

# Chapter 4

## Network Layer: The Data Plane

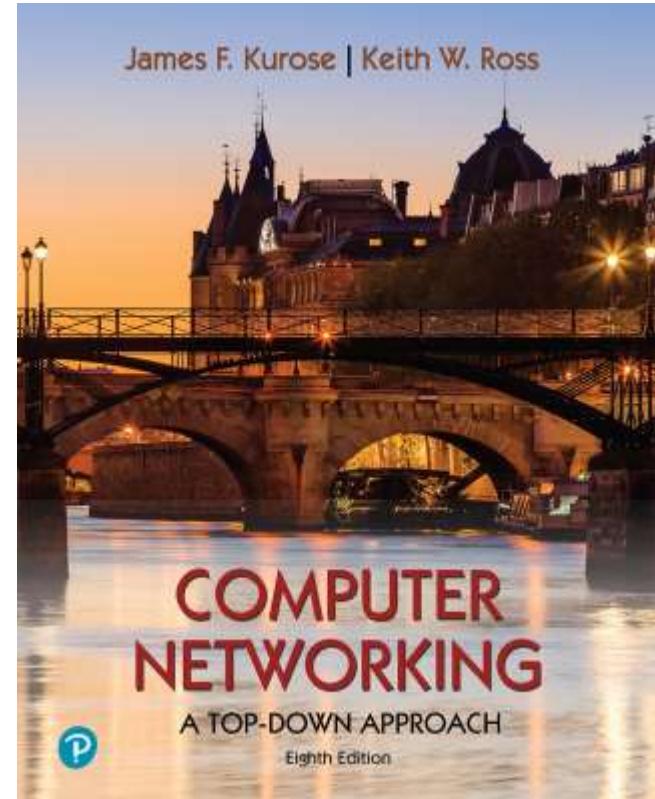
### A note on the use of these Powerpoint slides:

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

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



**Computer Networking:  
A Top-Down Approach**  
8<sup>th</sup> 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.

Decomposed the network layer into TWO parts:  
**Data Plane** (Chapter 4) & **Control Plane** (Chapter 5)



# Chapter 4: outline

## 4.1 Overview of Network layer

- data plane
- control plane

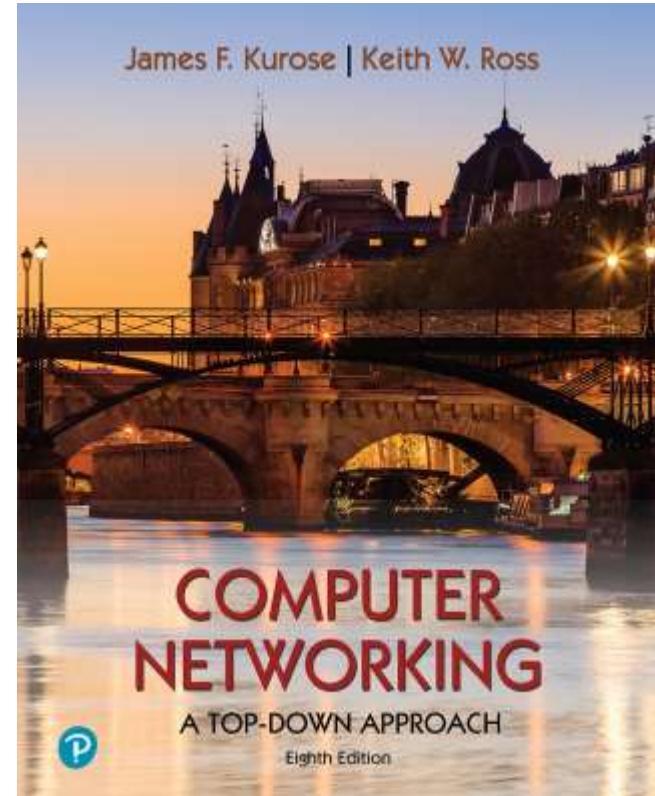
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

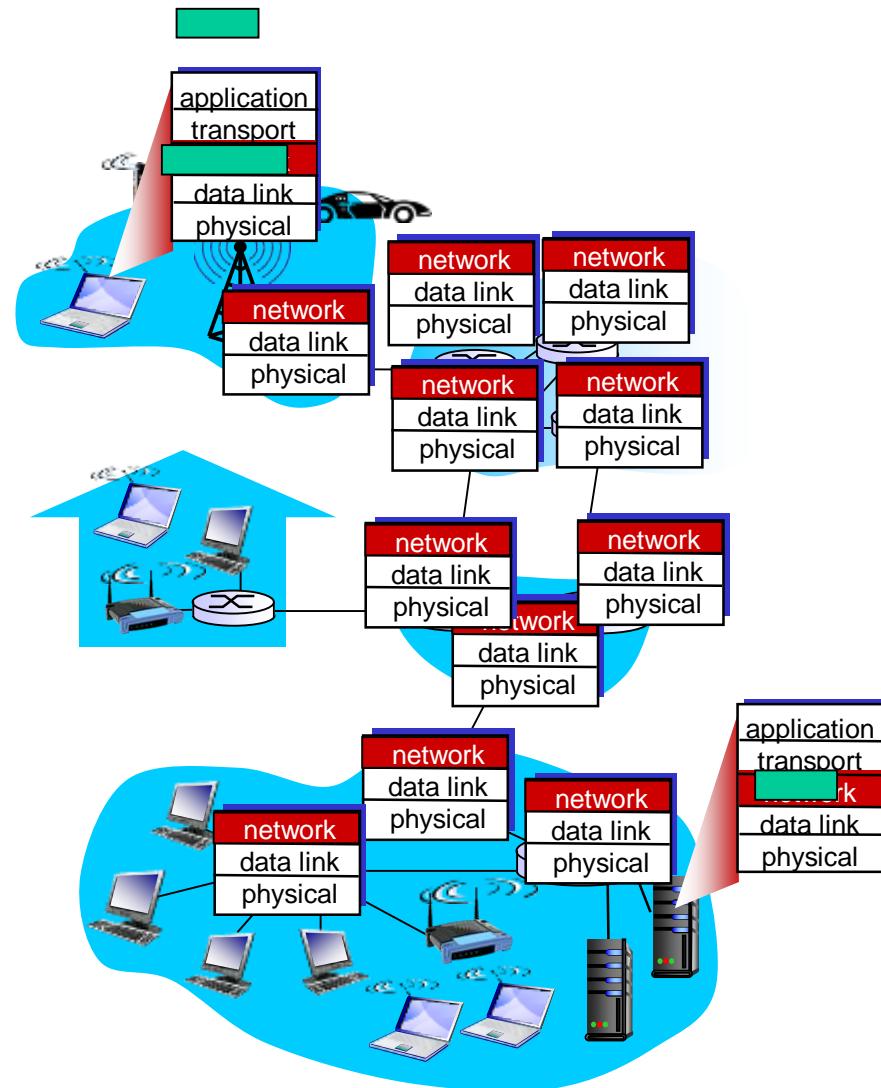
- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

## 4.4 Generalized Forwarding and SDN



# Network layer

- transport segment from sending to receiving host
  - on sending side encapsulates segments into datagrams
  - on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- 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

Two Functions



routing

1<sup>st</sup>: Forwarding

2<sup>nd</sup>: Routing

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

Determine route taken by packets from source to destination.

## Analogy:

Process of getting through single interchange.

## Analogy:

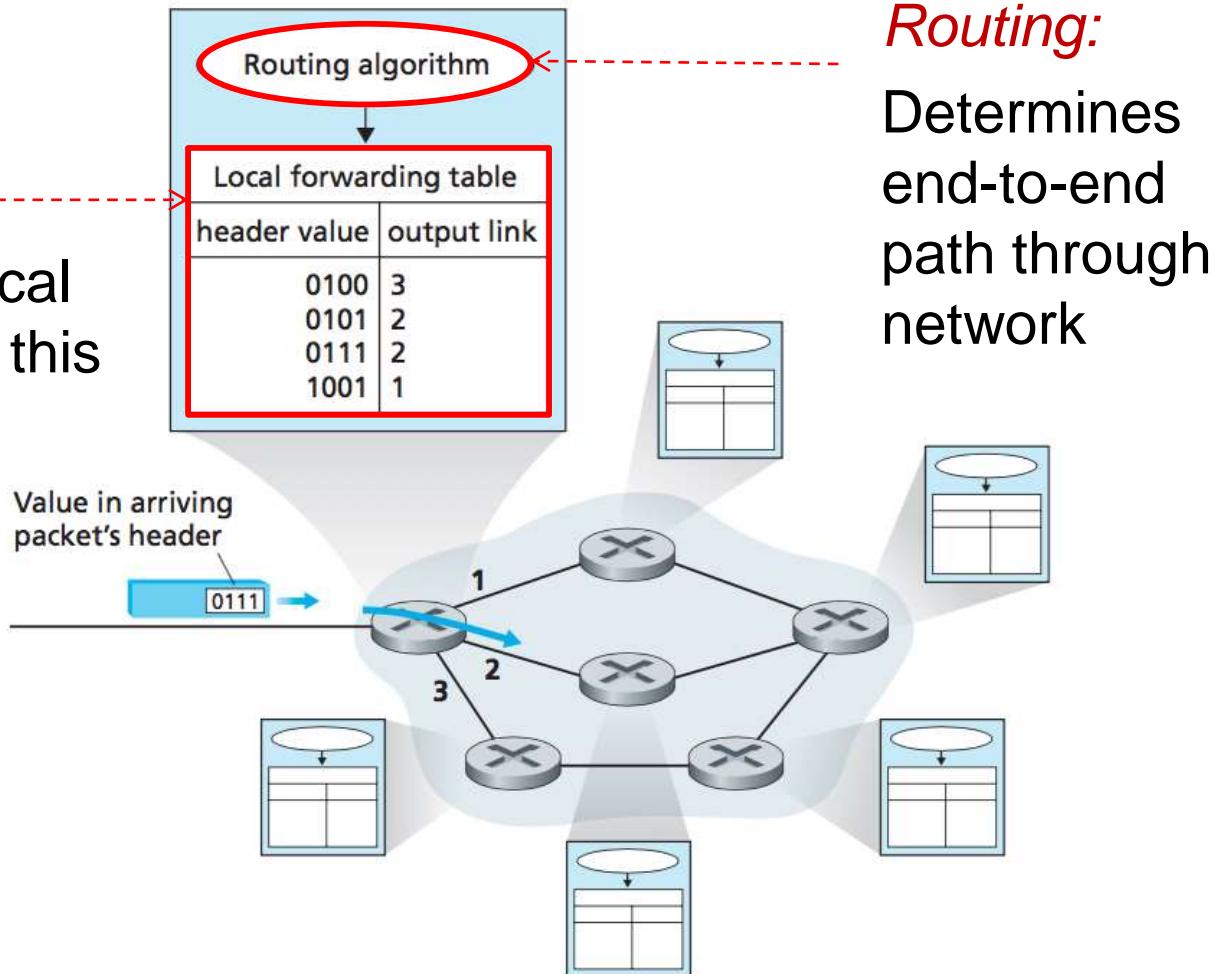
Process of planning trip from source to destination.

# Two key network-layer functions

Interplay between forwarding and routing

*Forwarding:*

Determines local forwarding for this router.



**Figure:** Routing algorithm determine values in forwarding tables

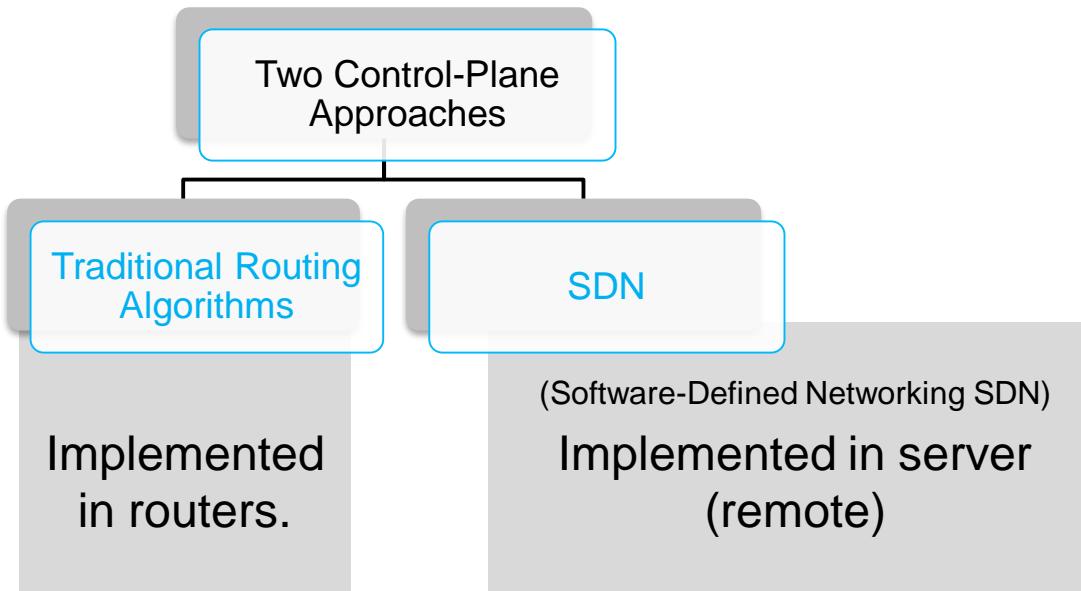
# Network layer: data plane, control plane

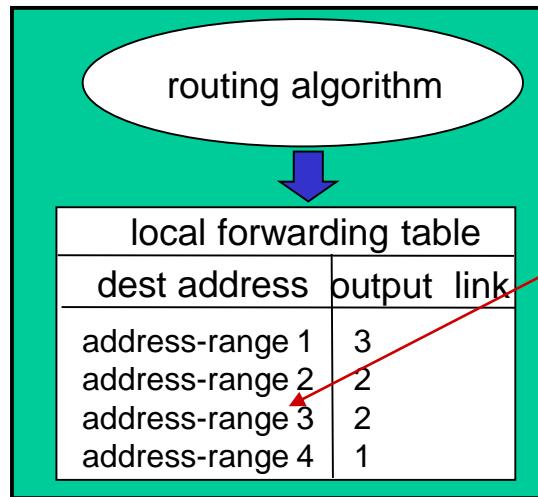
## Data plane

- **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.

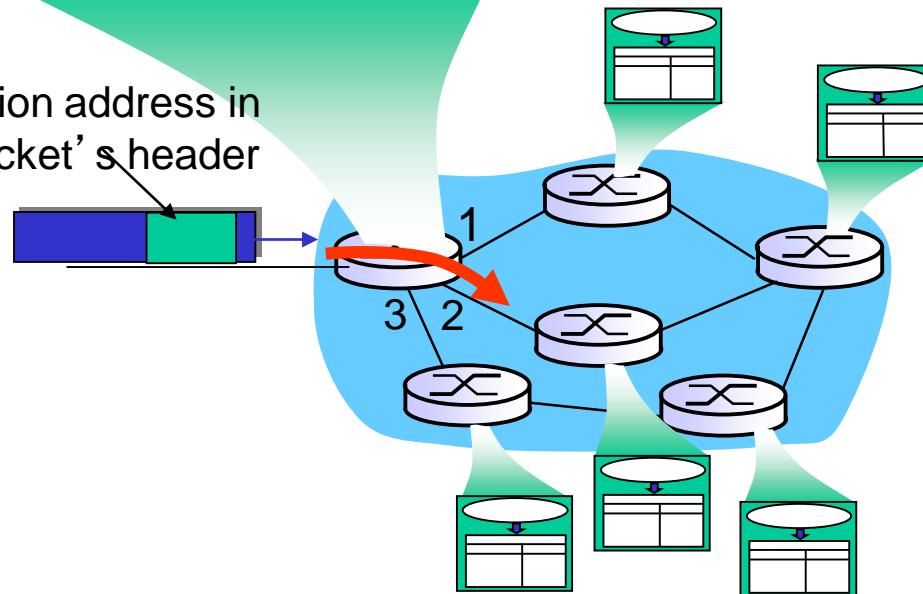




4 billion IP addresses, so rather than list individual destination address, list *range* of addresses (aggregate table entries)

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

IP destination address in arriving packet's header

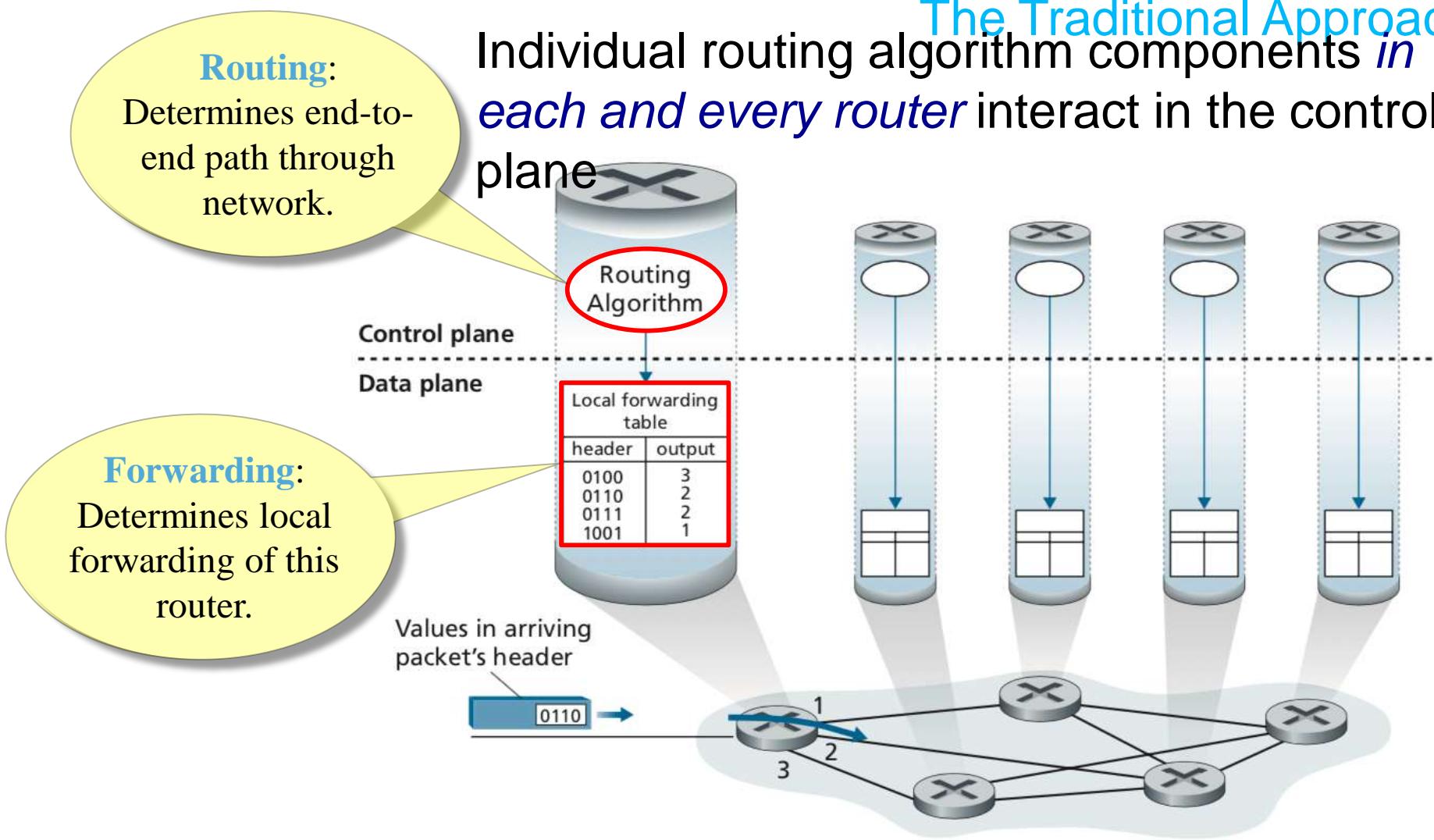


# Forwarding and Routing

## (a) Per-router control plane

### The Traditional Approach

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



**Figure:** Routing algorithm determine values in **forwarding tables**

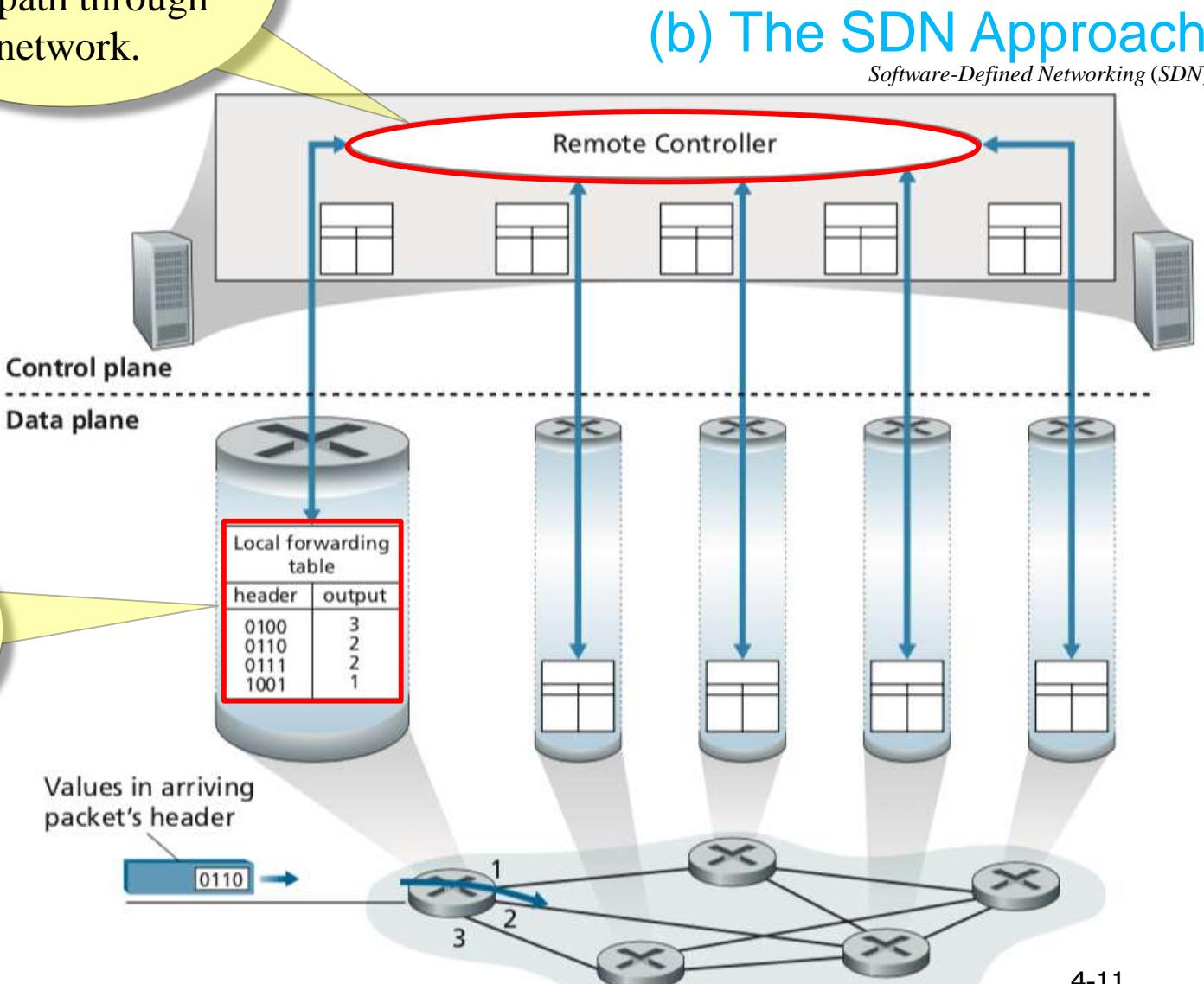
4-10

# Forwarding and Routing

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

## Routing:

Determines end-to-end path through network.



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

# Forwarding and Routing

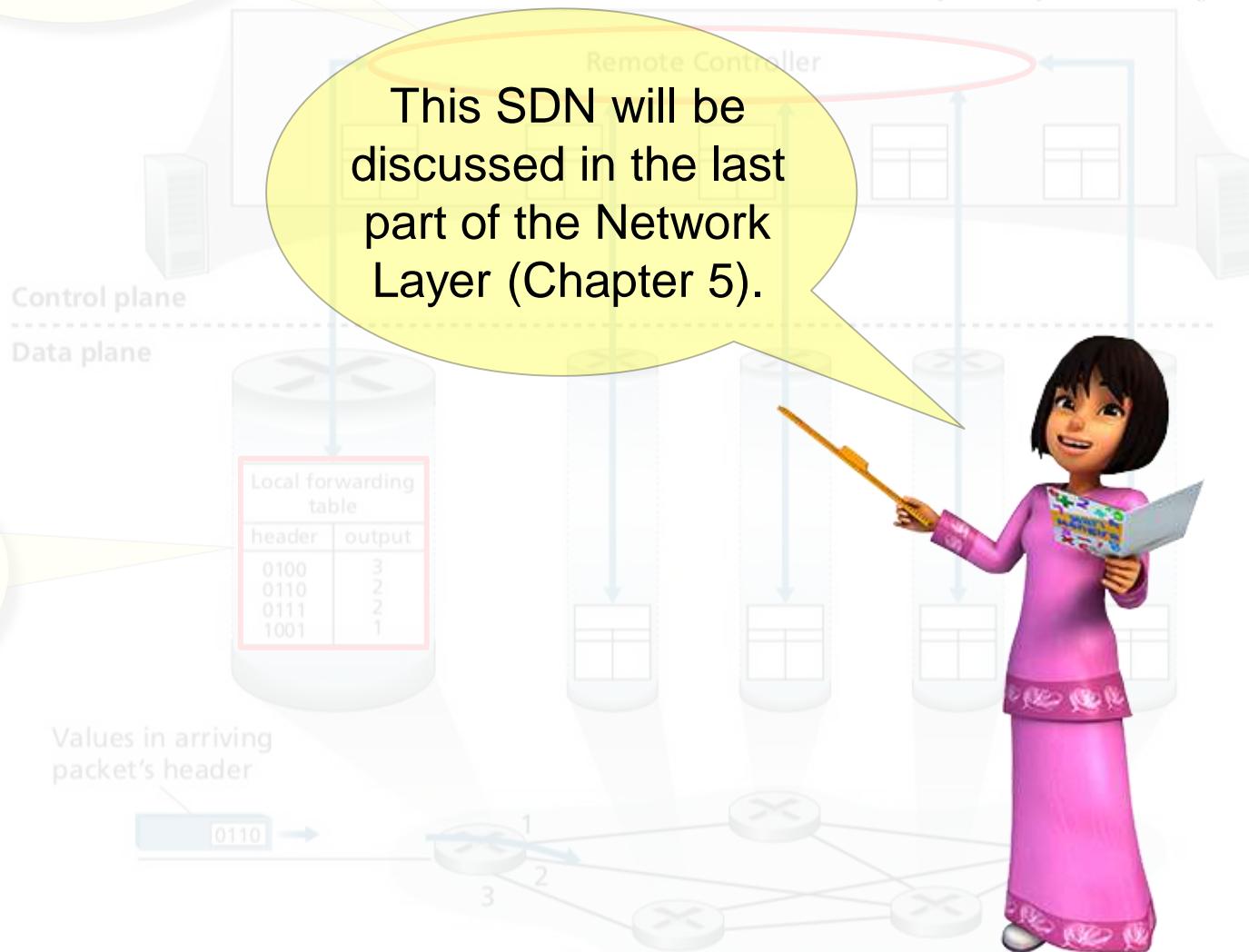
## (b) The SDN Approach

Software-Defined Networking (SDN)

This SDN will be discussed in the last part of the Network Layer (Chapter 5).

**Forwarding:**  
Determines local forwarding of this router.

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



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



**Q:** What possible service models that the network layer could provide to transport end-to-end delivery of *datagrams* from sender to receiver?

