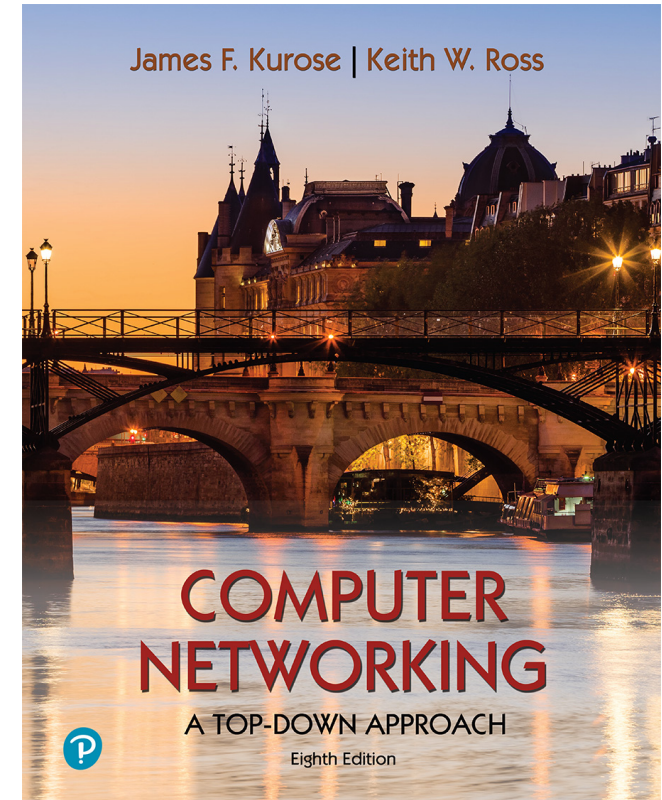


Chapter 1

Introduction



Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

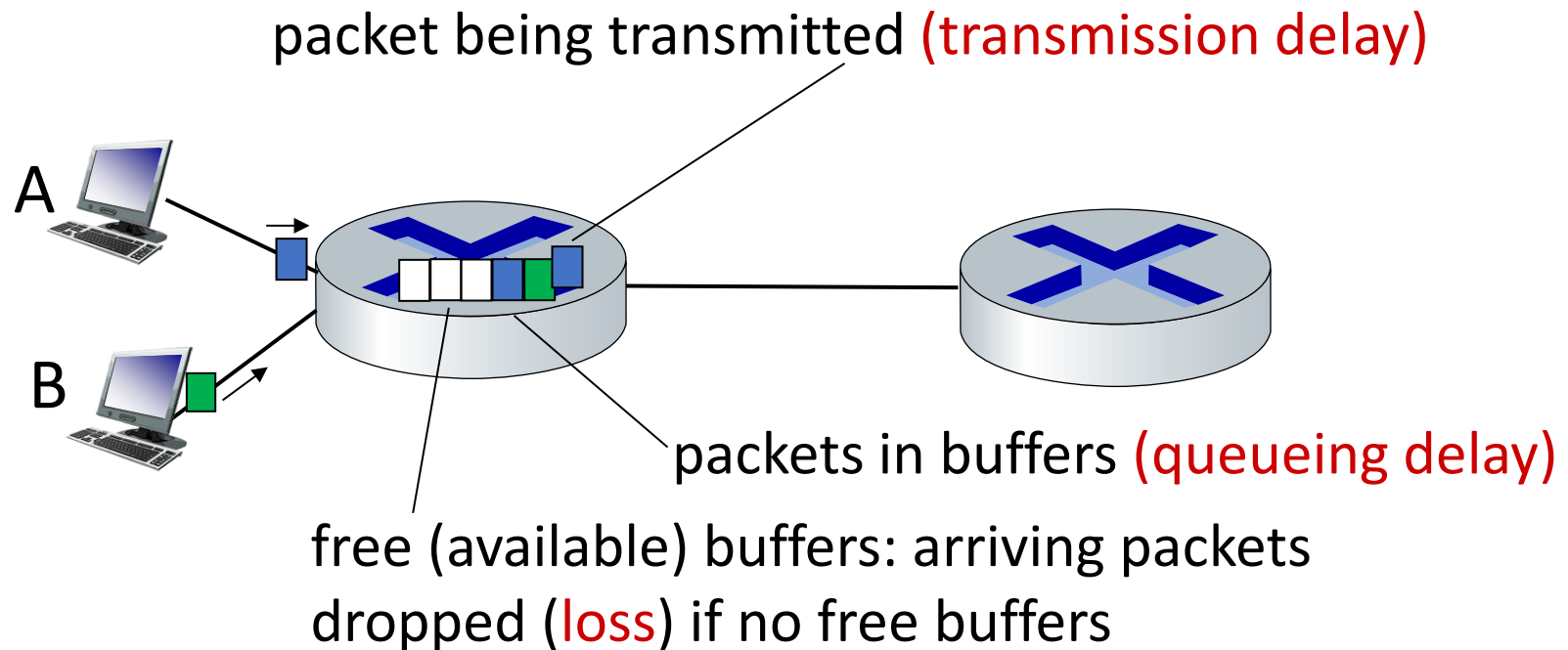
Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- **Performance: loss, delay, throughput**
- Security
- Protocol layers, service models

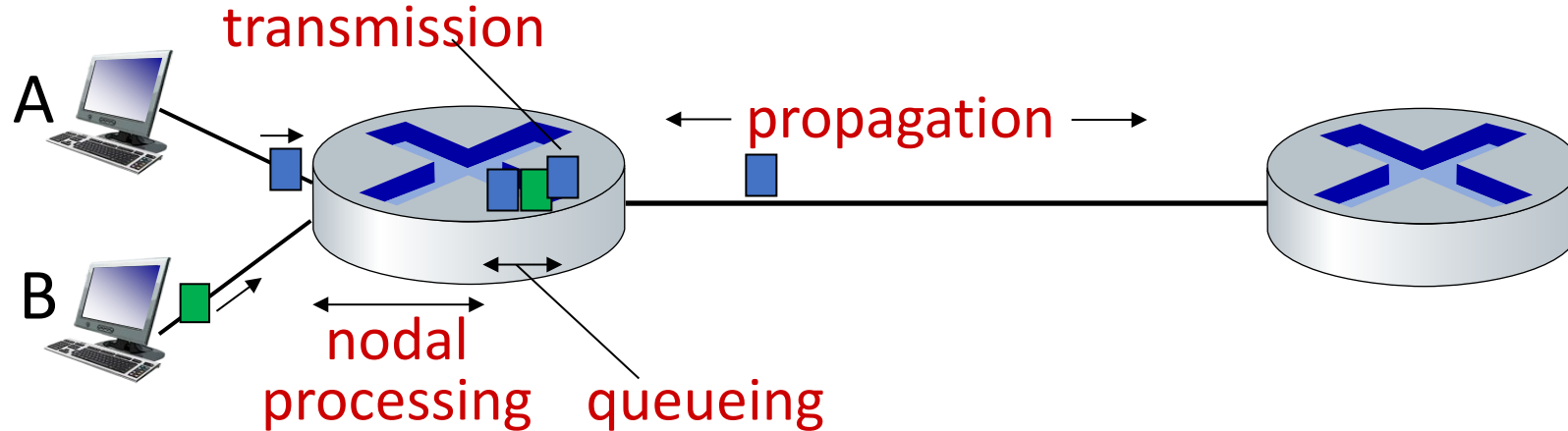


How do packet delay and loss occur?

- packets *queue* in router buffers, waiting for turn for transmission
 - queue length grows when arrival rate to link (temporarily) exceeds output link capacity
- packet *loss* occurs when memory to hold queued packets fills up



Packet delay: four sources



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

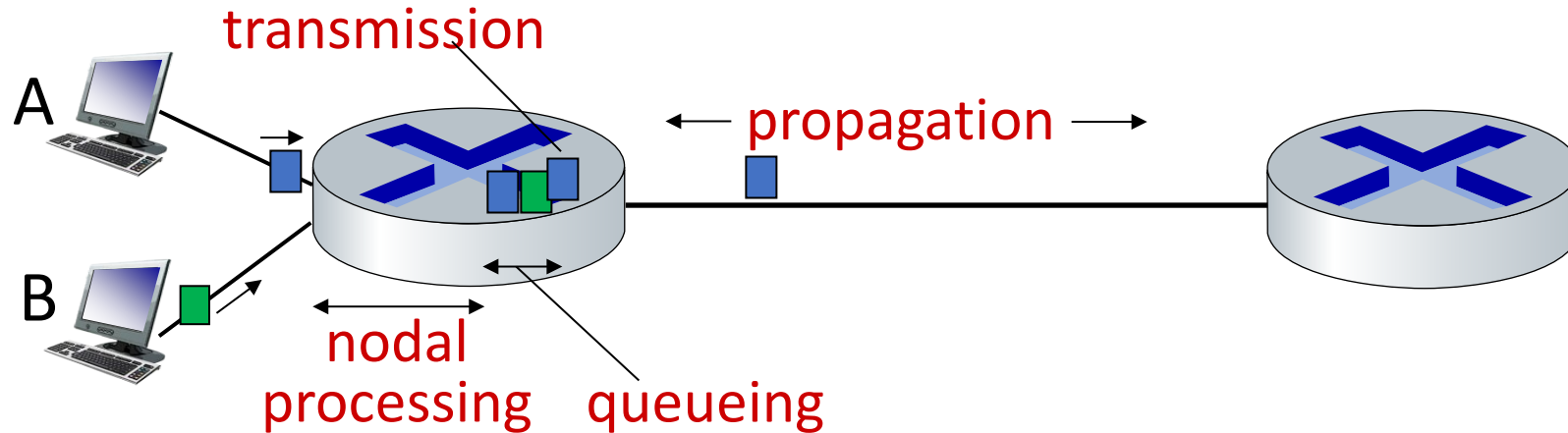
d_{proc} : nodal processing

- check bit errors
- determine output link
- typically < microsecs

d_{queue} : queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Packet delay: four sources



