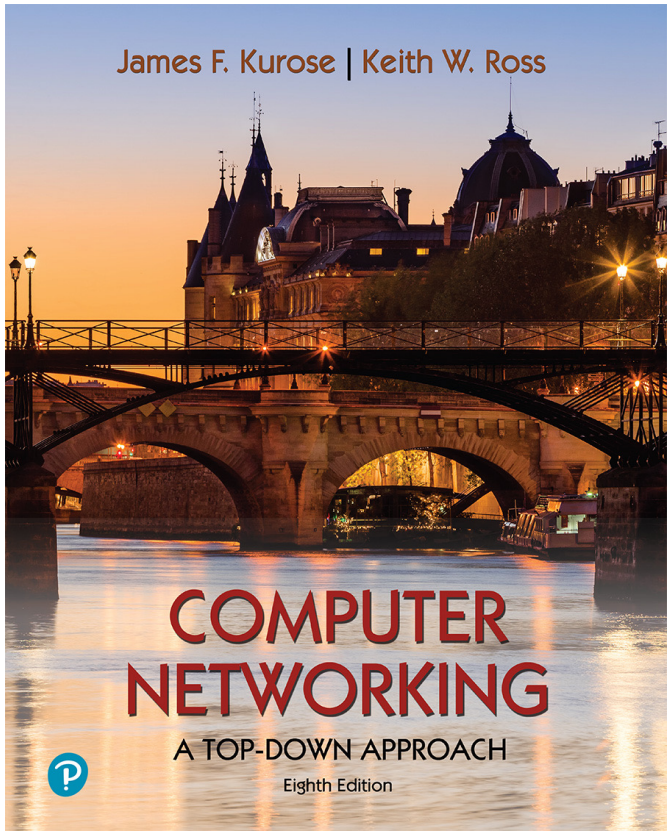
Chapter 4
Network Layer: The Data Plane



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
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

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
  - addressing
  - generalized forwarding
- instantiation, implementation in the Internet
  - IP protocol
  - NAT
- Generalized Forwarding and SDN

The Network Layer

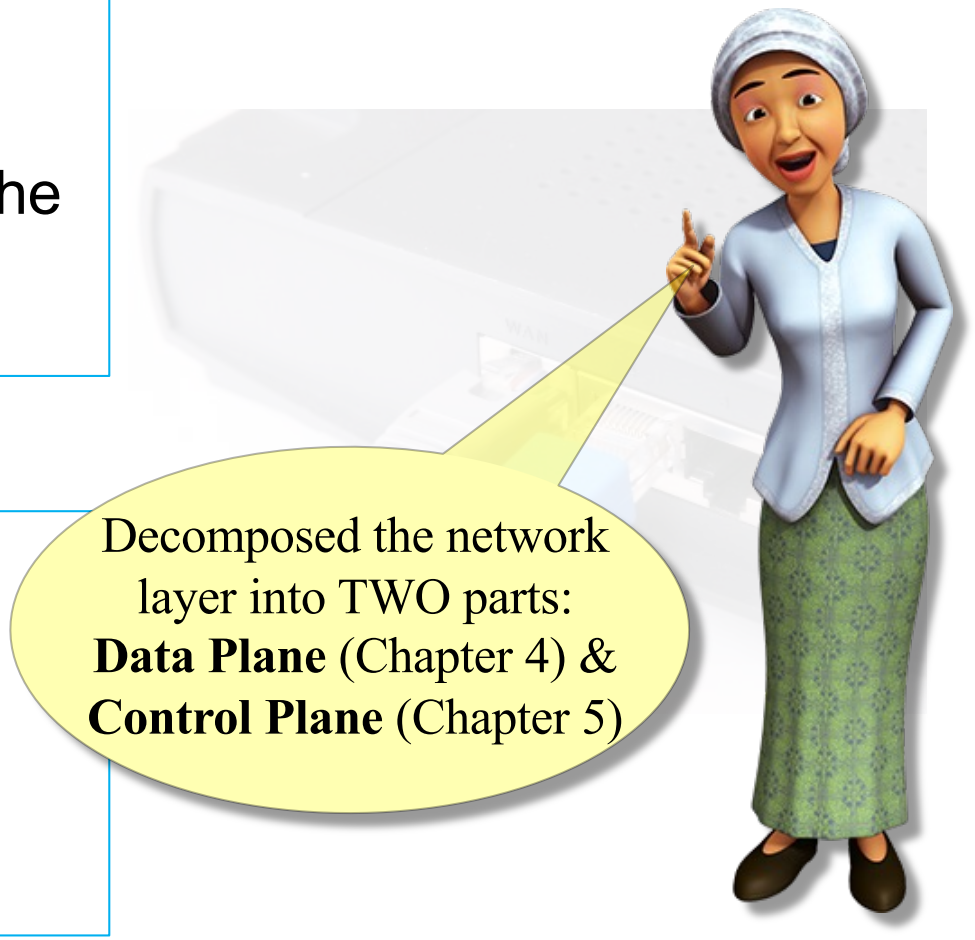
Overview

Transport layer

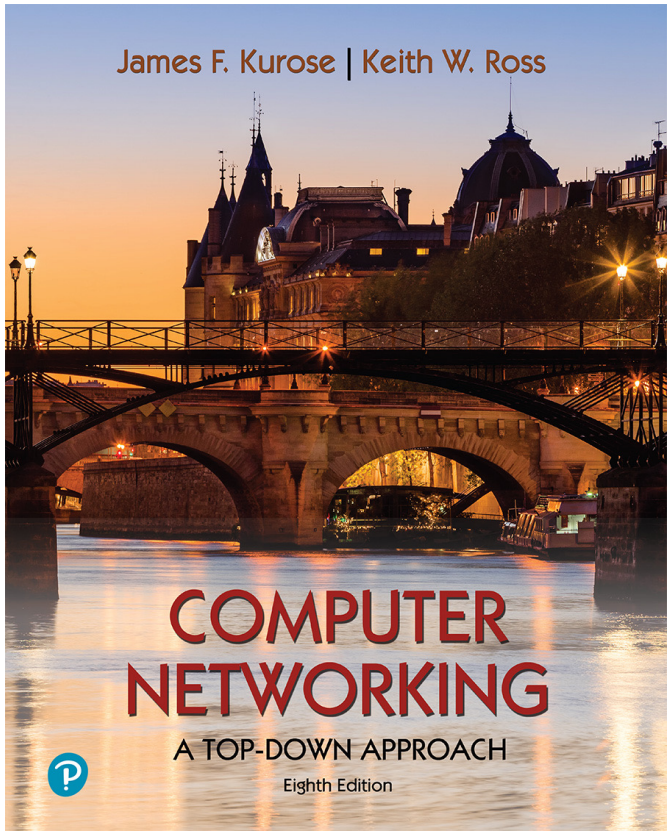
- Provides various forms of process-to-process communication by relying on the network layer's host-to-host communication service.

Network layer

- Can provide its host-to-host communication service.
- the most complex layer in the protocol stack.



Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

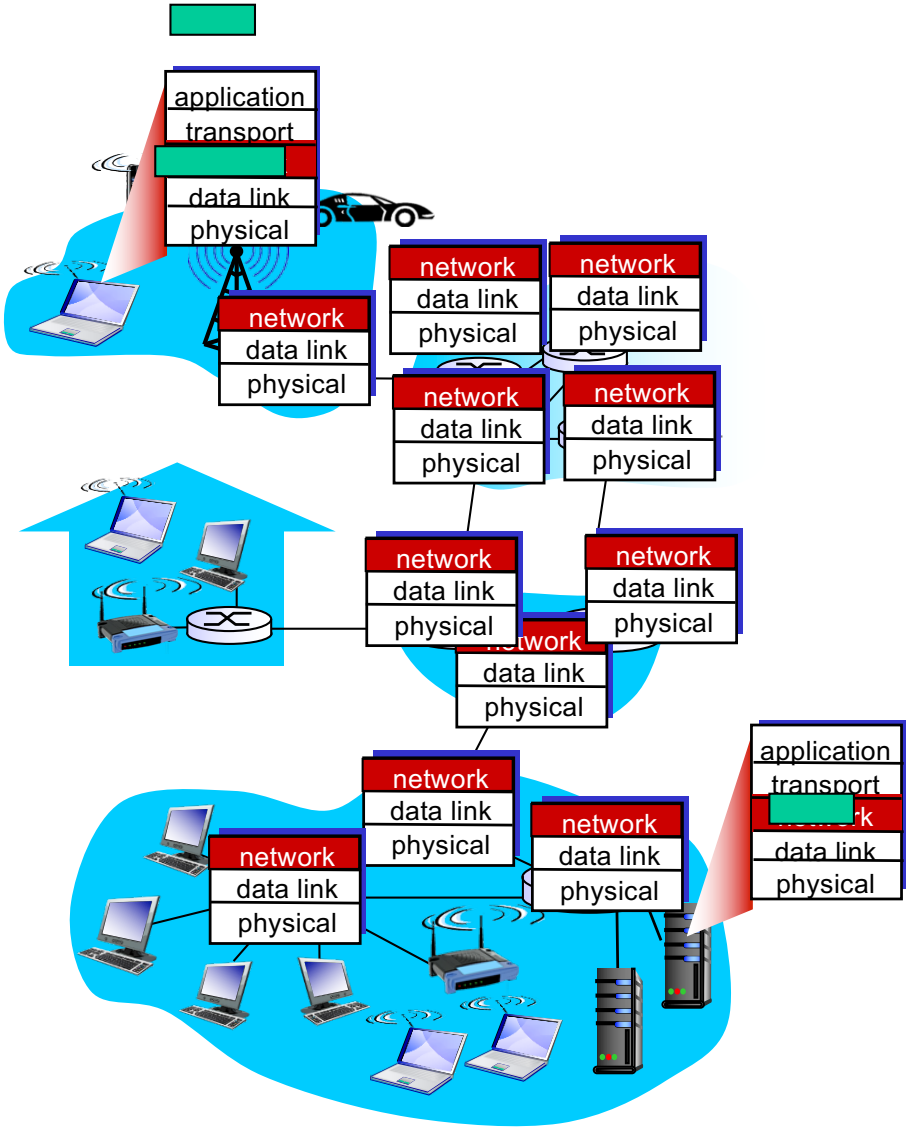
input ports, switching, output ports, buffer management, scheduling

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

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

Network layer services and protocols

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



Two key network-layer functions

forwarding

routing

Two Functions

1st: Forwarding

2nd: Routing

Move packets from router's input to appropriate router output.

Analogy:

Forwarding: process of getting through single interchange.

Determine route taken by packets from source to destination.

* routing algorithms

Analogy:

Routing: Process of planning trip from source to destination.

Two key network-layer functions

Interplay between forwarding and routing

Forwarding:

Determines local forwarding for this router.

Routing:

Determines end-to-end path through network

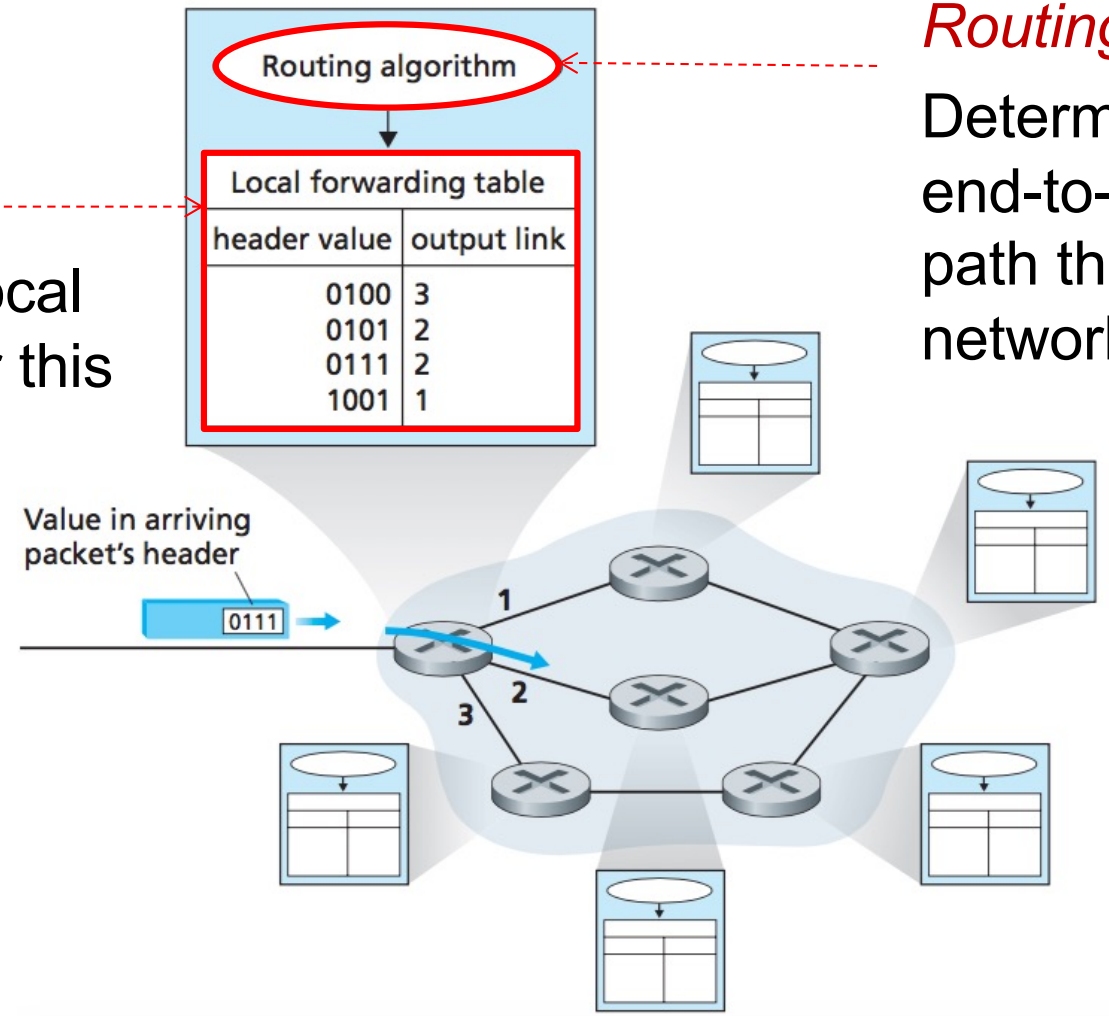
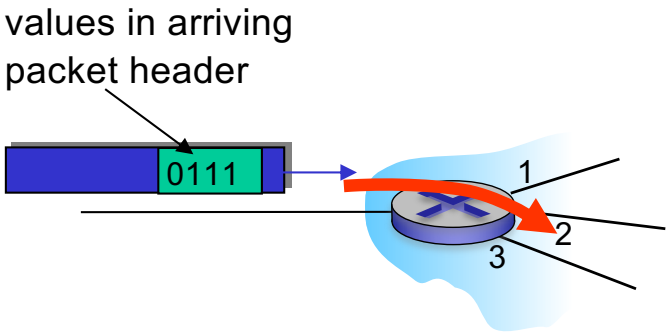


Figure: Routing algorithm determine values in forwarding tables

Network layer: data plane, control plane

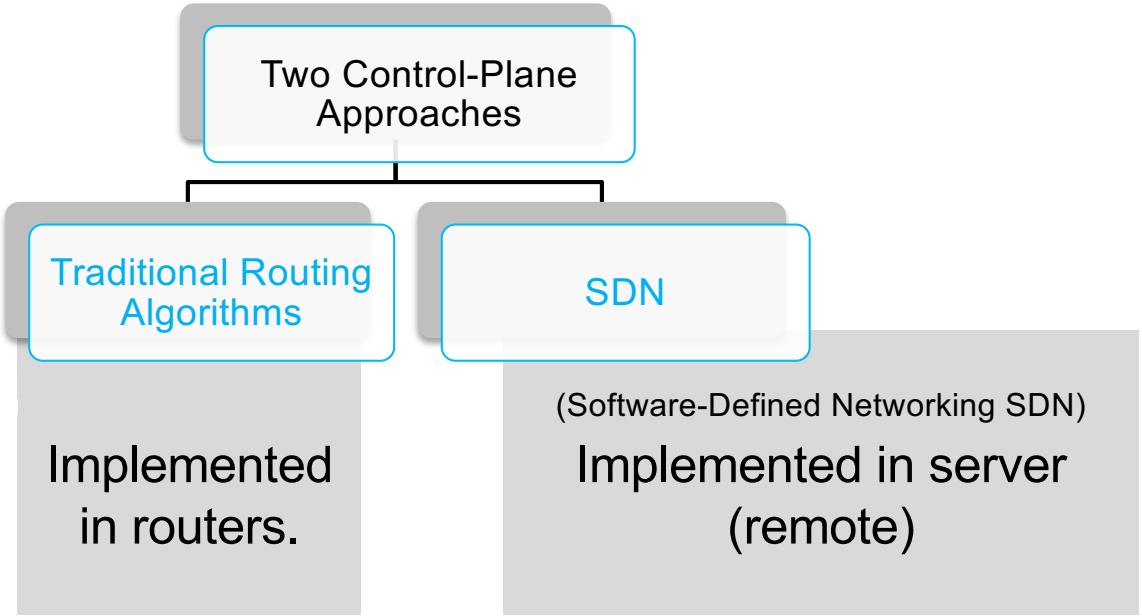
Data plane

- **Forwarding** function: Determines how datagram arriving on router input port is forwarded to router output port.
- Local, per-router function.



Control plane

- **Routing** function: Determines how datagram is routed among routers along end-end path from source host to destination host.
- Network-wide logic.



routing algorithm

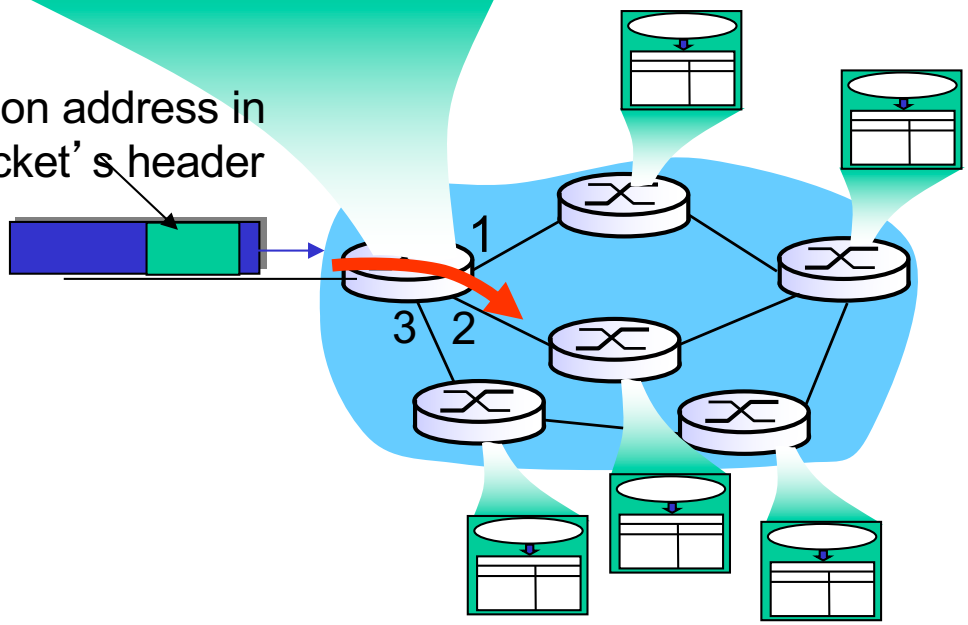
local forwarding table

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

4 billion IP addresses, so rather than list individual destination address, list *range* of addresses (aggregate table entries)

IPv4: 2^{32} = 4,294,967,296

IP destination address in arriving packet's header



Forwarding and Routing

(a) Per-router control plane

The Traditional Approach

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

Routing: Determines end-to-end path through network.

Forwarding: Determines local forwarding of this router.

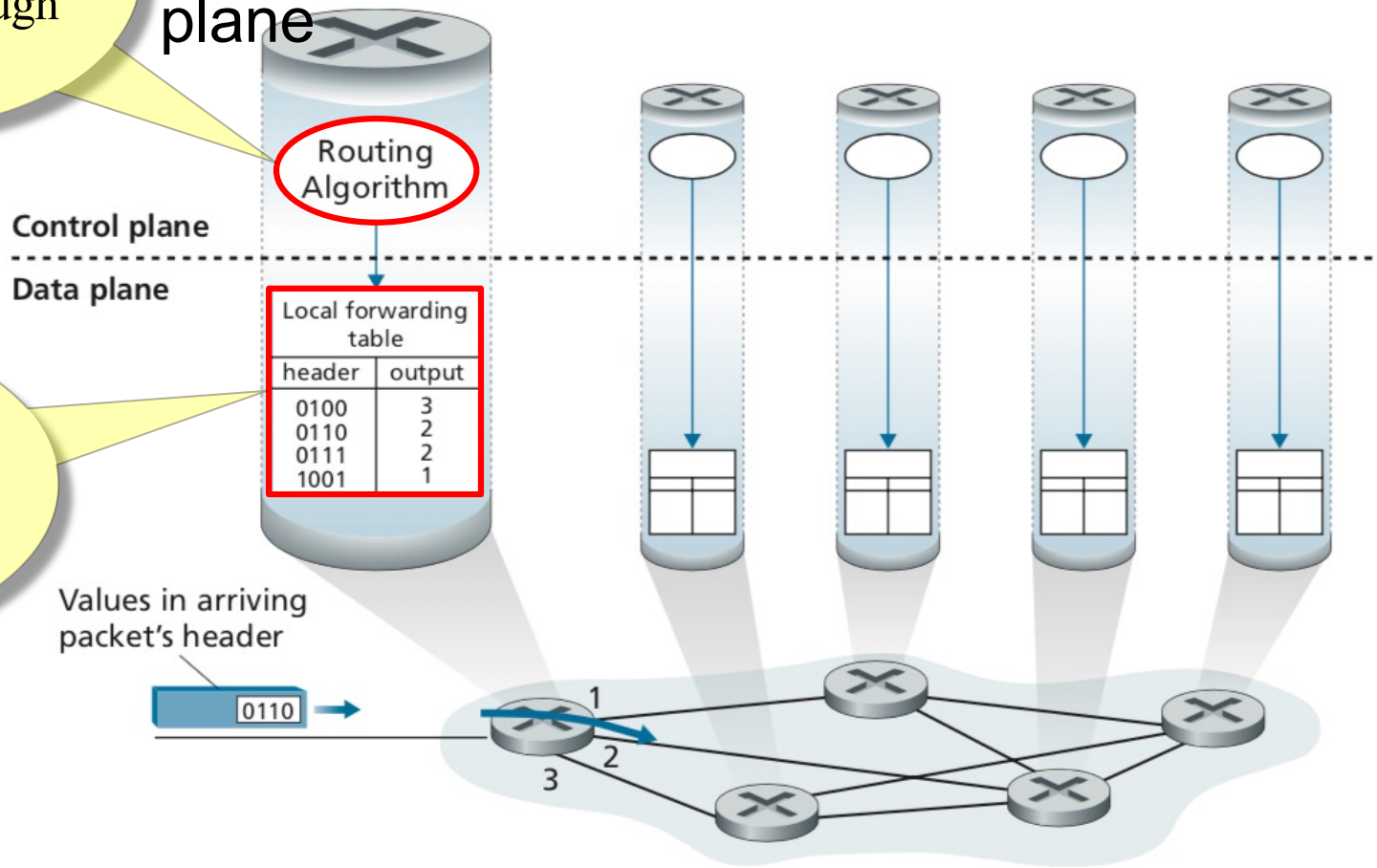


Figure: Routing algorithm determine values in forwarding tables

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 307)

Forwarding and Routing

(b) The SDN Approach

Software-Defined Networking (SDN)

Routing: Determines end-to-end path through network.

Remote controller computes, installs forwarding tables in routers

Forwarding: Determines local forwarding of this router.

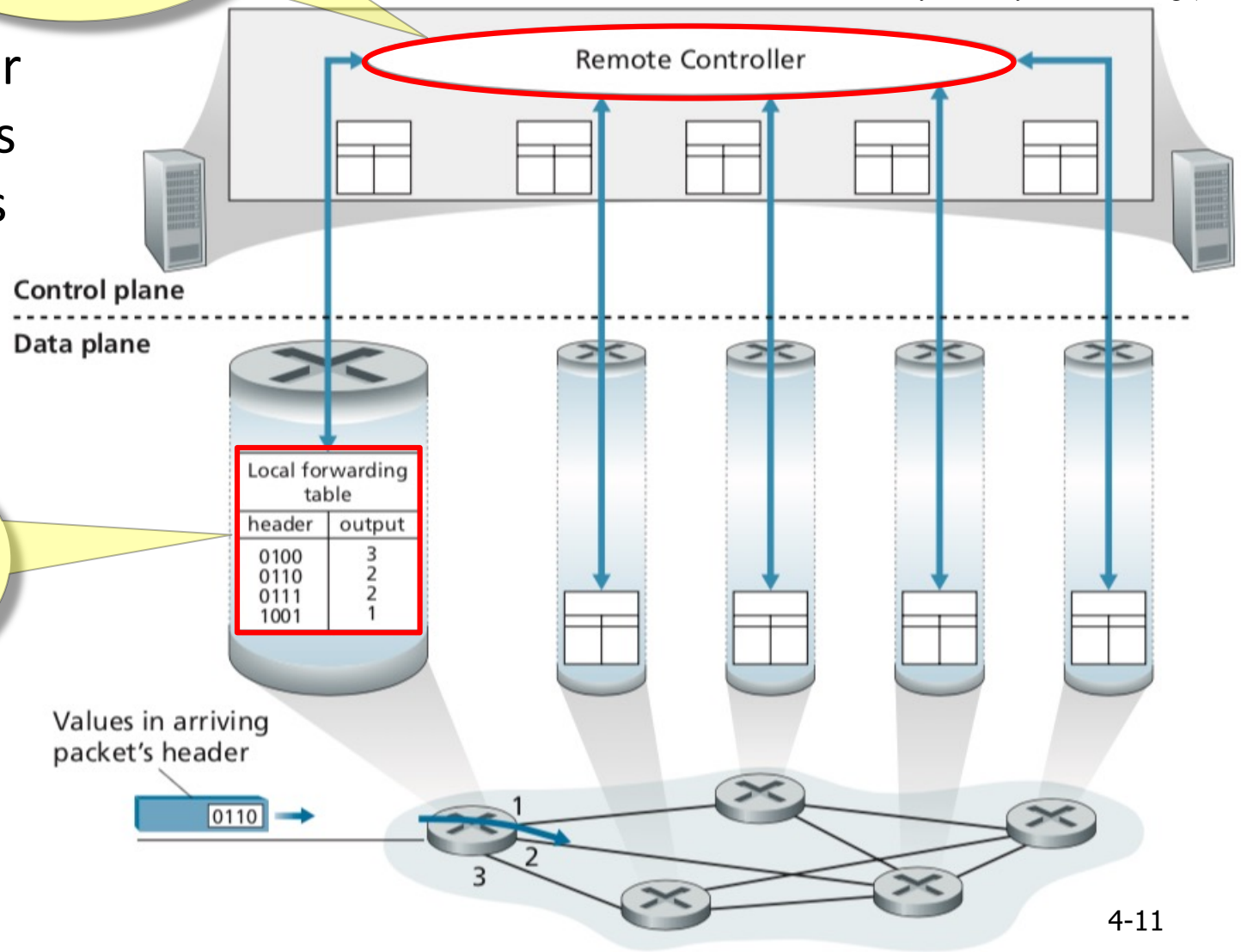


Figure: A remote controller determines and distributes values in forwarding tables

Network service model

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

example services for individual datagrams:

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

example services for a flow of datagrams:

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

Network-layer service model

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

Internet "best effort" service model

No guarantees on:

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Network layer service models:

Network Architecture	Service Model	Quality of Service (QoS) 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

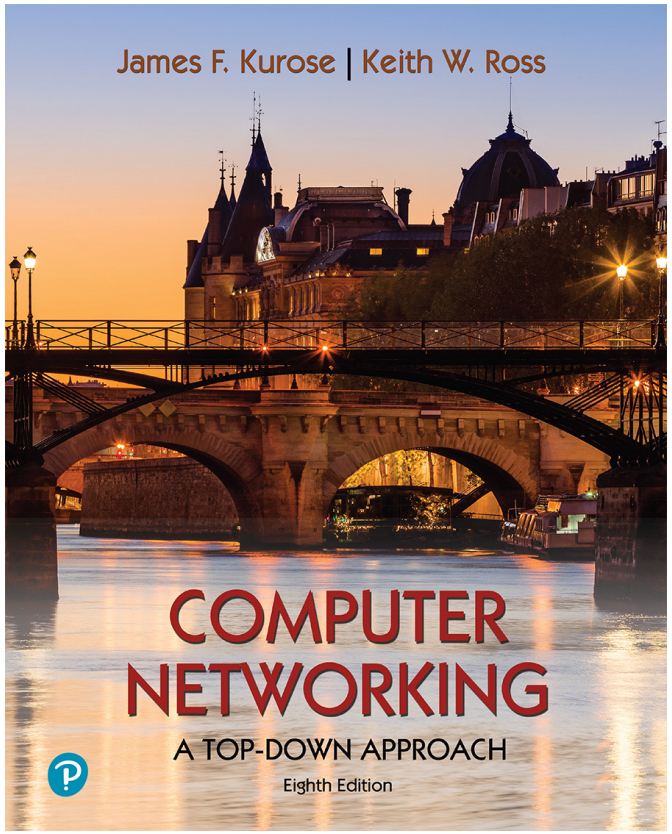
Asynchronous Transfer Mode (ATM) - Network Architecture
CBR: Constant Bit Rate, VBR=Variable Bit Rate, ABR=Available Bit Rate, UBR= Unspecified Bit Rate

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be "good enough" for "most of the time"
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients' networks, allow services to be provided from multiple locations
- congestion control of "elastic" services helps

It's hard to argue with success of best-effort service model

Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

input ports, switching, output ports

buffer management, scheduling

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

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

Router architecture overview

TWO key router functions:

- run *routing algorithms* / protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from a router's incoming links to the appropriate outgoing links at a router

- Four router components can be identified:

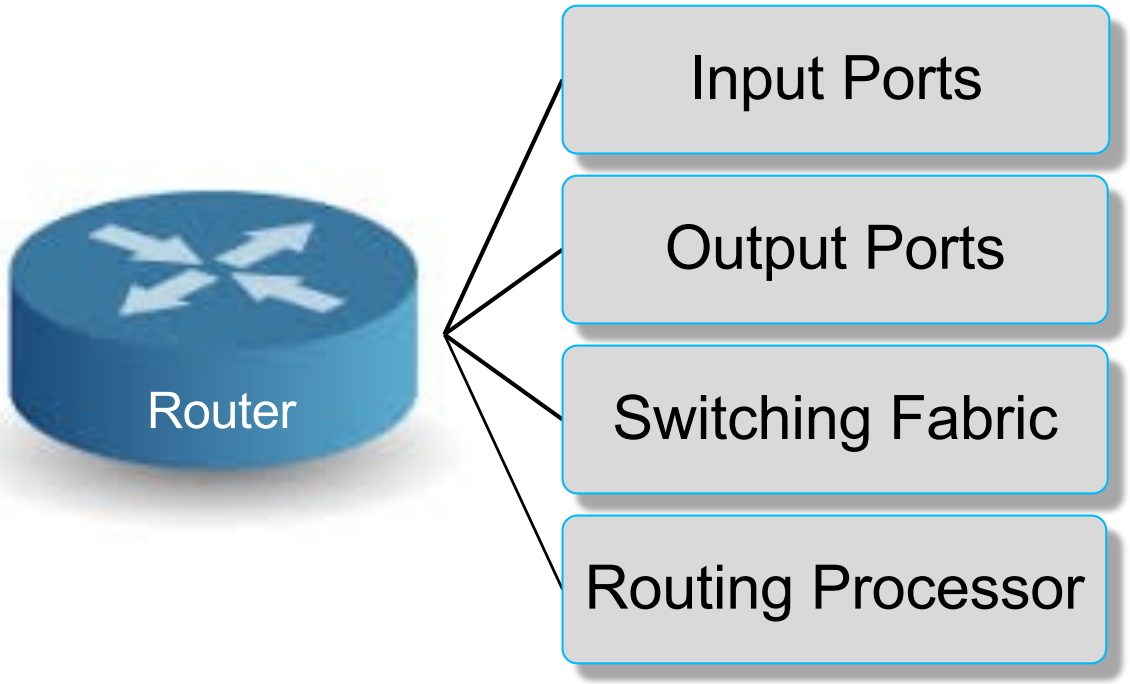
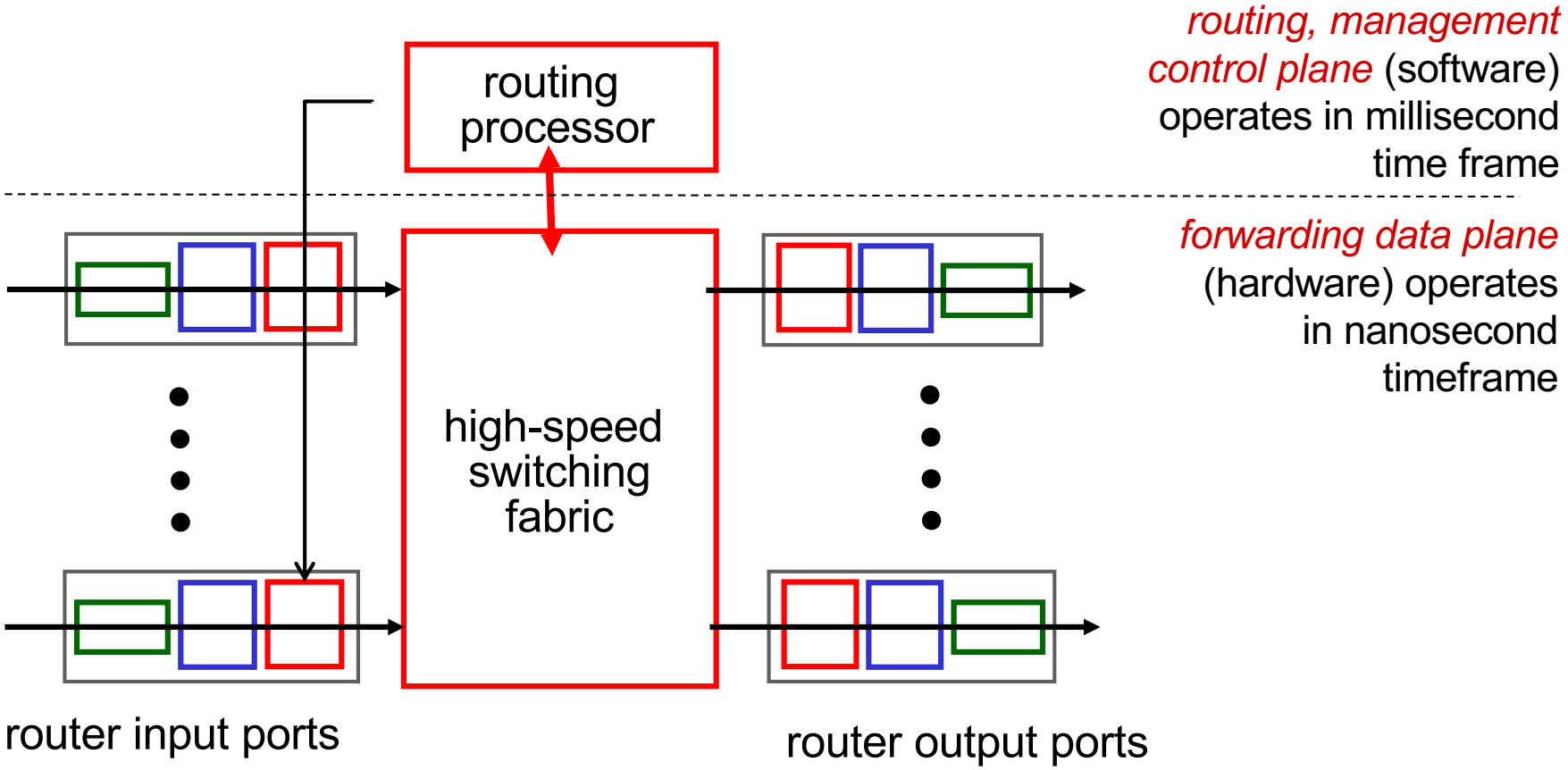


Figure: Four router components

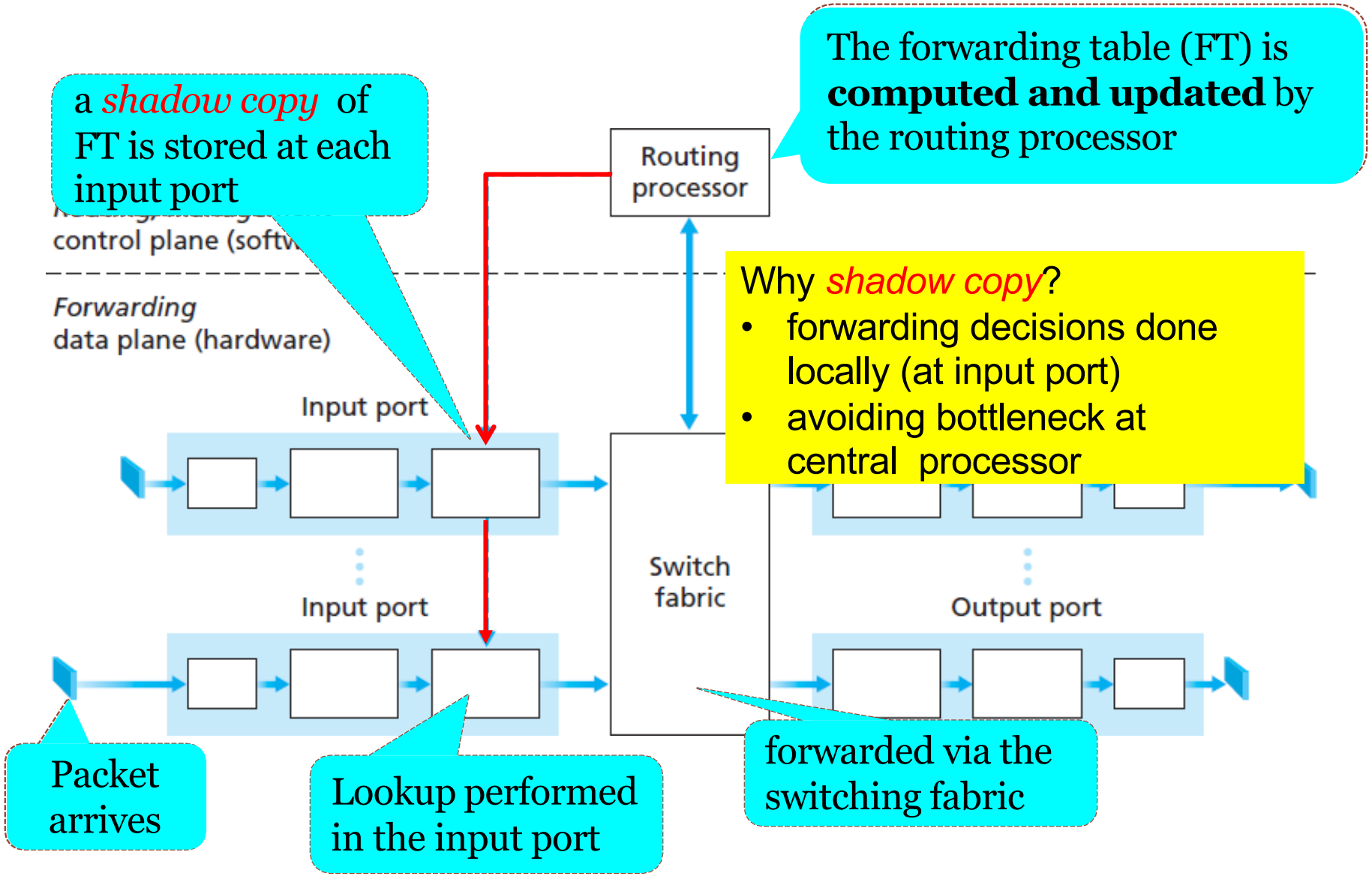
RIP (Routing Information Protocol)
OSPF (Open-Shortest Path First)
BGP (Border Gateway Protocol)

Router architecture overview

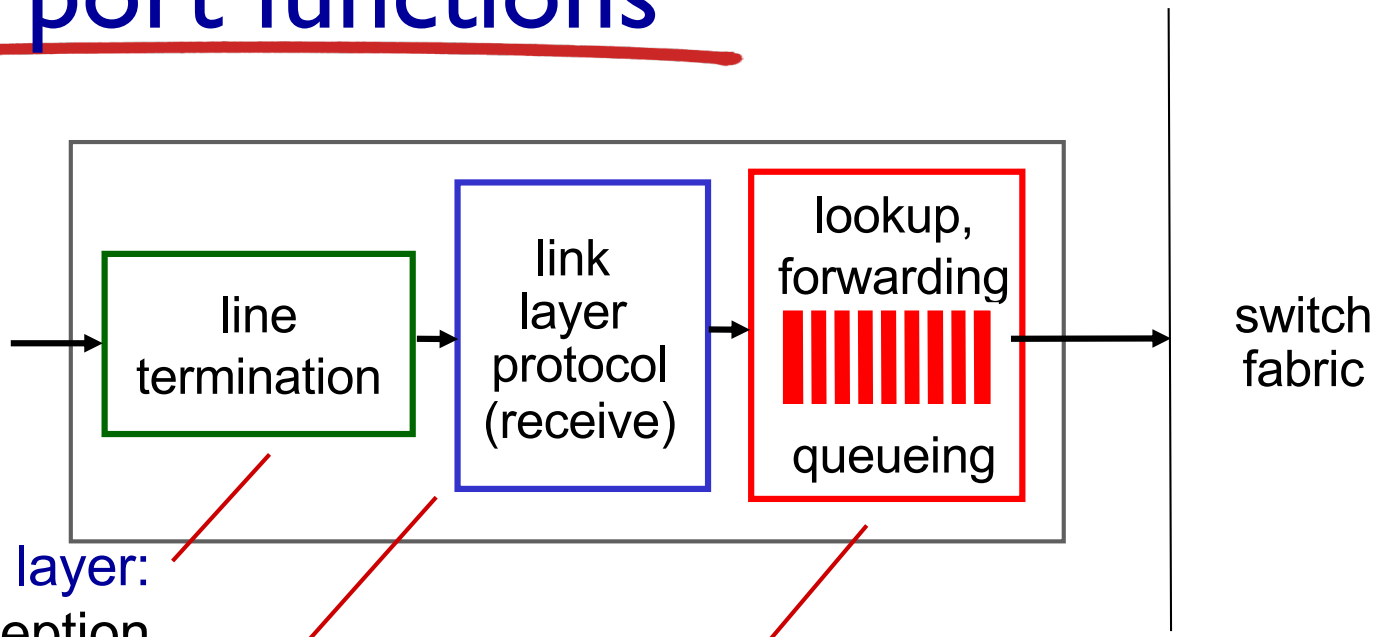
- high-level view of generic router architecture:



Router architecture overview



Input port functions



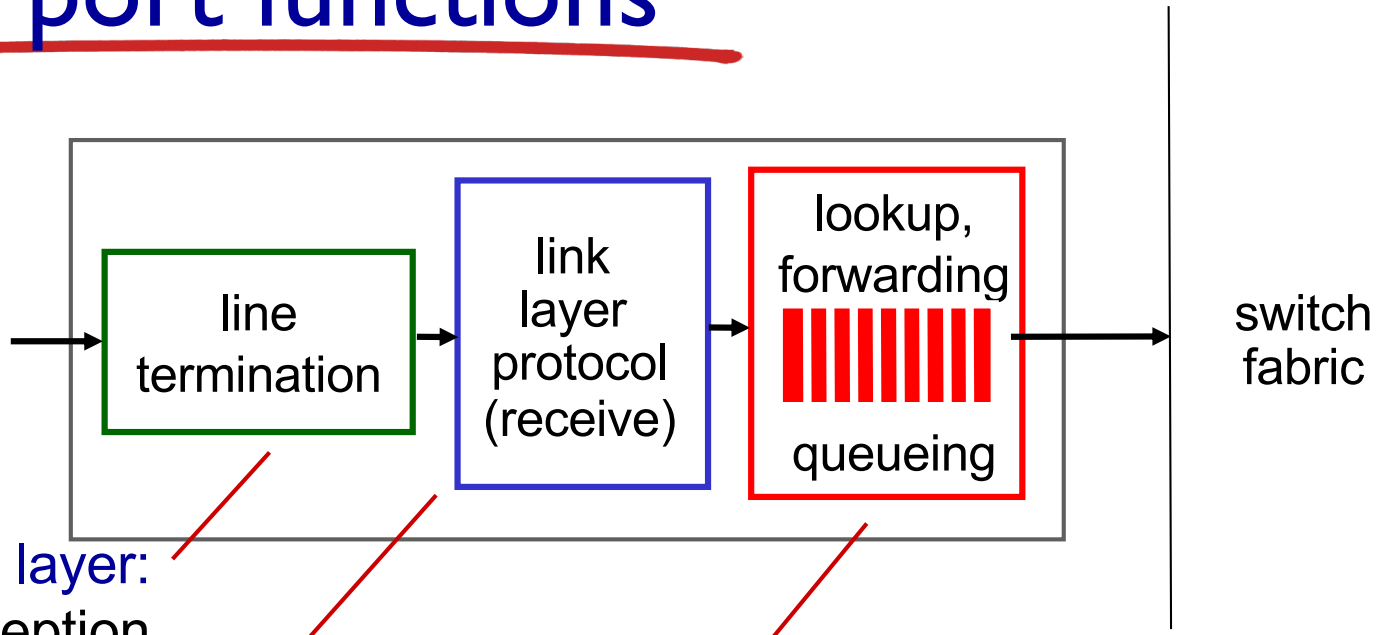
physical layer:
bit-level reception

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 into switch fabric

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 (SDN):* forward based on any set of header field values

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010000 00000100	0
11001000 00010111 00010000 00000101 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:

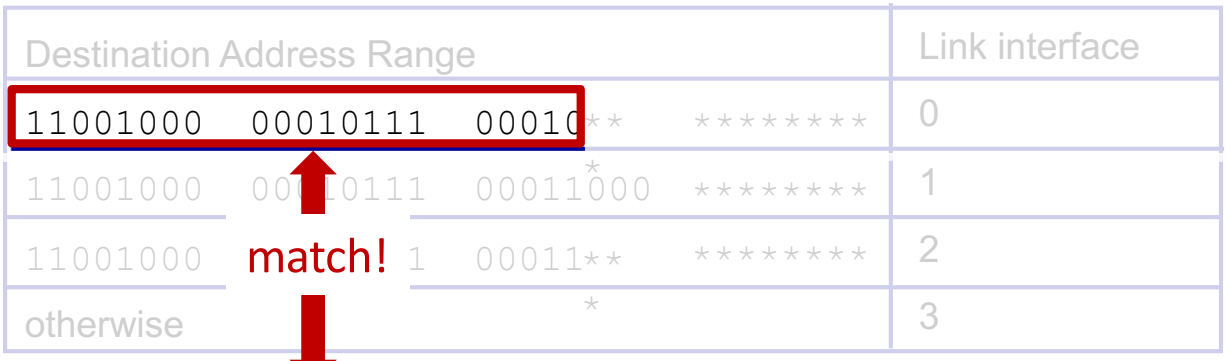
11001000 00010111 00010110 10100001 which interface?

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.



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 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 up ~1M routing table entries in TCAM

Switching Fabrics

- Transfer packet from **input link** to appropriate **output link**
- Switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate.
  - *N* inputs: switching rate = *N* times line rate desirable.

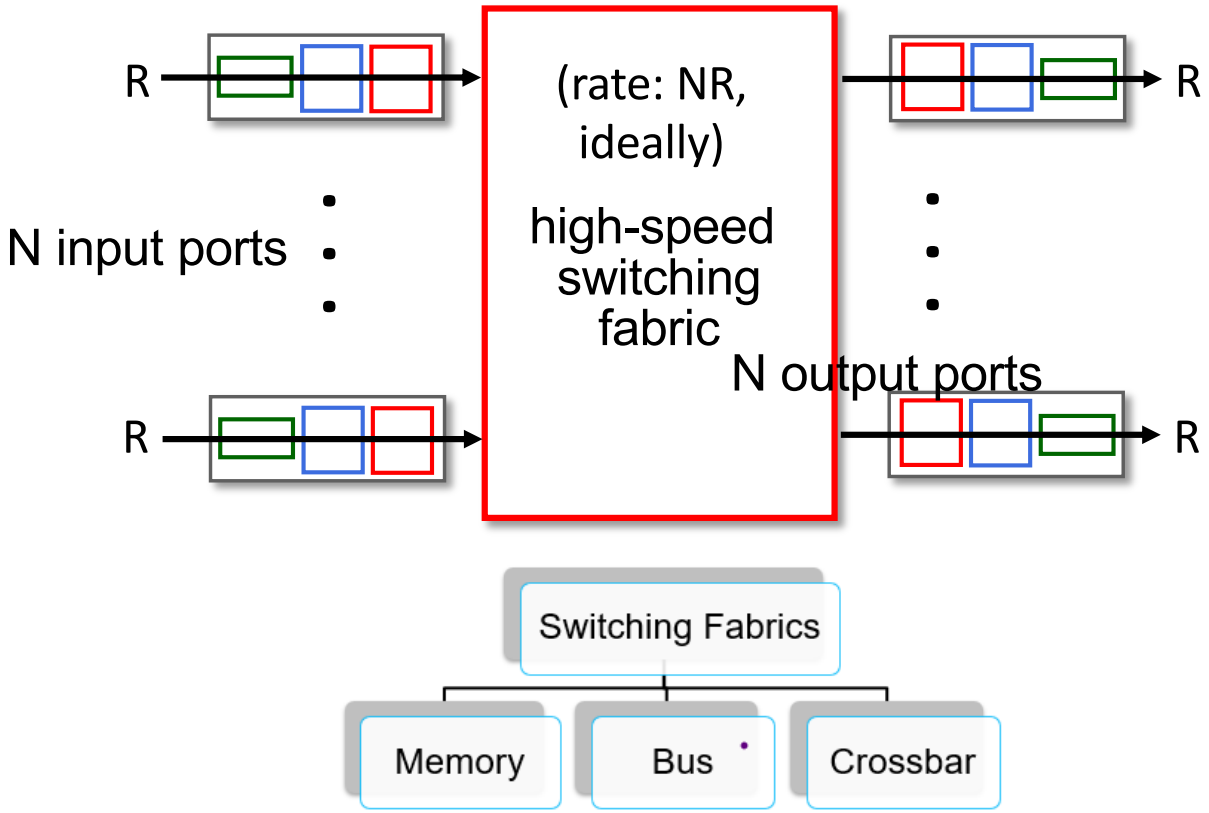
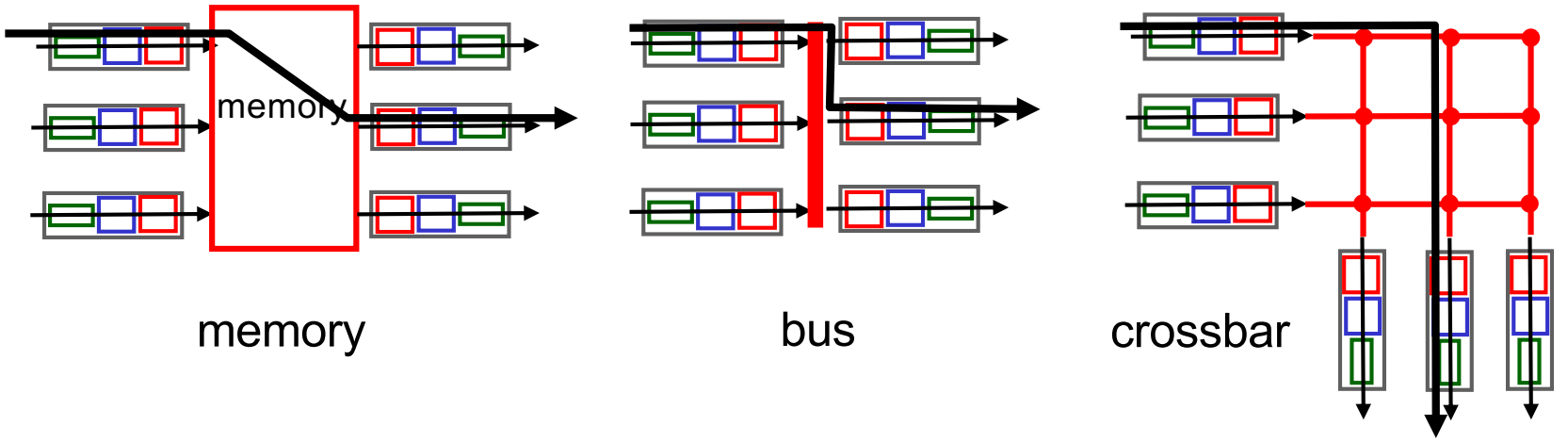


Figure: Three type of switching fabrics techniques

Switching fabrics

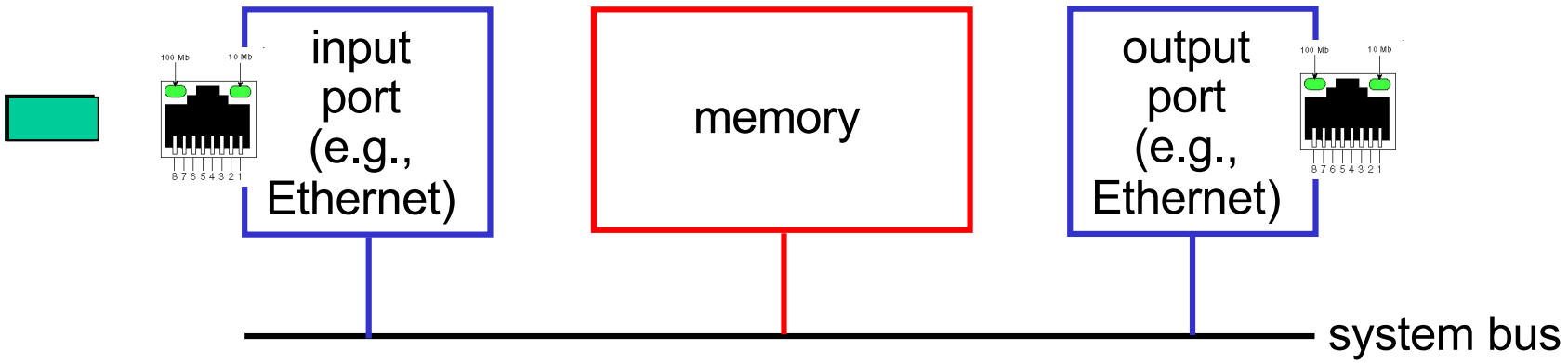
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



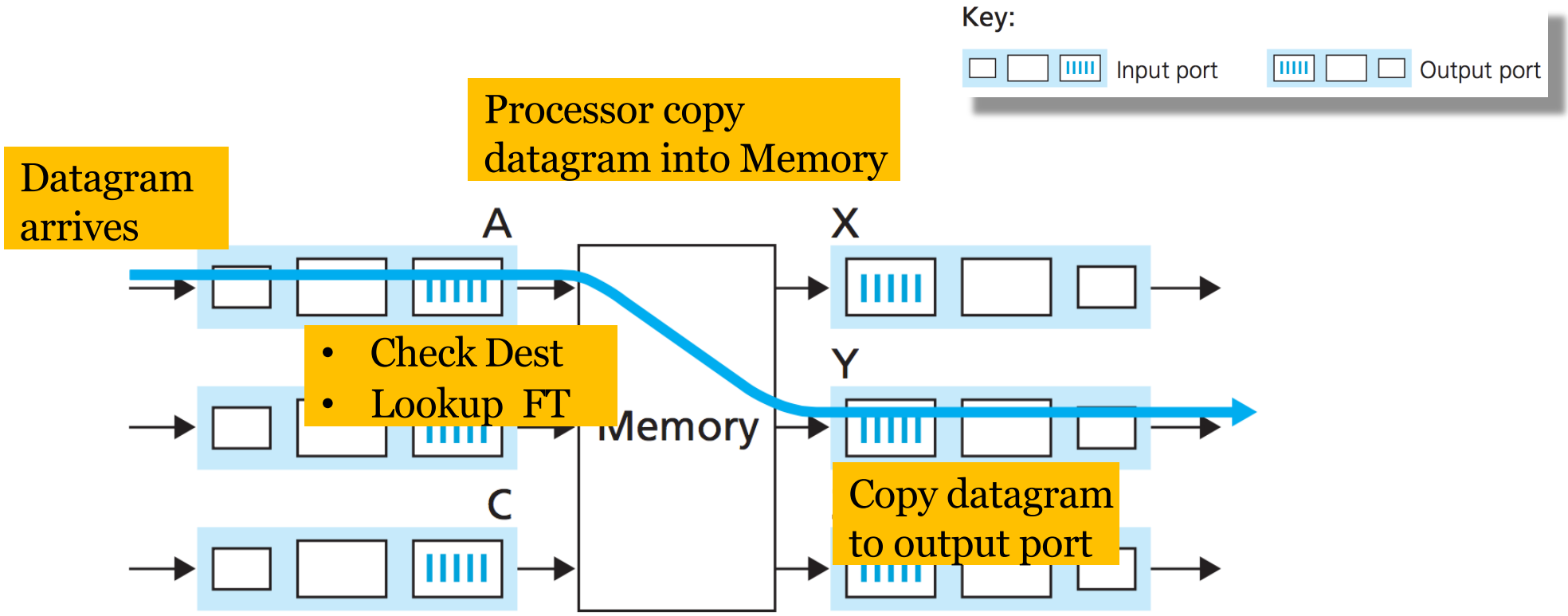
Switching via memory

first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)

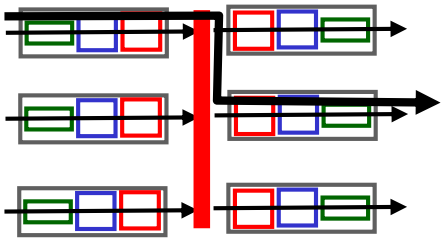


Switching via memory

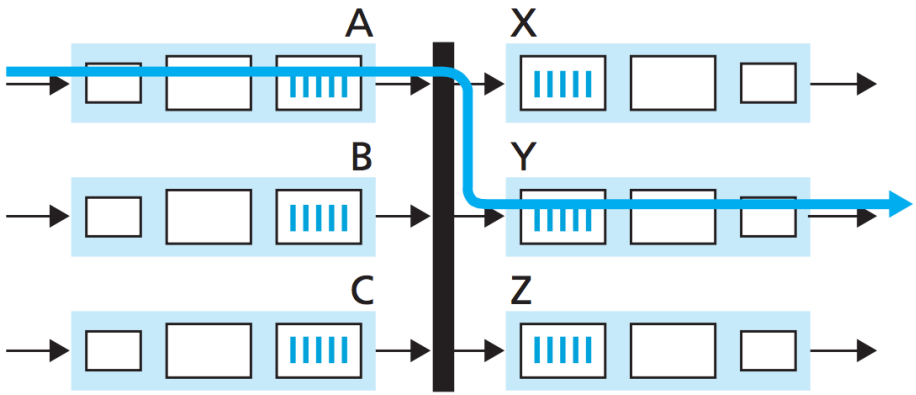


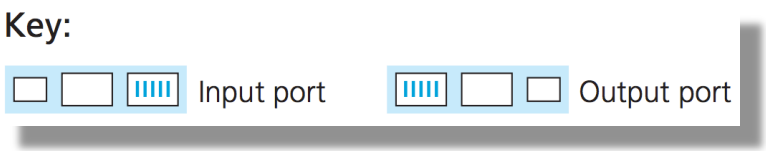
Switching via a bus

- 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

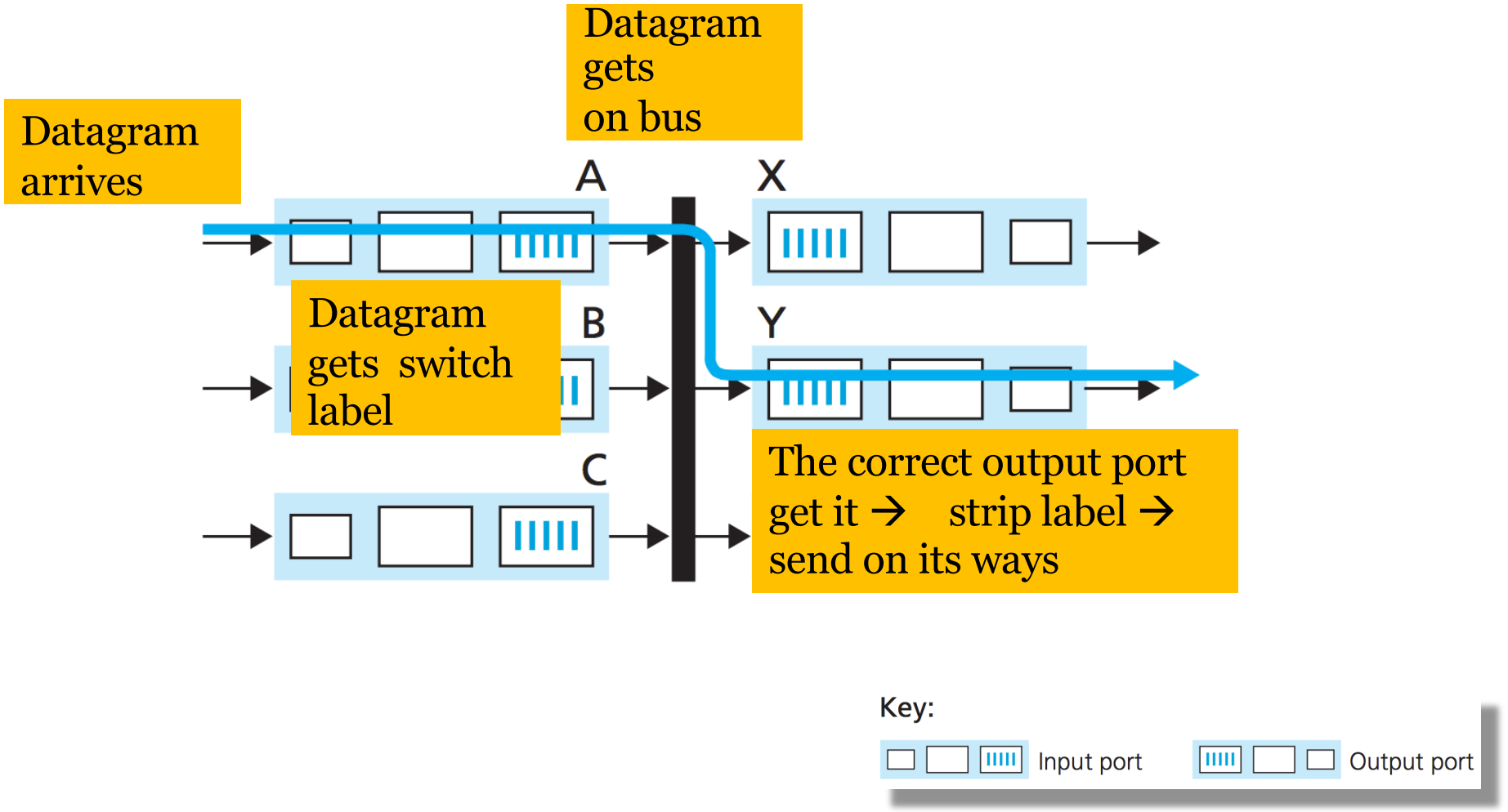


bus



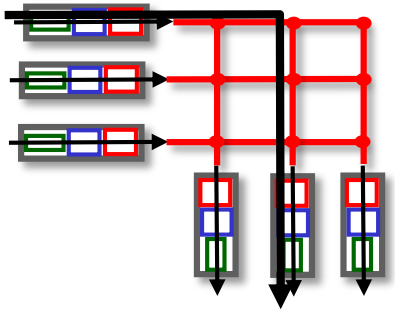


Switching via a bus

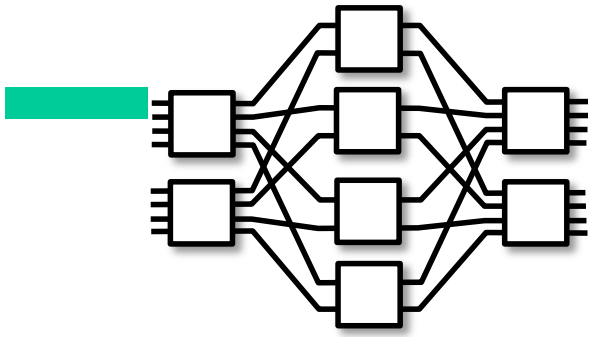


Switching via interconnection network (crossbar)