- ❖ guaranteed delivery.
- ❖ guaranteed delivery with bounded delay (e.g. less than 40 msec delay).
- ❖ In-order packets delivery.
- ❖ Guaranteed minimal bandwidth between hosts (e.g. 1 Mbps).
- ❖ Security (e.g. encryption)

# 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	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

# Chapter 4: outline

## 4.1 Overview of Network layer

- data plane
- control plane

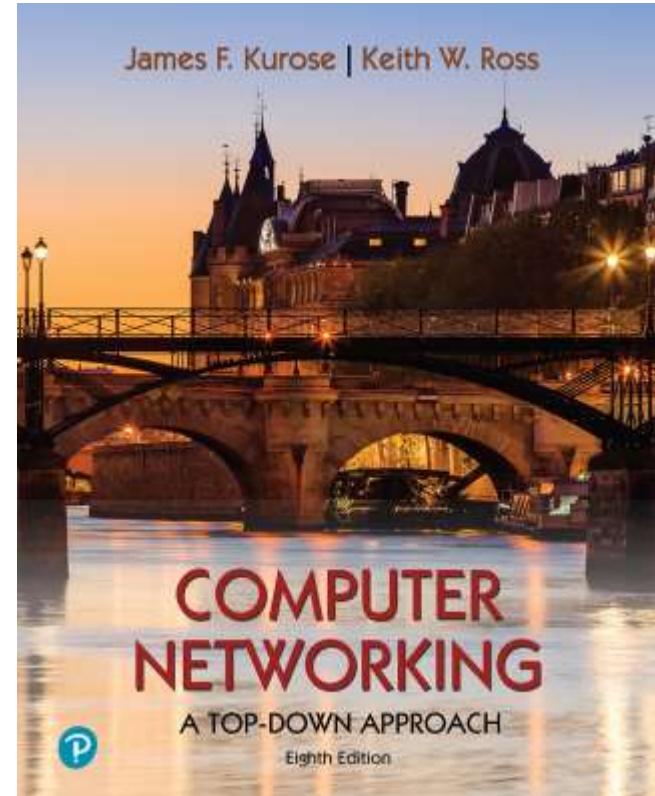
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

## 4.4 Generalized Forwarding and SDN

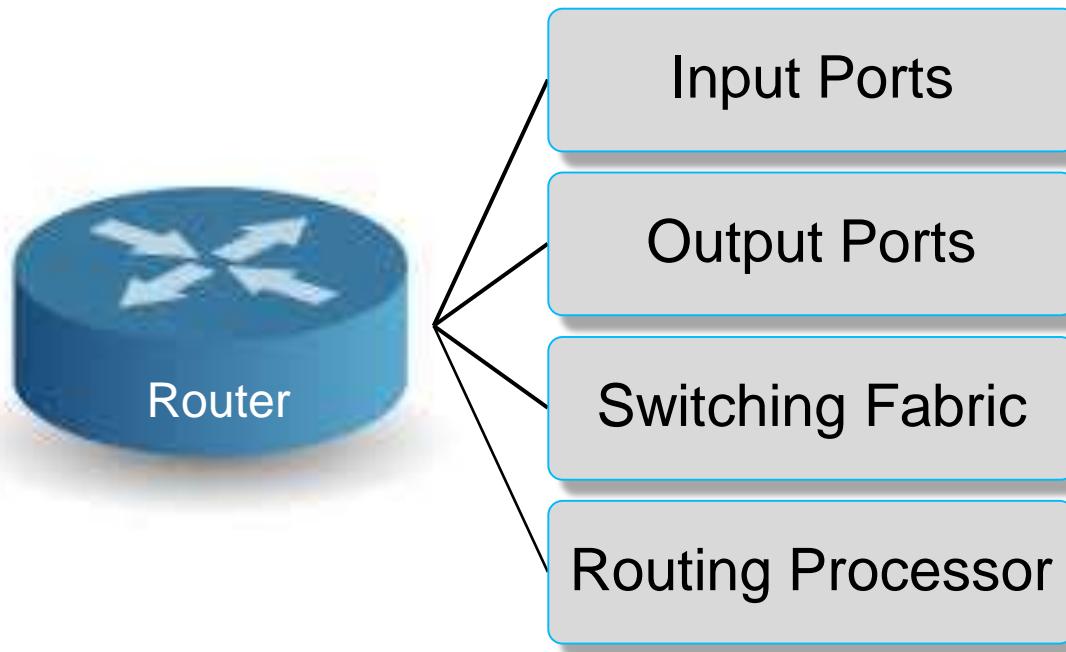


# Router architecture overview

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

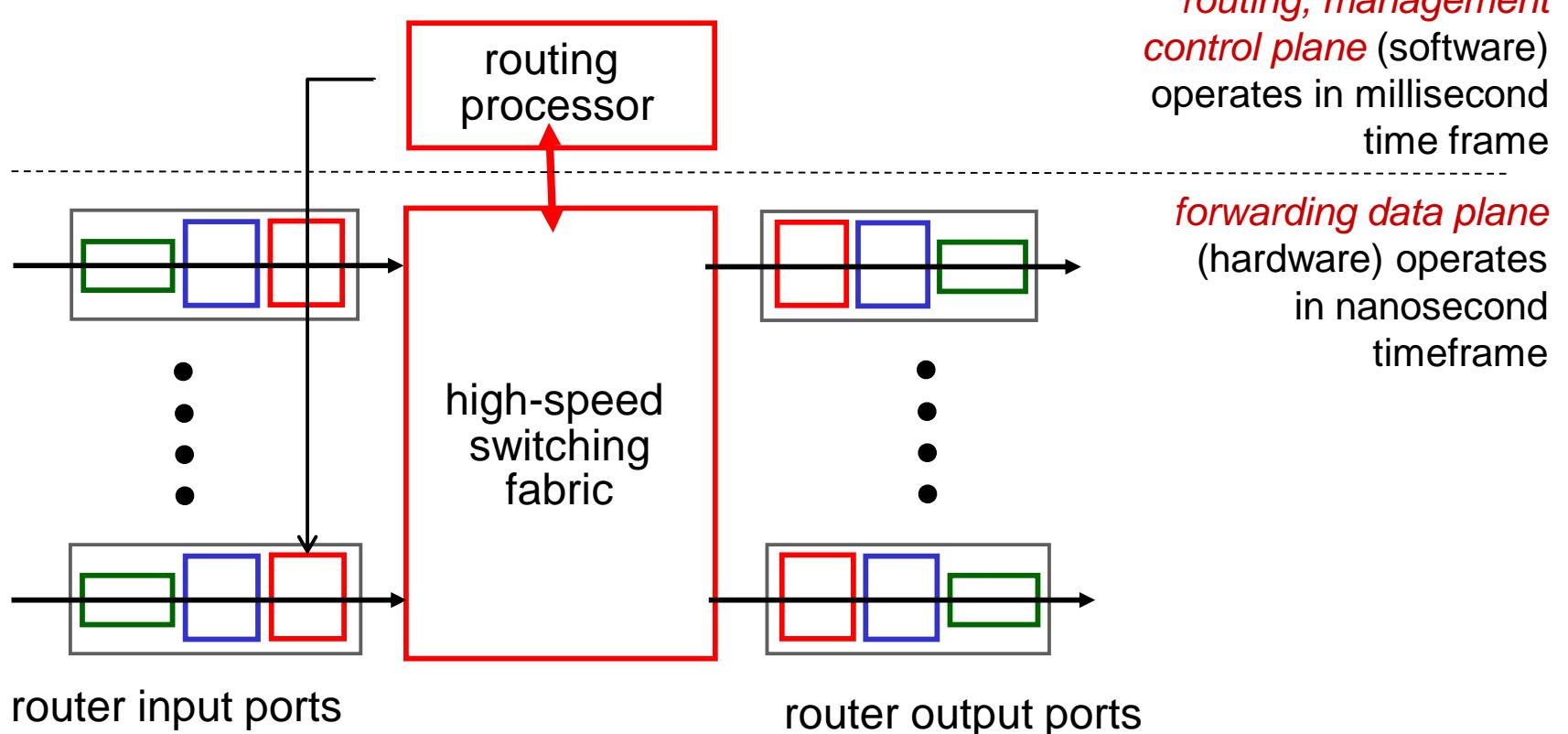
- ❖ Four router components can be identified:



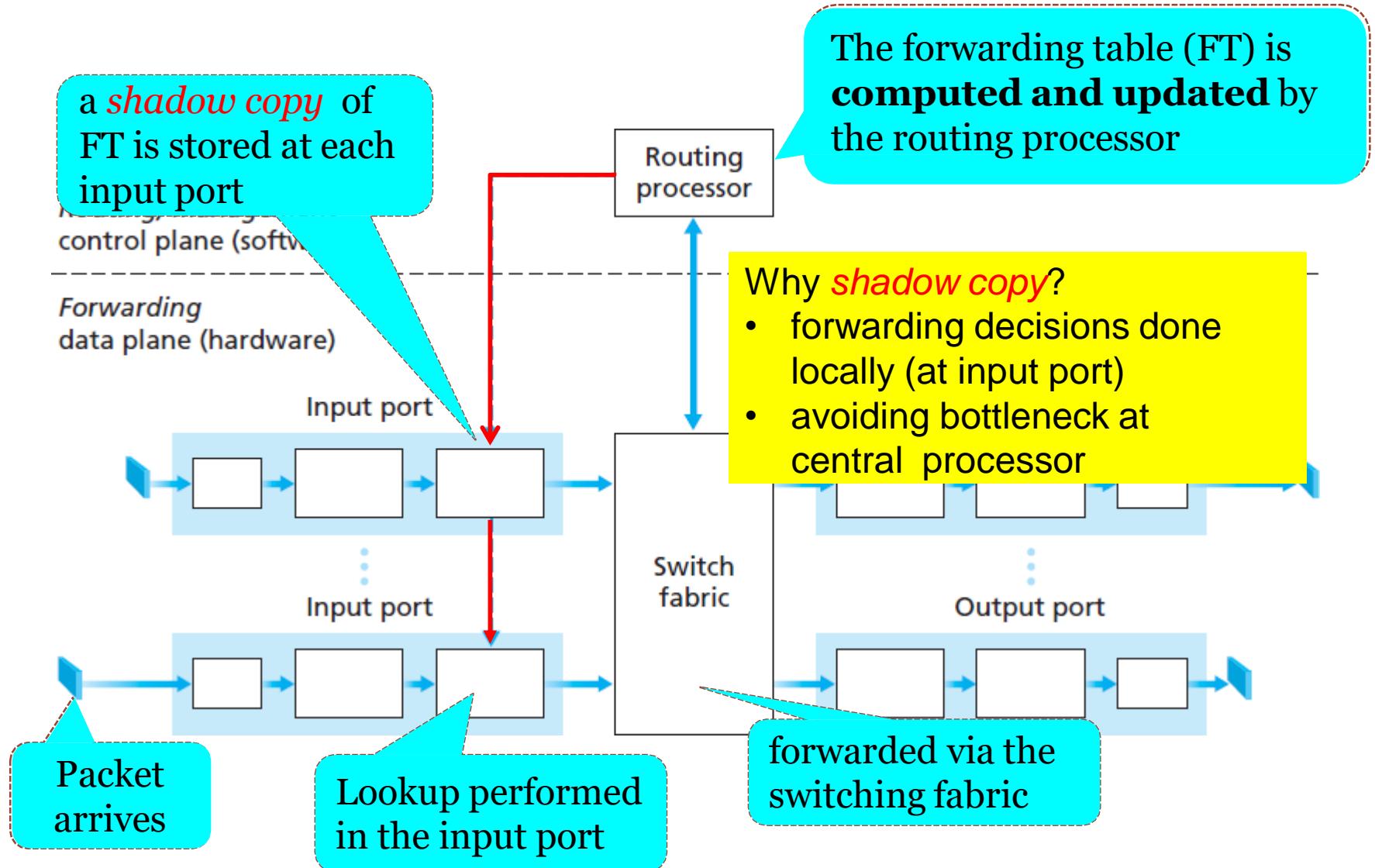
**Figure:** Four router components

# Router architecture overview

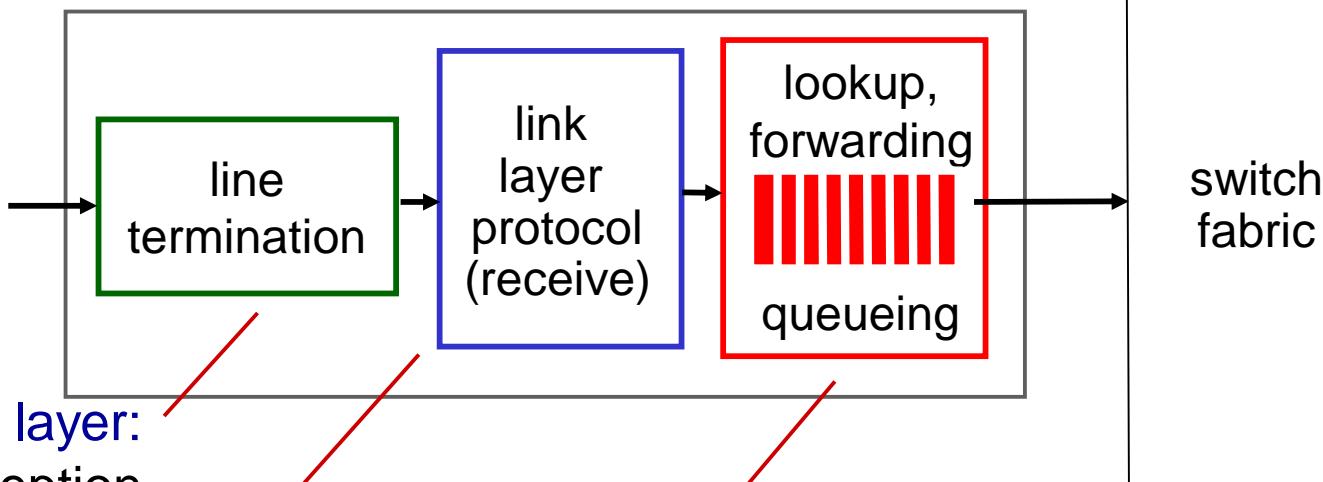
- ## ■ high-level view of generic router architecture:



# Router architecture overview



# Input port functions



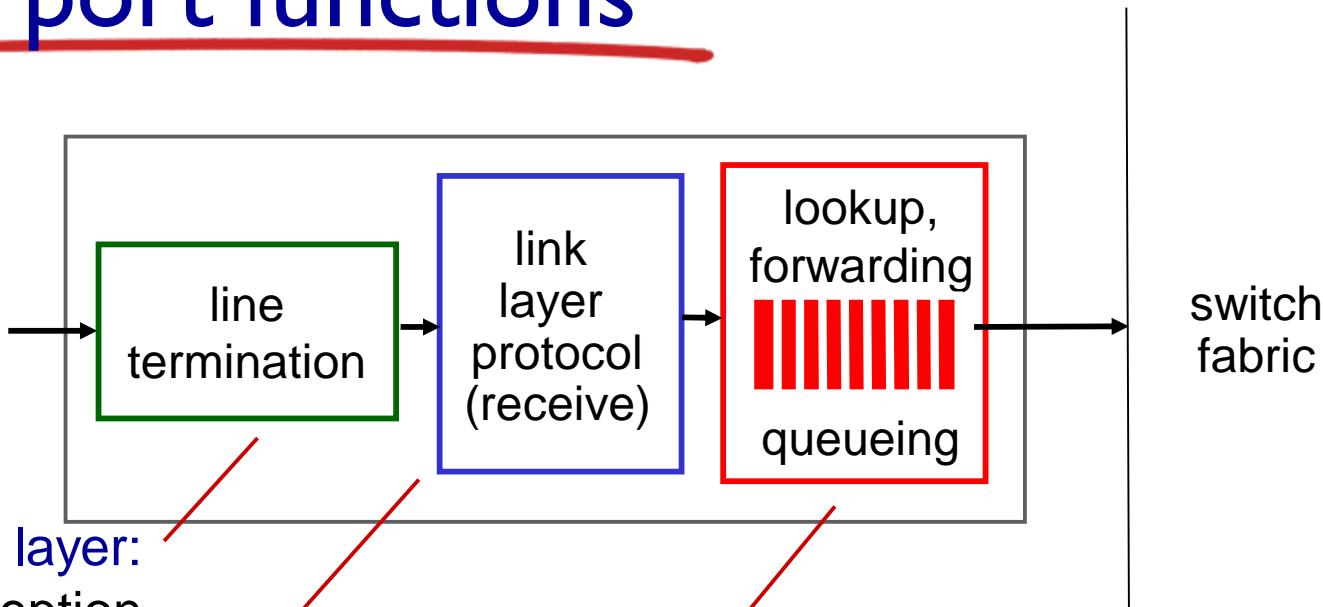
physical layer:  
bit-level reception

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 5

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 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

**Q:** but what happens if ranges don't divide up so nicely?

# Longest prefix matching

longest prefix 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

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

examples:

↑  
**match!**  
↓

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:

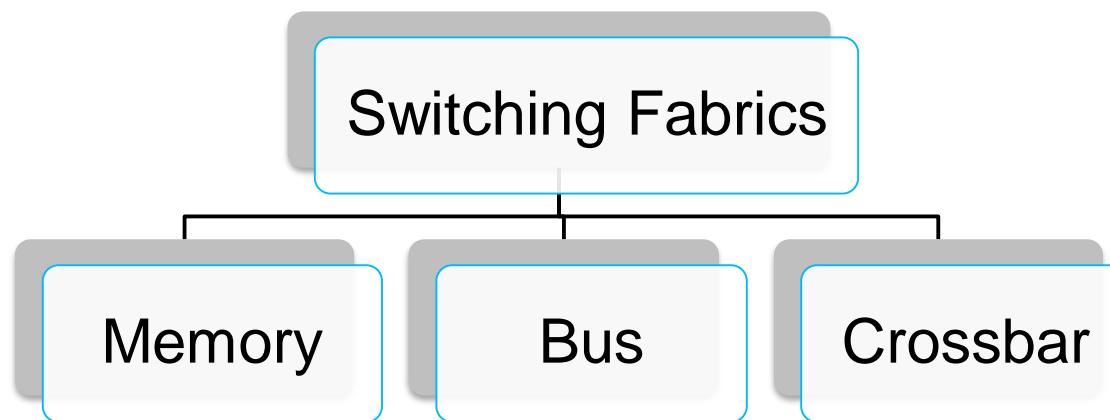
11001000 00010111 00010110	10100001	which interface?
11001000 00010111 00011000	10101010	which interface?

# 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 up ~1M routing table entries in TCAM

# Switching Fabrics

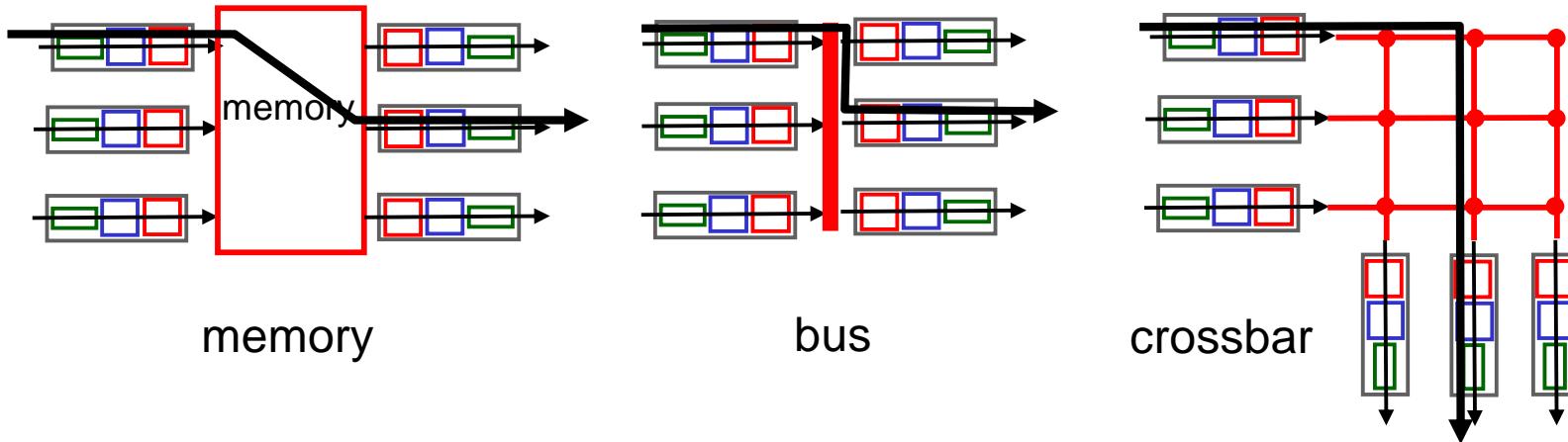
- ❖ Transfer packet from **input buffer** to appropriate **output buffer**.
- ❖ **Switching rate**: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate.
  - $N$  inputs: switching rate =  $N$  times line rate desirable.



**Figure:** Three type of switching fabrics techniques

# Switching fabrics

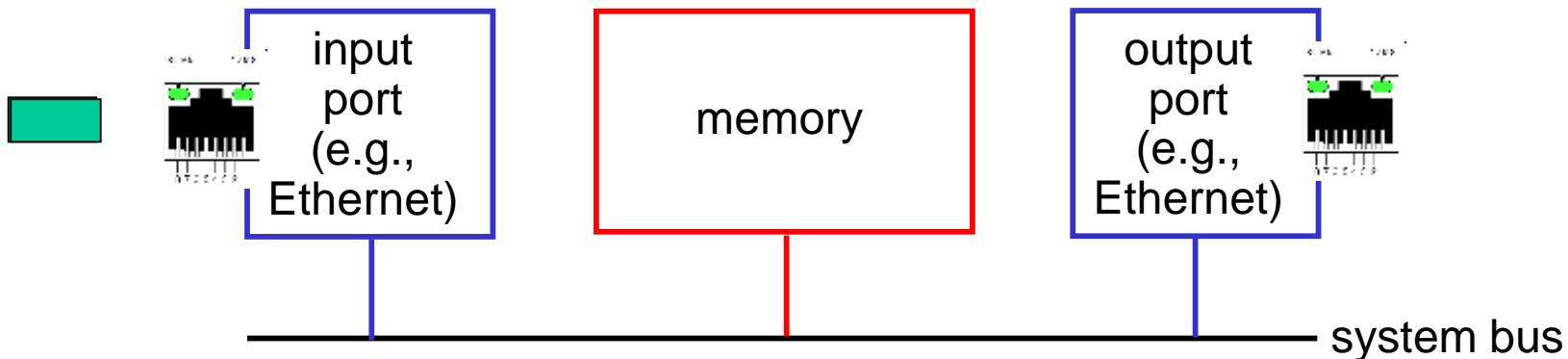
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transferred from inputs to outputs
  - often measured as multiple of input/output line rate
  - $N$  inputs: switching rate  $N$  times line rate desirable
- three types of switching fabrics



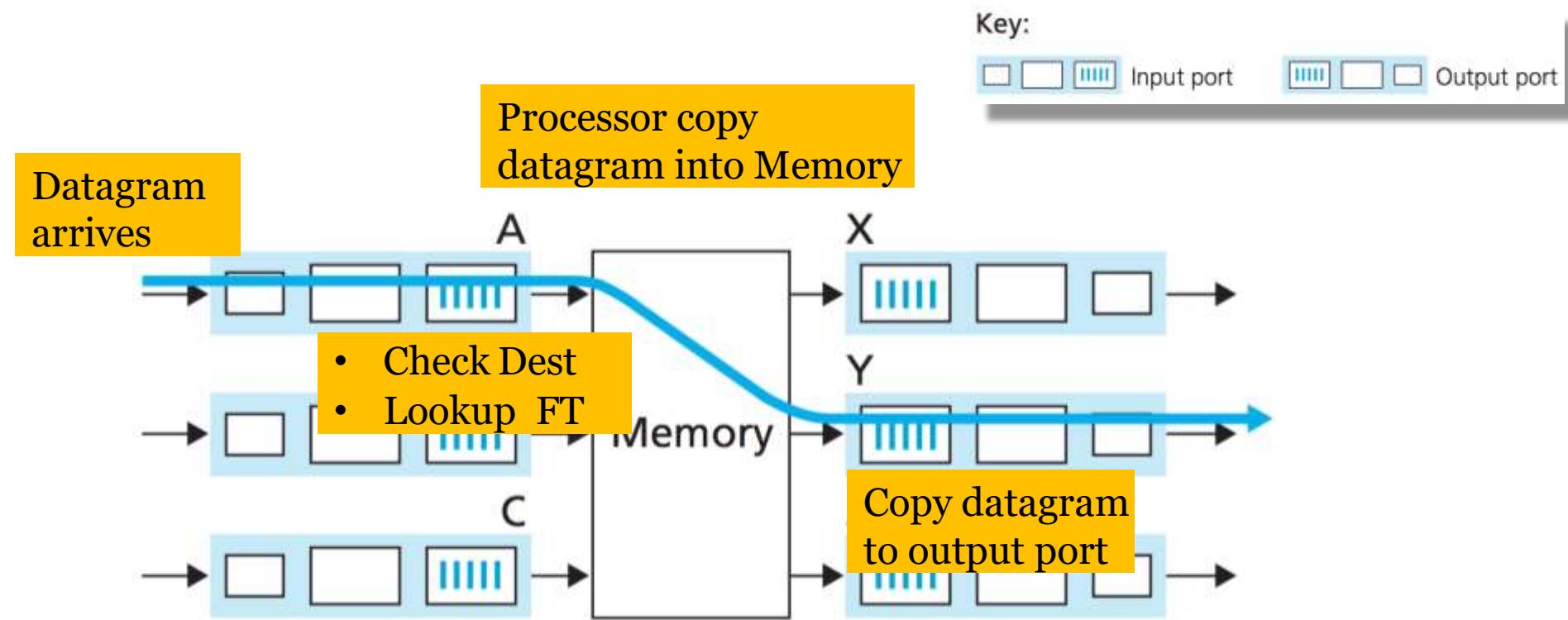
# Switching via memory

*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)

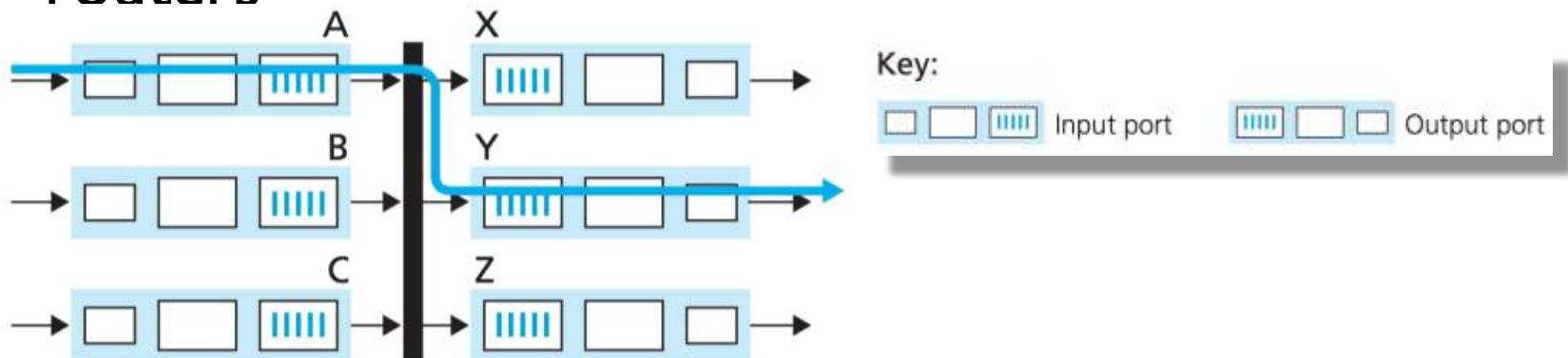
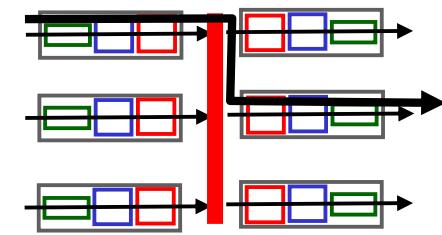


