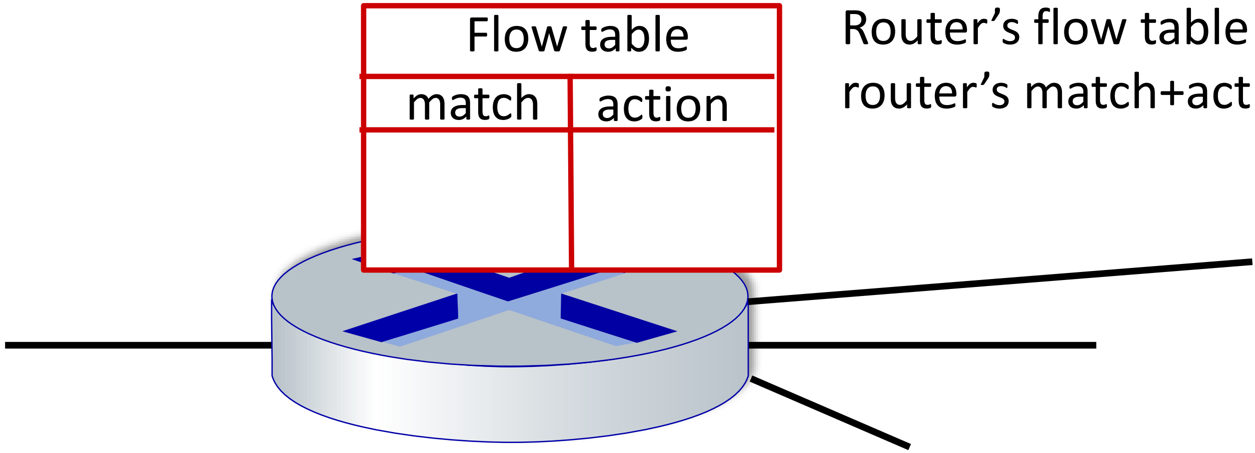
Generalized Forwarding and SDN

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
  - *Counters:* #bytes and #packets, and time since the table entry was last updated.

