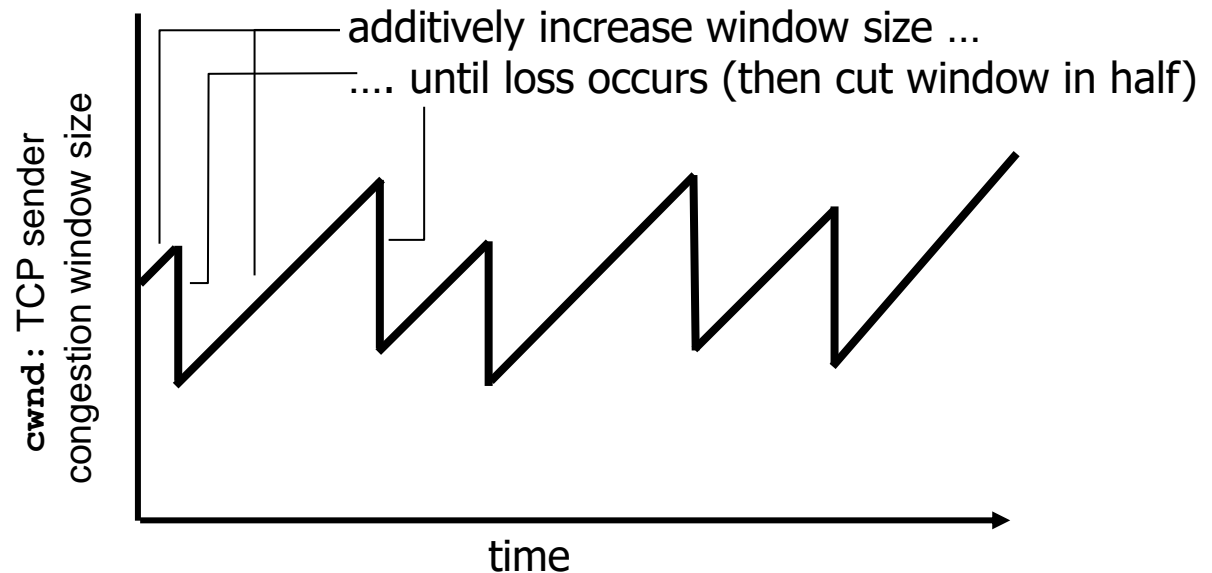# Overview of last class

TCP congestion control: additive increase multiplicative decrease

- *approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase `cwnd` by 1 MSS every RTT until loss detected
  - *multiplicative decrease:* cut `cwnd` in half after loss

AIMD saw tooth behavior: probing for bandwidth

additively increase window size …
…. until loss occurs (then cut window in half)

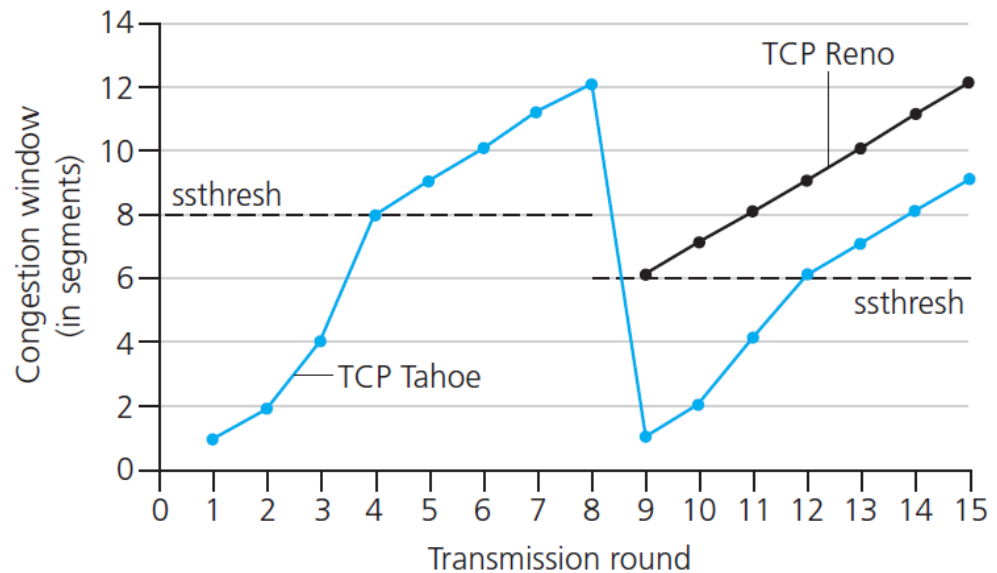cwnd: TCP sender congestion window size

time

TCP: switching from slow start to CA

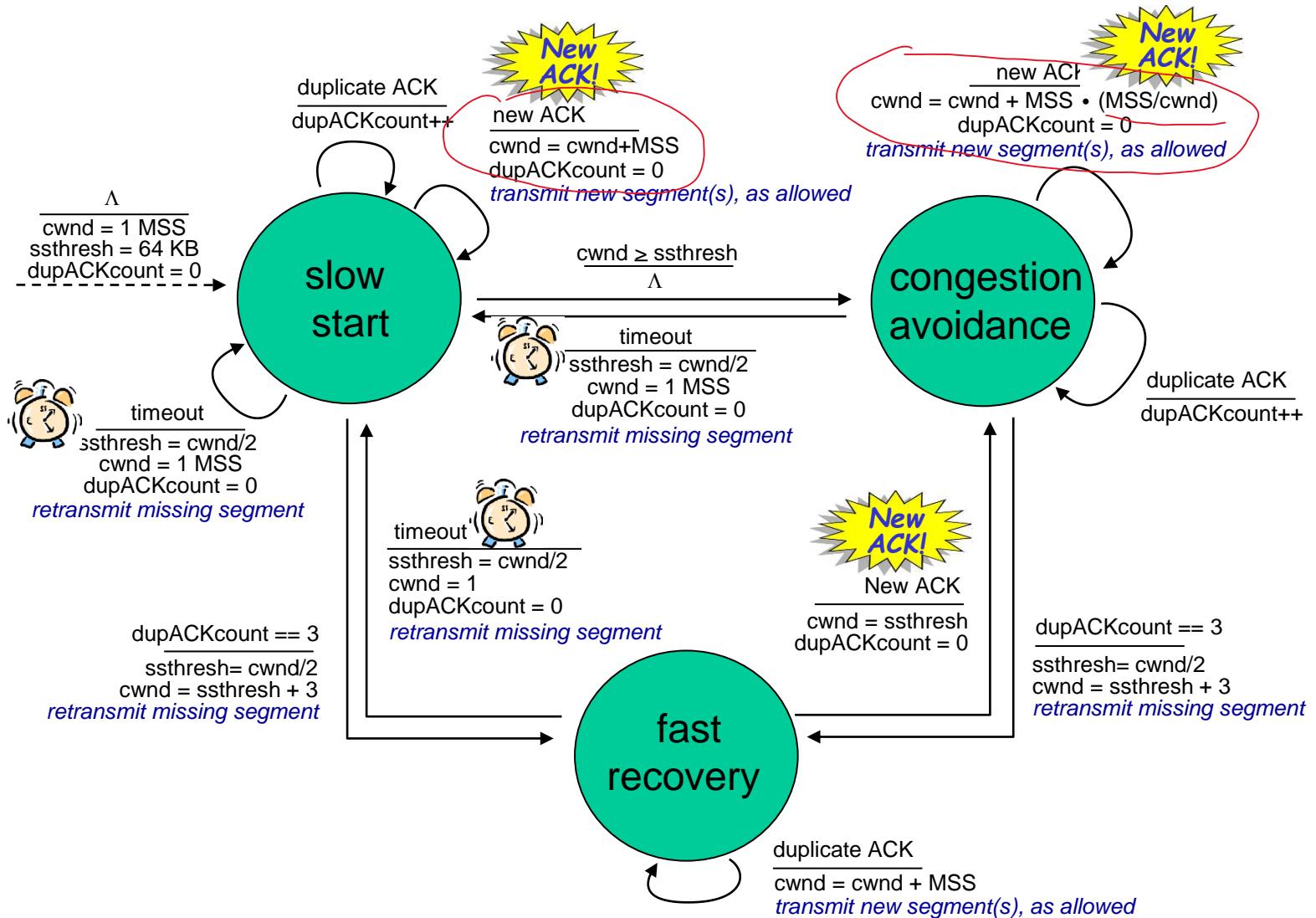Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before timeout.