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

d_{trans} : transmission delay:

- L : packet length (bits)
- R : link transmission rate (bps)

▪ $d_{\text{trans}} = L/R$

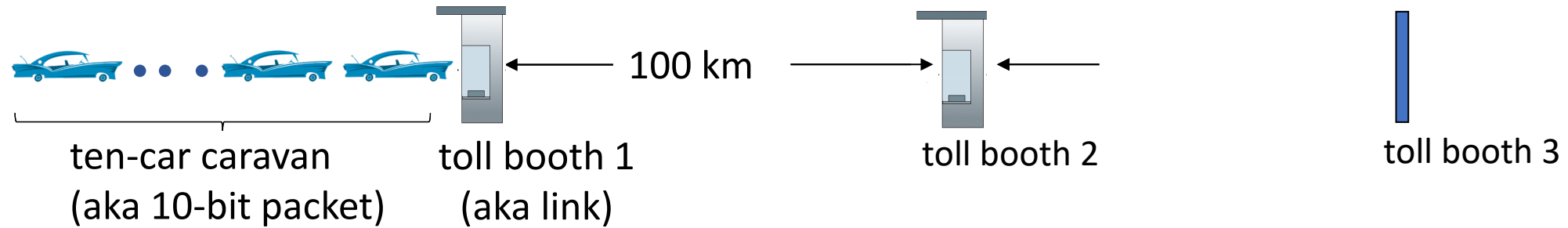
d_{prop} : propagation delay:

- d : length of physical link
- s : propagation speed ($\sim 2 \times 10^8$ m/sec)

▪ $d_{\text{prop}} = d/s$

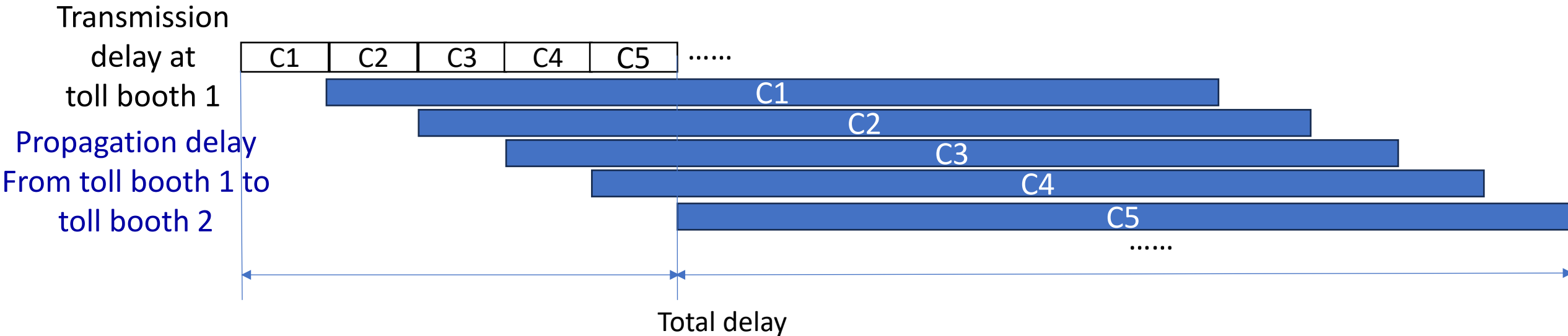
d_{trans} and d_{prop}
very different

Caravan analogy



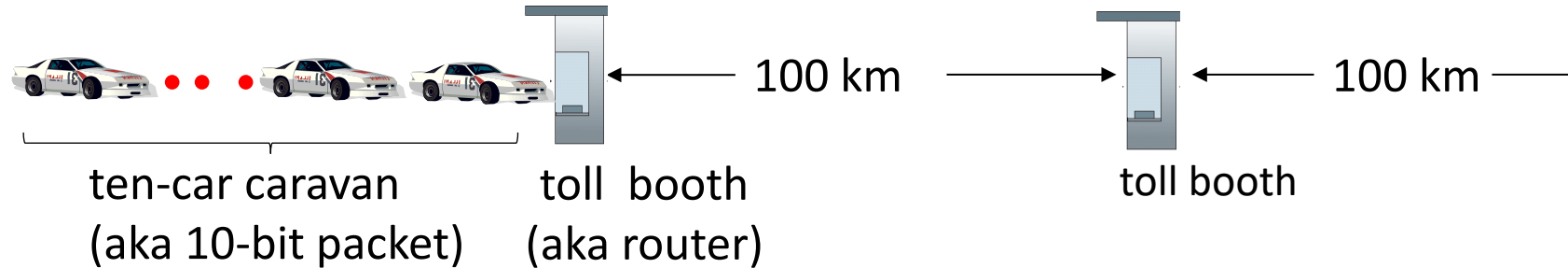
- car \sim bit; caravan \sim packet; toll service \sim link transmission
- toll booth takes 12 sec to service car (bit transmission time)
- “propagate” at 100 km/hr
- **Q: How long until all caravans (packets) arrive at 2nd toll booth?**
- time to “push” entire caravan through toll booth onto highway = $12 * 10 = 120$ sec
- time for last car to propagate from 1st to 2nd toll booth: $100\text{km} / (100\text{km/hr}) = 1$ hr
- **A: 62 minutes**

How long until all cars (packets) arrive at 2nd toll booth



- Total delay for sending all cars from source to toll booth 1 to toll booth 2 =
 - Transmission delay of all cars (also the last car) at toll booth 1 = 12×10 (120 sec), plus
 - Propagation delay of the last car from toll booth 1 to toll booth 2 = $100\text{km}/(100\text{km/hr})$ (1 hr)
 - = 62 minutes

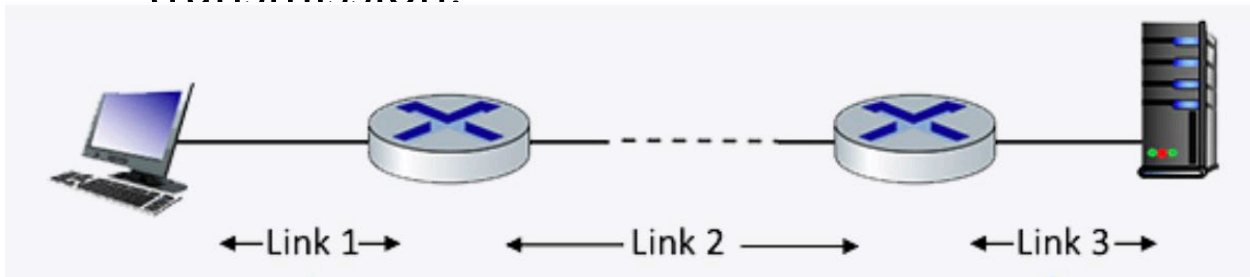
Caravan analogy



- suppose cars now “propagate” at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- **Q: Will cars arrive to 2nd booth before all cars serviced at first booth?**
A: Yes! after 7 min, first car arrives at second booth; three cars still at first booth
1st car: transmission delay 1 min + prop delay $100\text{km}/(1000\text{km/hr}) = 0.1 \text{ hr} = 6 \text{ min}$

Quiz

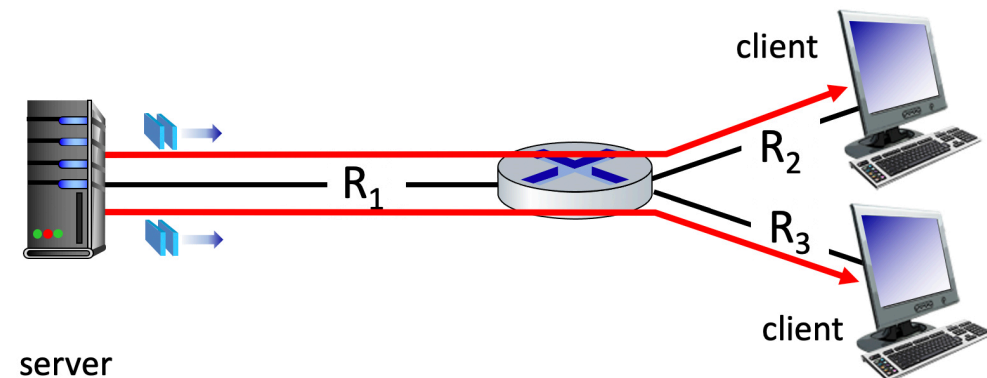
- Performance: Delay. Consider the network shown in the figure below, with three links, each with a transmission rate of 1 Mbps, and a propagation delay of 2 msec per link. Assume the length of a packet is 1000 bits.
- What is the end-end delay of a packet from when it first begins transmission on link 1, until it is received in full by the server at the end of link 3.
- You can assume that queueing delays and packet processing delays are zero, but make sure you include packet transmission time delay on all links. Assume store-and-forward packet transmission.
- ANS: at each link, transmission delay is $1000 \text{ bits} / 1 \text{ Mbps} = 1 \text{ ms}$, propagation delay is 2 ms, so delay is $1 + 2 = 3 \text{ ms}$.
- To traverse all three links, total delay is $3 + 3 + 3 = 12 \text{ ms}$.



Quiz

- Performance: Packet Transmission Delay. Consider the network shown below, with a sending server on the left, sending packets to two different client receivers on the right. The sender is sending packets to the receivers over separate TCP connections. The links have transmission rates of $R_1 = 1$ Mbps. Assume that $R_2 = R_3 = 1$ Mbps. Assume that the propagation delay is 2 msec per link. Suppose each packet is 1000 bits in size.
- What is the end-to-end delay of a packet from when it first begins transmission at the sender, until it is received in full by one of the two clients at the right (the answer is the same for both clients)? Assume store-and-forward packet transmission. You can assume the queueing delay and processing delay are both zero.