- overcome bus bandwidth limitations
- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- multistage switch: *nxn* switch from multiple stages of smaller switches
- exploiting parallelism:
  - fragment datagram into fixed length cells on entry
  - switch cells through the fabric, reassemble datagram at exit



3x3 crossbar

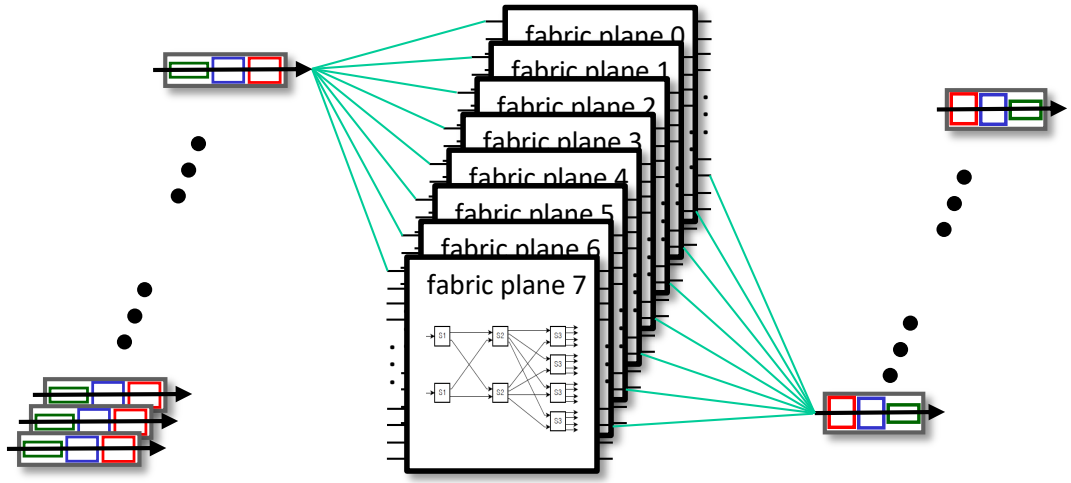


8x8 multistage switch
built from smaller-sized switches

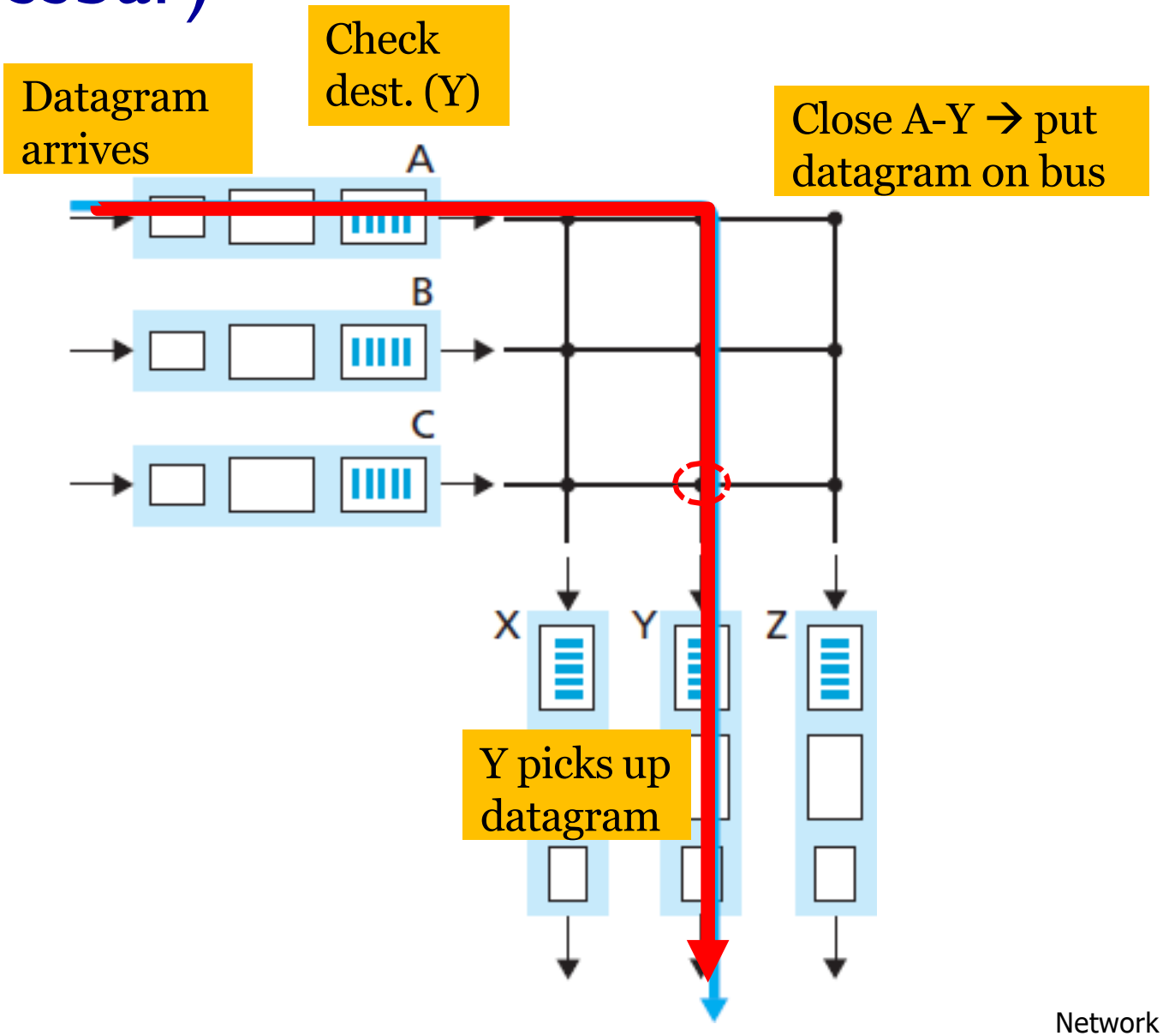
Switching via interconnection network (crossbar)

- scaling, using multiple switching "planes" in parallel:
  - speedup, scaleup via parallelism

- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100's Tbps switching capacity

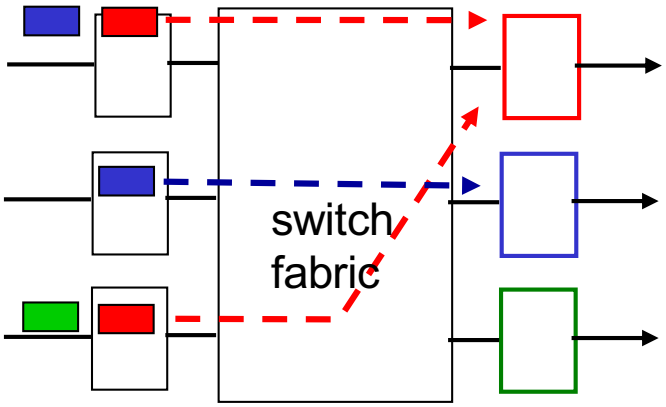


Switching via interconnection network (crossbar)

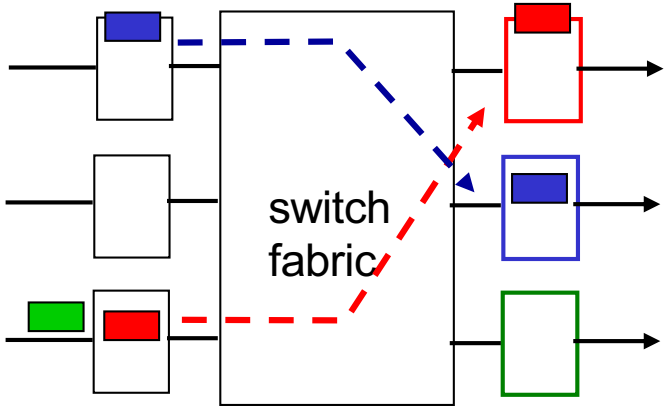


Input port queuing

- If 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

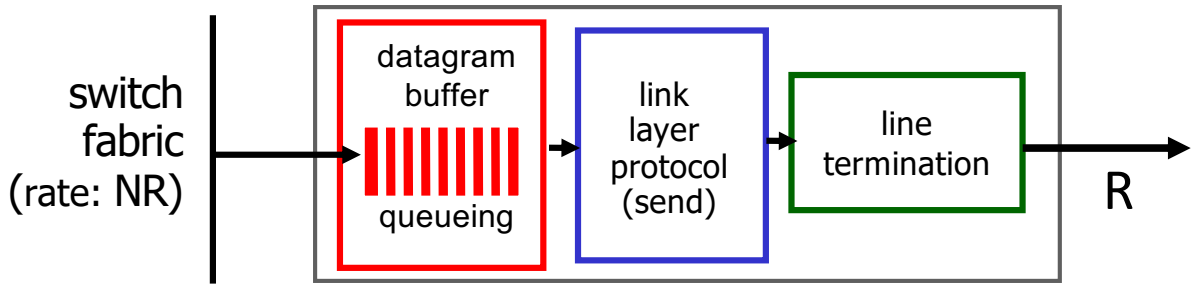


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet
experiences HOL
blocking

Output port queuing



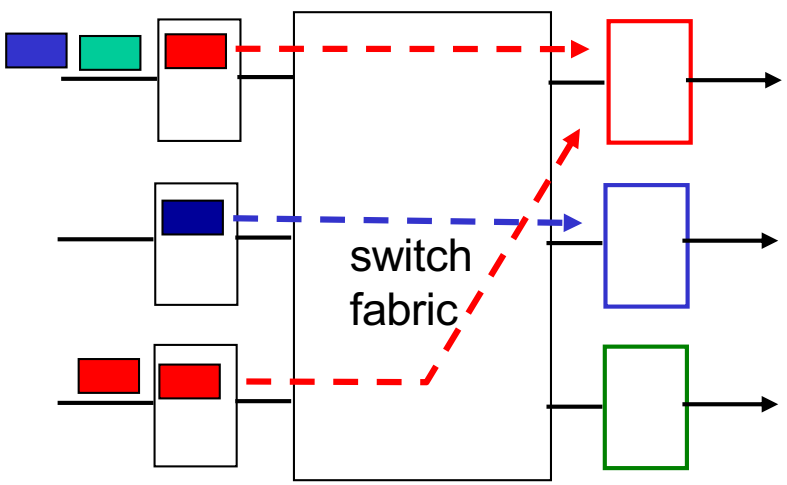
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

→ Datagrams can be lost due to congestion, lack of buffers

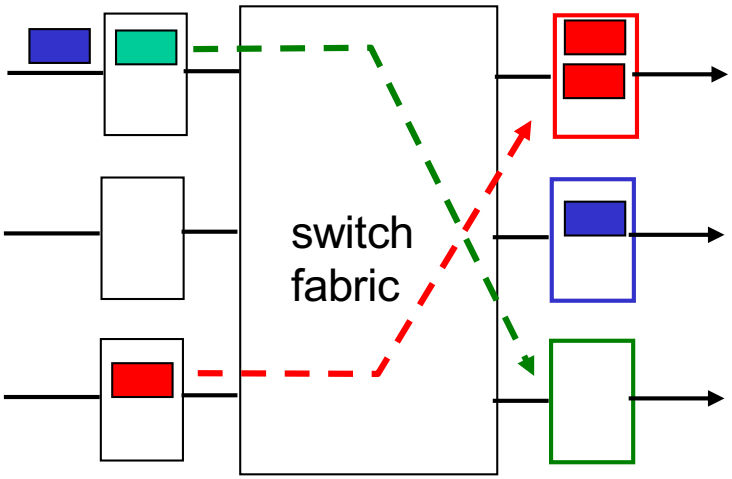
- *Scheduling discipline* chooses among queued datagrams for transmission

→ Priority scheduling – who gets best performance, network neutrality

Output port queueing



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!*

How much buffering?

- RFC 3439 rule of thumb:
  - average buffering size = RTT * link capacity C
  - e.g., RTT=250 msec ("typical"), C = 10 Gpbs link
    - average buffering size =250 msec * 10 Gpbs

$$\gg = 2.5 \text{ Gbit buffer}$$

- More recent recommendation: with *N* flows, buffering equal to

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

- but *too* much buffering can increase delays (particularly in home routers)
  - long RTTs: poor performance for realtime apps, sluggish TCP response
  - recall delay-based congestion control: "keep bottleneck link just full enough (busy) but no fuller"

Exercise

Supposed the link capacity for a router is 20*Mbps* and RTT for a packet is 400*msec*.

If the TCP flow of packets is 16, calculate the buffering needed for the router.

Solution:

- Link capacity, C = 20*Mbps*
- RTT = 400*msec*
- TCP flows, N = 16

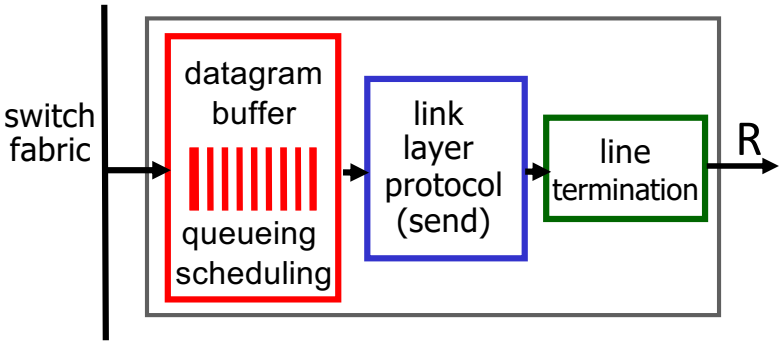
Buffer size, $B = \frac{RTT * C}{\sqrt{N}}$

$$= \frac{400 msec * 20 Mbps}{\sqrt{16}}$$

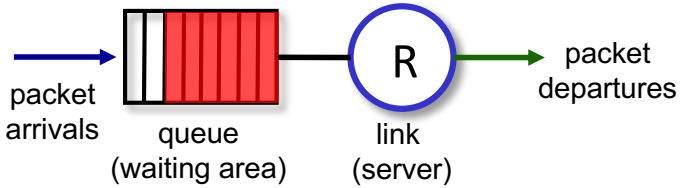
$$= \frac{0.4 sec * 20 Mbps}{4}$$

$$= 0.4 * 5 = 2 Mb$$

Buffer Management



Abstraction: queue



buffer management:

- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

Chapter 4: data plane

4.1 Overview of Network layer
- data plane
- control plane

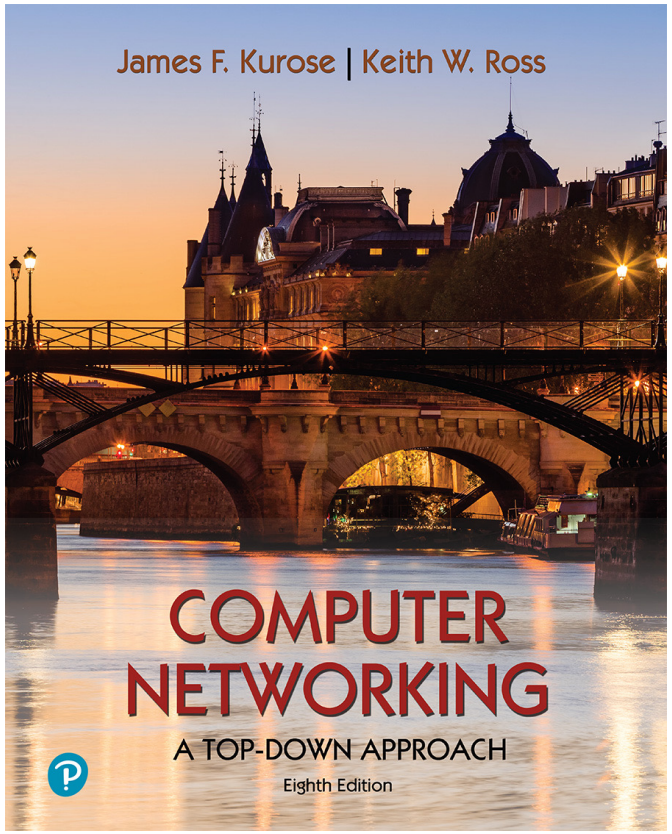
4.2 What's inside a router

input ports, switching, output ports

buffer management, scheduling

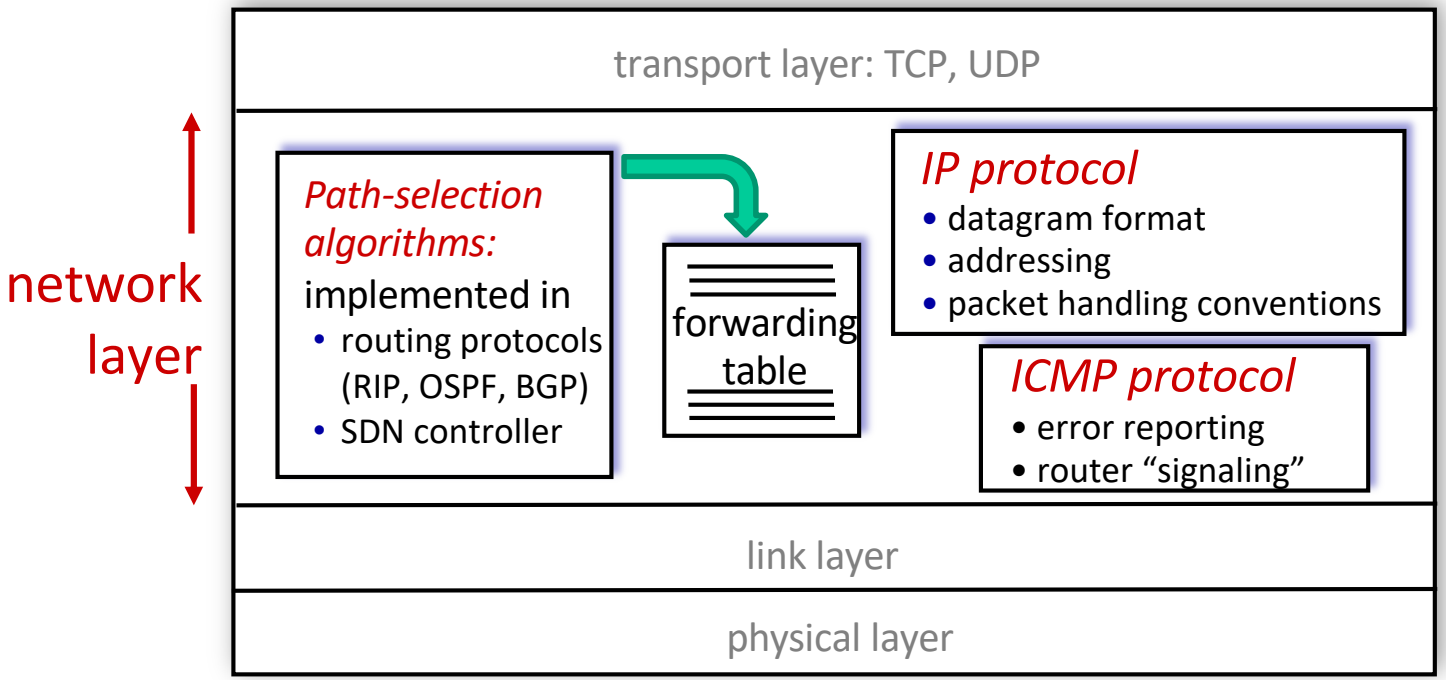
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action



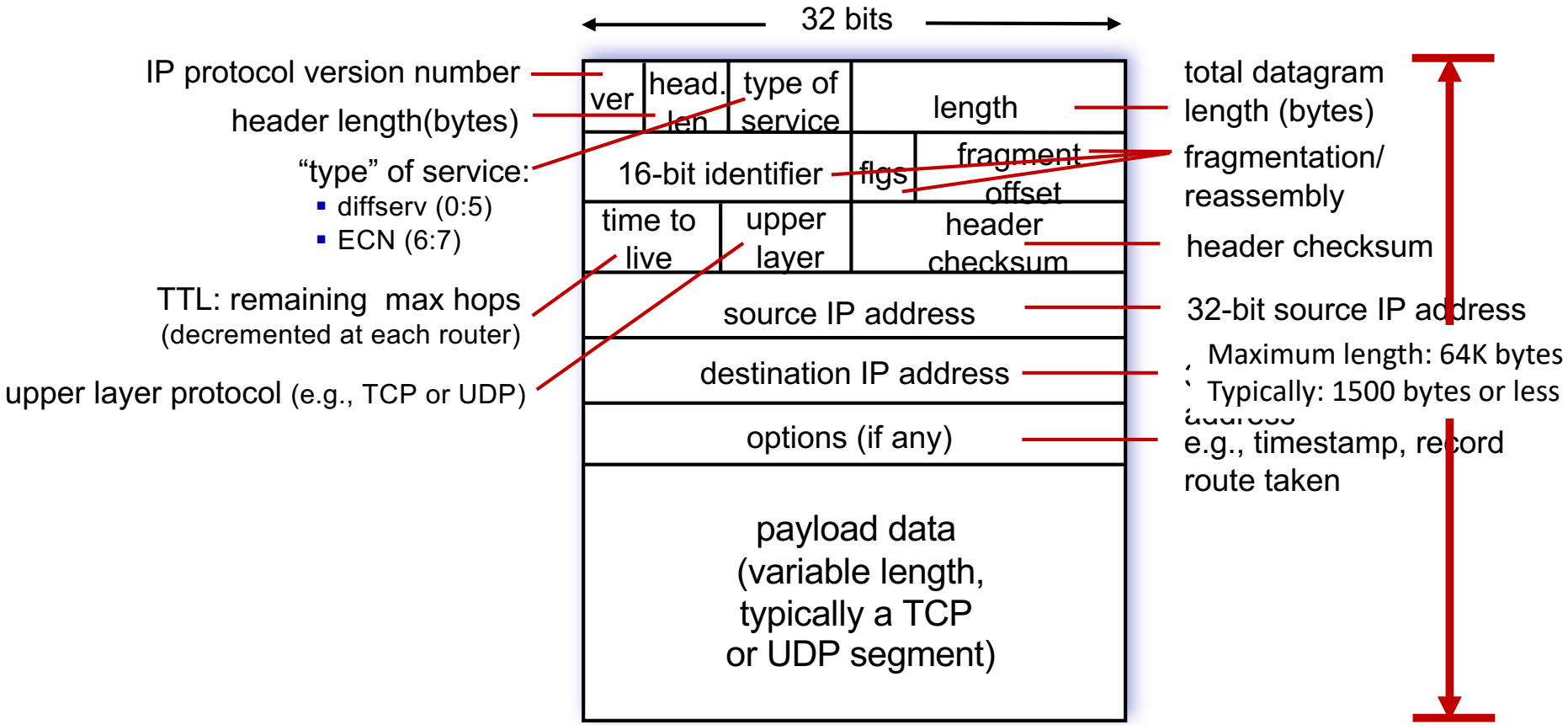
Network Layer: Internet

host, router network layer functions:



RIP (Routing Information Protocol)
OSPF (Open Shortest path First)
BGP (Border Gateway Protocol)

IP Datagram format



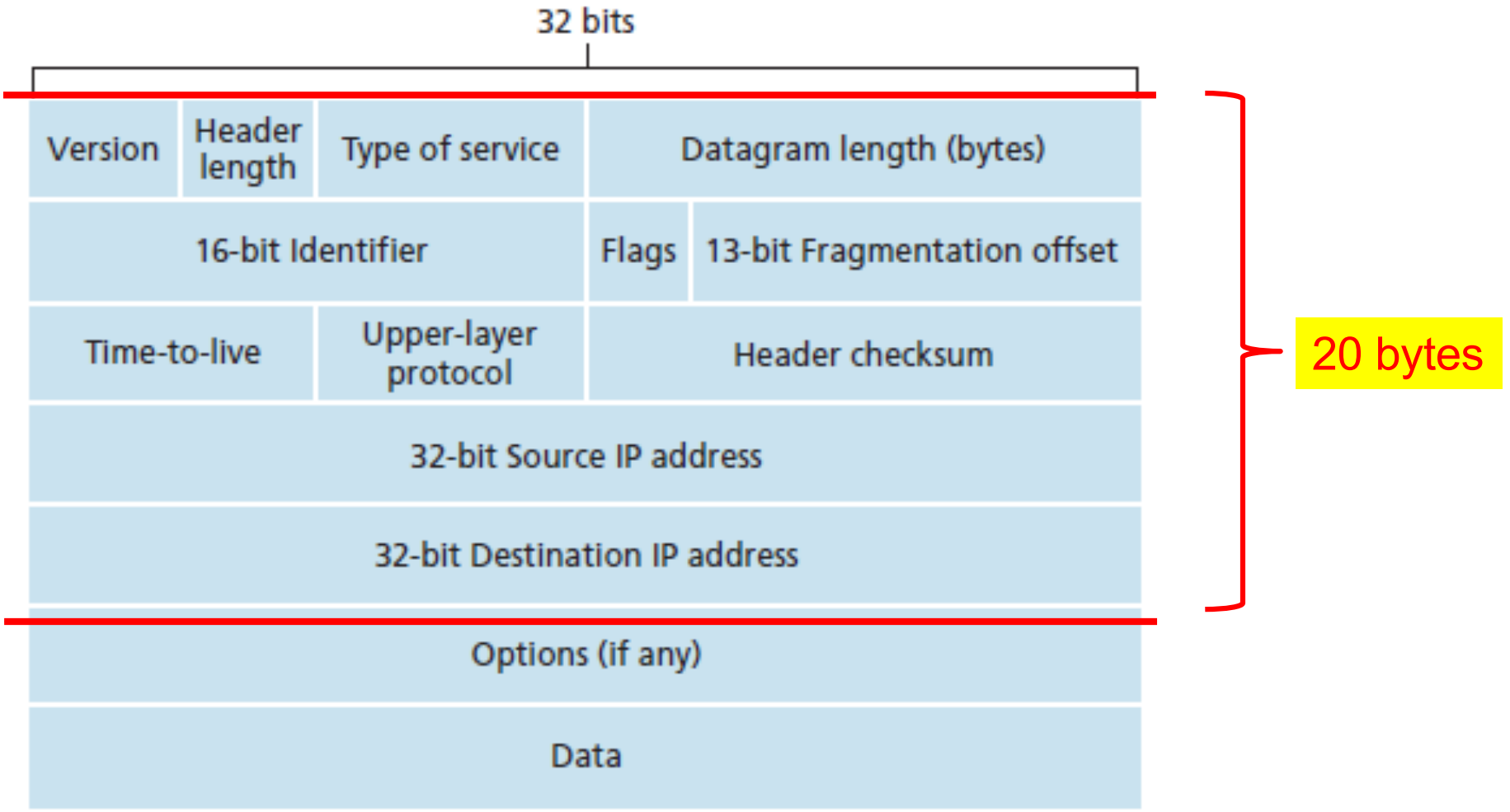
IP datagram format

- Understanding datagram format is important
- A datagram has its own header (overhead) ➔ **20 bytes**
- A datagram holds TCP/UDP segment within it ➔ TCP/UDP has own overhead

overhead

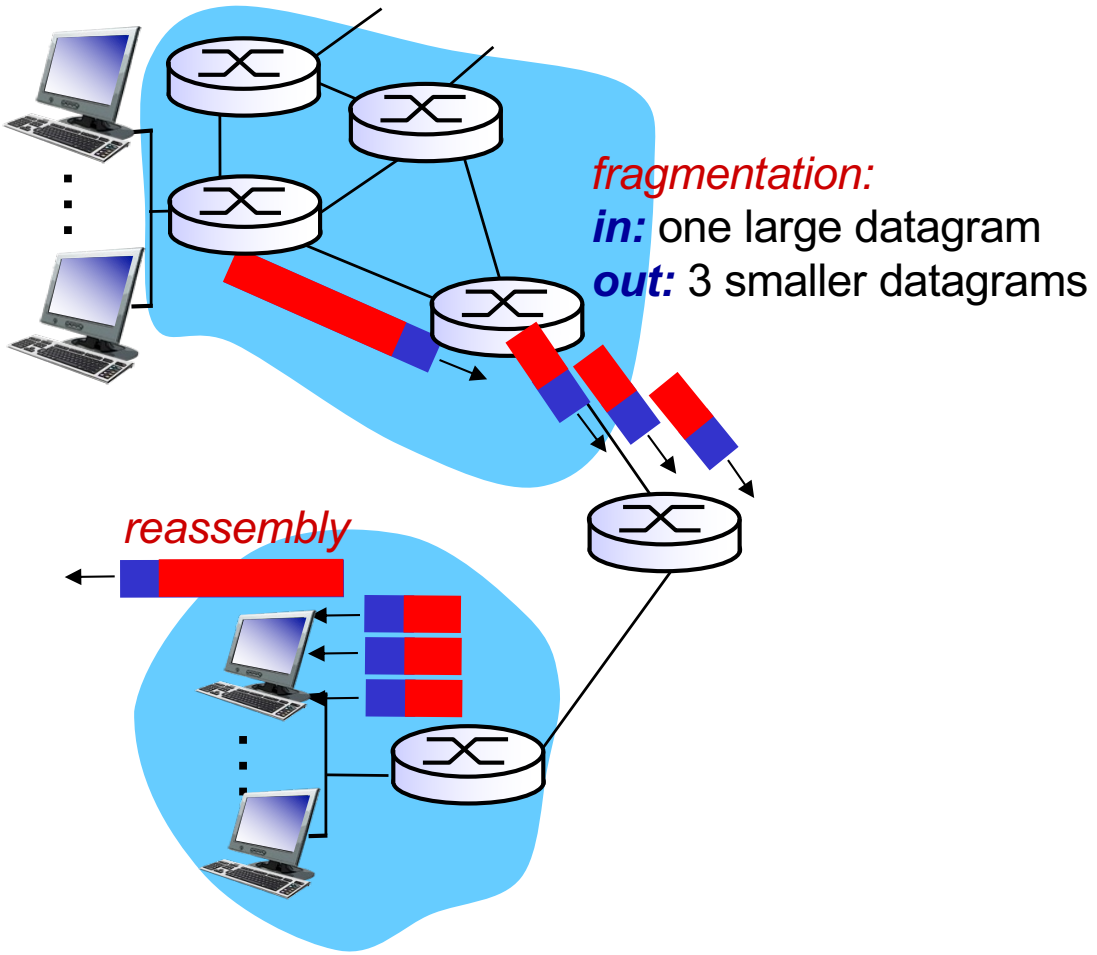
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IP datagram format