Implementation:

- variable **ssthresh**
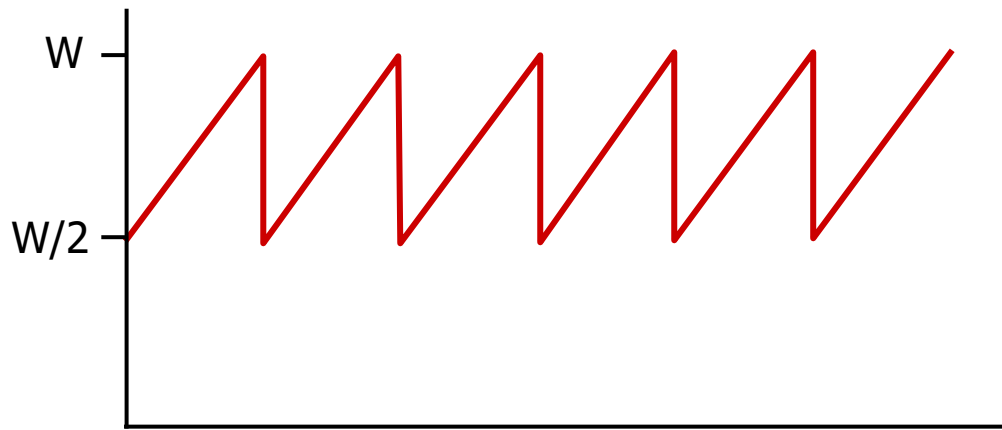- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event

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

Summary: TCP Congestion Control

duplicate ACK
_____
dupACKcount++

New ACK!

new ACK
cwnd = cwnd+MSS
dupACKcount = 0
transmit new segment(s), as allowed

New ACK!

new ACK
cwnd = cwnd + MSS · (MSS/cwnd)
dupACKcount = 0
transmit new segment(s), as allowed

Λ
_____
cwnd = 1 MSS
ssthresh = 64 KB
dupACKcount = 0

slow start

cwnd ≥ ssthresh
_____
Λ

congestion avoidance

timeout
_____
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment

duplicate ACK
_____
dupACKcount++

timeout
_____
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment

timeout
_____
ssthresh = cwnd/2
cwnd = 1
dupACKcount = 0
retransmit missing segment

New ACK!

New ACK
_____
cwnd = ssthresh
dupACKcount = 0

dupACKcount == 3
_____
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment

dupACKcount == 3
_____
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment

fast recovery

duplicate ACK
_____
cwnd = cwnd + MSS
transmit new segment(s), as allowed

TCP throughput

- **W: window size** (measured in bytes) **where loss occurs**
  - avg. window size (# in-flight bytes) is ¾ W
  - avg. thruput is 3/4W per RTT

$$\text{avg TCP thruput} = \frac{3}{4} \frac{W}{RTT} \text{ bytes/sec}$$
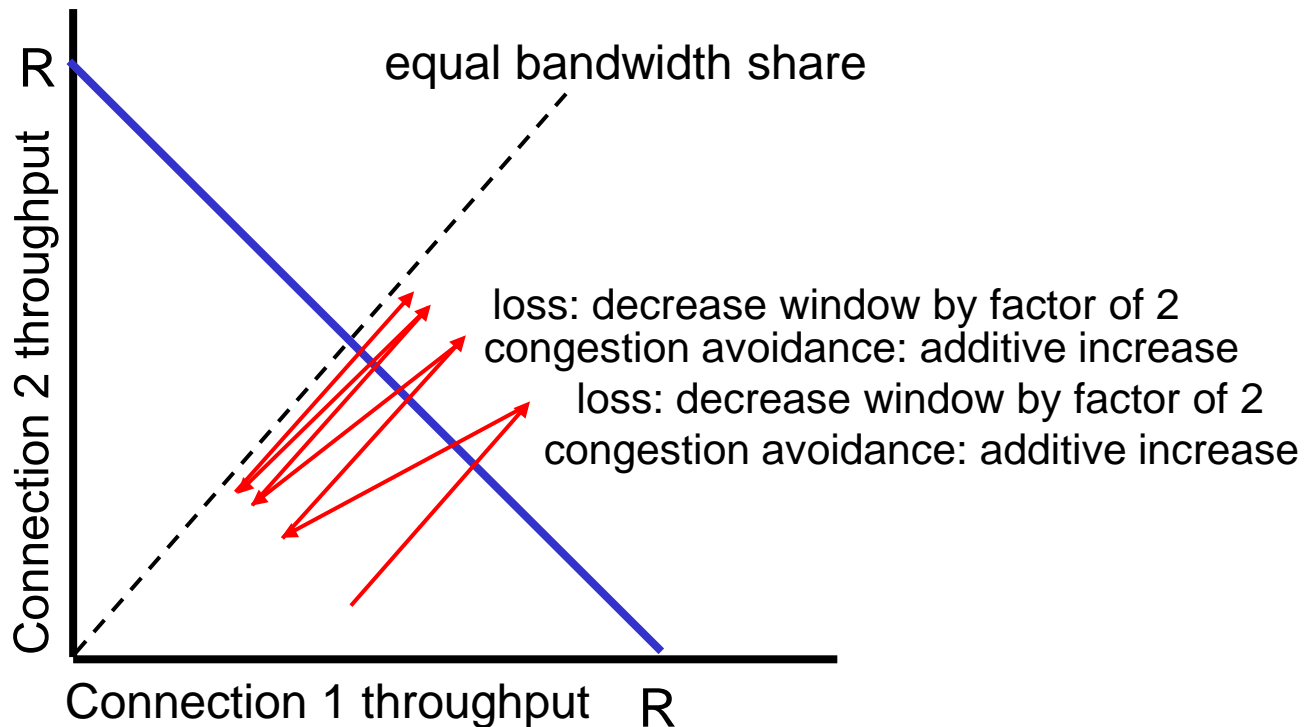

$$\text{TCP throughput} = \frac{1.22 \cdot MSS}{RTT \sqrt{L}}$$

TCP fairness

two competing sessions:
- additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally

equal bandwidth share

loss: decrease window by factor of 2
congestion avoidance: additive increase
loss: decrease window by factor of 2
congestion avoidance: additive increase

