

# Computer Networks

## Chapter 4 – The Network Layer: Data Plane

Edition8-2

# Topics

- Network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - addressing
  - generalized forwarding
  - Internet architecture
- Instantiation, implementation in Internet
  - IP protocol
  - NAT, middleboxes, ...

# Contents

## 4.1 - Overview of Network Layer

## 4.2 - What's Inside a Router?

## 4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

## 4.4 - Generalized Forwarding and SDN

## 4.5 – Middleboxes

## 4.6 - Summary

# 4.1 Overview of Network Layer

- Primary role of network layer is:

**move packets from a sending host to a receiving host**

- To do so, two important **network-layer functions** can be identified as:

1. Forwarding

2. Routing

- Network layer communication services for **host-to-host communication** (endpoints are HOST to HOST, HOST to ROUTER, ROUTER to ROUTER)
- There is a piece of network layer **in each and every HOST and ROUTER**

## 4.1 Network Layer: Data plane + Control Plane

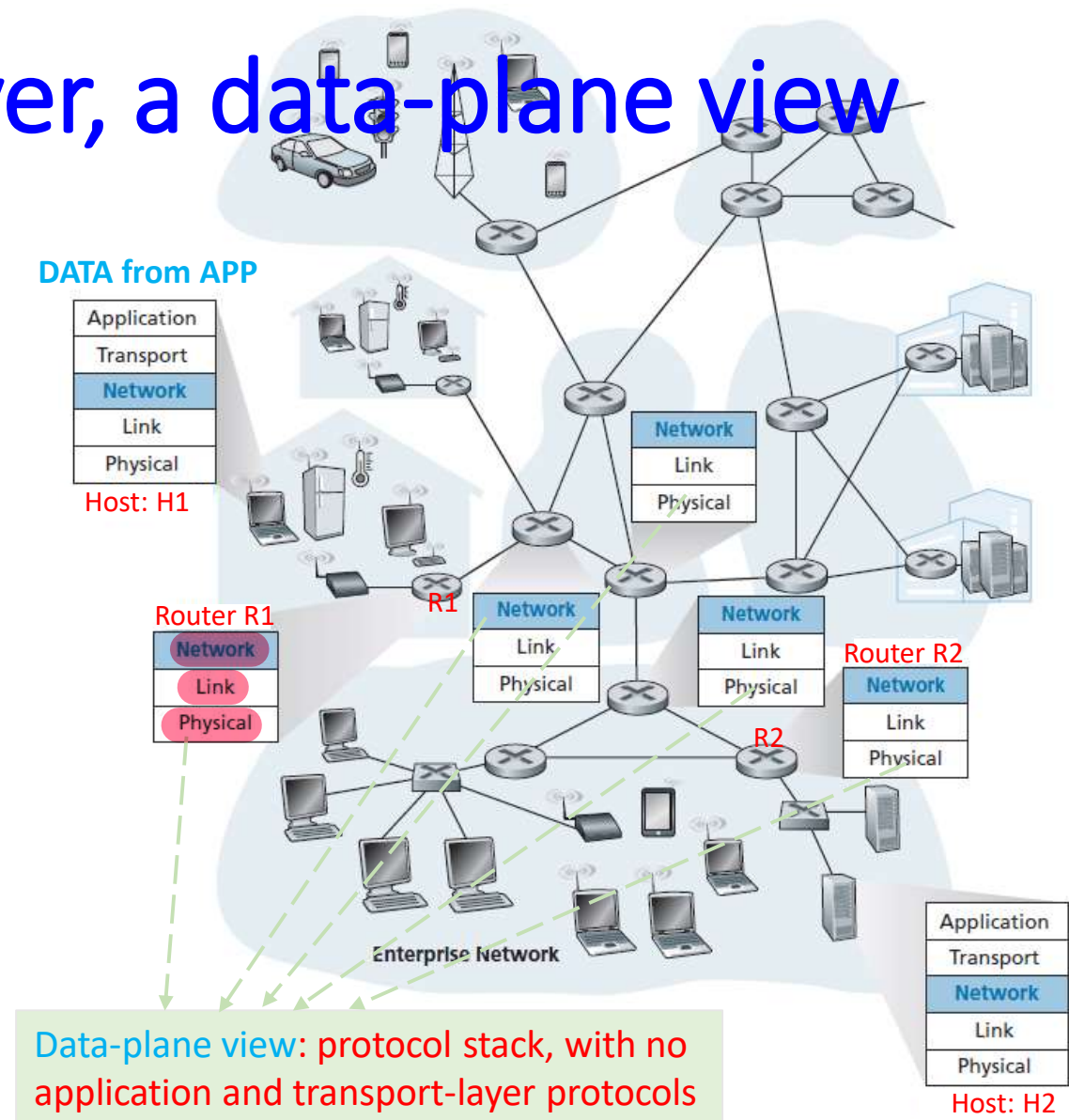
- Network layer (most complex layer in protocol stack): **data plane part** and **control plane part**
- **Data plane** functions (**Forwarding**: Per-router functions), Chapter 4
  - DATA Packets arriving on one of a router's input links is **forwarded** to one or several of router's output links
  - DATA Packets may be **blocked** or **duplicated** at router, or may have certain header field values **rewritten**
- **Control plane routing** functions (**Routing**: Network-wide logic that controls how a packet is routed among routers along an end-to-end path from source host to destination host), Chapter 5

# Traditional and Modern Networking

- Networking technologies:
  - Traditional IP forwarding
  - Generalized forwarding
- **Traditional networking:** control-plane routing functions and data-plane forwarding functions have been implemented together, monolithically, within a **ROUTER**
- **Generalized forwarding** (Software-defined networking -SDN): data plane and control plane explicitly separated. Control plane functions implemented as a separate service, typically in **remote servers** running “**controller**” **APPs**

# Figure 4.1 Network layer, a data-plane view

- **H1** is sending data to **H2**, consider role of **data plane of network layer** in **hosts** and in intervening **routers**
- Network layer in **H1** takes DATA segments from transport layer in **H1**, encapsulates each segment into a datagram, and sends datagrams to **R1**. **R1 forwards datagrams** from its input link to its appropriate output link
- At **H2**, network layer receives datagrams from **R2**, extracts transport-layer segments, and delivers segments up to transport layer at **H2**



# Data plane and Control plane roles

- **Primary data-plane role of each router:** forward datagrams from its input links to its output links
  - A **truncated protocol stack** of each router **participates** in forwarding (application and transport protocols are not participates in forwarding)
- **Primary role of network control plane:** end-to-end routing by coordinating per-router forwarding actions so that datagrams are transferred end-to-end, along paths of routers between source and destination hosts
  - All protocol layers of each ROUTER (including application and transport) are participating in end-to-end routing



# 4.1.1 Forwarding: The Data Plane

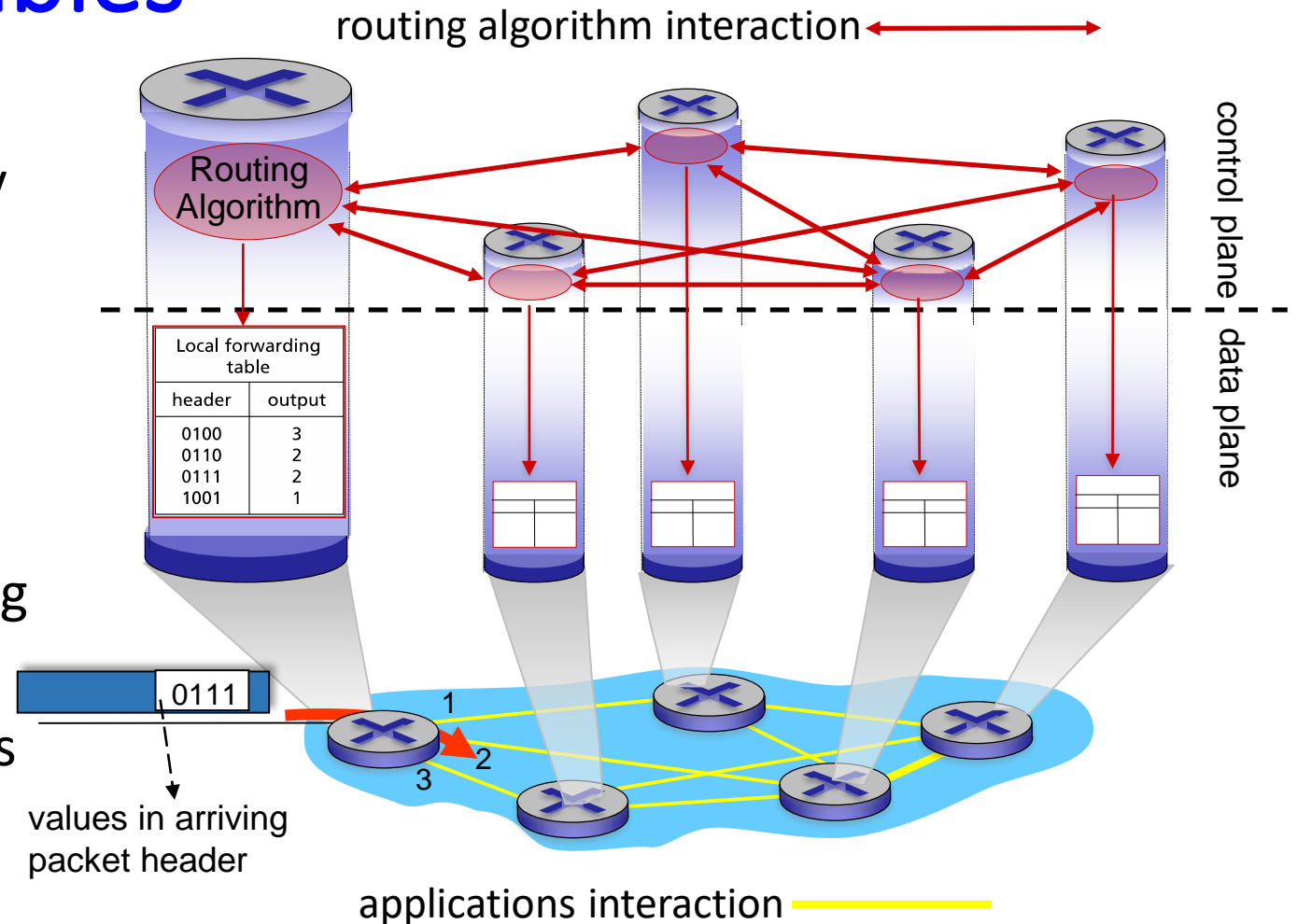
1. **Forwarding**: When a packet arrives at a router's input link, router moves packet to appropriate output link
  - Example: a packet arriving from **H1** to **R1** in Figure 4.1 is forwarded to next router on a path to **H2**
  - Forwarding is a function implemented in data plane
  - **Forwarding** takes place at **very short timescales** (typically a few **nanoseconds**), and is typically implemented in **hardware**
  - In more general case (Section 4.4), a packet might be
    - **blocked from exiting a router** (for example, if packet originated at a known malicious sending host, or if packet were destined to a forbidden destination host), or be
    - **duplicated** and sent over multiple outgoing links

# Routing: The Control Plane

2. **Routing: Network layer in a ROUTER** must determine route or path taken by packets as they flow from a sender to a receiver
- Routing algorithms that calculate paths are referred to as **routing algorithms**
  - A routing algorithm would determine, for example, path along which packets flow from H1 to H2 in Figure 4.1
  - Routing protocols are implemented in control plane of network layer or application layer which **needs all five layers**
  - **Routing** refers to a network-wide process that determines **end-to-end paths that packets take from source to destination**. Routing (algorithm and protocol) takes place on much longer timescales (typically **seconds**), and is often implemented in **software**

# Figure 4.2 Routing algorithms determine values in forward tables

- Routing algorithm create an updated forwarding table for any router
- A router forwards a packet by examining value of one or more fields in arriving packet's header, and then using these header values to index into its forwarding table of arriving interface
- Forwarding table for those values indicates outgoing link interface

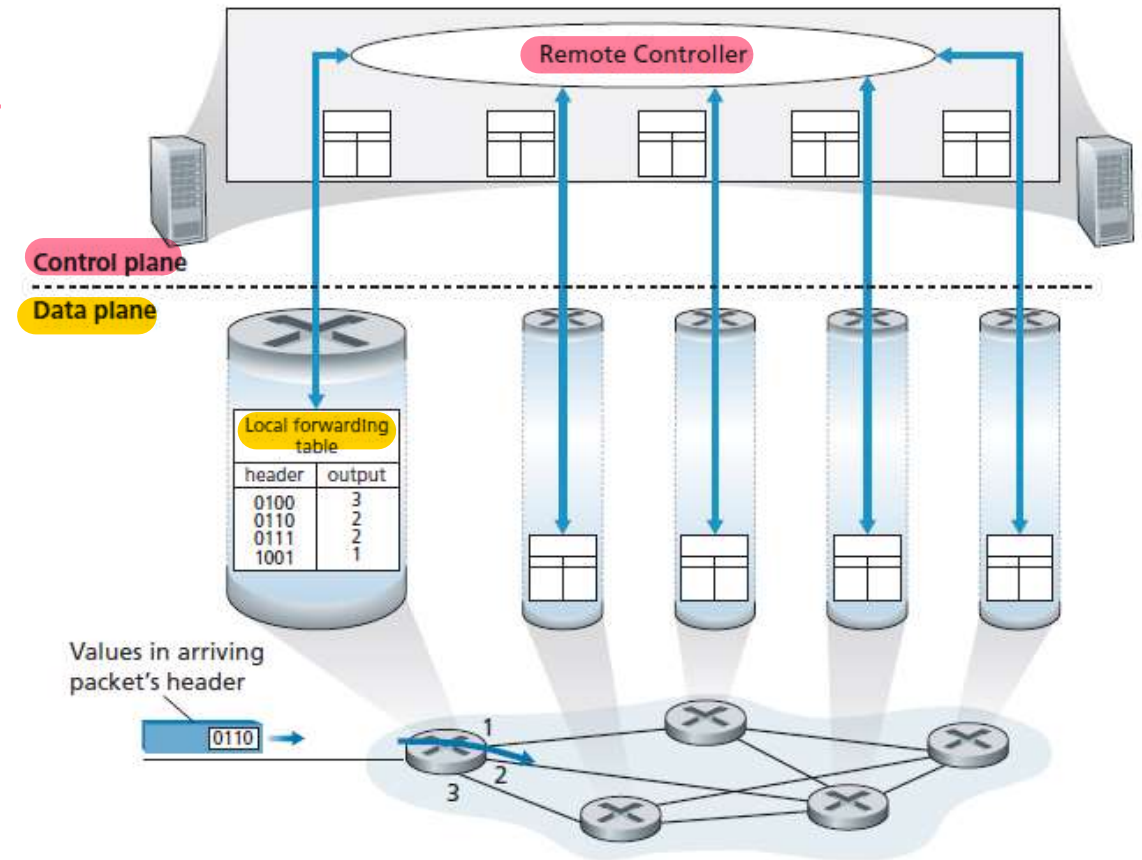


# Control Plane: Traditional Approach, Figure 4.2

- **Routing algorithm** determines contents of routers' forwarding tables  
In traditional approach, a routing algorithm runs in each and every router
- Both forwarding and routing functions are contained within a router
- **Routing algorithm** function in one router communicates with **routing algorithm** function in other routers to compute values for its forwarding table
- **Routing algorithms** exchange routing messages containing routing information

# Control Plane: software-defined networking

- Physically separation of control function from router (Figure 3)
- A remote controller (run on a server machine) computes and distributes forwarding tables to be used by each and every router
- Routing device performs forwarding only, so we do not call it “Router”. It is just a **packet switch**
- Data plane functions of Figures 4.2 and 4.3 are identical

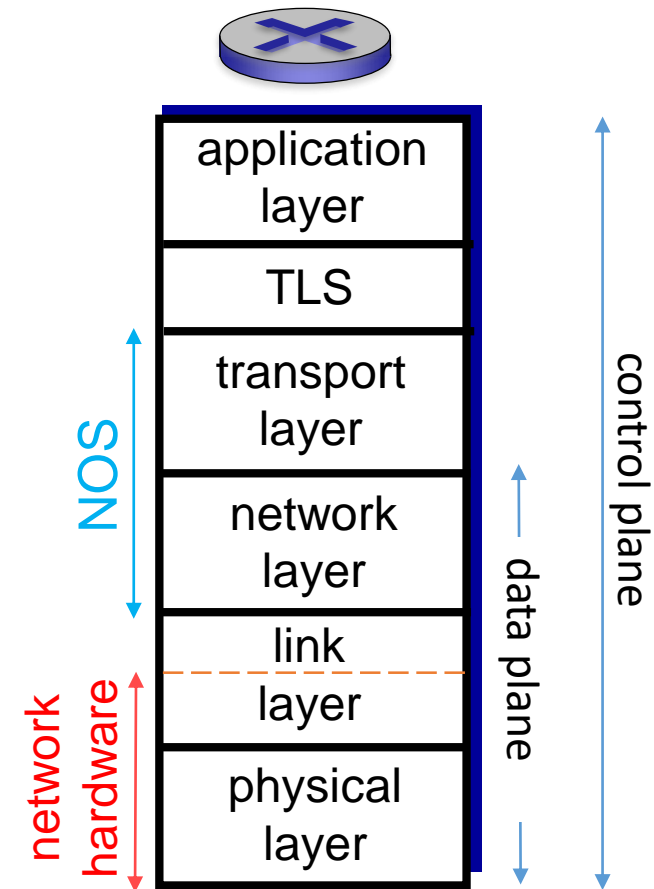
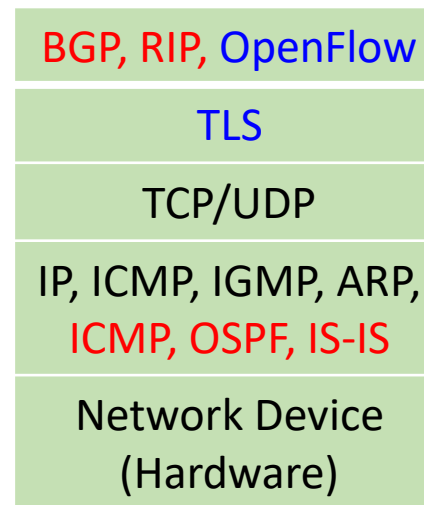


# SDN Approach

- Remote controller might be implemented in a remote data center with high reliability and redundancy, and might be managed by ISP or some third party
- How might routers and remote controller communicate? By exchanging messages containing forwarding tables and other pieces of routing information
- Control-plane approach is at heart of **software-defined networking (SDN)**, controller that computes forwarding tables and interacts with packet switches is implemented in software
- Software implementations are also open, similar to Linux OS code, allowing ISPs (and networking engineers) to innovate and propose changes to software that controls network-layer functionality

# Router architecture: Traditional and SDN

- Routing algorithms on traditional approach:
  - **BGP: TCP** on port 179
  - **RIP: UDP** on port 520
  - **ICMP:** on IP
  - **IS-IS:** on link layer (using MAC address)
  - **OSPF:** on IP in IPv4 , on link layer in IPv6 (using MAC address)
- SDN: OpenFlow (TCP)
- Network OS (NOS):
  - TCP/IP
  - SNA (IBM)
  - Cisco IOS software
  - DECnet
  - ...




## 4.1.2 Network Service Model

- Transport layer transmits packets into network (passes it to network layer)
  - Can network layer deliver in order packets to destination?
  - Will amount of time between sending of two sequential packet transmissions be same as amount of time between their reception?
  - Will network provide any feedback about congestion in network?
- **Network service model** defines characteristics of end-to-end delivery of packets between sending and receiving hosts



# Network Services

Services by network layer could include:

- **Guaranteed delivery**: guarantee a packet sent by a source host will eventually arrive at destination host
- **Guaranteed delivery with bounded delay** : delivery within a specified host-to-host delay bound
- **In-order packet delivery** : packets arrive at destination in order that they were sent
- **Guaranteed minimal bandwidth**:  transmission bit rate between sending and receiving hosts be above an specified minimum
- **Security** : encrypt all datagrams at source and decrypt them at destination, thereby providing confidentiality to all transport-layer segments

# Individual datagrams, flow of datagrams

Example services for individual datagrams:

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

Example services for a flow of datagrams from a source process to a destination process in a session:

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

# Internet's network layer service model

- A single service: known as **best-effort service**
- No guarantee for: **delivery, order, minimum delay, minimum bandwidth, security**
- Packets are neither guaranteed to be received in order in which they were sent, nor is their eventual delivery even guaranteed. There is no guarantee on end-to-end delay nor is there a minimal bandwidth guarantee
- Other network architectures go beyond Internet's best-effort service
- ATM network architecture provides for guaranteed in-order delay, bounded delay, and guaranteed minimal bandwidth

# Proposed service model extensions to Internet architecture

- Intserv (integrated services) architecture aims to provide end-end delay guarantees and congestion-free communication
  - Internet's best-effort combined with adequate bandwidth provisioning and bandwidth-adaptive application-level (DASH, Section 2.6.2) have been good to enable a range of applications such as Netflix, Skype and so on
- Diffserv (Differentiated Services) (RFC 2475)
  - ...

# Network architectures and Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees			
		Bandwidth	Loss	Order	Delay
Internet	best effort	No	No	No	No
Internet	Intserv Guaranteed (RFC 1633)	Yes	Yes	Yes	Yes
Internet	Diffserv (RFC 2475)	Possible	Possible	Possible	No
ATM	Available Bit Rate (ABR)	Guaranteed min	No	Yes	No
ATM	Constant Bit Rate (CBR)	Constant rate	Yes	Yes	Yes

- ATM: asynchronous transfer mode
  - CBR: Constant bit rate
  - VBR: Variable bit rate
  - ABR: Available bit rate
  - UBR: unspecified bit rate

# Reflections on best-effort service

- **Simplicity of best-effort** has allowed Internet to be widely deployed

## Enhancing best-effort services:

1. Sufficient **provisioning of bandwidth** allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of time”
2. Bring APPs close to client’s network allows low latency and high throughput. **Content distribution networks (CDN)** allows APPs be everywhere
3. Scalability: **Datacenters** allow replication of an APP on several servers
4. Congestion control of “elastic” services helps

# Overview of remaining sections of Chapter 4

Data-plane component of network layer in following sections in this chapter

- **Section 4.2:** internal hardware operations of a router:
  - input and output packet processing
  - router's internal switching mechanism
  - packet queueing and scheduling
- **Section 4.3:** traditional IP forwarding, based on destination IP address, IP4/6 protocols
- **Section 4.4:** generalized forwarding
  - Based on a large number of header values
    - Packets may be blocked or duplicated at router, or may have certain header field values rewritten, all under software control
    - It is a key component of a modern network data plane, in software-defined networks (SDN)
- **Section 4.5:** Beside routers, there are other network equipment ("middleboxes") within network that **sit on data path** and **perform functions other than forwarding**.