- ANS: at each link, transmission delay is $1000 \text{ bits} / 1 \text{ Mbps} = 1 \text{ ms}$, propagation delay is 2 ms, so delay is $1 + 2 = 3 \text{ ms}$.
- To traverse all two links, total delay is $3 + 3 = 6 \text{ ms}$.
- The 2nd stage, packet transmission in parallel.

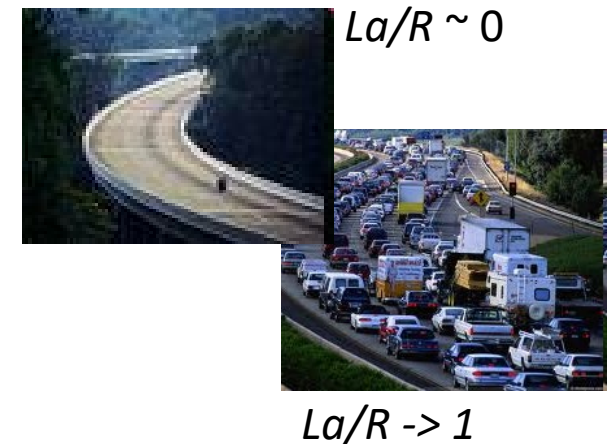
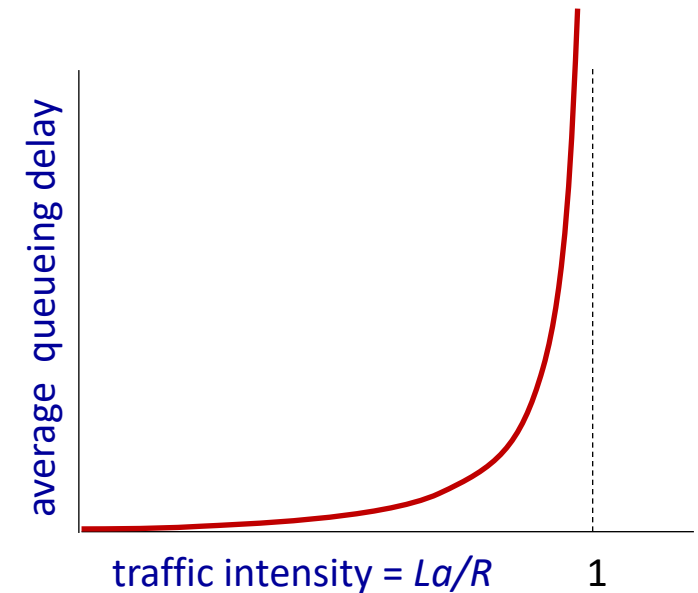


Packet queueing delay (revisited)

- a : average packet arrival rate
- L : packet length (bits)
- R : link bandwidth (bit transmission rate)

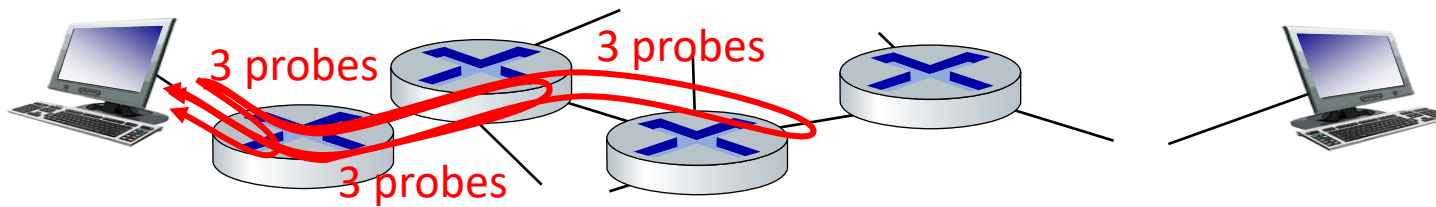
$$\frac{L \cdot a}{R} : \frac{\text{arrival rate of bits}}{\text{service rate of bits}} \quad \text{“traffic intensity”}$$

- $La/R \sim 0$: avg. queueing delay small
- $La/R \rightarrow 1$: avg. queueing delay large
- $La/R > 1$: more “work” arriving is more than can be serviced - average delay infinite!



“Real” Internet delays and routes

- what do “real” Internet delay & loss look like?
- **traceroute** program: provides delay measurement from source to router along end-end Internet path towards destination. For all i :
 - sends three packets that will reach router i on path towards destination (with time-to-live field value of i)
 - router i will return packets to sender
 - sender measures time interval between transmission and reply



Real Internet delays and routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

3 delay measurements from
gaia.cs.umass.edu to cs-gw.cs.umass.edu

3 delay measurements
to border1-rt-fa5-1-0.gw.umass.edu

trans-oceanic link

looks like delays
decrease! Why?

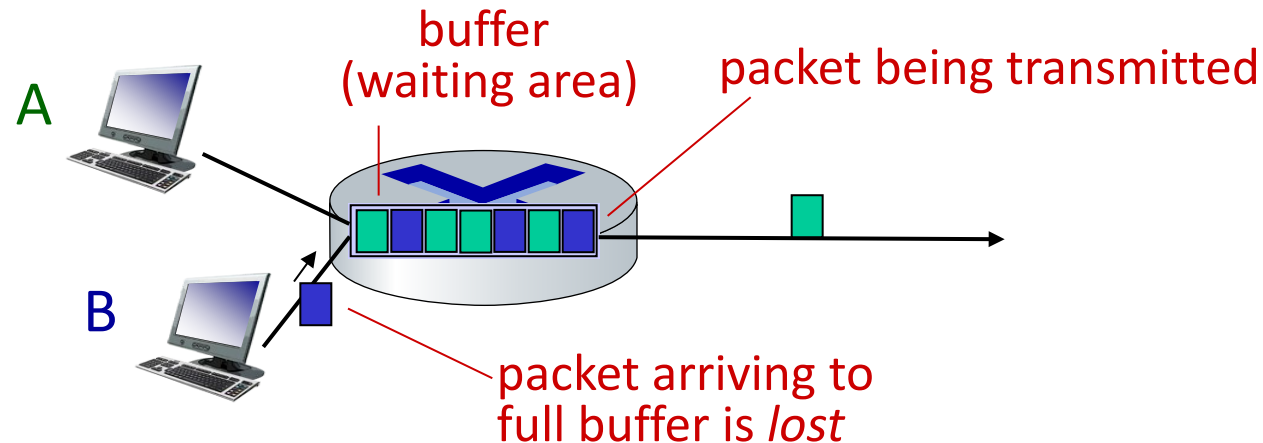
* means no response (probe lost, router not replying)

hop	hostname (ip)	1st probe	2nd probe	3rd probe
1	cs-gw (128.119.240.254)	1 ms	1 ms	2 ms
2	border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)	1 ms	1 ms	2 ms
3	cht-vbns.gw.umass.edu (128.119.3.130)	6 ms	5 ms	5 ms
4	jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)	16 ms	11 ms	13 ms
5	jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)	21 ms	18 ms	18 ms
6	abilene-vbns.abilene.ucaid.edu (198.32.11.9)	22 ms	18 ms	22 ms
7	nycm-wash.abilene.ucaid.edu (198.32.8.46)	22 ms	22 ms	22 ms
8	62.40.103.253 (62.40.103.253)	104 ms	109 ms	106 ms
9	de2-1.de1.de.geant.net (62.40.96.129)	109 ms	102 ms	104 ms
10	de.fr1.fr.geant.net (62.40.96.50)	113 ms	121 ms	114 ms
11	renater-gw.fr1.fr.geant.net (62.40.103.54)	112 ms	114 ms	112 ms
12	nio-n2.cssi.renater.fr (193.51.206.13)	111 ms	114 ms	116 ms
13	nice.cssi.renater.fr (195.220.98.102)	123 ms	125 ms	124 ms
14	r3t2-nice.cssi.renater.fr (195.220.98.110)	126 ms	126 ms	124 ms
15	eurecom-valbonne.r3t2.ft.net (193.48.50.54)	135 ms	128 ms	133 ms
16	194.214.211.25 (194.214.211.25)	126 ms	128 ms	126 ms
17	***			
18	***			
19	fantasia.eurecom.fr (193.55.113.142)	132 ms	128 ms	136 ms

* Do some traceroutes from exotic countries at www.traceroute.org

Packet loss

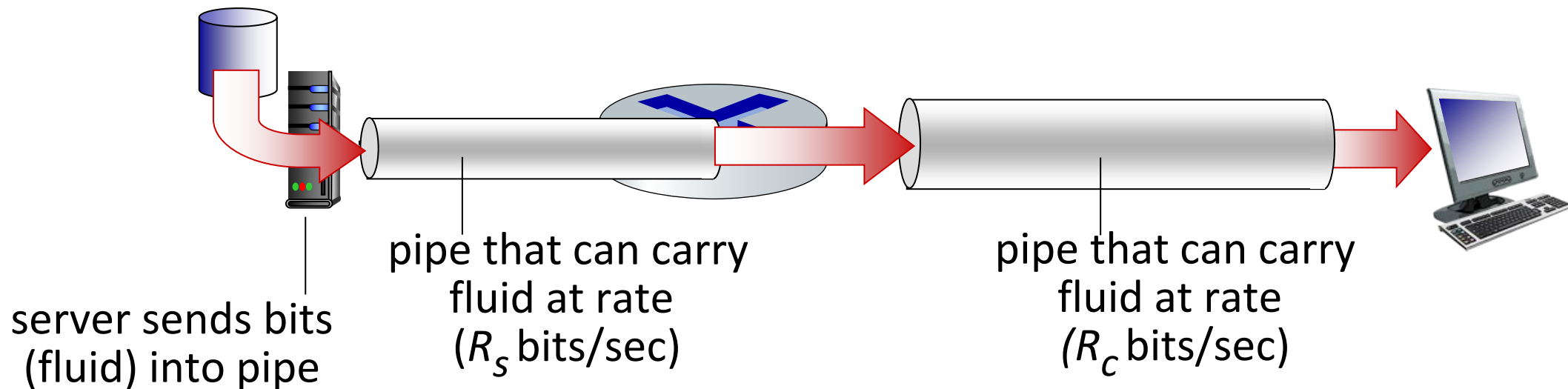
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation (on publisher's website) of queuing and loss