Connection 2 throughput

Connection 1 throughput

R

R

Class Today

Chapter 4: The Network Layer: Data Plane

Chapter 4: network layer

chapter goals:

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding
- instantiation, implementation in the Internet

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it
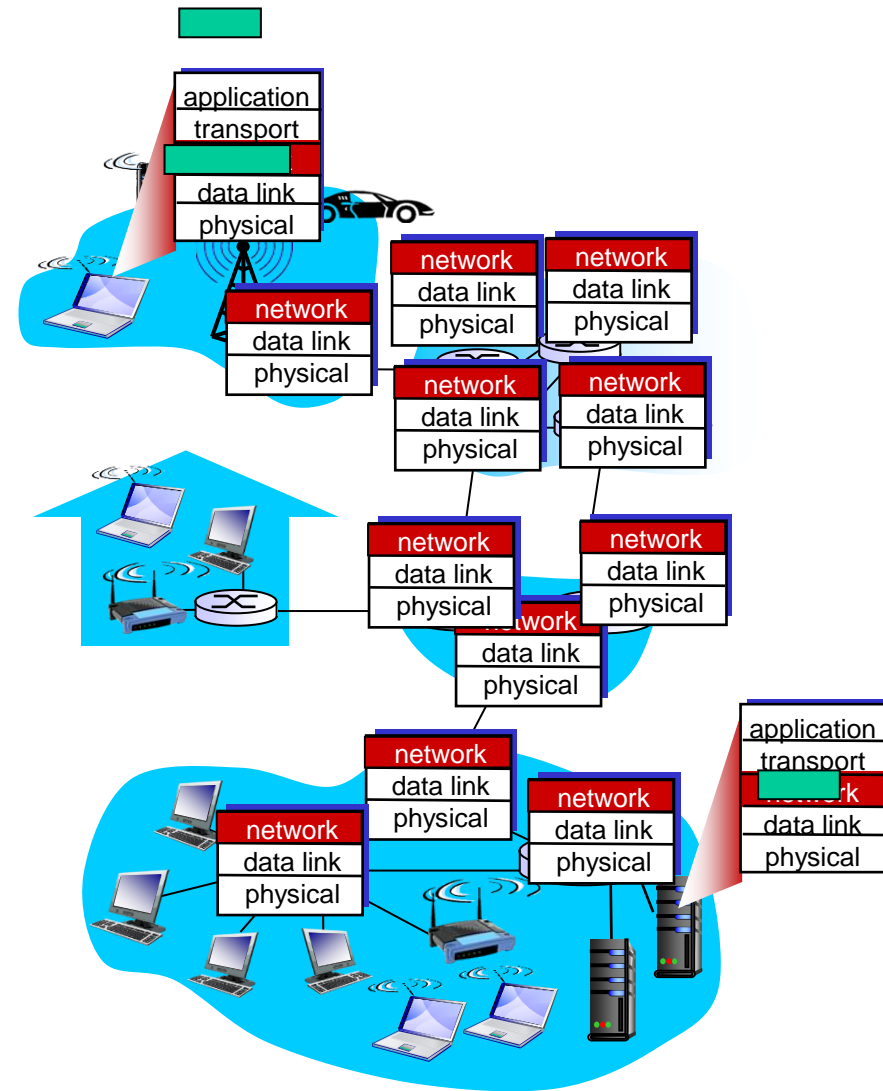
Two key network-layer functions

network-layer functions:
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip
- *forwarding:* process of getting through single interchange
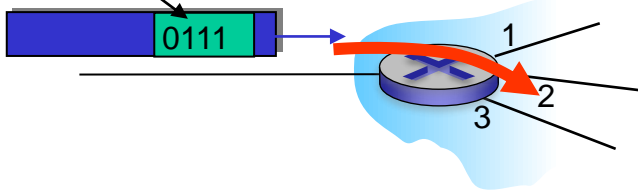- *routing:* process of planning trip from source to destination

Network layer: data plane, control plane

Forwarding

Data plane

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

values in arriving packet header

Routing

Control plane

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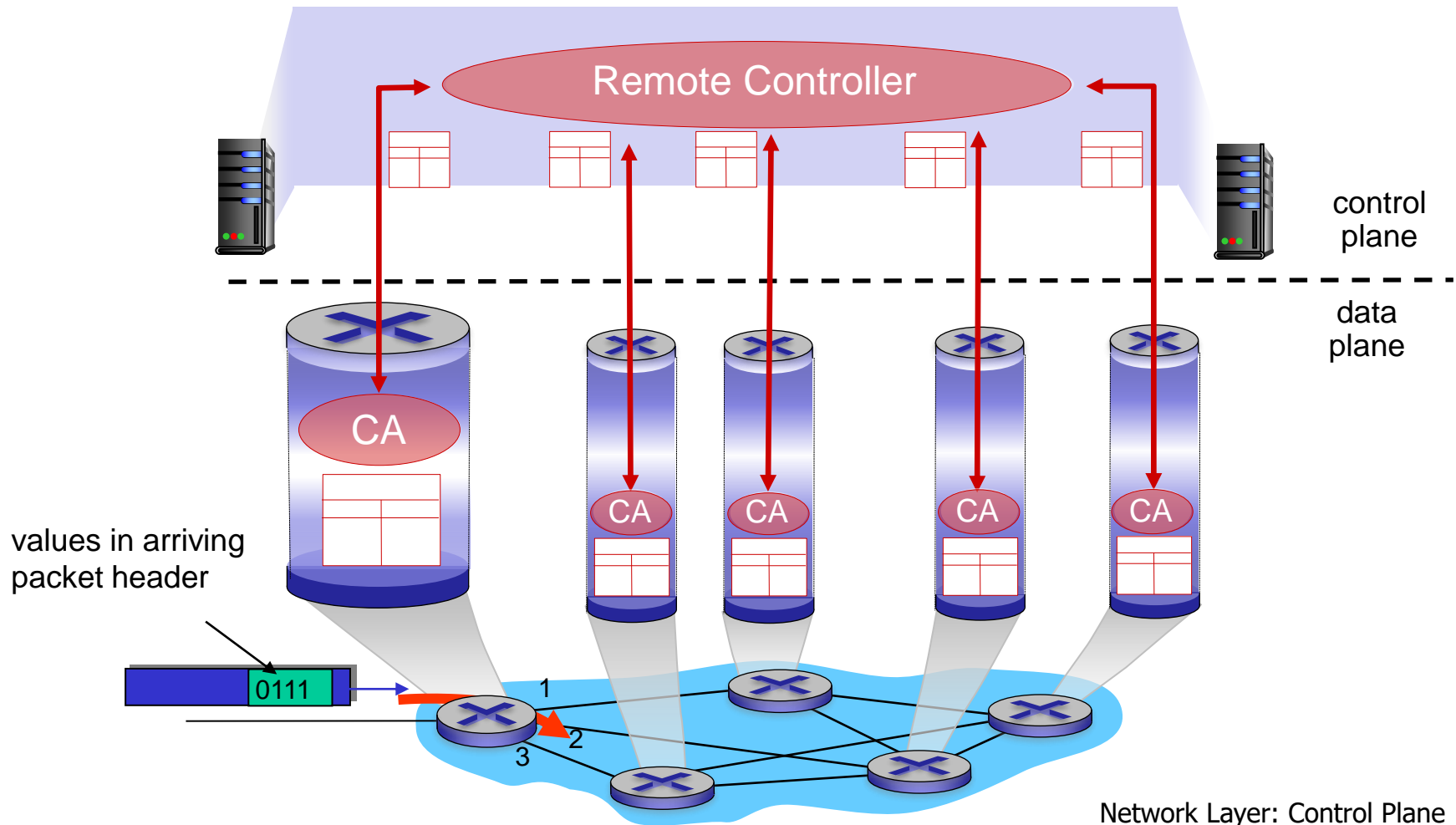
Per-router control plane