# Some terms

- Terms **forwarding** and **switching** are often used interchangeably
- Term **packet switch** means a general **packet-switching device** that transfers a packet from input link interface to output link interface(s), according to values in a packet's header fields
  - Packet-switching device based on Link-layer address (Link-layer switch = LAN switch)
    - forwarding decision based on values in fields of link-layer frame (link-layer or layer 2 packet switch) (Chapter 6)
  - Packet-switching device based on Network-Layer address (Router):
    - forwarding decision based on header field values in network-layer datagram (network-layer or layer 3 packet switch)



# Contents

4.1 - Overview of Network Layer

**4.2 - What's Inside a Router?**

4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

4.4 - Generalized Forwarding and SDN

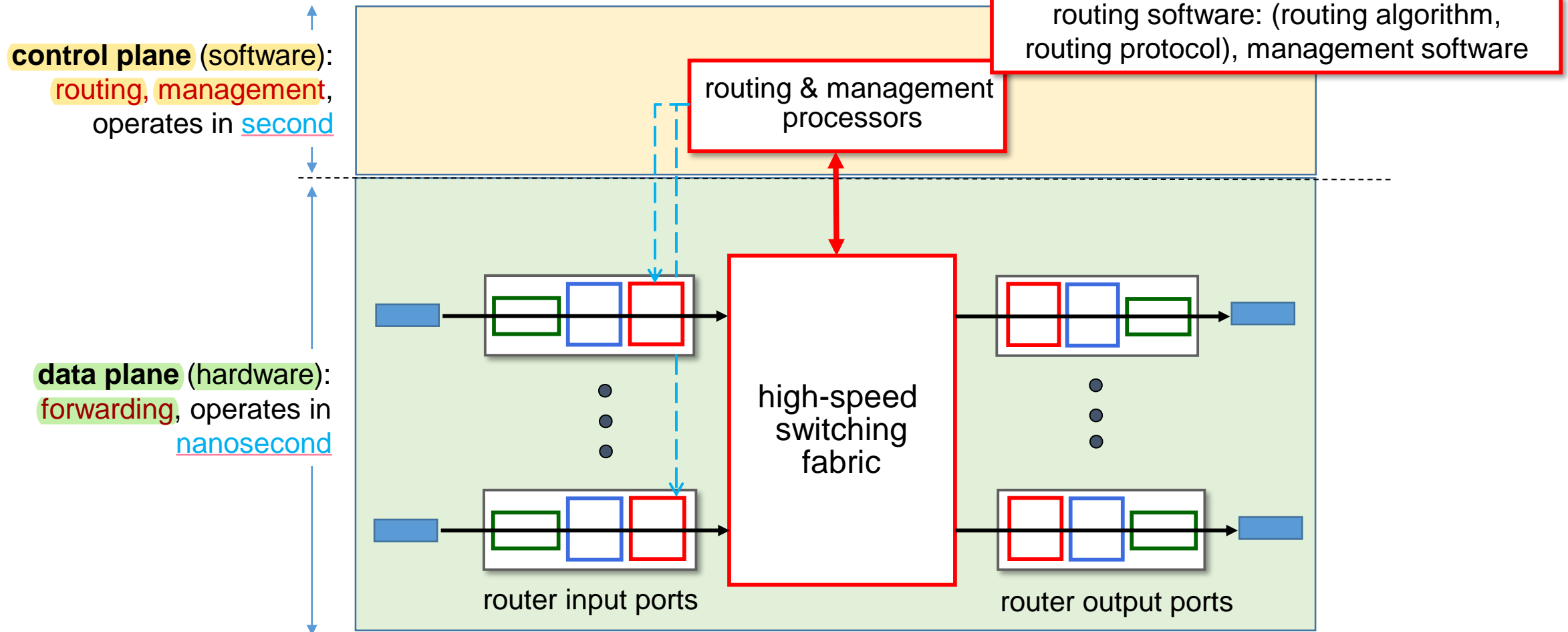
4.5 – Middleboxes

4.6 - Summary

## 4.2 What's Inside a Router?

- A high-level view of a generic router architecture is shown in Figure 4.4. Four router components can be identified:
- Hardware:
  - Input ports
  - Switching fabric
  - Output ports
  - Routing processor
- Software:
  - Routing Algorithm
  - Routing Protocol

# Figure 4.4 Router architecture



Routing process produces an updated forwarding table (look up table) for each input port. Routing processes in different routers use routing protocol to communicate.

# Input ports

## Input port functions:

- **Physical layer function** of terminating an incoming physical link at a router
- **Link-layer functions** needed **to interoperate with link layer at other side of incoming link**
- **Lookup function**, forwarding table is consulted to determine router output port to which an arriving packet will be forwarded via switching fabric
  - Term “port” refers to physical input and output router
  - Number of ports supported by a router can range from small number in enterprise routers, to **hundreds of 100Gbps ports** in a router at an ISP’s edge, where number of incoming lines tends to be greatest
  - Example: Juniper MX2020, **edge router**, supports up to 800 **100Gbps** Ethernet ports, with an overall router system capacity of **80Tbps**
- **Control packets** (packets carrying routing protocol information delivered from other routers) **are forwarded from an input port to routing processor**

# Switching fabric and Output ports

## Switching fabric

- It connects router's input ports to its output ports. This switching fabric is completely contained within router (a network inside of a network router)

## Output ports

- It stores packets received from switching fabric and transmits these packets on outgoing link by performing necessary link-layer and physical-layer functions. When a link is bidirectional (that is, carries traffic in both directions), an **output port will typically be paired with input port for that link on same line card**

# Routing processor

Routing processor performs control-plane functions

- In traditional routers
  - it executes routing protocols (Sections 5.3 and 5.4),
  - maintains routing tables and attached link state information, and
  - computes forwarding table for router
- In SDN routers
  - It is responsible for communicating with remote controller in order to (among other activities) receive forwarding table entries computed by remote controller, and install these entries in router's input ports
- It also performs network management functions (Section 5.7)

# Hardware or Software implementation

- A router's input ports, output ports, and switching (forwarding) fabric are almost always implemented in hardware
- Why hardware?
- Consider 100Gbps input link and a 64-byte IP datagram, input port has only 5.12ns to process datagram before another datagram may arrive
- A line card with  $N$  ports must operate  $N$  times faster. It is too fast for software implementation
- Forwarding hardware (switching fabric) implemented either using a router vendor's own hardware designs, or constructed using silicon chips (from Intel, Broadcom, ...)
- Data plane operates nanosecond time scale
- Control functions responding to
  - attached links that go up or down
  - communicating with remote controller (in SDN case)
  - performing management functions (at millisecond or second timescale)
- Control plane functions are thus usually implemented in software and execute on routing processor (typically a traditional CPU)

# Forwarding Table lookup

- Lookup performed in each of input port
- Router uses forwarding (routing) table to look up output port to which an arriving packet will be forwarded via switching fabric

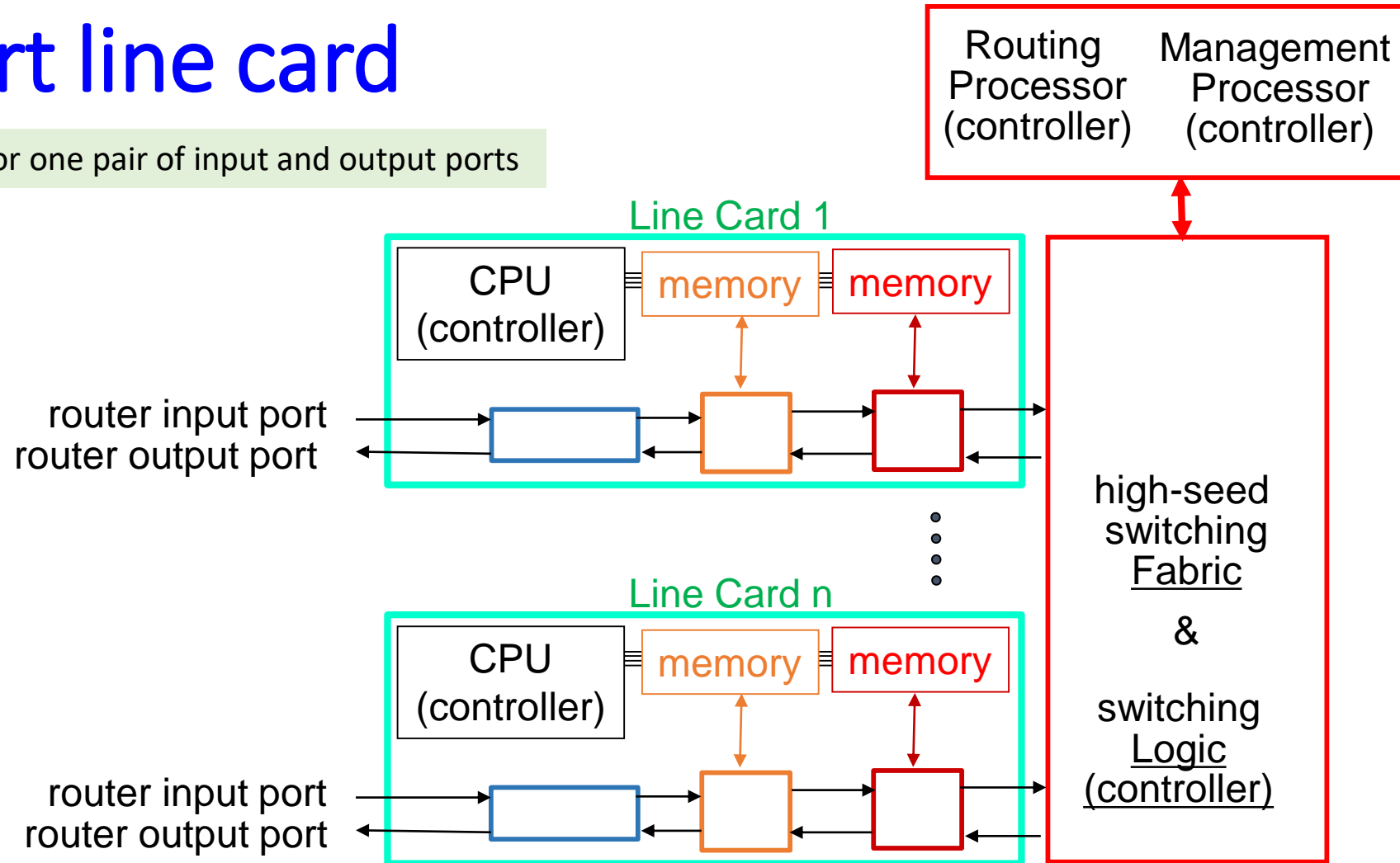
## Forwarding table:

- computed and updated by routing processor (using a routing protocol to interact with routing processors in other network routers)
- or
- is received from a remote SDN controller
  - Forwarding table is copied from routing processor to **line cards** over a separate bus (e.g., a **PCI bus**) indicated by dashed line from routing processor to input line cards in Figure 4.4.
  - Forwarding decisions can be made locally, at each input port, without invoking centralized routing processor on a per-packet basis and thus avoiding a centralized processing bottleneck



# 2-Port line card

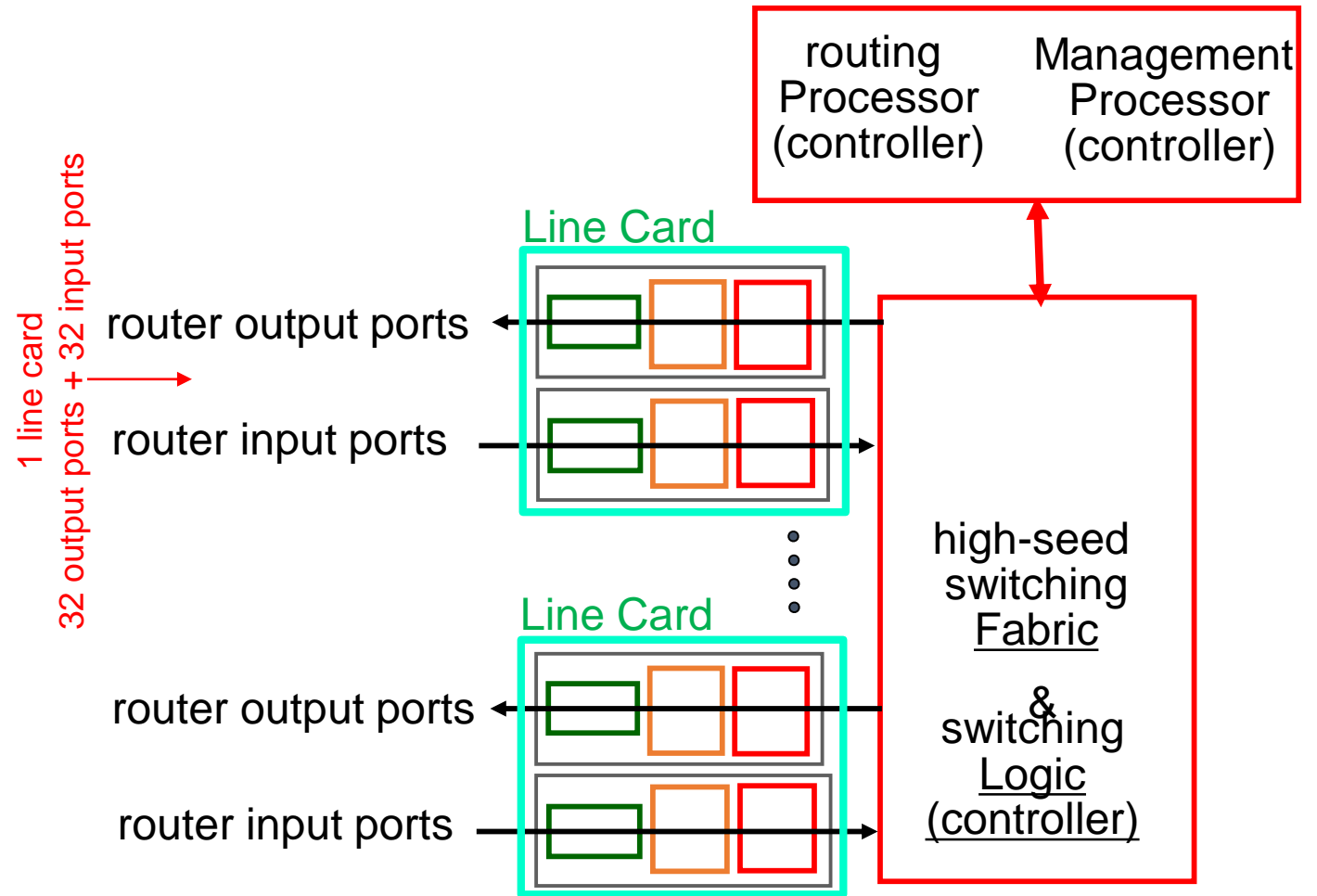
One IP address for one pair of input and output ports



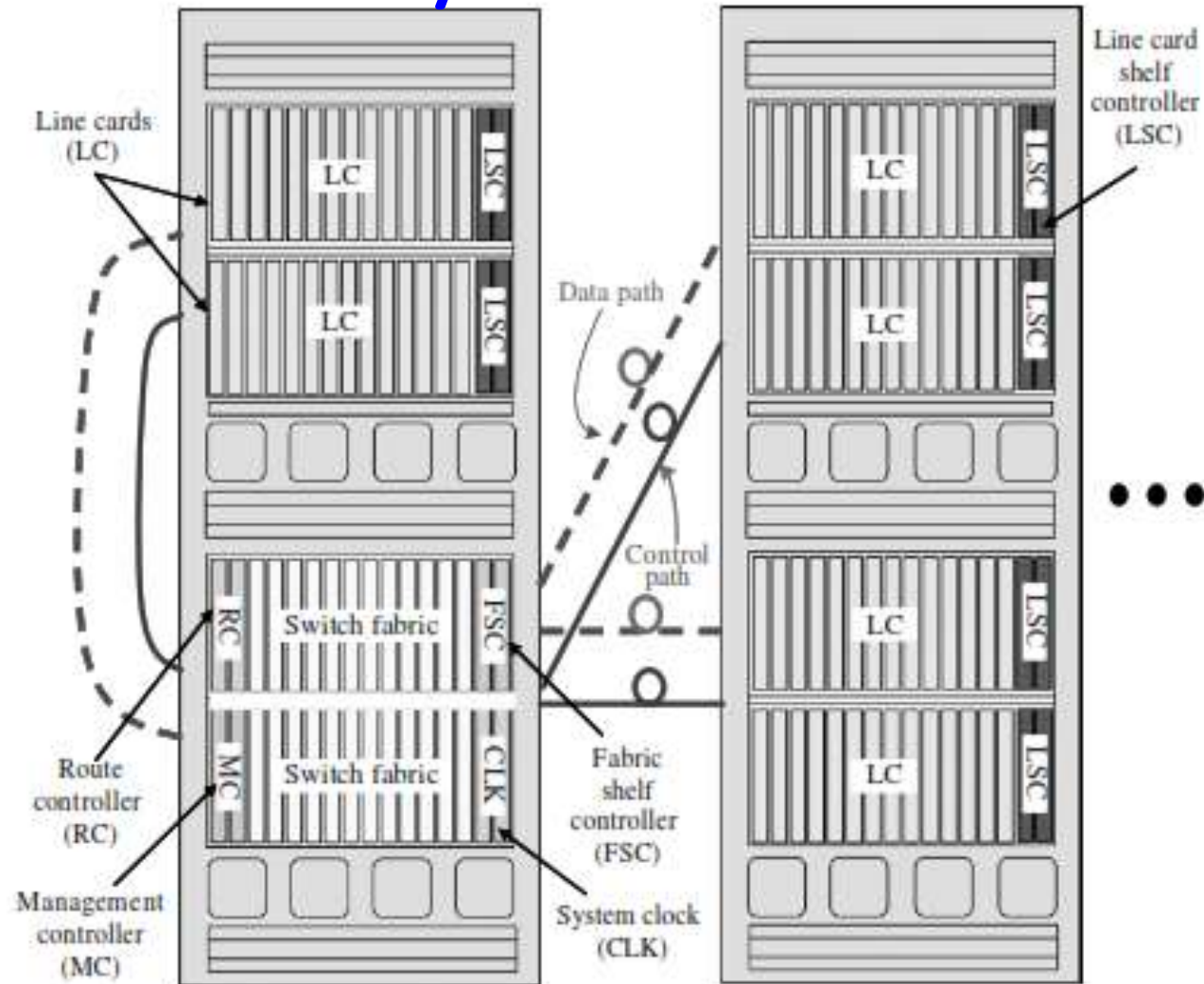
memory: DRAM provides storage for routing (forwarding) tables and for other routing processes

# Multi-port line card

- Example: **CX-10256-S/OMP-800** edge Router
- Each **line card** contains **64x100Gbps ports**, broken down into 32 switch **ports (output)**, which connect to **an** external network, and 32 fabric **ports (input)**, which connect to switching fabric **cards**.



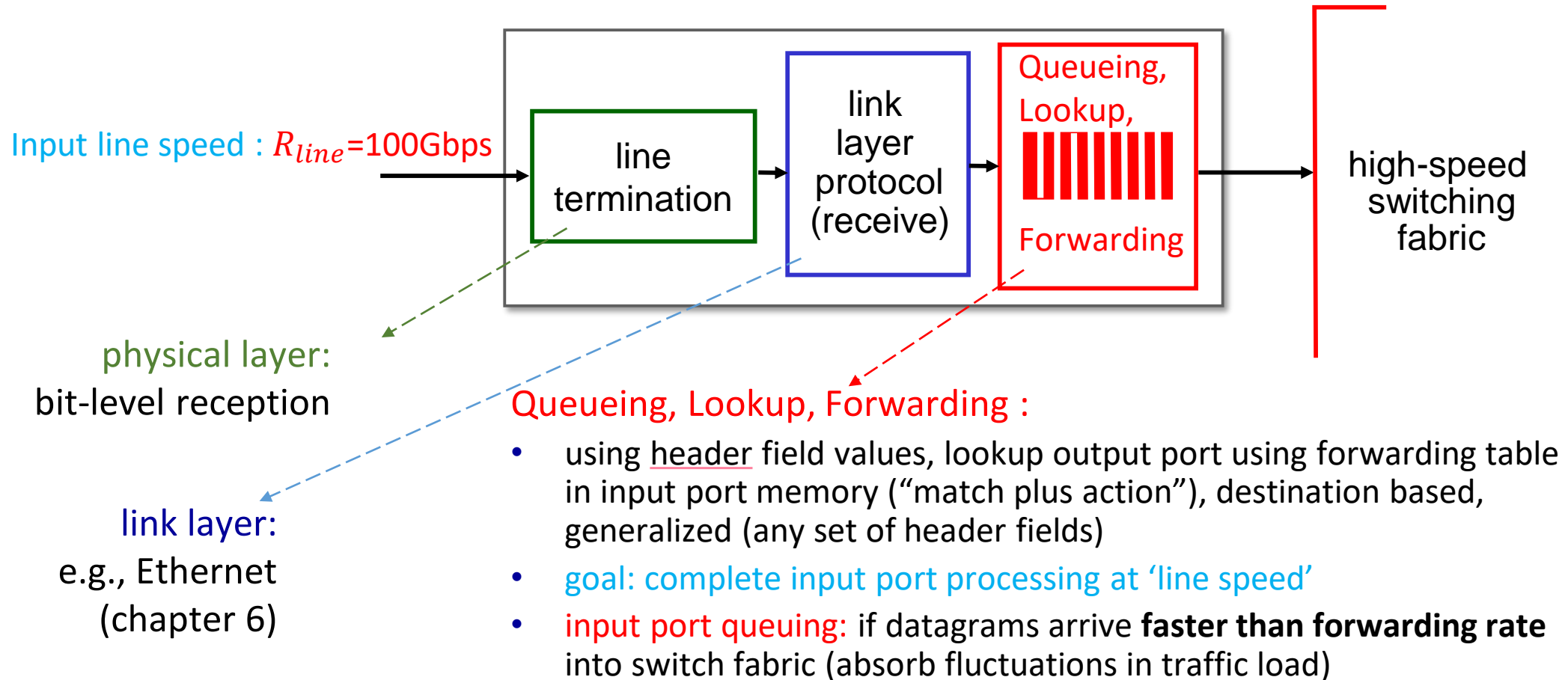
# Multi-Rack Router System



# Current core router manufacturers

- Nokia (7950 Extensible Routing System [XRS] Series)
- Brocade Communications Systems (NetIron XMR Series)
- Cisco Systems (CRS, NCS series)
- Extreme Networks (Black Diamond 20808)
- Ericsson (SSR series)
- Huawei Technologies Ltd. (NetEngine 5000E, NetEngine 80E, NetEngine 80)
- Juniper Networks (Juniper T-Series and PTX Series)
- MikroTik (MikroTik Cloud Core Router series)
- ZTE (ZXR10 Series : T8000, M6000)

## 4.2.1 Figure 4.5 Input port processing



## 4.2.1 Input Port Processing and Destination-Based Forwarding

- **Traditional router:** output port to which an incoming packet is to be switched is based on **packet's destination IP address**

- In 32-bit IP addresses, there are more than 4 billion possible addresses (destinations)
- How many entries in forwarding table?
  - Consider a 4-port router, and following 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

# How many entries?

- For this example, it is not necessary to have 4 billion entries in router's forwarding table. We could have following forwarding table with just **four entries**:

Destination Address Range		Link Interface
11001000 00010111 00010***	*****	0
11001000 00010111 00011000	*****	
11001000 00010111 00011***	*****	2
Otherwise		3

Destination Address Range		Link Interface
11001000 00010111 00010000	00000000	0
11001000 00010111 00010111	11111111	
11001000 00010111 00011000	00000000	1
11001000 00010111 00011000	11111111	
11001000 00010111 00011001	00000000	2
11001000 00010111 00011111	11111111	
Otherwise		3

Address block assign to interface 2 is wider than it should be

# Longest prefix matching rule

- Router matches a  prefix of packet's destination IP address with entries in table

Example:

- packet's destination IP address: 11001000 00010111 00010110 10100001; first 21 bits matches first entry in table (link interface 0)

Example,

- packet's destination IP address: 11001000 00010111 00011000 10101010
  - first 24 bits match **second** entry in table
  - first 21 bits match **third** entry in the table
- There are 2 matches, router uses **longest prefix matching rule**:
- It finds longest matching entry in table and forwards packet to link interface associated with longest prefix match (**24 bits, second entry**), more detail in Section 4.3



# Longest prefix matching - example

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 <b>match!</b> 1 00011*** *****	2
otherwise	3

examples:

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

# Longest prefix matching - example

First 21 bits match third entry in table

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 - example

First 24 bits match second entry in table

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?

