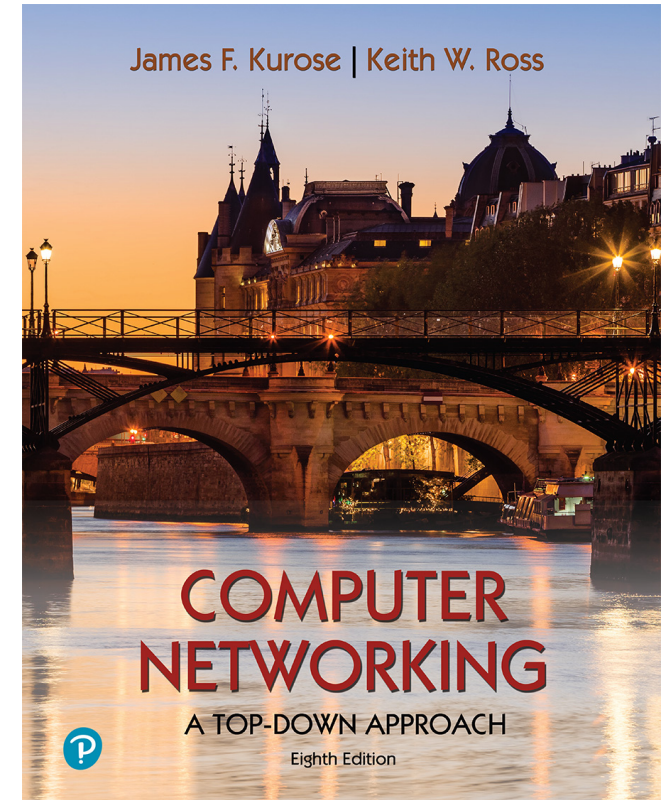


# Chapter 4

## Network Layer: Data Plane



### *Computer Networking: A Top-Down Approach*

8<sup>th</sup> edition

Jim Kurose, Keith Ross  
Pearson, 2020

Acknowledgement: Based on the textbook's website:  
[https://gaia.cs.umass.edu/kurose\\_ross/index.php](https://gaia.cs.umass.edu/kurose_ross/index.php)

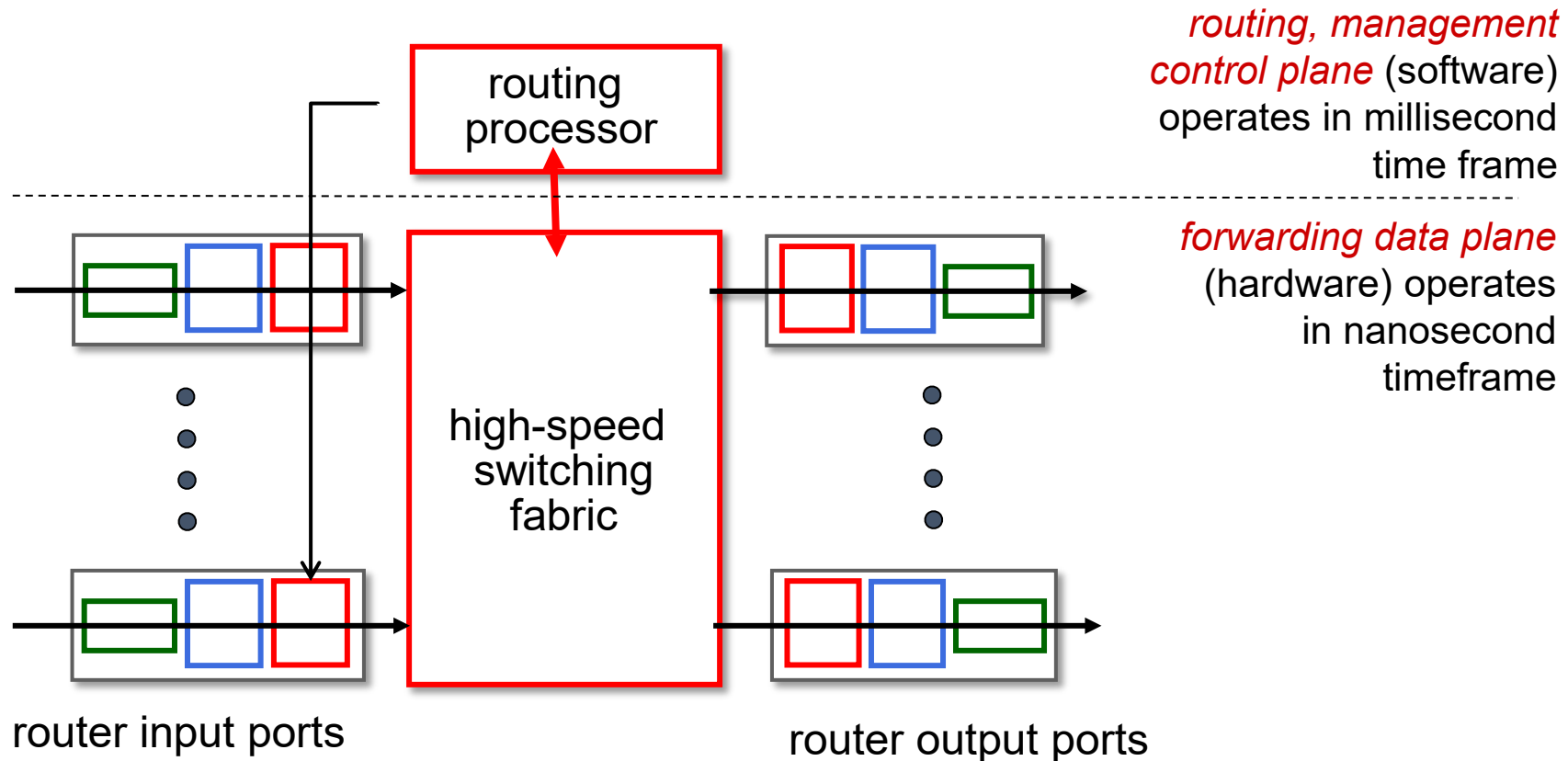
# Network layer: “data plane” roadmap

- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6
- Generalized Forwarding, SDN
  - Match+action
  - OpenFlow: match+action in action
- Middleboxes