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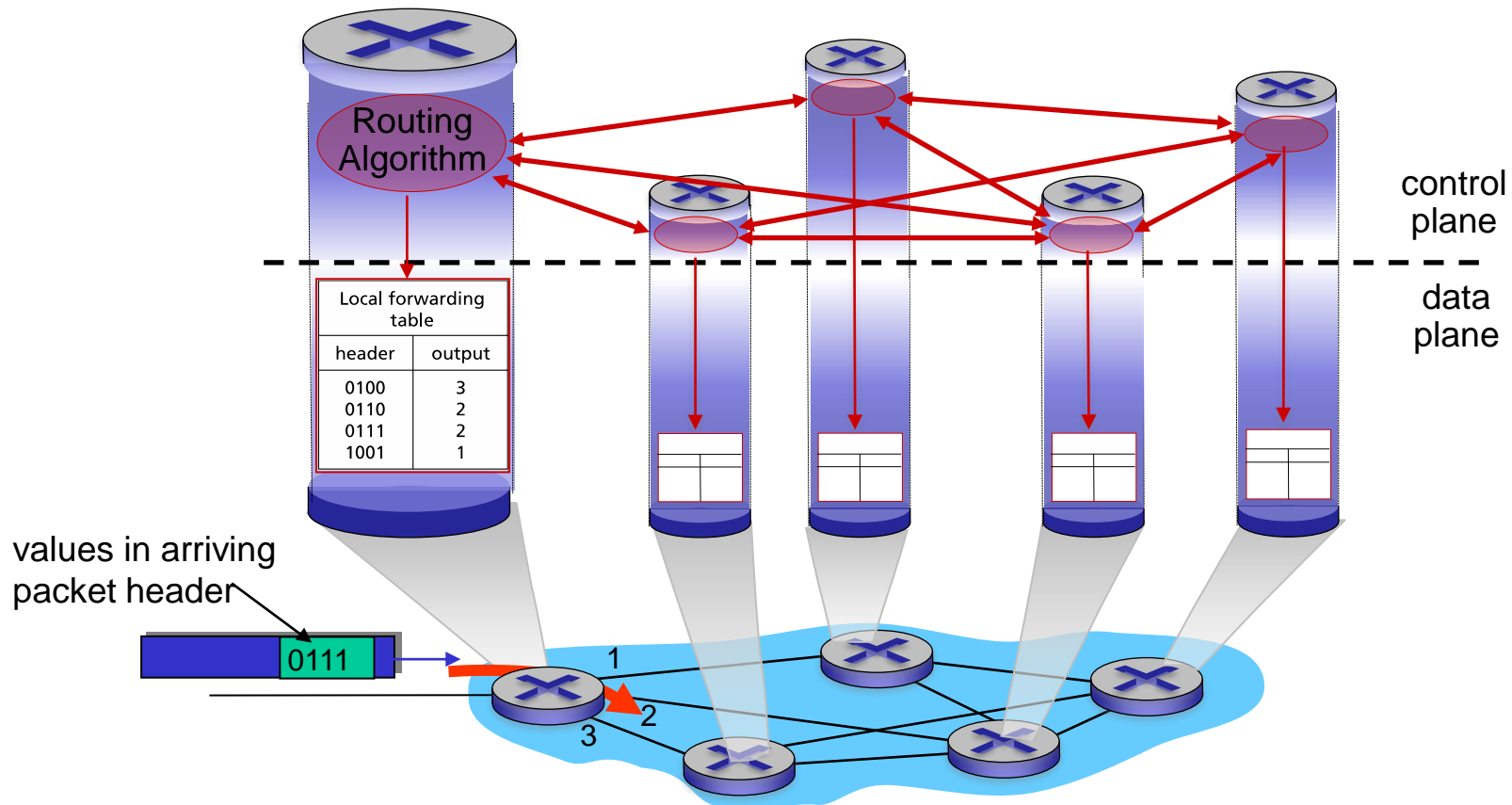
Logically centralized control plane

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

Remote Controller

control plane

data plane

values in arriving packet header

0111

1

3

2

Network service model

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

example services for individual datagrams:

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

example services for a flow of datagrams:

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

Network layer service models:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

CBR: Peak-rate allocation

rate ↑ Traffic

Peak Rate (CBR)

Effective Bandwidth (VBR)

ABR

$\|$

Stochastic QoS
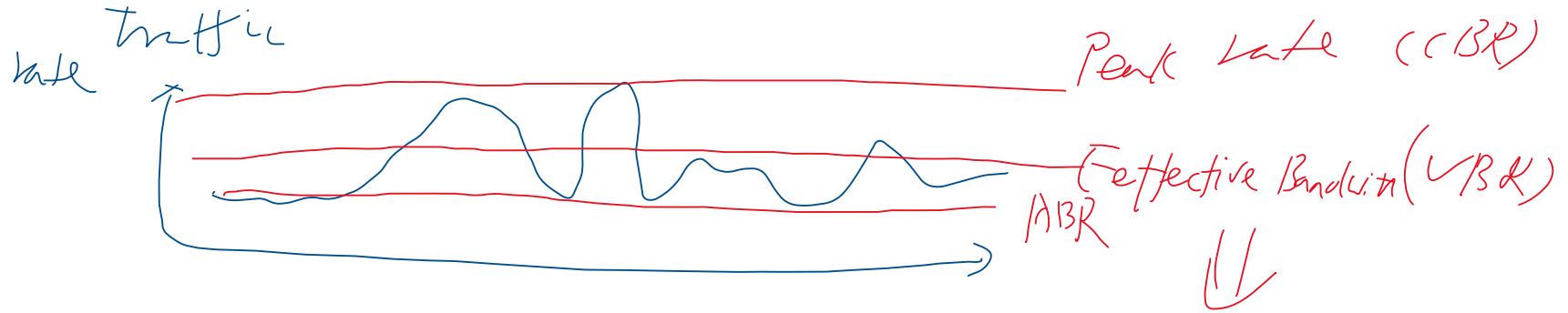
Prob(delay > 10ms)
$\leq 5\%$