# Separation of address block - example

- Address block:

11001000 00010111 00010000 00000100

through

11001000 00010111 00010000 00000111

is separated from:

11001000 00010111 00010000 00000000

through

11001000 00010111 00010111 11111111


and assigned to a network  
reachable from interface 3

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

# Separation of address block - example

				Destination Address Range		Link Interface		
				11001000 00010111 00010000 00000000		0		
				through				
Destination Address Range				Link Interface		00010111 11111111		
11001000 00010111 00010*** *****				0		00010000 00000100		3
				ugh				
11001000 00010111 00010000 000001**				3		00010000 00000111		
11001000 00010111 00011000 *****				1		00011000 00000000		1
11001000 00010111 00011*** *****				2		ugh		
Otherwise				3		00011000 11111111		
				11001000 00010111 00011001 00000000		2		
				through				
				11001000 00010111 00011111 11111111		3		
				Otherwise				

# TCAM for longest prefix matching

- Longest prefix matching often performed using **Ternary Content Addressable Memories** (TCAMs)
  -  TCAM: retrieve address in **one clock cycle**, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM

# Other actions in input port

- After lookup: packet sent into switching fabric
- A packet may be temporarily **blocked** from entering switching fabric if packets from other input ports are currently using fabric
  - **Blocked packet** will be queued at input port and then scheduled to cross fabric at a later point in time
- Other actions in input port processing:
  1. physical and link layers processing
  2. packet's **version number**, **checksum** and **time-to-live** (Section 4.3) are checked and **latter two fields rewritten**
  3. counters used for network management (such as number of IP datagrams received) must be updated

# Match plus Action abstraction

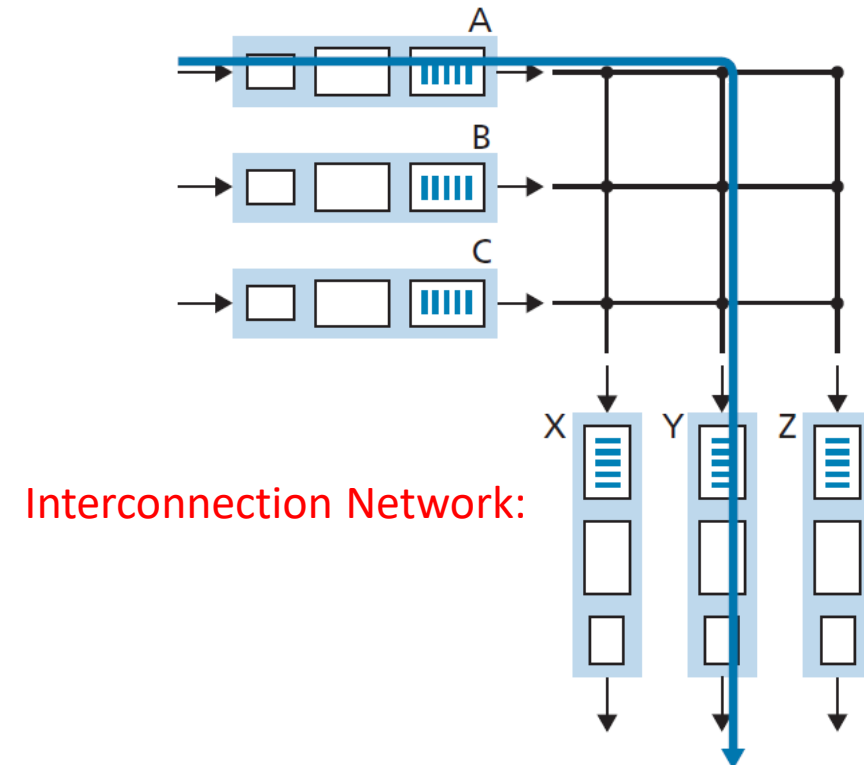
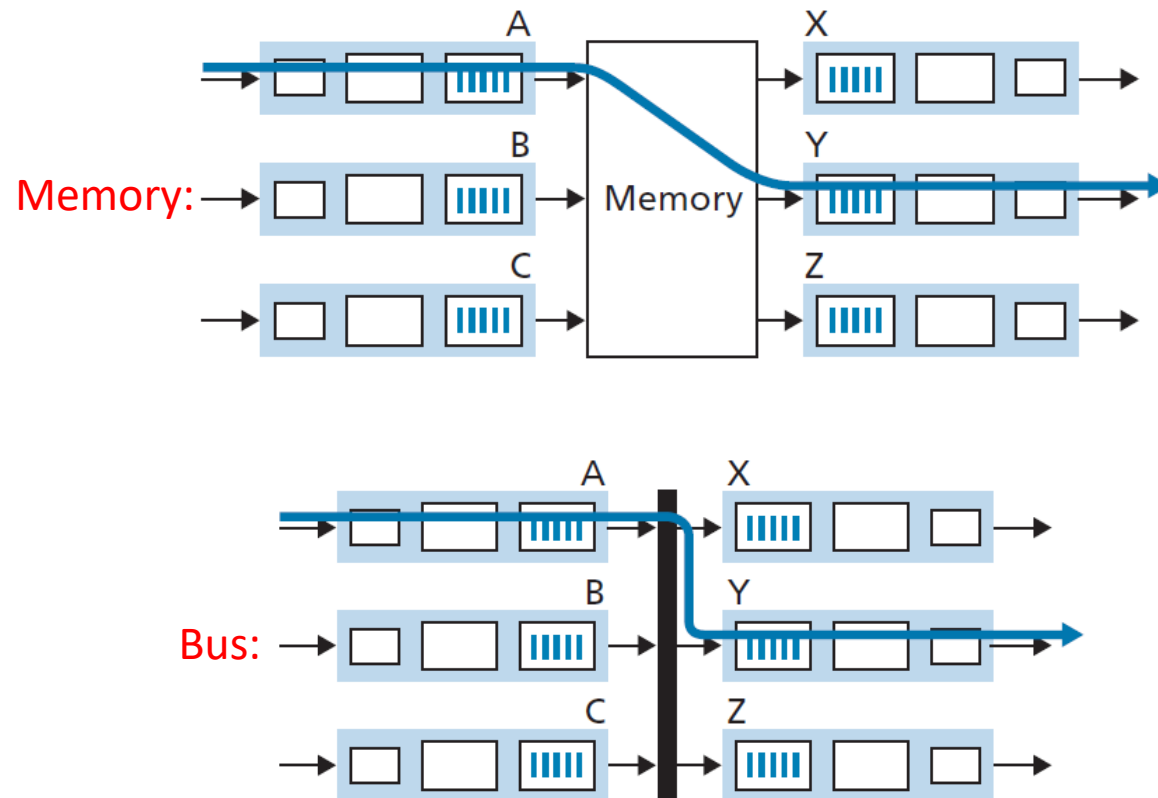
- Looking up at traditional routers:
  - a destination IP address (“match”)
  - sending packet into switching fabric to specified output port (“action”)
- Looking up at link-layer switches (Chapter 6):
  - a destination MAC address (“match”)
  - sending frame into switching fabric towards output port (“action”) (other actions may be taken)
- Looking up at firewalls (filter out selected incoming packets):
  - a combination of source/destination IP addresses, transport-layer port numbers, ... (“match”)
  - drop packet (“action”)
- Looking up at Network Address Translator (NAT, Section 4.3):
  - destination transport-layer port number (“match”)
  - destination transport-layer port number rewritten (“action”)
- “match plus action” is prevalent in network devices, and is central to notion of generalized forwarding (Section 4.4)



## 4.2.2 Switching



- Switching can be accomplished in a number of ways, Figure 4.6:



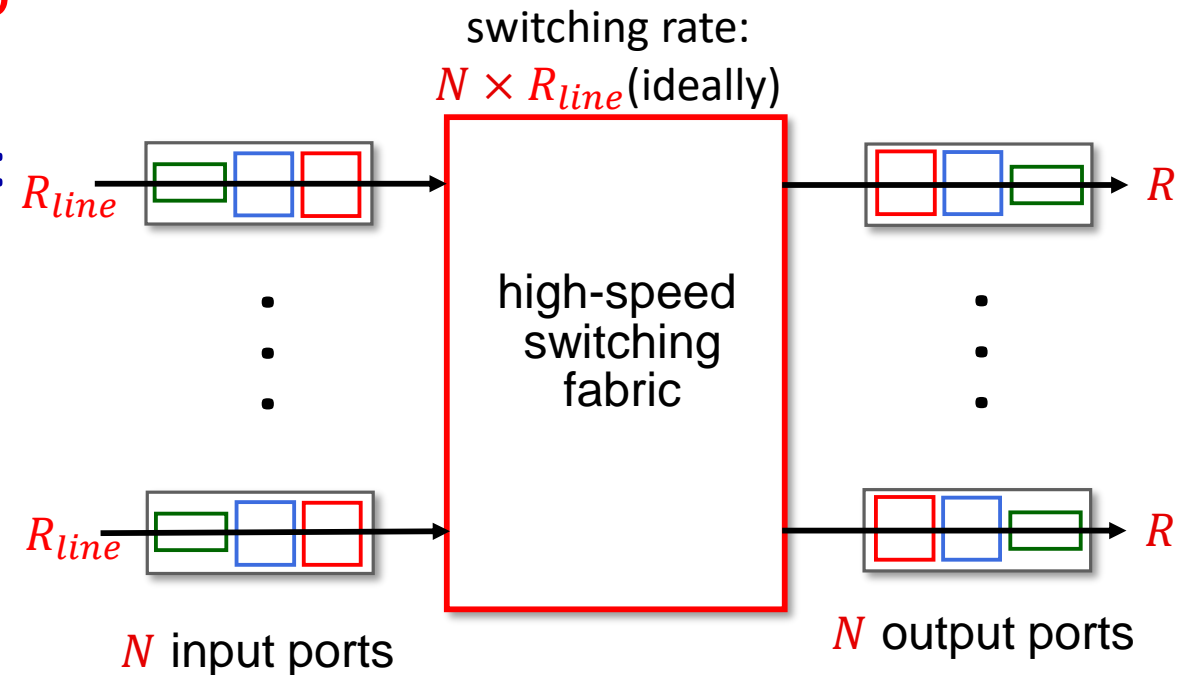
# Switching rate of switch fabric

- Transfer packet from input link to appropriate output link

- $R_{switch}$  [packet/sec] switching rate:  
rate at which packets can be transfer from inputs to outputs

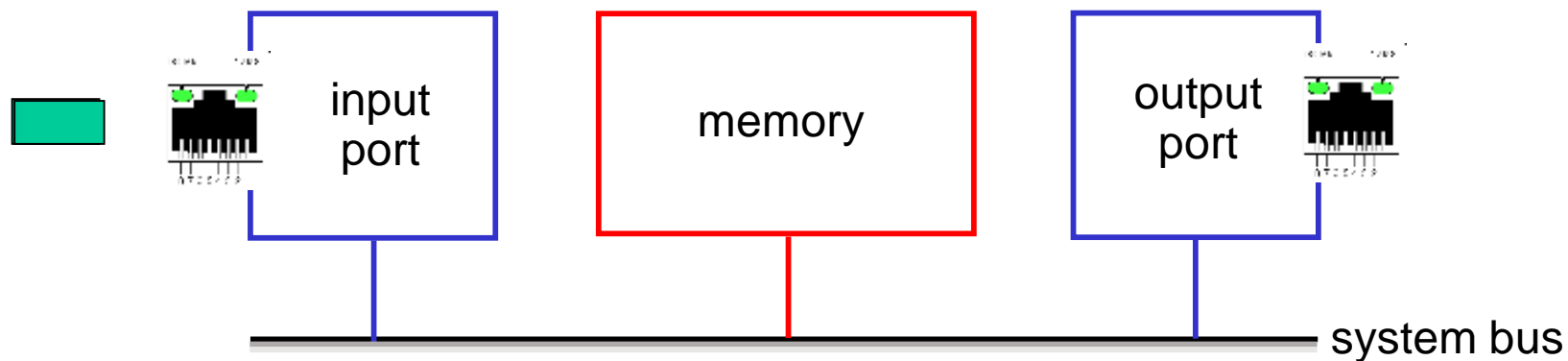
- Often measured as multiple of input/output line rate ( $R_{line}$  packet/sec)

$N$  inputs: Switching rate  $N$  times line rate is desirable (switching rate:  $N \times R_{line}$ )



# Switching via memory

- Simplest, earliest routers were a computers
- ① Input port interrupt routing processor, ② packet copied from input port into processor memory (*written into memory*), ③ routing processor extracted destination IP address, ④ looks up forwarding table, ⑤ copies packet to output port's buffers (*read from memory*)



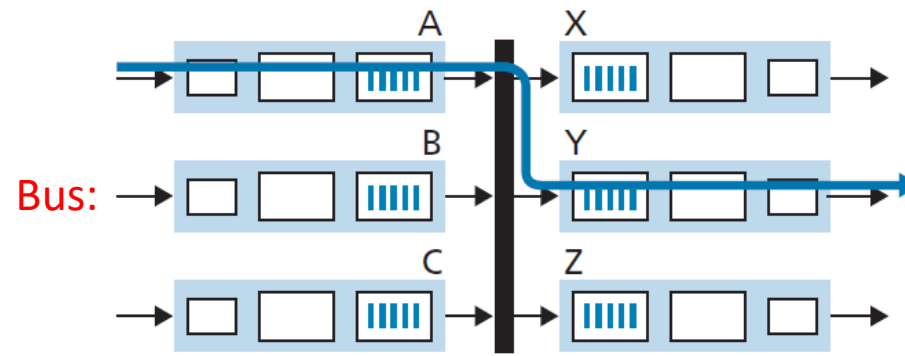
# Switching via memory

- **Memory bandwidth:**  $B$  packets per second (read from memory or write into memory)
- Forwarding throughput (rate for packets transferred from input ports to output ports): Read a packet from input port memory, process (lookup, ...) AND THEN write into output port memory, so  $R_{switch} < \frac{B}{2}$
- **Input traffic (workload):** Two packets cannot be forwarded at same time, even if they have different destination ports, since only one memory read/write can be done at a time over shared system bus,  $\max(R_{switch}) = R_{line} \left[ \frac{\text{packet}}{\text{sec}} \right]$
- For a router with high memory bandwidth if  $\frac{B}{2} \geq R_{line}$  then  $R_{switch} = R_{line}$
- **Some modern routers switch via memory**
  - Lookup and storing packet into appropriate memory location are performed **by processing on input line cards**
  - Very much like **shared memory multiprocessors**
  - **Cisco's Catalyst 8500 series** switches [Cisco 8500 2020] internally switches packets via a **shared memory**

# Shared memory packet switch

- There is a **common pool of memory** that is **shared by all input and output ports** on router
- In a router with  $N$  input ports and  $N$  output ports, this pool needs to support up to  $N$  input queues and up to  $N$  output queues
- Process on a line card **writes packets** (switching packet) **into memory of appropriate output port**
- One benefit of **shared memory** is ability to dynamically allocate more or less of same memory to different ports **on demand** depending on packet arrivals on a given output port

# Switching via a bus

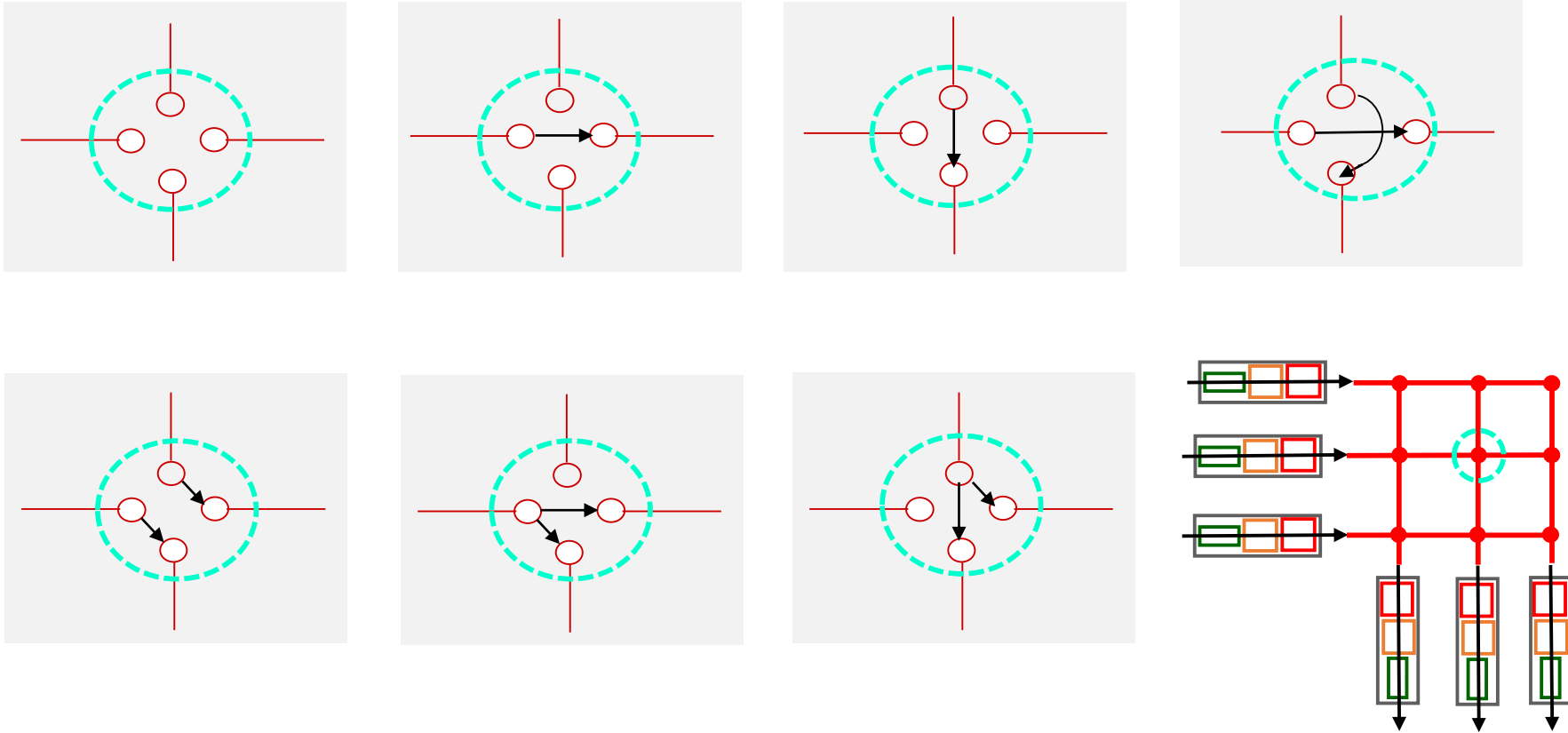


- Input port transfers a packet directly to output port over a **shared bus**, **without intervention by routing processor**
- Input port adds a label (header) to packet indicating an output port, and transmitting packet onto bus
- All output ports receive packet, but only port that matches label will keep packet
- Label is removed at output port
- Switching rate is limited to **bus speed**:  $R_{switch} < \text{Bus Speed} \left[ \frac{\text{packet}}{\text{sec}} \right]$
- Two packets cannot be forwarded at same time, even if they have different destination ports, since only one packet can cross bus at a time,  $\max(R_{switch}) = R_{line}$
- **Cisco 6500** router internally switches packets over a 32-Gbps bus (sufficient for small LANS and enterprise networks)

# Switching via an Interconnection Network

- $2N$  buses connected to  $N$  input ports to  $N$  output ports
- Input port buses have **cross-point** with output port buses
- **Cross-points** can be opened or closed at any time by switch fabric controller (whose logic is part of switching fabric itself)
- Packet **from an input** can reach to **one or several outputs**
- Several input packets can move to different output ports at same time
- **Non-blocking**: a packet being forwarded to an output port will not be blocked from reaching that output port as long as no other packet is currently being forwarded to that output port
  - if two packets from two different input ports are destined to same output port, one will have to wait at input, since only one packet can be sent over any given bus at a time
- Cisco 12000 series uses a crossbar switching network
- Cisco 7600 series can be configured to use **either a bus or crossbar switch**

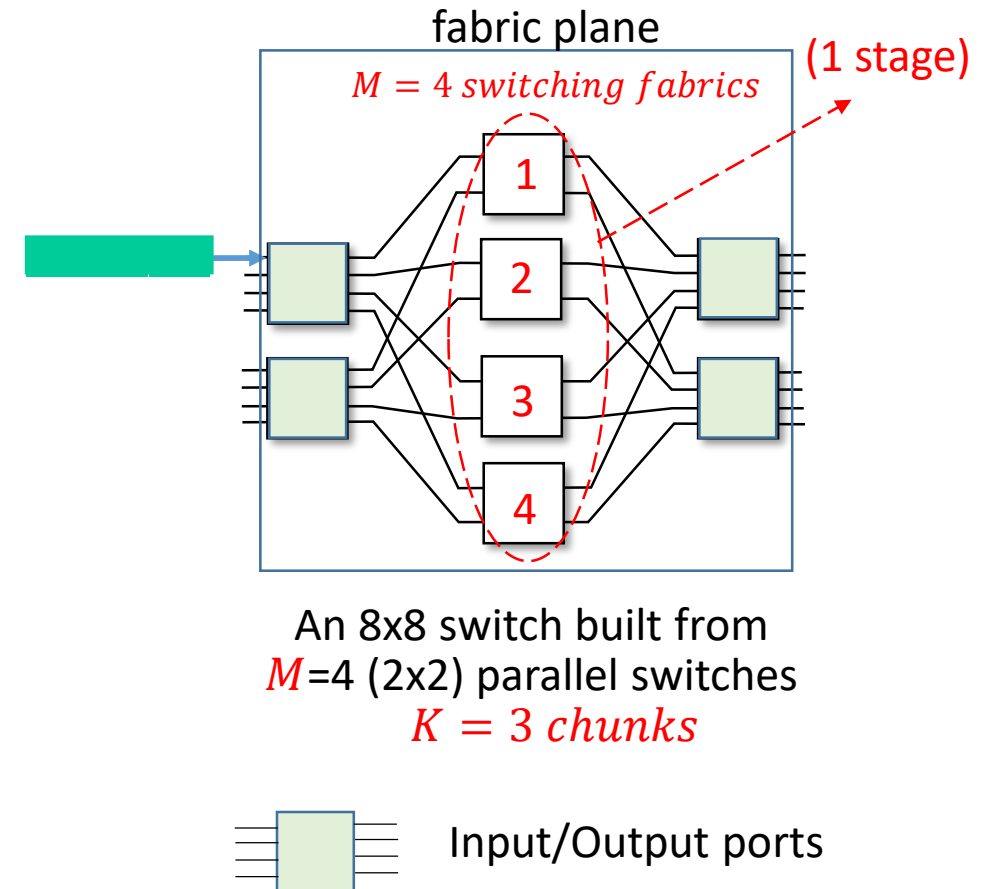
# Cross-point





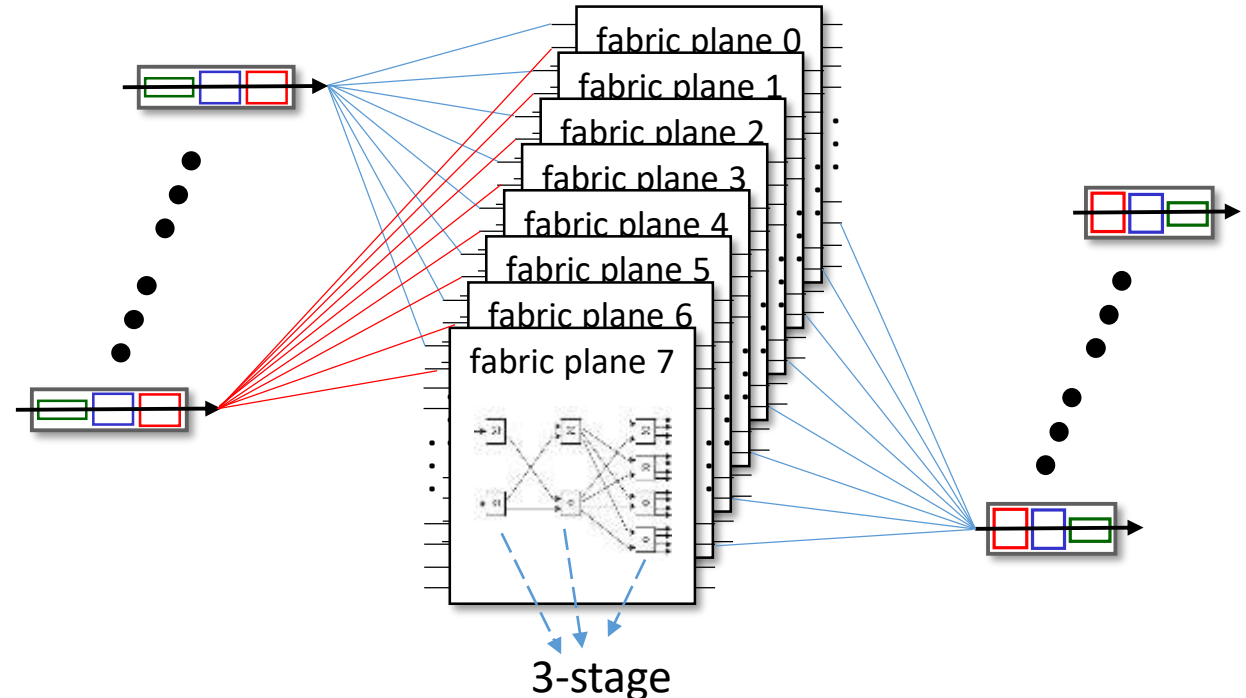
# Multiple switch fabrics as a switch fabric

- A router's switching capacity can also be scaled by running multiple switching fabrics in parallel
- Input ports and output ports are connected to  $M$  switching fabrics that operate in parallel
- An input port **breaks a packet into**  $K \leq M$  smaller chunks, and sends chunks through  $K$  of these  $M$  switching fabrics to selected output port. Output port reassembles  $K$  chunks back into original packet



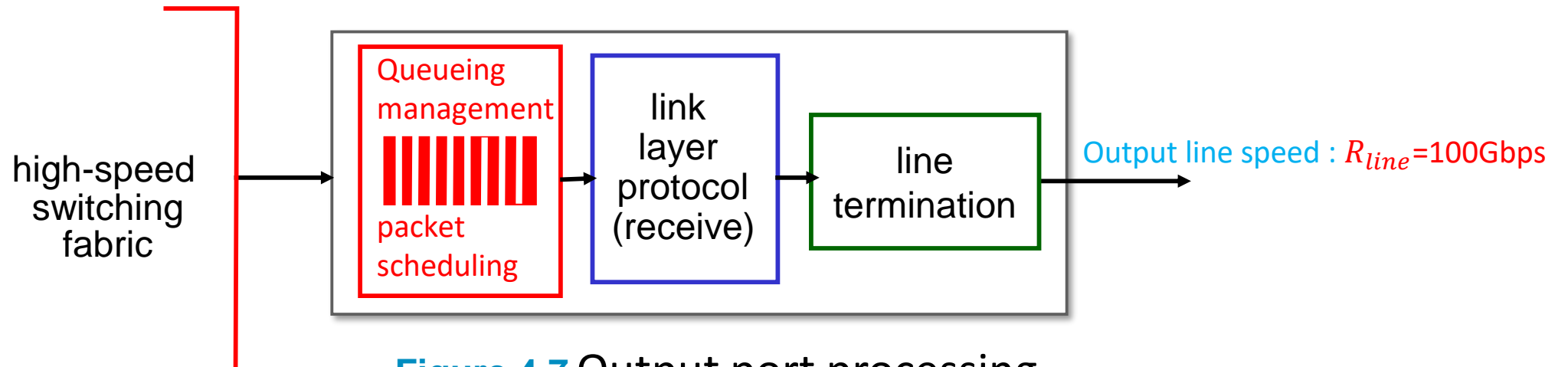
# Multiple fabric planes each multi stages