# Switching via memory

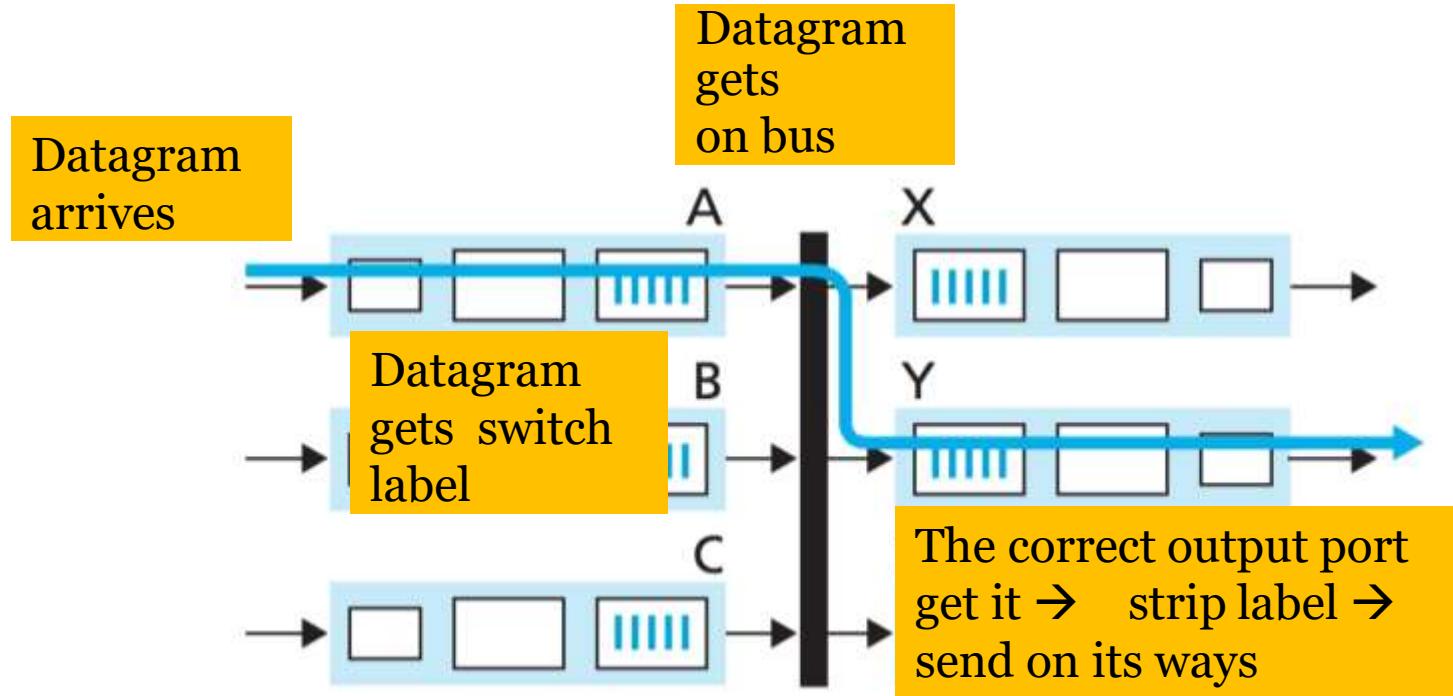


# Switching via a bus

- 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



# Switching via a bus



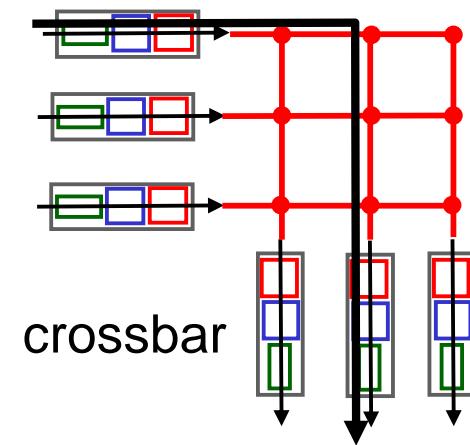
Key:

Input port

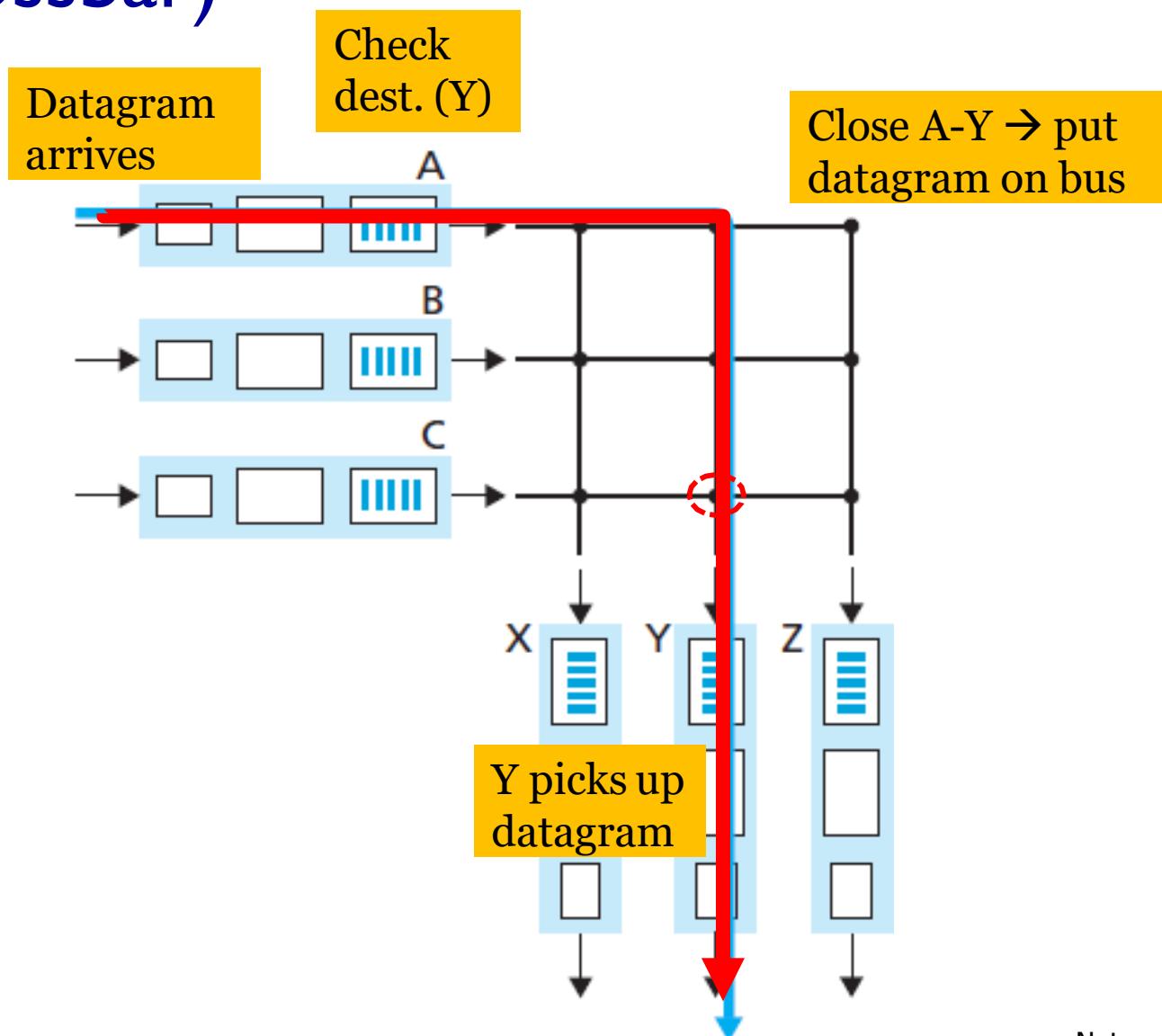
Output port

# Switching via interconnection network (crossbar)

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced parallelism design:
  - fragmenting datagram into fixed length cells on entry
  - switch cells through the fabric, reassemble datagram at exit
- Cisco 12000: switches 60 Gbps through the interconnection network

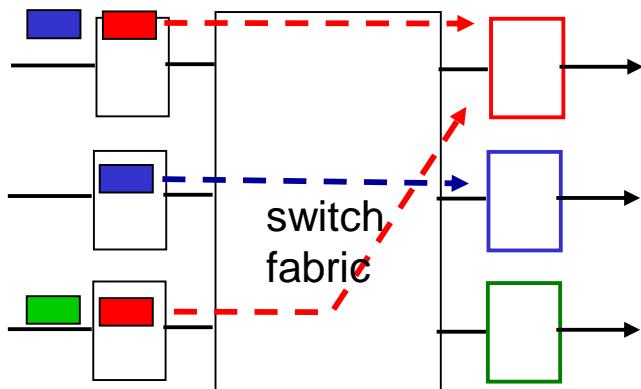


# Switching via interconnection network (crossbar)

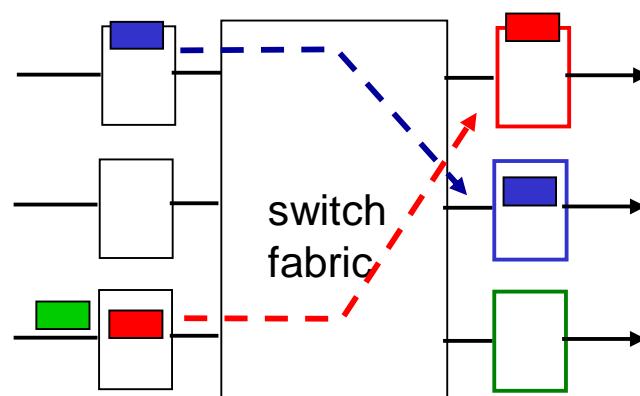


# Input port queuing

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

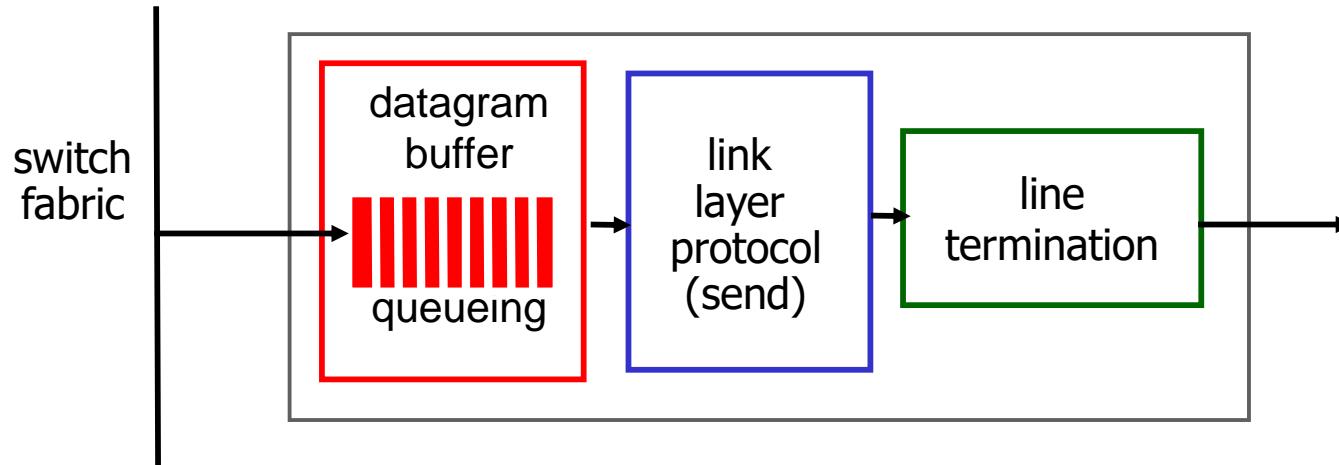


output port contention:  
only one red datagram can be  
transferred.  
*lower red packet is blocked*



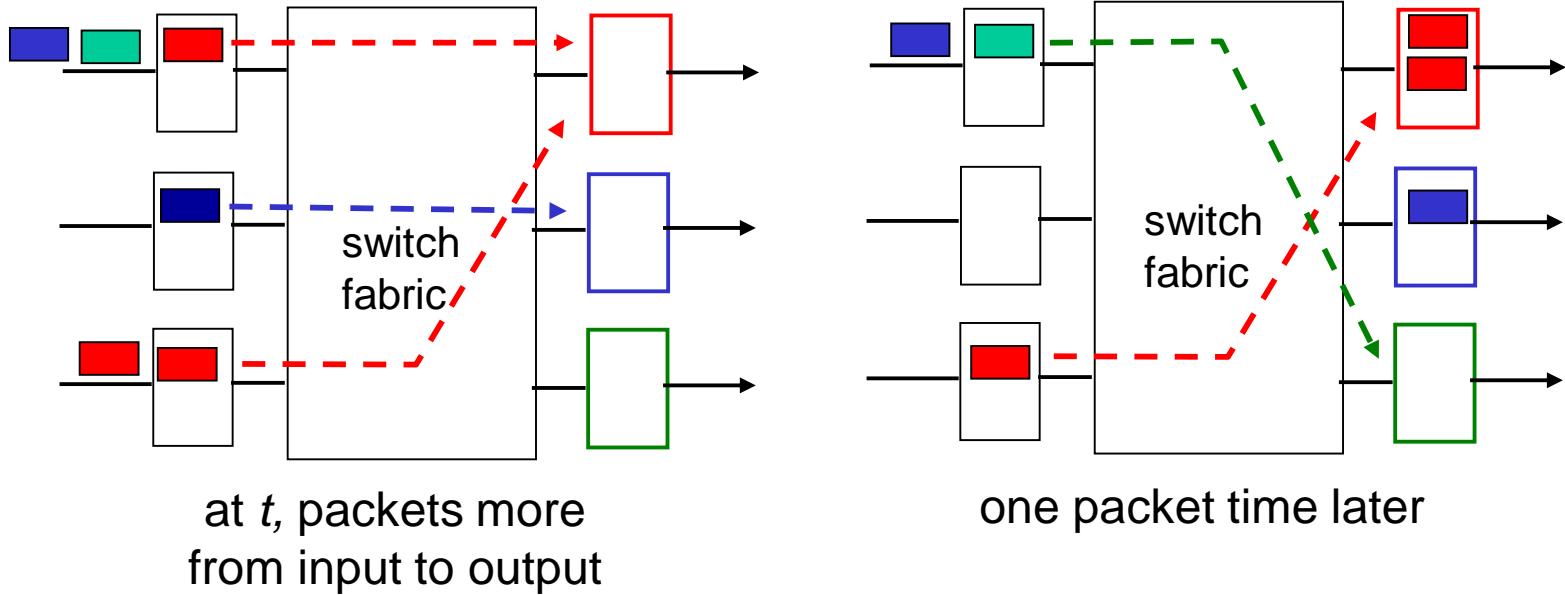
one packet time later:  
green packet  
experiences HOL  
blocking

# Output port queuing



- ***buffering*** required from fabric faster rate  
Datagram (packets) can be lost due to congestion, lack of buffers
- ***scheduling*** datagrams  
Priority scheduling – who gets best performance, network neutrality

# Output port queueing



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

# 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  
» = 2.5 Gbit buffer
- recent recommendation: with  $N$  flows, buffering equal to

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

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

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

### **Solution:**

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

$$\text{Buffer size, } B = \frac{RTT * C}{\sqrt{N}} = \frac{400msec * 20Mbps}{\sqrt{16}} \\ = \frac{0.4sec * 20Mbps}{4} \\ = 0.4 * 5 = 2Mb$$

# Chapter 4: outline

## 4.1 Overview of Network layer

- data plane
- control plane

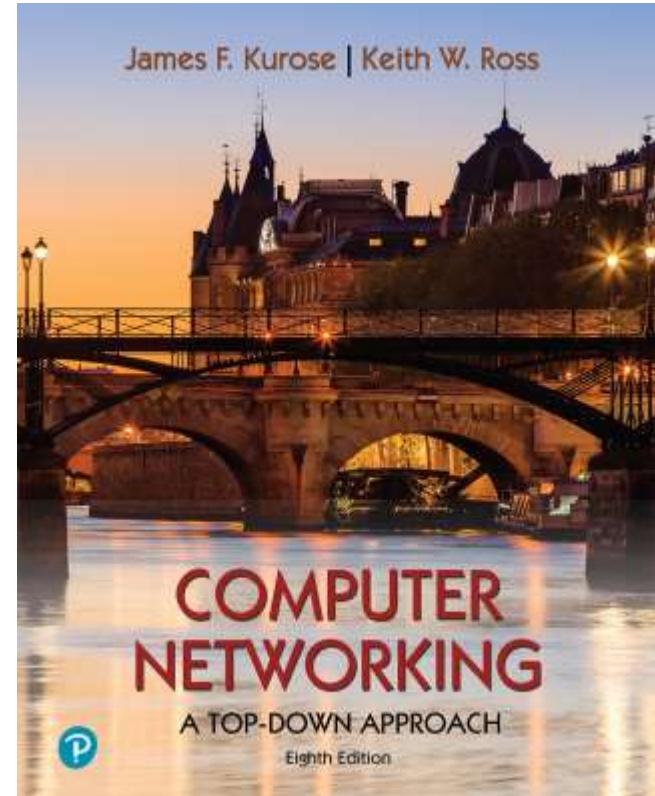
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

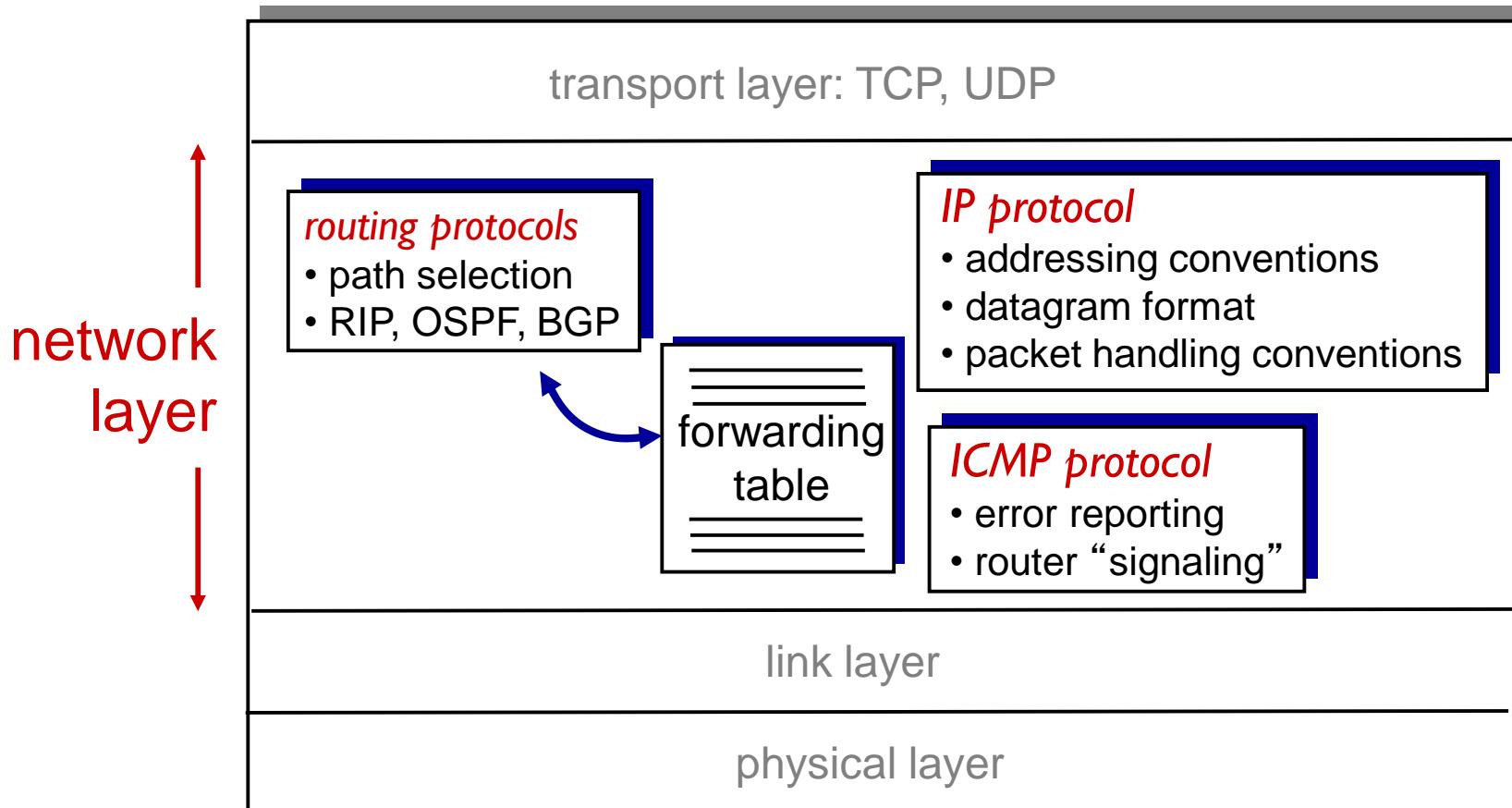
- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

## 4.4 Generalized Forwarding and SDN



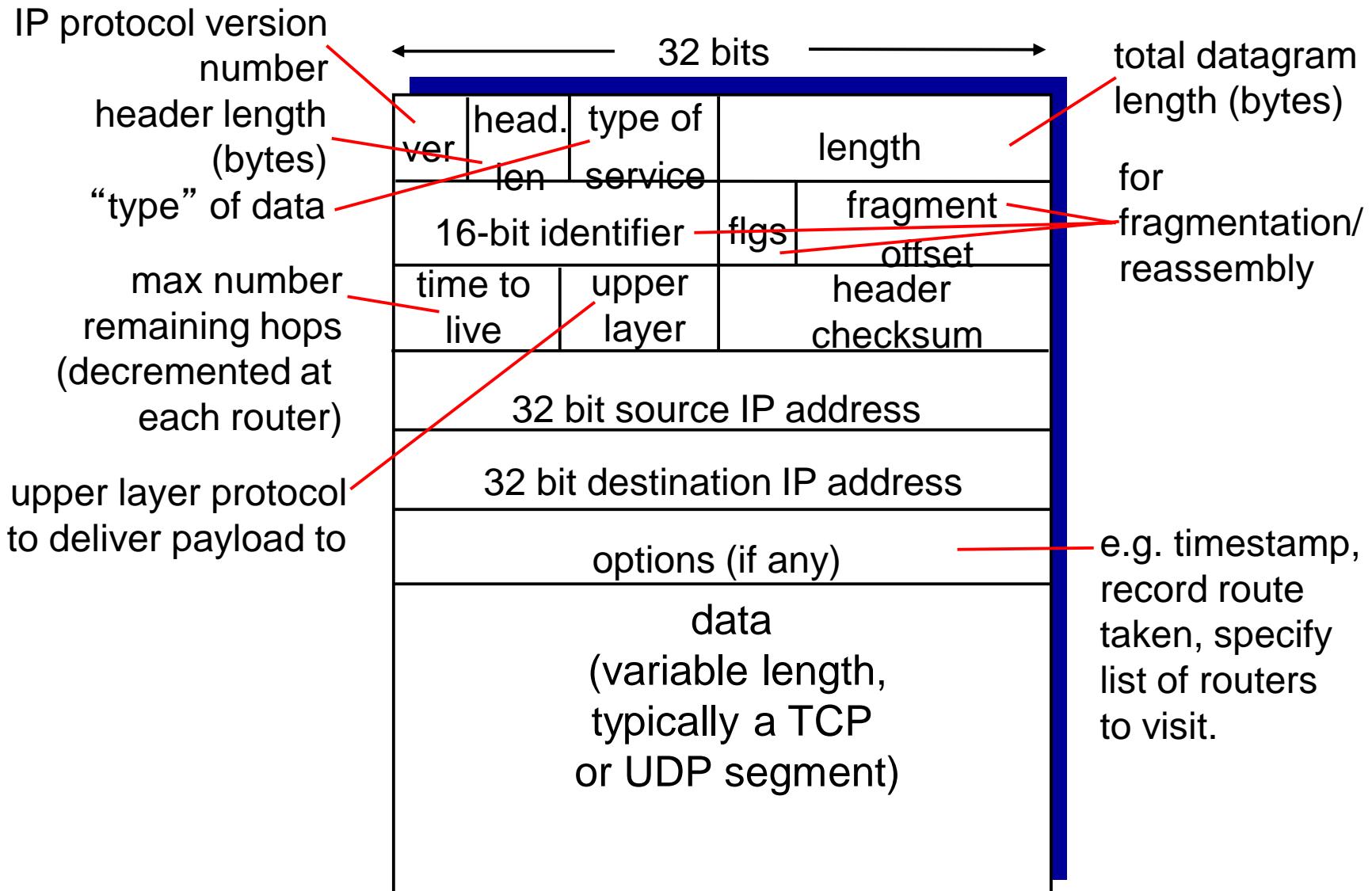
# The Internet network layer

host, router network layer functions:



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

# IP datagram format



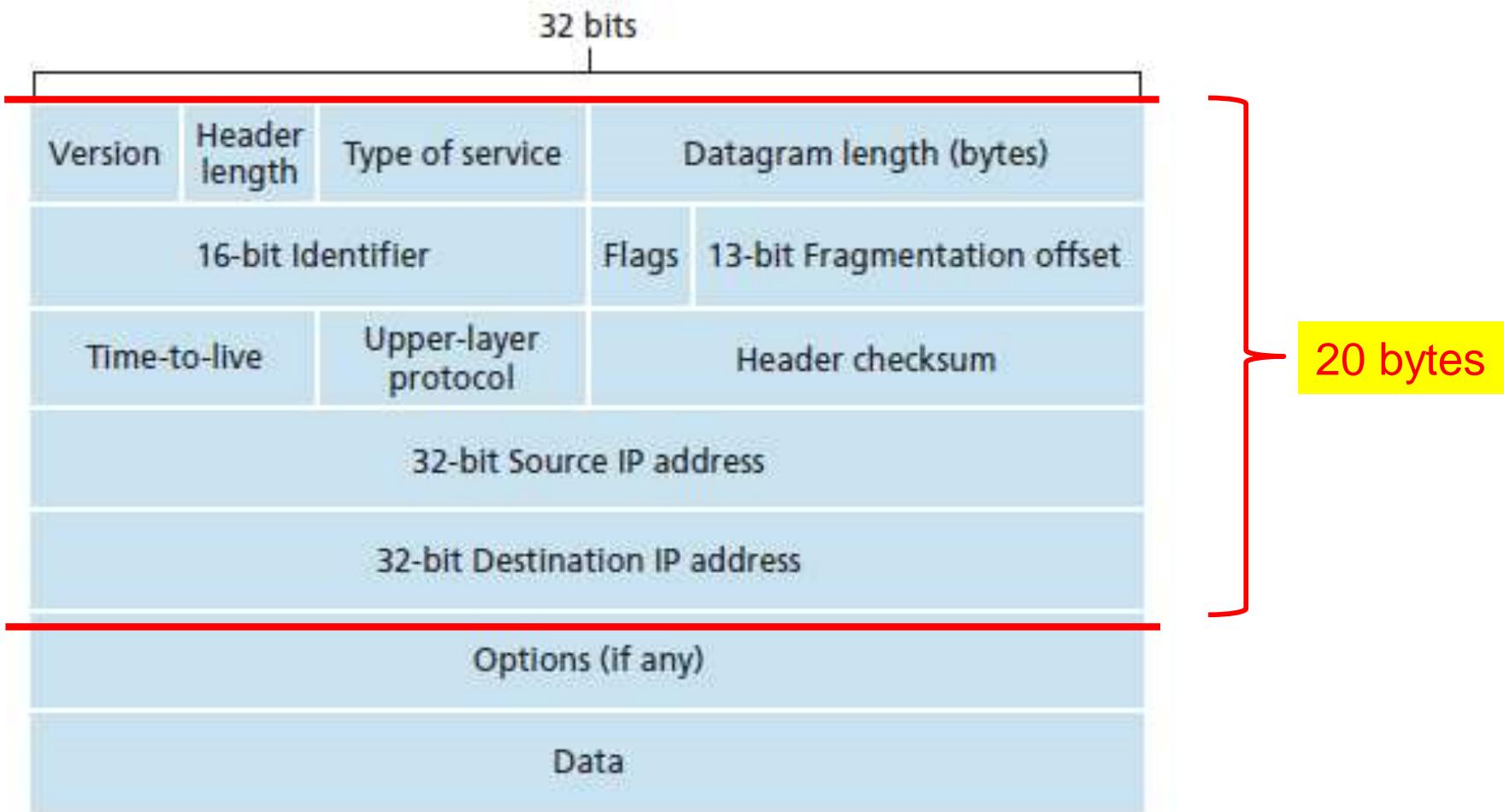
# IP datagram format

- 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



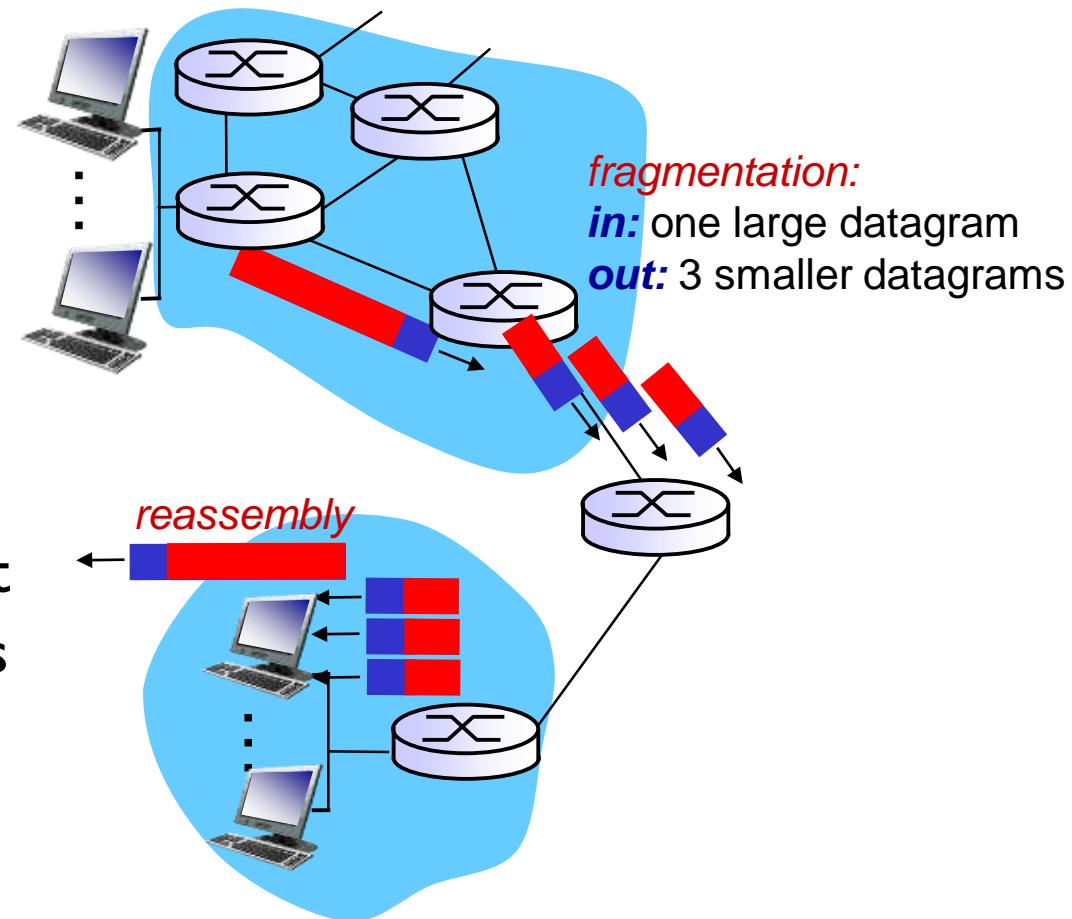
- How much overhead?
  - 20 bytes of TCP
  - 20 bytes of IP
  - = 40 bytes + app layer overhead

# IP datagram format



# IP fragmentation, reassembly

- network links have MTU (max.transfer size) - 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

Fragmentation:

In: one large datagram (4,000 bytes)  
Out: 3 smaller datagrams

Link MTU: 1,500 bytes

=  $4000 / 1500$  bytes  
= 3 datagrams

Reassembly:

In: 3 smaller datagrams  
Out: one large datagram (4,000 bytes)

# 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

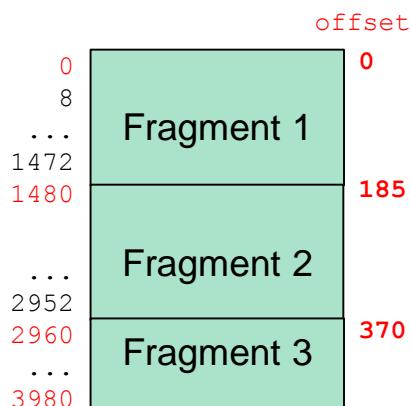
Why 1480  
not 1500?

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 $185 \cdot 8 = 1,480$ )	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 $370 \cdot 8 = 2,960$ )	flag = 0 (meaning this is the last fragment)

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

Fragments of  
same datagram

Offset value specified in  
units of 8-byte chunks



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:

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: outline

## 4.1 Overview of Network layer

- data plane
- control plane

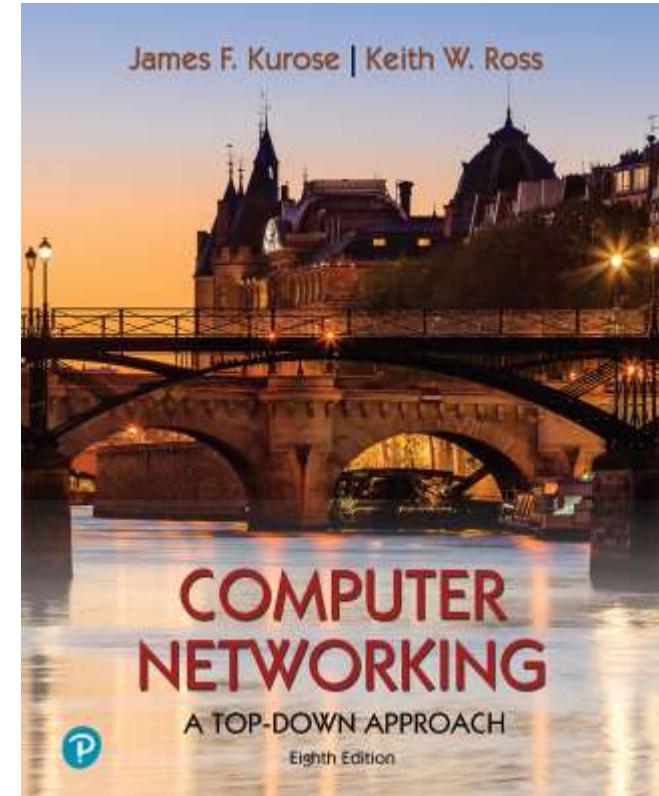
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

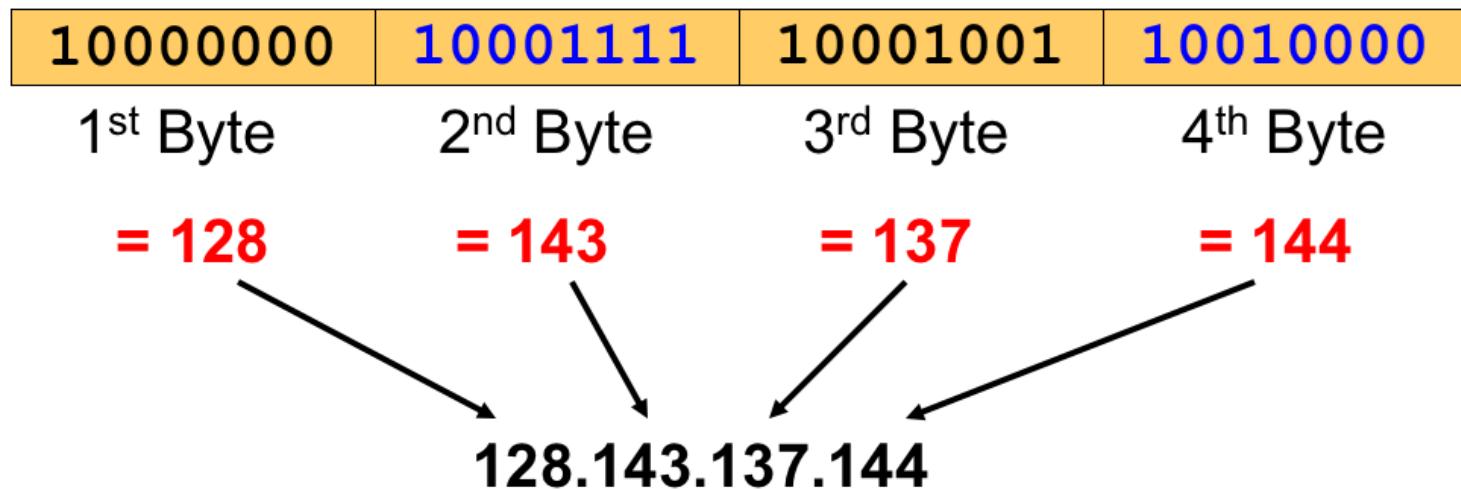
- datagram format
- fragmentation
- **IPv4 addressing**
- network address translation (NAT)
- IPv6

## 4.4 Generalized Forwarding and SDN



## What is an IP Address?

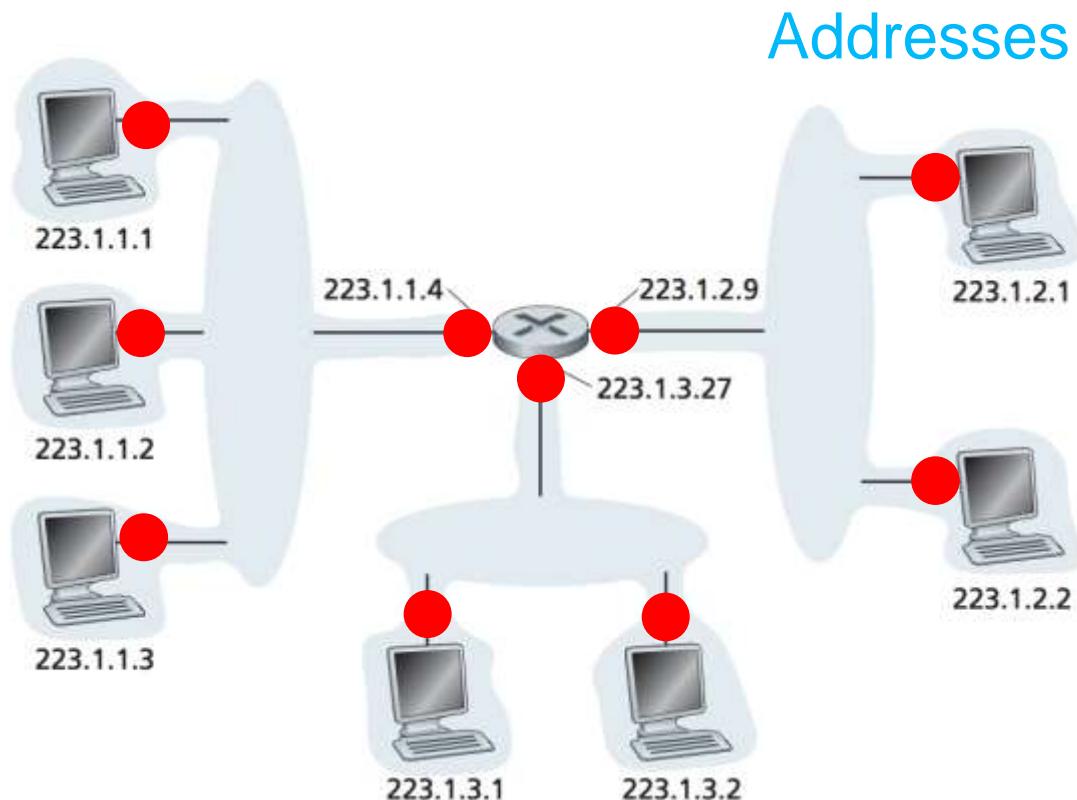
- ❖ IP address is used to identify a host within a network.
  - are written in a so-called **dotted decimal notation**
  - Each byte is identified by a decimal number in the range [0...255]:
- ❖ **Example:**



# IPv4 Addressing

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

- router's typically have multiple interfaces.
- host typically has one or two interfaces.  
(e.g., wired Ethernet, wireless 802.11)



*One IP address associated  
with each interface !*

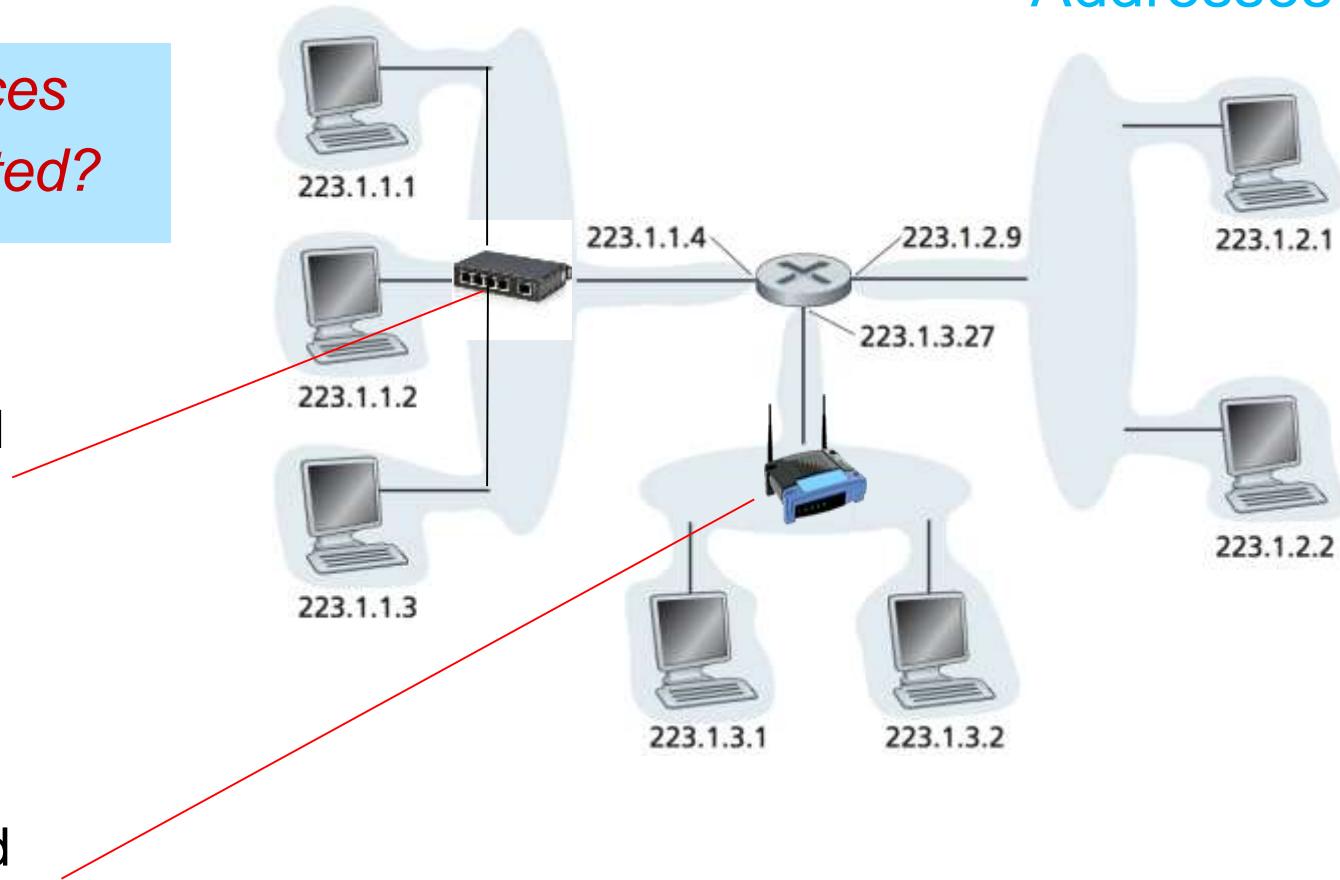
# IPv4 Addressing

Addresses

*Q: How are interfaces actually connected?*

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

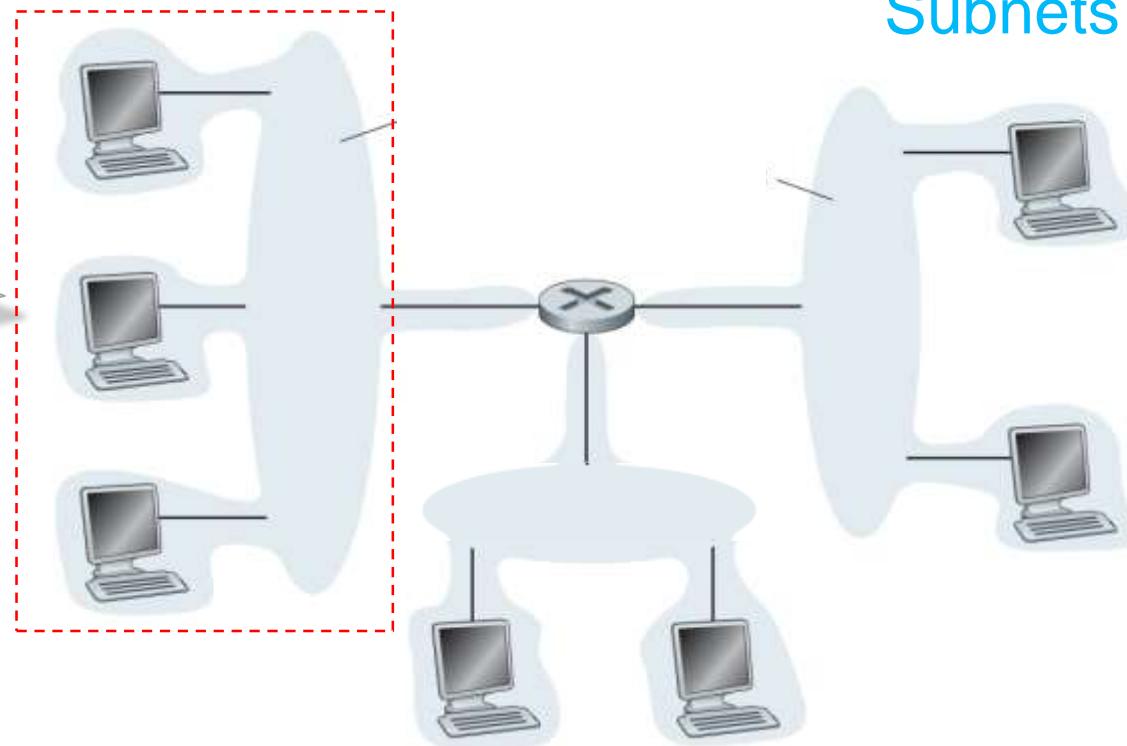
*A: wireless WiFi interfaces connected by WiFi base station (**AP – access point**)*



# IPv4 Addressing – Subnet/Subnetting

## Subnets

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

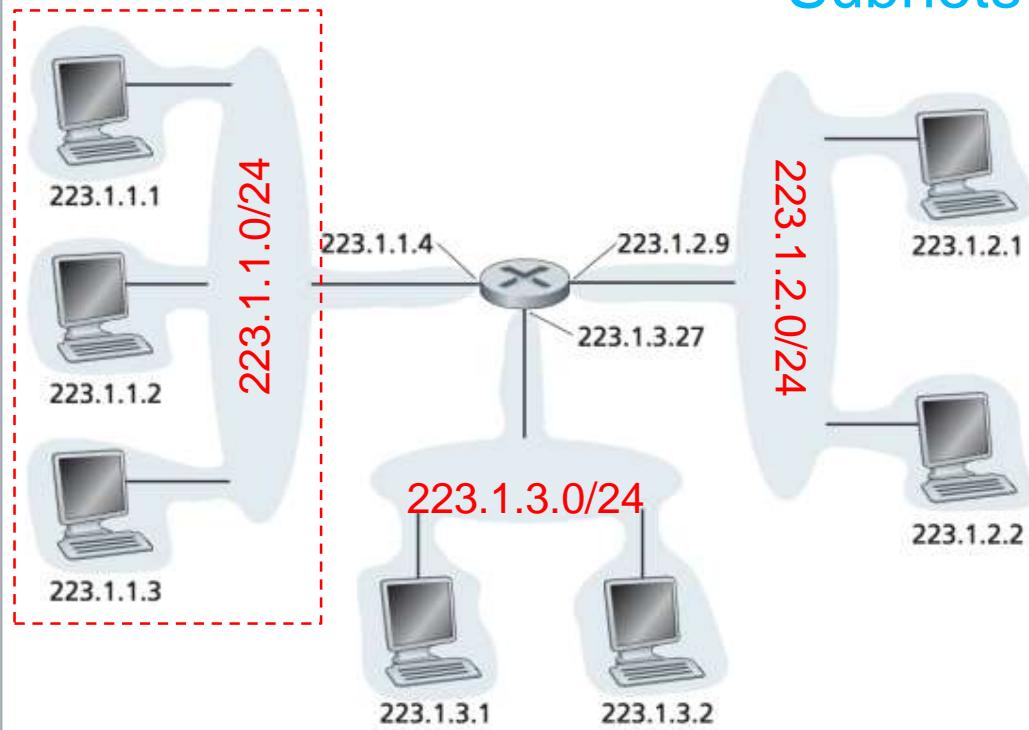


### What's a **subnet** ?

- ❖ Subnet or subnetwork 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

## Subnets



- ❖ A subnet is a group of hosts identified with **having the same 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 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 subnet addresses

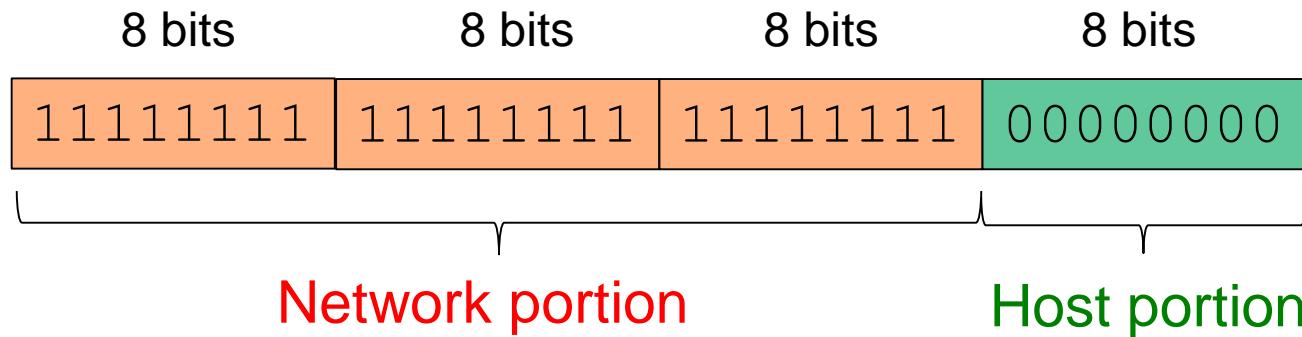
## Subnet Mask

- ⊕ 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

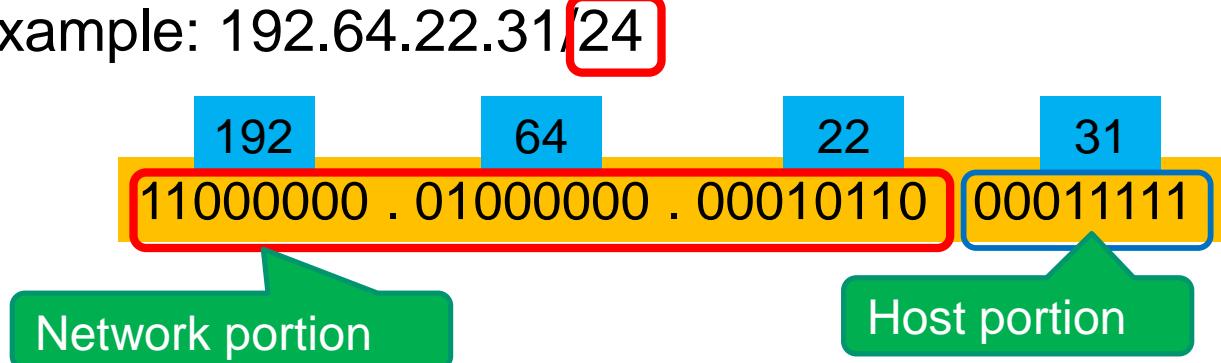
- IP address with **w.x.y.z /24** address.



- Thus, subnet mask = **255.255.255.0**

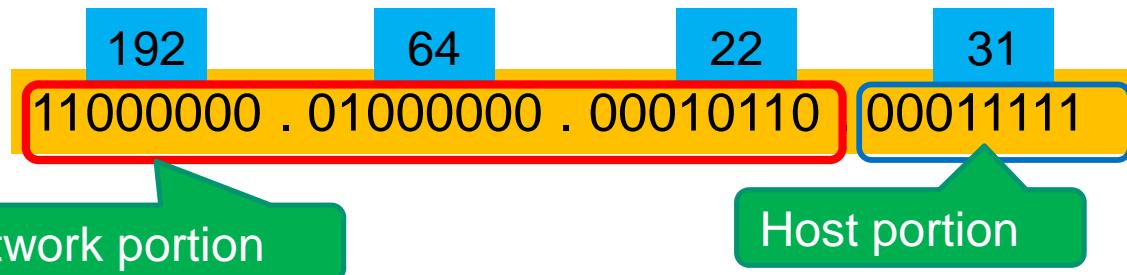
## Subnet Mask

- + 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 bit – subnet mask }**)
    - The individual (unique) address for a host
  - Example: 192.64.22.31,**24**



# IPv4 Addressing

192.64.22.31/24



- † Given the information we find that:
  - The host has **8 bits** →
    - this subnet can have up to  **$2^8 = 256 \text{ hosts}$**
    - 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** – the **network address** → to address the subnet
  - The **last IP** – the **broadcast address** → used to broadcast info throughout the subnet

# IPv4 Addressing

- 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

## Example

### Subnets Mask

Convert these subnet mask into decimal values.