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane
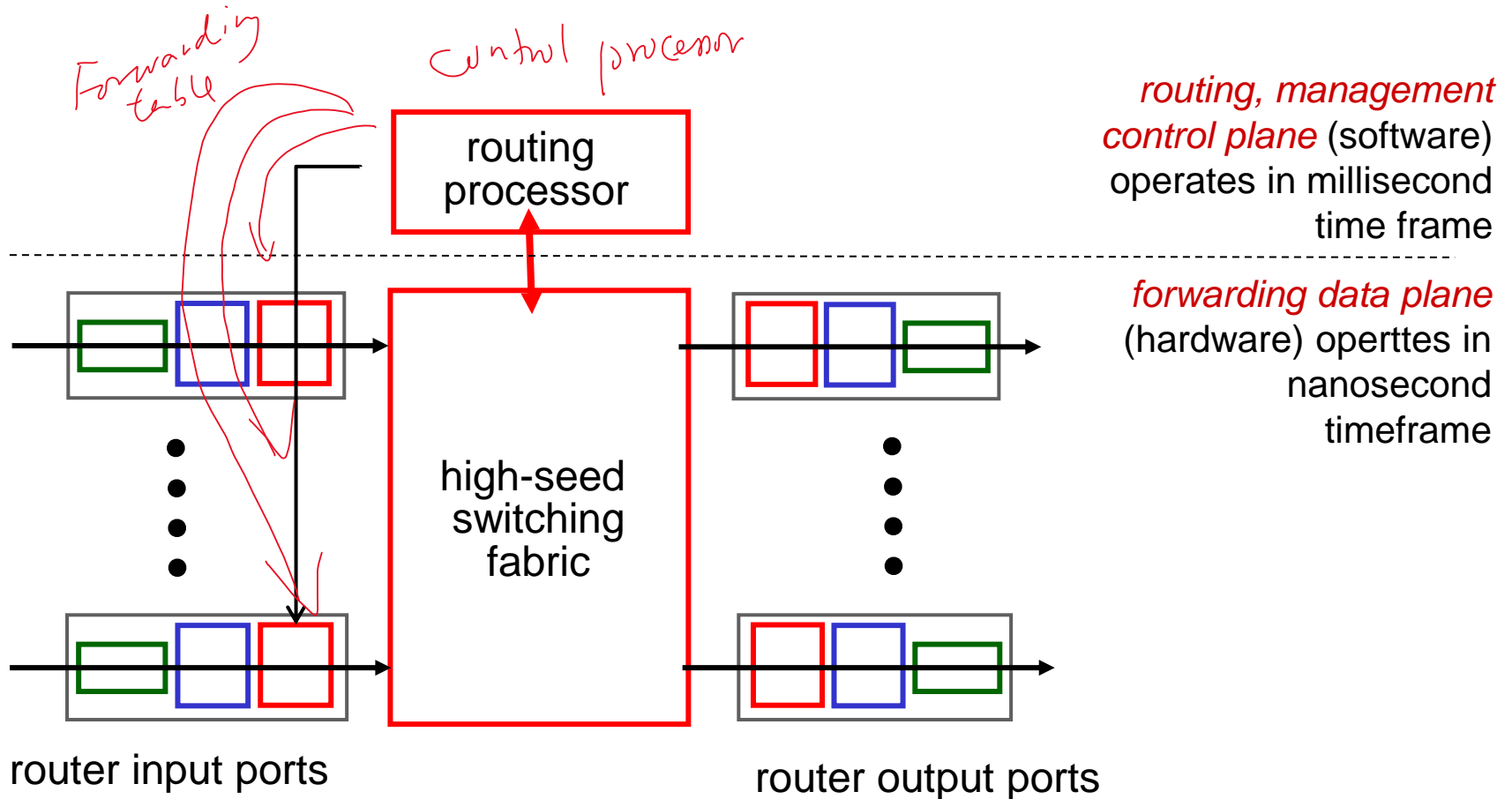
4.2 What's inside a router

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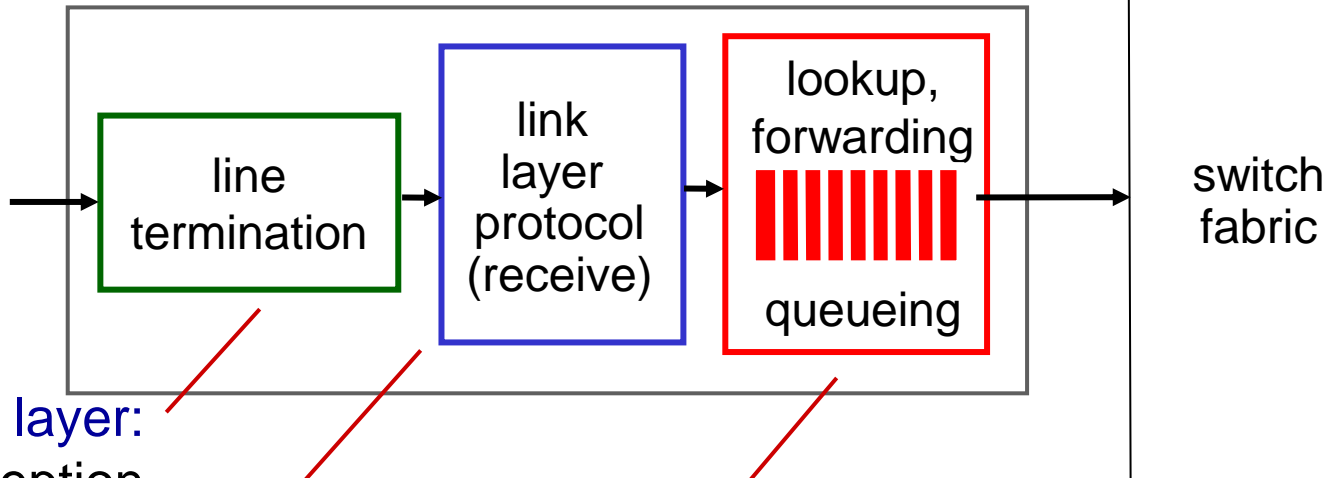- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Router architecture overview

- high-level view of generic router architecture:

Forwarding table

Control process

routing processor

high-seed switching fabric

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

forwarding data plane (hardware) operttes in nanosecond timeframe

router input ports

router output ports

Input port functions

physical layer:
bit-level reception

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

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory ("*match plus action*")

- goal: complete input port processing at 'line speed'

- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions
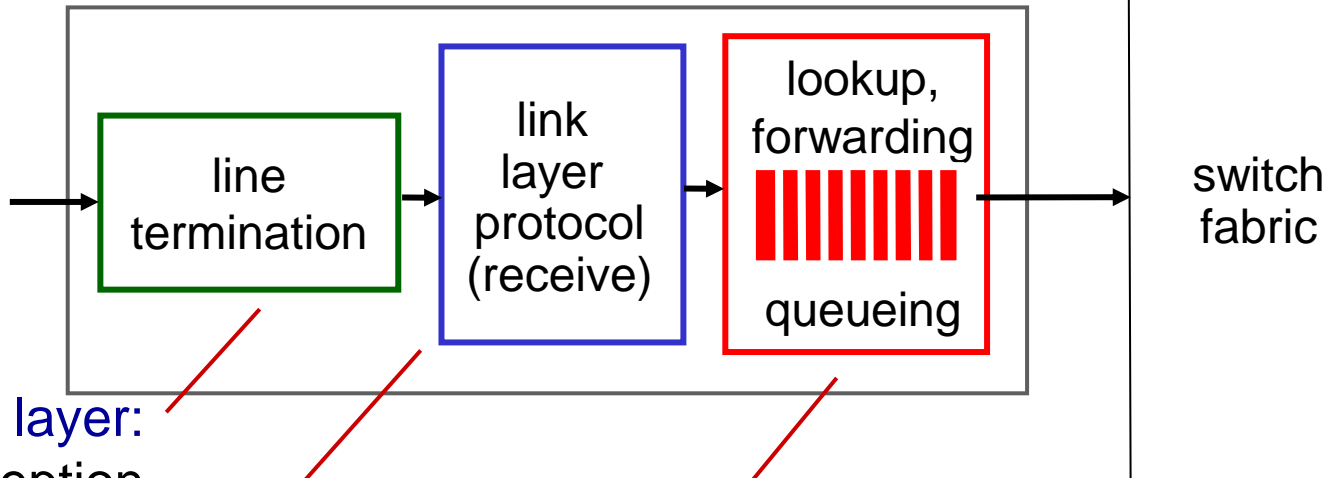
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory ("*match plus action*")

- *destination-based forwarding:* forward based only on destination IP address (traditional)

- *generalized forwarding:* forward based on any set of header field values

Longest prefix matching

longest prefix matching
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:

DA: 11001000 00010111 00010110 10100001   which interface?

DA: 11001000 00010111 00011000 10101010   which interface?

Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing

- longest prefix matching: often performed using ternary content addressable memories (TCAMs)

  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size

  - Cisco Catalyst: can up ~1M routing table entries in TCAM

Switching fabrics

- **transfer packet from input buffer to appropriate output buffer**

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

- **three types of switching fabrics**

memory   bus   crossbar
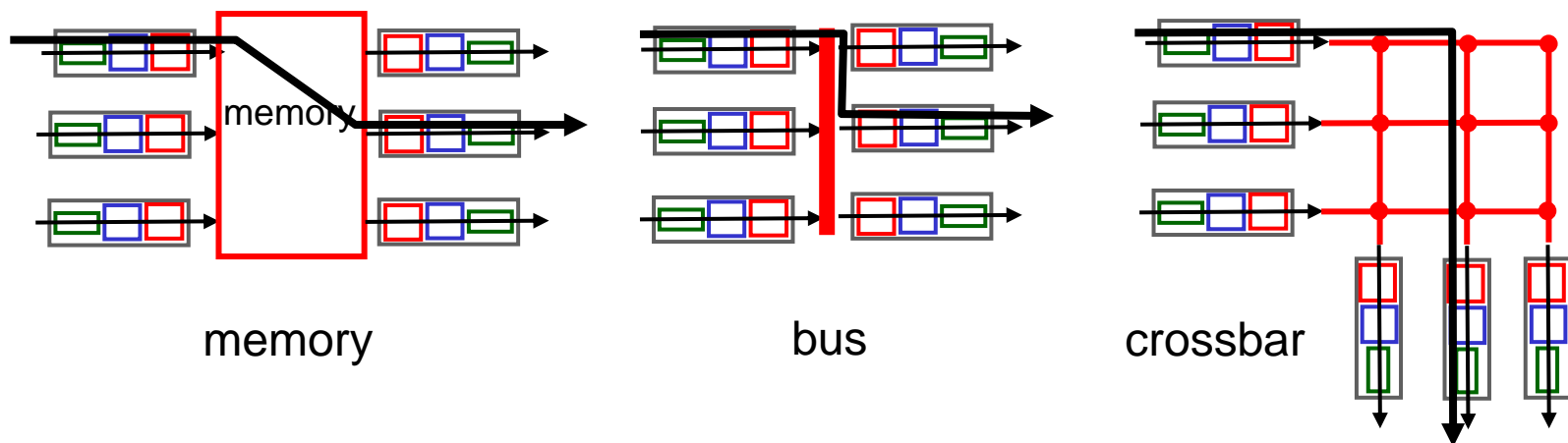
Switching via memory

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)

input port (e.g., Ethernet) — memory — output port (e.g., Ethernet) — system bus
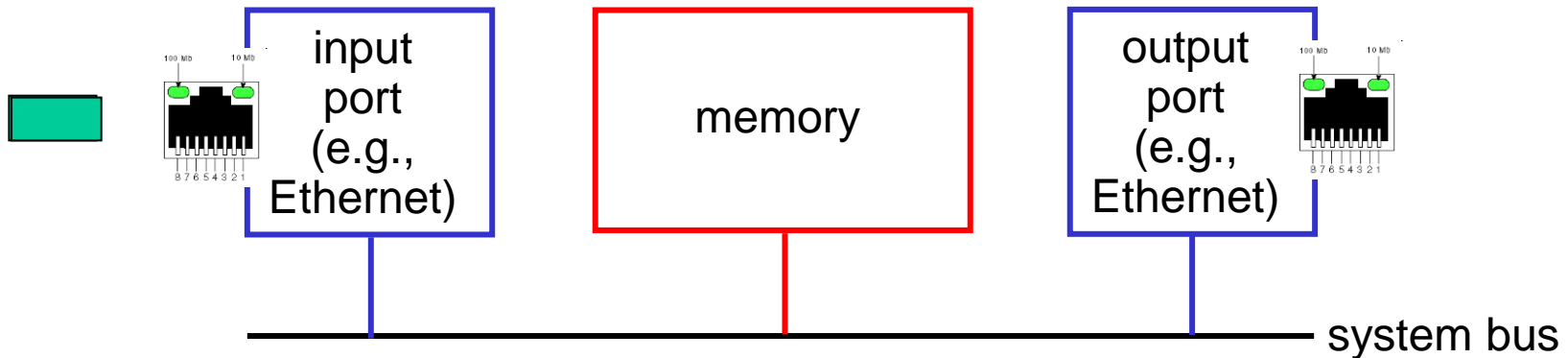
Switching via a bus

pkts processed Serial

- datagram from input port memory to output port memory via a shared bus

- *bus contention:* switching speed limited by bus bandwidth

- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

bus
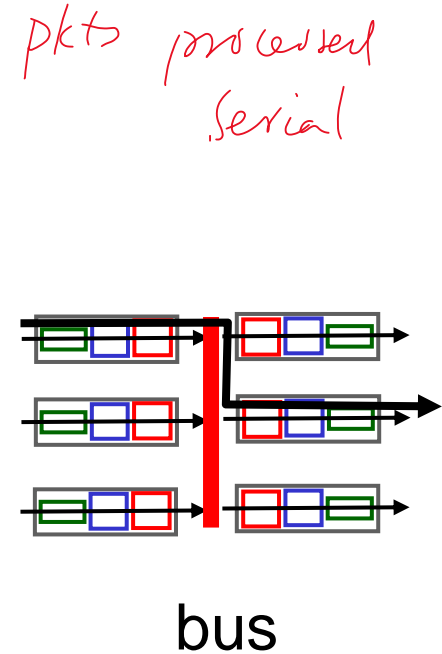
Switching via interconnection network

- overcome bus bandwidth limitations

- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor

- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.