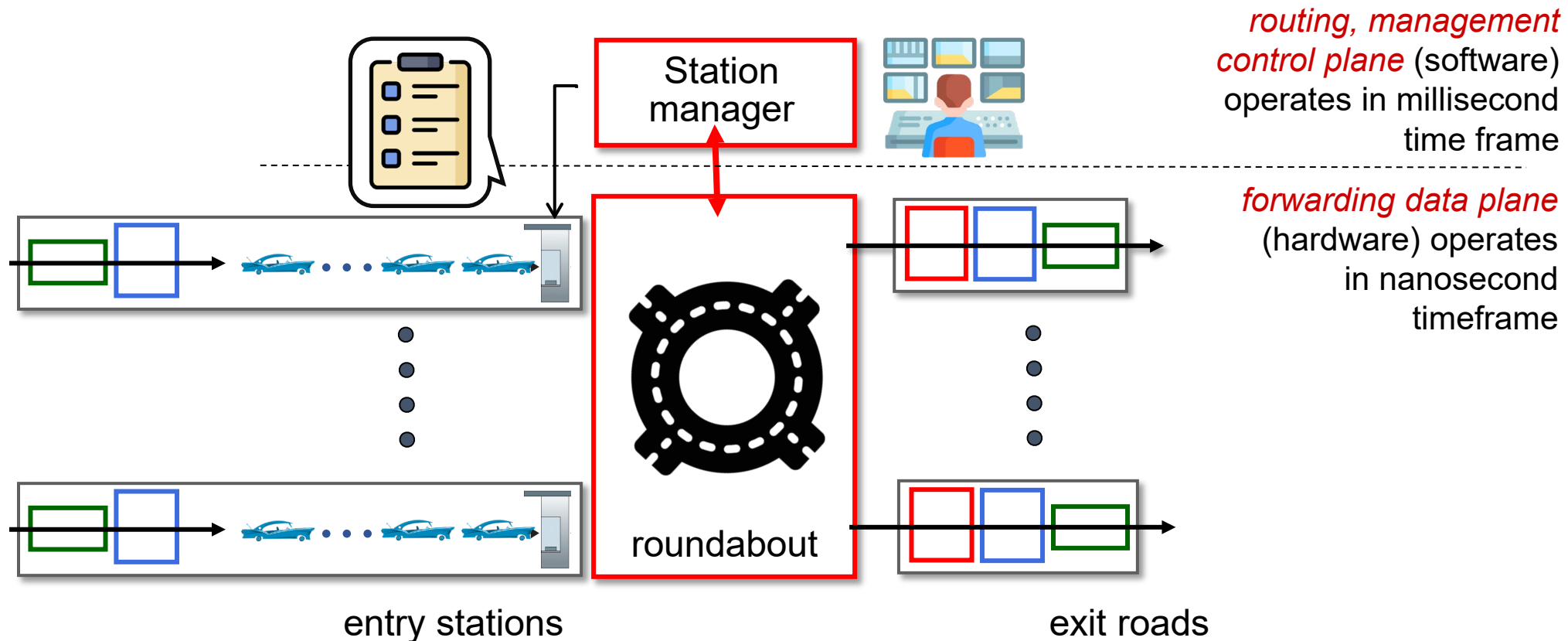
# Router architecture overview

high-level view of generic router architecture:

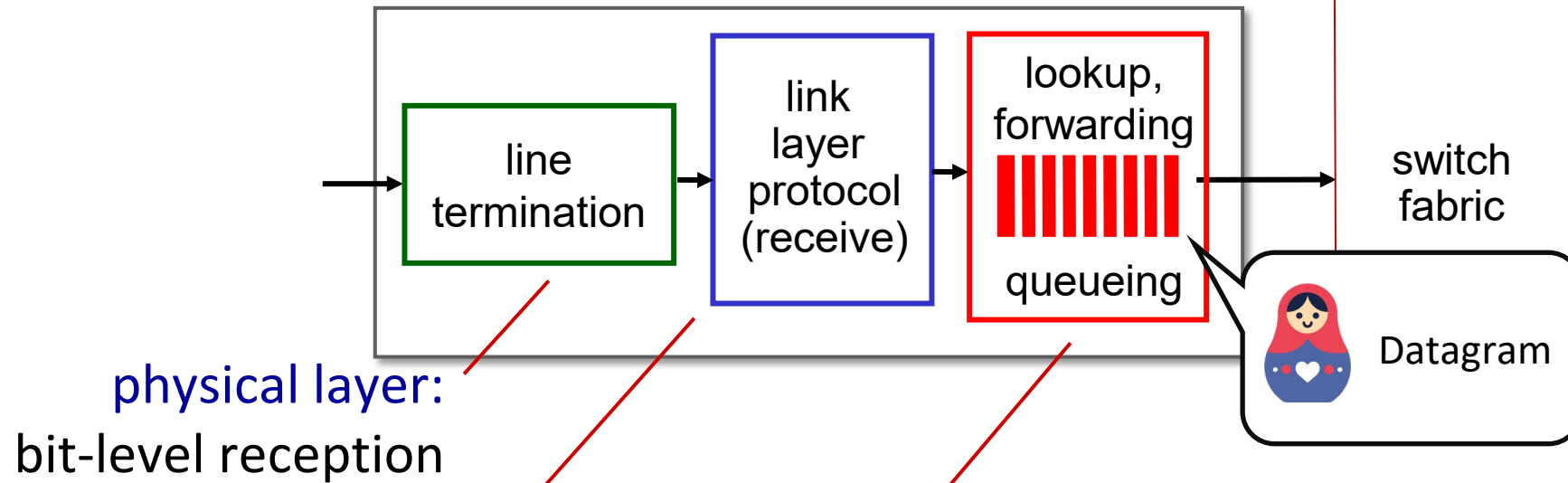


# Router architecture overview

analogy view of generic router architecture:



# Input port functions



physical layer:  
bit-level reception

link layer:  
e.g., Ethernet  
(chapter 6)

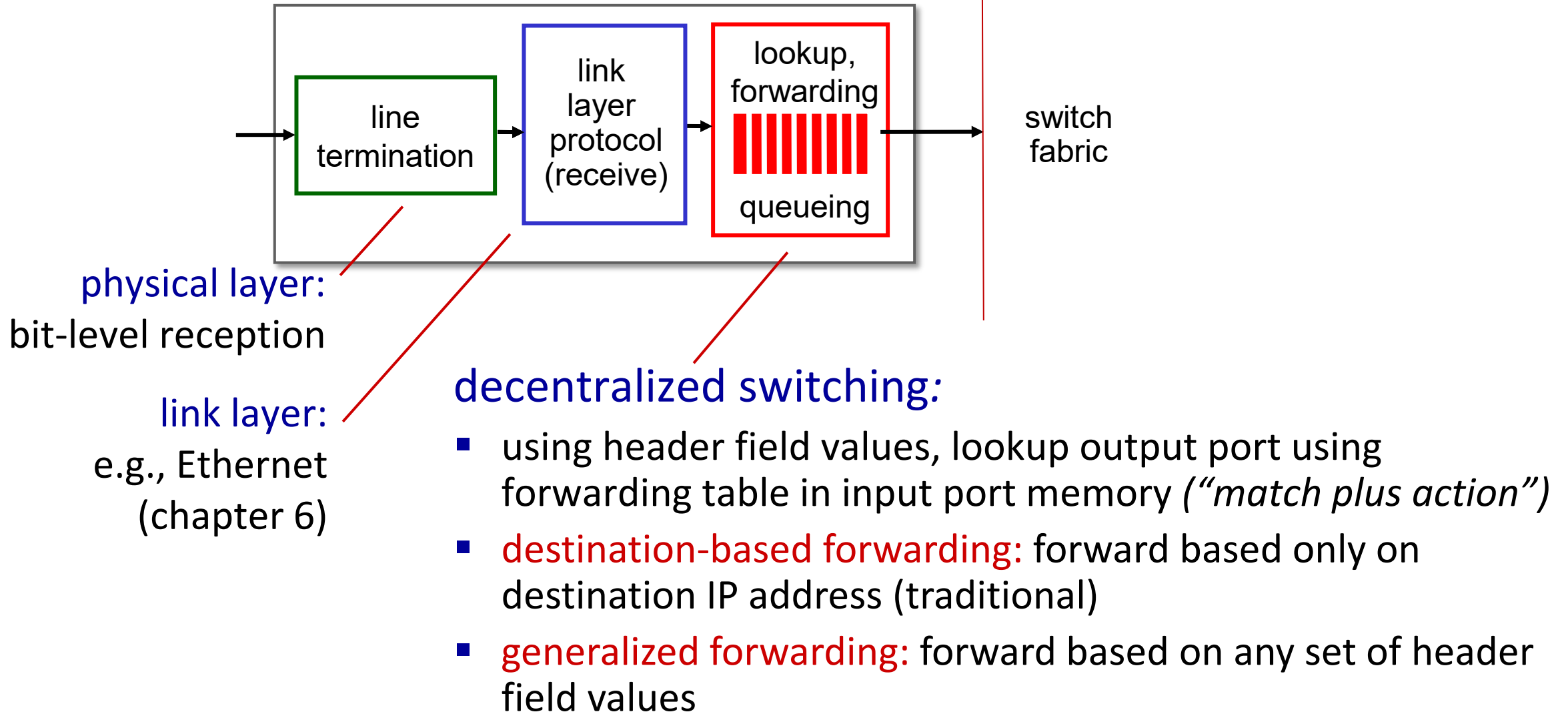


Frame

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’
- **input port queueing:** if datagrams arrive faster than forwarding rate into switch fabric

# Input port functions



# Destination-based forwarding

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	n 3
11001000 00010111 00011000 11111111 through 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 match! 1 00011*** *****	2
otherwise	3

examples:

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

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

match!

examples:

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

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

match!

examples:

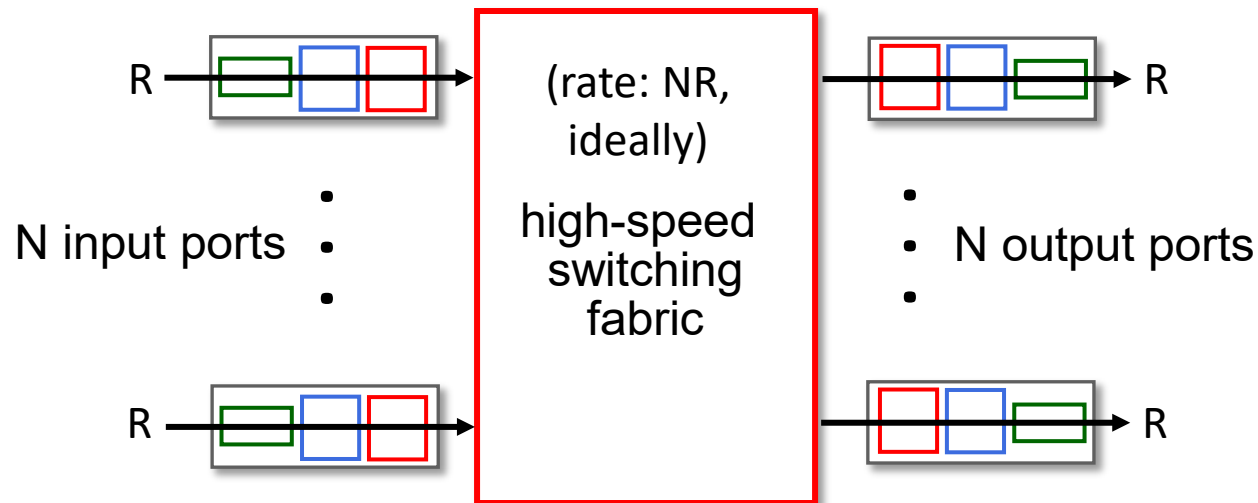
11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