(a) /8       $\begin{array}{cccc} 11111111 & .00000000 & .00000000 & .00000000 \end{array}$

**255.0.0.0**

(b) /16       $\begin{array}{cccc} 11111111 & .11111111 & .00000000 & .00000000 \end{array}$

**255.255.0.0**

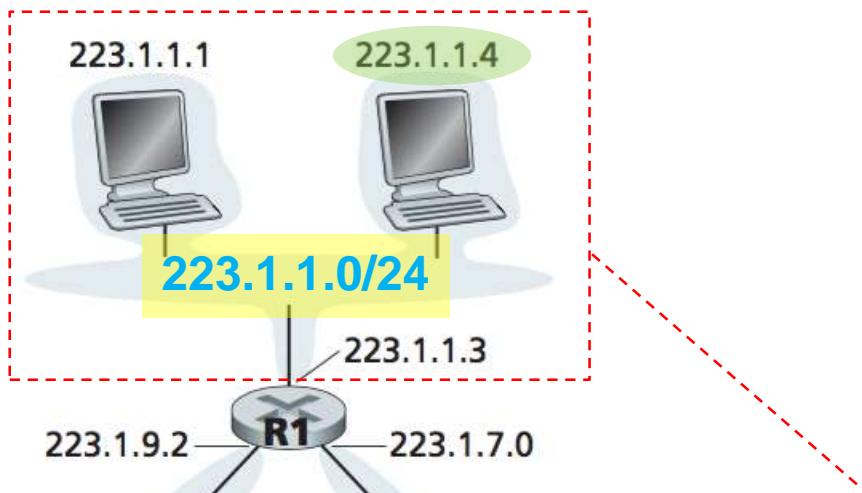
(c) /24       $\begin{array}{cccc} 11111111 & .11111111 & .11111111 & .00000000 \end{array}$

**255.255.255.0**

(d) /25       $\begin{array}{cccc} 11111111 & .11111111 & .11111111 & .10000000 \end{array}$

**255.255.255.128**

# IPv4 Addressing



223. 1 . 1 . 4  
AND 255.255.255.0

## Subnets Mask

*To determine subnet address:*

IP address **AND** subnet mask

### Example:

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

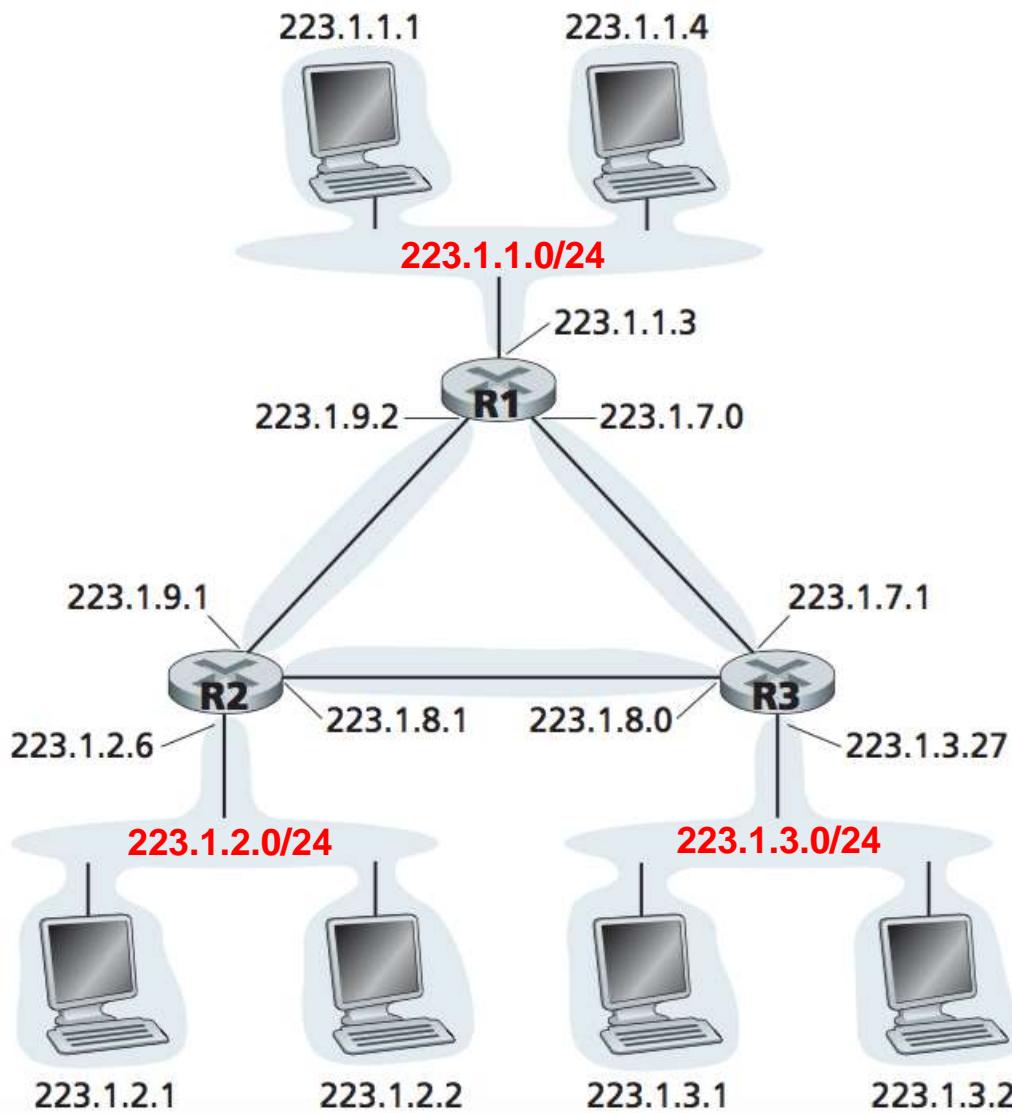
11011111.00000001.00000001.00000100

11111111.11111111.11111111.00000000

11011111.00000001.00000001.00000000

223.1.1.0/24

# IPv4 Addressing



If the subnet mask is  
255.255.255.0, define  
all the subnet addresses  
with the subnet mask.

Solutions:

223.1.x.y  
AND 255.255.255.0

## Example

182.126.103.215/30

182.126.103.215/30

30 bits for the network portion and 2 bits for the host portion

Network portion      Host portion

182.126.103.11010111

AND 255.255.255.11111100

182.126.103.11010100

Network Address 182.126.103.212

182.126.103.213  
182.126.103.214

Usable IP Address

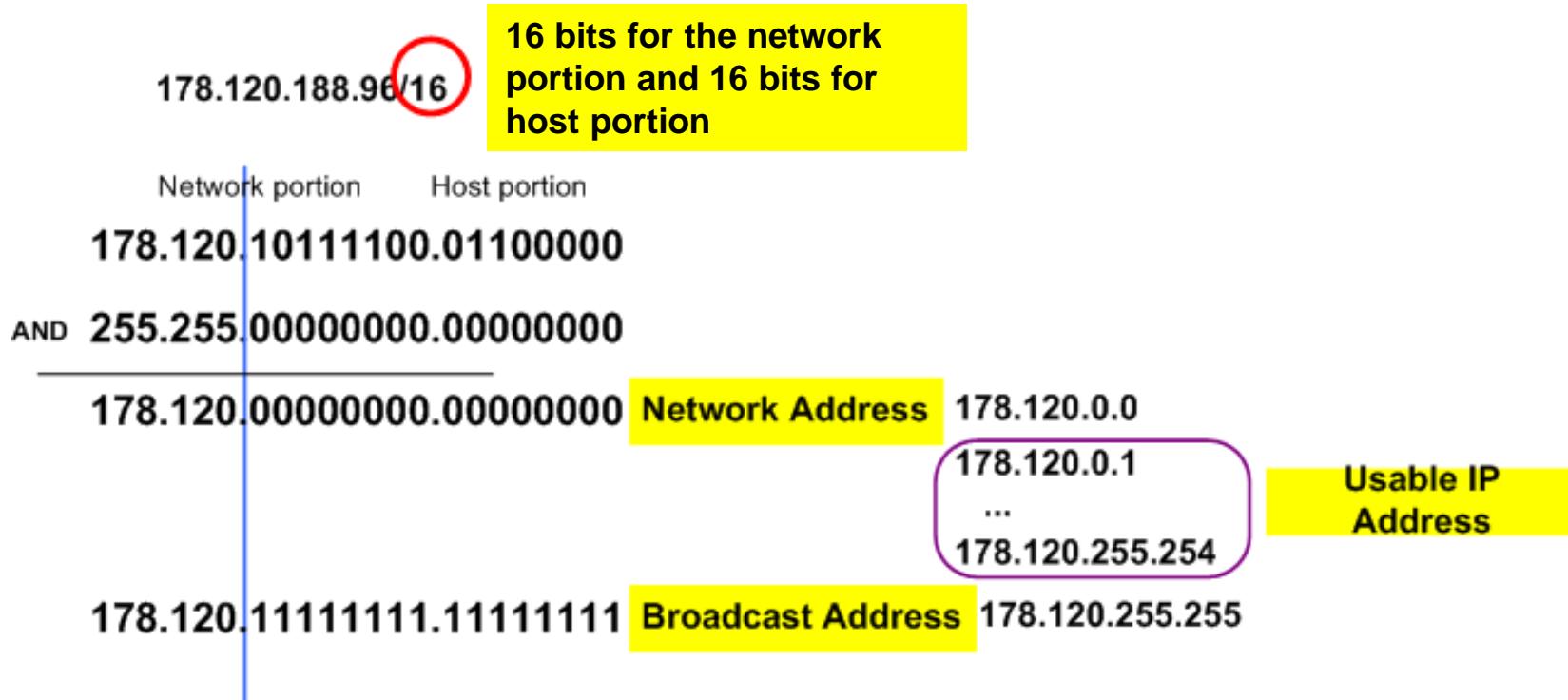
182.126.103.11010111

Broadcast Address 182.126.103.215

Network portion of the address DOES NOT CHANGE.

## Example

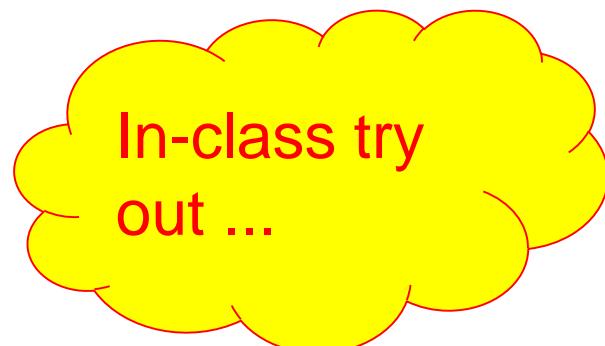
178.120.188.96/16



## Class Exercise

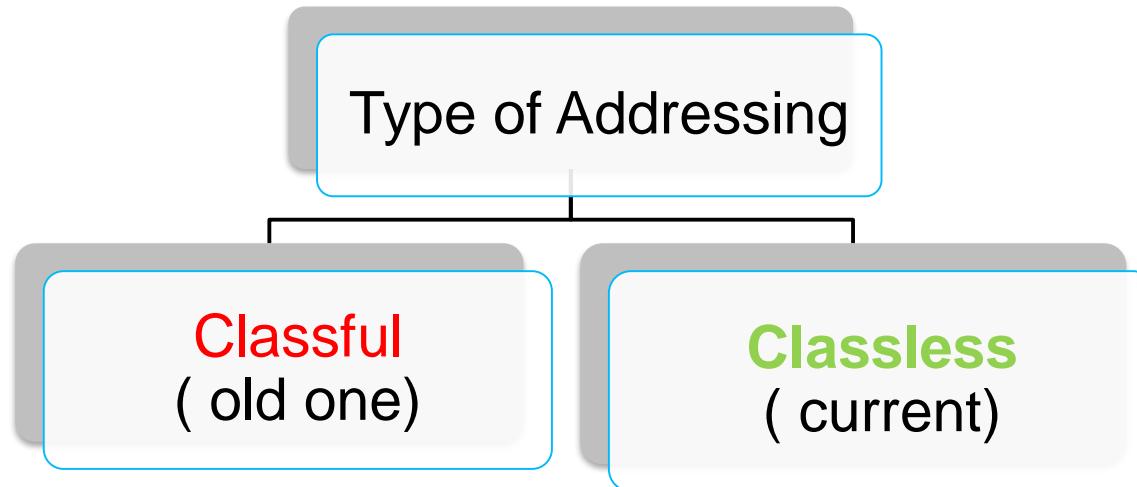
- + Given these IP addresses, determine the network portion and the host portion

- 192.168.12.0/24
- 192.168.12.25/25
- 172.16.32.0/16
- 172.16.32.0/20
- 10.10.10.0/8



In-class try  
out ...

# 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

## Classful addressing

Class	Address Range	Supports
<b>Class A</b>	1.0.0.1 to 126.255.255.254 <b>/8</b> 255.0.0.0	Supports 16 million hosts on each of 127 networks.
<b>Class B</b>	128.1.0.1 to 191.255.255.254 <b>/16</b> 255.255.0.0	Supports 65,000 hosts on each of 16,000 networks.
<b>Class C</b>	192.0.1.1 to 223.255.254.254 <b>/24</b> 255.255.255.0	Supports 254 hosts on each of 2 million networks.
<b>Class D</b>	224.0.0.0 to 239.255.255.255	Reserved for <b>multicast</b> groups.
<b>Class E</b>	240.0.0.0 to 254.255.255.254	Reserved for future use, or Research and Development Purposes.

## 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

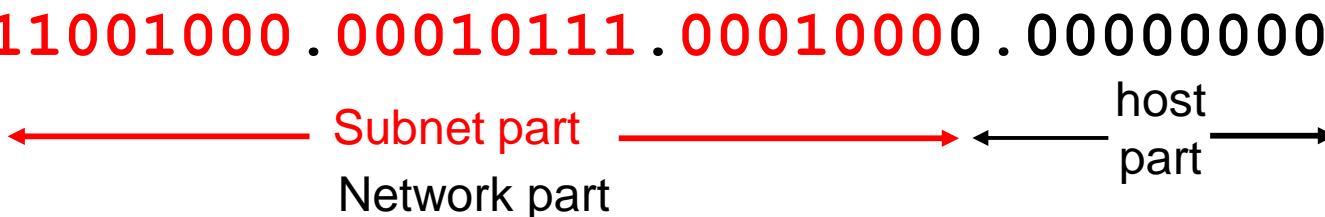
## Classless Inter Domain Routing (CIDR)

- more flexible than original system of internet address scheme ( i.e. classful addressing)
- can **avoid situations** where large numbers of IP addresses are **unused**
- address format: **a . b . c . d/x**

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

200.23.16.0/23

11001000.00010111.00010000.00000000

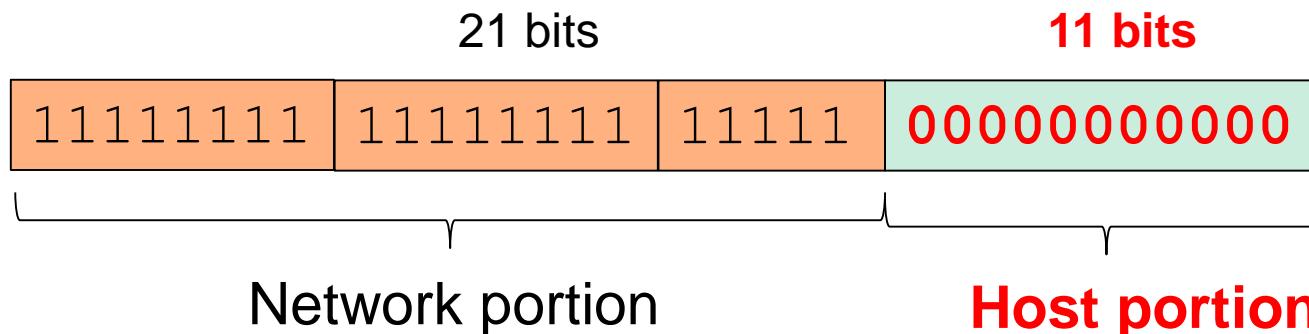


- **subnet portion** of address is **arbitrary length**

## Classless Inter Domain Routing (CIDR)

**Example:**

- An organization needs **2000 hosts** and apply for IP address
- Host needed is 2000  $\rightarrow 2^{11} = 2048$  (i.e. only extra 48)
- Thus an **IP address  $\rightarrow 163.44.224.0/21$**  is given
  - How we get ... **/21** ?  $\rightarrow 32 - 11 = 21$



**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 ( $2^3$ ).

ISP's block:

200.23.16.0/20

→ 11001000.00010111.0001000.00000000

Network portion

need to borrow 3 bits  
from host portion  
to define 8 subnets

Subnet 0:

200.23.16.0/23

→ 11001000.00010111.00010000.00000000

Subnet 1:

200.23.18.0/23

→ 11001000.00010111.00010010.00000000

...

Subnet 7:

200.23.30.0/23

→ 11001000.00010111.00011110.00000000

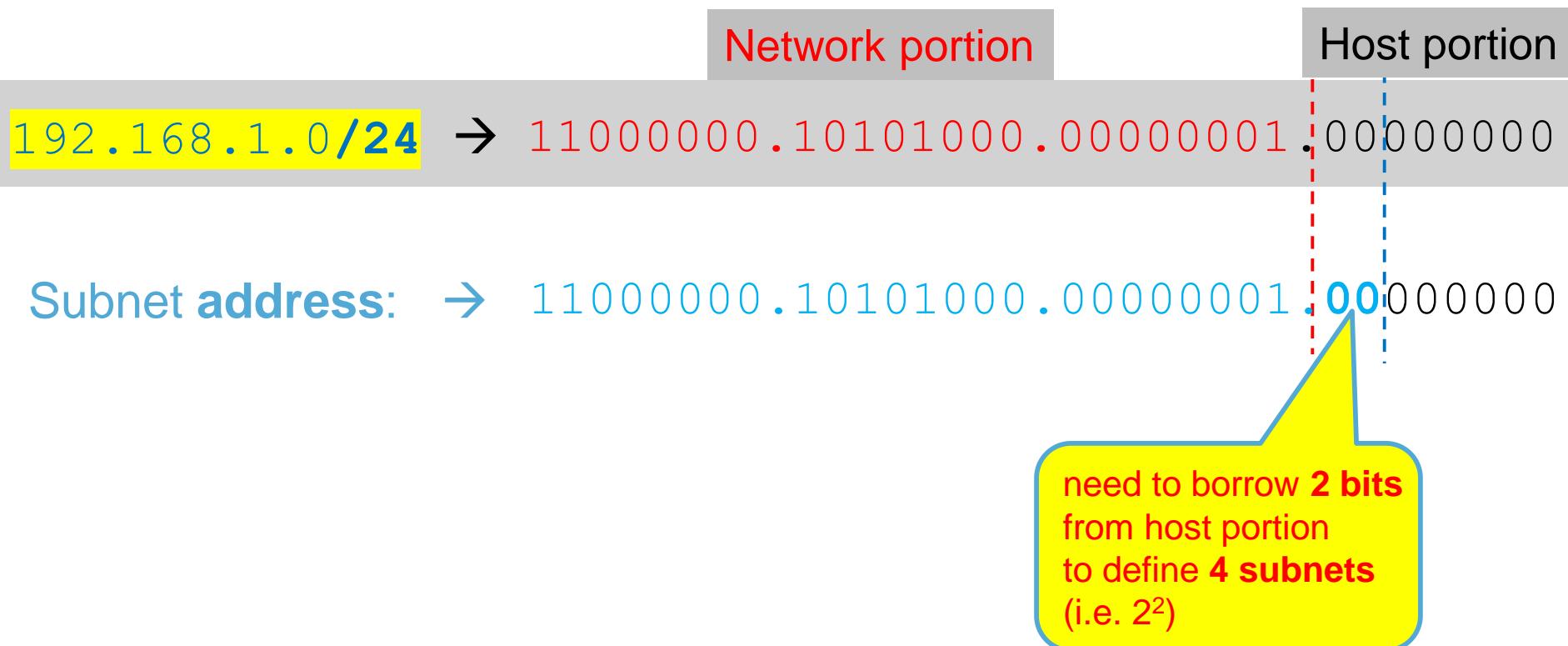
- + Say, Company A (CA) is given the IP address  
**192.168.1.0/24**
- + CA has 4 departments under it:
  - RND,
  - Sales,
  - Finance,
  - Human Resource (HR)
- + CA decided to divide the network according to purpose.  
So, the network needs to have 4 major subnets.

**192.168.1.0/24** to be divided into **4 subnets**

**192.168.1.0/24** → 11000000.10101000.00000001.00000000

## Example: To subnet a small network

**192.168.1.0/24** to be divided into **4 subnets**



## Example: To subnet a small network

**192.168.1.0/24** to be divided into **4 subnets**

Network portion	Host portion
<b>192.168.1.0/24</b> → 11000000.10101000.00000001.00000000	

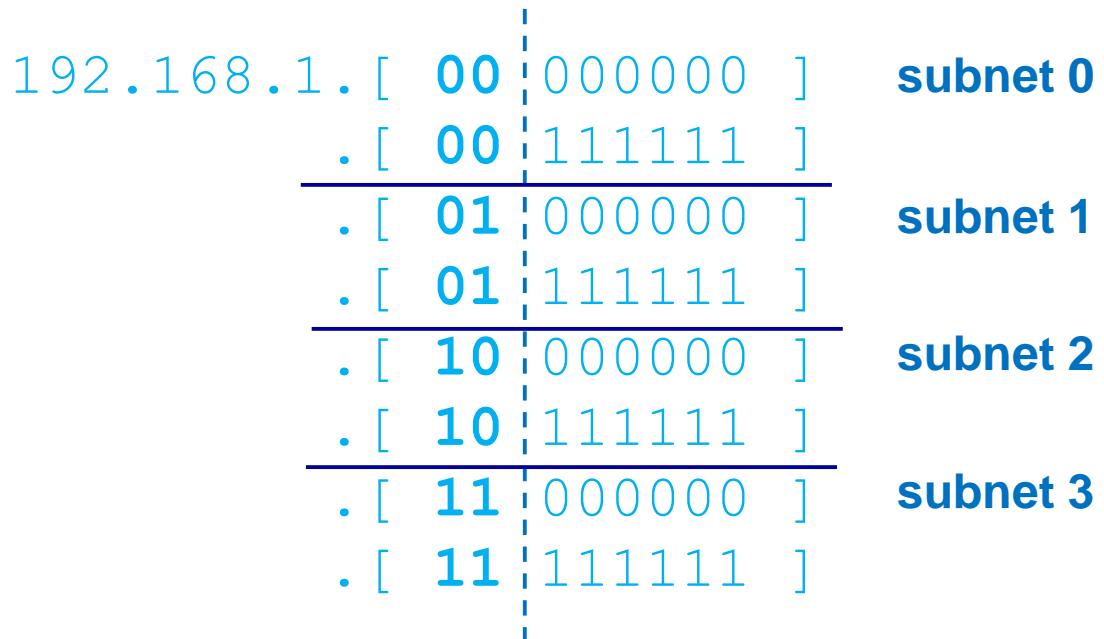
**Subnet address:** → 11000000.10101000.00000001.00000000

**Subnet 0:** → 11000000.10101000.00000001.00000000  
**192.168.1.0/26**

The subnet mask for each subnet  
is /26 OR 255.255.255.192

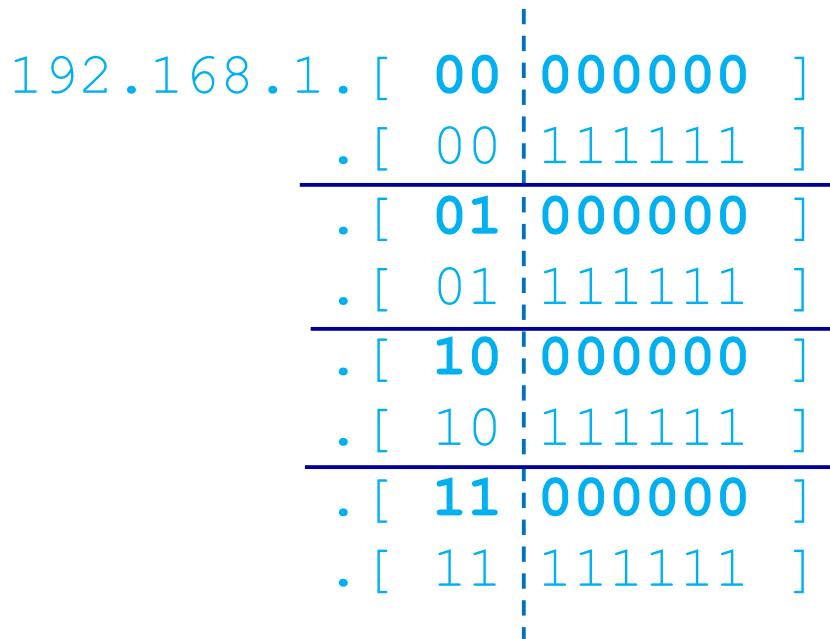
6 bit left for host

Subnet 0:  $\rightarrow 11000000.10101000.00000001.00000000$   
192.168.1.0/26



## Example: To subnet a small network

Subnet 0:  $\rightarrow 11000000.10101000.00000001.00000000$   
192.168.1.0/26



**subnet 0**; network address = 192.168.1.0

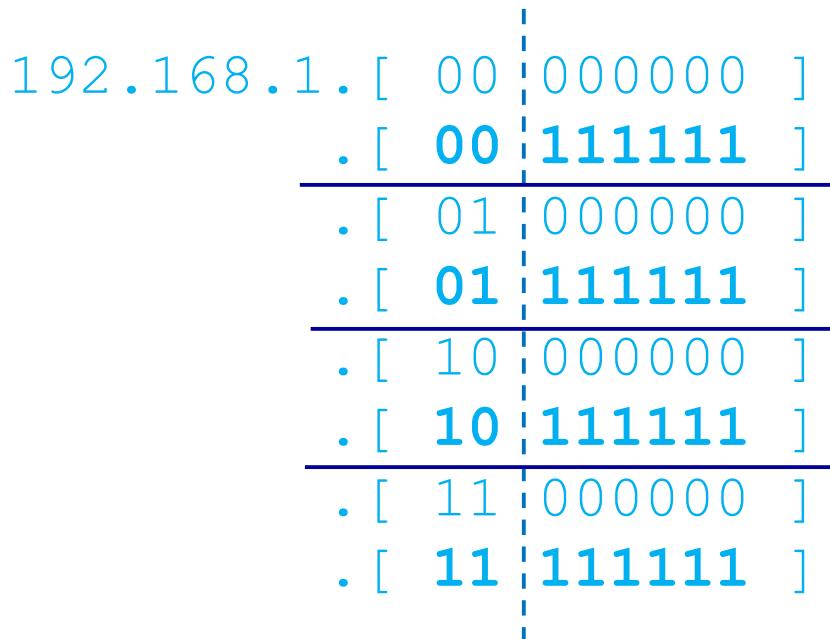
**subnet 1**; network address = 192.168.1.64

**subnet 2**; network address = 192.168.1.128

**subnet 3**; network address = 192.168.1.192

## Example: To subnet a small network

Subnet 0:  $\rightarrow 11000000.10101000.00000001.00000000$   
192.168.1.0/26



**subnet 0; broadcast address = 192.168.1.63**

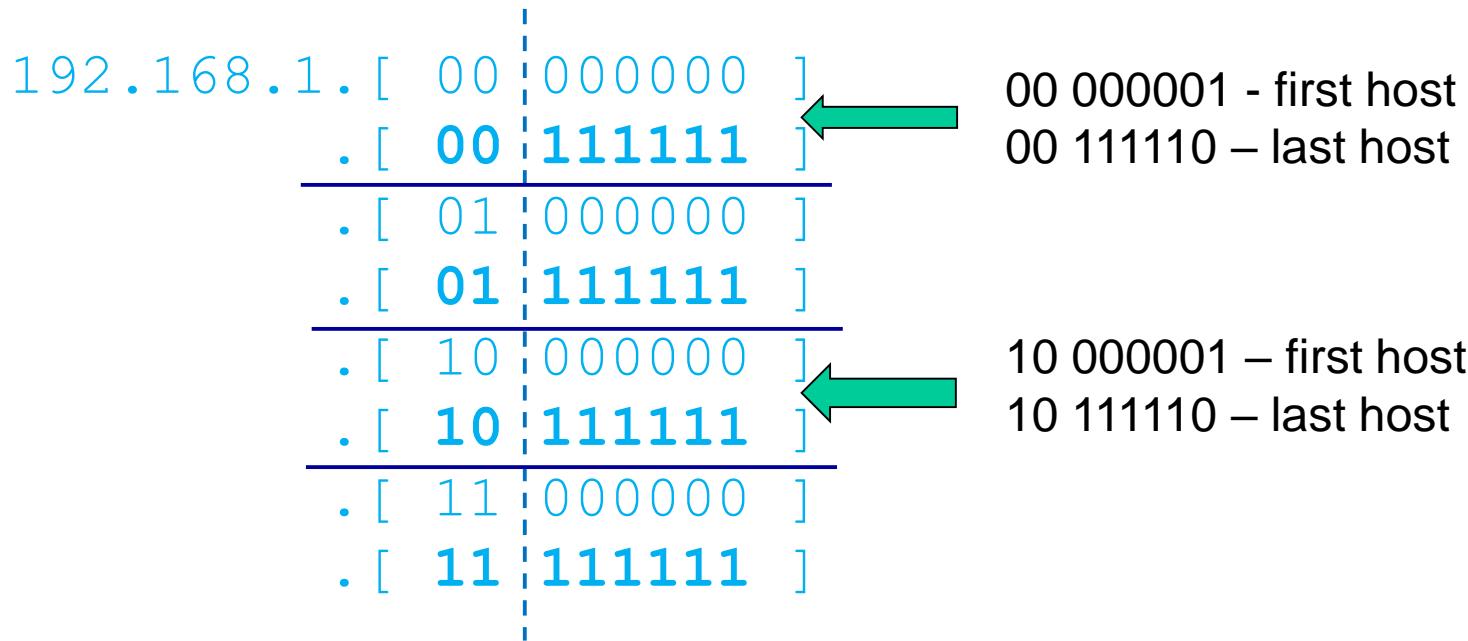
**subnet 1; broadcast address = 192.168.1.127**