- Multiple fabric planes, each plane made of multi parallel stages
- Packets from different input ports proceed towards **same output port at same time**
- Cisco CRS employs a three-stage **non-blocking** switching strategy
  - basic unit: 8 switching planes
  - each plane: 3-stage
  - up to 100'sTbps switching rate



## 4.2.3 Output Port Processing

- Output port processing takes packets that have been stored in output port's memory (buffer) and transmits them over output link
- This includes selecting (i.e., scheduling) and de-queueing packets for transmission, and performing needed network-layer, link-layer and physical-layer transmission functions



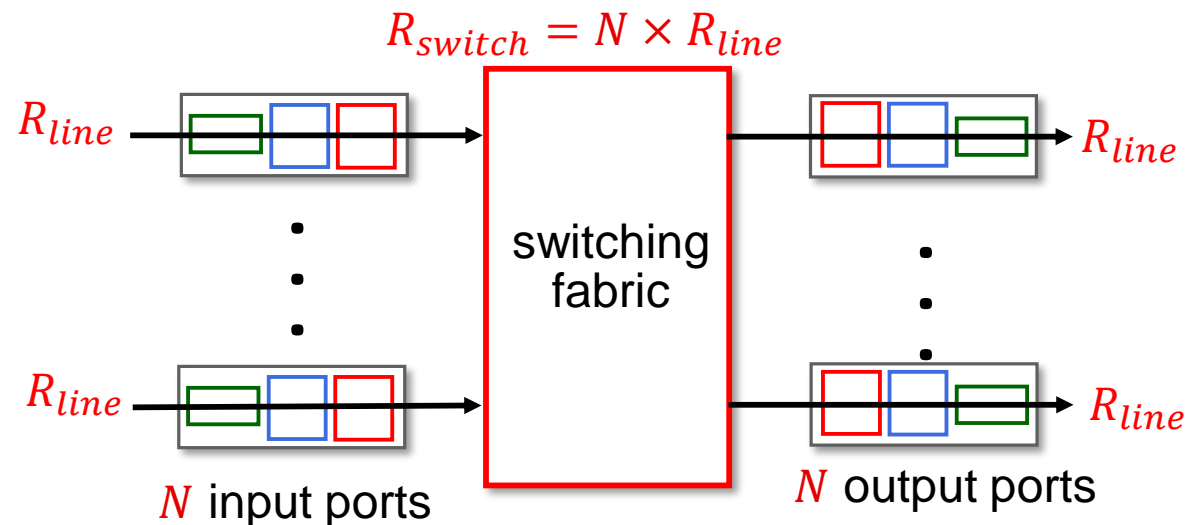
**Figure 4.7** Output port processing

## 4.2.4 Where Does Queuing Occur?

- Packet queues may form at both input ports and output ports
- Queueing depend on:
  - traffic load (packet arrival)
  - line speed -  $R_{line}$
  - speed of switching fabric-  $R_{switch}$  (relative to line speed and number of input ports)
- Router's memory can be exhausted and packet drop (loss) will occur when no memory is available to store arriving packets

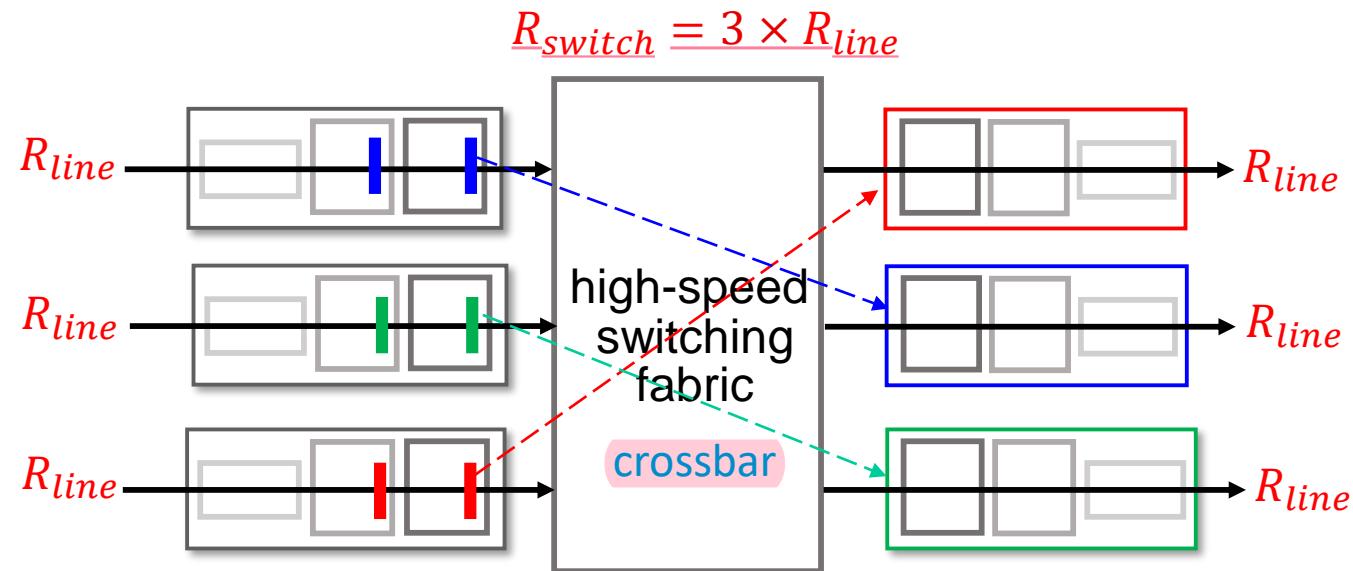
# Example

- $R_{line}$  packets per second: input and output line speeds (transmission rates)
- $N$  input ports and  $N$  output ports
- All packets have same fixed length
- Packets arrival to input ports is synchronous



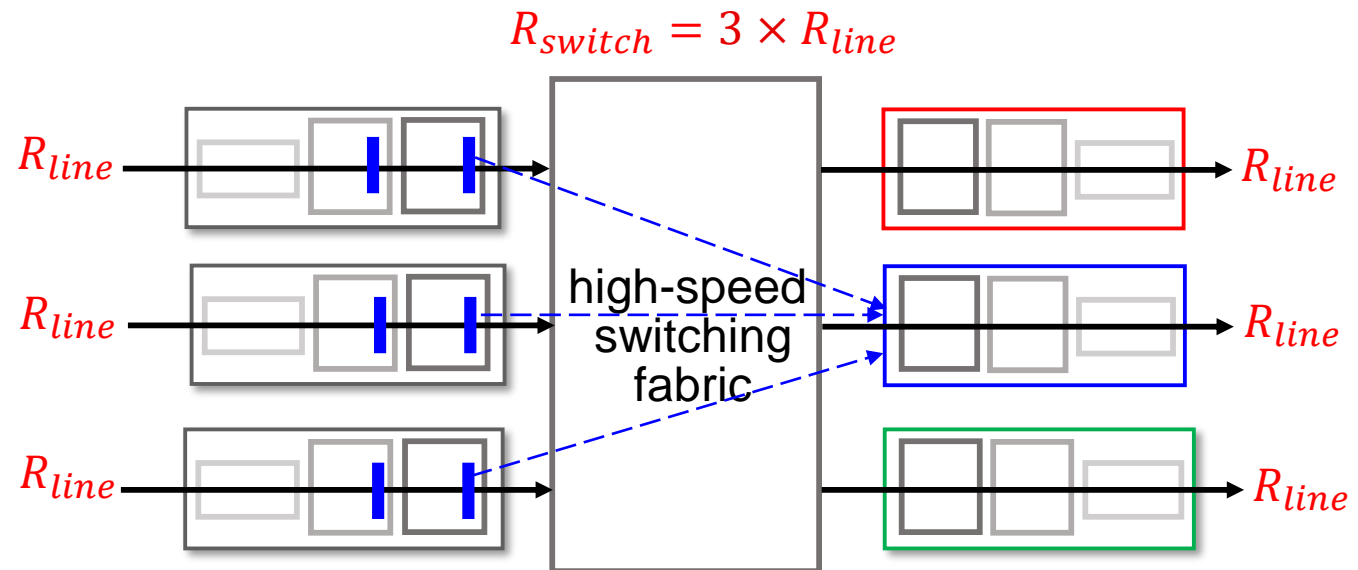
# Example: Best case

- Packets arrival to input ports is **synchronous**
- Only negligible queuing will occur at input ports and at output ports

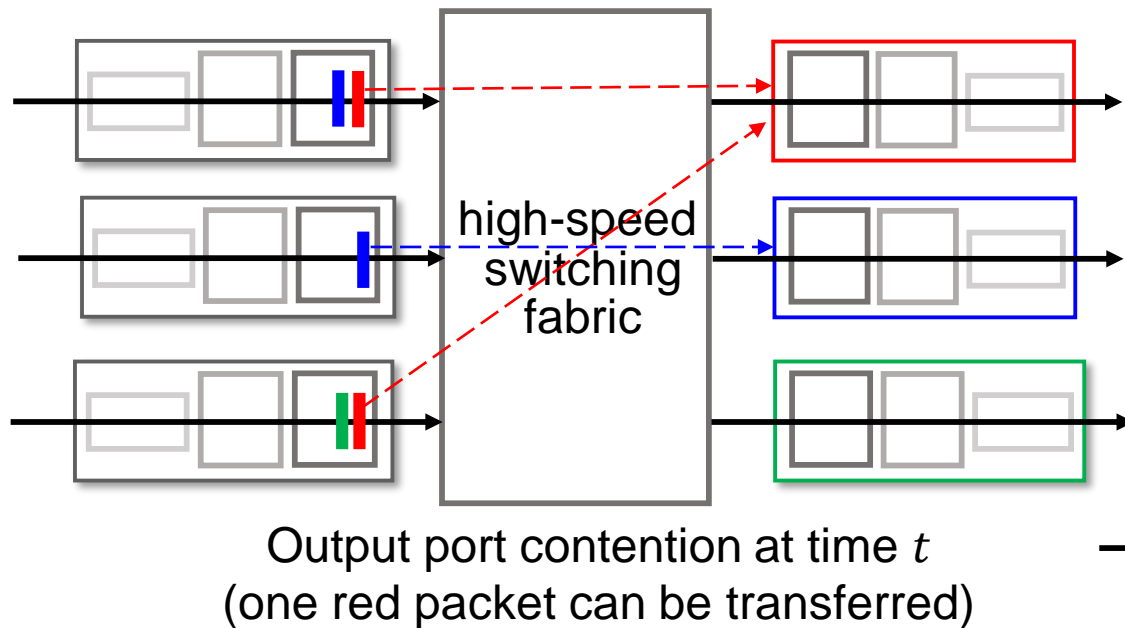


# Example: Worst case

- All  $N$  input lines are receiving packets, and
- All packets forwarded to same output port, each batch of 3 packets (one packet per input port) can be cleared through switch fabric before next batch arrives
- Negligible queuing will occur at input ports and queuing at output port
- If  $R_{switch} < N \times R_{line}$ :  
packet queuing can build up at input ports

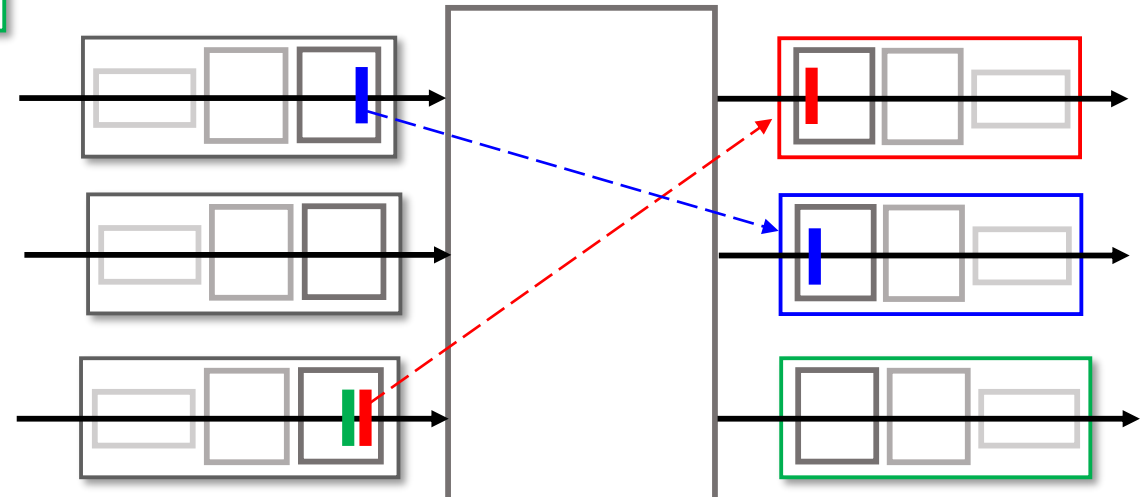


## Figure 4.8 HOL blocking at input



Example: If two packets at front of two input queues are destined for same output queue, one will be **blocked** and must wait at input

**Green packet** experiences head-of-the-line (HOL) blocking





# Queueing at output port

- Packet queues can form at output ports even when switching fabric is  $N$  times faster than port line speeds
- Suppose  $R_{switch} = N \times R_{line}$  and packets arriving at each of input ports are destined to same output port
- At output ports, in time it takes to send a single packet onto outgoing link,  $N$  new packets will arrive at this output port ( $R_{switch} = N \times R_{line}$ )
- $N$  arriving packets will have to queue (wait) for transmission over outgoing link
- $N$  more packets can possibly arrive in time it takes to transmit just one of  $N$  packets that had just previously been queued. And so on.
- When there is not enough memory to buffer an incoming packet:
  - either drop arriving packet (**drop-tail**)
  - or remove one or more already-queued packets to make room for newly arrived packet

# How Much Buffering Is “Enough?”

Packet queue will form when:

- Packet arrival rate temporarily exceeds rate at which packets can be forwarded
  - Longer amount of time that this mismatch persists, longer queue will grow

How much buffering should be provisioned at a port?

- Answer is much more complicated than one might imagine and can teach us quite a bit about **fine interaction among congestion-aware senders at network's edge and network core**

# How Much Buffering Is “Enough?”

For many years, recommendation **was** (RFC 3439):

- Amount of buffering  $B$  should be equal to an average round-trip time ( $RTT$ , say **250msec**) times port bandwidth  $R_{line}$ 
  - (Bandwidth Delay Product)

$$B = RTT \times R_{line}$$

e.g.,  $R_{line} = 10\text{Gpbs}$  port,  $B=2.4\text{Gbits}=0.3\text{GBytes}$

- Recent recommendation: When there are  $M$  TCP flows passing through **a port**, and  $M$  is very large, amount of buffering needed in that port is:

$$B = RTT \times R_{line} / \sqrt{M}$$

# Is larger buffer better?

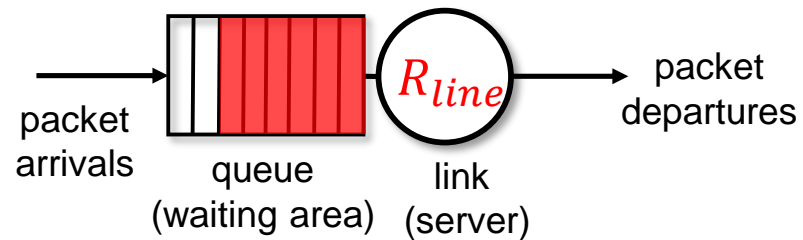
- Larger buffers allow a router to **absorb larger fluctuations in packet arrival rate**, thereby **decreasing router's packet loss rate**
- But larger buffers also mean potentially longer queueing delays
- For **gamers** and for interactive **teleconferencing** users, **tens of milliseconds count**
- Increasing amount of per-hop buffer by a factor of **10** to decrease packet loss could increase end-end delay by a factor of **10**
- **long RTTs**: make TCP senders less responsive and slower to respond to incipient congestion and/or packet loss
- **long RTTs**: delay-based congestion control keep bottleneck link full and push a lot packets in a single buffer
- These delay-based considerations show that buffering is a **double edged sword**
  - **Buffering** can be used to absorb short-term statistical fluctuations in traffic but can also lead to increased delay and attendant concerns
  - Buffering is a bit like salt—just right amount of salt makes food better, but too much makes it inedible

## 4.2.5 Packet Scheduling at output port

**Packet scheduling:** deciding which packet to send next on link

**Packet scheduler:** An algorithm at output port chooses one packet, among those queued, for transmission

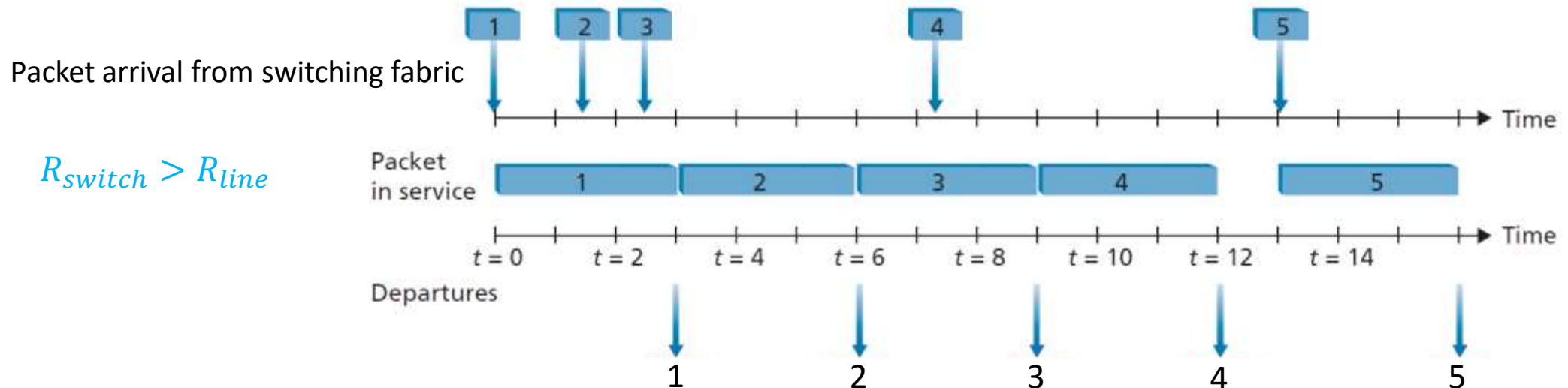
- First-in-First-Out (FIFO) or First-come-first-served (FCFS)
- Priority
- Round robin
- Weighted fair queueing



**Figure 4.11** FIFO queueing abstraction at output port

# Example of First-in-First-Out (FIFO)

- Number indicating order in which packet arrived
- Time a packet spends in service = **Packet transmission time**



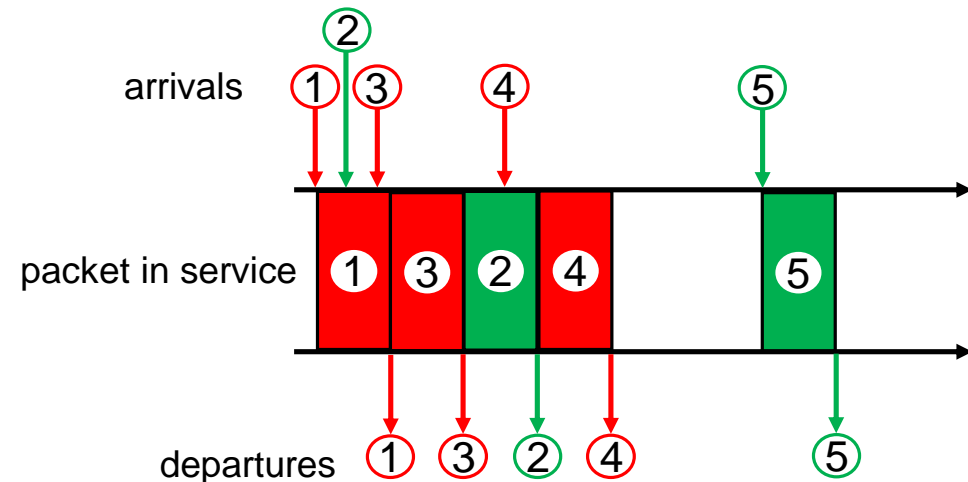
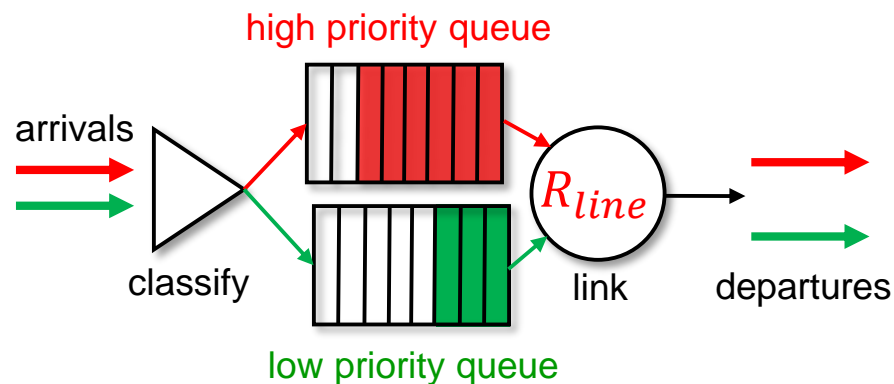
**Figure 4.12** The FIFO queue in operation

# Priority Queuing

- Arriving traffic classified, queued by class
  - In practice, a network operator may configure routers so that packets carrying network management information (indicated by source or destination port number) receive priority over user traffic
  - Real-time voice-over-IP packets might receive priority over non-real-time traffic such e-mail packet
- Send packet from highest priority queue that has buffered packets
  - choice among packets in same priority class is typically done in a FIFO manner

# Priority Queuing

- Packet 4 arrives during transmission of packet 2 (a low-priority packet). Under a **non-preemptive priority queuing** discipline, transmission of a packet is not interrupted once it has begun
  - packet 4 queues and is transmitted after packet 2 is completed



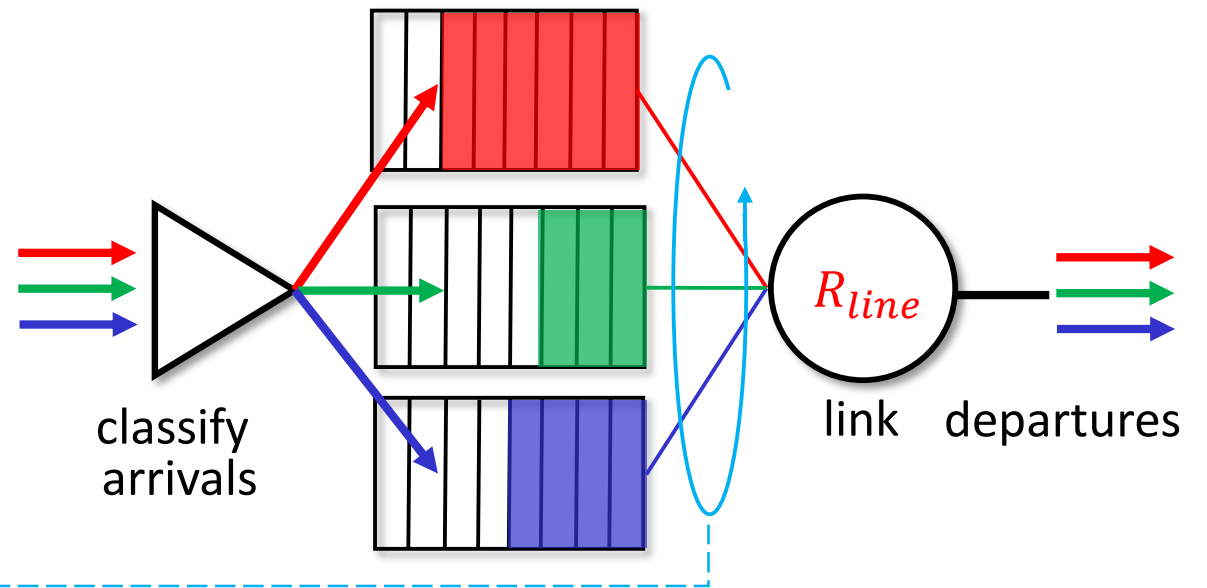
**Figure 4.13** The priority queueing model



# Round Robin and Weighted Fair Queuing (WFQ)

## Round Robin (RR) scheduling:

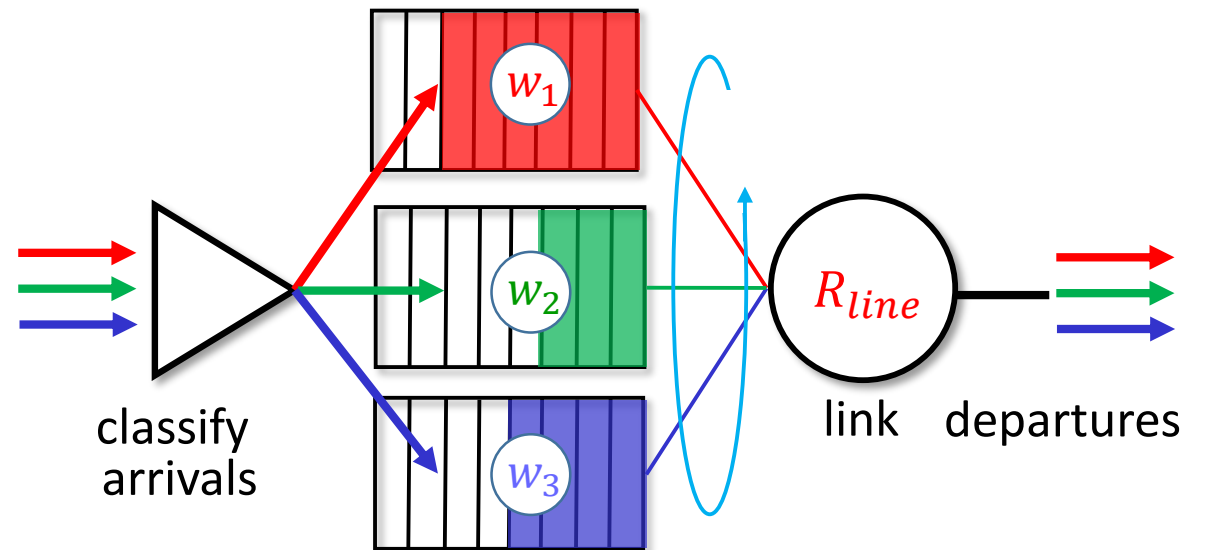
- arriving traffic classified, queued by class
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) **in turn**



# Round Robin and Weighted Fair Queuing (WFQ)

## Weighted Fair Queuing (WFQ):

- Generalized Round Robin
- Each class,  $i$ , has weight,  $w_i$ , and gets weighted amount of service in each cycle:  $w_i / \sum w_j$
- For an output link with transmission rate  $R_{line}$ , class  $i$  will always achieve a throughput of at least  $R_{line} \times w_i / \sum w_j$



# Contents

4.1 - Overview of Network Layer

4.2 - What's Inside a Router?