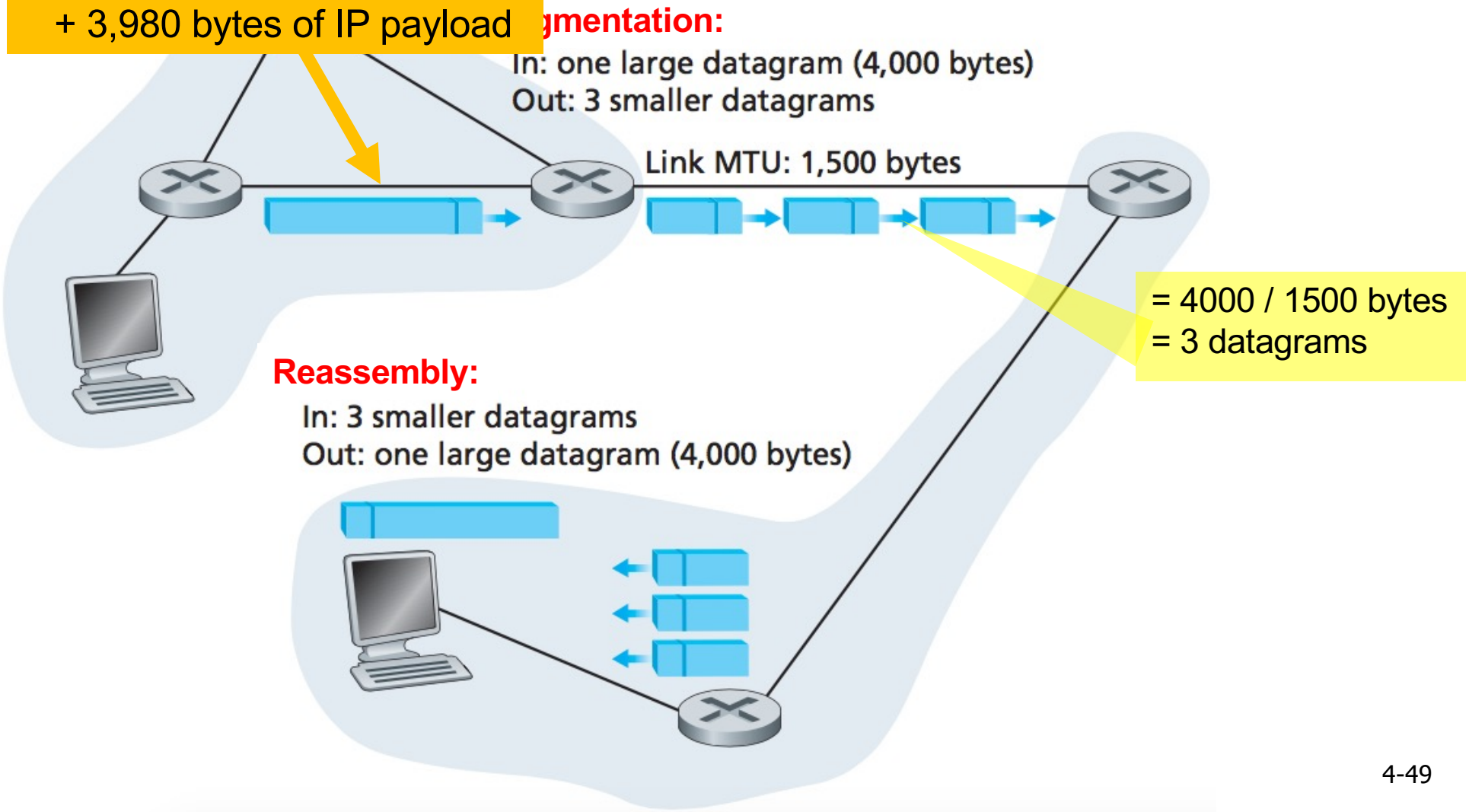
IP fragmentation, reassembly

- network links have MTU (max.transfer unit) - largest possible link-level frame
  - different link types, different MTUs
  - Ethernet MTU 1500 bytes
- 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

Original datagram :
4000 bytes
= 20 bytes of IP header
+ 3,980 bytes of IP payload



IP fragmentation, reassembly

Fragment #1	length=1500	ID=777	flag=1	offset=0
Fragment #2	length=1500	ID=777	flag=1	offset=185
Fragment #3	length=1040	ID=777	flag=0	offset=370

IP fragmentation, reassembly

Data (1480) + Header (20) = 1500 bytes

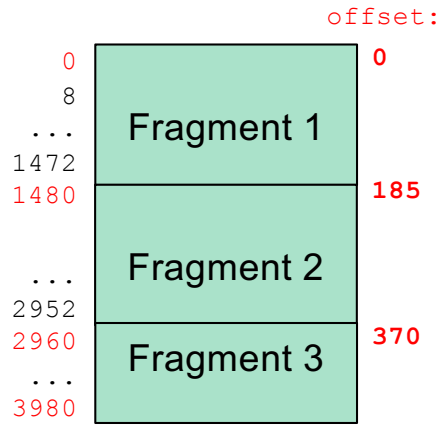
Why 1480 not 1500?

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

Fragments of same datagram

Fragment #1	length=1500	ID=777	f	offset=0
Fragment #2	length=15			fset=185
Fragment #3	length=1040	ID=777	flag=0	offset=370

Offset value specified in units of 8-byte chunks



Exercise

A datagram of 5000 bytes arrived at a router and must be forwarded to a link with an MTU of 1500 bytes. Suppose that the original datagram is stamped with an identification number of 333. Draw all IP fragments generated after fragmentation that reflect the requirement of original payload data in the datagram.

Datagram

length=5000	ID=333	flag=0	offset=0

Solution

Solution:

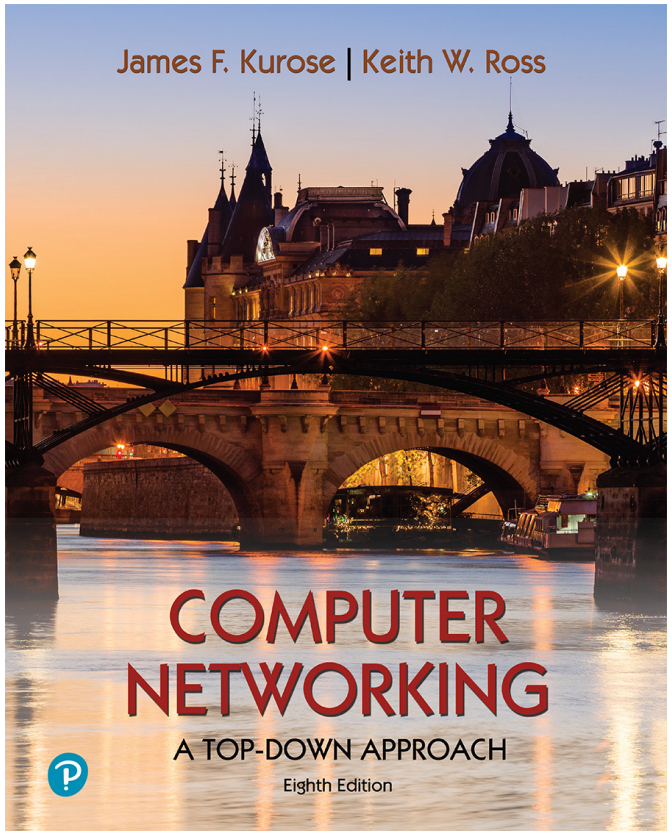
Datagram	length=5000	ID=333	flag=0	offset=0

- 20 header + 4980 data

- Total fragments = 5000 / 1500 = 4
- 4980 = 1480 + 1480 + 1480 + 540

Fragment#1	length=1500	ID=333	flag=1	offset=0
Fragment#2	length=1500	ID=333	flag=1	offset=185
Fragment#3	length=1500	ID=333	flag=1	offset=370
Fragment#4	length= 560	ID=333	flag=0	offset=555

Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

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

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

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

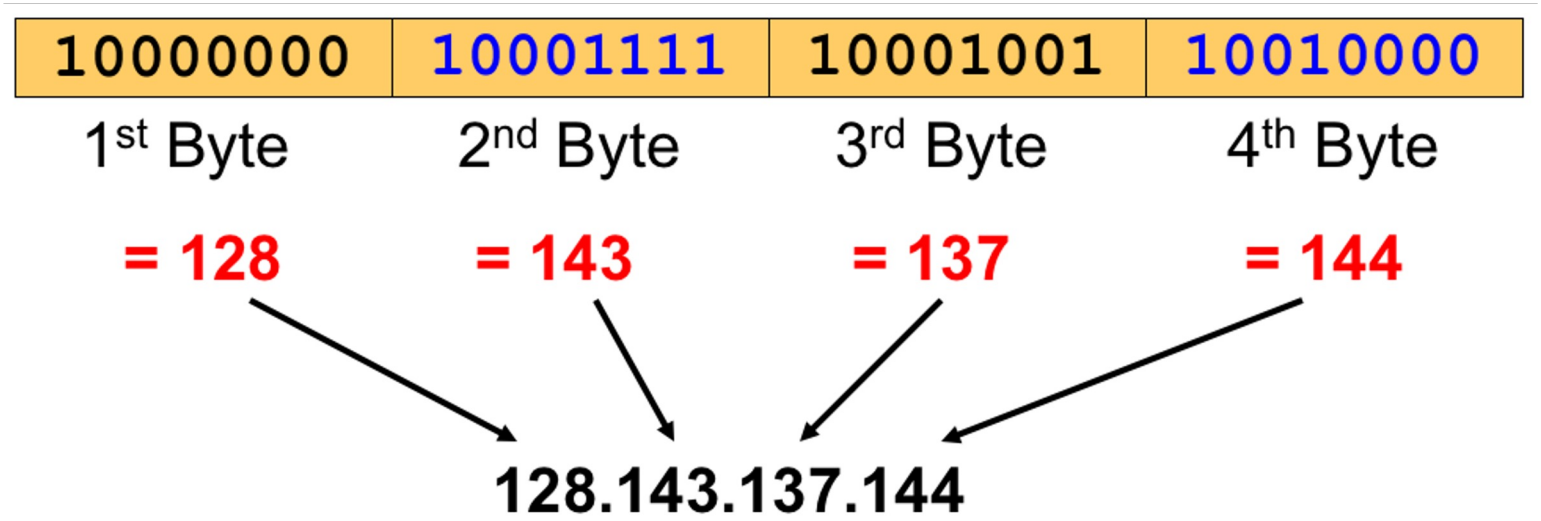
What is an IP Address?

- IP address is used to identify a host within a network.

  are written in a so-called ***dotted decimal* notation**

  Each byte is identified by a decimal number in the range [0…255]:

- **Example:**

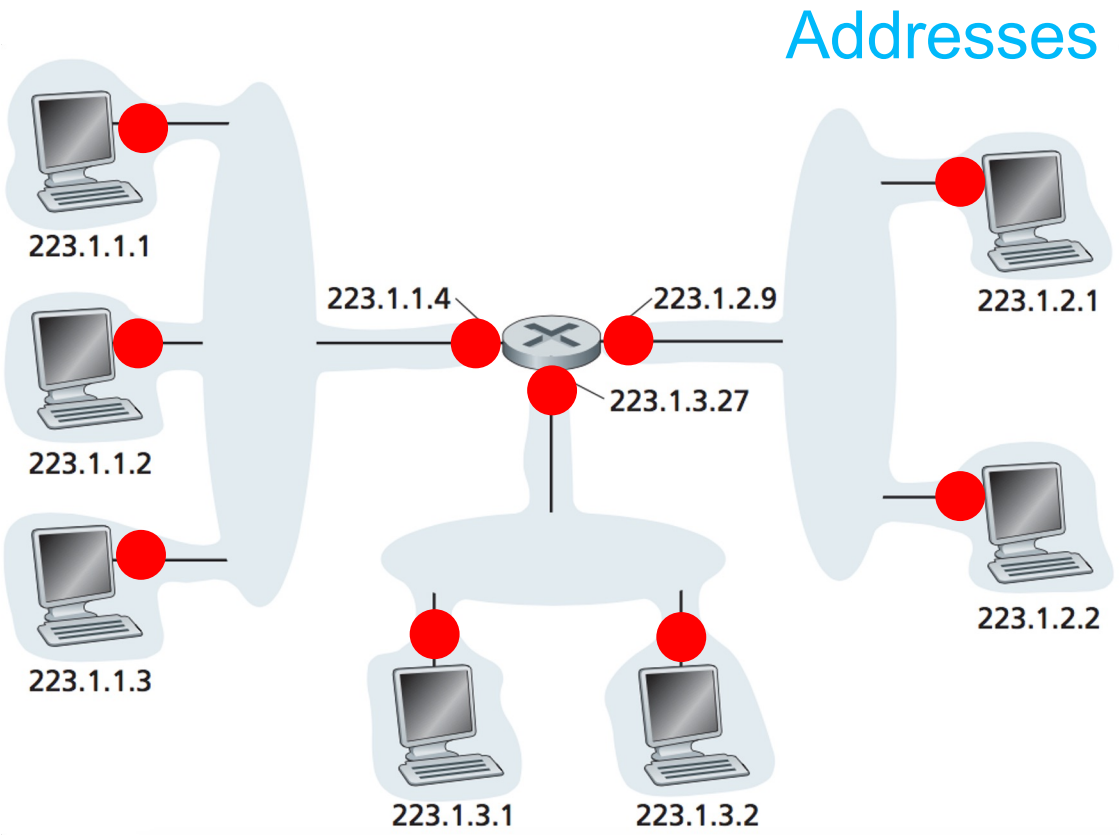


IPv4 Addressing

- *IP address:* 32-bit identifier for each host or router *interface*
- *Interface:* connection between host/router and physical link.

router's typically have multiple interfaces.

host typically has one or two interfaces. (e.g., wired Ethernet, wireless 802.11)



One IP address associated with each interface !

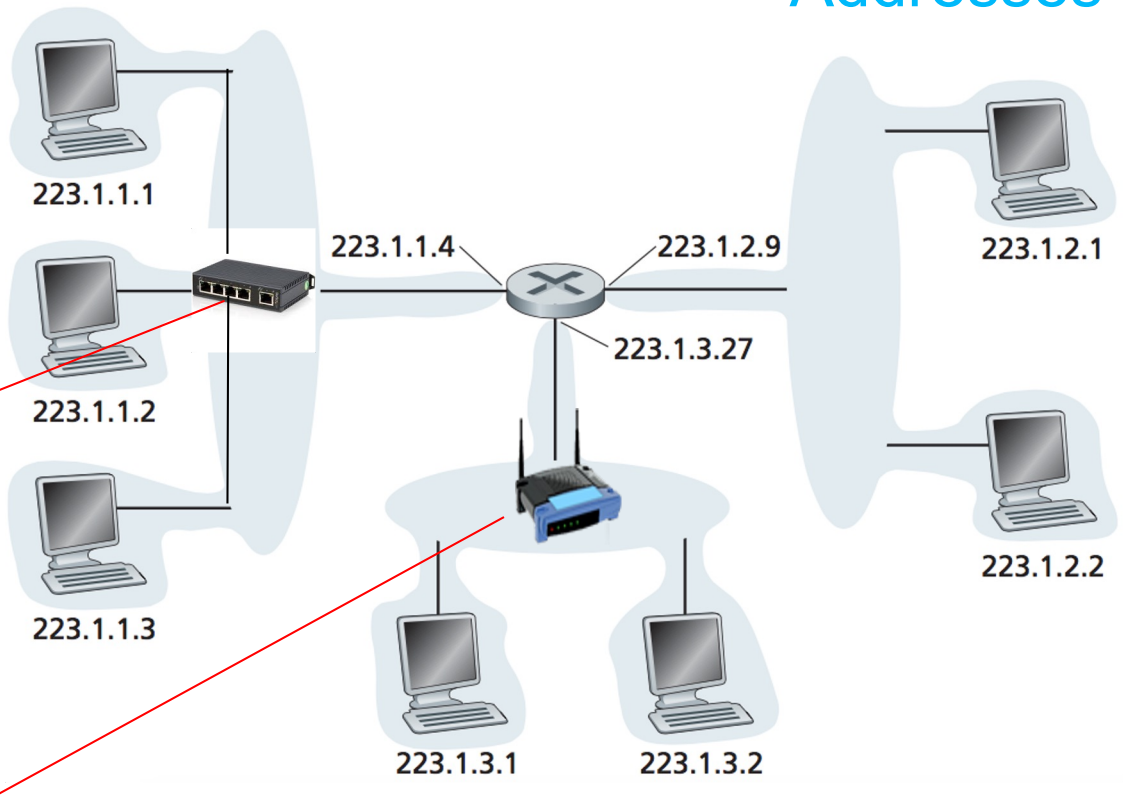
IPv4 Addressing

Addresses

Q: How are interfaces actually connected?

A: wired Ethernet interfaces connected by **Ethernet switches** (Link Layer)

A: wireless WiFi interfaces connected by WiFi base station (**AP** – access point)

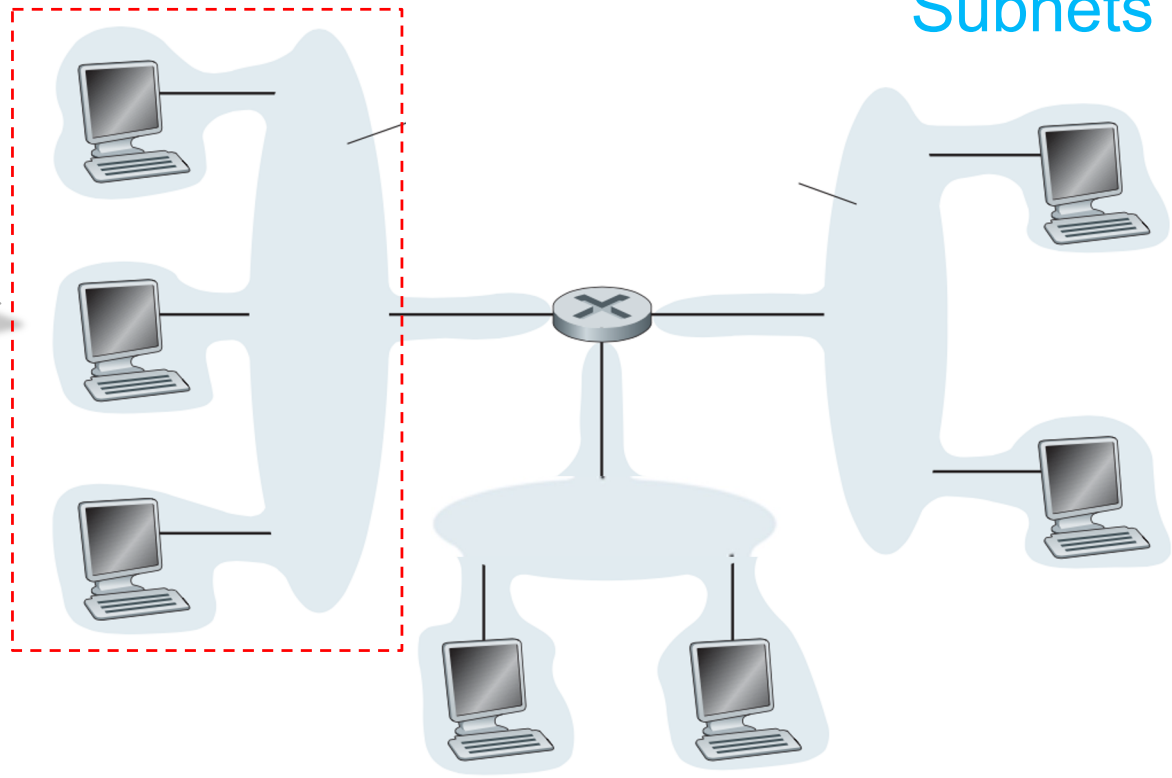


IPv4 Addressing – **Subnet/Subnetting**

IPv4 Addressing

Subnets

Interconnecting three host interfaces and one router interface forms a **subnet**.



What's a **subnet** ?

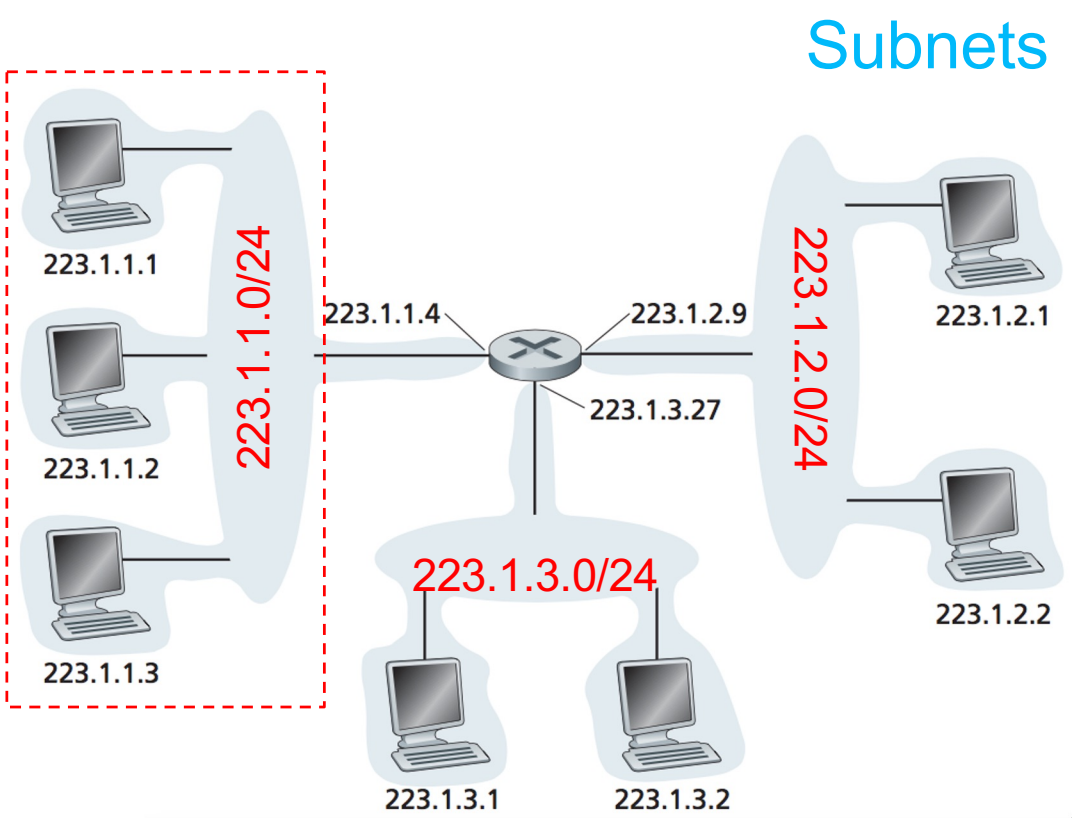
- device interfaces that can physically reach each other without passing through an intervening router
- is a logical, visible subdivision of an *IP network*. The practice of dividing a network into two or more networks is called *subnetting*.

IPv4 Addressing

- A subnet is a group of host identified with having the same network portion of their IP address
- **Example**:
  - 223.1.1.1, 223.1.1.2, 223.1.1.3, 223.1.1.4, …

    223.1.1.1, 223.1.1.2, 223.1.1.3, 223.1.1.4, …
- This network portion is indicated by the subnet mask with notation **/24**
- Subnet mask defines the subnet address

→ 223.1.1.0/24



Network consisting of 3 subnets

IPv4 Addressing

Subnet Mask

- A computer OR a router must be able to identify whether a host with a given IP address is on its subnet or not.
- The **subnet mask** is:

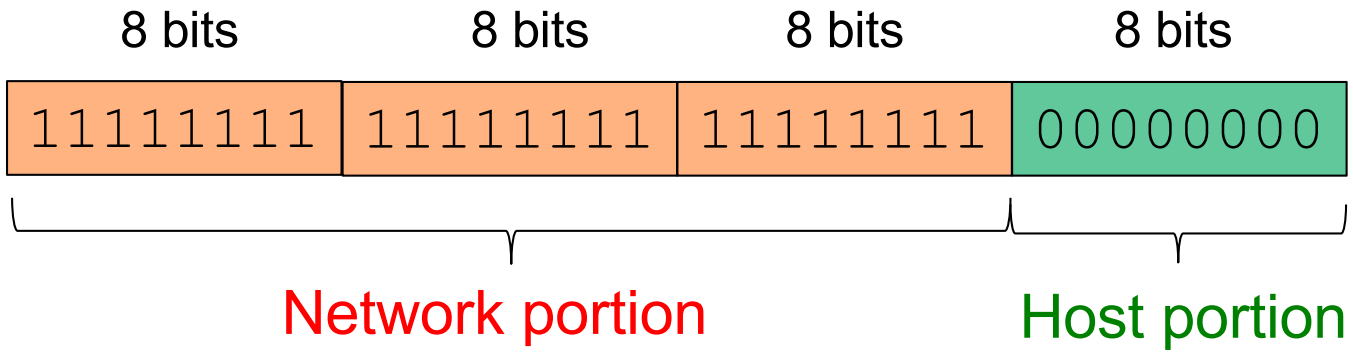
used to separate the network portion of an IP address from the host portion.

a set of 32 bits IP address which the bits in the network portion of the address are set to **1s** and the host portion is set to **0s**.

IPv4 Addressing

Subnet Mask

- IP address with **w.x.y.z /24** address.



- Thus, subnet mask = **255.255.255.0**

IPv4 Addressing

Subnet Mask

- Each IP address can be divided into network portion and host portion

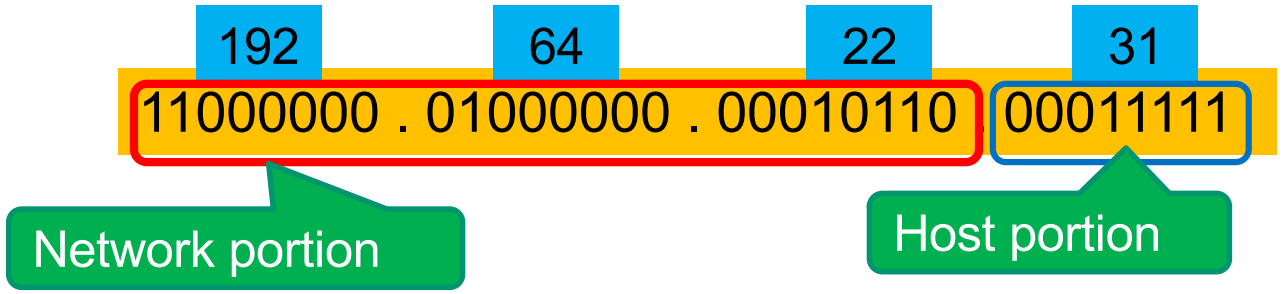
  Network portion (indicated by subnet mask)

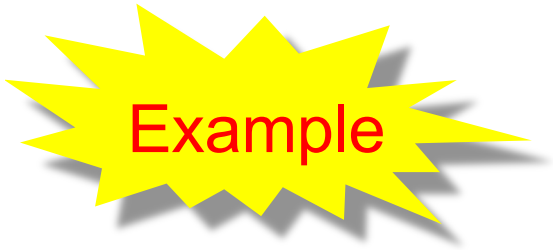
  - all host with same network portion are in the same subnet
  - these host can physically reach each other without intervening router

  Host portion (the remaining bits → {32 bits – subnet mask })

  - The individual (unique) address for a host

Example: 192.64.22.31/24





IPv4 Addressing

Subnet Mask

Convert these subnet mask into decimal values.

(a) /8 11111111.00000000.00000000.00000000
255.0.0.0

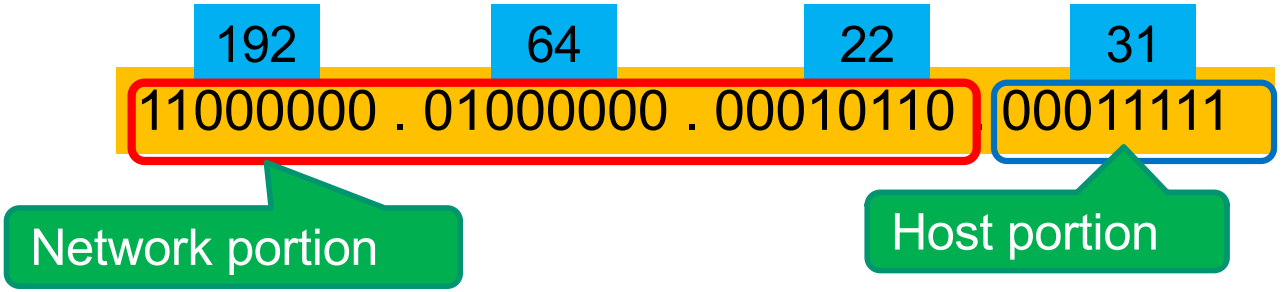
(b) /16 11111111.11111111.00000000.00000000
255.255.0.0

(c) /24 11111111.11111111.11111111.00000000
255.255.255.0

(d) /25 11111111.11111111.11111111.10000000
255.255.255.128

IPv4 Addressing

192.64.22.31/24



- Given the information we find that:

  The host portion has 8 bits →

  - this subnet can have up to 2^8 = **256 hosts**
  - With IP addresses ranging from 192.64.22.0 – 192.64.22.255

  But the **first** and **last** IP address **cannot** be assigned to host

  - so only **254 hosts available** (or usable)
  - …thus useable IP addresses → **192.64.22.1 – 192.64.22.254**

  The **first IP** – becomes network address → to address the subnet

  The **last IP** – becomes broadcast address → used to broadcast info throughout the subnet

Subnet Network Address (Binary Method) and Broadcast Address

- Given an IP address and subnet mask, we can find the network address and broadcast address …
- Example: IP address = 192.168.1.10/24

  Set the host portion as **all zero(0)** to get network address

  Set the host portion as **all one(1)** to get broadcast address

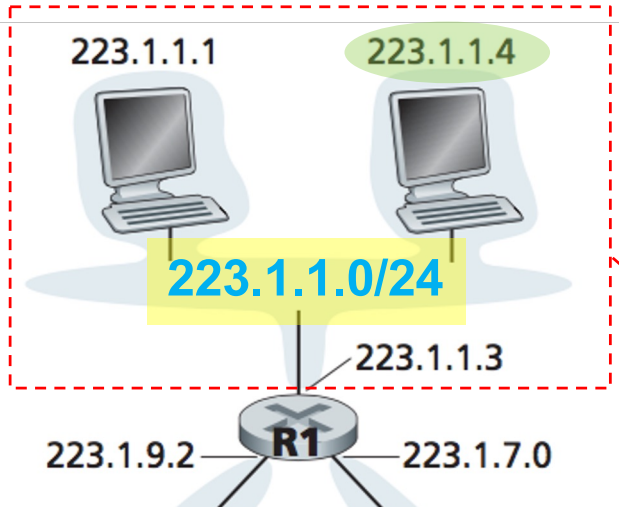
192.168.1.10

192.168.1.**[00000000]** → Network address: **192.168.1.0**

192.168.1.**[11111111]** → Broadcast address: **192.168.1.255**

IPv4 Addressing

Subnet Network Address (AND Method)



To determine subnet address:

IP address **AND** subnet mask

Example:
If the subnet mask is **255.255.255.0**, define the *subnet address*.