# Longest prefix matching

- 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: ~1M routing table entries in TCAM

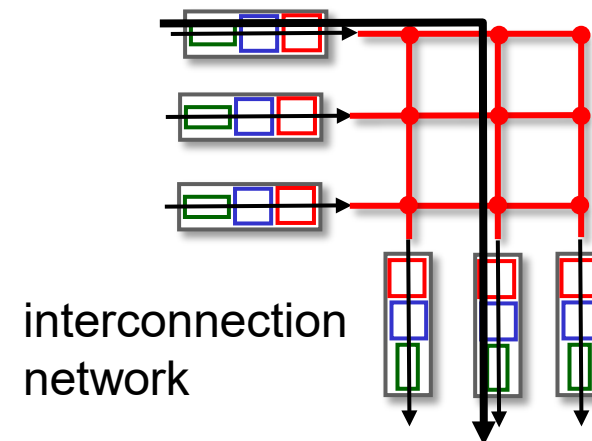
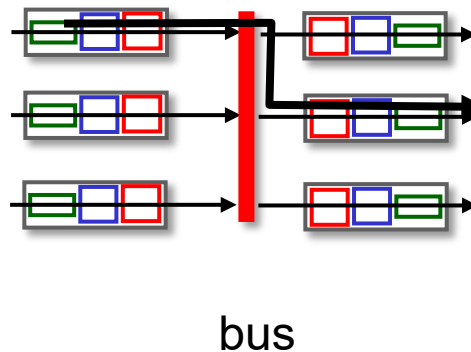
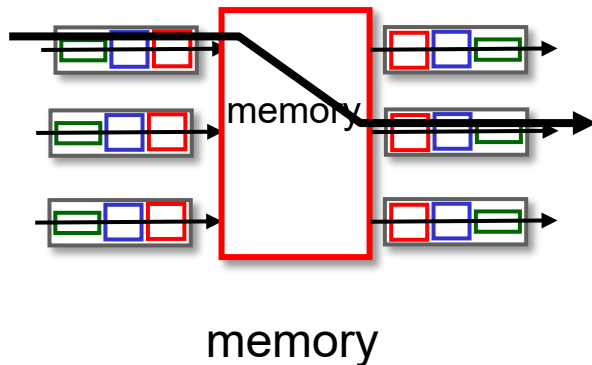
# Switching fabrics

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



# Switching fabrics

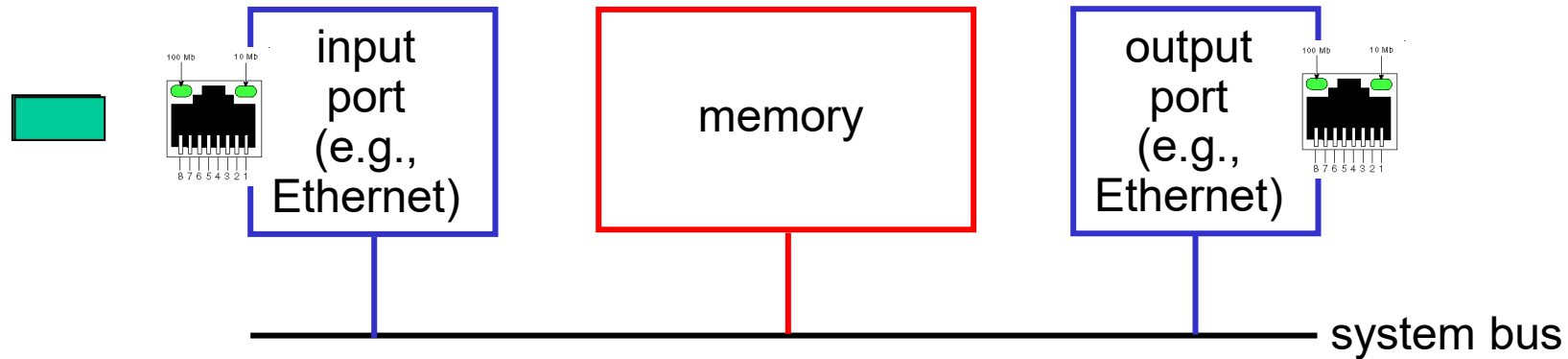
- transfer packet from input link to appropriate output link
- **switching rate**: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



# Switching via memory

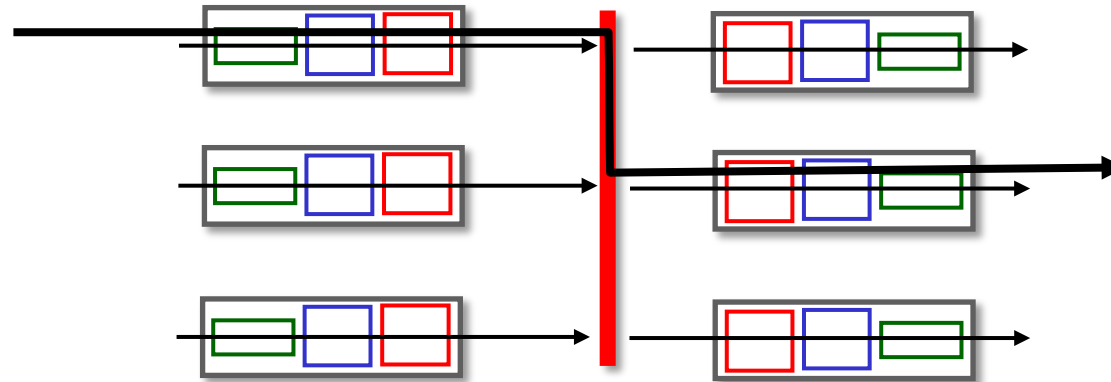
## first generation routers:

- 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 a bus

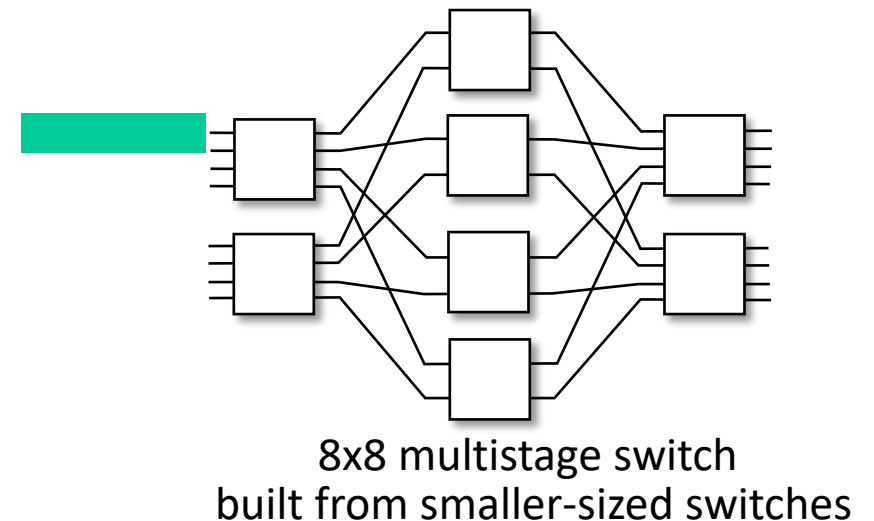
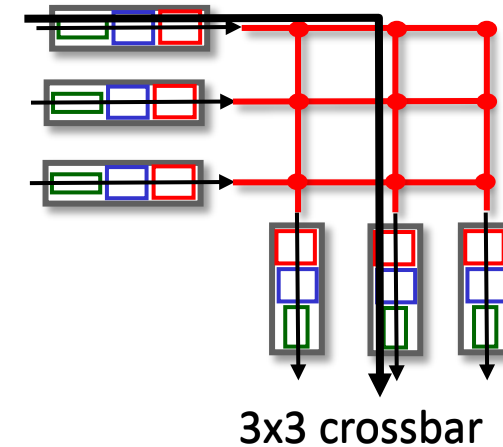
- datagram from input port memory to output port memory via a shared bus
- *bus contention*: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers





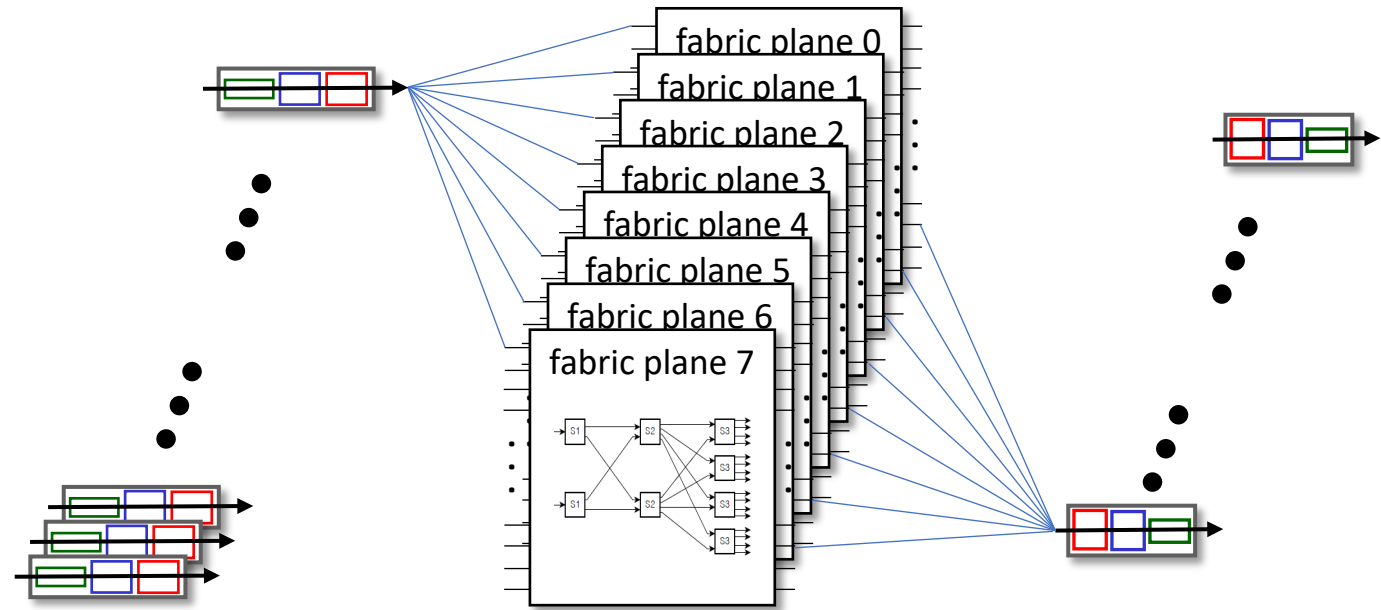
# Switching via interconnection network

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



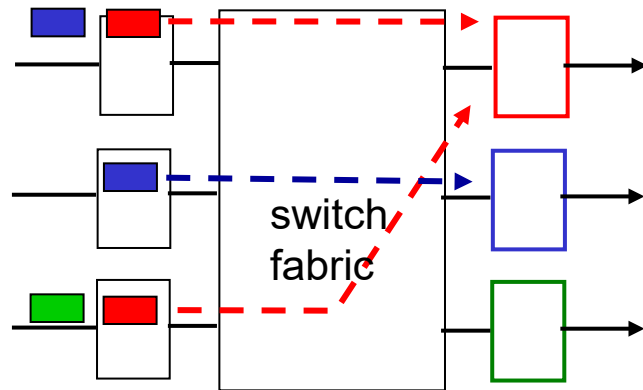
# Switching via interconnection network

- scaling, using multiple switching “planes” in parallel:
  - speedup, scaleup via parallelism
- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100's Tbps switching capacity