**4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More**

4.4 - Generalized Forwarding and SDN

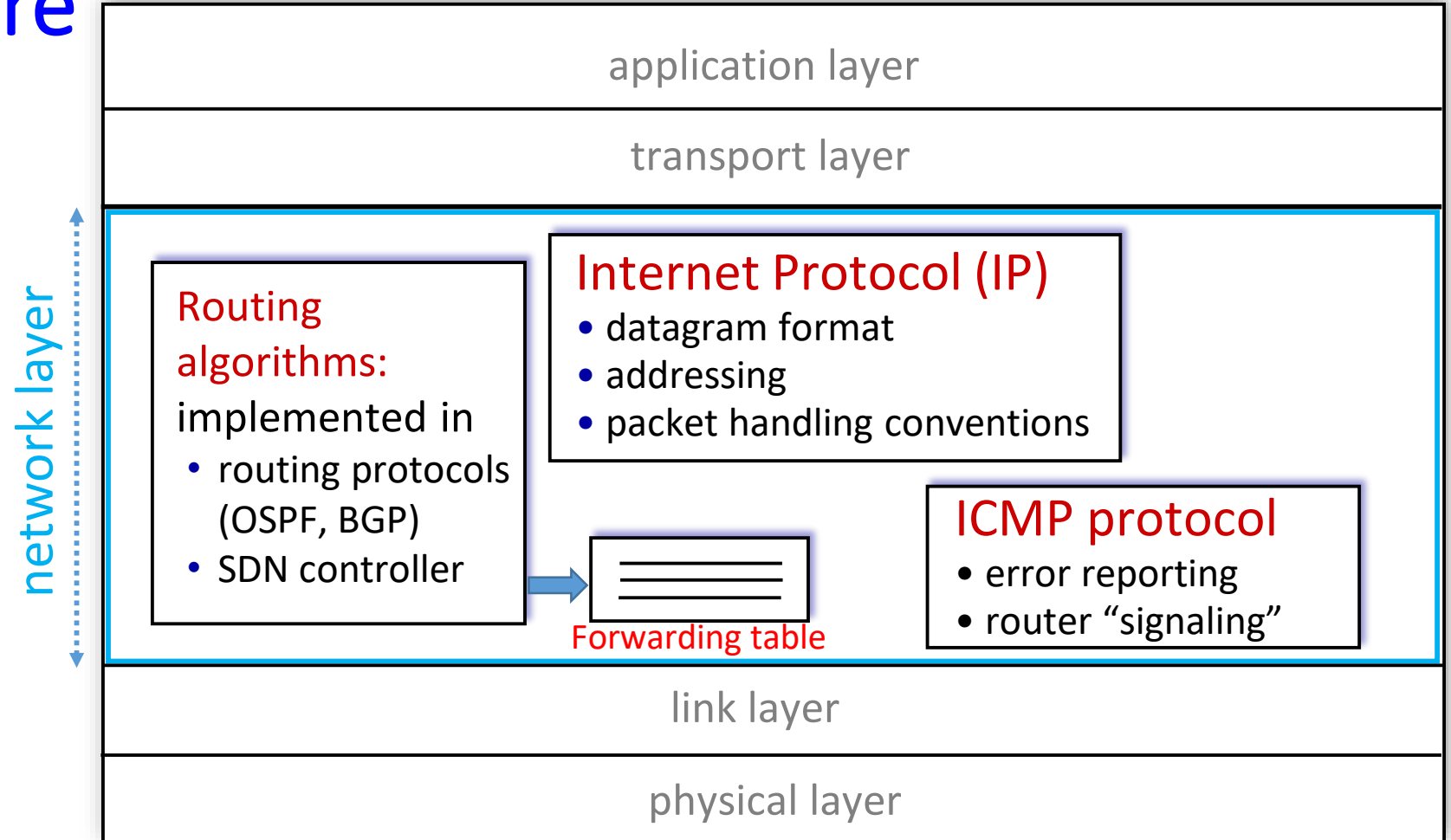
4.5 – Middleboxes

4.6 - Summary

## 4.3 The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

Two versions of IP in use:

- widely deployed IP version 4
- IP version 6

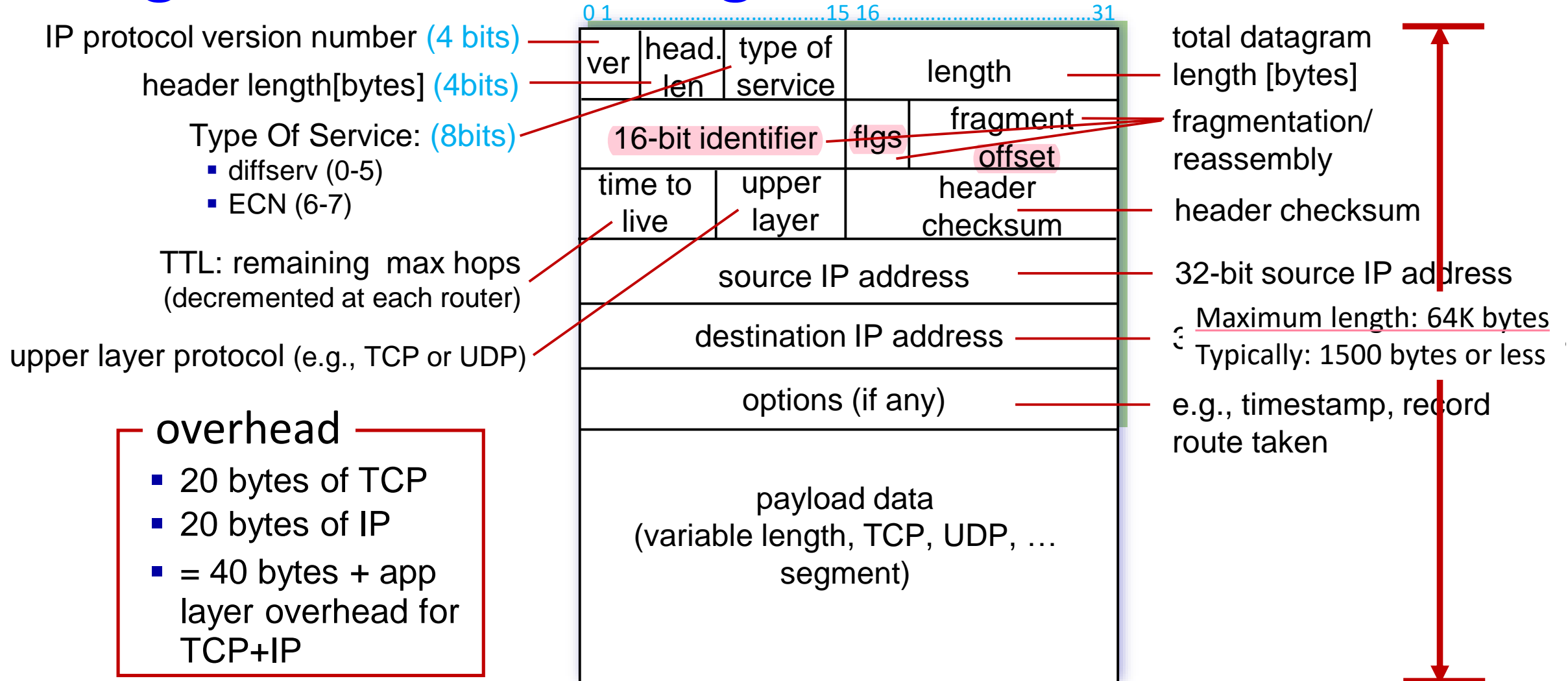


**ICMP: Internet Control Message Protocol (host, router)**

## 4.3.1 IPv4 Datagram Format

- Internet's network-layer packet is referred to as a **datagram**
- Datagram plays a central role in Internet
- Key fields in IPv4 datagram are following: (next slide)

# Figure 4.17 IPv4 datagram format



# IPv4 datagram format-Datagram length

- **Datagram length:** Total length of IP datagram (header plus data), measured in bytes
- It is 16 bits long, theoretical maximum size of IP datagram is 65,535 bytes
- However, datagrams are rarely larger than 1,500 bytes, which allows an IP datagram to fit in payload field of a maximally sized **Ethernet frame**

# IPv4 datagram format-Type Of Services

- **Type of service:** **TOS** bits are included in IPv4 header to allow different types of IP datagrams to be distinguished from each other.
- **diffserv** (Differentiated services), 6 bits (0-5) **classifying** and managing network traffic and **providing quality of service** (QoS)
- Example: distinguish real-time datagrams (such as those used by an IP telephony application) from non-real-time traffic (for example, FTP)
  - Specific **level of service** to be provided is a policy issue determined and configured by network administrator for a router
- **Two** of TOS bits are used for **Explicit Congestion Notification (ECN)**



# IPv4 datagram format-Header checksum

- **Header checksum:** Detecting bit errors in a header of received IP datagram. Each 2bytes in header as a number, summing these numbers using 1s complement arithmetic.
  - Routers discard datagrams when error detected
  - IP header is checksummed at IP layer, TCP/UDP checksum is computed over entire TCP/UDP segment
  - Checksum is recomputed and stored again at each router, since TTL field, and possibly options field , will change
- **Why error checking at both transport and network layers?**
  - TCP/UDP and IP do not necessarily both have to belong to same protocol stack
  - TCP can, run over a different network-layer protocol (for example, ATM) and IP can carry data that will not be passed to TCP/UDP

# Protocol field

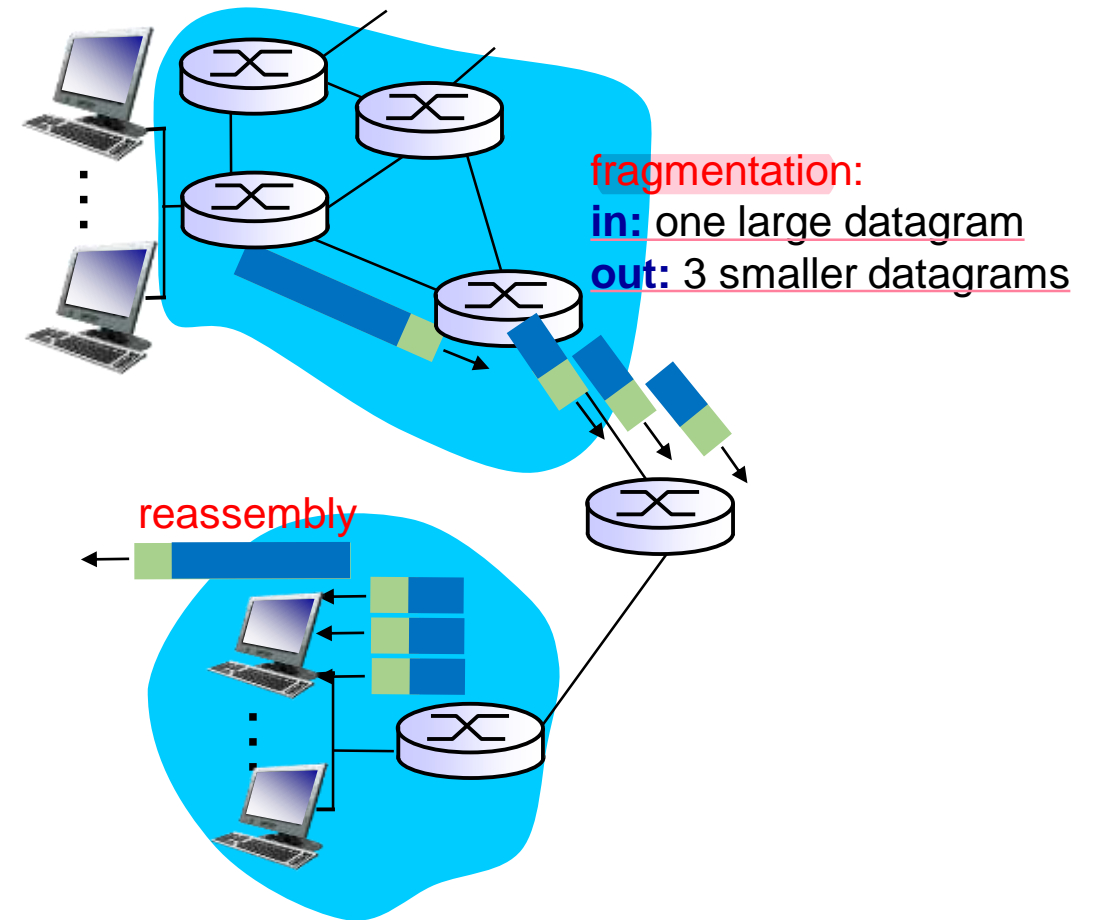
- **Protocol:** This field is typically used only when an IP datagram reaches its final destination
- It indicates **specific protocol** payload data is passed ( 1 for ICMP, 4 for IP-in-IP, 6 for TCP, 8 for **Exterior Gateway Protocol (EGP)**, 17 for UDP, ...)
- It has a role that is analogous to role of **port number field in transport-layer segment**
  - Port number is glue that binds transport and application layers together
  - Protocol number is glue that binds network and transport layers together,
  - Link-layer frame also has a special field that binds link layer to network layer (Chapter 6)

# Options field

- **Options:** Datagram headers can be of variable length because of header options it carries
  - timestamp, record route taken, ...
- Some datagrams may require options processing and others may not, so, amount of time needed to process an IP datagram at a router can vary greatly
- This become important for IP processing in high performance routers and hosts

# IP fragmentation, reassembly

- Network layer technology has its own largest possible link-level frame (Maximum Transfer Unit-MTU)
  - Example: Ethernet's MTU=1500Bytes, FFDI's MTU=4000 Bytes, ...
- IP datagram divided (“fragmented”) in routers (one datagram becomes several datagrams) and “reassembled” at final destination
- IP header Identifier, Flags and Fragment offset are used to “reassembled” a fragmented packet at final destination



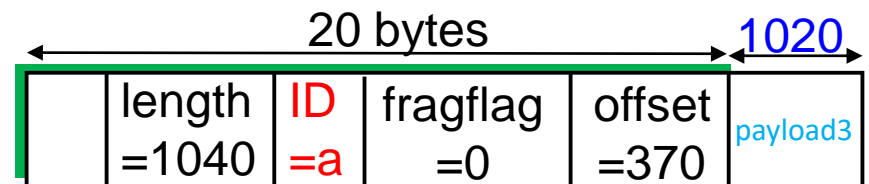
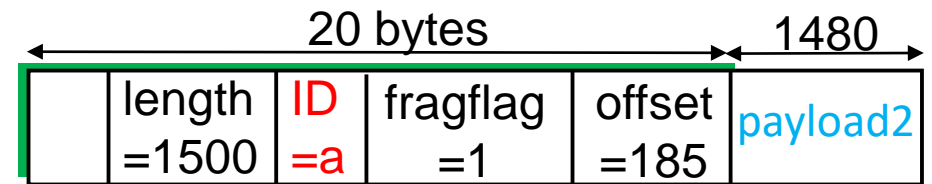
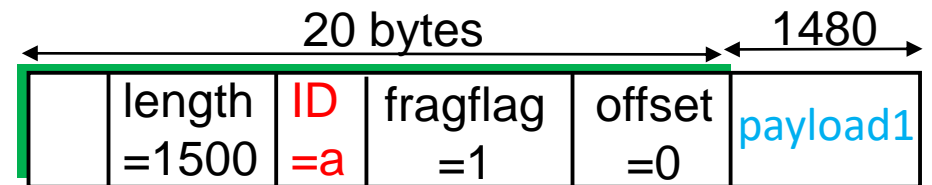
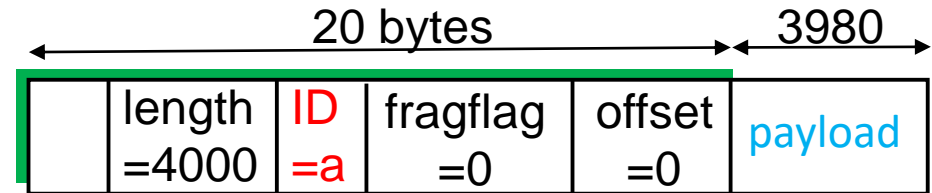
# Example: Fragmentation and reassembly

- A 4000 byte datagram enters into a router
- Output interface of router has MTU=1500Bytes. So datagram is fragmented to 3 datagrams

more fragment flag =1, offset = 0

more fragment flag =1, offset =  $1480/8 = 185$

more fragment flag =0, offset =  $1480/8 + 1480/8 = 370$




$$1020 = 3980 - 1480 - 1480$$

## 4.3.2 IPv4 Addressing

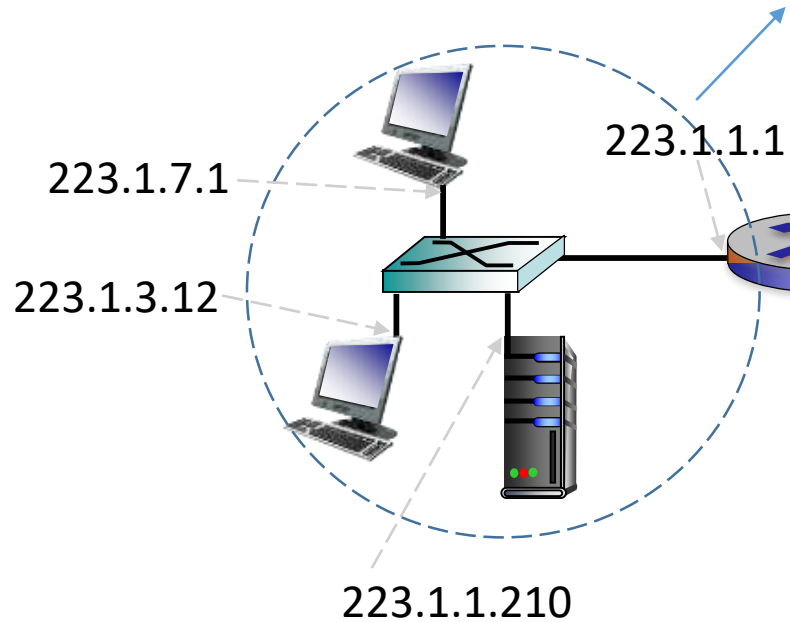
- A **host** typically has only a single link (interface, port) into network
- A **router** necessarily has two or more **interface (Input/output pair)** into network
- **Each host and router interface** has its own IP address (IP interface)
- IP address is associated with an interface
- Dotted-decimal IP address notation: 223.1.1.1 =  
= 11011111 00000001 00000001 00000001

# Internet's address assignment strategy

- $x$  most significant bits of an address of form  $a.b.c.d/x$  is network portion of IP address, referred to as **prefix** of address
- An organization is typically assigned a range of addresses with a **common prefix**, IP addresses of devices within organization will **share common prefix**
- A  $/x$  subnet can contain up to  $2^{32-x} - 2$  IP interfaces,  $2 \leq x \leq 30$
- In a subnet, all IP interfaces have similar leftmost  $x$  bits
- Example: IP address of a subnet 223.1.1.1/24 
  - $/24$  is known as a **subnet mask**, leftmost 24 bits of 32-bit quantity define **subnet address**
  - 32-bit IP address is divided into two parts and has dotted-decimal form  $a.b.c.d/x$ , where  $x$  indicates number of bits in left part of address
  - **Classless Interdomain Routing** (CIDR—pronounced **cider**) [RFC 4632]

# Subnet

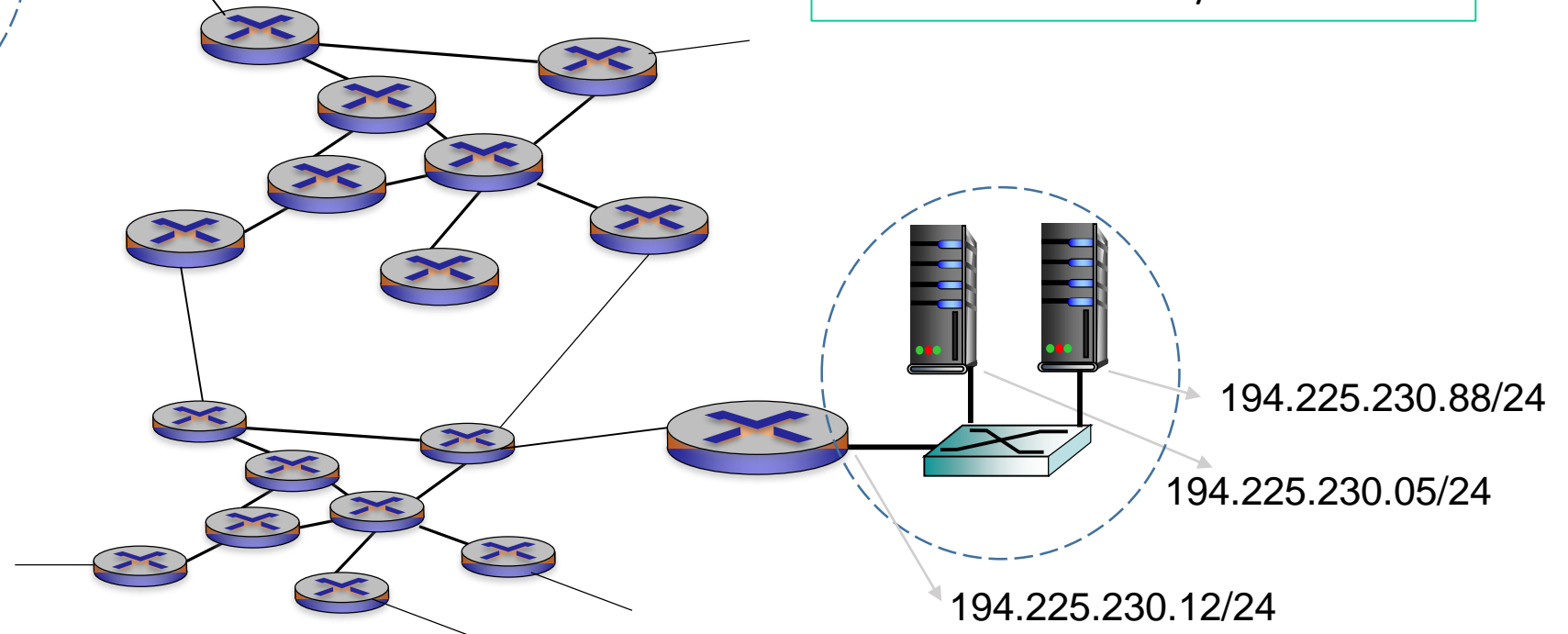
Example: IP address  $11011111\ 00000001\ 00000001\ 00000001 = 223.1.1.1$   
 A portion of IP address (red) is determined by **subnet** to which interface is located