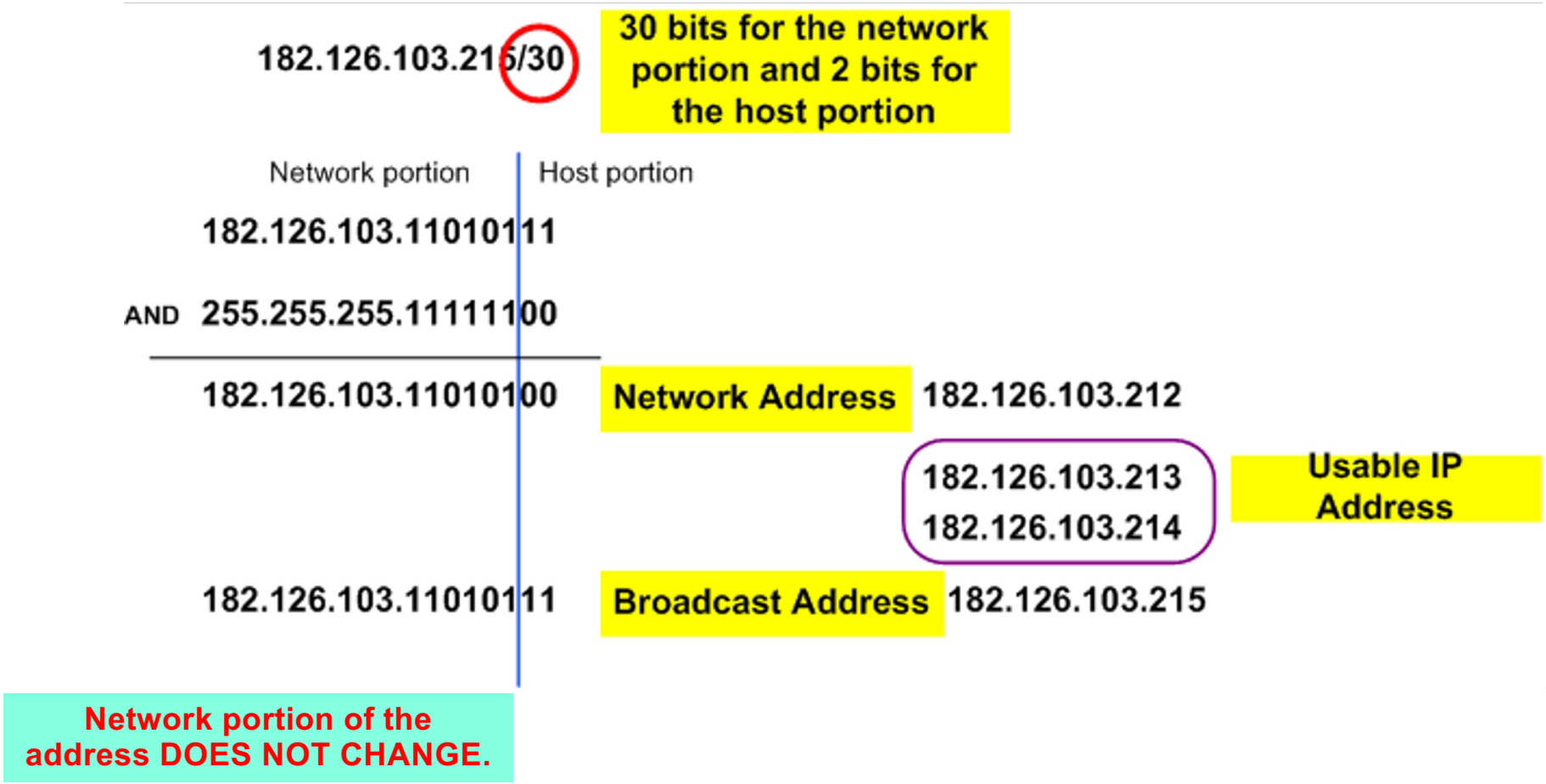
```
      223. 1 . 1 .4
AND   255.255.255.0
```

```
11011111.00000001.00000001.00000100
11111111.11111111.11111111.00000000
11011111.00000001.00000001.00000000
```

223.1.1.0/24

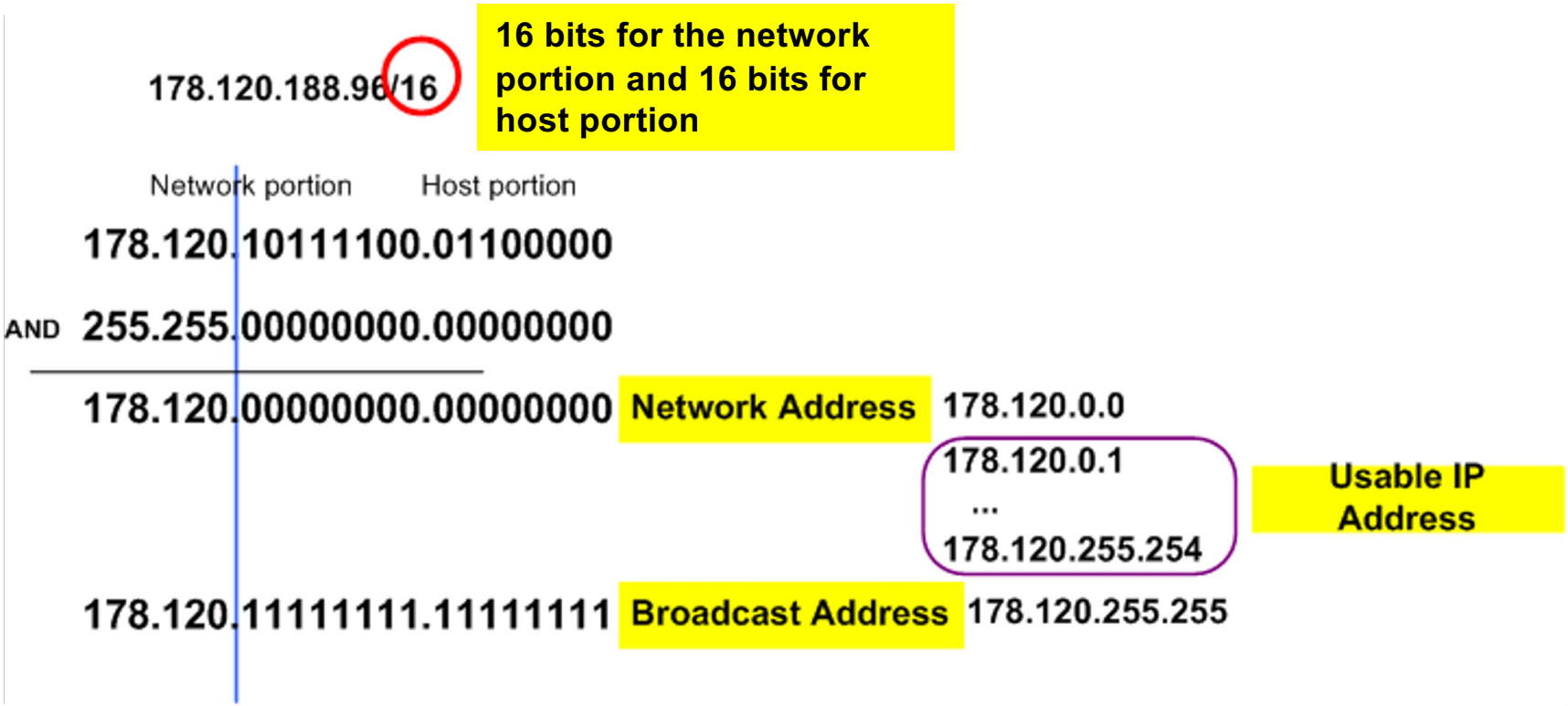
Example

182.126.103.215/30



Example

178.120.188.96/16

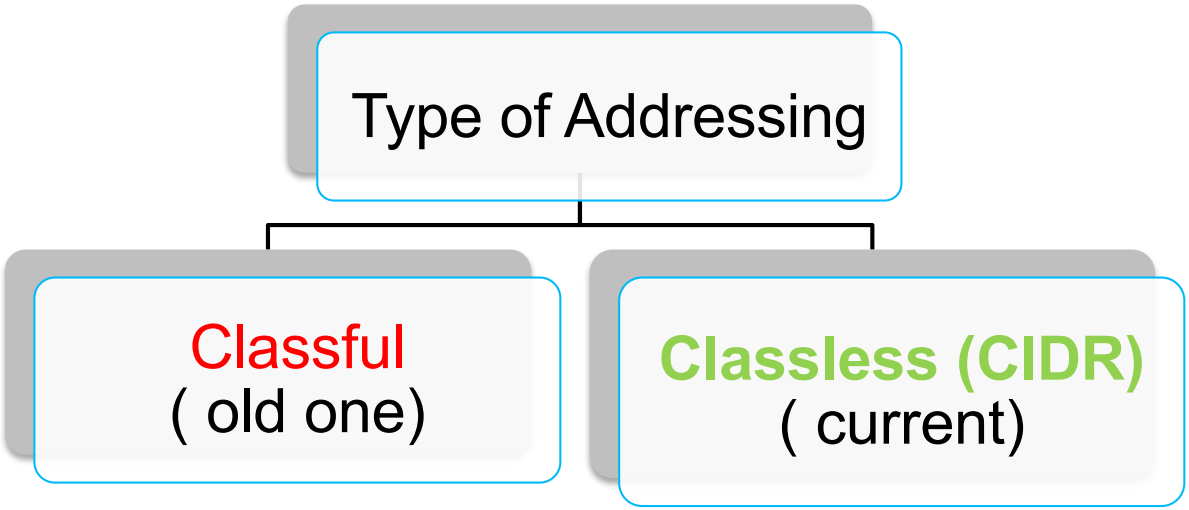


Network portion of the address DOES NOT CHANGE.

Class Exercise

- Given these IP addresses, determine the network portion and the host portion
  a. 192.168.12.0/24
  b. 192.168.12.25/25
  c. 172.16.32.0/16
  d. 172.16.32.0/20
  e. 10.10.10.0/8

IPv4 Addressing – **Type of Addressing**



- In the beginning, addressing scheme known as classful addressing →

  where IP blocks were given according to size

  Class A – big size of organization

  Class B – medium size of organization

  Class C – small size of organization

Classful addressing

Class A
Subnet Mask

Netwok	Host	Host	Host
255	0	0	0

Class B
Subnet Mask

Netwok	Network	Host	Host
255	255	0	0

Class C
Subnet Mask

Netwok	Network	Network	Host
255	255	255	0

www.smartPCtricks.com

Classful addressing

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved for future use, or Research and Development Purposes.

/8 255.0.0.0

/16 255.255.0.0

/24 255.255.255.0

Classful addressing

Example:

- An organization needs **2000** hosts and apply class B
- **Class B** allocated 65000 addresses, and leaving more than 63000 not used.

- **PROBLEMS**: Wasted addresses and not optimized

Large organizations do not fully utilize the IP address they have

Had problem supporting the rapidly growing number of organizations with small and medium-sized subnets

Classless Inter Domain Routing (CIDR)

more flexible than original system of internet address scheme (classful addressing), subnet portion of address of arbitrary length

can avoid situations where large numbers of IP addresses are **unused**

address format: **a.b.c.d/x**

200.23.16.0/23

where **x** is no. of bits in subnet portion of address often referred to as **prefix**

11001000.00010111.00010000.00000000

Subnet part / Network part

host part

Classless Inter Domain Routing (CIDR)

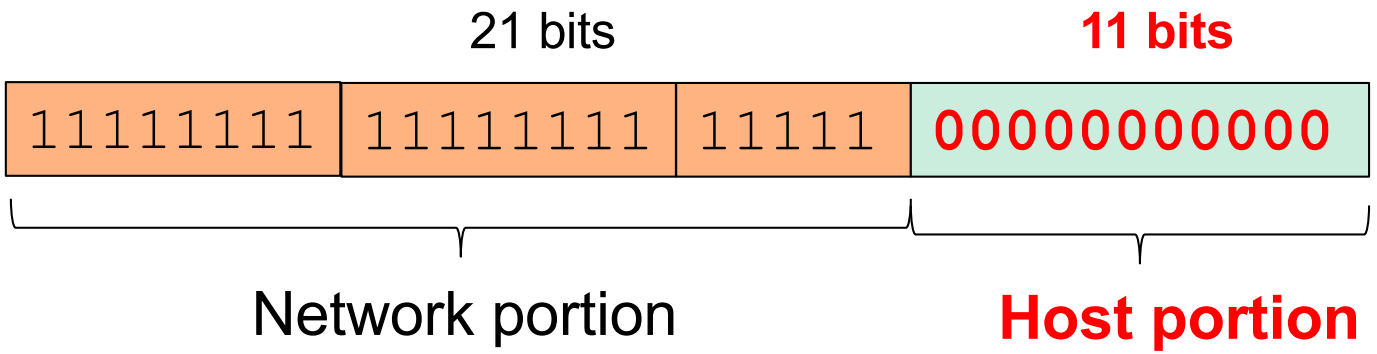
Example:

An organization needs **2000** hosts and apply for IP address

Host needed is 2000 → $2^{11} = 2048$ (i.e. only extra 48)

Thus an IP address→ **163.44.224.0/21** is given

- How we get … **/21** ? → 32 – **11** = 21

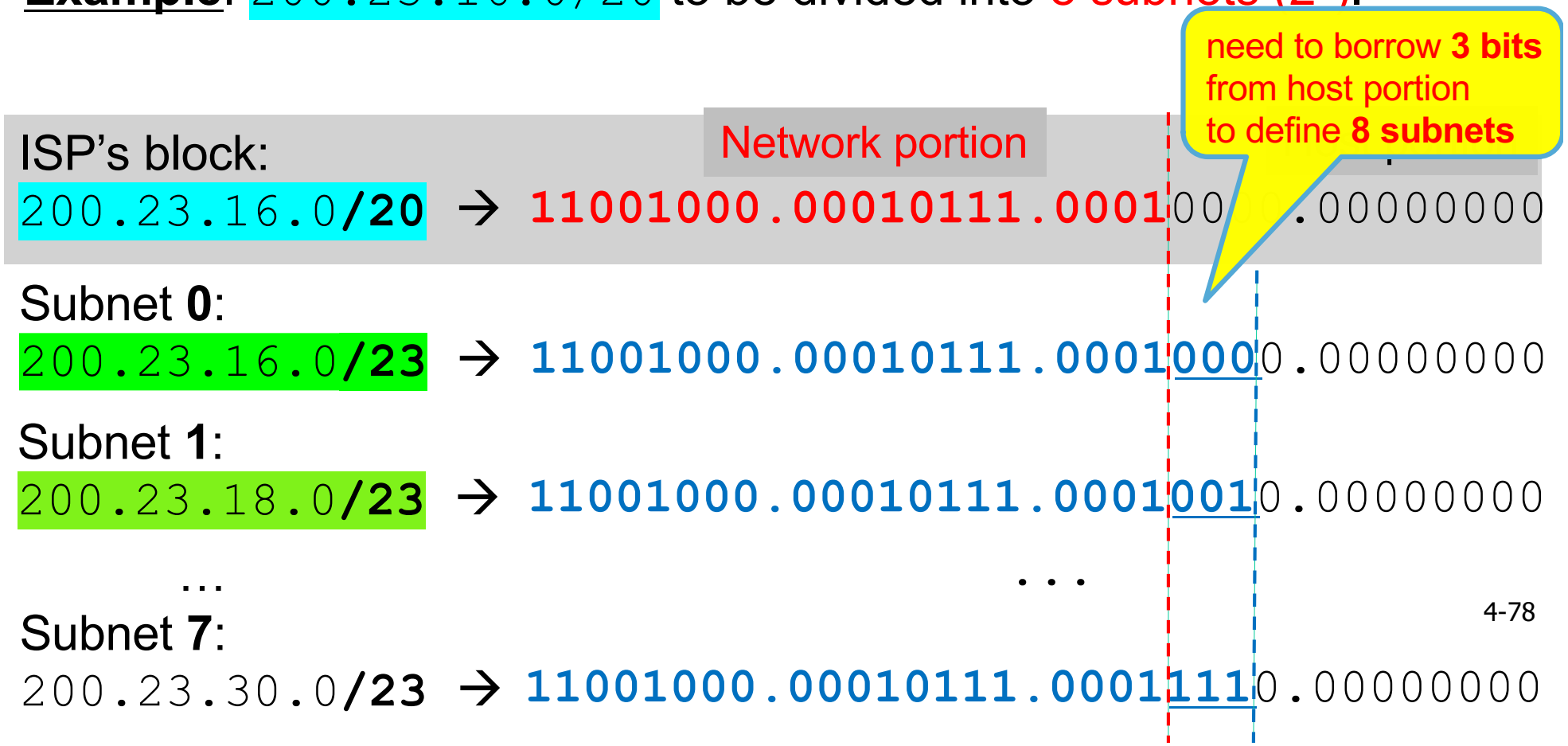


Subnetting: To subnet a small network

Q: How does *a network* get subnets part of an IP address?

A: by subnetting the given IP address

Example: 200.23.16.0/20 to be divided into 8 subnets (2^3).



Subnetting (Example): To subnet a small network

- Say, Company A (CA) is given the IP address 192.168.1.0/24
- CA has 4 departments under it:
  - RND
  - Sales
  - Finance
  - Human Resource (HR)
- CA decided to divide the network according to purpose. So, the network needs to have 4 subnets.

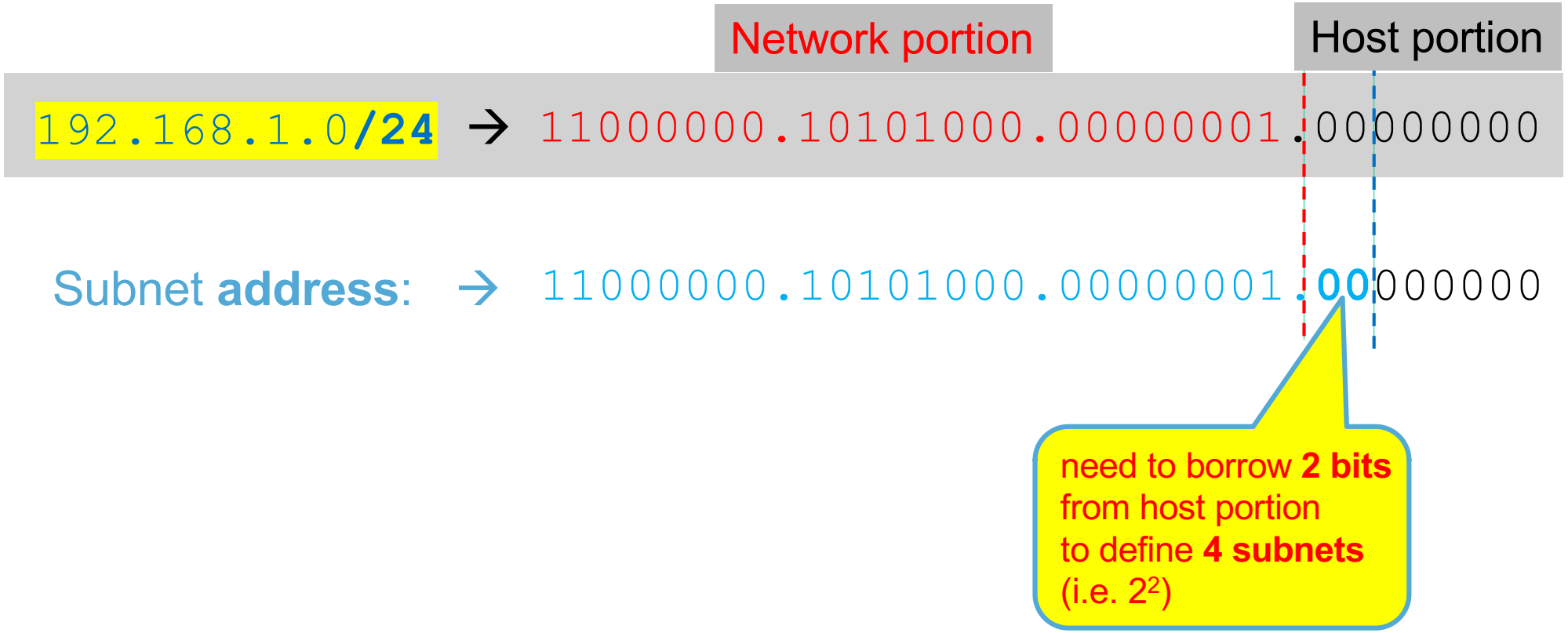
Subnetting (Example): To subnet a small network

192.168.1.0/24 to be divided into 4 subnets

192.168.1.0/24 → 11000000.10101000.00000001.00000000

Subnetting (Example): To subnet a small network

192.168.1.0/24 to be divided into 4 subnets



Subnetting (Example): To subnet a small network

192.168.1.0/24 to be divided into 4 subnets

Network portion | Host portion

192.168.1.0/24 → 11000000.10101000.00000001.00000000

Subnet **address**: → 11000000.10101000.00000001.00000000

Subnet **0**: → **11000000.10101000.00000001.00**000000

192.168.1.0/26

The subnet mask for each subnet is **/26** OR **255.255.255.192**

6 bit left for host

Subnetting (Example): To subnet a small network

Subnet **0**: → **11000000.10101000.00000001.00**000000

192.168.1.0**/26**

192.168.1.[	**00**	000000]	**subnet 0**
.[	**00**	111111]	
.[	**01**	000000]	**subnet 1**
.[	**01**	111111]	
.[	**10**	000000]	**subnet 2**
.[	**10**	111111]	
.[	**11**	000000]	**subnet 3**
.[	**11**	111111]	

Subnetting (Example): To subnet a small network

Subnet **0**: → **11000000.10101000.00000001.00**000000

192.168.1.0**/26**

192.168.1.[**00** | **000000**] **subnet 0**; network address = 192.168.1.**0**

.[00 | 111111]

.[**01** | **000000**] **subnet 1**; network address = 192.168.1.**64**

.[01 | 111111]

.[**10** | **000000**] **subnet 2**; network address = 192.168.1.**128**

.[10 | 111111]

.[**11** | **000000**] **subnet 3**; network address = 192.168.1.**192**

.[11 | 111111]

Subnetting (Example): To subnet a small network

Subnet **0**: → **11000000.10101000.00000001.00**000000

192.168.1.0**/26**

192.168.1.[00 | 000000]
.[**00** | **111111**] **subnet 0**; broadcast address = 192.168.1.**63**

.[01 | 000000]
.[**01** | **111111**] **subnet 1**; broadcast address = 192.168.1.**127**

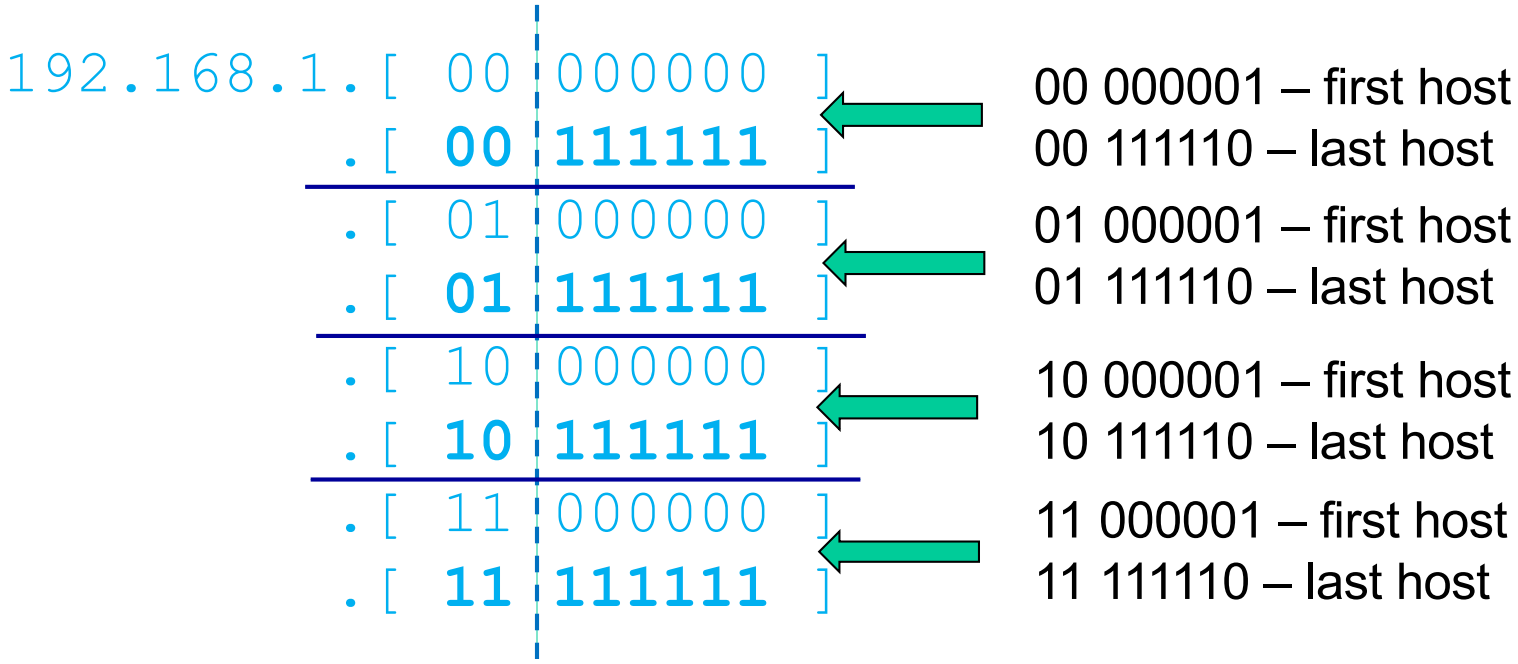
.[10 | 000000]
.[**10** | **111111**] **subnet 2**; broadcast address = 192.168.1.**191**

.[11 | 000000]
.[**11** | **111111**] **subnet 3**; broadcast address = 192.168.1.**255**

Subnetting (Example): To subnet a small network

Subnet **0**: → **11000000.10101000.00000001.00**000000

192.168.1.0**/26**



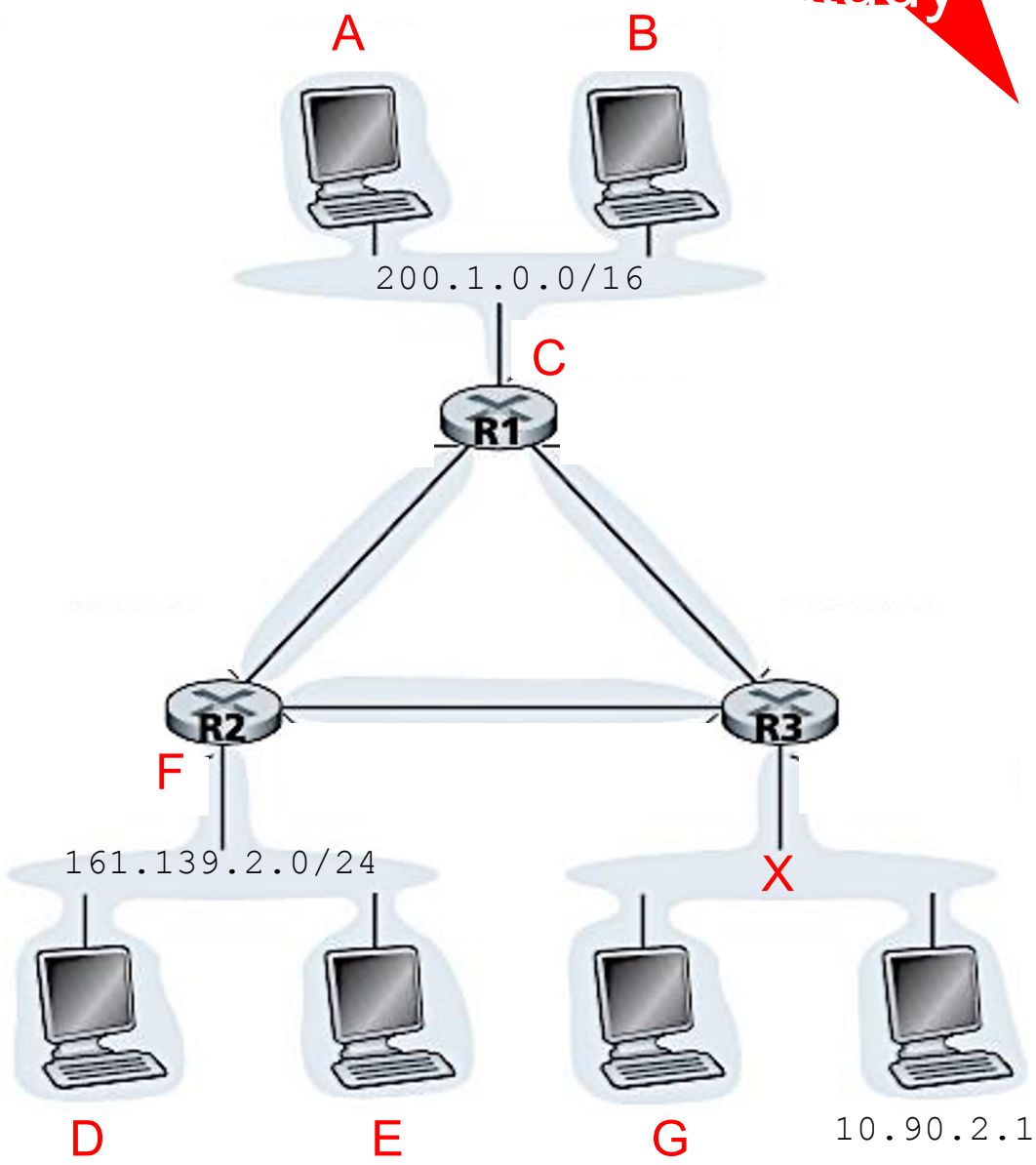
Subnetting (Example): To subnet a small network

192.168.1.0/24 be divided into 4 subnets we get:

Subnet	Department	Network address	Broadcast address	Range of usable address	Custom subnet mask (slash format and dotted decimal format)
0	SALES	192.168.1.0	192.168.1.63	192.168.1.1 to 192.168.1.62	/26 255.255.255.192
1	HR	192.168.1.64	192.168.1.127	192.168.1.65 to 192.168.1.126	/26 255.255.255.192
2	FINANCE	192.168.1.128	192.168.1.191	192.168.1.129 to 192.168.1.190	/26 255.255.255.192
3	RND	192.168.1.192	192.168.1.255	192.168.1.193 to 192.168.1.254	/26 255.255.255.192

Self study

Exercise



Given a network with hosts and interconnecting routers.