**subnet 2; broadcast address = 192.168.1.191**

**subnet 3; broadcast address = 192.168.1.255**

## Example: To subnet a small network

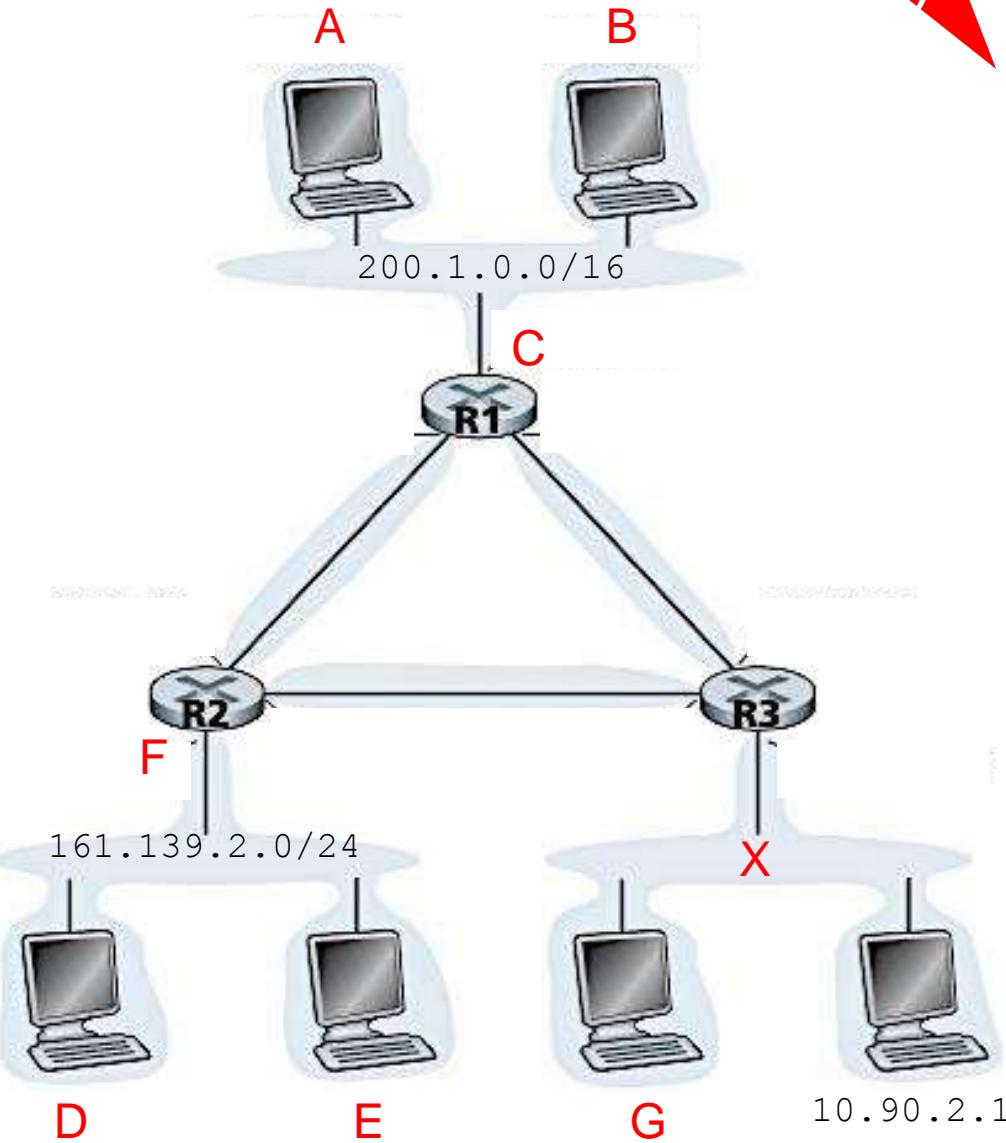
Subnet 0:  $\rightarrow 11000000.10101000.00000001.00000000$   
192.168.1.0/26



## Example: To subnet a small network

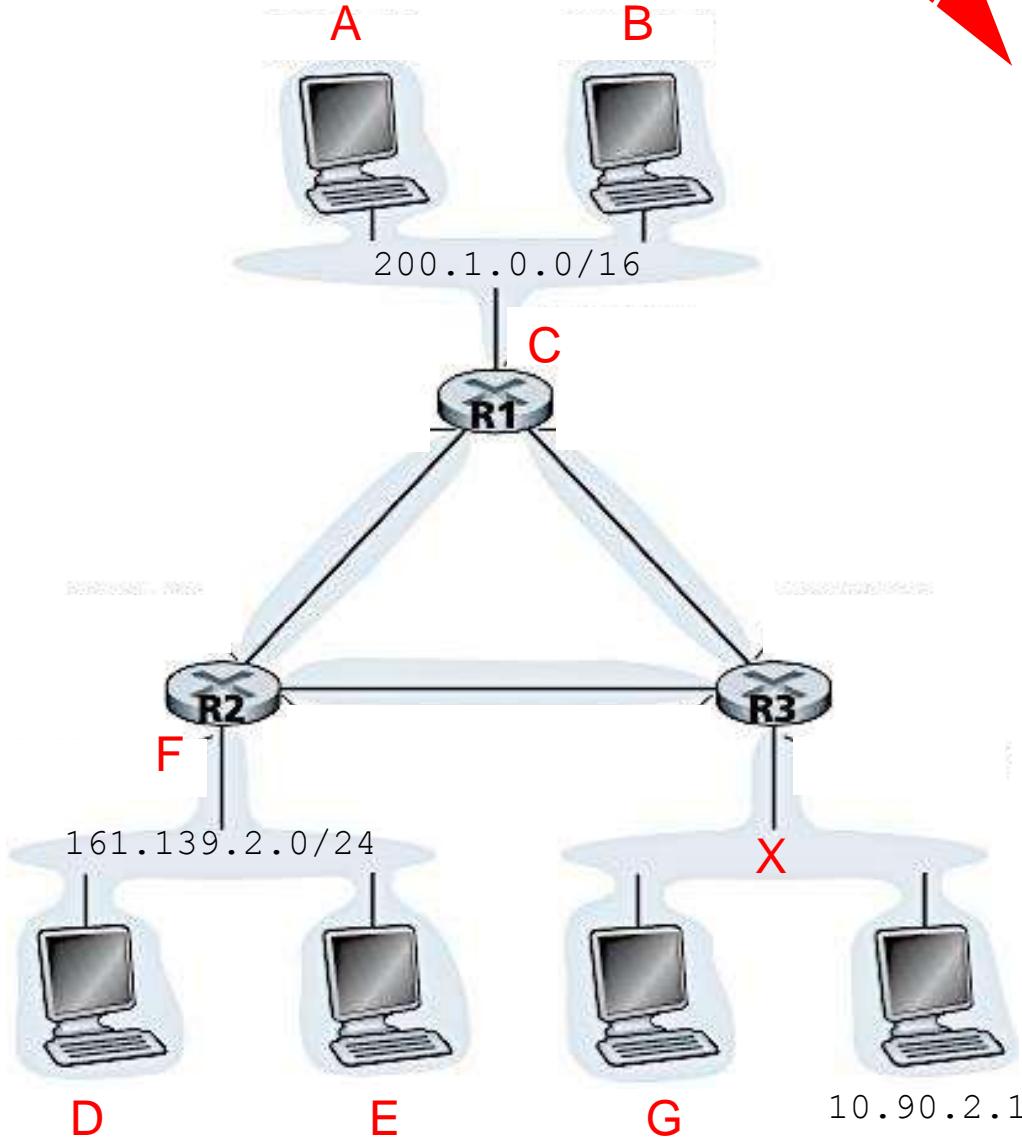
192.168.1.0/24 be divided into 4 subnets we get:

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



Given a network with hosts and interconnecting routers.

- What are the subnet mask addresses for the network that consists of host A, B, D, and E?
- What is the subnet address of network X if the subnet mask address is 255.0.0.0?
- Get a valid IP address for each host A, B, D, E and G.
- Get a valid IP address for each router's interface at C and F.



(a) Net. consists of host A, B:  
**255.255.0.0**

Net. consists of host D, E:  
**255.255.255.0**

$$\begin{array}{r}
 10.90.2.1 \\
 255.0.0.0 \\
 \hline
 10.0.0.0
 \end{array}$$

(c) Valid IP address:

- A: 200.1.0.100
- B: 200.1.50.88
- D: 161.139.2.4
- E: 161.139.2.100
- G: 10.92.5.25

(d) Valid IP address:

- C: 200.1.0.1
- F: 161.139.2.1

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.

- (a) What is the subnet mask for the given IP address?
- (b) How many hosts can be supported?
- (c) List all subnets.
- (d) What is the new subnet mask for each subnet?
- (e) How many hosts can be supported for each 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?

Solutions:

0. 0000 0001

1. 1111 1110

- (a) 11111111.11111111.11110000.00000000  
255.255.240.0

(b) Total hosts  $\rightarrow 2^{12} - 2 = 4096 - 2 = 4094$  hosts

(c) Subnet 5  $\rightarrow 2^3 = 8$  subnet; 3 bits more for subnet;

11001000.00010111.00000 <b>000</b> 0.00000000	(200.23.0.0/23)
11001000.00010111.0000 <b>001</b> 0.00000000	(200.23.2.0/23)
11001000.00010111.0000 <b>010</b> 0.00000000	(200.23.4.0/23)
11001000.00010111.0000 <b>011</b> 0.00000000	(200.23.6.0/23)
11001000.00010111.0000 <b>100</b> 0.00000000	(200.23.8.0/23)
11001000.00010111.0000 <b>101</b> 0.00000000	(200.23.10.0/23)
11001000.00010111.0000 <b>110</b> 0.00000000	(200.23.12.0/23)
11001000.00010111.0000 <b>111</b> 0.00000000	(200.23.14.0/23)

Solutions:

(d) 11111111.11111111.1111**111**0.00000000  
255.255.254.0

(e) Total hosts in subnet  $\rightarrow 2^9 = 512 - 2 = 510$  hosts;

(f) Subnet #3 = 200.23.6.0/23

11001000.00010111.0000 <b>011</b> 0.00000000	(200.23.6.0/23)
11001000.00010111.0000 <b>011</b> 0.00000001	(200.23.6.1/23)
11001000.00010111.0000 <b>011</b> 0.00000010	(200.23.6.2/23)
11001000.00010111.0000 <b>011</b> 0.00000011	(200.23.6.3/23)
11001000.00010111.0000 <b>011</b> 0.00000100	(200.23.6.4/23)
11001000.00010111.0000 <b>011</b> 0.00000101	(200.23.6.5/23)

(g) Need = 5 subnets, generated subnets = 8, extra =  $8 - 5 = 3$  subnets

# Extra Exercise (Same/Equal Size) : 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**

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  **$2^x = 4 \rightarrow x = 2$  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**

192 . 168 . 1 . 0  
**11000000 . 10101000.00000001.00000000**

Borrow from  
this end – THE  
LEFTMOST

**STEP 4 : Arrange the new subnets**

192 . 168 . 1 . 0  
**11000000 . 10101000.00000001.00000000 [0]**

.00111111 [63]

---

.01000000 [64]

.01111111 [127]

---

.10000000 [128]

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

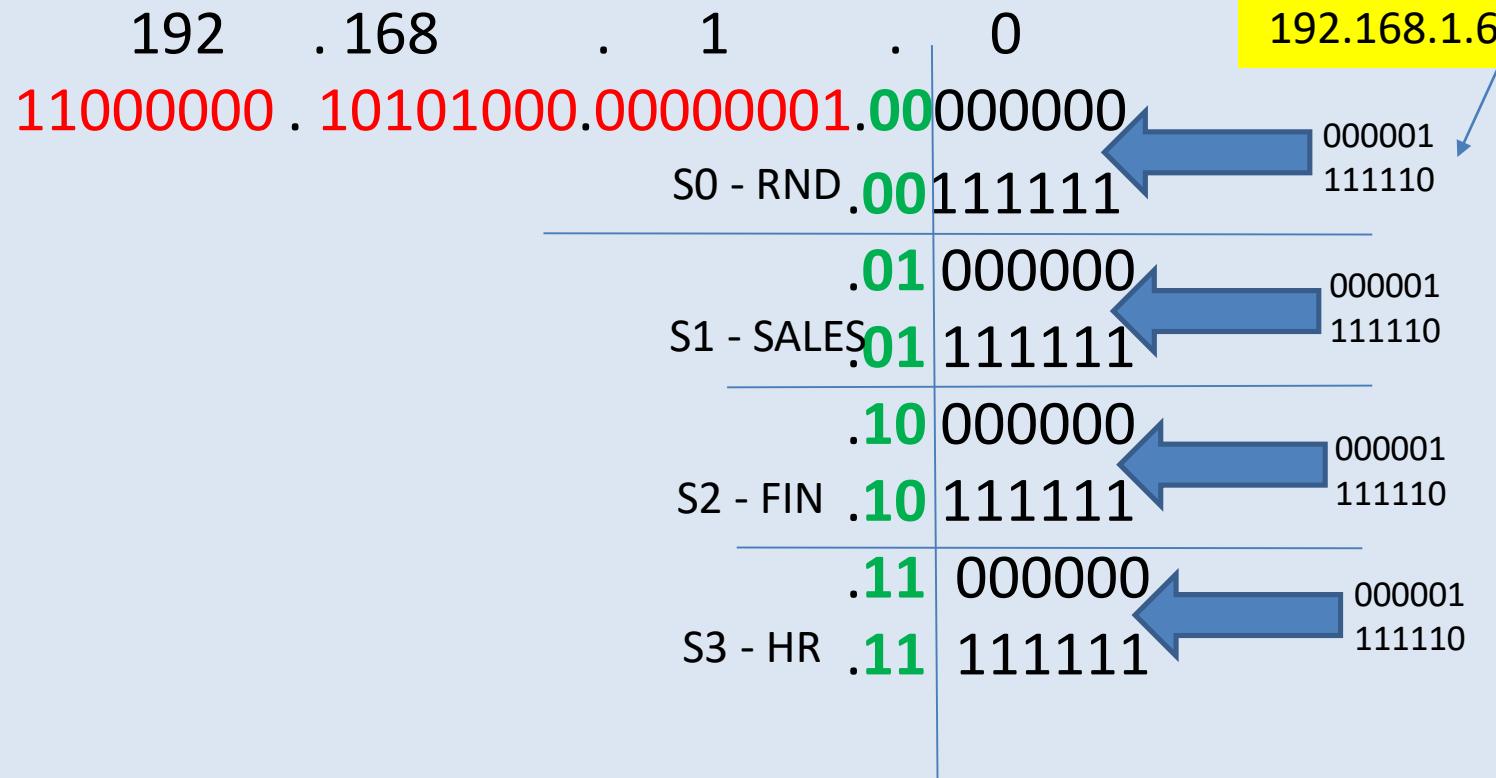
**Network address 192.168.1.0/24  
to divide into SAME/EQUAL SIZED 4 dept : RND, SALES, FIN, HR**

**STEP 4 : Arrange the new subnets**



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

**STEP 5 : Assign the new subnets**



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

**Q:** How does an **ISP** get block of addresses?

**A:** **ICANN:** Internet Corporation for Assigned Names and Numbers

(<http://www.icann.org/>)

- allocates addresses
- manages DNS
- assigns domain names,  
resolves disputes
- ICANN allocates addresses to regional Internet registries
  - like ARIN, RIPE, APNIC, and LACNIC
  - ...which handle the allocation/management of addresses within their regions.



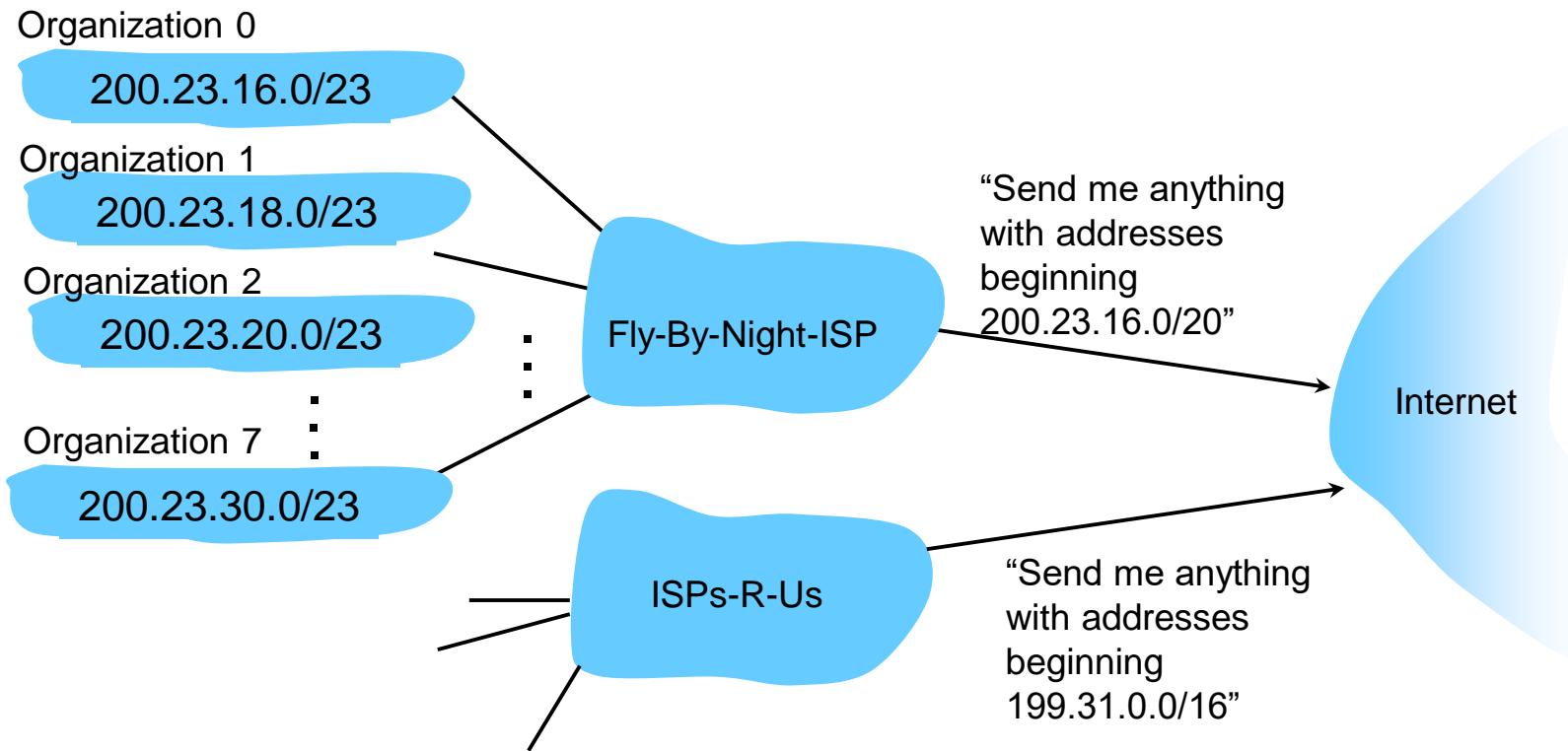
**Q:** How does a **host** get IP address?

**A:** host can get IP address from

- local **ISP** (e.g. Telekom, Celcom, U, DiGi, Maxis, etc)
- local organization like UTM (...and UTM also subscribe to local ISP, i.e. Time Telekom )

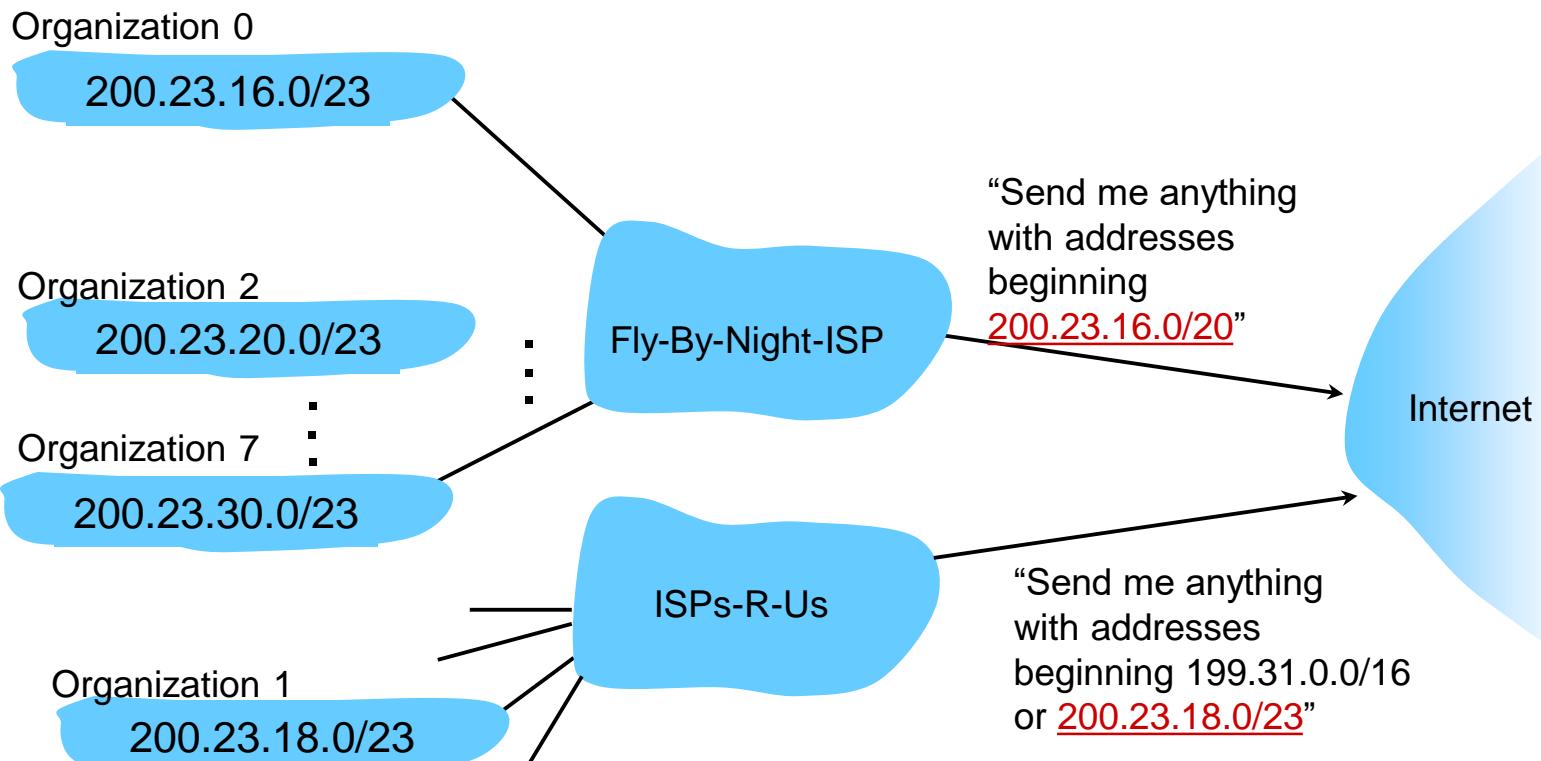
# Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



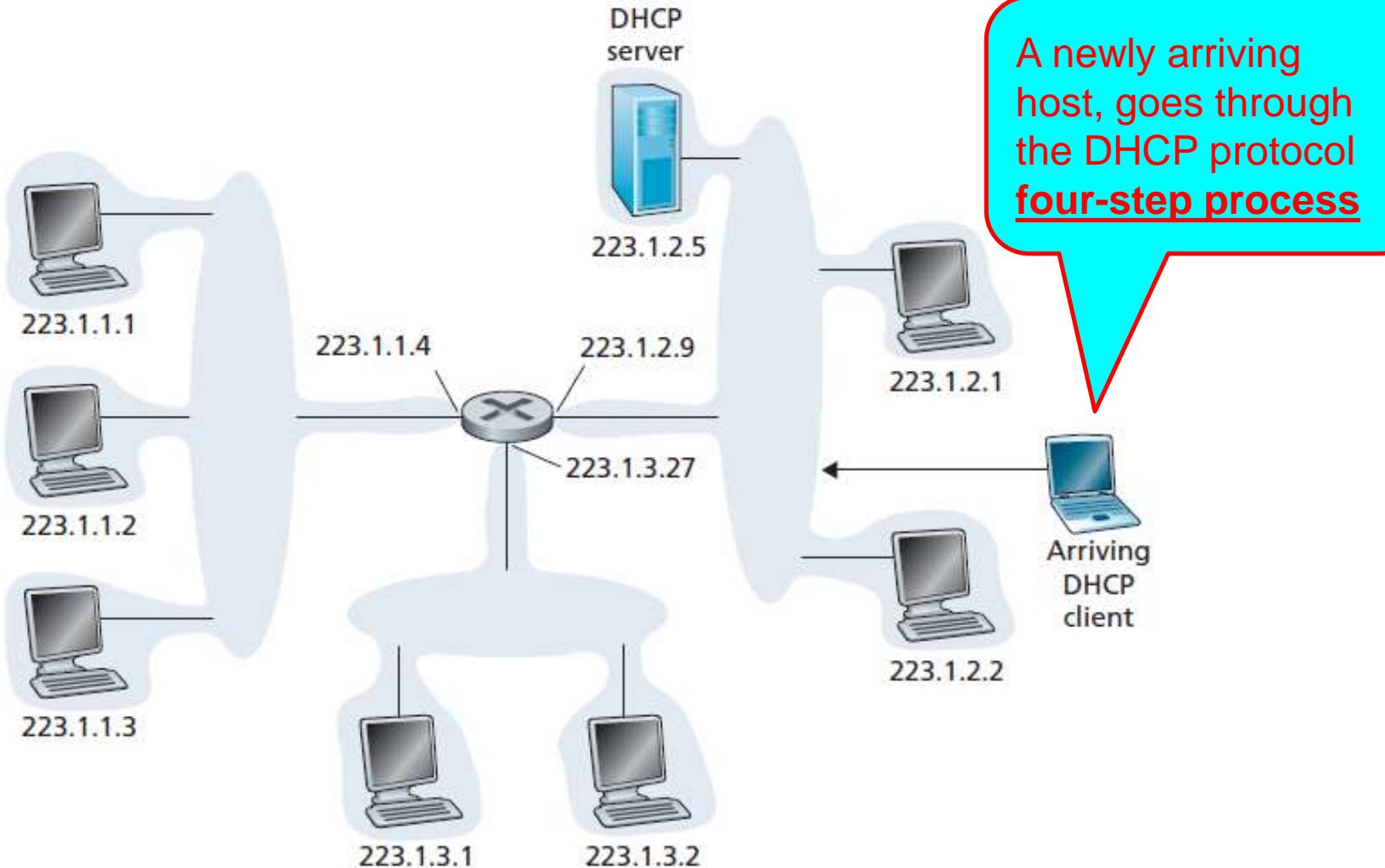
# Hierarchical addressing: more specific routes

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



- ⊕ 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.