Subnet mask:  $11111111\ 11111111\ 11111000\ 00000000$

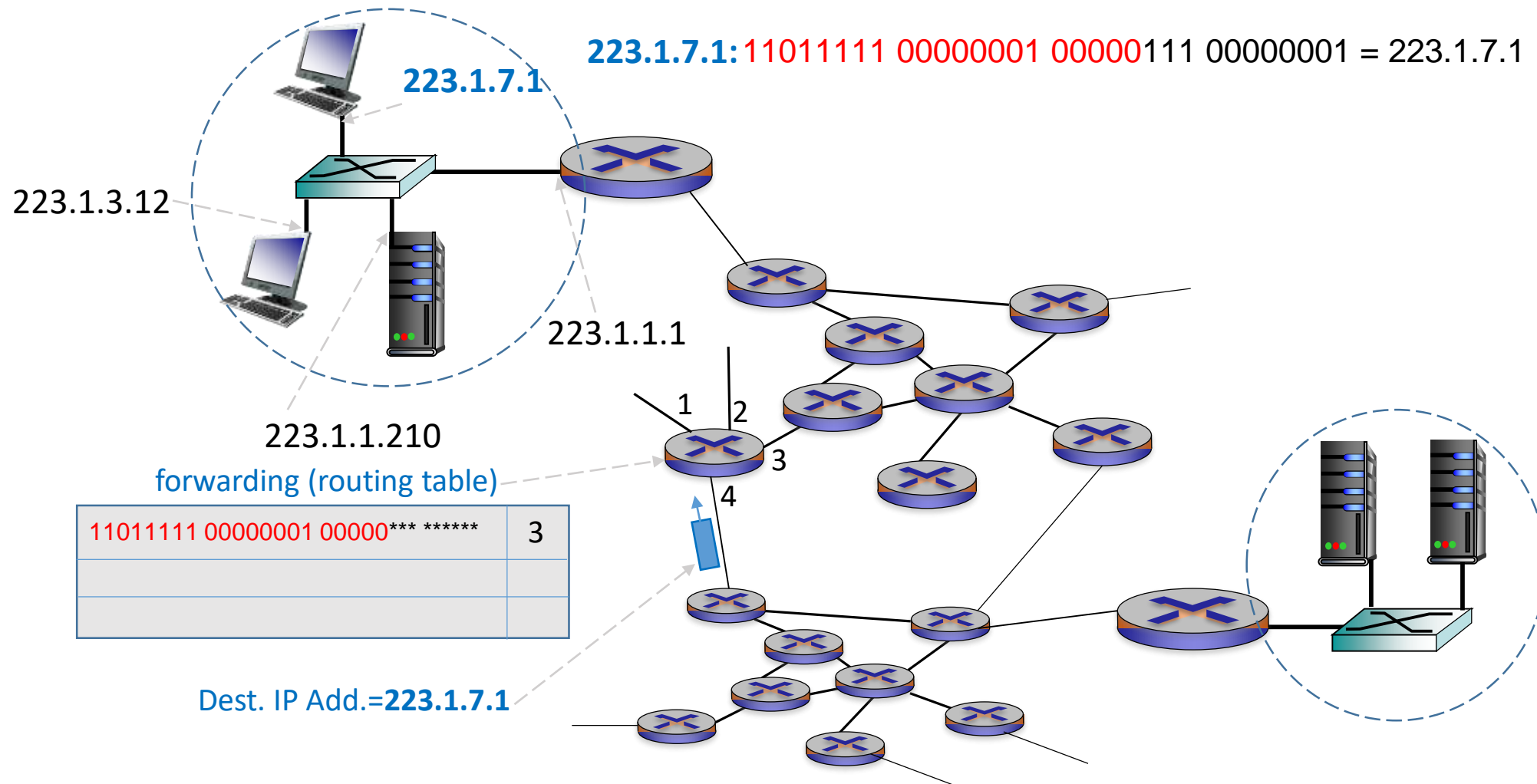
How we write interface IP address together with subnet mask:  
 223.1.1.1/21

A **subnet** is a routerless network (contains no routers) and also called **IP network**





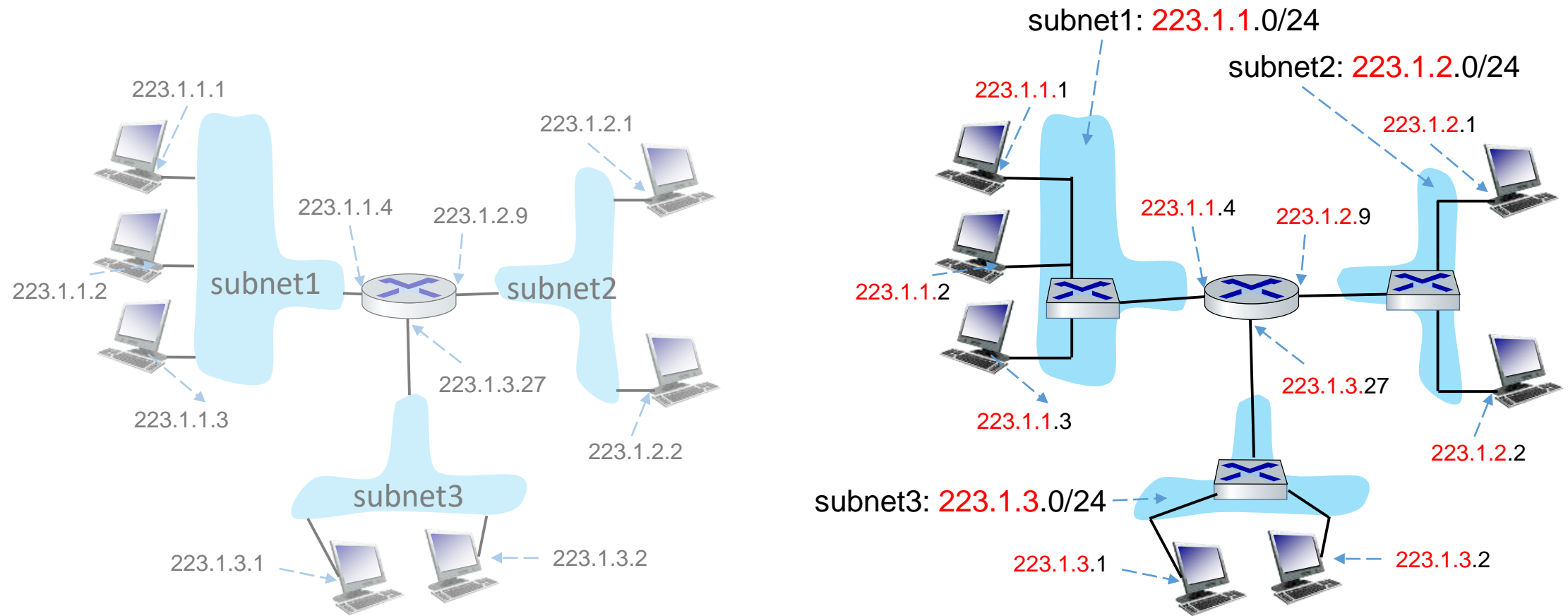
# Routing table: Prefix



# Routing table: Prefix

- Only  $x$  leading prefix bits of IP address are considered by routers outside organization's network
- A router outside organization forwards a datagram whose destination address is inside organization, only leading  $x$  bits of address need be considered
- This reduces size of forwarding table in routers, a single entry of form  $a.b.c.d/x$  will be sufficient to forward packets to any destination within organization

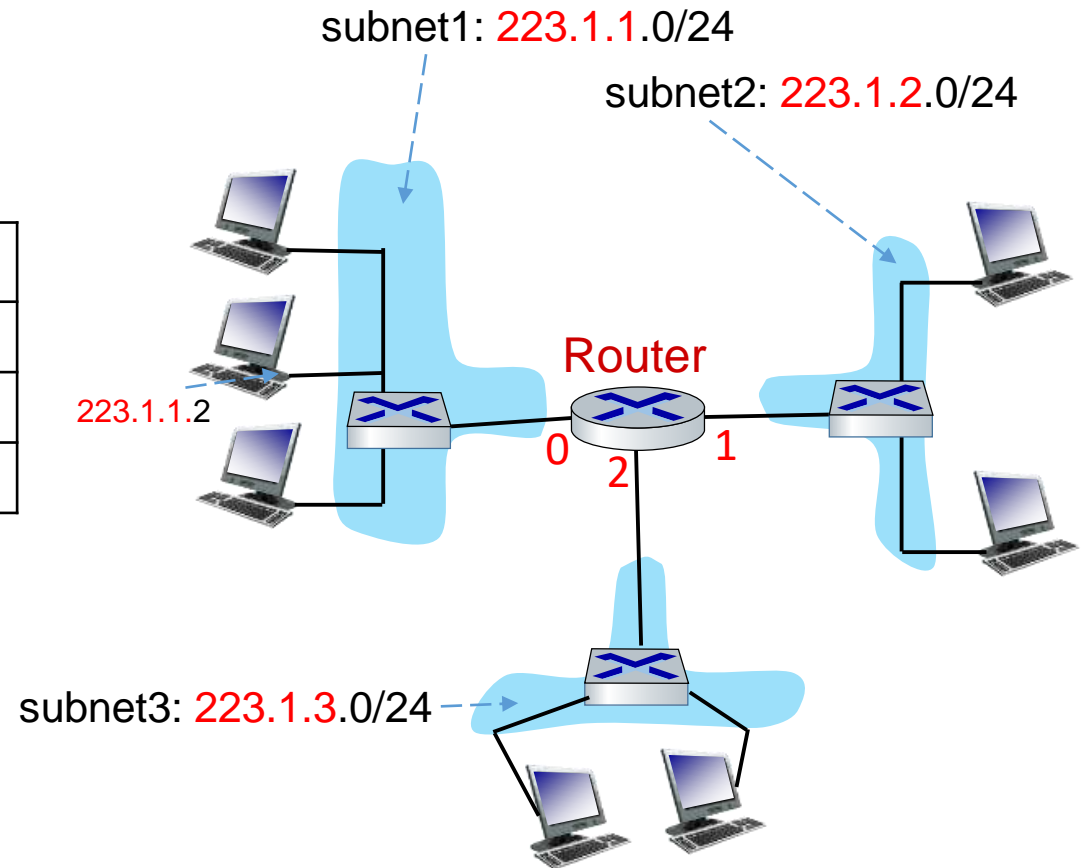
# Figure 4.18 Interface addresses and subnets



# Forwarding Table and Subnets

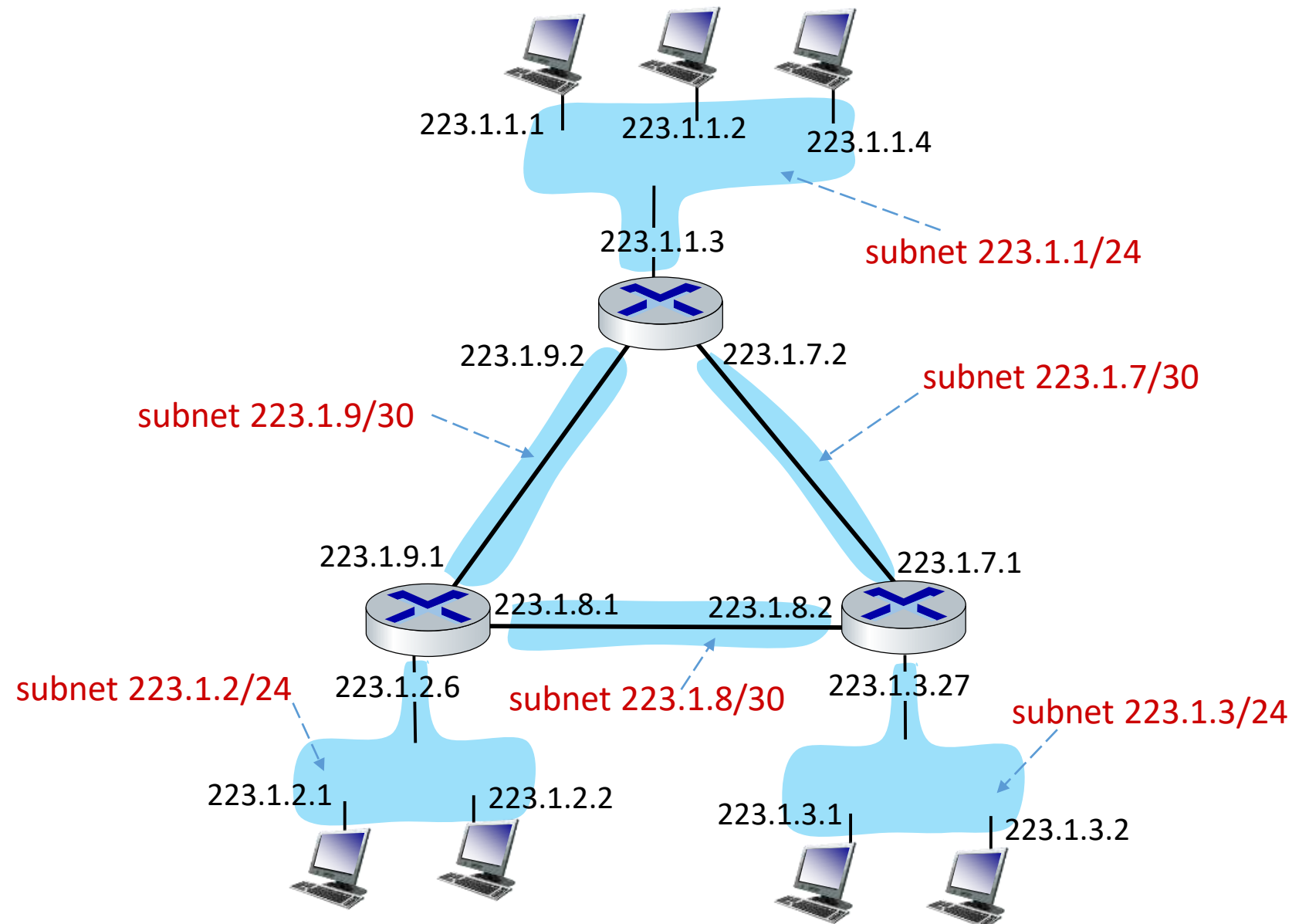
Forwarding table of Router

Destination Address Range	Link interface
11011111 00000001 00000001 *****	0
11011111 00000001 00000002 *****	1
11011111 00000001 00000003 *****	2



# Figure 4.20

- Three routers interconnecting six subnets



# Obtaining a Block of Addresses for ISPs

- IP addresses are managed under authority of **Internet Corporation for Assigned Names and Numbers (ICANN, <http://icann.org>)**
- ISPs obtain block of address from **Address Supporting Organization of ICANN**
- ICANN allocates addresses to **4 regional Internet registries:**
  - **ARIN, RIPE, APNIC, LACNIC**, which together form **Address Supporting Organization of ICANN**, and handle allocation/management of addresses within their regions. ICANN allocated last chunk of IPv4 addresses to ASOs in **2011**
- ICANN also manage DNS root servers, assign domain names and resolving domain name disputes

# Obtaining a Block of Addresses for organizations

- Obtaining a block of IP addresses for use within an organization's subnet, a network administrator might first contact its ISP, which would provide addresses from a larger block of addresses that had already been allocated to ISP
- Example, ISP itself have been allocated address block 200.23.16.0/20

11001000 00010111 00010000 00000000 200.23.16.0/20

- ISP could divide its address block into eight equal-sized contiguous address blocks and give one of these address blocks out to each of up to eight organizations that are supported by this ISP:

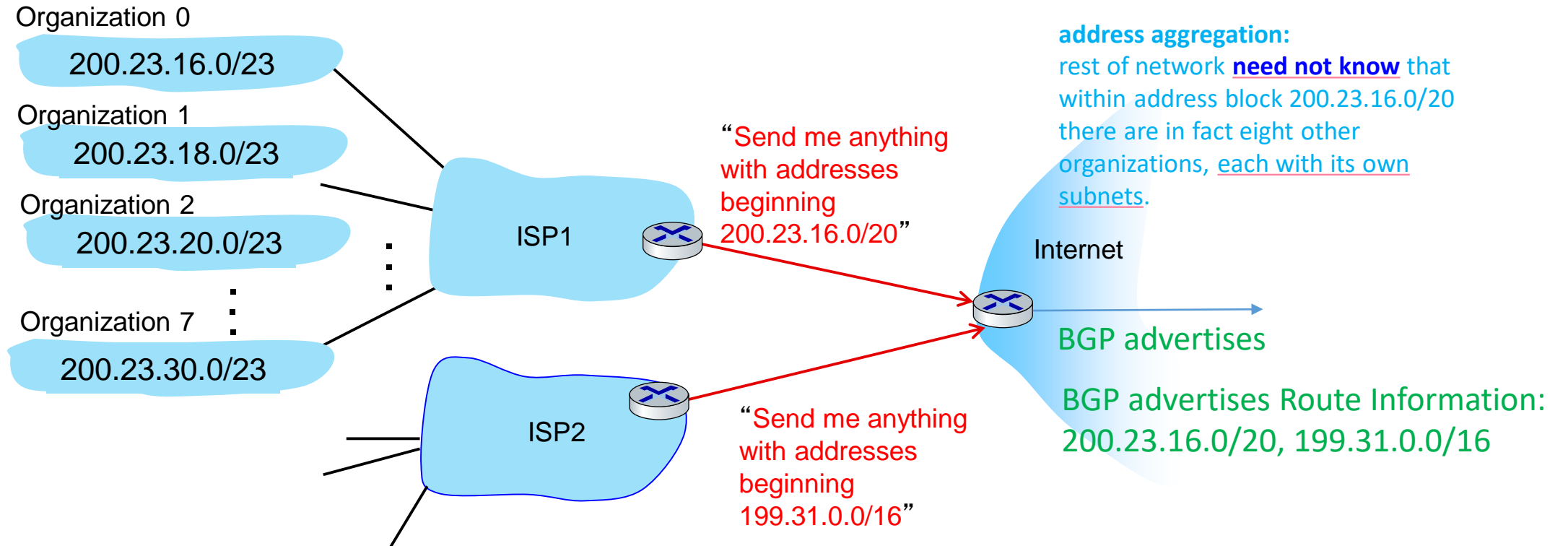
Organization 0	11001000	00010111	0001 <u>000</u> 0	00000000	200.23.16.0/23
Organization 1	11001000	00010111	0001 <u>001</u> 0	00000000	200.23.18.0/23
Organization 2	11001000	00010111	0001 <u>010</u> 0	00000000	200.23.20.0/23
.....					
Organization 7	11001000	00010111	0001 <u>111</u> 0	00000000	200.23.30.0/23

# Address aggregation (route aggregation or route summarization)

- Suppose, as shown in Figure 4.21, that ISP (which we'll call Fly-By-Night-ISP) advertises to outside world that it should be sent any datagrams whose first 20 address bits match 200.23.16.0/20
- Rest of world need not know that within address block 200.23.16.0/20 there are in fact eight other organizations, each with its own subnets.
- This ability to use a single prefix to advertise multiple networks is often referred to as address aggregation (also route aggregation or **route summarization**)

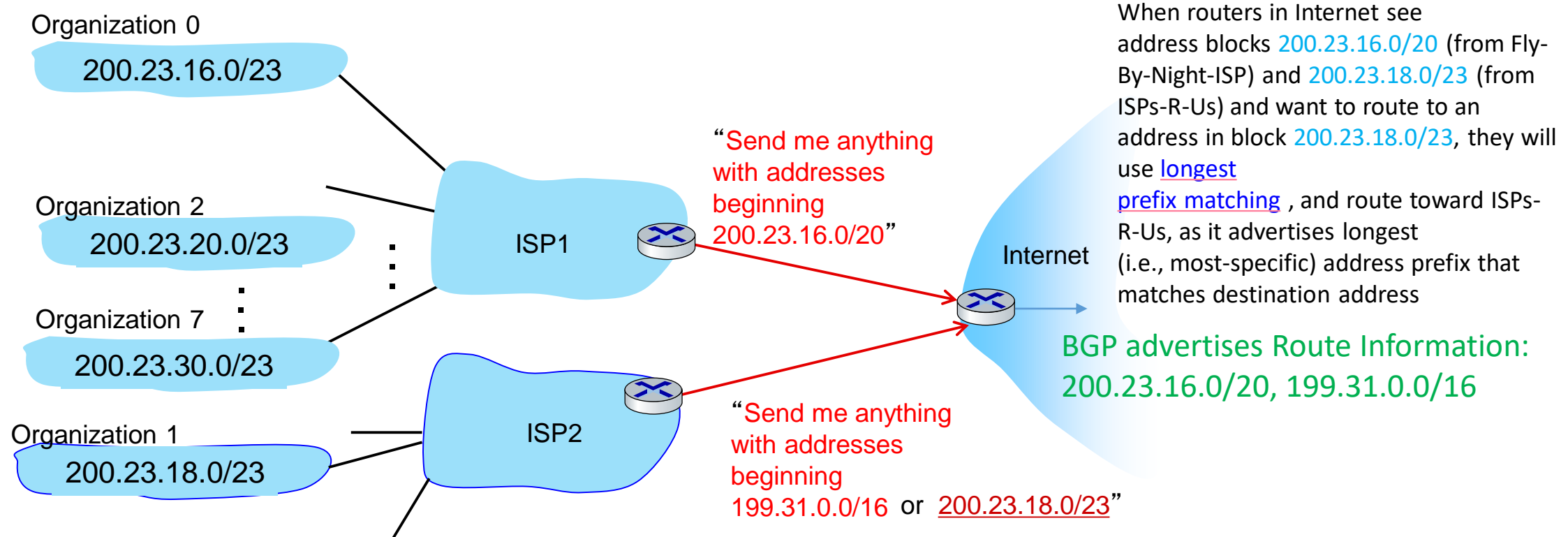


# Figure 4.21 Hierarchical addressing and route aggregation



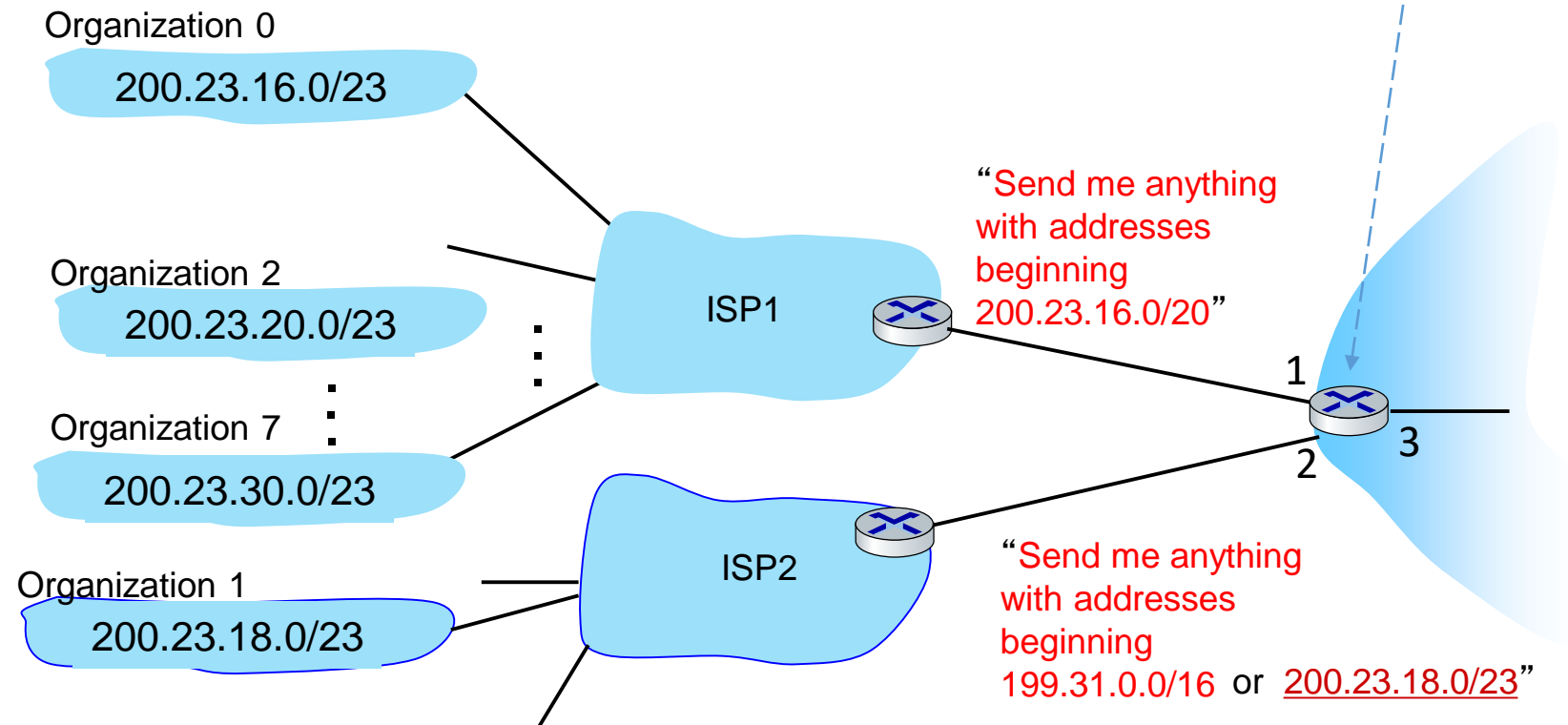
# Figure 4.22 ISPs-R-Us has a more specific route to Organization 1