(a) What are the subnet mask for the network that consists of host A, B, D and E?

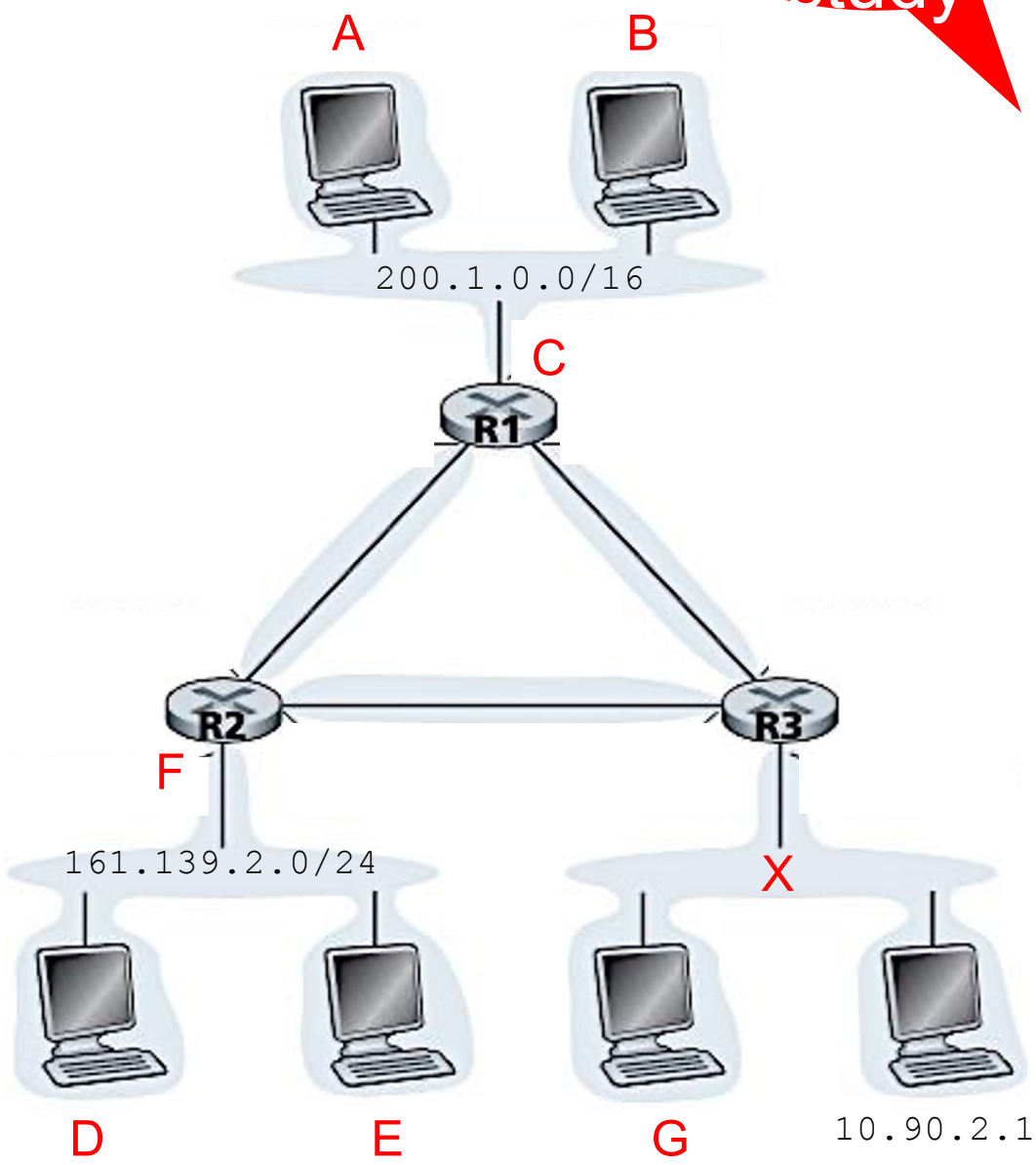
(b) What is the subnet network address of X if the subnet mask is 255.0.0.0?

(c) Get first valid usable IP address for host A, B, D, E and G.

(d) Get last valid usable IP address for each router's interface at C and F.

Self study

Solution



(a) Net. consists of host A, B:

255.255.0.0

Net. consists of host D, E:

255.255.255.0

(b)

```
  10.90.2.1
255. 0.0.0
----------
 10. 0.0.0
```

(c) First valid usable IP address:

A: 200.1.0.1
B: 200.1.0.2
D: 161.139.2.1
E: 161.139.2.2
G: 10.90.2.2

(d) Last valid usable IP address:

C: 200.1.255.254
F: 161.139.2.254

Self study

Exercise

Given an IP address as `200.23.0.0/20`. The network need to be divided into 5 subnets with first subnet label as Subnet 0 and last subnet label as Subnet 4.

(a) What is the original subnet mask for the given IP address network?
(b) How many hosts can be supported in the original IP address network?
(c) List all new subnet address in binary and dotted decimal format.
(d) What is the new subnet mask for each subnet?
(e) How many hosts can be supported for each new subnet?
(f) List the first 5 valid IP addresses for the Subnet 3.
(g) How many subnet remain unused and can be utilized in future?

Self study

Solution

Solutions:

(a) 11111111.11111111.11110000.00000000
255.255.240.0

(b) Total original hosts → $2^{12} - 2 = 4096 - 2 = 4094$ hosts (12 bit for host portion)

(c) Number of Subnets Needed = 5 → nearest $2^3 = 8$ subnets; 3 bits for subnet;

```
11001000.00010111.00000000.00000000  (200.23.0.0/23)
11001000.00010111.00000010.00000000  (200.23.2.0/23)
11001000.00010111.00000100.00000000  (200.23.4.0/23)
11001000.00010111.00000110.00000000  (200.23.6.0/23)
11001000.00010111.00001000.00000000  (200.23.8.0/23)
11001000.00010111.00001010.00000000  (200.23.10.0/23)
11001000.00010111.00001100.00000000  (200.23.12.0/23)
11001000.00010111.00001110.00000000  (200.23.14.0/23)
```

Self study

Solution

Solutions:

(d) 11111111.11111111.11111110.00000000
255.255.254.0

(e) Total hosts in new subnet → 2^9 = 512 – 2 = 510 hosts (9 bits for host portion)

(f) Subnet 3 = 200.23.6.0/23

```
11001000.00010111.0000011 0.00000001   (200.23.6.1/23)
11001000.00010111.0000011 0.00000010   (200.23.6.2/23)
11001000.00010111.0000011 0.00000011   (200.23.6.3/23)
11001000.00010111.0000011 0.00000100   (200.23.6.4/23)
11001000.00010111.0000011 0.00000101   (200.23.6.5/23)
```

(g) Need 5 subnets, generated subnets = 8, extra = 8 – 5 = 3 subnets for future use

Extra Exercise: Subnet small network

- Say, Company A (CA) is given the Network address 192.168.1.0/24
- CA has 4 departments under it: RND, Sales, Finance, HR
- Each department needs the following number of usable host: RND(50), Sales(30), Finance(20), HR(10)
- Task:
  - **Divide equally for all dept. (same/equal size)**

Network address 192.168.1.0/24
to divide into SAME/EQUAL SIZED 4 dept : RND, SALES, FIN, HR

STEP 1 : Divide the Network and host portion (HP)

192 . 168 . 1 . 0
11000000 . 10101000.00000001.00000000

NP = 24 bits

**when subnetting ONLY HOST PORTION is borrowed from (disturbed)

STEP 2 : Decide how many bits need to BORROW from HP?

4 depts. ➔ 4 subnet
So **$2^x = 4$ ➔ x = 2 bits**

Borrow 2 bits

Network address 192.168.1.0/24
to divide into SAME/EQUAL SIZED 4 dept : RND, SALES, FIN, HR

STEP 3 : Borrow 2 bits from the host portion

Borrow from this end – THE LEFTMOST

192 . 168 . 1 . 0
11000000 . 10101000.00000001.00000000

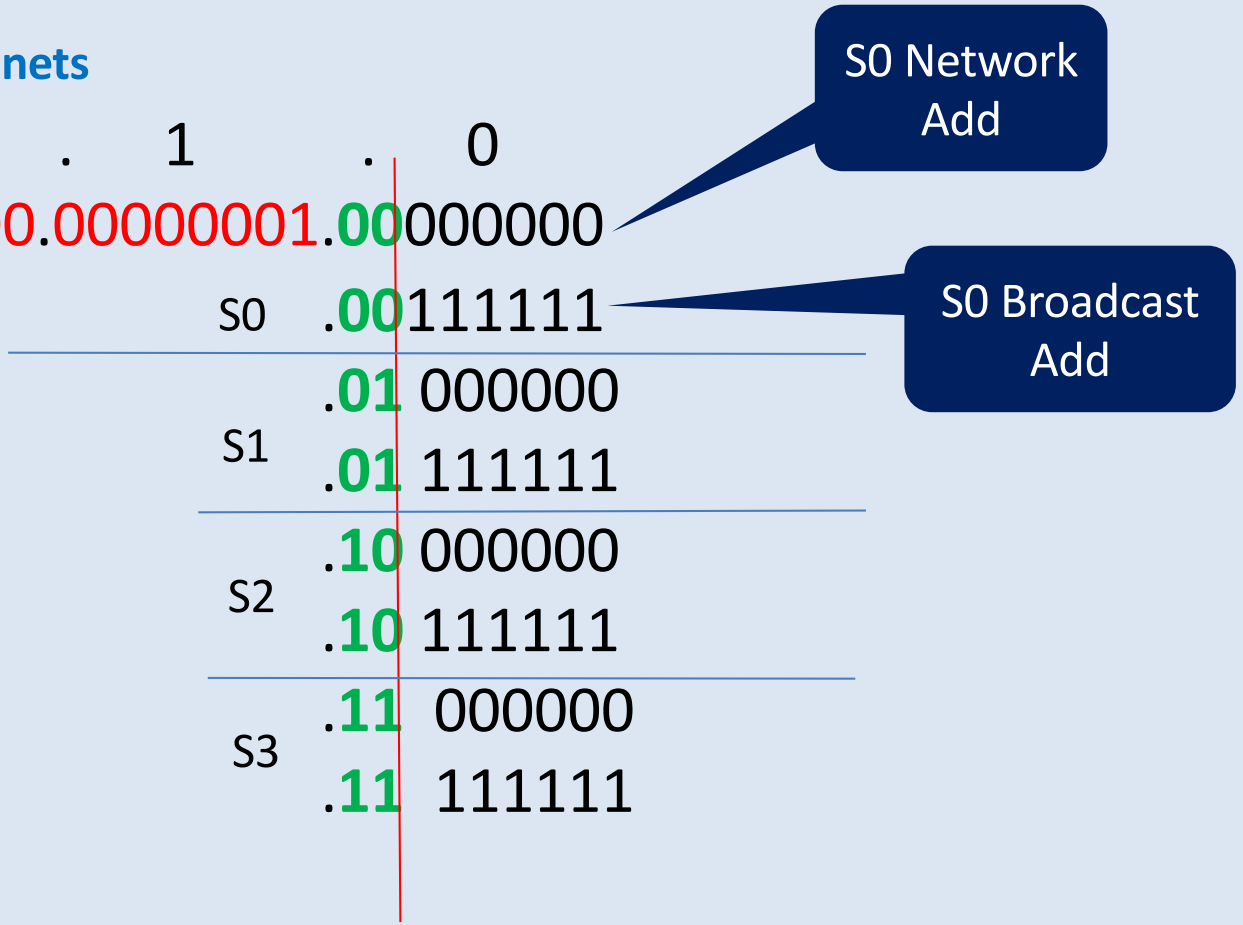
STEP 4 : Arrange the new subnets

Within the subnet , the HP gets all 0 and all 1

192 . 168 . 1 . 0
11000000 . 10101000.00000001.00000000 [0]
.00111111 [63]
.01000000 [64]
.01111111 [127]
.10000000 [128]

Network address 192.168.1.0/24
to divide into SAME/EQUAL SIZED 4 dept : RND, SALES, FIN, HR

STEP 4 : Arrange the new subnets

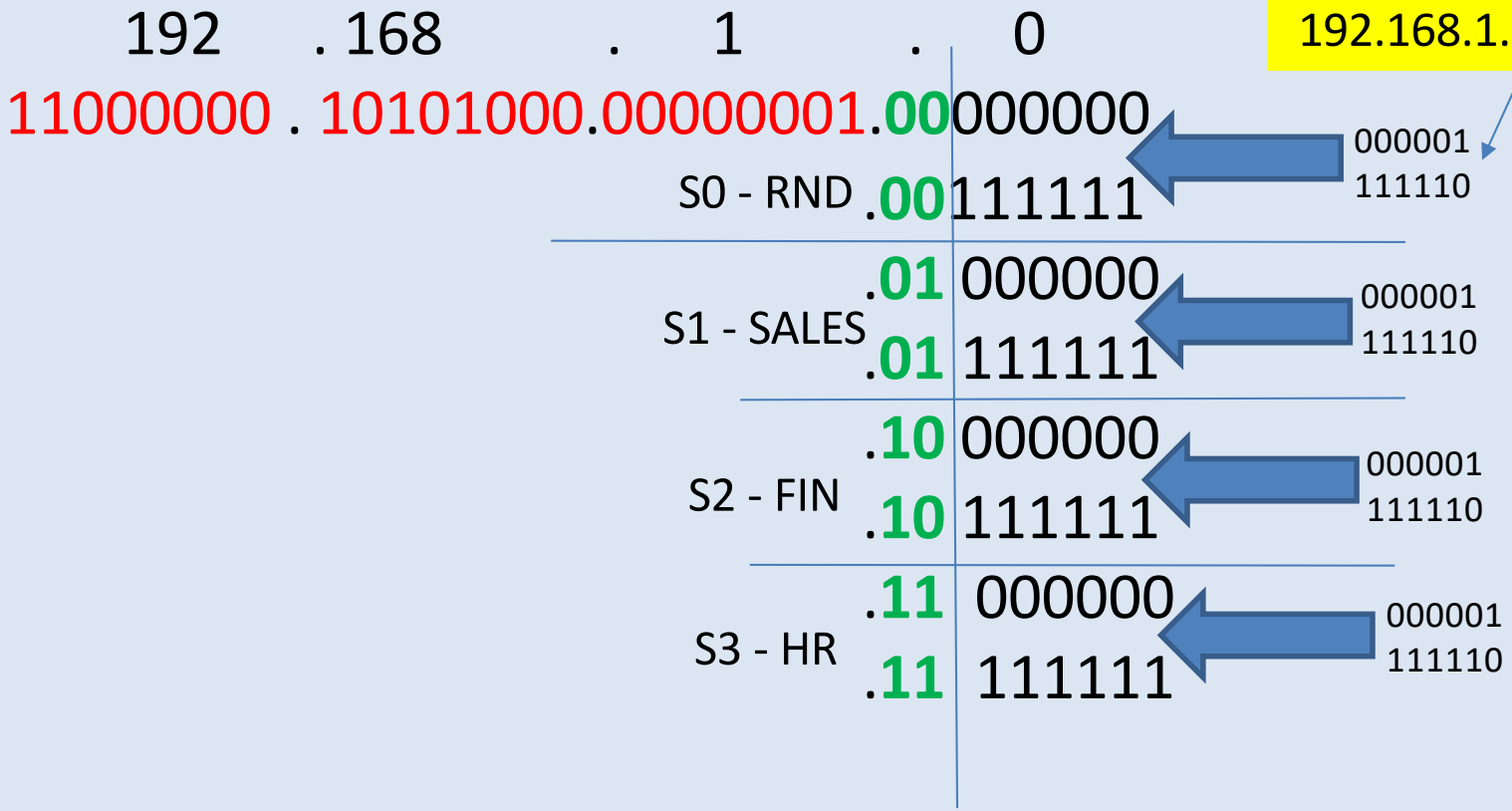


S0 (Subnet 0)
network address:
192.168.1.0
Broadcast address:
192.168.1.63

Network address 192.168.1.0/24
to divide into SAME/EQUAL SIZED 4 dept : RND, SALES, FIN, HR

S0 (Subnet 0)
Usable/valid IP address:
192.168.1.1 to
192.168.1.62

STEP 5 : Assign the new subnets



STEP 6 : Complete the subnetting table

SUBNET	NETWORK ADDRESS	USABLE/VALID IP ADD. RANGE	BROADCAST ADDRESS	SUBNET MASK
0	192.168.1.0	192.168.1.1 – 192.168.1.62	192.168.1.63	/26 OR 255.255.255.192
1	192.168.1.64	192.168.1.65 – 192.168.1.126	192.168.1.127	/26 OR 255.255.255.192
2	192.168.1.128	192.168.1.126 – 192.168.1.190	192.168.1.191	/26 OR 255.255.255.192
3	192.168.1.192	192.168.1.190 – 192.168.1.254	192.168.1.255	/26 OR 255.255.255.192

Convert calculation in table to binary format for complete understanding

Obtaining a block of addresses (network address)

Q: How does a network get IP address block for itself (network part of address)

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

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

ICANN allocates addresses to 5 regional Internet registries (RRs)

- like ARIN, RIPE, APNIC, and LACNIC
- …which handle the allocation/management of addresses within their regions.

Obtaining a host addresses

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

A: local **ISP** (e.g. Telekom, Celcom, DiGi, Maxis) & local organization like UTM:

DHCP: Dynamic Host Configuration Protocol: dynamically get address from server. “plug-and-play” concept

OR

hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)

CHAPTER 4

Dynamic Host Configuration Protocol (DHCP)

- In an organization, IP address can be assigned to host

  Statically – User manually configure IP address at host (normally done by 'network admin')

  Dynamically - done using **DHCP**

- DHCP can be configured to assign

  the same IP address to a given host each time it connects to the network

  OR

  different IP address each time the host connects to the network.

CHAPTER 4

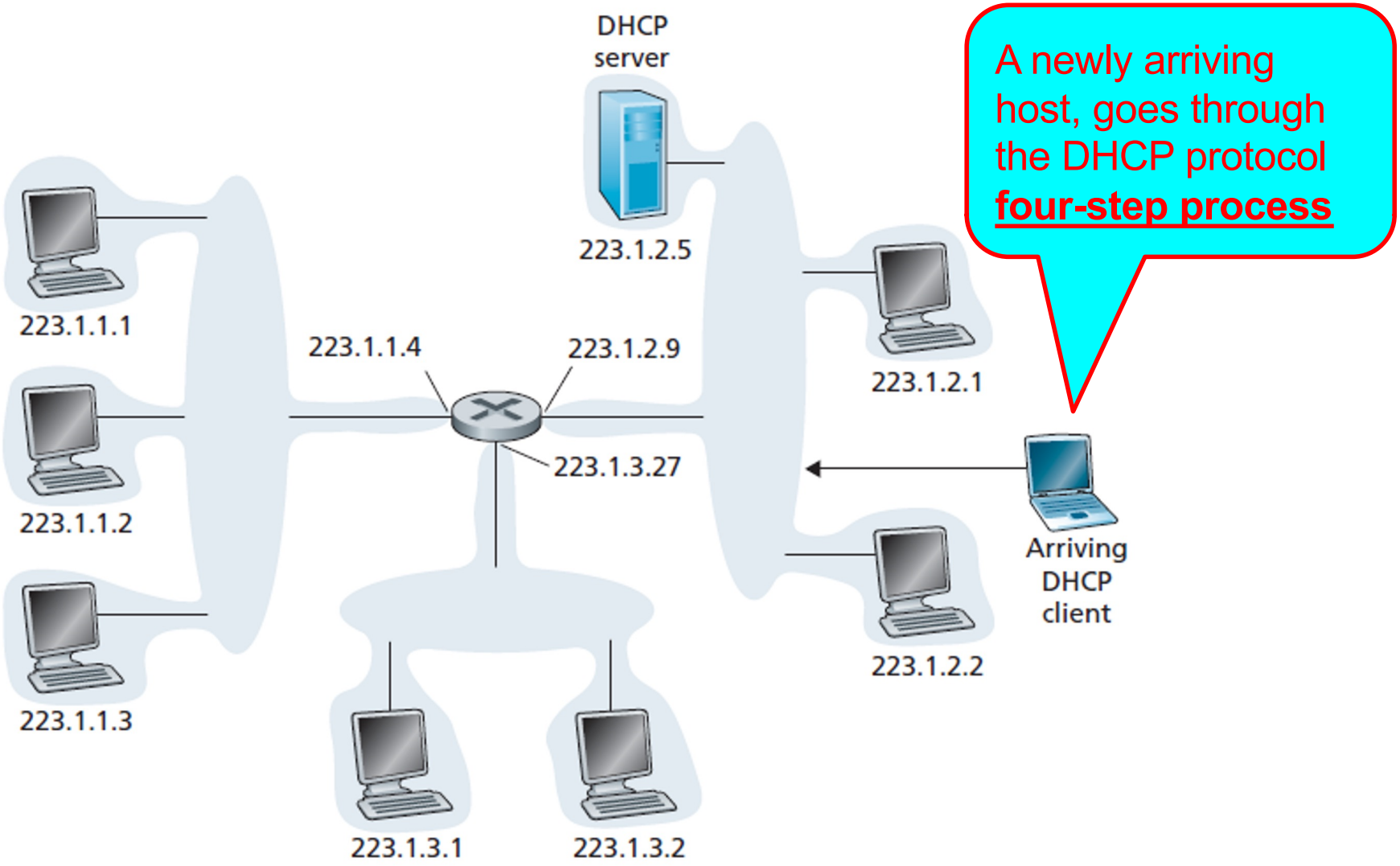
Dynamic Host Configuration Protocol (DHCP)

- *Goal:* allow host to *dynamically* obtain its IP address from DHCP server when it joins network
  - host will *lease* an IP address, 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/leave network

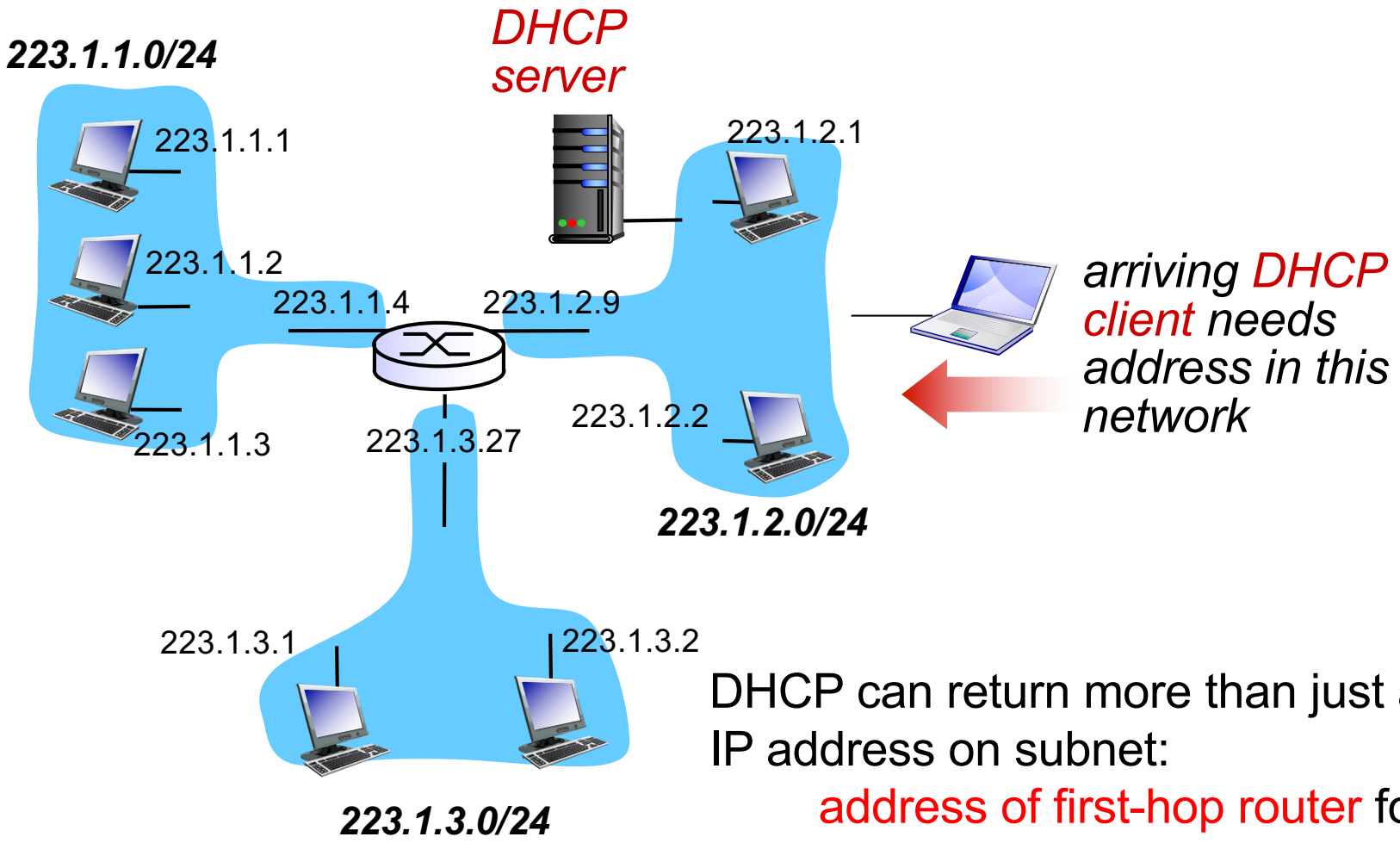
- DHCP is a plug-and-play protocol
  - Makes network admin job easier
  - Makes network mobility easier too

CHAPTER 4

Dynamic Host Configuration Protocol (DHCP)



DHCP client-server scenario



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 – 4 Steps Process

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 4 steps process
  - *DHCP server discovery*
  - *DHCP server offer*
  - *DHCP request*
  - *DHCP ACK*

DHCP – 4 Steps Process

- *DHCP server discovery*

  newly arriving host needs to find a DHCP server

  → done by broadcasting the DHCP discover message

  - source: this host 0.0.0.0
  - dest: 255.255.255.255 (i.e. broadcast to all)
  - UDP: Port 67

- *DHCP server offer*

  DHCP server(s) will respond to the client with a DHCP offer message that is broadcast to all

  A client might receive offers from > 1 DHCP servers

DHCP – 4 Steps Process

- *DHCP request*

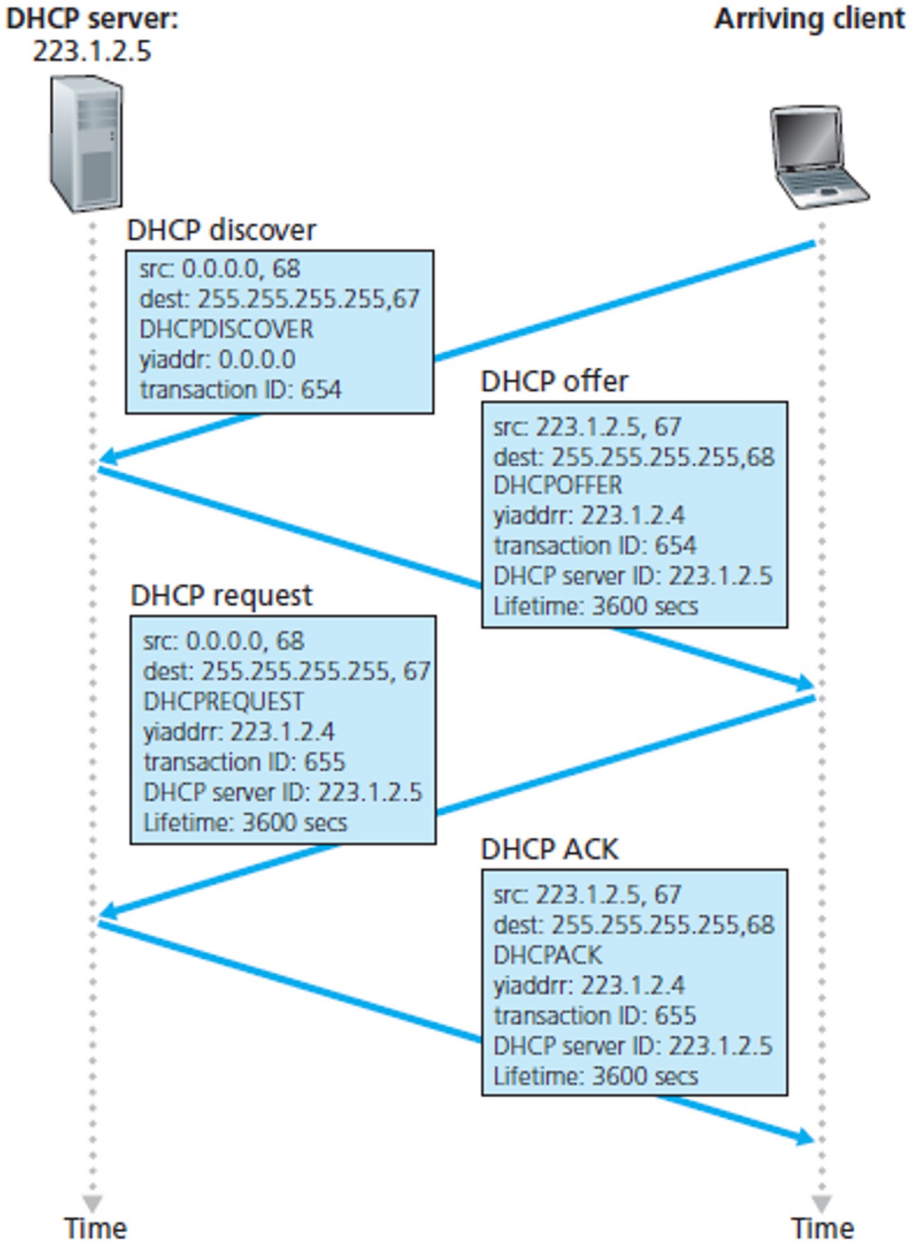
  Client choose from among the offers

  Respond to its selected offer with a DHCP request message.