- Cisco 12000: switches 60 Gbps through the interconnection network

crossbar
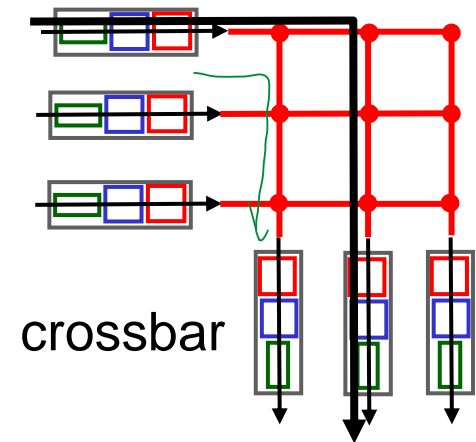
Input port queuing

- fabric slower than input ports combined **->** queueing may occur at input queues
  - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
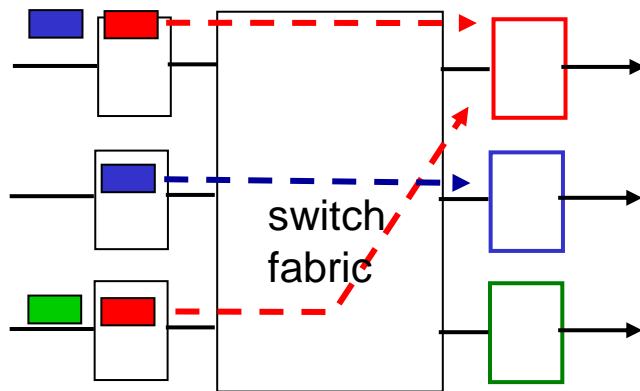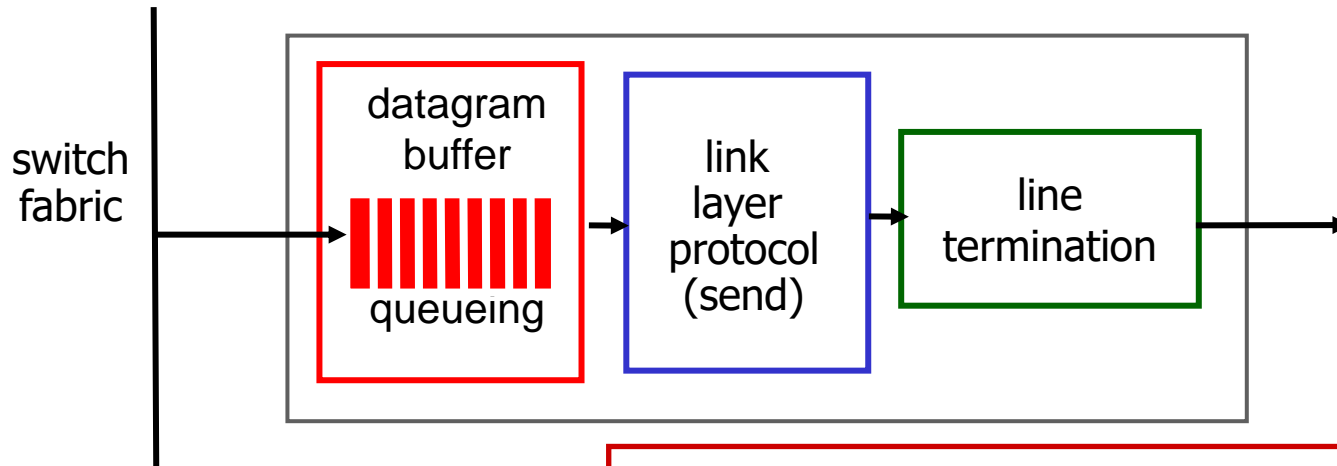
output port contention:
only one red datagram can be transferred.
lower red packet is blocked
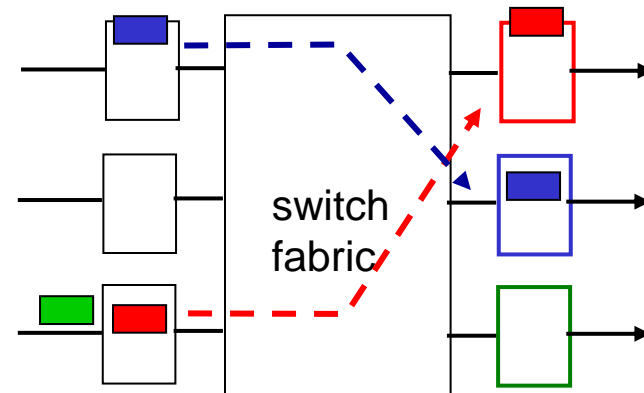
one packet time later:
green packet
experiences HOL
blocking

Output ports

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

- *buffering* required when datagrams arrive from fabric faster than the transmission rate