- Organization 1 moves from ISP1 to ISP2
- ISPs-R-Us now advertises a more specific route to Organization 1



# Figure 4.22 ISP2 has a more specific route to Organization 1

Prefix	Interface
11001000 00010111 0001**** *	1
11001000 00010111 0001001* *	2
11000111 00011111 ***** *	2



# Loopback Interface

- Any packet that a computer sends on loopback network is addressed to same computer
- IP specifies a loopback network with IPv4 address 127.0.0.0/8 and IPv6 address ::1
- Most IP implementations support a loopback interface to represent loopback facility
- Loopback interface does not associate with **hardware interface**
- Standard domain name for the address is "localhost"
  - `http://127.0.0.1` or `http://localhost`

# Obtaining a Host Address: The Dynamic Host Configuration Protocol (DHCP)

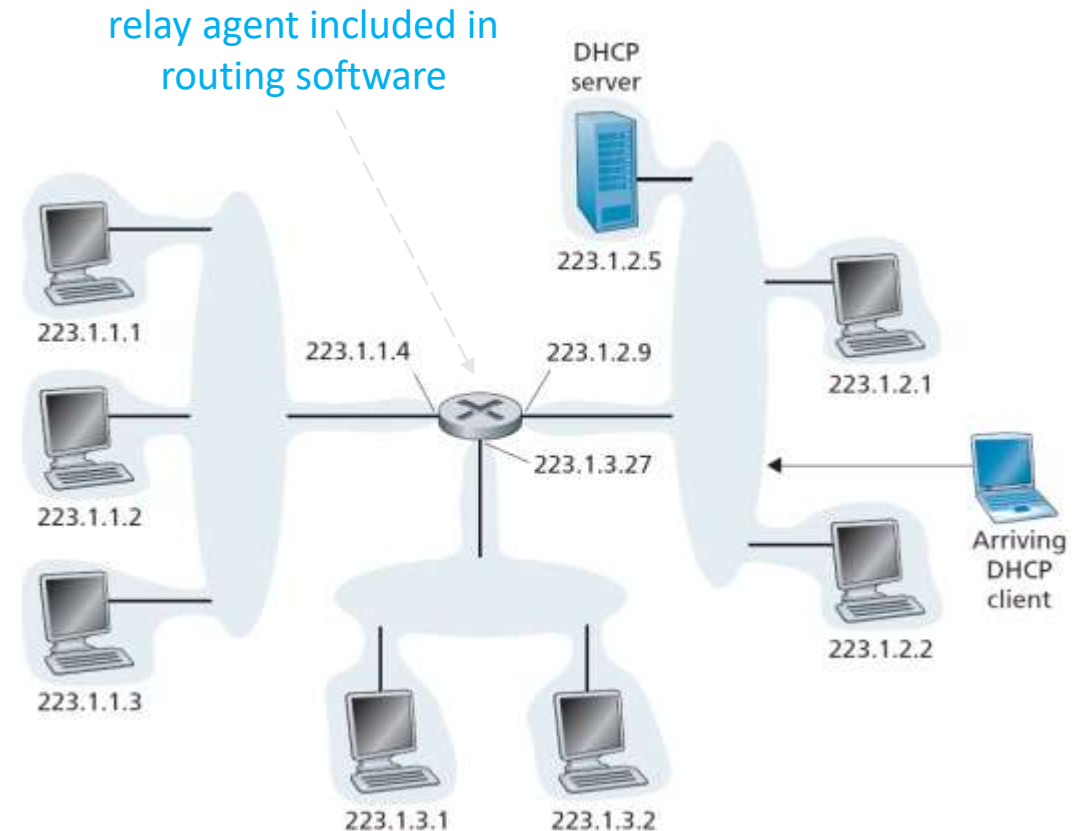
- An organization assigns individual IP addresses to host and router interfaces in its organization
- A system administrator will typically **manually configure IP addresses into routers** (often remotely, with a network management tool) and **servers**
- **Host** addresses can also be configured manually, but typically this is done using **Dynamic Host Configuration Protocol (DHCP)** [RFC 2131]
- DHCP allows a host to obtain (be allocated) an IP address automatically
- A network administrator can configure DHCP so that a given host receives **same IP address** each time it connects to network, or a host may be assigned a **temporary IP address** that will be different each time host connects to network
- DHCP also allows a host to learn its **subnet mask**, address of its **first-hop router** (called **default gateway**), and addresses of its **local DNS servers**

# DHCP client/server

- DHCP is a client-server protocol. A client is typically a newly arriving host wanting to obtain network configuration information, including an IP address for itself
- In simplest case, each subnet will have a DHCP server
- If no server is present on subnet, a DHCP relay agent (a router or a host) that knows address of a DHCP server for that network is needed
- Router's software include a DHCP server and relay agent software
  - Relay agents receive DHCP messages and then generate a new DHCP message to send DHCP server
  - Relay agent sets gateway IP address (giaddr field of DHCP packet) and, if configured, adds relay agent information option (option82) in packet and forwards it to DHCP server
  - Reply from server is forwarded back to client after removing option 82

# DHCP: 4-step process

- DHCP protocol is a **four-step process**:
  - 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



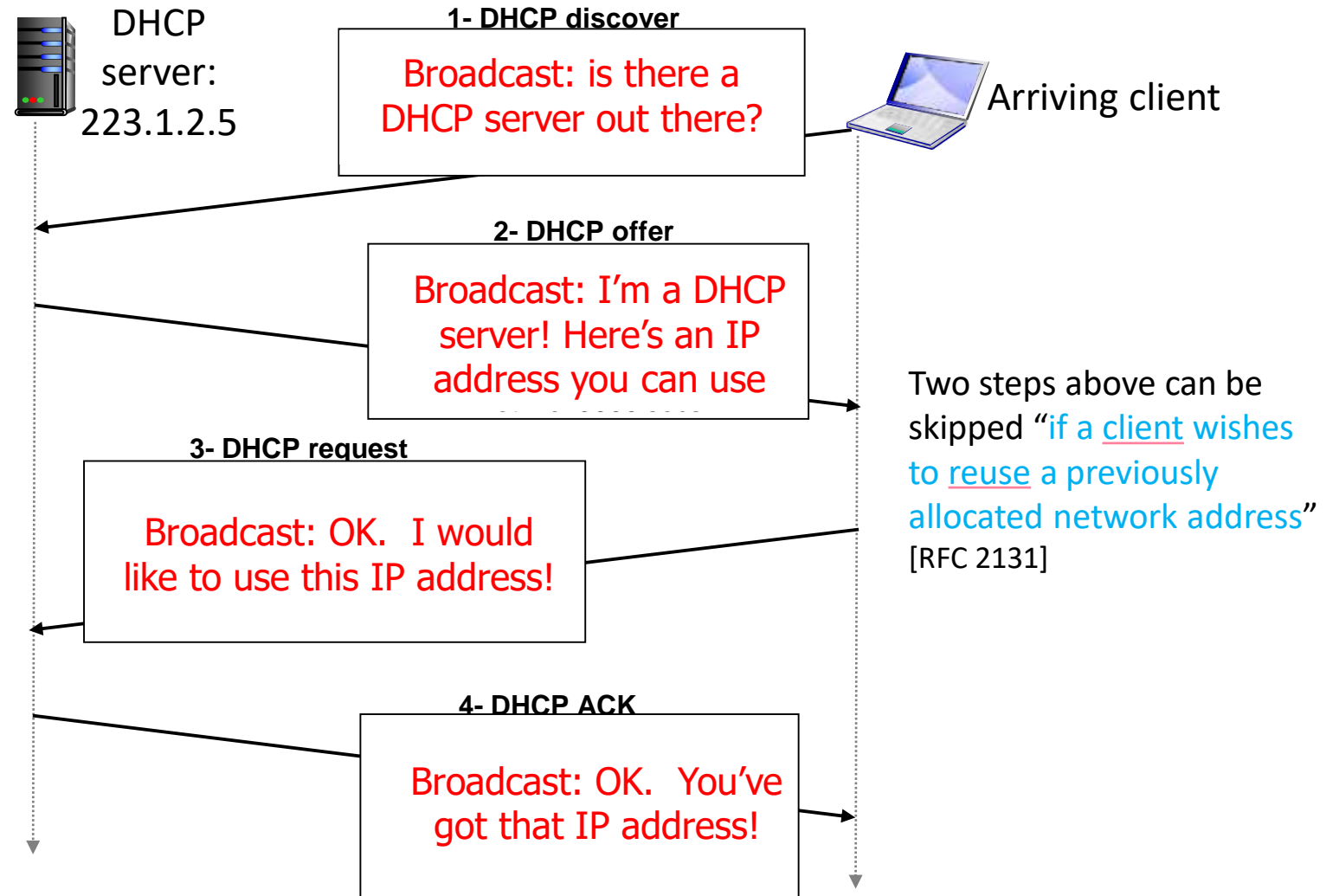
**Figure 4.23** DHCP client and server, DHCP server is available on subnet in which client arrives

# Figure 4.24 DHCP client-server interaction

DHCP uses default **UDP** port numbers **67** for server and **68** for client

Transaction ID: A random 32-bit number chosen by client, used by client and server to associate messages and responses between client and server

`yiaddr`: server offers an **IP address** to client





# DHCP server discovery

- **DHCP server discovery:** Client sends a **DHCP discover message** within a UDP packet to port 67
- Host **doesn't know** IP address of **subnet** to which it is attaching, and IP address of a **DHCP server** for this network
- So, DHCP discover message is broadcasted (destination IP address: 255.255.255.255. Source IP address of client is 0.0.0.0. at this stage)
- DHCP client passes IP datagram to **link layer**, which then **broadcasts** this frame to all nodes attached to subnet (link-layer broadcasting in Section 6.4)

# DHCP server offer(s)

- **DHCP server offer(s)**: A DHCP server receiving a DHCP discover message responds to client with a **DHCP offer message** that is broadcast to all nodes on subnet, again using IP broadcast address of 255.255.255.255
- **Several DHCP servers can be present on subnet**, client may find itself in enviable position of being able to **choose from among several offers**
- Each server offer message contains **transaction ID** of received discover message, proposed IP address for client, network mask, and an IP **address lease time** (amount of time for which IP address will be valid)
- It is common for server to set lease time to several hours or days

# DHCP request, DHCP ACK

- **DHCP request:** Client will choose from among one or more server offers and respond to its selected offer with a **DHCP request message**, echoing back configuration parameters
- **DHCP ACK:** Server responds to DHCP request message with a **DHCP ACK message**, confirming requested parameters
- Once client receives DHCP ACK, interaction is complete and client can use DHCP-allocated IP address for lease duration

# More on DHCP

- Since a client may want to use its address beyond the lease's expiration, DHCP also provides a mechanism that allows a client to **renew its lease on an IP address** (DHCP discover and DHCP offer are not needed for renewal)
- Since a new IP address is obtained from DHCP each time a node connects to a new subnet, a **TCP connection to a remote application cannot be maintained as a mobile node moves between subnets**
- **Mobile cellular networks** allow a host to retain its IP address and ongoing TCP connections as it moves between base stations in a provider's cellular network

# Recall from Chapter 3: Connection migration

- **QUIC** uses a connection ID and can survive changes to endpoint addresses (IP address and port)
- During a QUIC connection, an endpoint can migrating to a new network using a new IP address and even new port number
- An endpoint **MUST NOT** initiate connection migration before handshake is confirmed

# Reserved private IPv4 network ranges

- Computers in private networks need not have globally unique IP addresses
- Private networks are widely used and connect to Internet using network address translation (NAT)

CIDR block	Address range	Number of addresses
10.0.0.0/8	10.0.0.0 – 10.255.255.255	16777216
172.16.0.0/12	172.16.0.0 – 172.31.255.255	1048576
192.168.0.0/16	192.168.0.0 – 192.168.255.255	65536

## 4.3.3 Network Address Translation (NAT)

- **NAT** [RFC 2663; RFC 3022]: all devices in local network share just **one** IPv4 address as far as outside world is concerned
- NAT-enabled router does not **look** like a router to outside world. NAT router behaves to outside world as a **single** device with a **single** IP address
- All traffic leaving router for Internet has a source IP address of 138.76.29.7, and all traffic entering router must have a destination address of 138.76.29.7
- **NAT-enabled router is hiding details of home network from outside world**

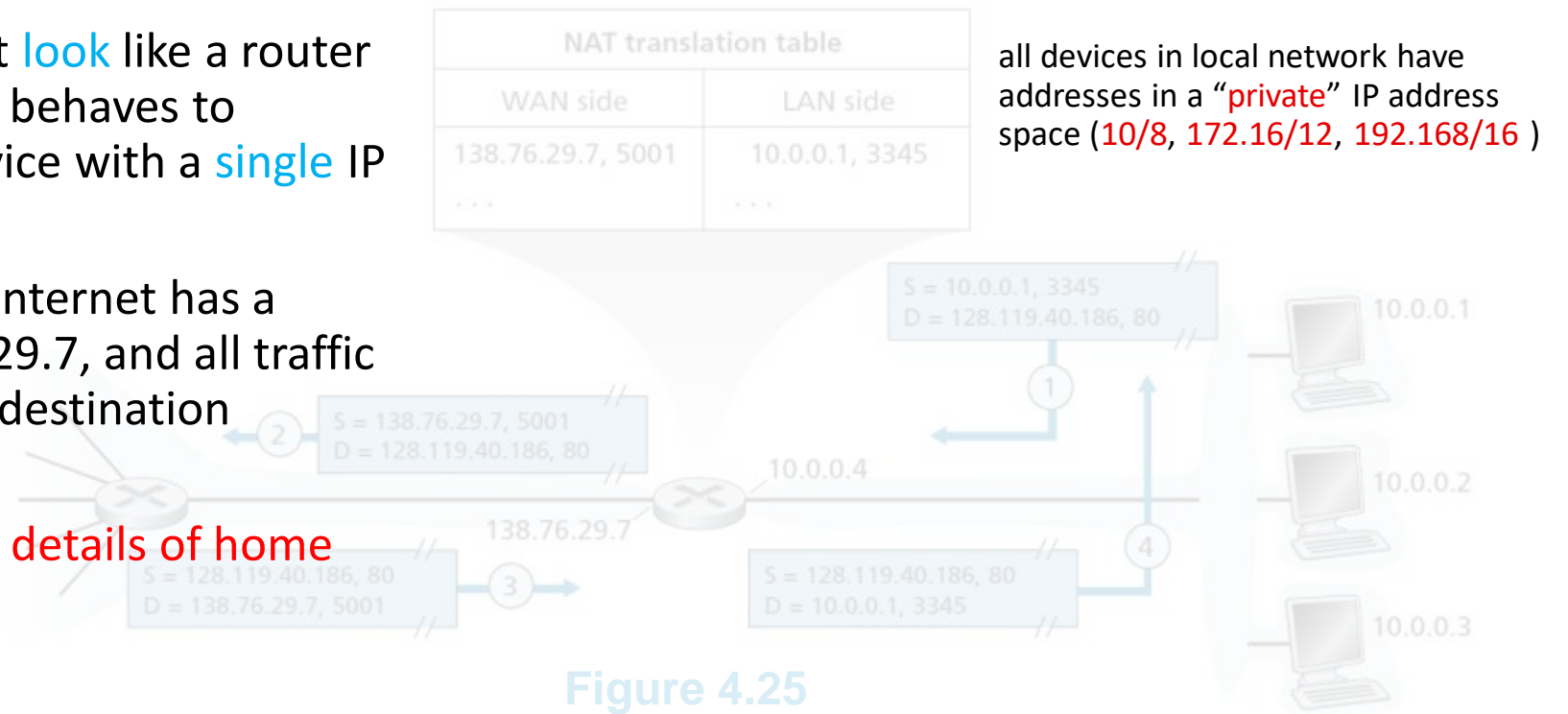


Figure 4.25

## 4.3.4 IPv6

### Motivations for introducing IPv6:

#### 1. Address space

- In February 2011, IANA allocated out last remaining pool of unassigned IPv4 addresses to a regional registry
- Once these addresses are exhausted, there are no more available address blocks that can be allocated from a central pool
- Considerable time would be needed to deploy IPv6
- What happened to IPv5?
  - It was initially envisioned that ST-2 protocol (Stream Protocol: a family of experimental **protocols**) would become IPv5, but ST-2 was later dropped



# IPv6

## Motivations for introducing IPv6:

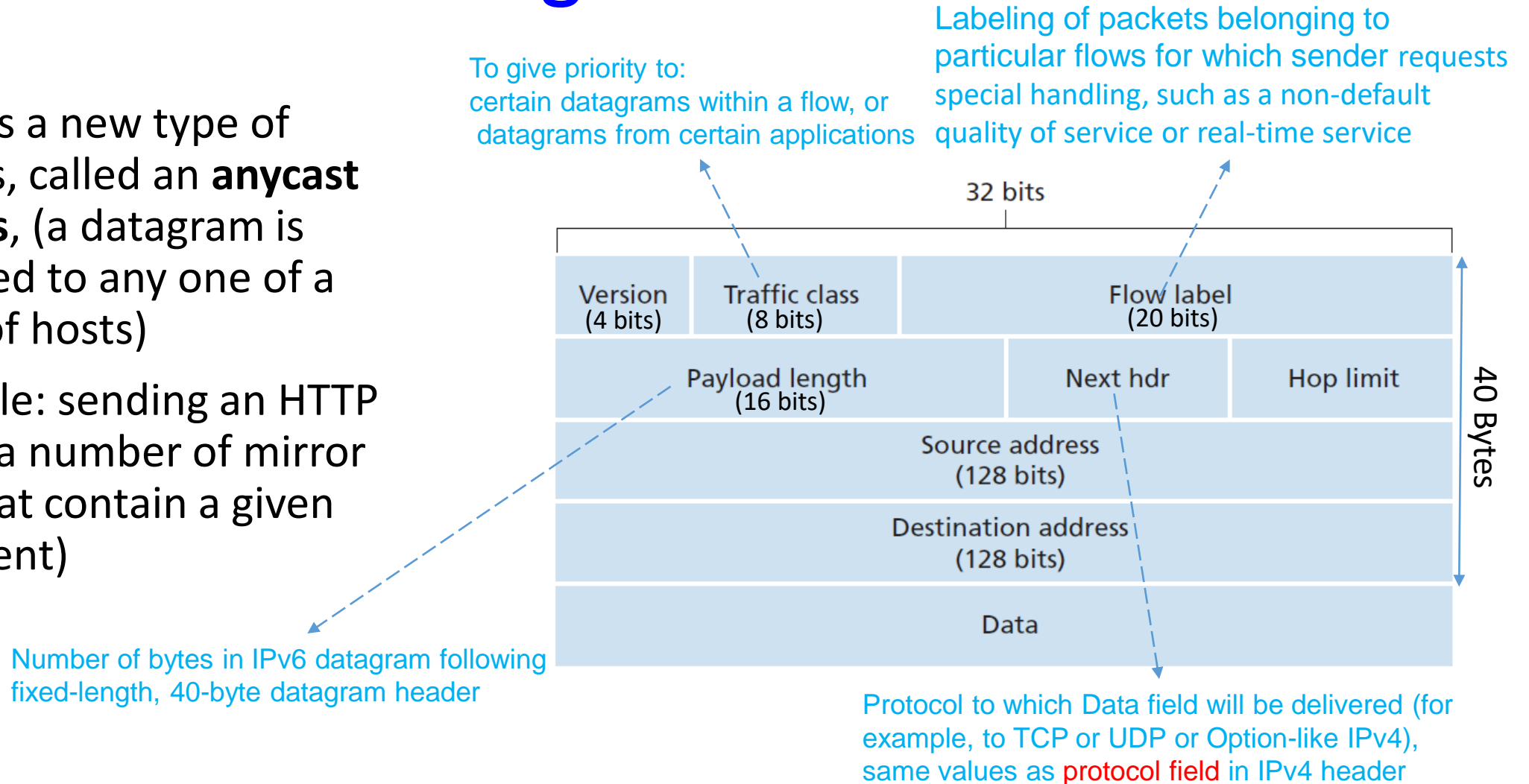
2. Speeding up processing/forwarding at routers: 40-byte fixed length header, no checksum
3. Enable different network-layer treatment of “flows”
4. No fragmentation/reassembly, No header option, No checksum

**ICMPv6:** Version ICMP (Internet Control Message Protocol) for working with IPv6

- additional message types, e.g. “Packet Too Big”
- multicast group management functions

# Figure 4.26 IPv6 datagram format

- IPv6 has a new type of address, called an **anycast address**, (a datagram is delivered to any one of a group of hosts)
- (Example: sending an HTTP GET to a number of mirror sites that contain a given document)



# Transitioning from IPv4 to IPv6

- **Tunneling:** packet within a packet [RFC 4213]
- IPv4 datagram contains an IPv6 datagram: protocol number field in IPv4 header is 41
- **1999:** Implementation of first IPv6 tunneling

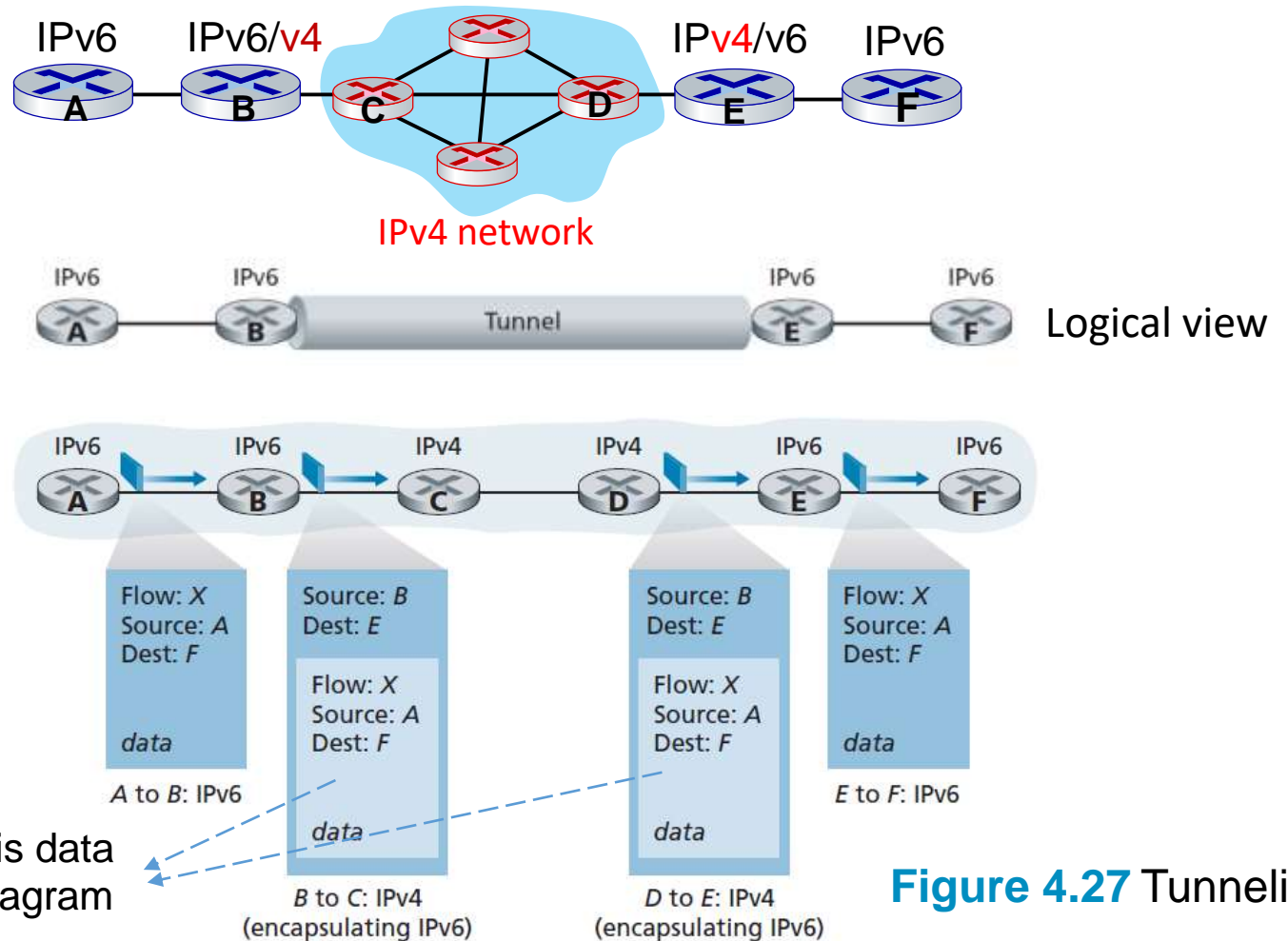


Figure 4.27 Tunneling

# Lesson learned from IPv4/v6 transition

- It is enormously difficult to change network-layer protocols
- Since early 1990s, numerous new network-layer protocols have been trumpeted as next major revolution for Internet, but most of these protocols have had limited penetration to date
- Introducing new protocols into network layer is like replacing foundation of a house (it is difficult to do without tearing whole house down or at least temporarily relocating house's residents)
- Google report: about 30.35–34.15% of clients accessing Google services do so via IPv6 (January 2021)
- By 2011, all major operating systems in use on personal computers and server systems had production-quality IPv6 implementations

# Contents

4.1 - Overview of Network Layer

4.2 - What's Inside a Router?