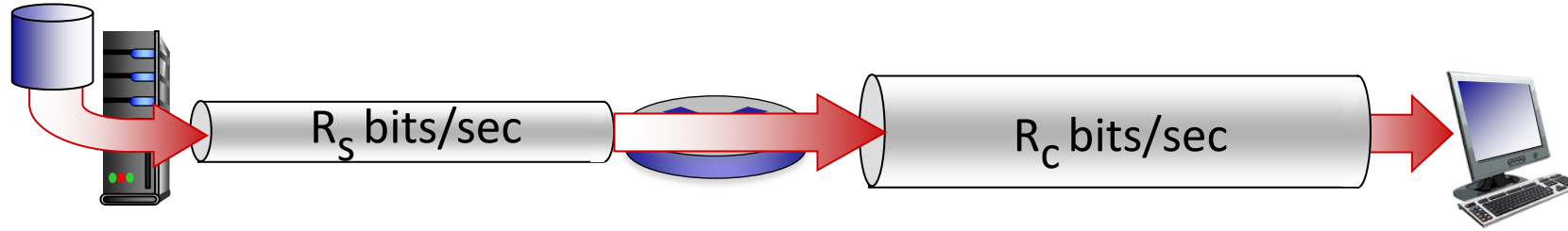
Throughput

- *throughput*: rate (bits/time unit) at which bits are being sent from sender to receiver
 - *instantaneous*: rate at given point in time
 - *average*: rate over longer period of time

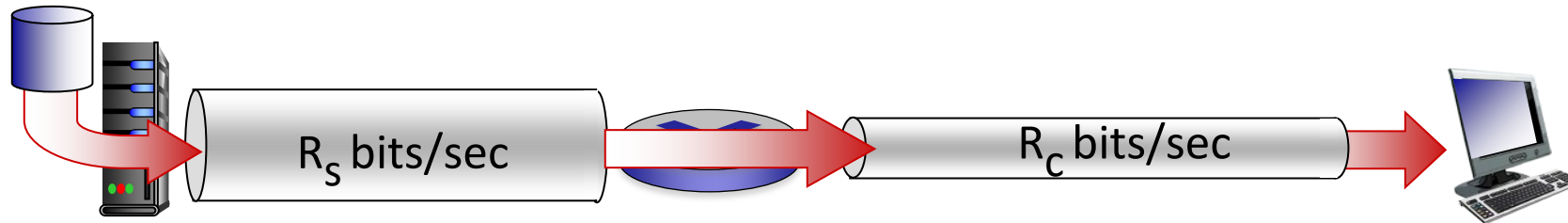


Throughput

$R_s < R_c$ What is average end-end throughput?



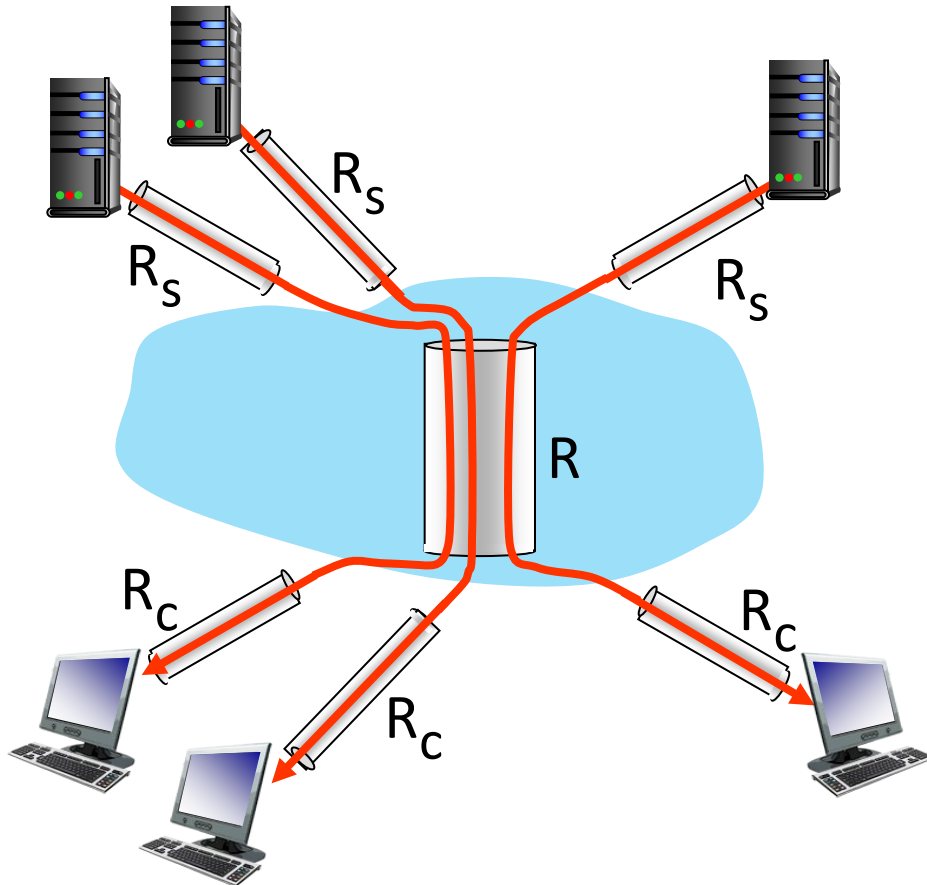
$R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

Throughput: network scenario

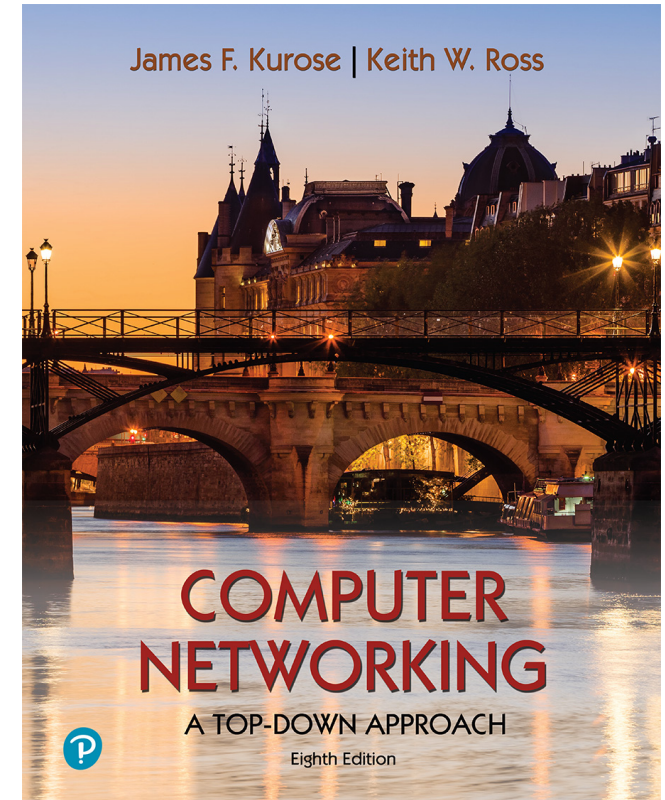


10 connections (fairly) share
backbone bottleneck link R bits/sec

- per-connection end-end throughput:
 $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck
- Link utilization: used bandwidth/available bandwidth. For the three links:
 - $\min(R_c, R_s, R/10)/R_s$
 - $\min(R_c, R_s, R/10)/(R/10)$
 - $\min(R_c, R_s, R/10)/R_c$

Chapter 2

Application Layer



Computer Networking: A Top-Down Approach

8th edition

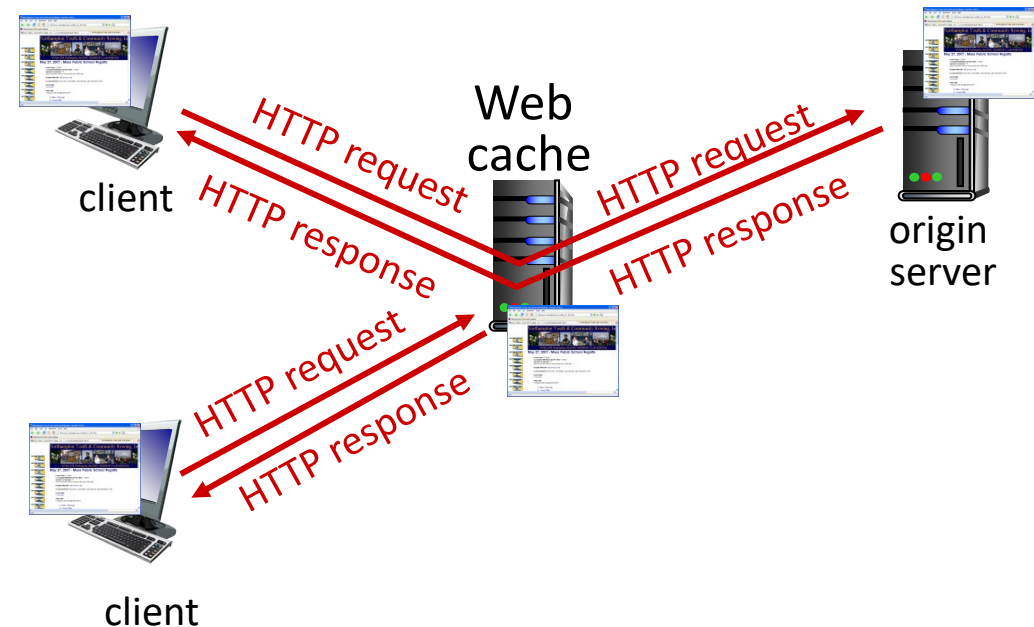
Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

Web caches

Goal: satisfy client requests without involving origin server

- user configures browser to point to a (local) *Web cache*
- browser sends all HTTP requests to cache
 - *if* object in cache: cache returns object to client
 - *else* cache requests object from origin server, caches received object, then returns object to client



Web caches (aka proxy servers)

- Web cache acts as both client and server
 - server for original requesting client
 - client to origin server
- server tells cache about object's allowable caching in response header:

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

Why Web caching?