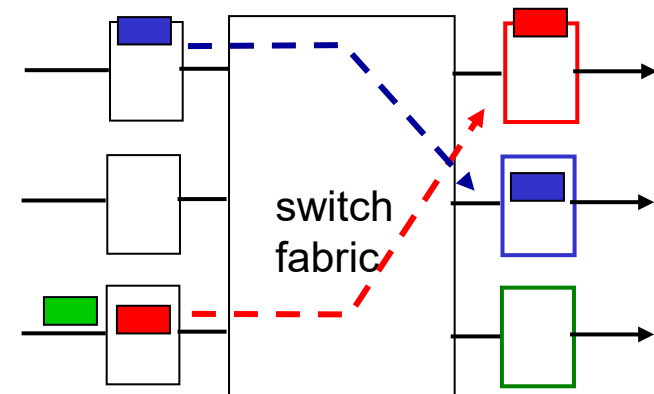


# Input port queuing

- If switch 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

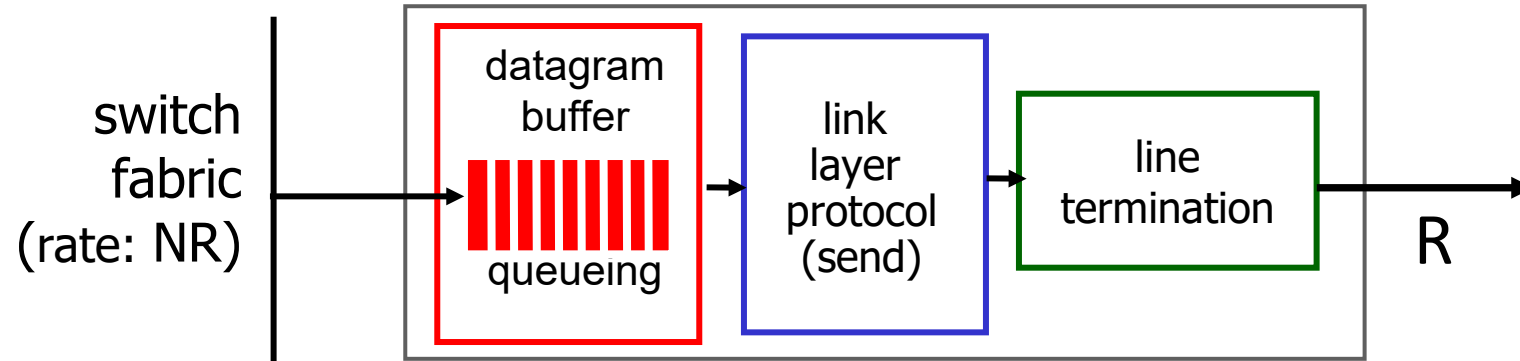


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

# Output port queuing



This is a really important slide

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



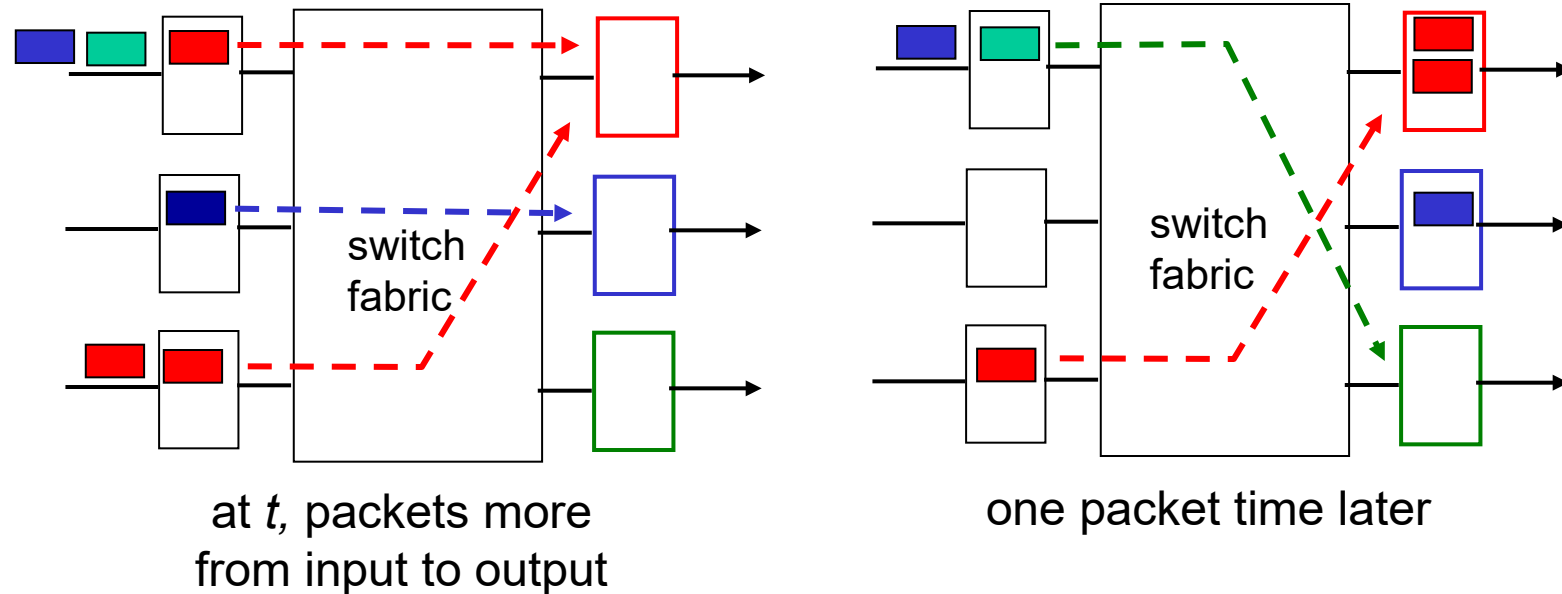
Datagrams can be lost due to congestion, lack of buffers