- *DHCP ACK*

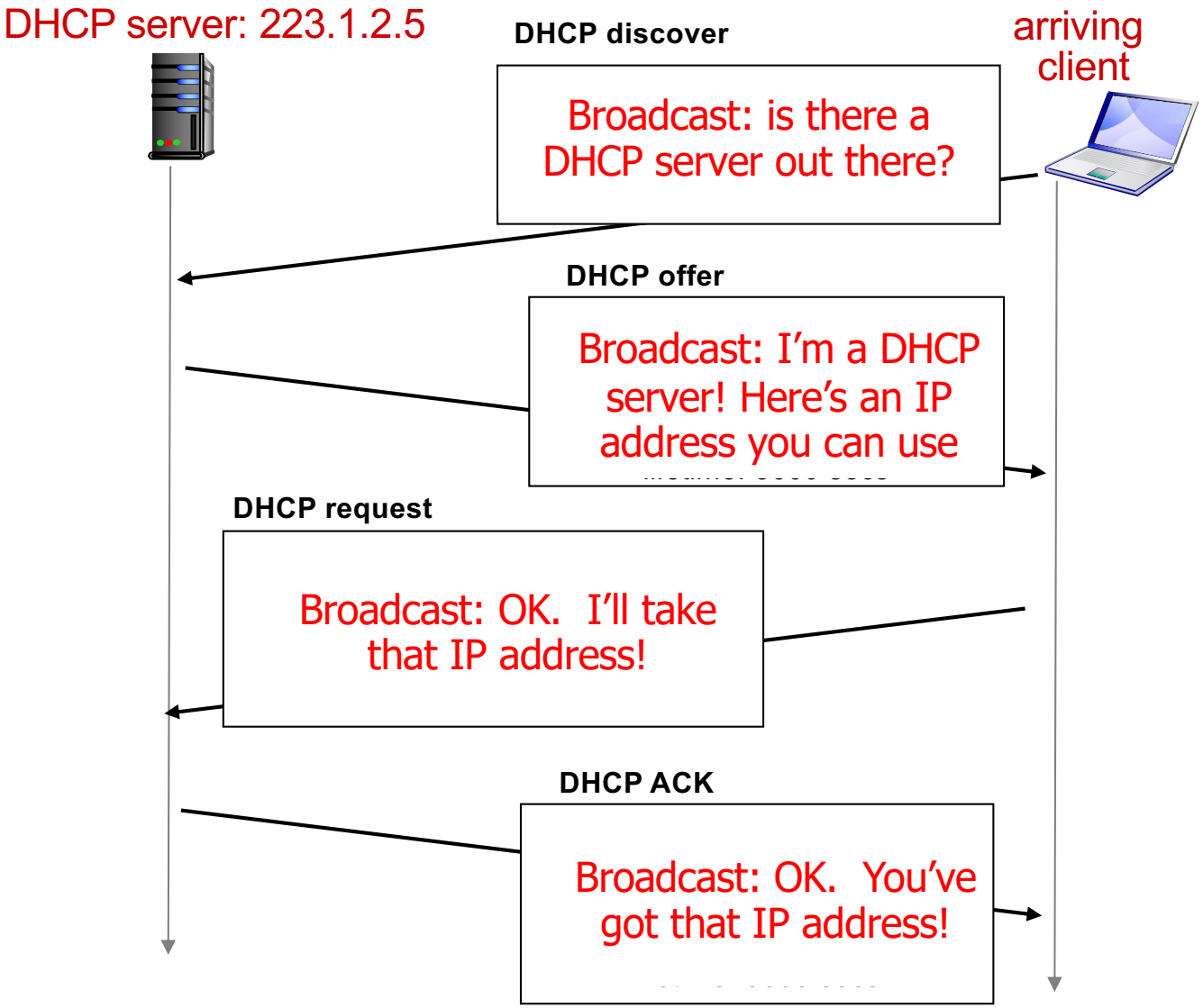
  The server responds to the DHCP request message with a DHCP ACK message, confirming the requested parameters.

DHCP – 4 Steps Process

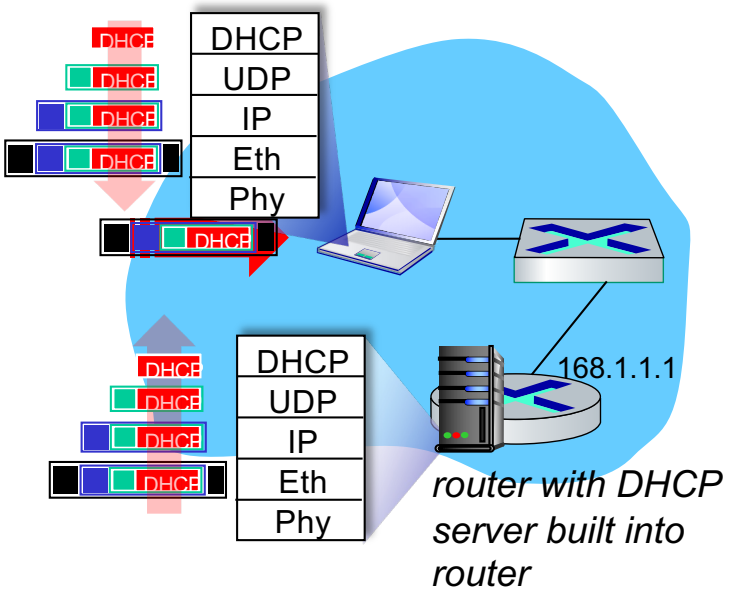


yiaddr = your IP address

DHCP client-server scenario

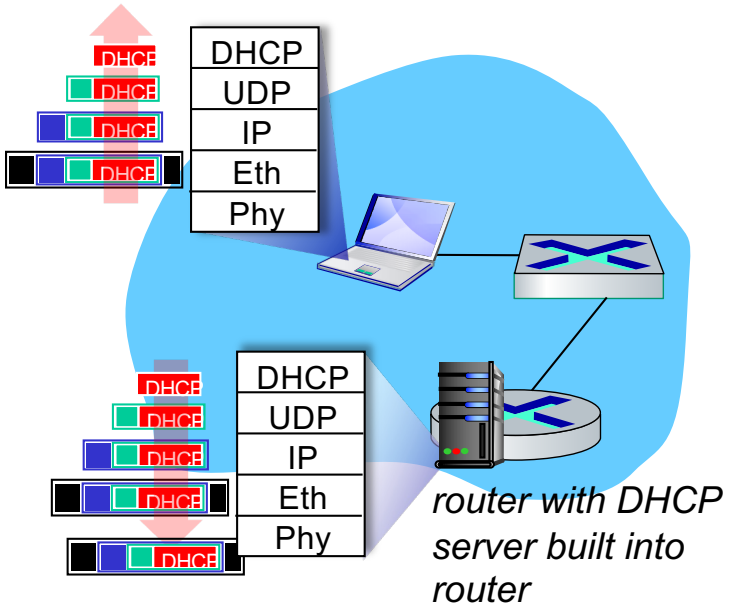


DHCP: example



- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply 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: Example Wireshark output (home LAN)

request

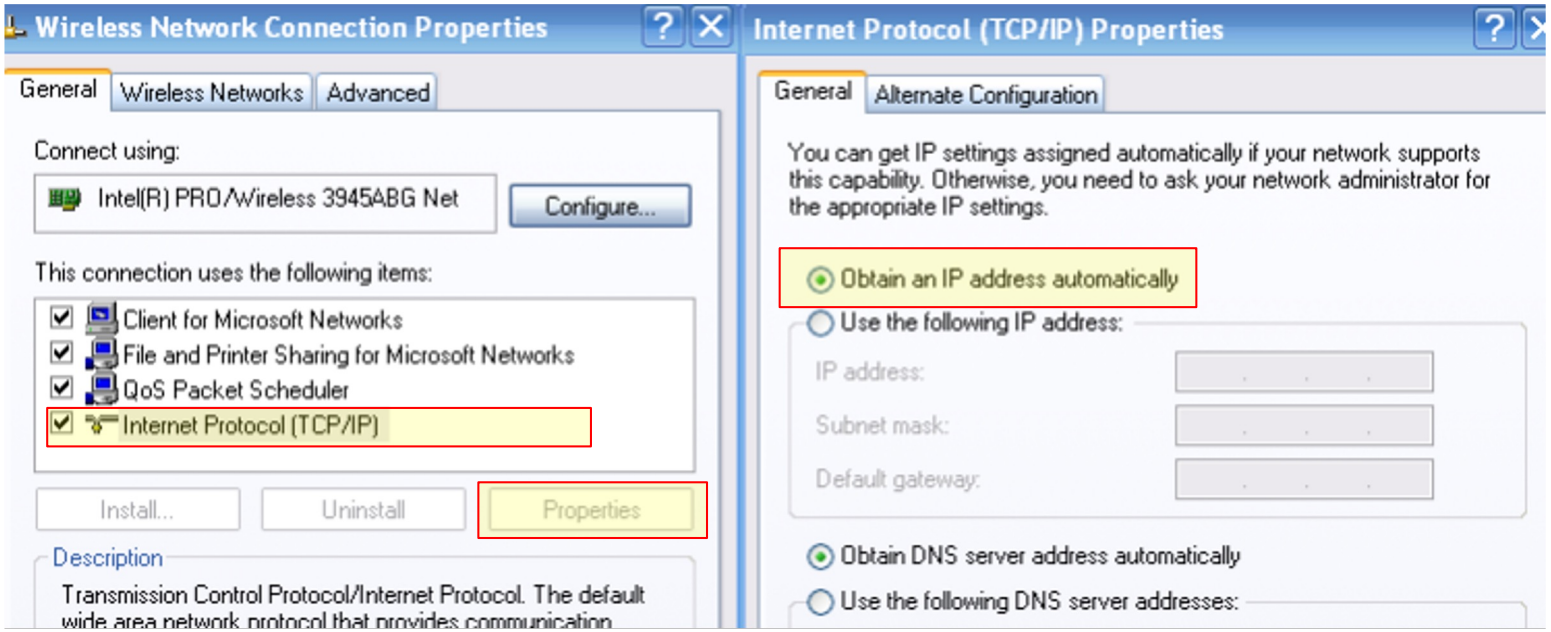
Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent 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
    ……

reply

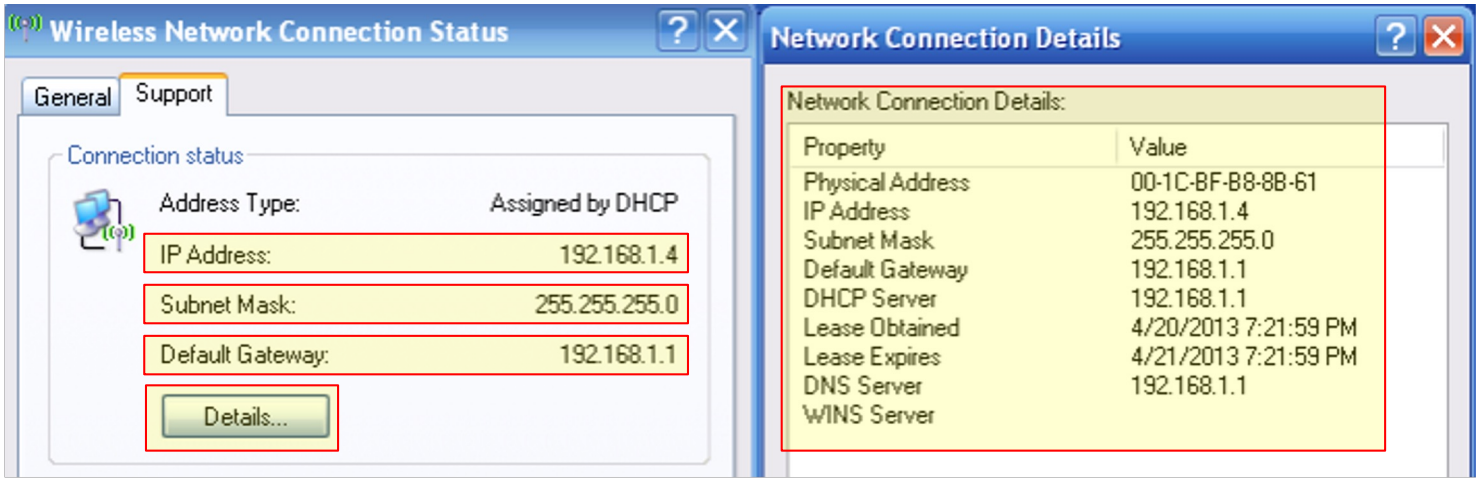
Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
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."

DHCP: Example Assignment (home LAN)

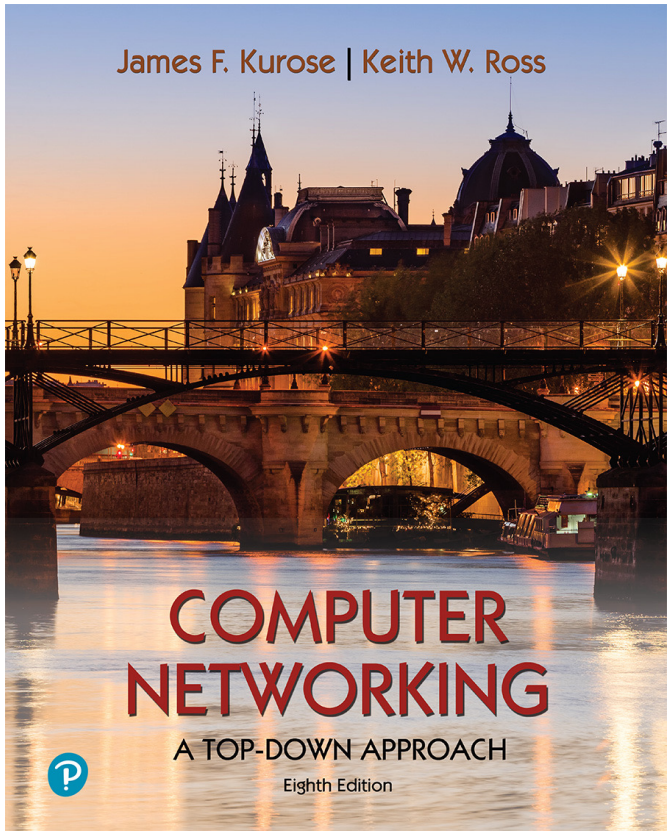
TCP/IP Client Setting



DHCP Server Assignment



Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

input ports, switching, output ports

buffer management, scheduling

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

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

NAT: network address translation

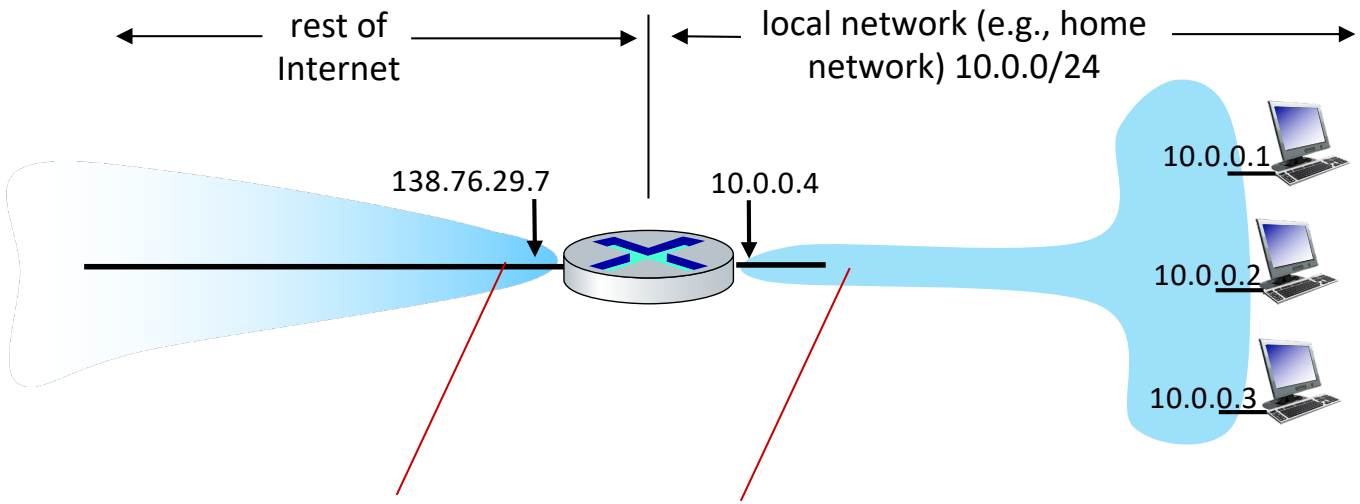
Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

NAT: network address translation

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



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

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

NAT: network address translation

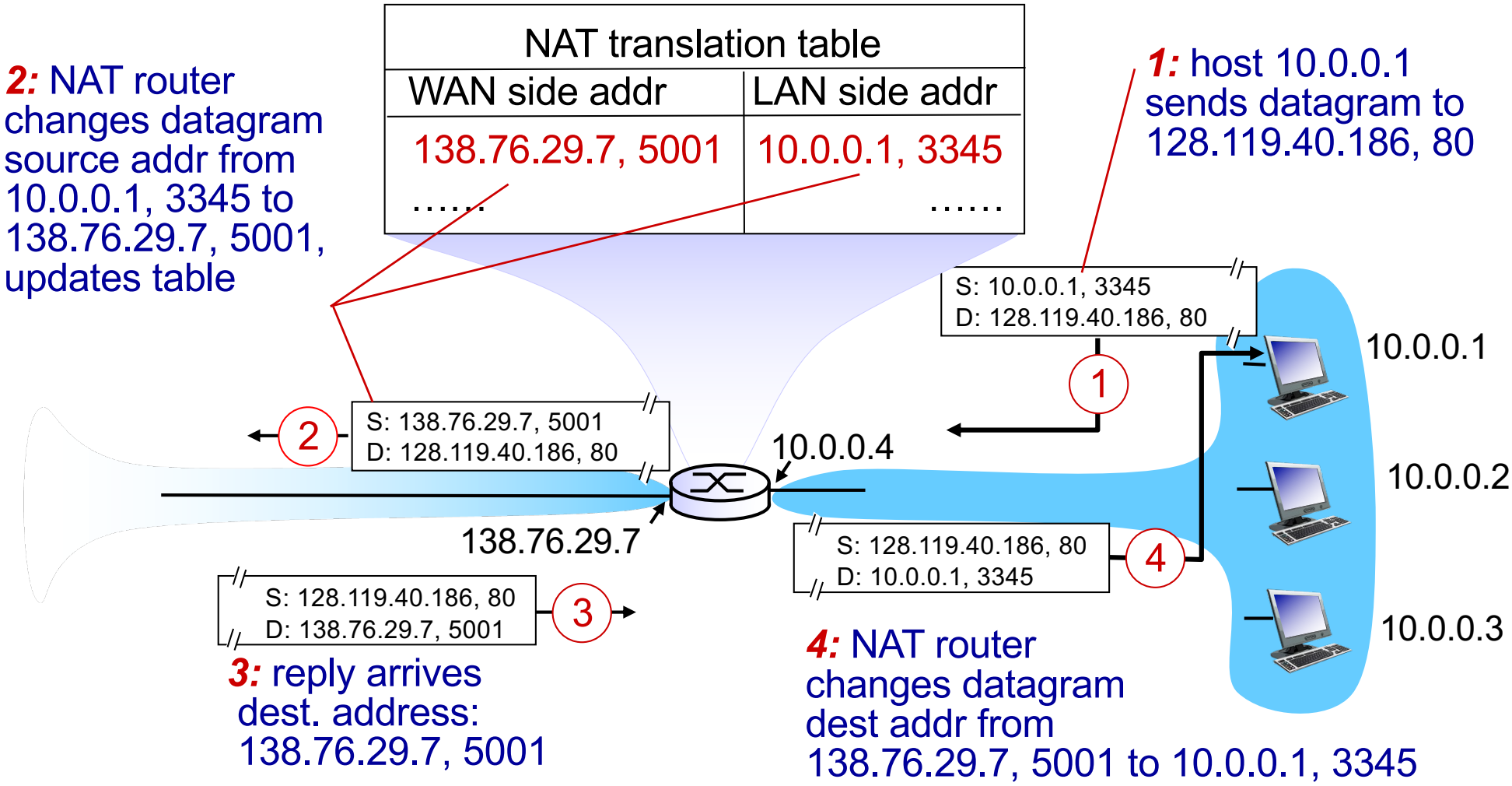
- all devices in local network have 32-bit addresses in a "private" IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  - just one IP address needed from provider ISP for *all* devices
  - can change addresses of host in local network without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

NAT: network address translation

implementation: NAT router must (transparently):

- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

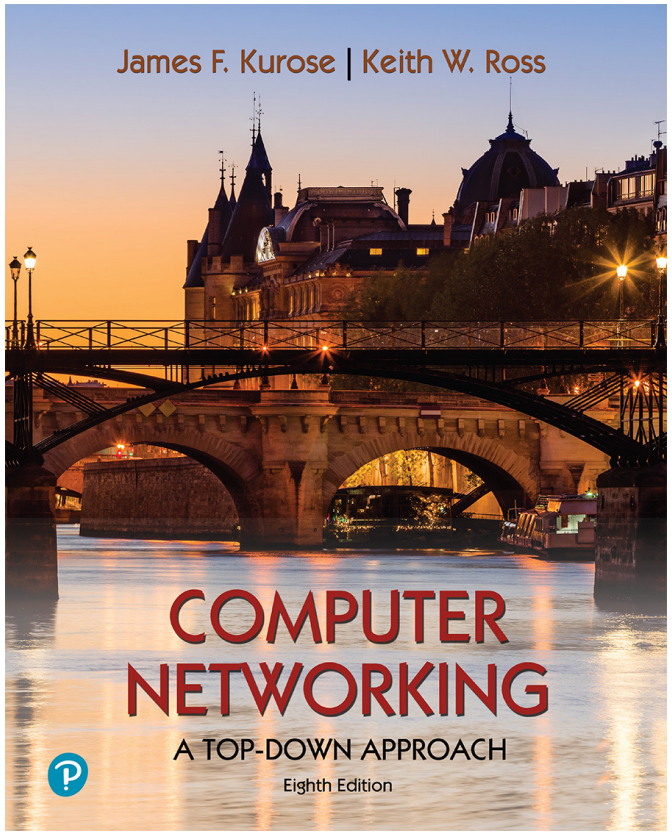


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

NAT: network address translation

- NAT has been controversial:
  - routers “should” only process up to layer 3
  - address “shortage” should be solved by IPv6
  - violates end-to-end argument (port # manipulation by network-layer device)
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G cellular nets

Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

input ports, switching, output ports

buffer management, scheduling

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

4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

IPv6: motivation

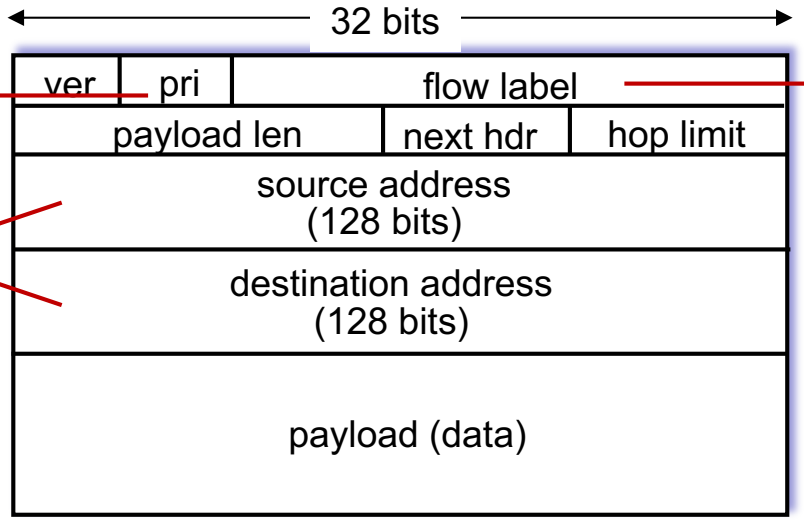
- *initial motivation:* 32-bit IPv4 address space soon to be completely allocated. IPv6 provided a larger IP address space → more IP address

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

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



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

IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label:

- labeling of packets belonging to particular flows for which the sender requests special handling
  - Example: real-time service, media streaming
  - But the real definition of flow is still vague (unclear)

next header: identify upper layer protocol for data

Traffic class:

used to give priority to datagrams

certain datagrams within a flow

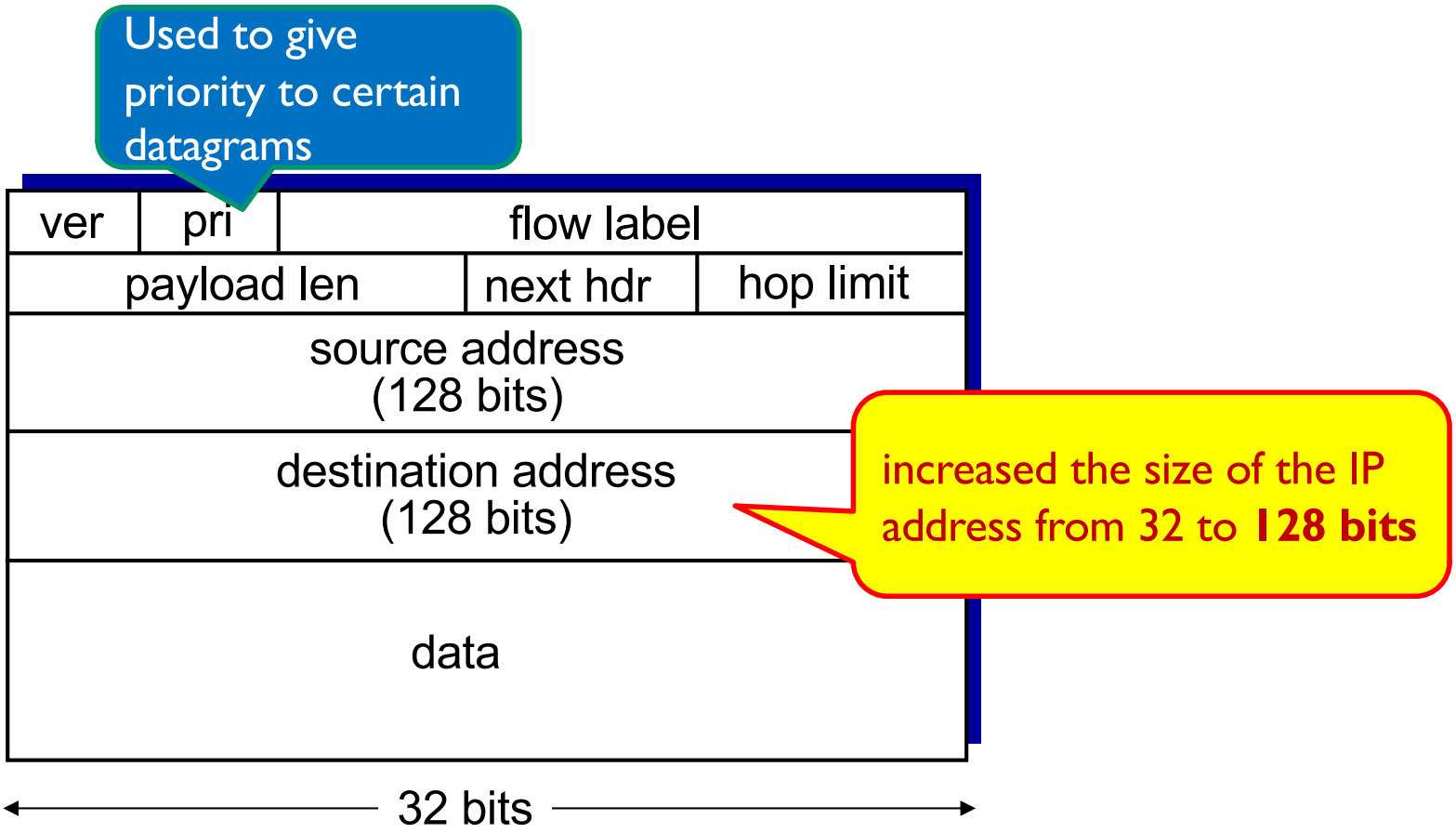
datagrams from certain applications (for example, ICMP) over others

Payload length:

What size of data sent (not including header)

If payload is 4bytes → total datagram size = 40 + 4 = 44 bytes

IPv6 datagram format

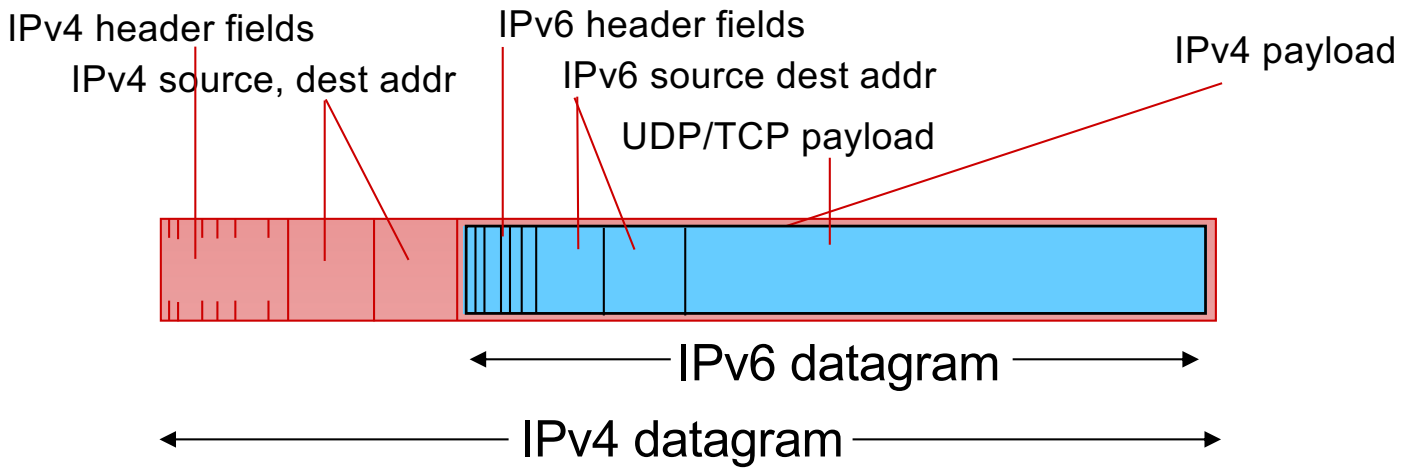


IPv6 datagram format and other changes from IPv4

- IPv6 datagram format:
  - Header : 40-byte fixed-length header ➔ allows faster processing of the IP datagram
  - Fragmentation: **NO** fragmentation allowed
  - *options:* allowed, but outside of header, indicated by "Next Header" field
  - Checksum: **NO** checksum at Network Layer for faster processing at each hop → use checksum on Transport Layer
  - *ICMPv6:* new version of ICMP
    - additional message types, e.g. "Packet Too Big"
    - multicast group management functions

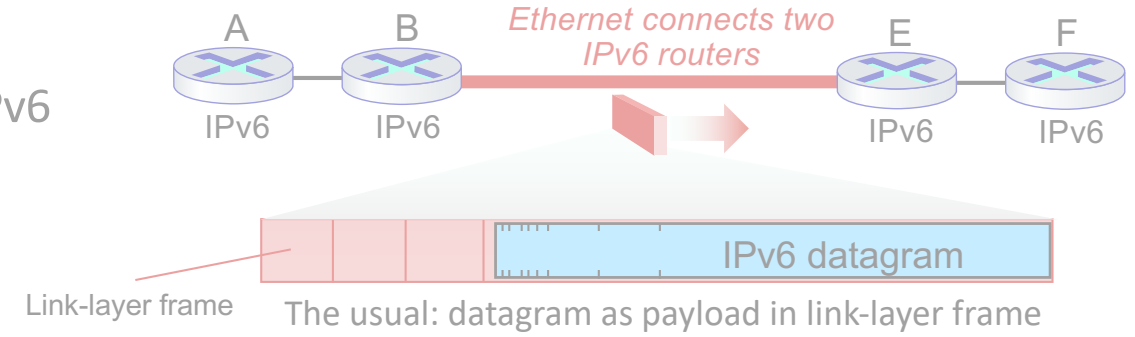
Transition from IPv4 to IPv6

- 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 ("packet within a packet")