- reduce response time for client request
 - cache is closer to client
- reduce traffic on an institution's access link
- Internet is dense with caches
 - enables “poor” content providers to more effectively deliver content

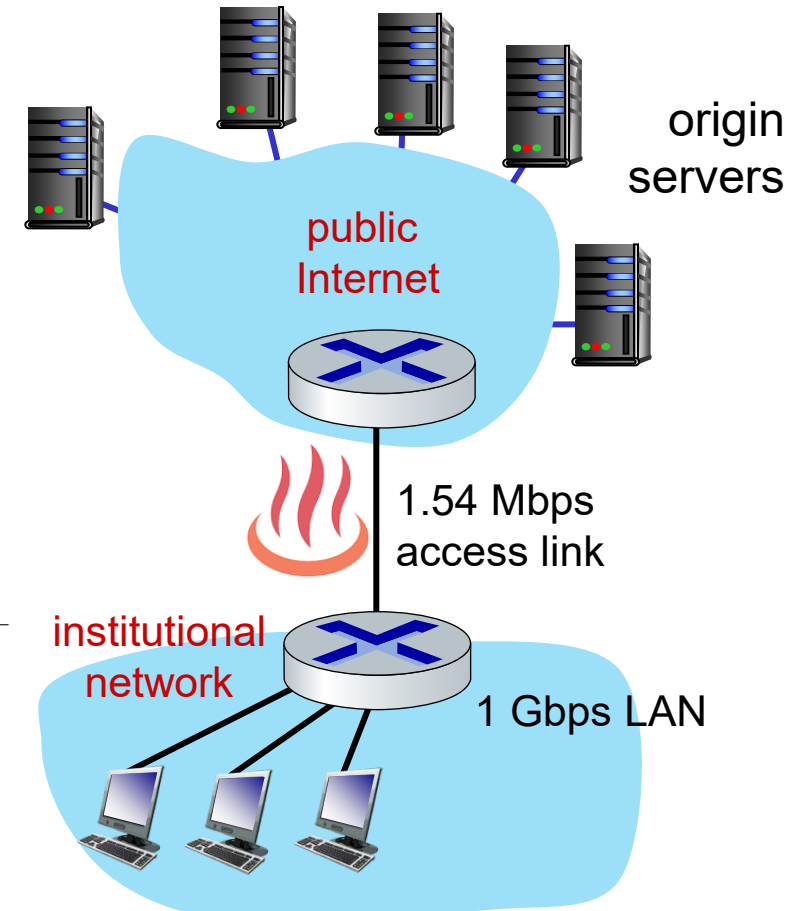
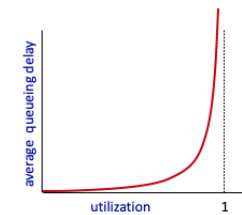
Caching example

Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
 - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = .97
(=1.50Mbps/1.54Mbps)
- LAN utilization: .0015 (=1.50Mbps/1000Mbps)
- end-end delay = Internet delay +
access link delay + LAN delay
= 2 sec + minutes + usecs



*problem: large
queueing delays
at high utilization!*

Option 1: buy a faster access link

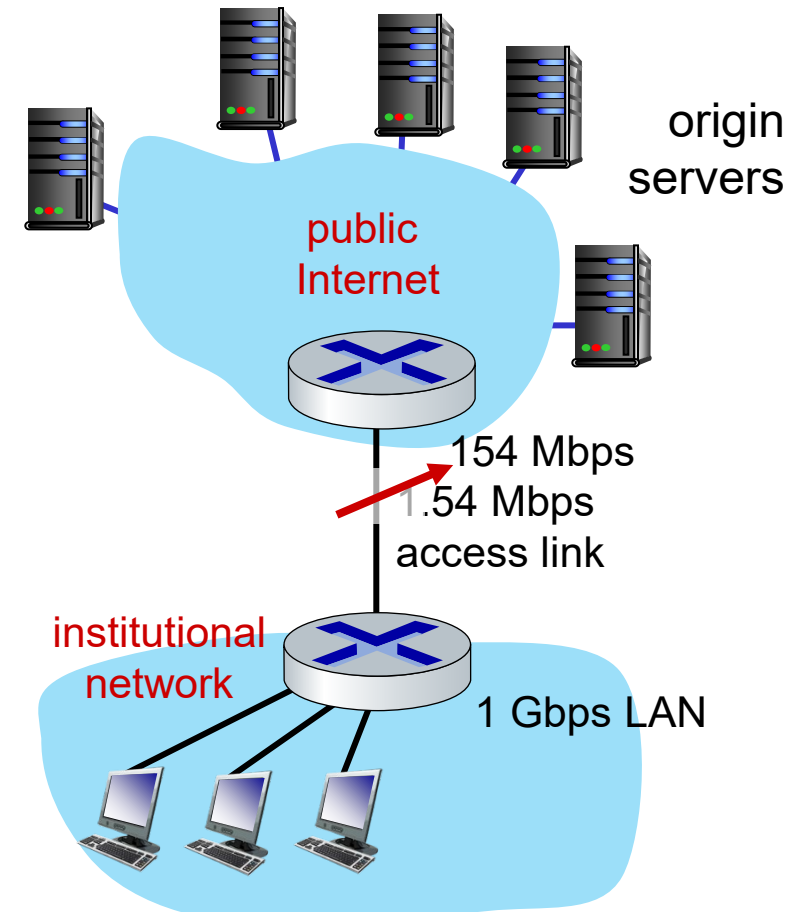
Scenario:

- access link rate: ~~1.54~~ 154 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
 - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = ~~.97~~ .0097
- LAN utilization: .0015
- end-end delay = Internet delay +
access link delay + LAN delay
= 2 sec + ~~minutes~~ + usecs

Cost: faster access link (expensive!) → msecs



Option 2: install a web cache

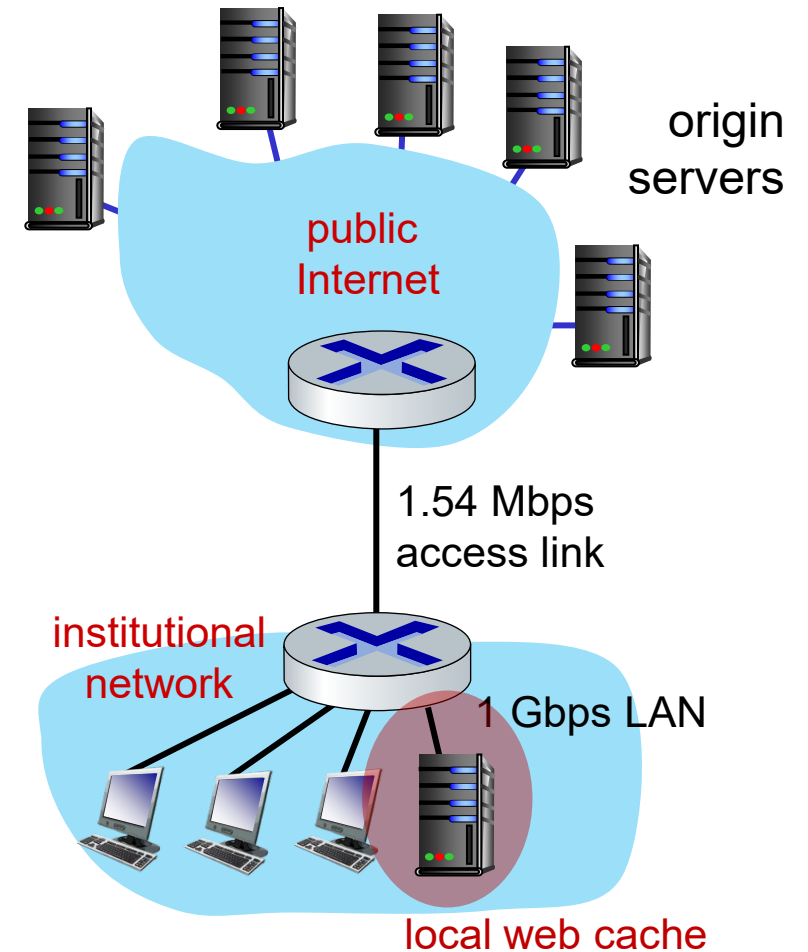
Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
 - avg data rate to browsers: 1.50 Mbps

Cost: web cache (cheap!)