Priority scheduling – who gets best performance, network neutrality

- *scheduling discipline* chooses among queued datagrams for transmission

Output port queueing

at *t*, packets more
from input to output

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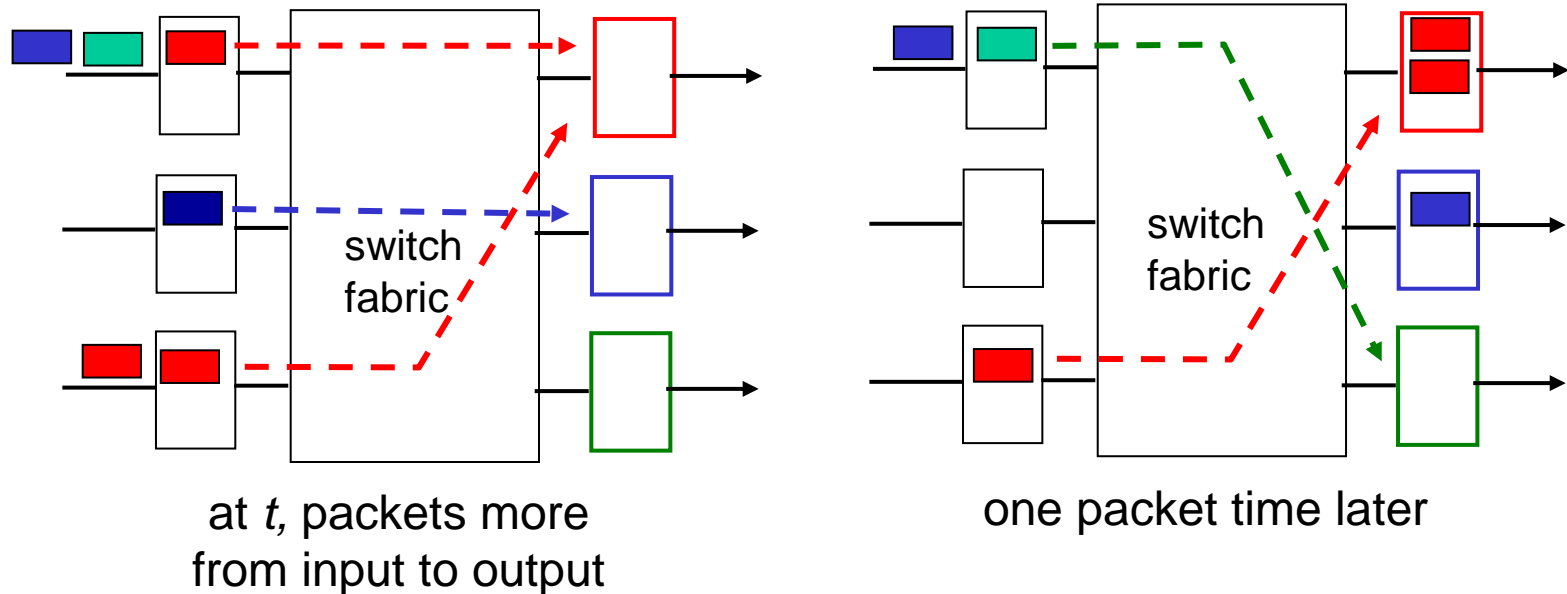
- **buffering when arrival rate via switch exceeds output line speed**
- *queueing (delay) and loss due to output port buffer overflow!*

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with *N* flows, buffering equal to

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

Scheduling mechanisms

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

packet
arrivals

queue
(waiting area)
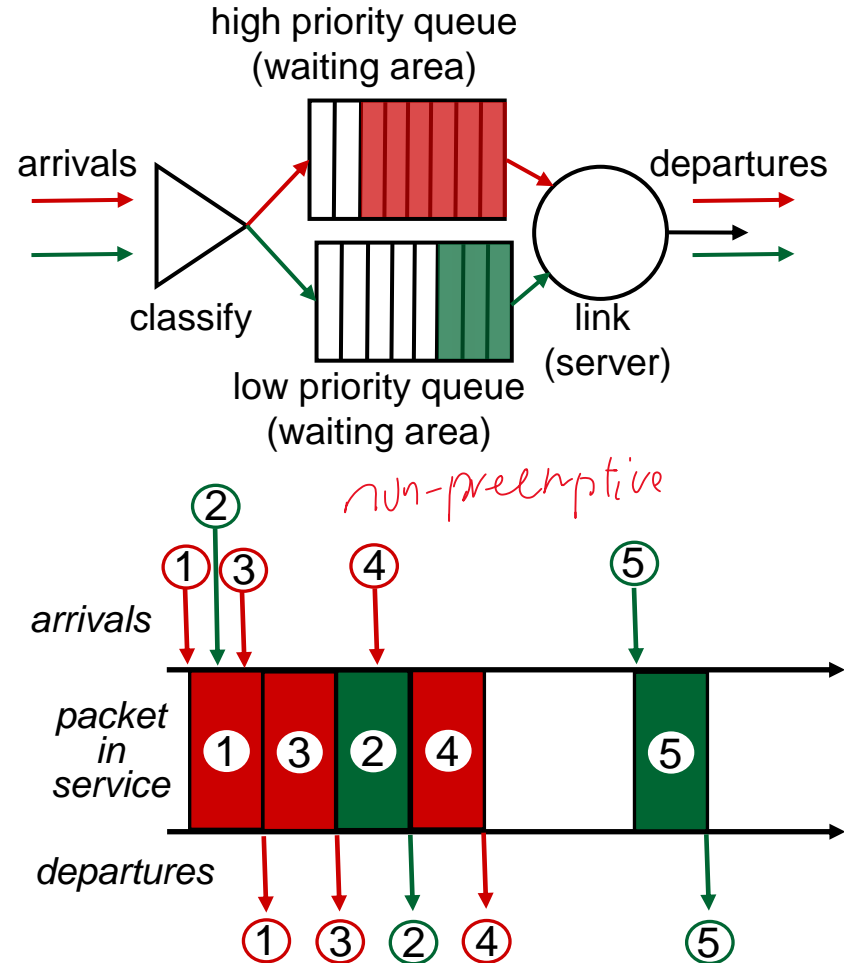
link
(server)
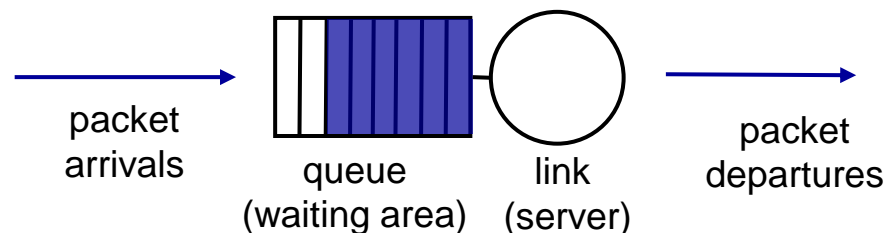
packet
departures

Scheduling policies: priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
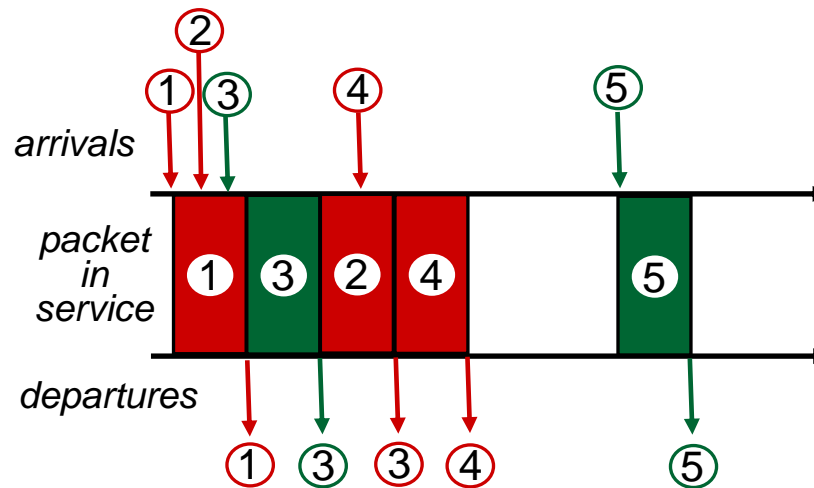 - real world example?

Differentiated Services

high priority queue (waiting area)

arrivals

classify

low priority queue (waiting area)

link (server)

departures

non-preemptive

arrivals

packet in service

departures

Scheduling policies: still more

Round Robin (RR) scheduling:

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

Scheduling policies: still more

Weighted Fair Queuing (WFQ):

$$W_1 + W_2 + W_3 = 100\%$$
$$50\% \quad 30\% \quad 20\%$$

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?