- **Scheduling discipline** chooses among queued datagrams for transmission



Priority scheduling – who gets best performance, network neutrality

# Output port queuing



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

# How much buffering?

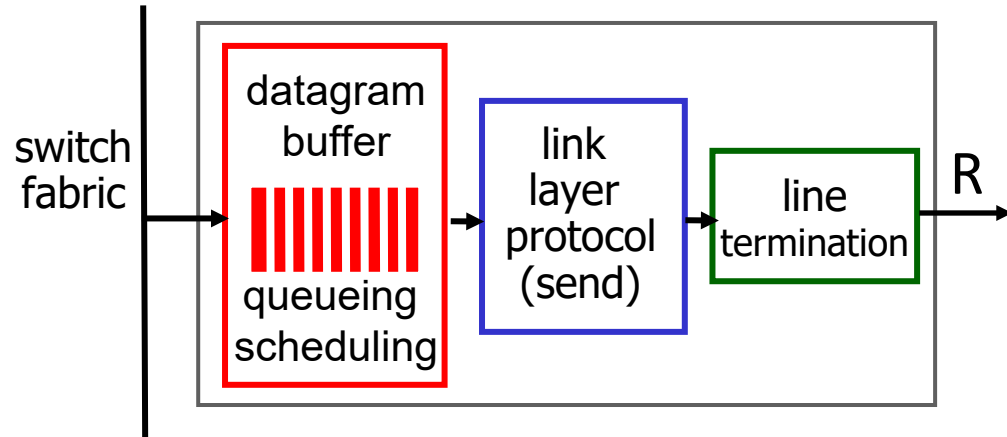
- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity  $C$ 
  - e.g.,  $C = 10$  Gbps link: 2.5 Gbit buffer

- more recent recommendation: with  $N$  flows, buffering equal to

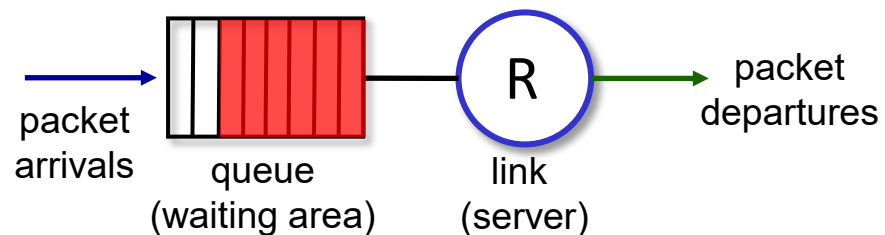
$$\frac{RTT \cdot C}{\sqrt{N}}$$

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

# Buffer Management



## Abstraction: queue



## buffer management:

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

# Packet Scheduling: FCFS

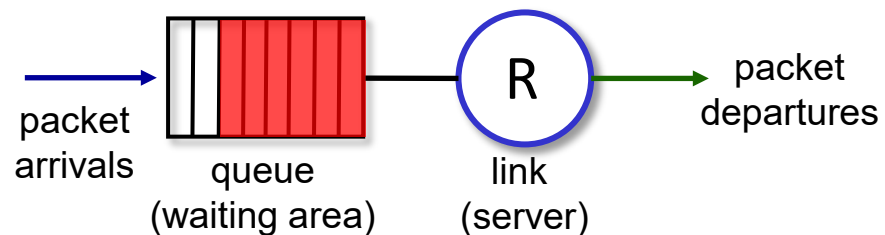
**packet scheduling:** deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

**FCFS:** packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)
- real world examples?

Abstraction: queue

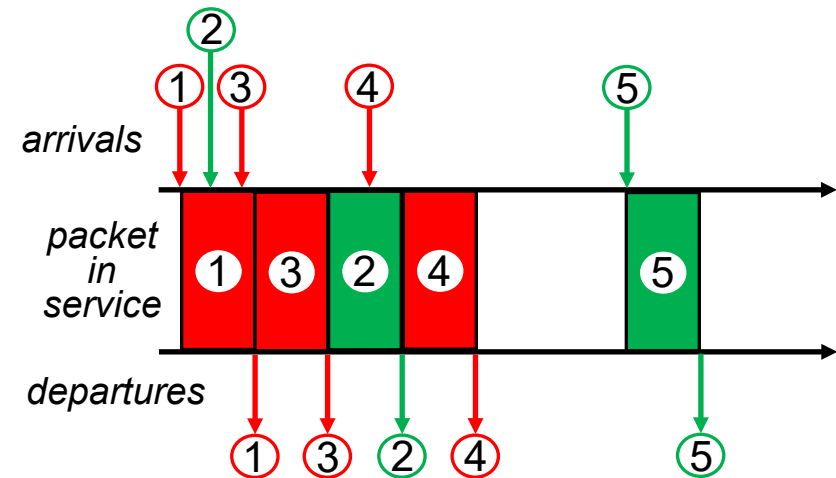
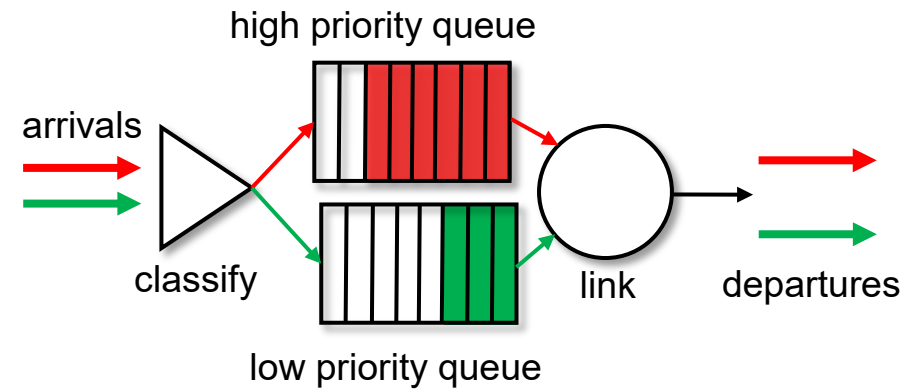