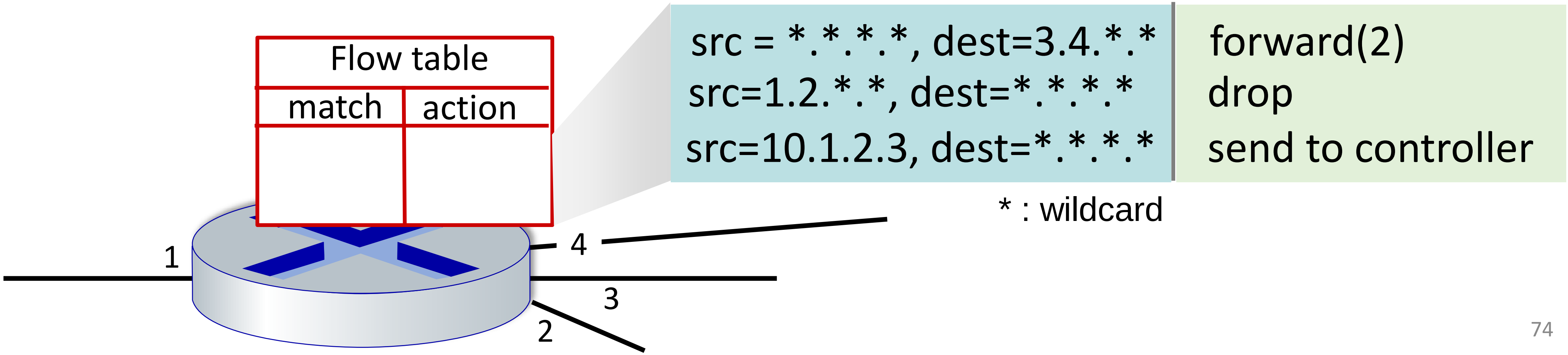


Flow table	
match	action

Router’s flow table define
router’s match+action rules

OpenFlow data plane abstraction

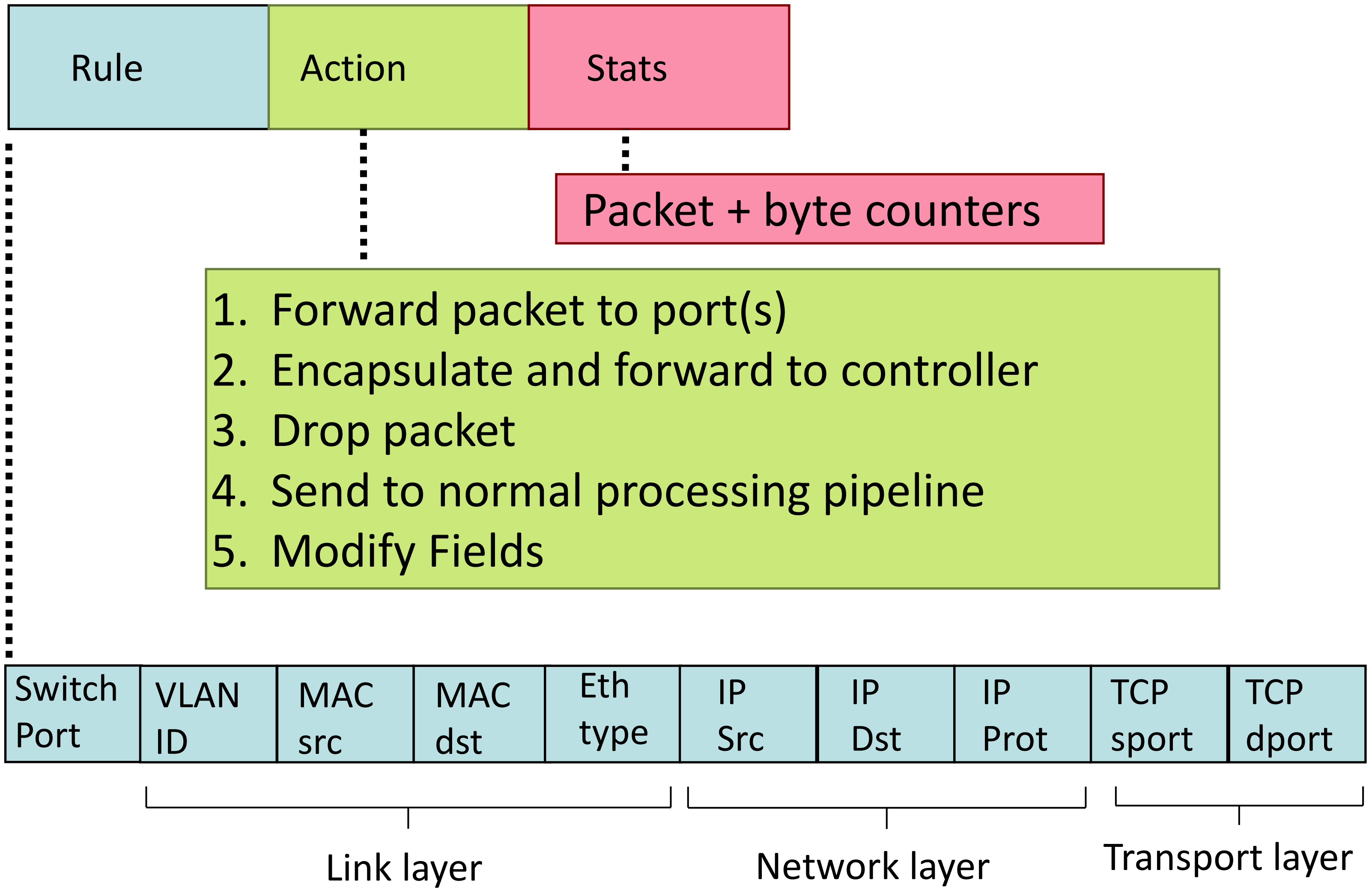
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward modify matched packet or send matched packet to controller
  - *Counters:* #bytes and #packets, and time since the table entry was last updated.