- ⊕ **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)
  - support for mobile users who want to join network
- ⊕ DHCP is a *plug-and-play* protocol
  - Makes network admin job easier
  - Makes network mobility easier too



⊕ DHCP 4 steps process

- *DHCP server discovery*
- *DHCP server offer*
- *DHCP request*
- *DHCP ACK*

## + *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 > **I DHCP servers**

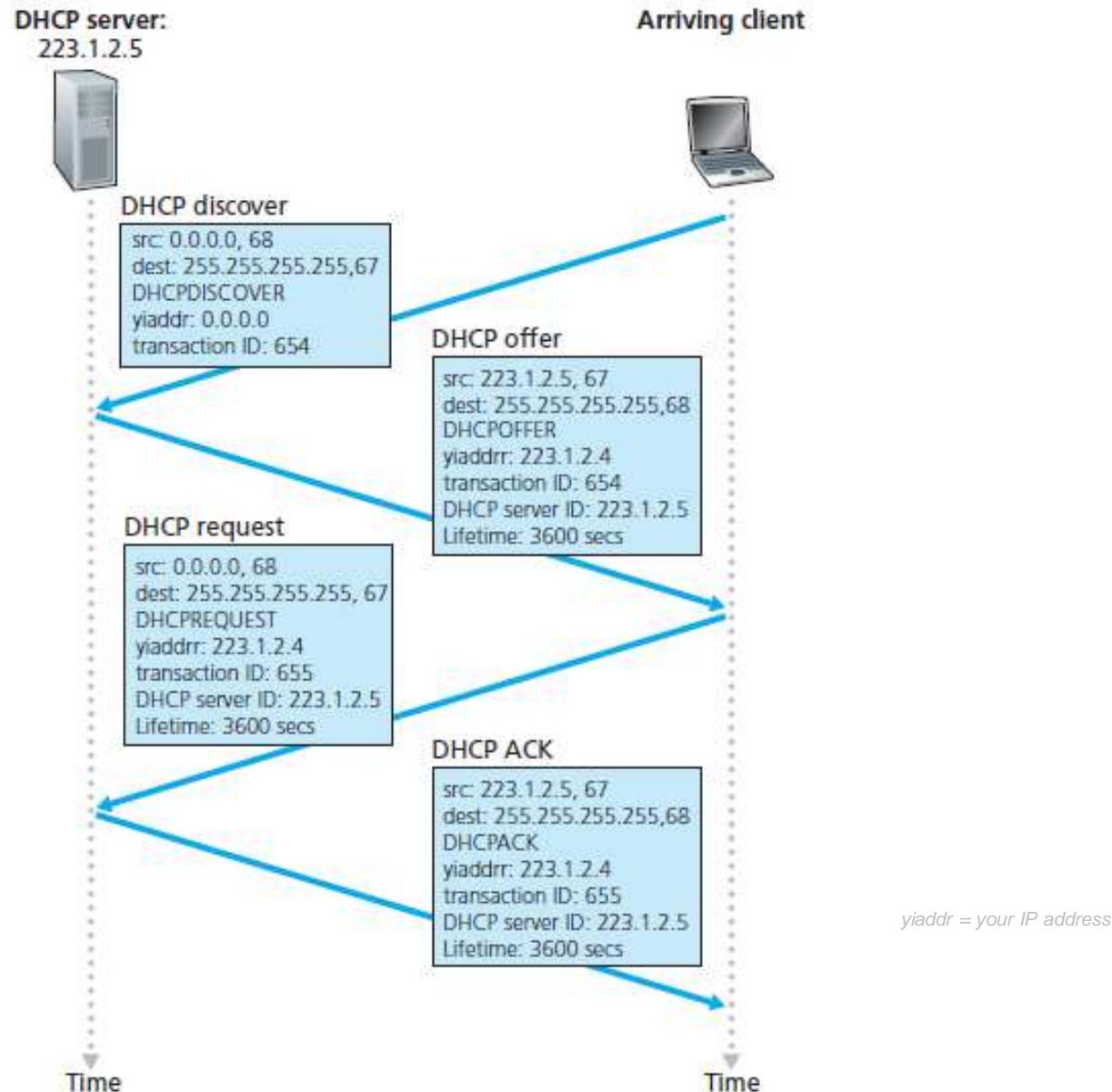
⊕ *DHCP request*

- Client choose from among the offers
- Respond to its selected offer with a **DHCP request message**.

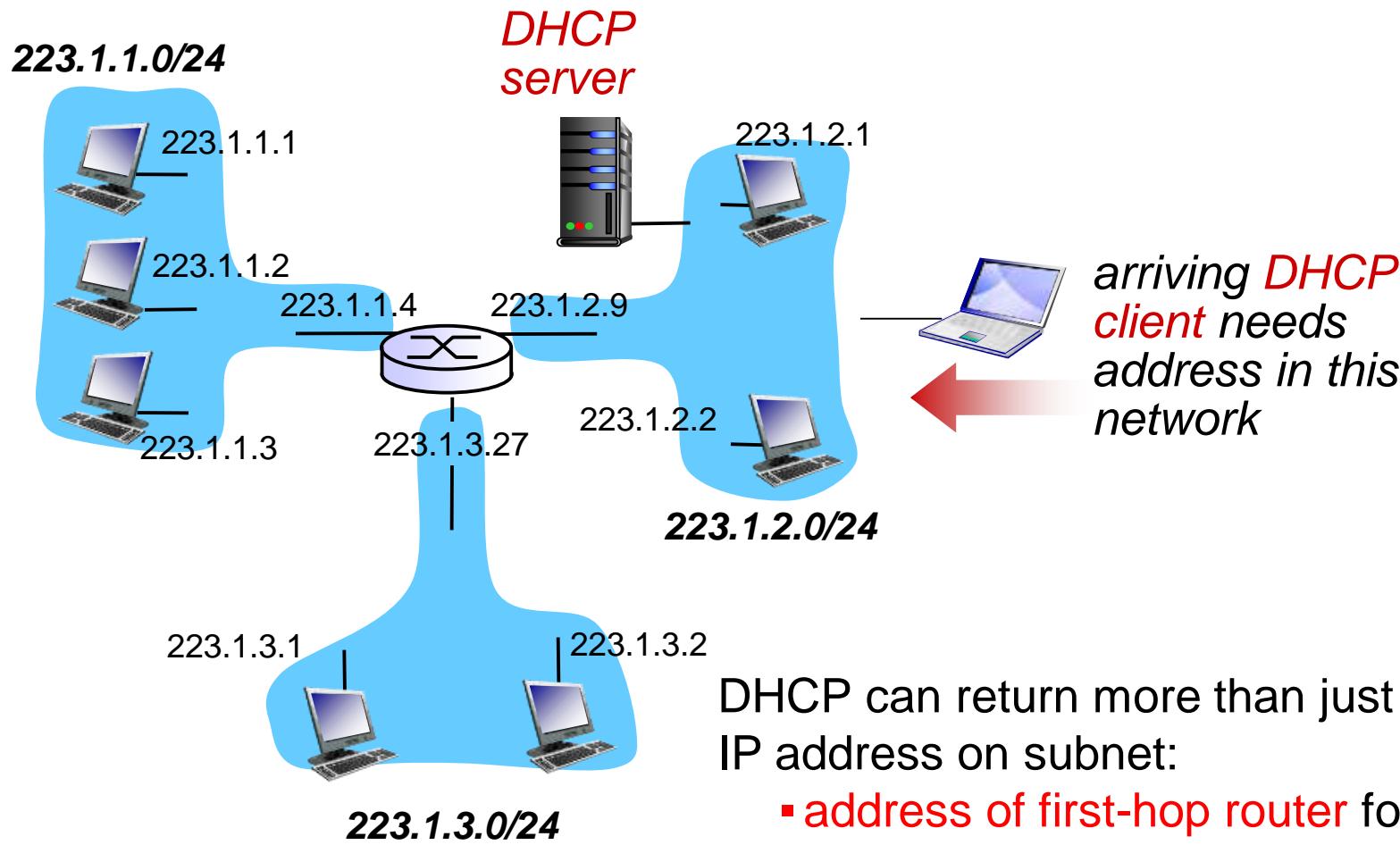
⊕ *DHCP ACK*

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

## DHCP – 4 Steps Process



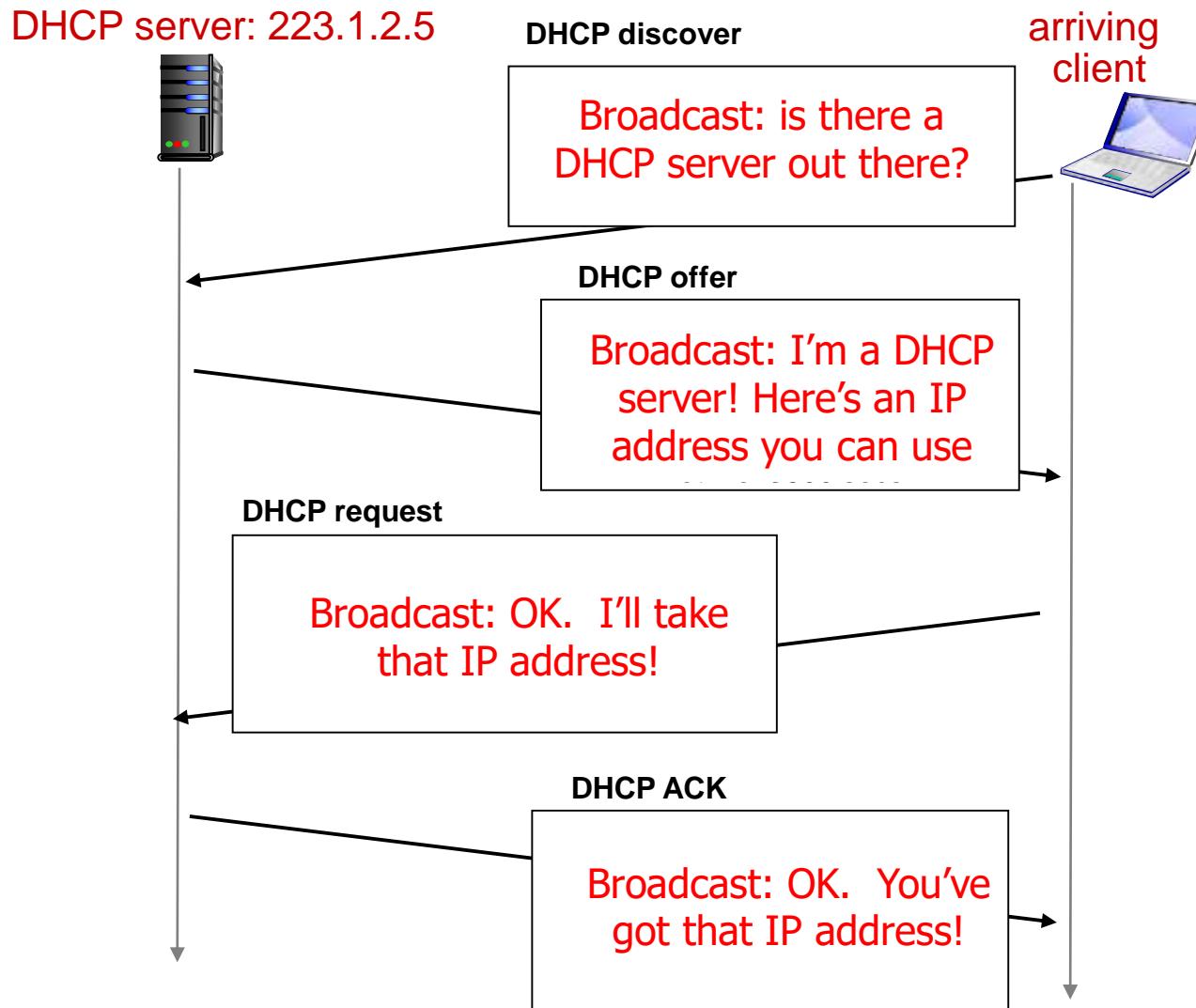
# DHCP client-server scenario



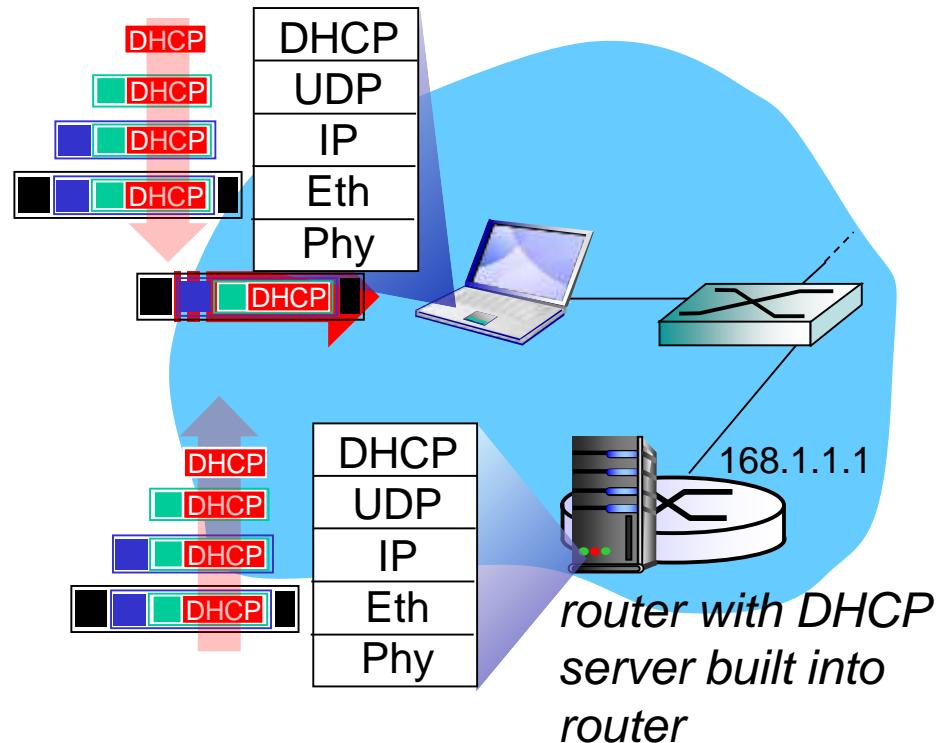
DHCP can return more than just allocated IP address on subnet:

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

# DHCP client-server scenario

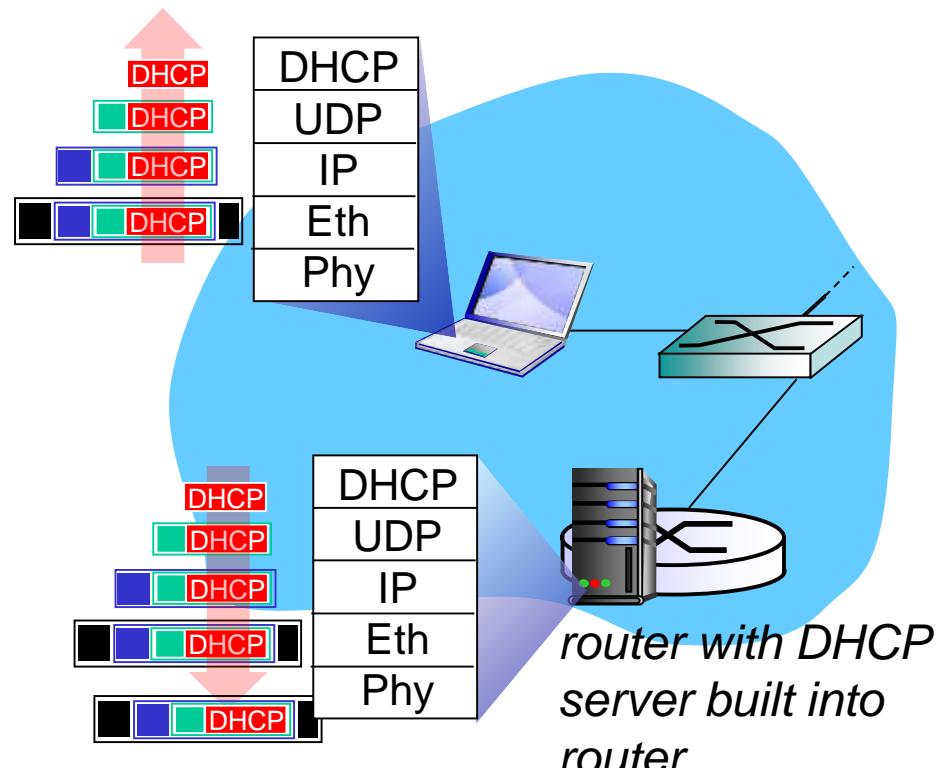


# DHCP: example



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

# DHCP: example



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

# DHCP: Example Wireshark output (home LAN)

request

reply

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

.....

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

The screenshot shows two windows side-by-side. On the left is the 'Wireless Network Connection Properties' window for 'Intel(R) PRO/Wireless 3945ABG Net'. It has tabs for General, Wireless Networks, and Advanced. Under 'Connect using:', 'Client for Microsoft Networks', 'File and Printer Sharing for Microsoft Networks', 'QoS Packet Scheduler', and 'Internet Protocol (TCP/IP)' are listed, with 'Internet Protocol (TCP/IP)' highlighted by a red box. Below these are 'Install...', 'Uninstall', and 'Properties...' buttons, with 'Properties...' highlighted by a red box. A 'Description' section follows, stating: 'Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication...'. On the right is the 'Internet Protocol (TCP/IP) Properties' window. It has tabs for General and Alternate Configuration. The General tab contains text: 'You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.' Below this are two radio button options: 'Obtain an IP address automatically' (selected) and 'Use the following IP address:' (disabled). Under 'Use the following IP address:' are fields for 'IP address', 'Subnet mask', and 'Default gateway', each with a corresponding disabled text input field. At the bottom are two more radio button options: 'Obtain DNS server address automatically' (selected) and 'Use the following DNS server addresses:' (disabled).

### DHCP Server Assignment

The screenshot shows two windows side-by-side. On the left is the 'Wireless Network Connection Status' window. It has tabs for General and Support. Under 'Connection status', it shows 'Address Type: Assigned by DHCP'. Below this are three red-bordered fields: 'IP Address: 192.168.1.4', 'Subnet Mask: 255.255.255.0', and 'Default Gateway: 192.168.1.1'. At the bottom is a 'Details...' button. On the right is the 'Network Connection Details' window. It has a table titled 'Network Connection Details:' with columns 'Property' and 'Value'. The table data is as follows:

Property	Value
Physical Address	00-1C-BF-B8-8B-61
IP Address	192.168.1.4
Subnet Mask	255.255.255.0
Default Gateway	192.168.1.1
DHCP Server	192.168.1.1
Lease Obtained	4/20/2013 7:21:59 PM
Lease Expires	4/21/2013 7:21:59 PM
DNS Server	192.168.1.1
WINS Server	

# Chapter 4: outline

## 4.1 Overview of Network layer

- data plane
- control plane

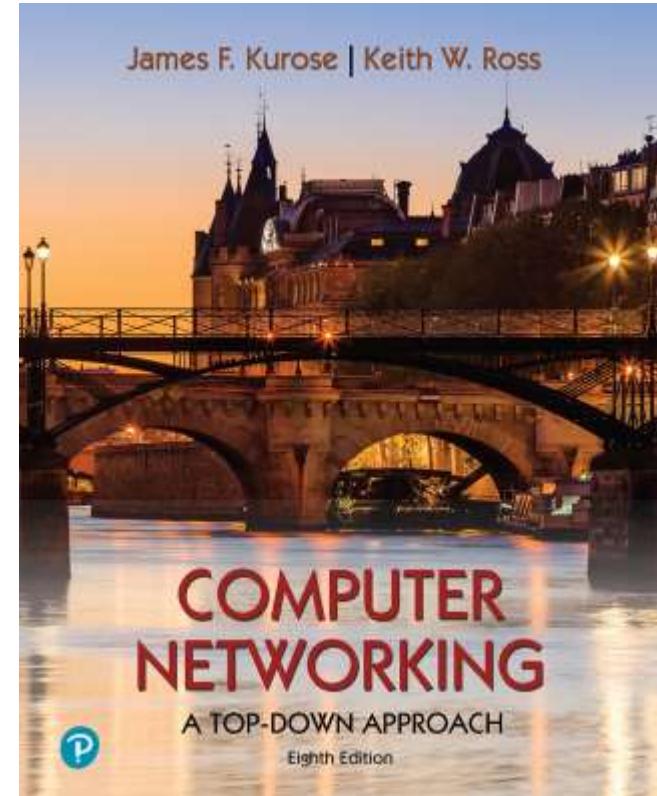
## 4.2 What's inside a router

input ports, switching, output ports

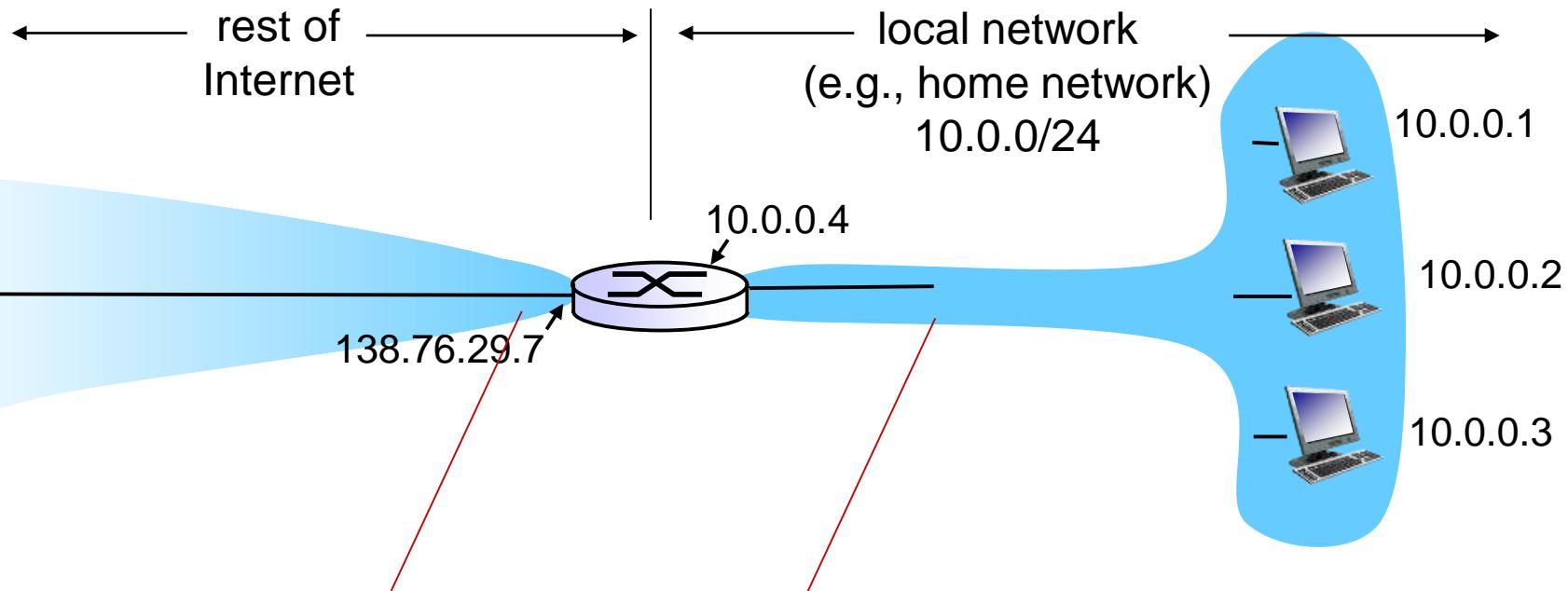
## 4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- **network address translation (NAT)**
- IPv6

## 4.4 Generalized Forwarding and SDN



# NAT: network address translation



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

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

# NAT: network address translation

***motivation:*** local network uses just one IP address as far as outside world is concerned:

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

# NAT: network address translation

*implementation:* NAT router must:

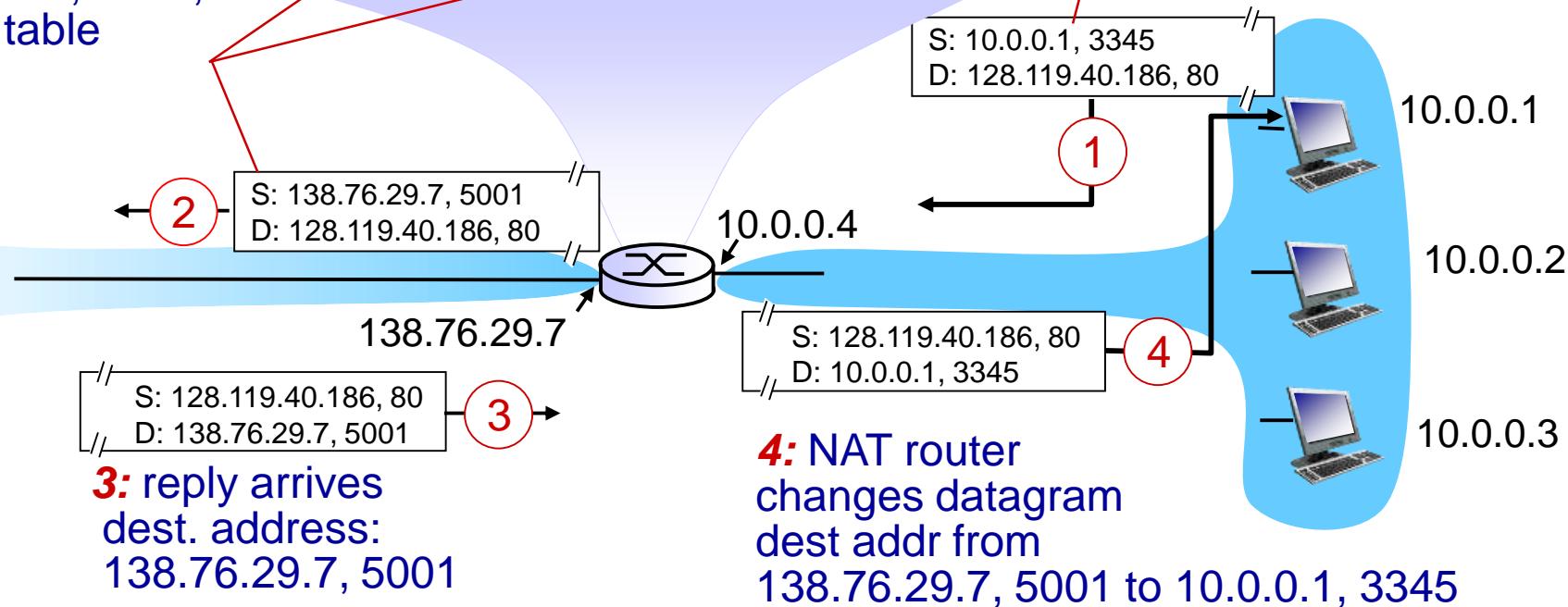
- *outgoing datagrams:* replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)  
    . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT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

# NAT: network address translation

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

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



\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# NAT: network address translation

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

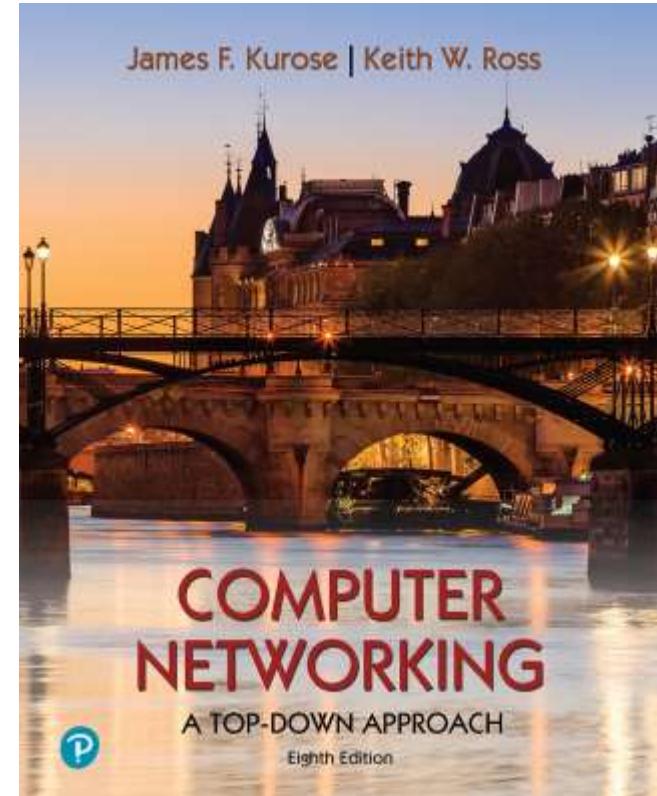
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- **IPv6**