Performance:

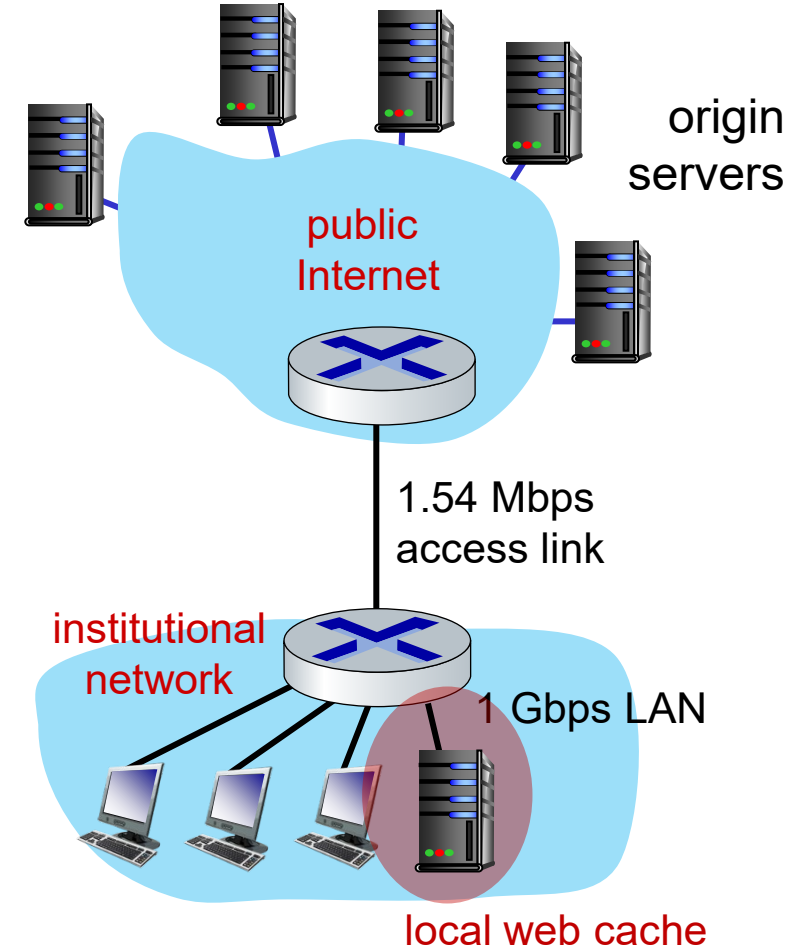
- LAN utilization: .?
 - access link utilization = ?
 - average end-end delay = ?
- How to compute link utilization, delay?*



Calculating access link utilization, end-end delay with cache:

suppose cache hit rate is 0.4:

- 40% requests served by cache, with low (msec) delay
- 60% requests satisfied at origin
 - rate to browsers over access link
 $= 0.6 * 1.50 \text{ Mbps} = .9 \text{ Mbps}$
 - access link utilization $= 0.9 / 1.54 = .58$ means low (msec) queueing delay at access link
- average end-end delay:
 $= 0.6 * (\text{delay from origin servers})$
 $+ 0.4 * (\text{delay when satisfied at cache})$
 $= 0.6 (2.01) + 0.4 (\sim \text{msecs}) = \sim 1.2 \text{ secs}$

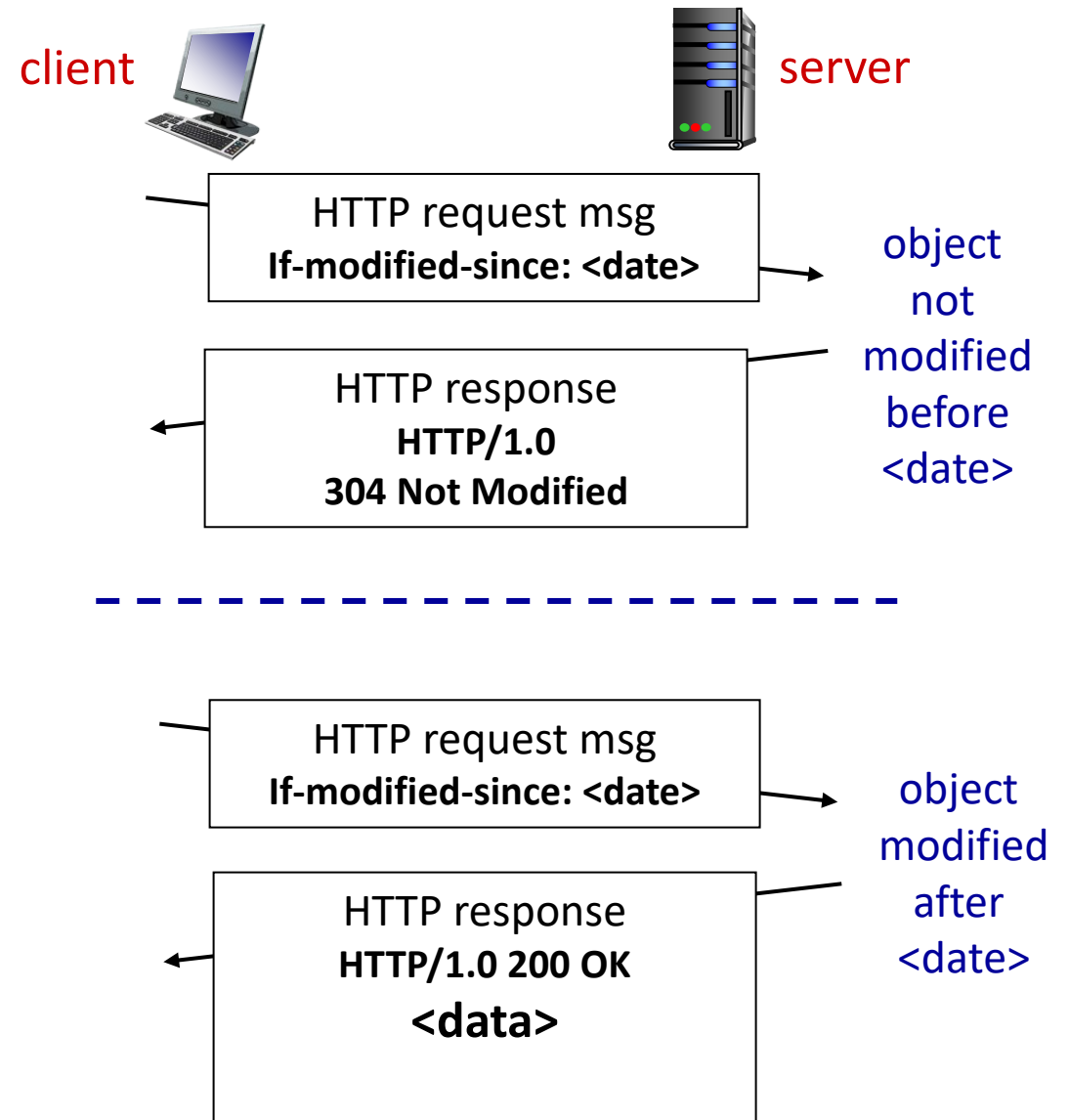


lower average end-end delay than with 154 Mbps link (and cheaper too!)

Browser caching: Conditional GET

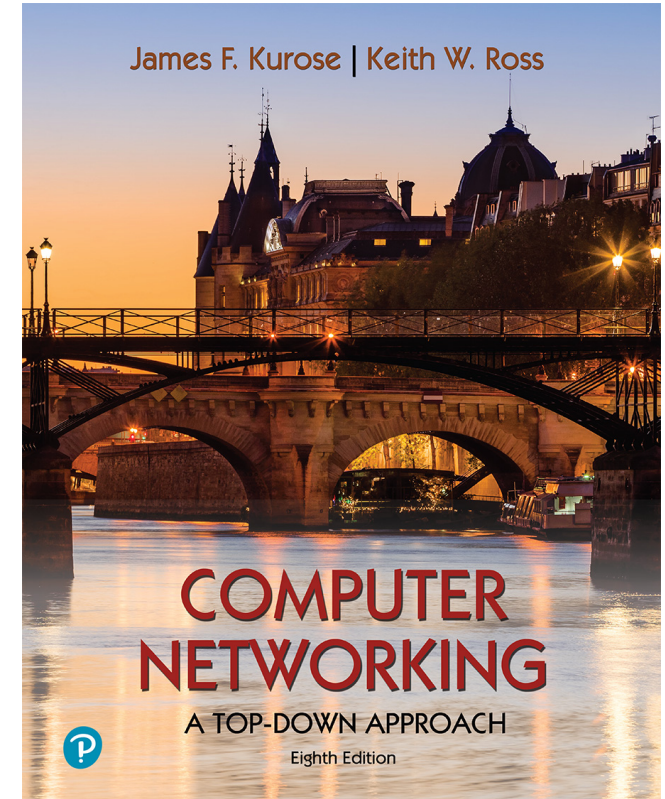
Goal: don't send object if browser has up-to-date cached version

- no object transmission delay (or use of network resources)
- **client:** specify date of browser-cached copy in HTTP request
If-modified-since: <date>
- **server:** response contains no object if browser-cached copy is up-to-date:
HTTP/1.0 304 Not Modified