4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

**4.4 - Generalized Forwarding and SDN**

4.5 – Middleboxes

4.6 - Summary

## 4.4 Generalized Forwarding and SDN

- Physical separation of **network control plane** from **data (forwarding) plane**,
- **A control plane software** controls **several devices** (packet switches)
- **Open Networking Foundation** (ONF) started SDN (2011)

- 2012: Openflow



- 2014: ONOS (Open Network Operation System)



- 2016: CORD (Central Office Re-architected as a Datacenter) NFV (Network functions virtualization) and Cloud technologies to build agile datacenters for the **Network Edge**

- Linux Foundation (Open Daylight Foundation)



- OpenDaylight controller a Java Virtual Machine software

# Forwarding: “match” plus “action”

- **Destination-based forwarding:**

1. “match”: looking up a **destination IP address**
2. “action”: sending packet into switching fabric to **specified output port**

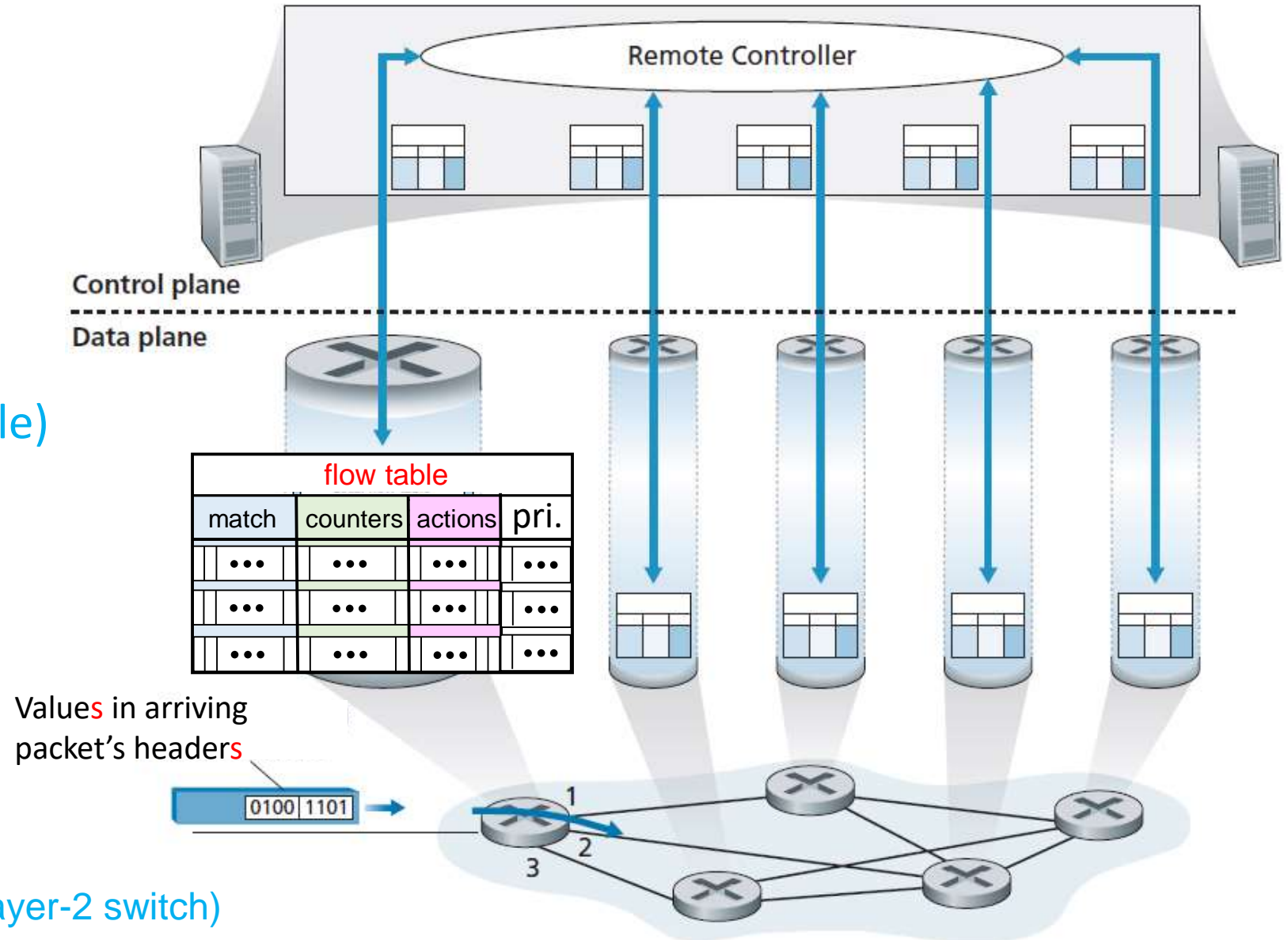
- **Generalized Forwarding:**

1. “match”: looking up **multiple header fields at different layers**
2. “action”:
  - a. Sending packet into switching fabric to one or more **specified output ports**
  - b. Rewriting header values (as in **NAT**), purposefully blocking/dropping a packet (as in a **firewall**)
  - c. **Load balancing** packets across **multiple outgoing interfaces** that lead to a service (as in load balancing)
  - d. Sending a packet to a special server for further processing and action (as in Deep Packet Inspection)

# Figure 4.28

## Generalized forwarding:

- Each packet switch contains a flow table (match-plus-action table) that is computed and distributed by **Remote Controller**



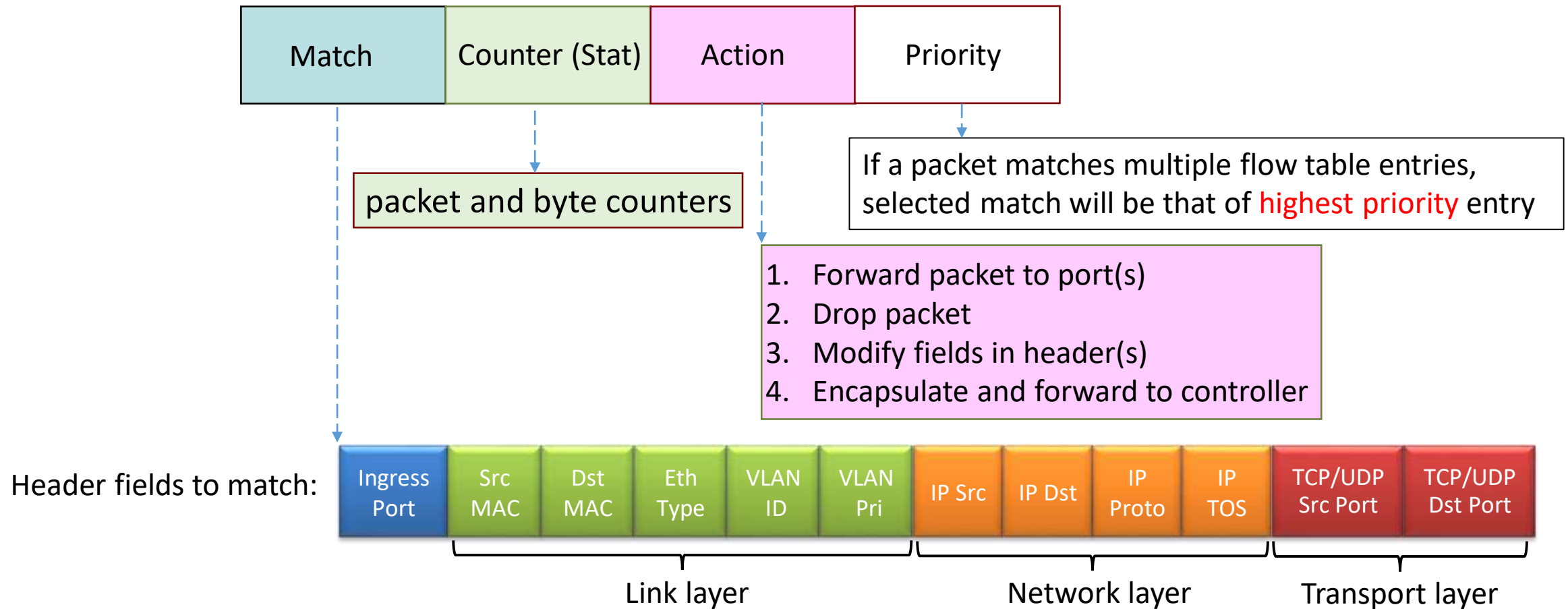
Packet switch:



( It is not layer-3 switch or layer-2 switch)



# OpenFlow 1.0: Flow table entries



**Figure 4.29** Packet matching fields, OpenFlow 1.0 flow table

## 4.4.1 Match - OpenFlow

- Figure 4.29 shows **11 packet-header fields** and **incoming port ID** that can be matched in an OpenFlow 1.0 match-plus-action rule
- OpenFlow-enabled device (packet switch) can perform as a **router** as well as a **LAN and VLAN switch** and other **middle boxes**
- **12 values** in OpenFlow 1.0 table has grown to **41 values** recent version
- Flow table entries may also have **wildcards**
  - Example: IP address of 128.119.\*.\* will match any datagram that has 128.119 as first 16 bits of its address

## 4.4.2 Action

- Each flow table entry has a list of **zero or more actions** that will be applied to a packet that matches a flow table entry
  - **If multiple actions**: they are **performed in order specified in list**

Among most important possible actions are:

- **Forwarding**: Forwarding a packet to a **particular output port, broadcast over all ports** (except port on which it arrived) or **multicast** over a selected set of ports
- Packet may be encapsulated and **sent to remote controller**, controller then may (or may not) take some action, **including installing new flow table entries**, return packet to device for forwarding under updated set of flow table rules, ...
- **Dropping**: Flow table entry with **no action** indicates a matched packet should be dropped
- **Modify-field**: Values in packet-header fields (all layer 2, 3, 4 fields, **except IP Protocol field**) may be re-written before packet is forwarded to chosen output port

# Entries examples

## Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-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

## Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
-------------	---------	---------	----------	---------	----------	--------	--------	---------	--------	------------	------------	--------

\* \* \* \* \* \* \* \* \* \* \* 22 drop

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

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
-------------	---------	---------	----------	---------	----------	--------	--------	---------	--------	------------	------------	--------

\* \* \* \* \* \* 128.119.1.1 \* \* \* \* drop

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

# Entries examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

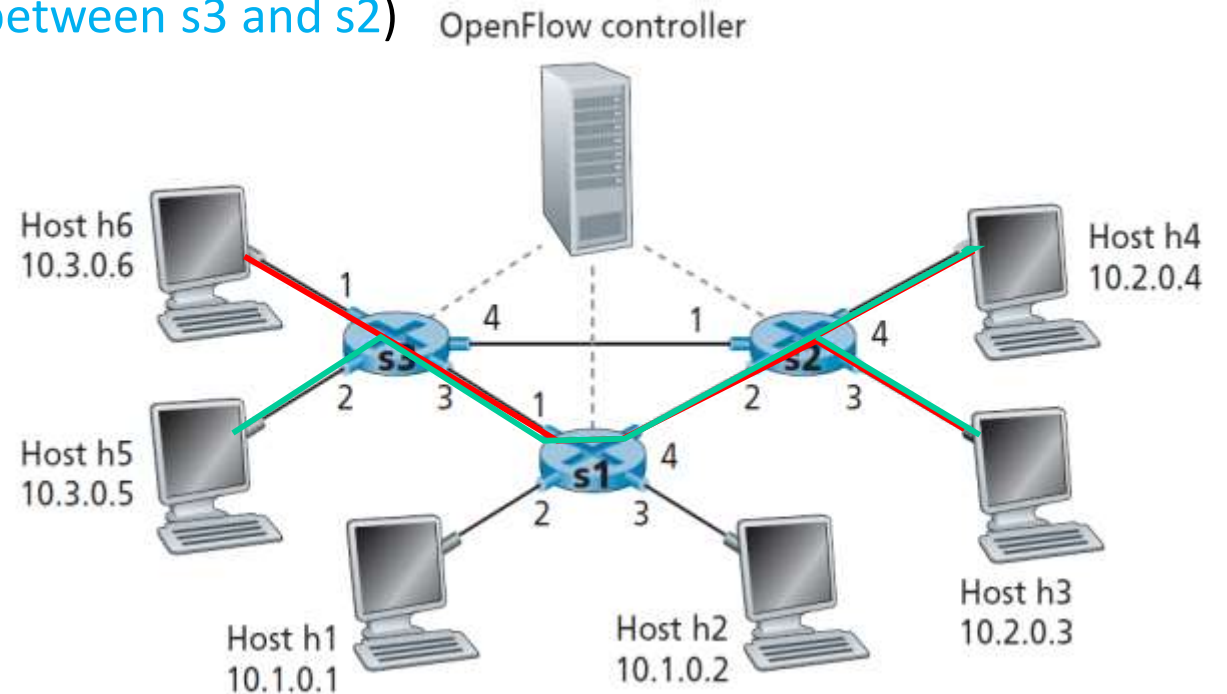
# Entries examples

Operation Mode	Switch Port	MAC src	MAC dst	Ether type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S_port	TCP D_port	Action	Counter
L2 Switching	*	*	00:1f..	*	*	*	*	*	*	*	Port1	243
Flow Switching	Port3	00:20..	00:2f..	0800	vlan1	1.2.3.4	1.2.3.9	4	4666	80	Port7	123
L3 Switching	*	*	*	*	*	*	1.2.3.4	*	*	*	Port6	452
VLAN Switching	*	*	00:3f..	*	vlan2	*	*	*	*	*	Port6 Port7 Port8	2341
Firewall	*	*	*	*	*	*	*	*	*	22	Drop	544
Default Route	*	*	*	*	*	*	*	*	*	*	Port1	1364

## 4.4.3 OpenFlow Examples of Match-plus-action in Action- Simple Forwarding

- Datagrams from hosts **h5** and **h6** should be sent to **h3** or **h4**, via **s1** and from there to **s2** (avoiding link between **s3** and **s2**)

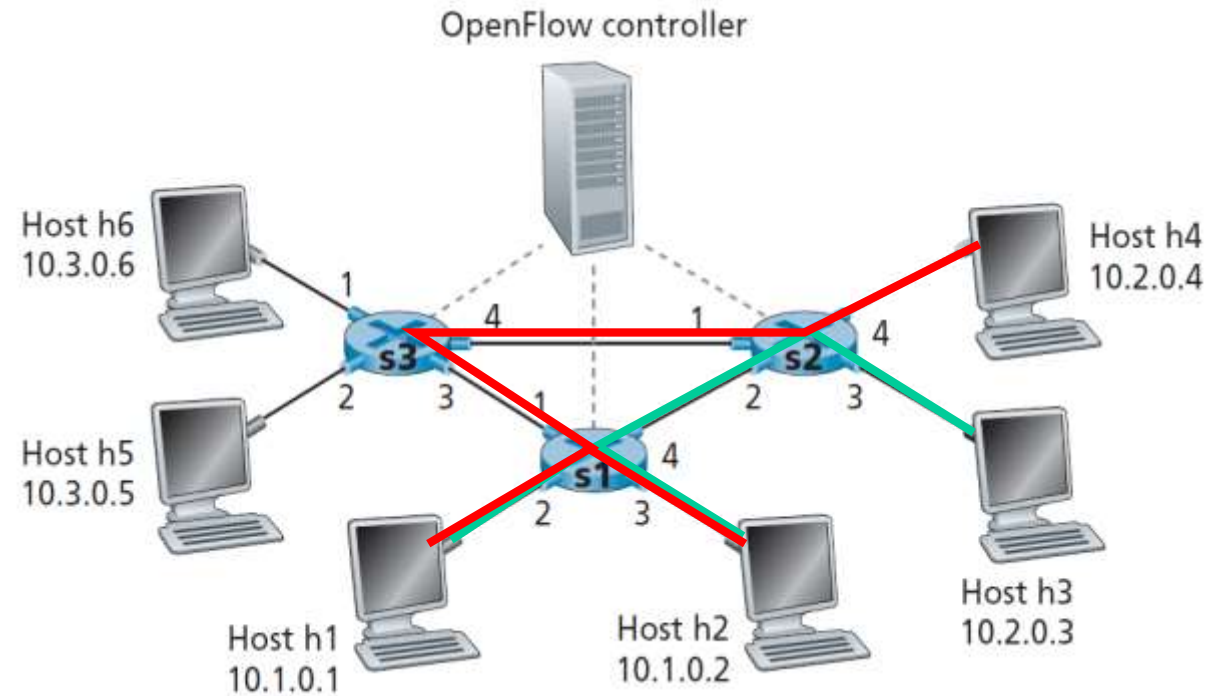
S3:	match	action
	IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)
S1:	match	action
	ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)
S2:	match	action
	ingress port = 2 IP Dst = 10.2.0.3	forward(3)
	ingress port = 2 IP Dst = 10.2.0.4	forward(4)



**Figure 4.30** OpenFlow match-plus-action network with three packet switches, 6 hosts, and an OpenFlow controller

# Examples - Load Balancing

- Datagrams from h3 destined to 10.1.\*.\* are to be forwarded over **link between s2 and s1**, while datagrams from h4 destined to 10.1.\*.\* are to be forwarded over **link between s2 and s3** (and then from s3 to s1)
- Note that this behavior couldn't be achieved with IP's destination-based forwarding



**Figure 4.30** OpenFlow match-plus-action network with three packet switches, 6 hosts, and an OpenFlow controller



# Examples - Load Balancing

s2 flow table:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
3	*	*	*	*	*	*	10.1.*.*	*	*	*	*	port2
4	*	*	*	*	*	*	10.1.*.*	*	*	*	*	port1

- Flow table entries are also needed at:
  - s3 to forward datagrams received on interface 4 from s2 over interface 3 toward s1
  - s1 to forward datagrams received from s2 to either h1 or h2

# Contents

4.1 - Overview of Network Layer

4.2 - What's Inside a Router?

4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

4.4 - Generalized Forwarding and SDN

**4.5 – Middleboxes**

4.6 - Summary



## 4.5 Middleboxes

- RFC 3234 **middleboxes** defines as:

“any intermediary equipment (“box”) performing functions apart from, standard functions of an IP router on data path between source and destination hosts”

- Beside routers, there are other network boxes within network that **sit on data path** and perform functions other than forwarding
- Cases: **Web caches** in Section 2.2.5; **TCP connection splitters** in section 3.7 (Chapter 3 slides-Appendix); and **Network Address Translation, Firewalls**, and **Intrusion Detection Systems** in Section 4.3.4, **LAN and VLAN switches** in Chapter 6
- A **modern router** easily perform **firewalling, load balancing**, ... with generalized “match plus action” operations

# Three types of services of middlebox

- Three types of services performed by middleboxes:
- **NAT Translation**
- **Security Services:**
  - **Firewalls** **block** traffic based on header-field values or **redirect** packets for additional processing, such as deep packet inspection (DPI)
  - **Intrusion Detection Systems (IDS)** are able to detect **predetermined patterns** and **filter packets accordingly**
  - Application-level **e-mail filters** block e-mails considered to be **junk**, phishing or otherwise posing a security threat
- **Performance Enhancement:** Services such as **compression**, content **caching**, and **load balancing** of service provided by server APPs

# Many other middleboxes

- It is not surprising that researchers are exploring use of commodity hardware (networking, computing, and storage) with specialized software built on top of a common software stack (like approach taken in SDN) to implement these services
- This approach has become known as **network function virtualization (NFV)**
- An alternate approach that has also been explored is to **outsource middlebox functionality to cloud**

# A middlebox operates on more than one layer

- **NAT**: rewrites **network-layer** IP addresses and **transport-layer** port numbers
- **Firewall**: blocks suspect datagrams using **application-layer**, **transport-layer**, and **network-layer** header fields
- **E-mail security gateways**: filters application-layer e-mail messages based on **whitelisted/blacklisted IP** addresses as well as e-mail message **content**

# Contents

4.1 - Overview of Network Layer

4.2 - What's Inside a Router?

4.3 - The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

4.4 - Generalized Forwarding and SDN

4.5 – Middleboxes

**4.6 - Summary**

## 4.6 Summary

- **Data plane functions** of network layer (**per-router** functions, determine how packets arriving on one of a router's input links are forwarded to one of router's output links)
- Internal operations of a router
  - input and output port functionality
  - destination based forwarding
  - router's internal switching mechanism
  - packet queue management and ...
- Traditional IP forwarding (based on destination address)
- Generalized forwarding (using values in several different fields in datagram's header)
- IPv4 and IPv6 protocols
- Internet addressing
- Middleboxes
- We study network layer's control plane in Chapter 5



# Appendix

# NET NEUTRALITY

- What is Net Neutrality?
- Technical: how an ISP should share/allocation its resources
  - packet scheduling, buffer management are *mechanisms*
- Social, economic principles:
  - protecting easy flow of information
  - encouraging innovation, competition
  - enforced legal rules and policies
- Different countries have different “takes” on network neutrality

# NET NEUTRALITY

2015 US FCC [Order on Protecting and Promoting an Open Internet](#) (3 rules):

- **no blocking**: shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management
- **no throttling**: shall not impair or degrade lawful Internet traffic on basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management
- **no paid prioritization**: shall not engage in paid prioritization
- 2015 FCC [OPPO Internet](#), was superseded by 2017 FCC [Restoring Internet Freedom Order](#), which rolled back these 3 prohibitions and focused instead on [ISP transparency](#)
- We aren't close to having seen final chapter written on net neutrality in US, or elsewhere

# INSPECTING DATAGRAMS: FIREWALLS AND INTRUSION DETECTION SYSTEMS

- Attackers, knowing IP address range of your network, can easily send IP datagrams to addresses in your range
- Attacker can do:
  - mapping your network with ping sweeps
  - port scans
  - crashing vulnerable hosts with malformed packets
  - scanning for open TCP/UDP ports on servers in your network
  - infecting hosts by including malware in packets
- Two popular defense mechanisms to malicious packet attacks:
  - firewalls
  - intrusion detection systems (IDSs)

# FIREWALL

- We install a firewall between our network and Internet
  - Most access routers today have firewall capability
- Firewalls inspect datagram and segment header fields, denying suspicious datagrams entry into internal network
  - For example, a firewall may be configured to block all **ICMP echo request packets** (Section 5.6), preventing from port scan across your IP address range
- Firewalls can block packets based on source and destination IP addresses and port numbers
- Firewalls can be configured to track TCP connections, granting entry only to datagrams that belong to approved connections

# INTRUSION DETECTION SYSTEM

- IDS situated at network boundary, performs “**deep packet inspection**,” examining not only header fields but also payloads in datagram (including application-layer data)
- IDS has a database of packet signatures that are known to be part of attacks
- This database is automatically updated as new attacks are discovered
- All packets pass through IDS, IDS attempts to match header fields and payloads to signatures in its signature database
- If such a match is found, an alert is created
- An **intrusion prevention system** (IPS) is similar to an IDS, except that it actually blocks packets in addition to creating alerts
- Can firewalls and IDSs fully shield your network from all attacks? The answer is clearly no, as attackers continually find new attacks for which signatures are not yet available. But firewalls and traditional signature-based IDSs are useful in protecting your network from known attacks