## 4.4 Generalized Forwarding and SDN



# IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated. IPv6 provided a **larger IP address space** → more IP address
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

# 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

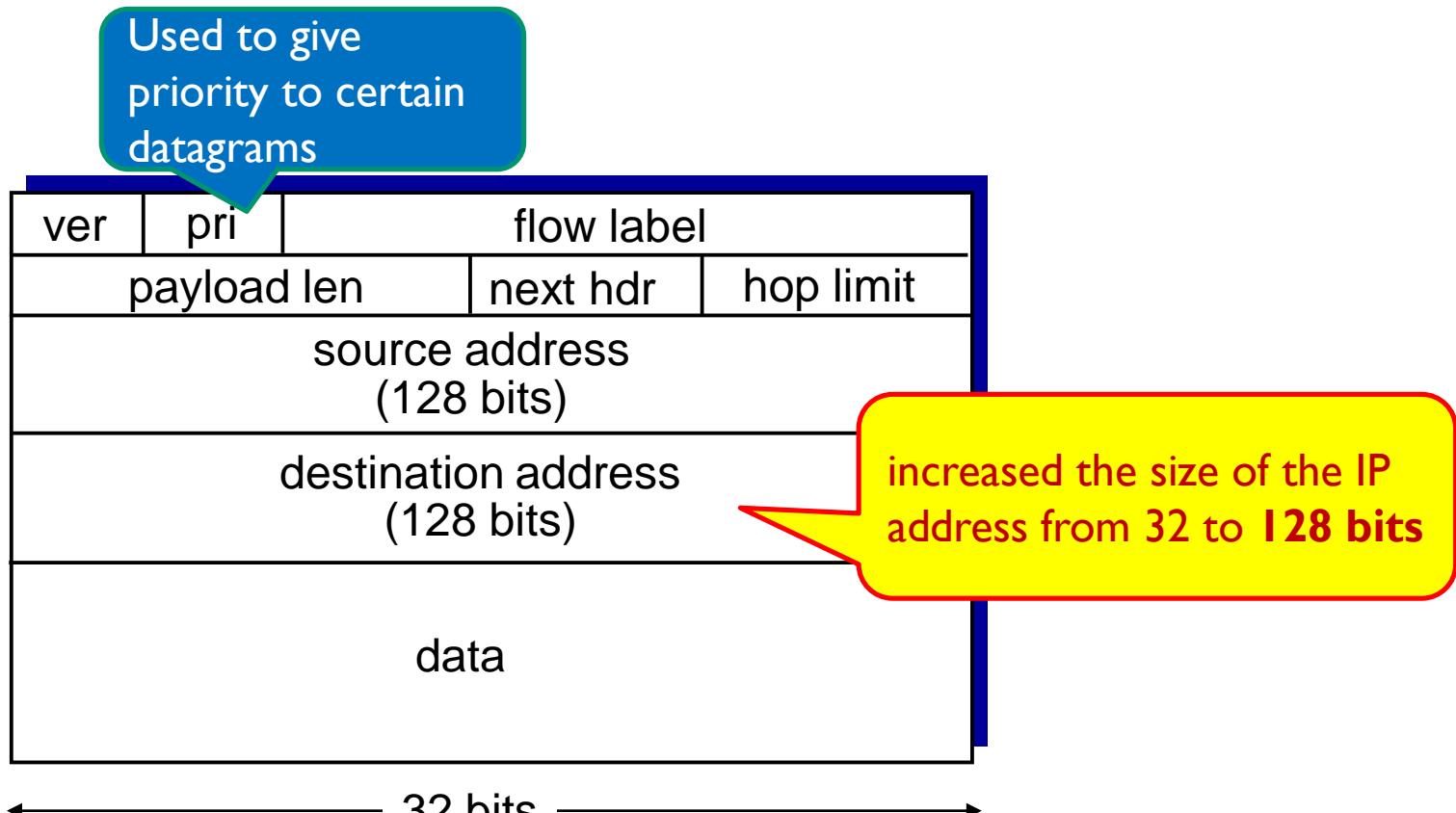
datagrams from certain applications (for example, ICMP) over others

*Payload length:*

What size of data sent (not including header)

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

# IPv6 datagram format



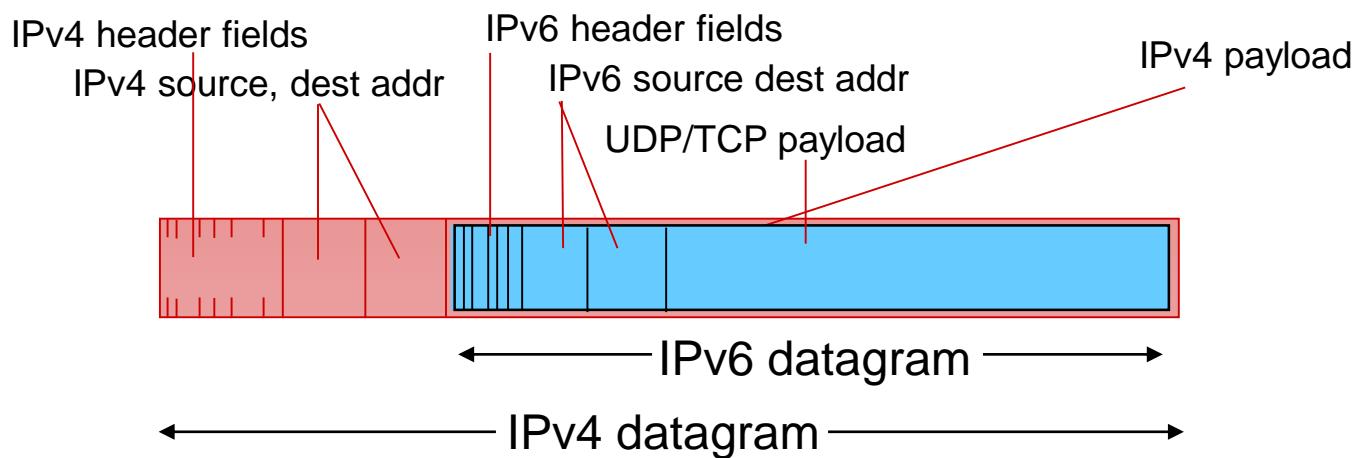
Network Layer: Data Plane 4-  
126

# 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

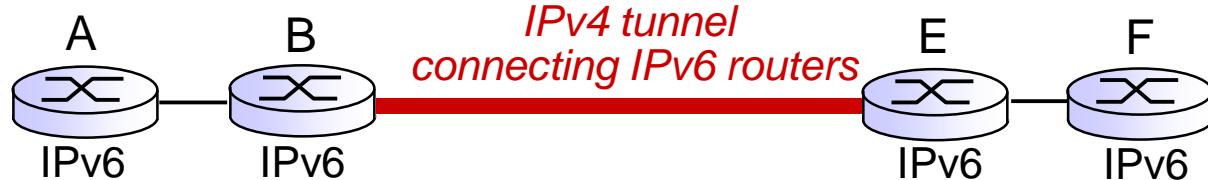
# Transition from IPv4 to IPv6

- 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

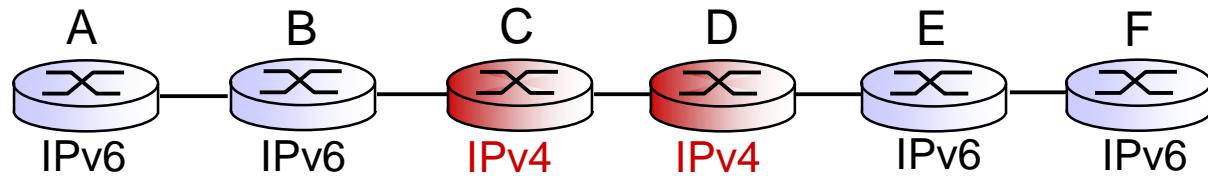


# Tunneling

logical view:

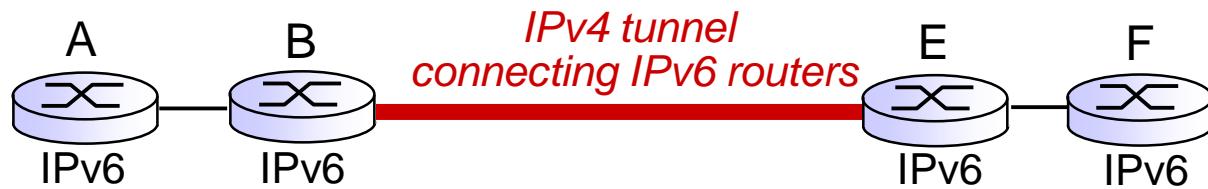


physical view:

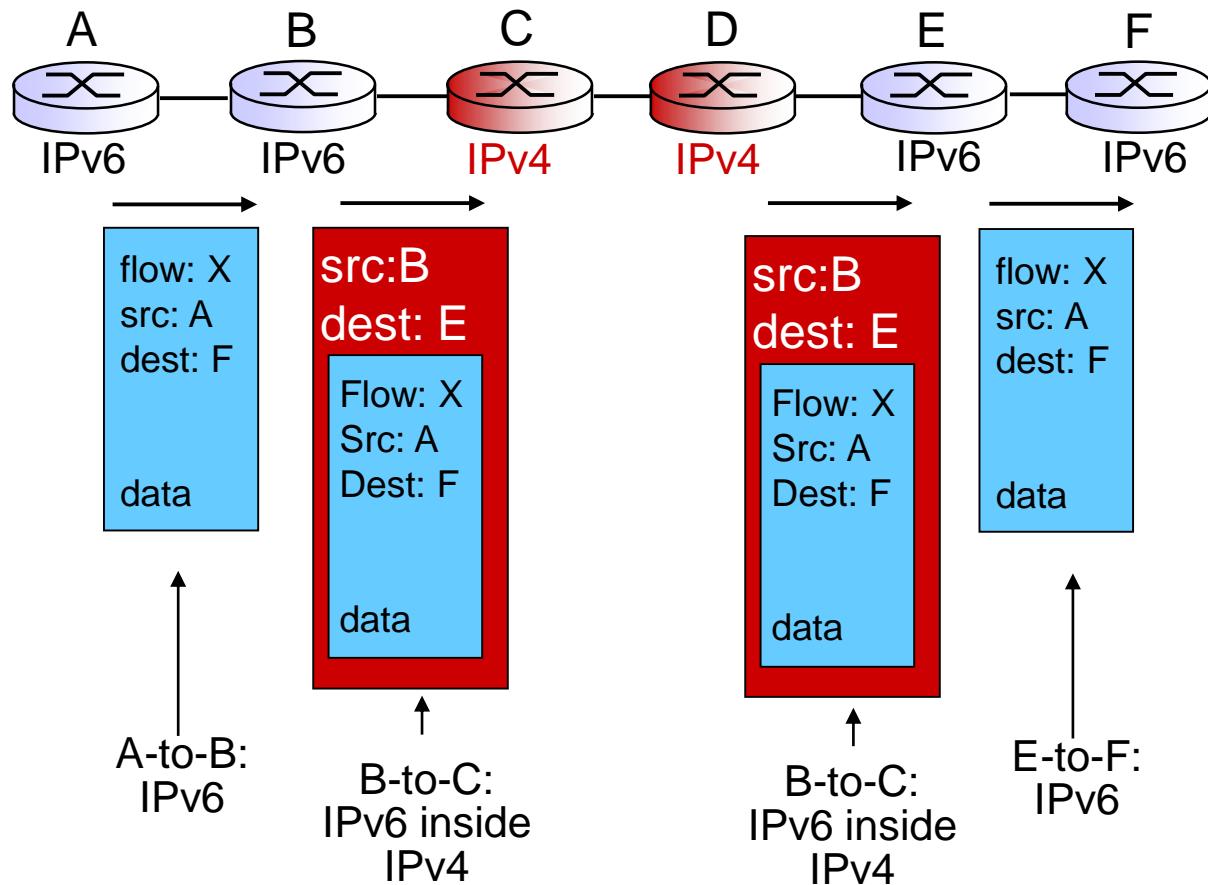


# Tunneling

logical view:



physical view:



# IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, ...
  - *Why?*

# Chapter 4: outline

## 4.1 Overview of Network layer

- data plane
- control plane

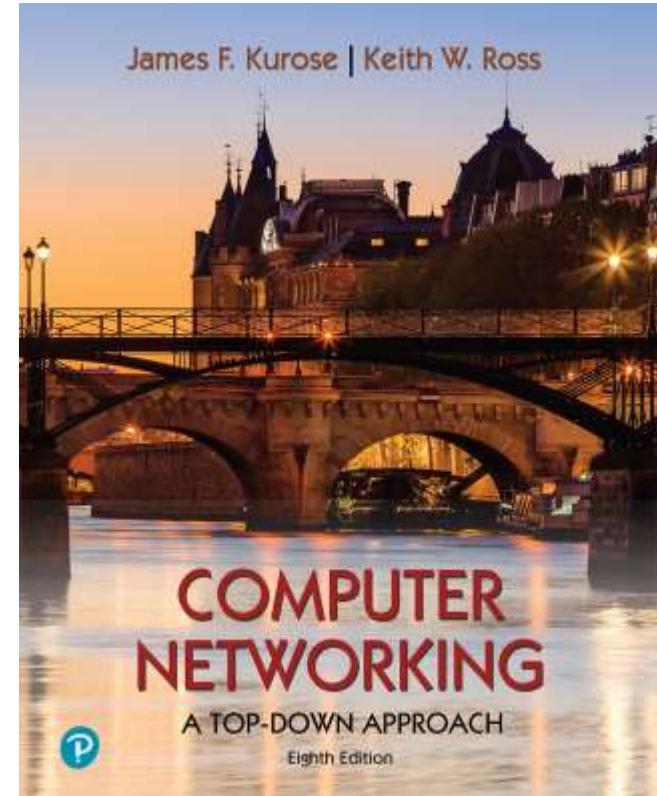
## 4.2 What's inside a router

input ports, switching, output ports

## 4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

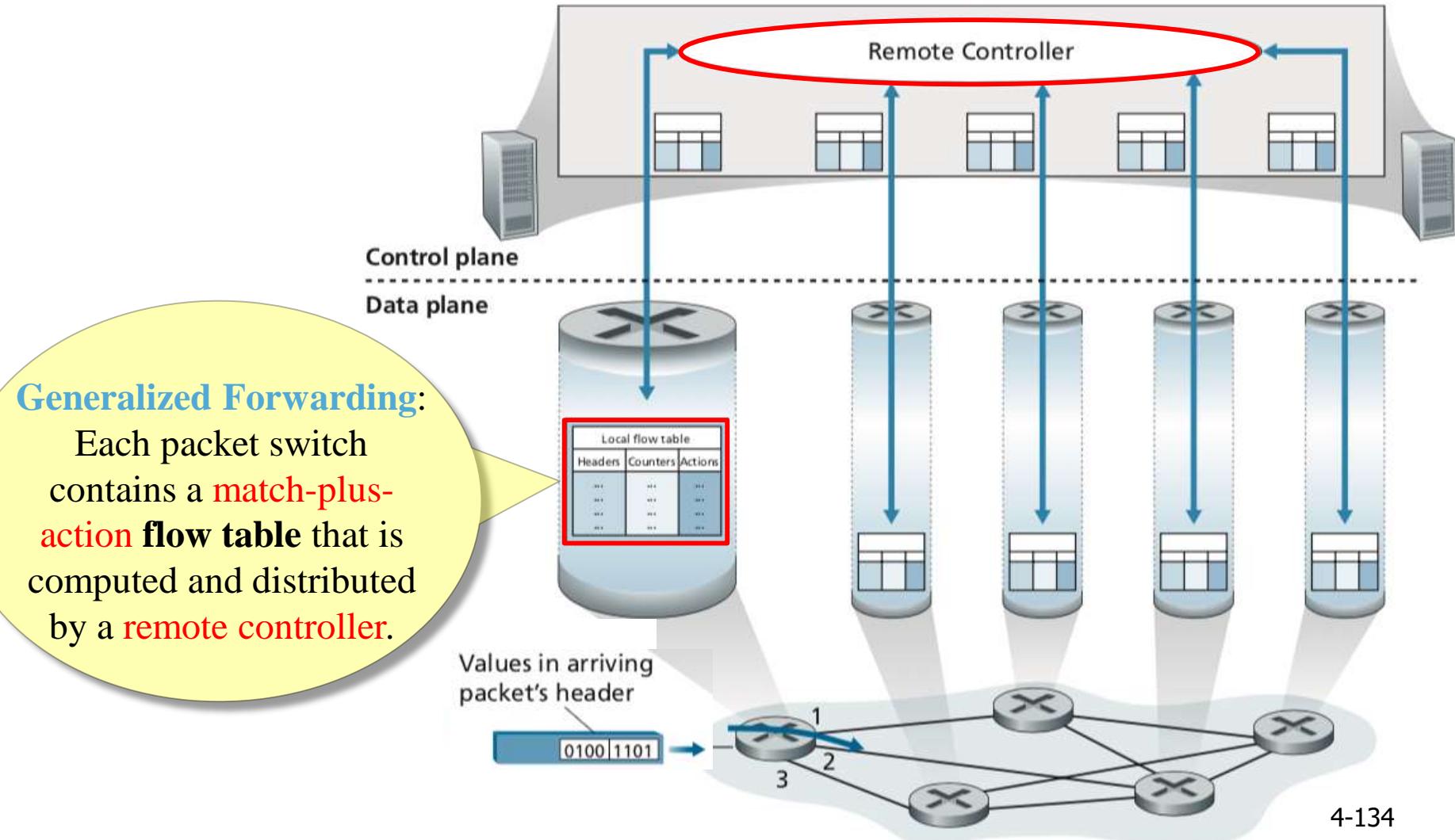
## 4.4 Generalized Forwarding and SDN



- ❖ Recall the **destination-based forwarding** (Section 4.1 and 4.2) as the two steps:
  - looking up a destination IP address (“**match**”),
  - then sending the packet into the switching fabric to the specified output port (“**action**”).

This considered as  
“**match-plus-action**”  
paradigm.

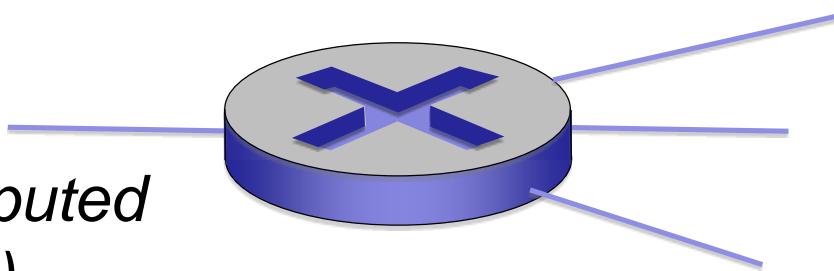




- ❖ Generalized forwarding will be based on a standard rule → **Open-Flow**.
- ❖ It is a highly visible standard that has pioneered the notion of the **match-plus-action** forwarding abstraction and controllers.
- ❖ *OpenFlow1.0* is used, which introduced key SDN abstractions and functionality

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

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



- *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

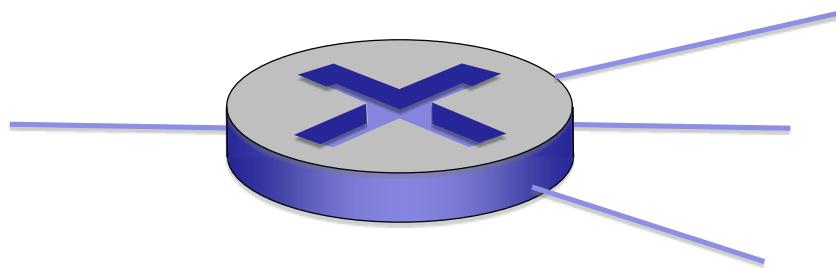
- *match*: IP addresses and TCP/UDP port numbers
  - *action*: permit or deny

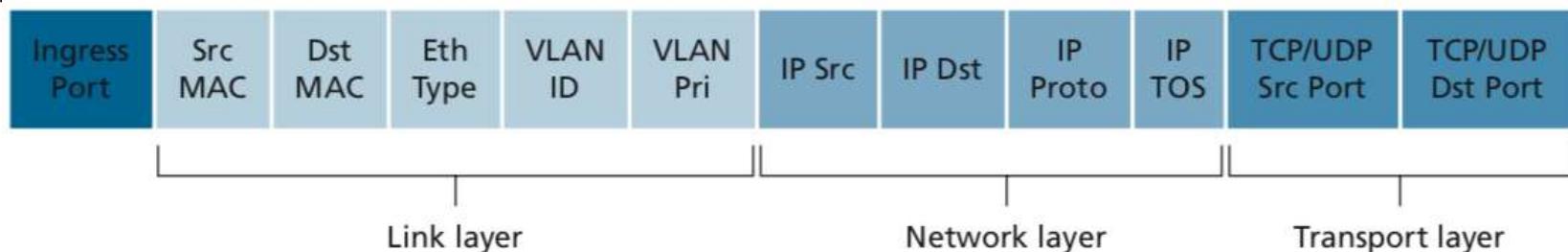
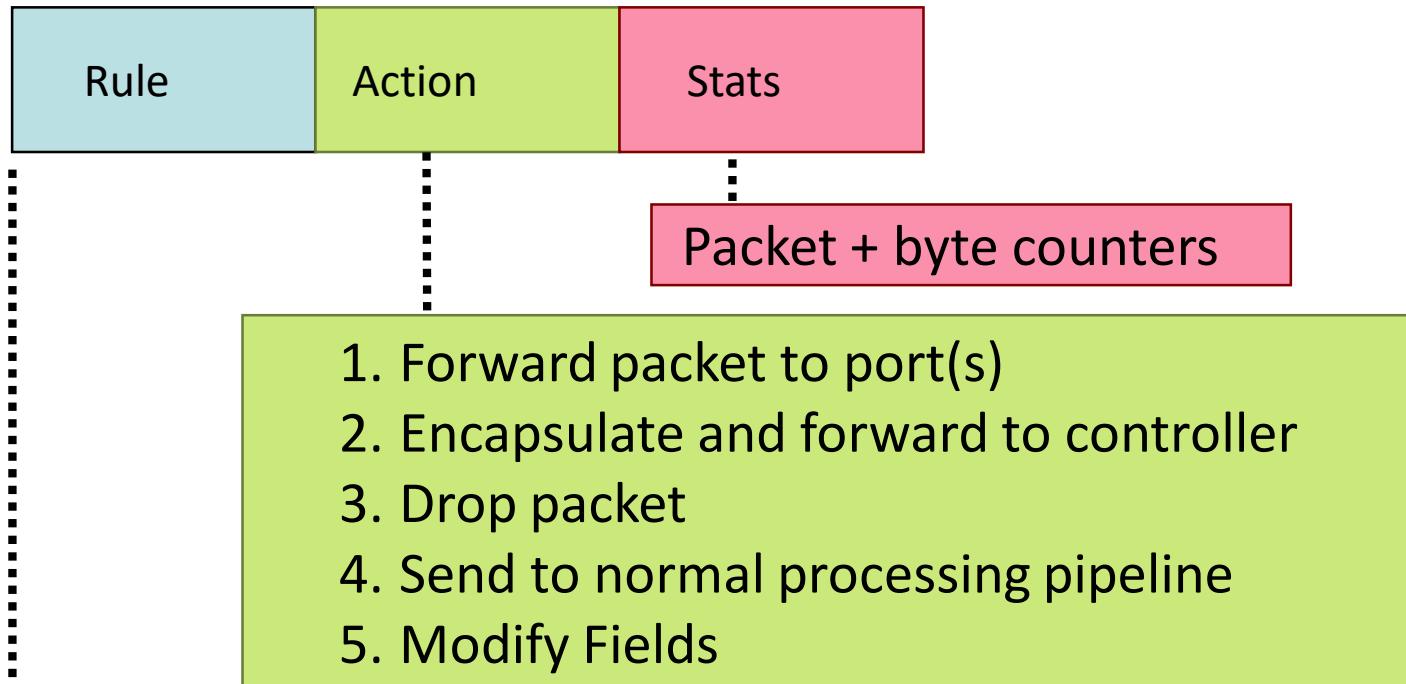
- NAT

- *match*: IP address and port
  - *action*: rewrite address and port

\* : wildcard

1.  $\text{src}=1.2.*.*$ ,  $\text{dest}=3.4.5.* \rightarrow \text{drop}$
2.  $\text{src} = *.*.*.*$ ,  $\text{dest}=3.4.*.* \rightarrow \text{forward}(2)$
3.  $\text{src}=10.1.2.3$ ,  $\text{dest} = *.*.*.* \rightarrow \text{send to controller}$





## 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*

## Examples 2

## Firewall:

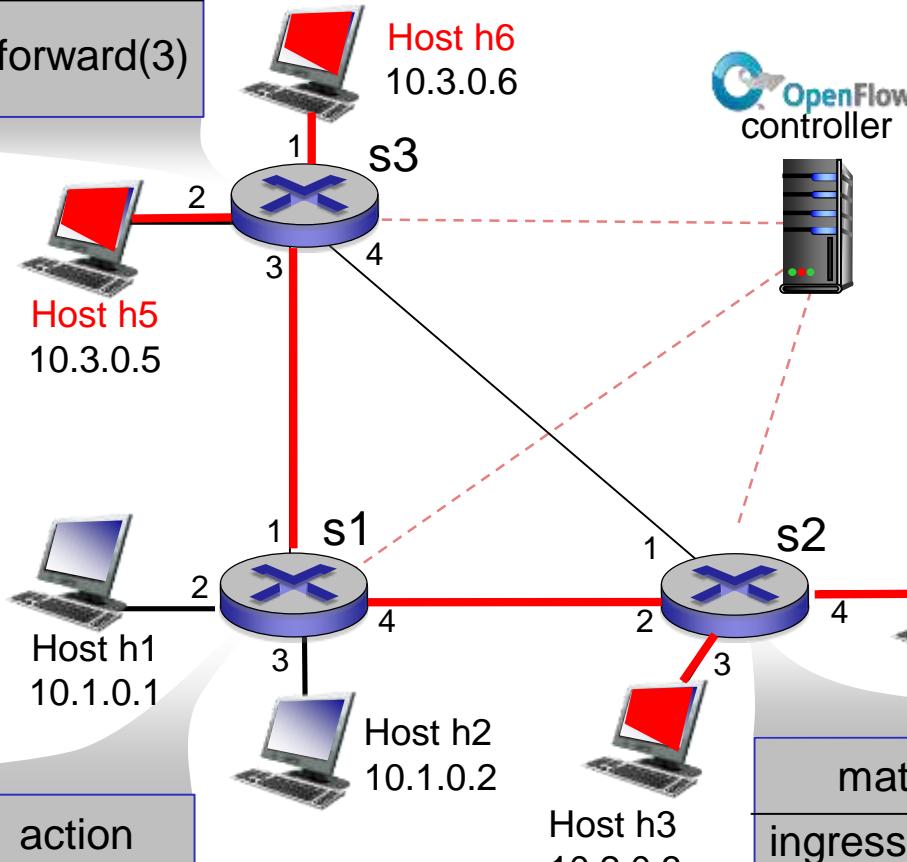
Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	*	*	*	*	*	22	Drop

*Do not forward (block) 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

*Do not forward (block) all datagrams sent by host 128.119.1.1*

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



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

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

## Examples 3

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