Chapter 3

Transport Layer



Computer Networking: A Top-Down Approach

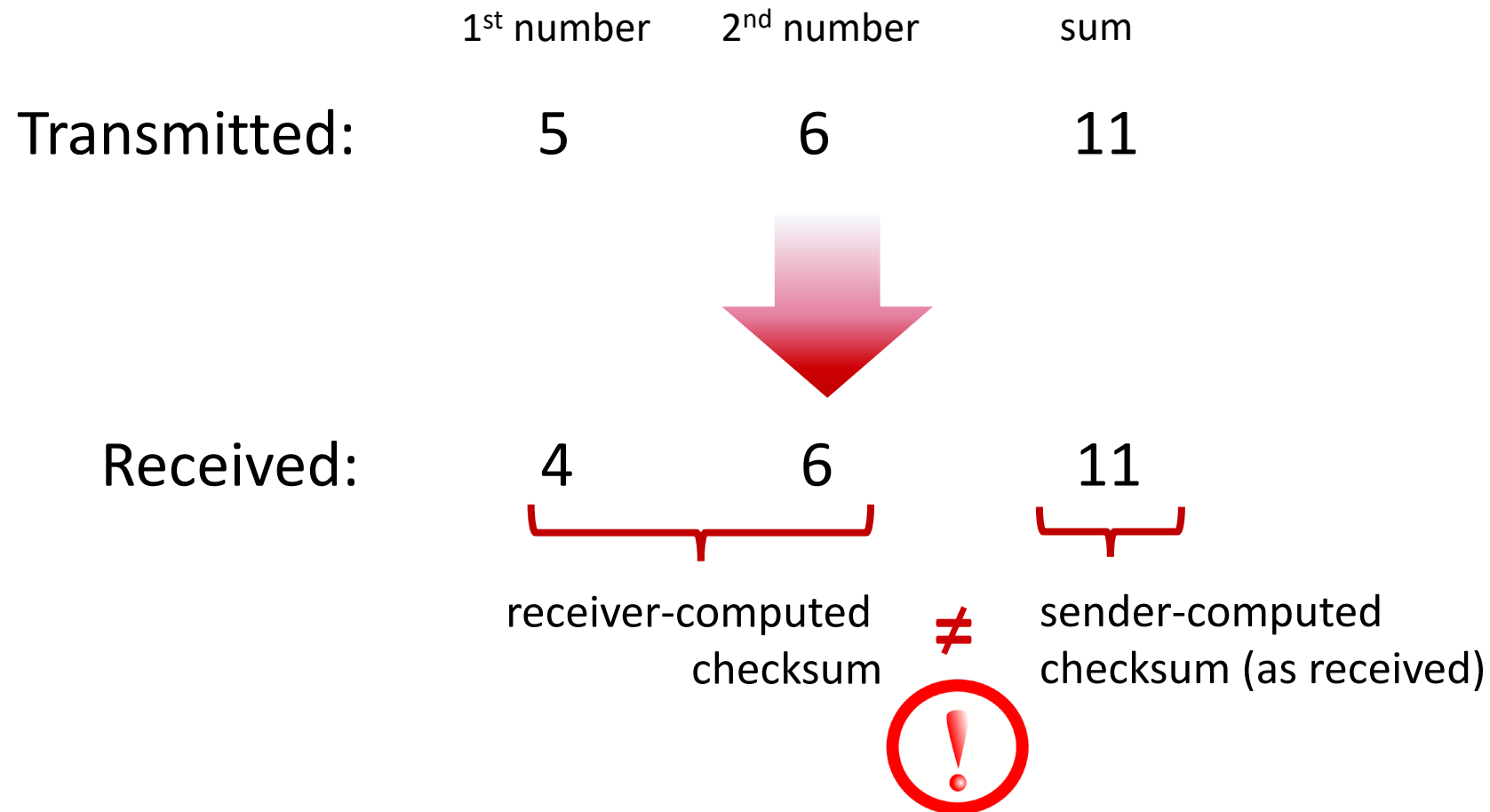
8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

UDP checksum

Goal: detect errors (*i.e.*, flipped bits) in transmitted segment



Internet checksum

Goal: detect errors (*i.e.*, flipped bits) in transmitted segment

sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of N-bit integers, where N may be 4, 8, 16...
- **checksum:** addition of segment content, then flip all the bits (one's complement sum)
- checksum value put into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
 - not equal - error detected
 - equal - no error detected. *But maybe errors nonetheless?* More later

Internet checksum: an example

example: add two N=16-bit integers

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
<hr/>																
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

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

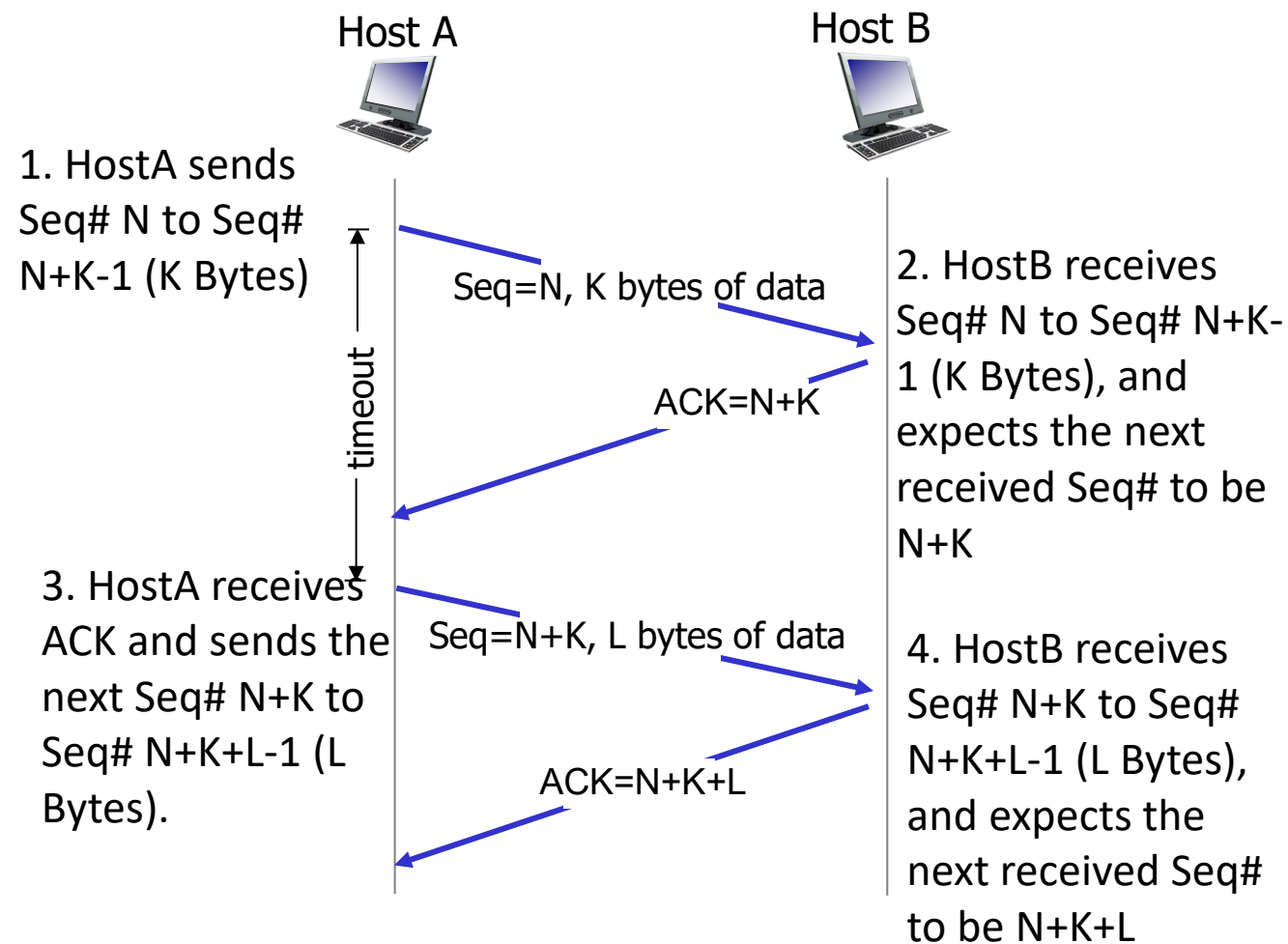
Internet checksum: weak protection!

example: add two N=16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		<hr/>															
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

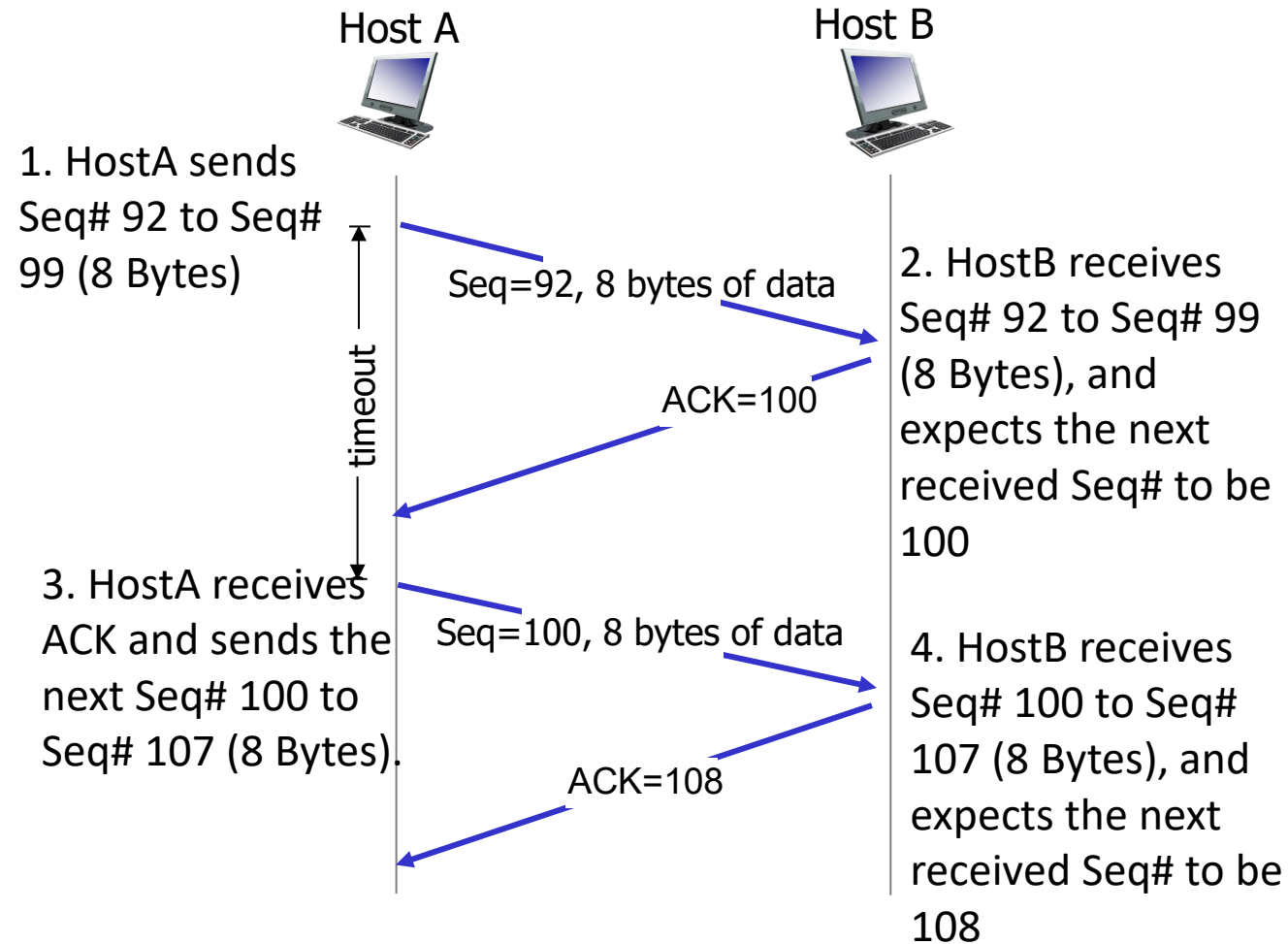
Even though numbers have changed (bit flips), *no* change in checksum!