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

* : wildcard

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 3

OpenFlow abstraction

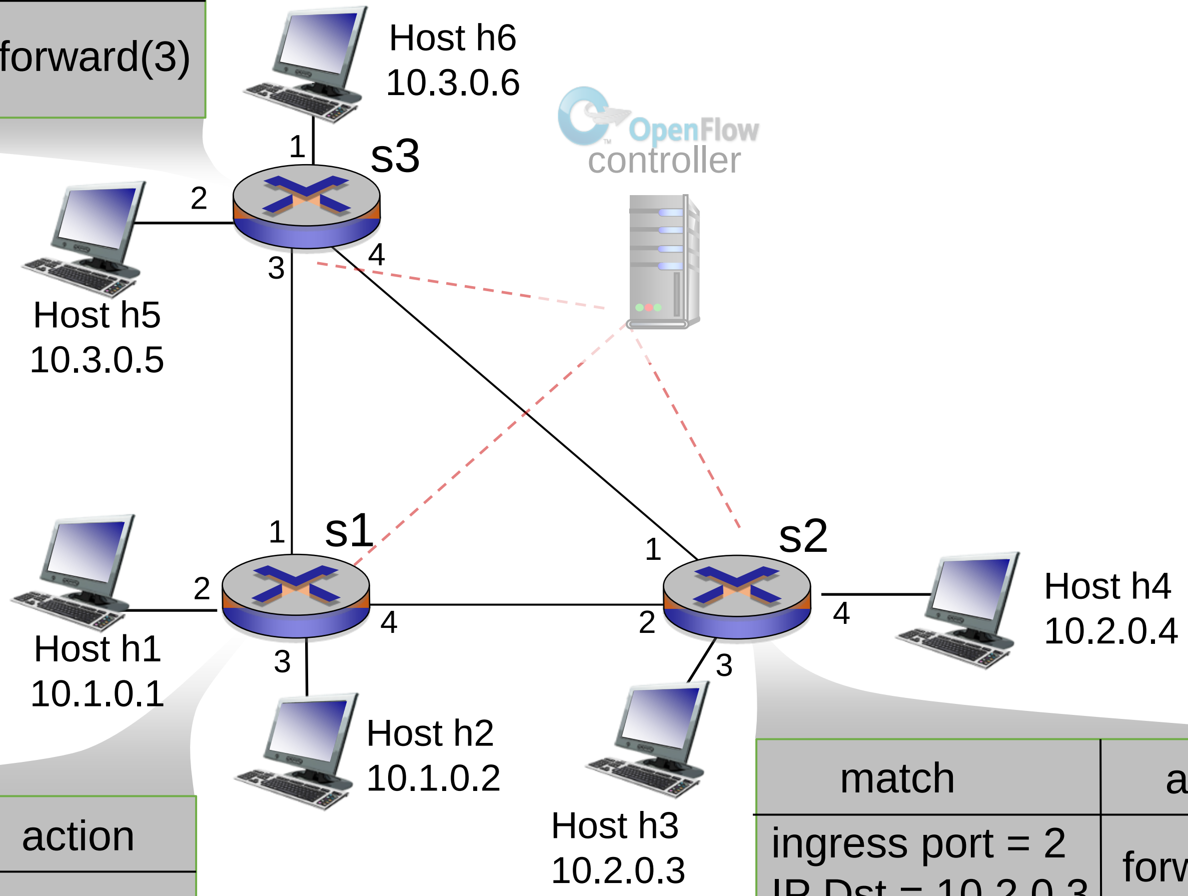
- *match+action:* unifies different kinds of devices

- 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

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

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

Answer: by the control plane (next chapter)