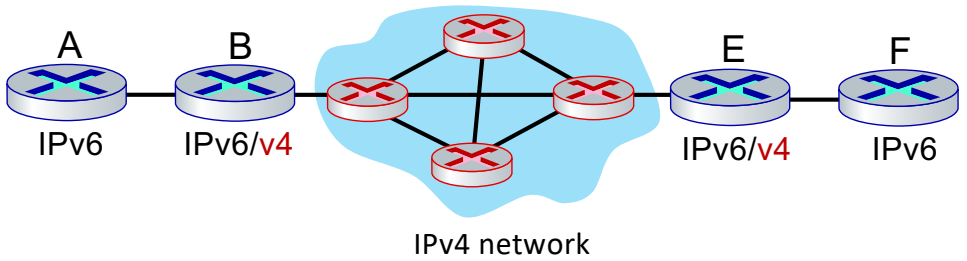


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

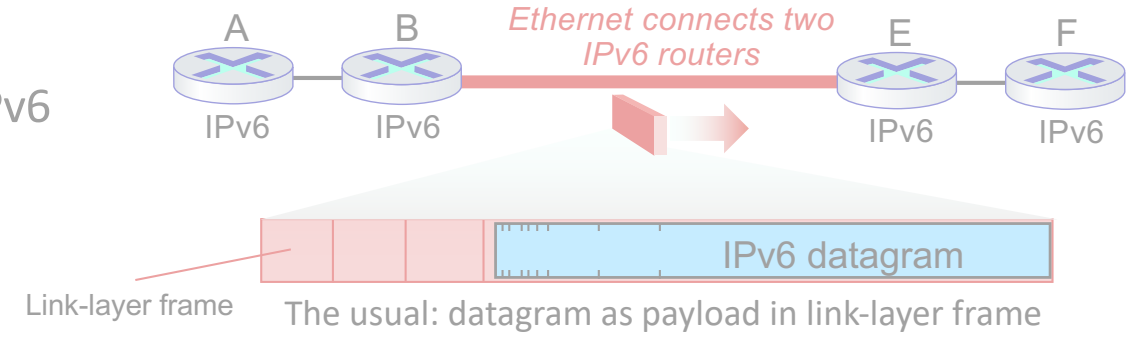


IPv4 network connecting two IPv6 routers

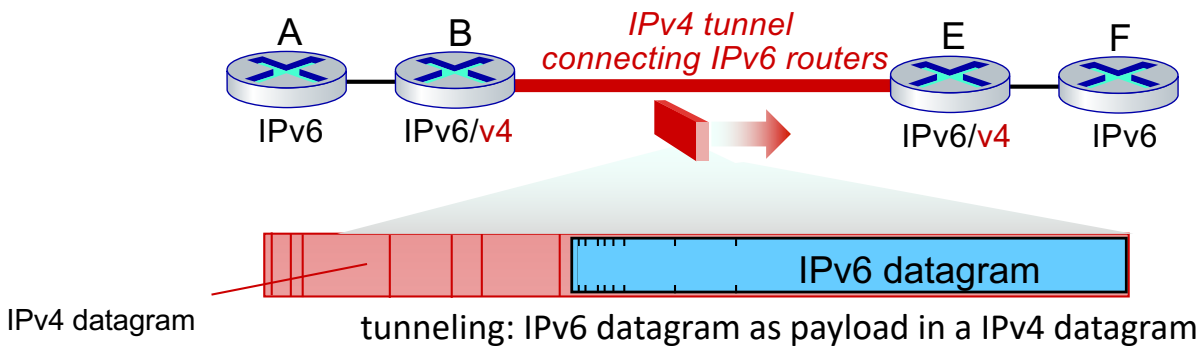


Tunneling and encapsulation

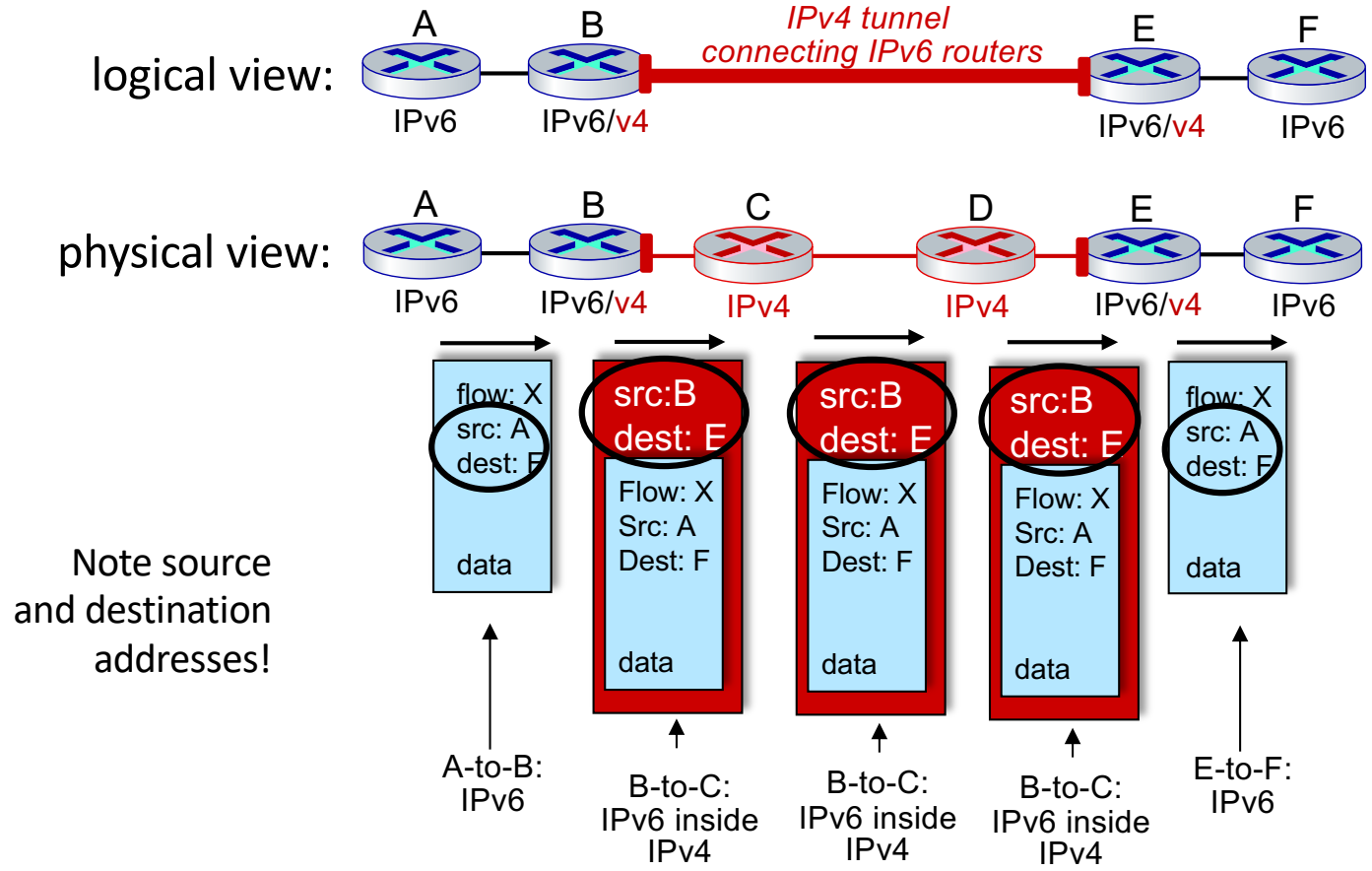
Ethernet connecting two IPv6 routers:



IPv4 tunnel connecting two IPv6 routers

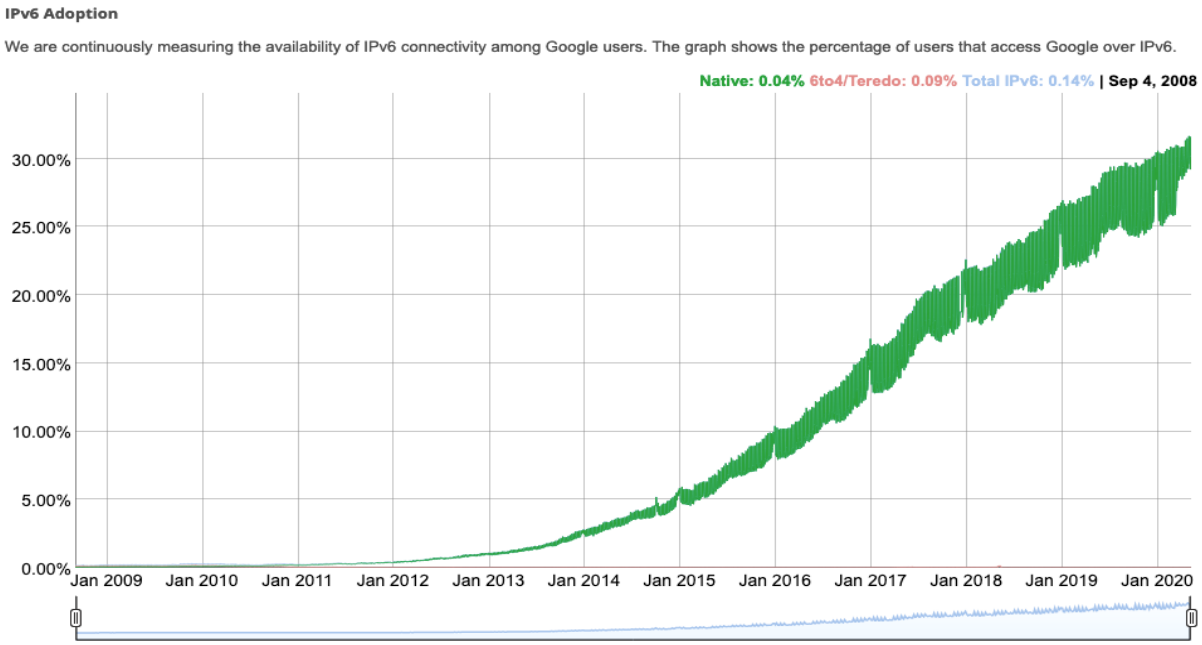


Tunneling


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: F
Flow: X
Src: A
Dest: F
data
src:B
dest: F
Flow: X
Src: A
Dest: F
data
src:B
dest: F
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
Note source
and destination
addresses!
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable



1

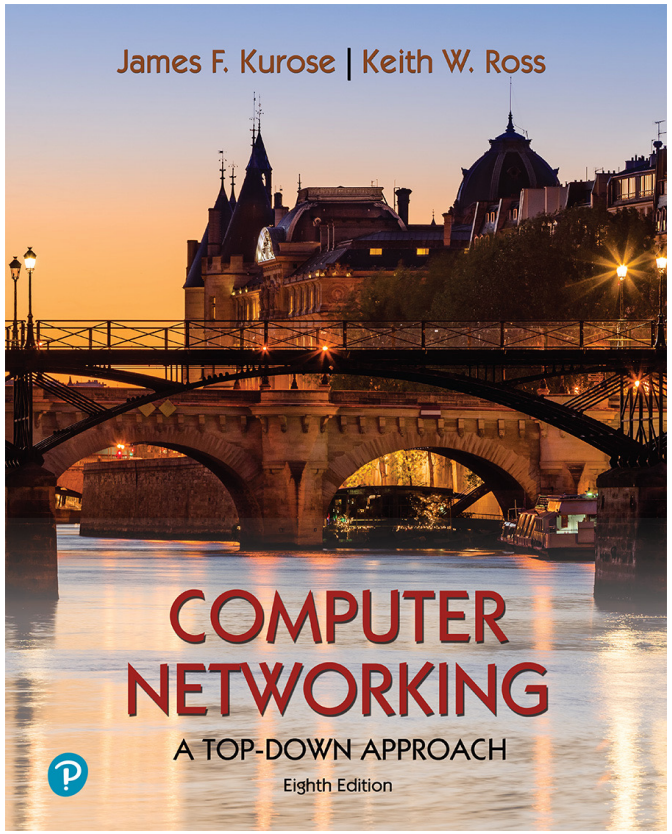
https://www.google.com/intl/en/ipv6/statistics.html

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
  - 25 years and counting!
  - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

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

Chapter 4: data plane



4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

input ports, switching, output ports

buffer management, scheduling

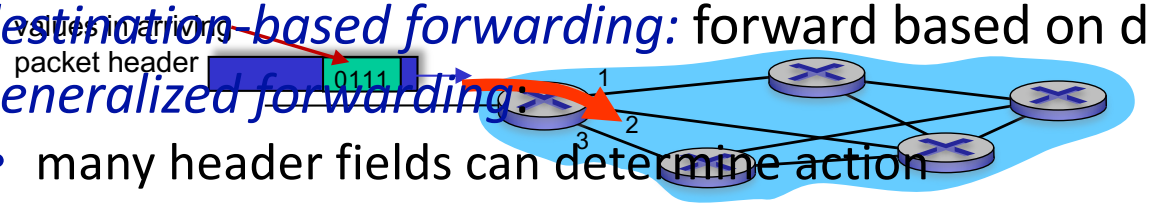
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

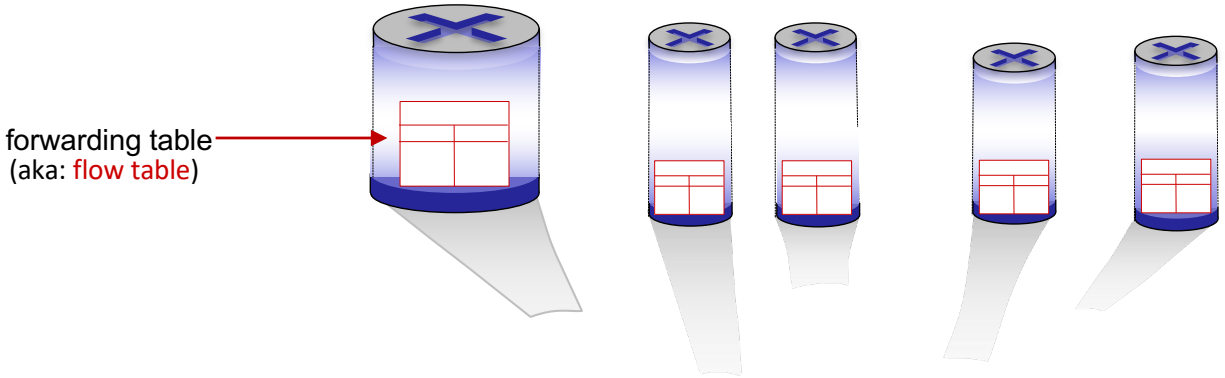
4.4 Generalized Forwarding and SDN
- Match+action
- OpenFlow: match+action in action

Generalized forwarding: match plus action

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

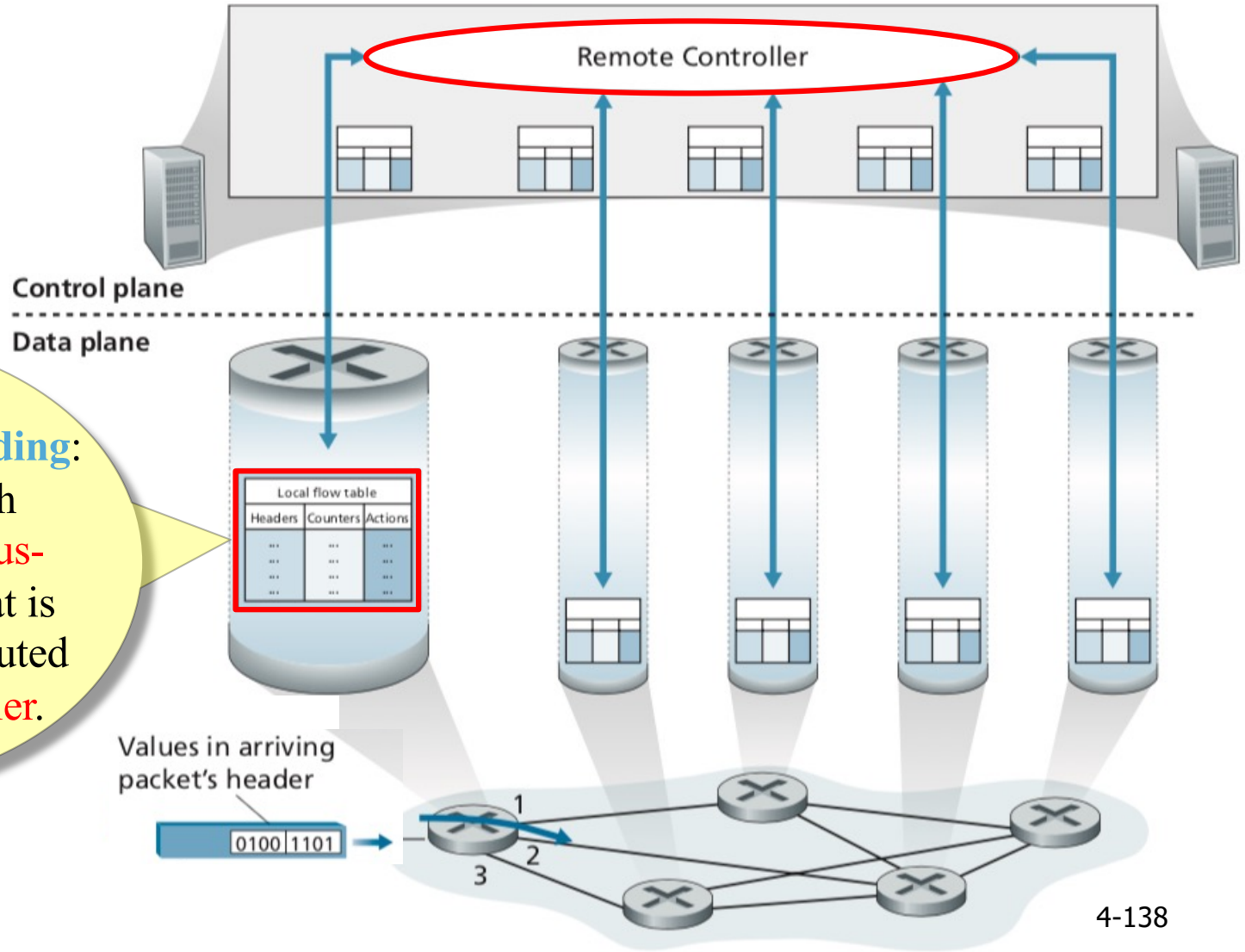
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet





Overview

Generalized Forwarding: Each packet switch contains a match-plus-action **flow table** that is computed and distributed by a remote controller.



Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 353)

Overview

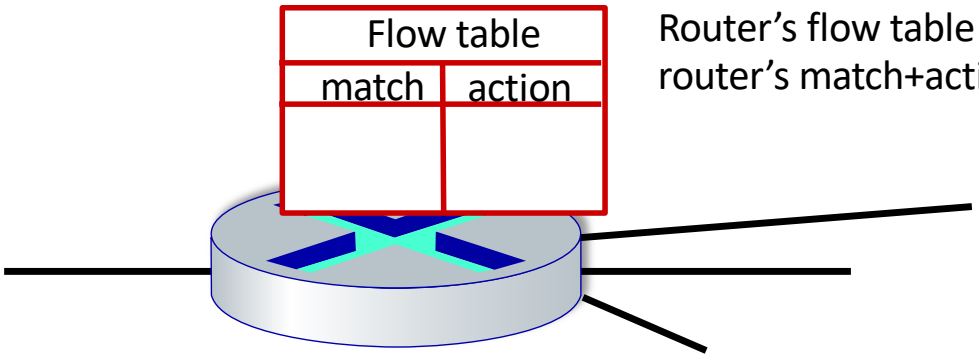
- Generalized forwarding will be based on a standard rule → Open-Flow.
- It is a highly visible standard that has pioneered the notion of the match-plus-action forwarding abstraction and controllers.

- *OpenFlow*1.0 is used, which introduced key SDN abstractions and functionality

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 353)

Flow table abstraction

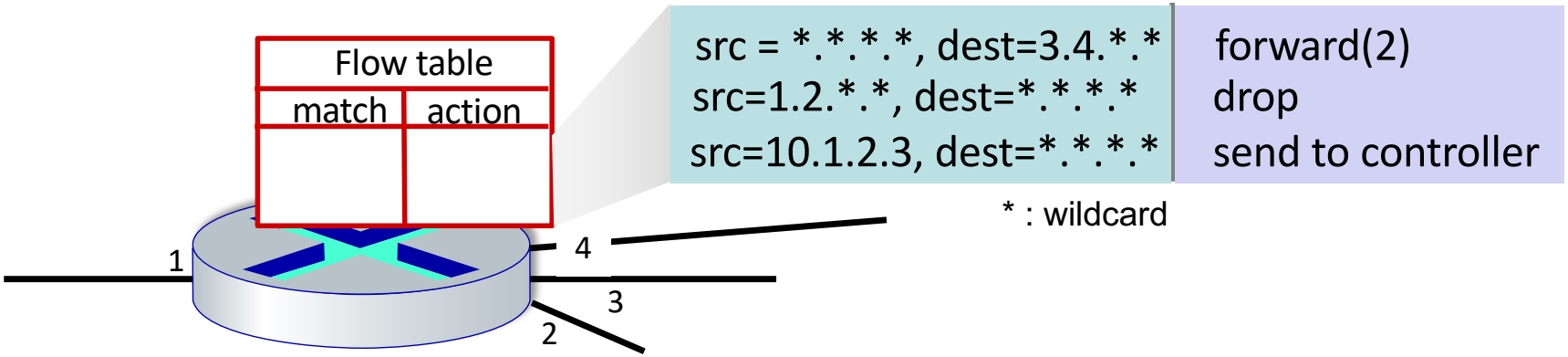
- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



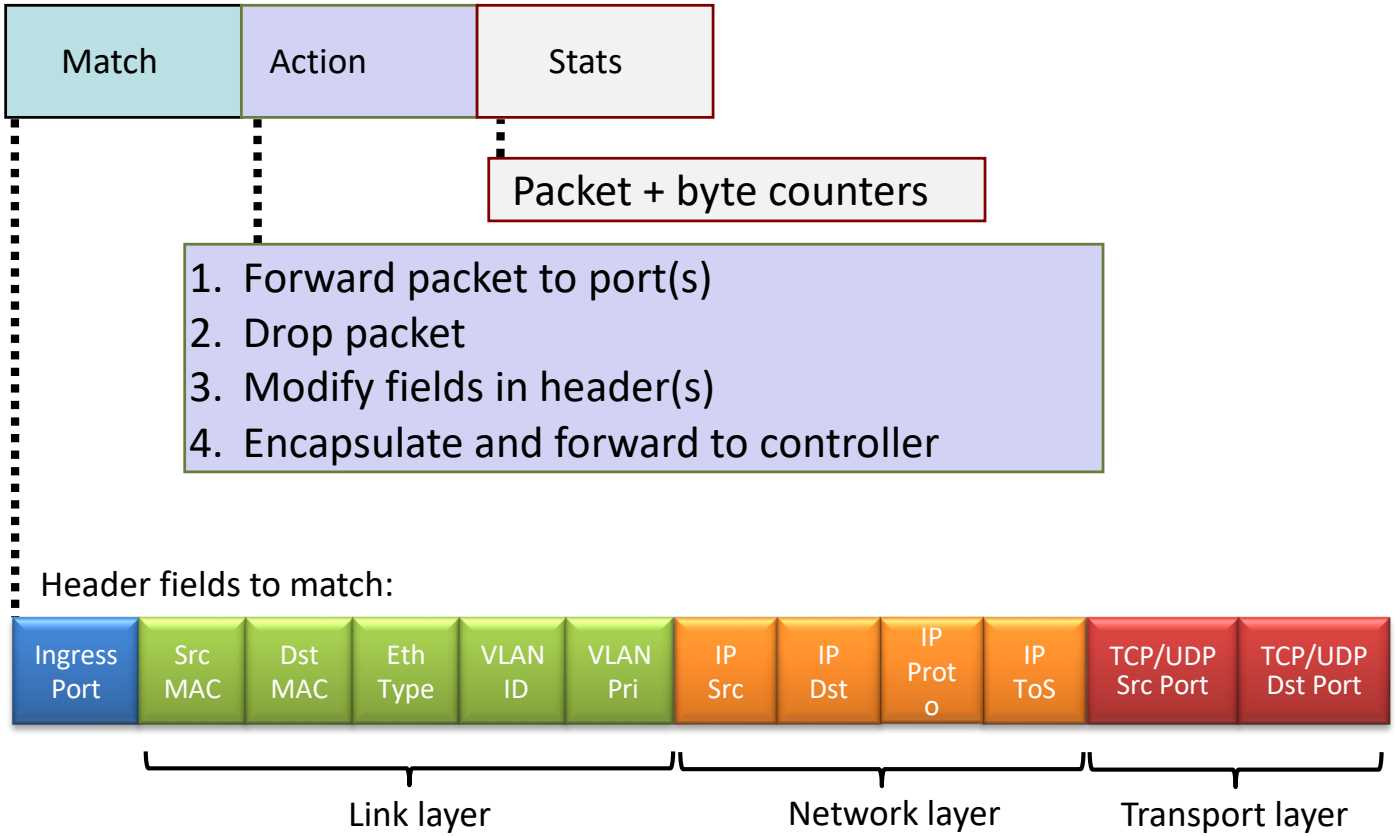
Router's flow table define router's match+action rules

Flow table abstraction

- flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - match: pattern 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



OpenFlow: flow table entries



OpenFlow: Example

Examples 1

Destination-based forwarding:

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

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

Destination-based layer 2 (switch) forwarding:

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	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: Example

Examples 2

Firewall:

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	*	*	*	*	*	22	Drop

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

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	Drop

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

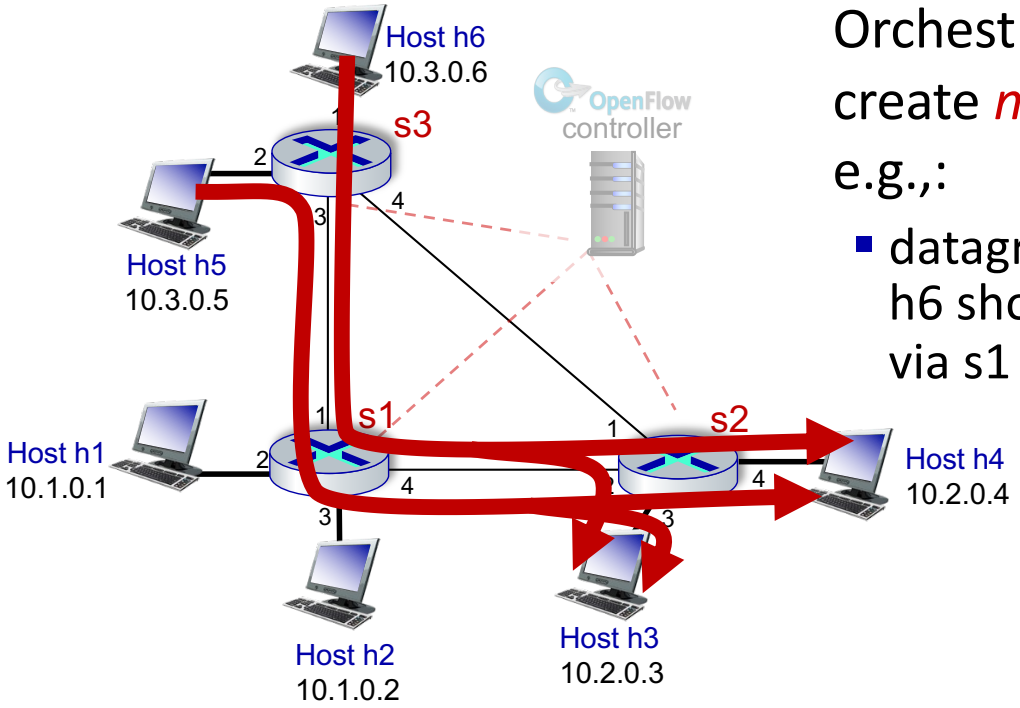
OpenFlow: Abstraction

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

▪ Router • *match:* longest destination IP prefix • *action:* forward out a link	▪ Firewall • *match*: IP addresses and TCP/UDP port numbers • *action:* permit or deny
▪ Switch • *match:* destination MAC address • *action:* forward or flood	▪ NAT • *match:* IP address and port • *action:* rewrite address and port

OpenFlow example

Examples 3



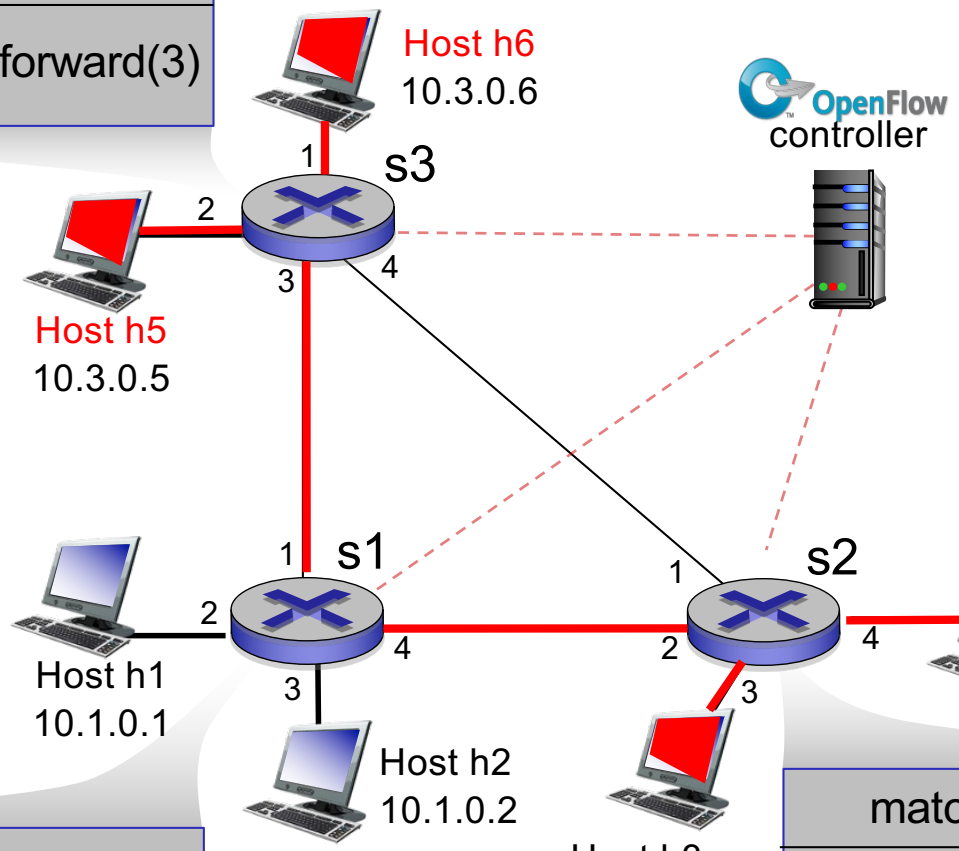
Orchestrated tables can create *network-wide* behavior, e.g.,:

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

OpenFlow: Example

Examples 3

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)