# Scheduling policies: priority

## *Priority scheduling:*

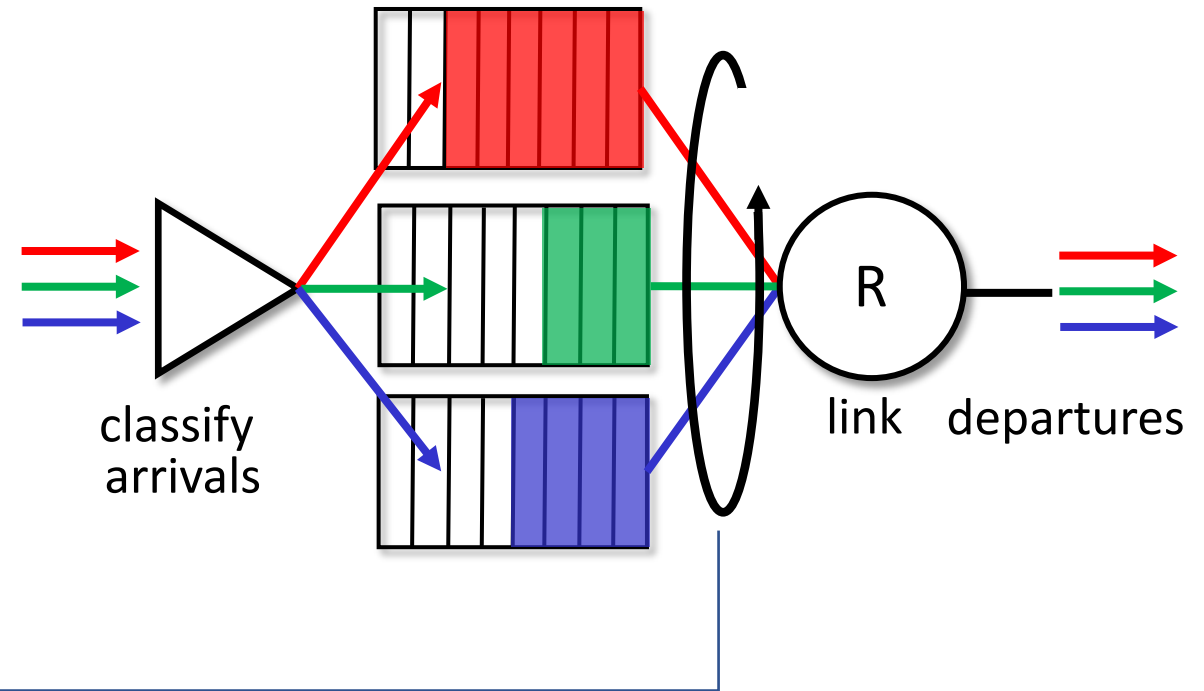
- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class



# Scheduling policies: round robin

## *Round Robin (RR) scheduling:*

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



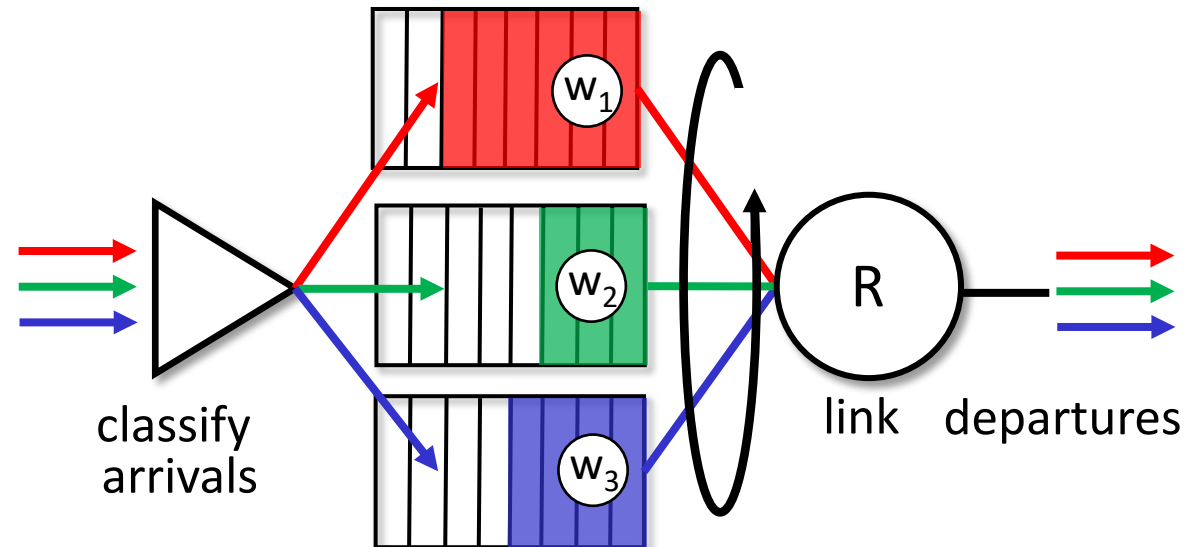
# Scheduling policies: weighted fair queueing

## *Weighted Fair Queuing (WFQ):*

- generalized Round Robin
- each class,  $i$ , has weight,  $w_i$ , and gets weighted amount of service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$

- minimum bandwidth guarantee (per-traffic-class)



# Sidebar: Network Neutrality

What is network neutrality?

- *technical*: how an ISP should share/allocation its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

*Different countries have different “takes” on network neutrality*

# Sidebar: Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet*: three “clear, bright line” 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 the 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”