TCP sequence numbers, ACKs

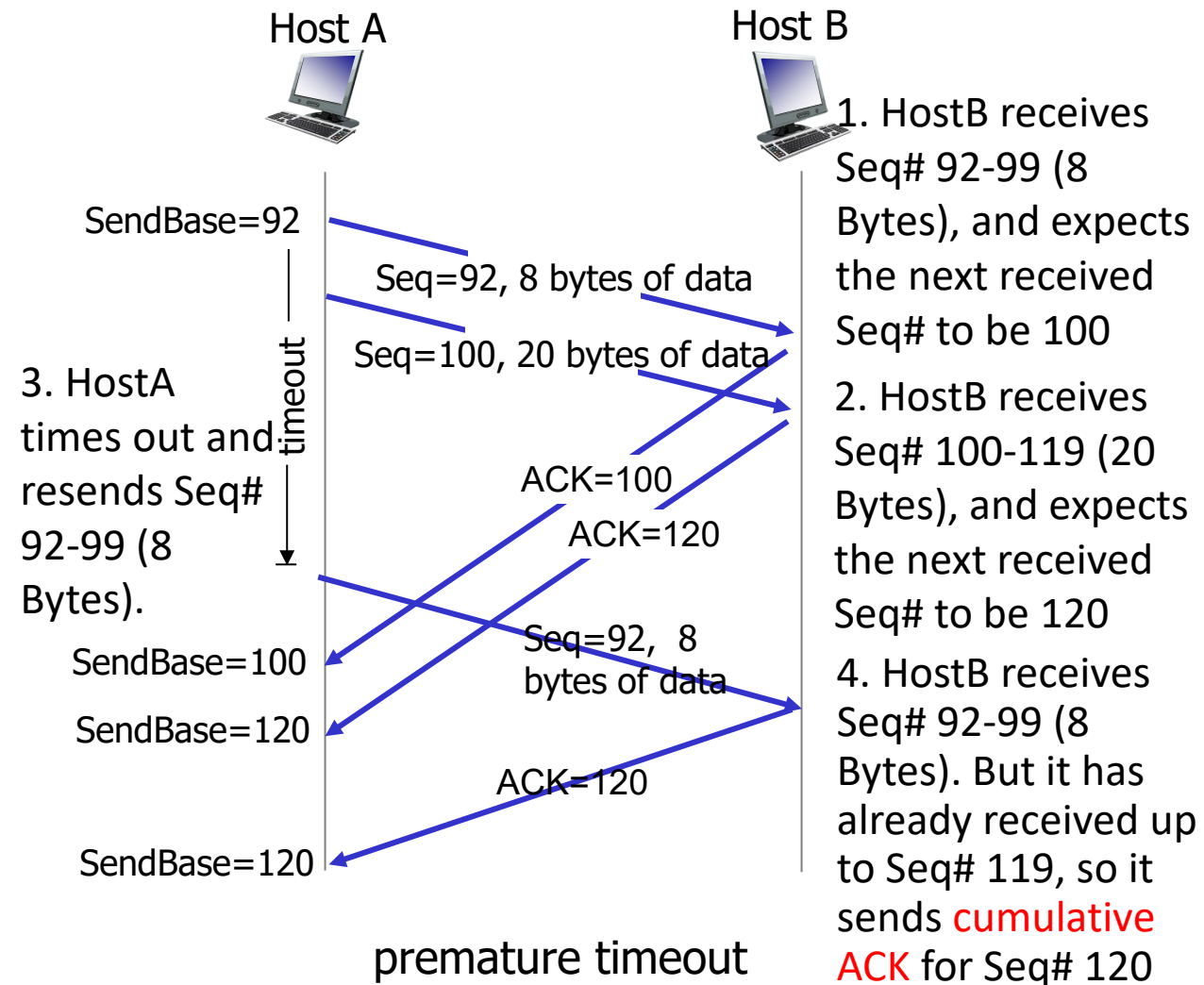
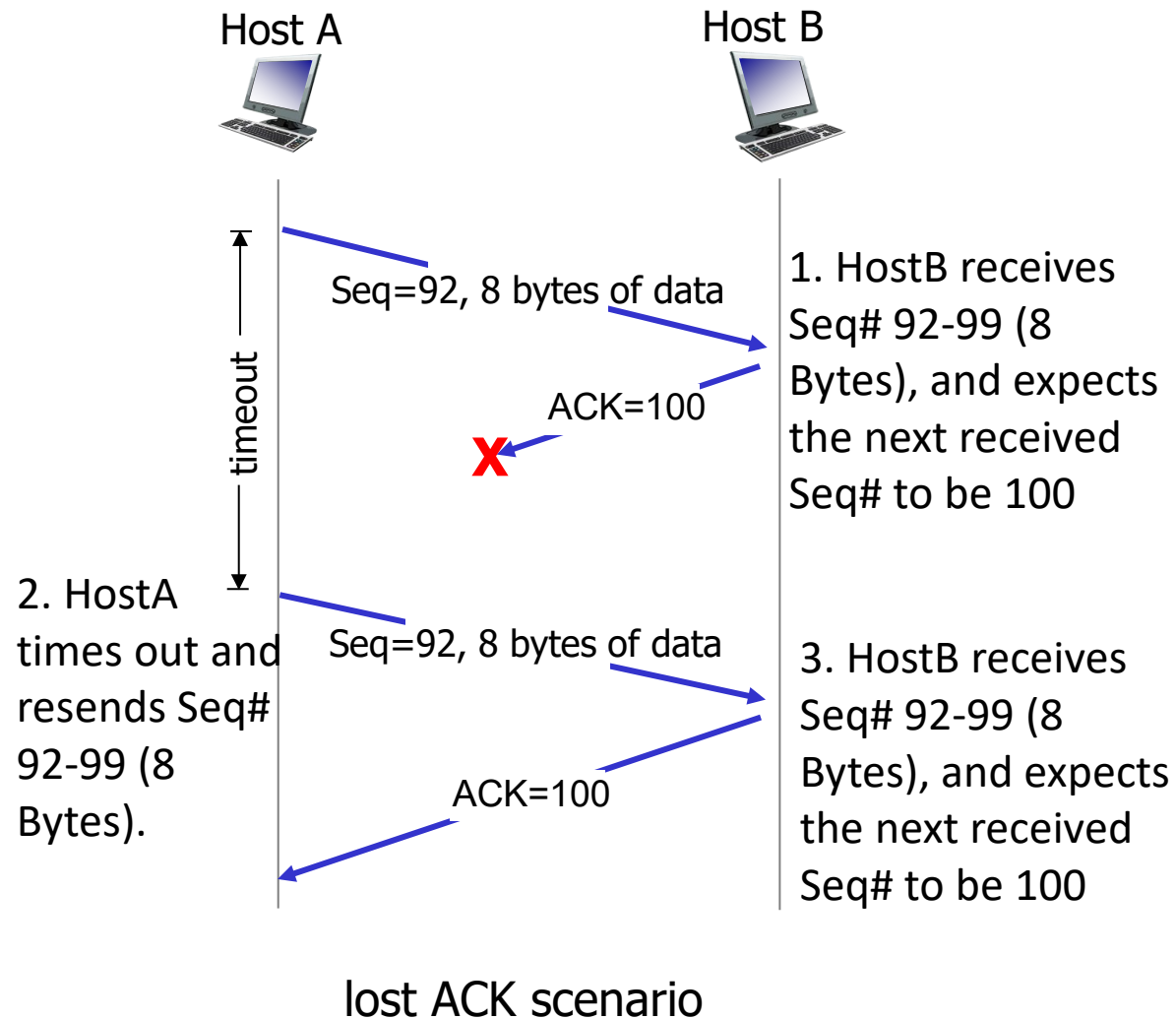


Receiver ACK=N means that "I have received all Bytes up to sequence#N-1, and I am expecting the next Byte I receive to have Seq # N".

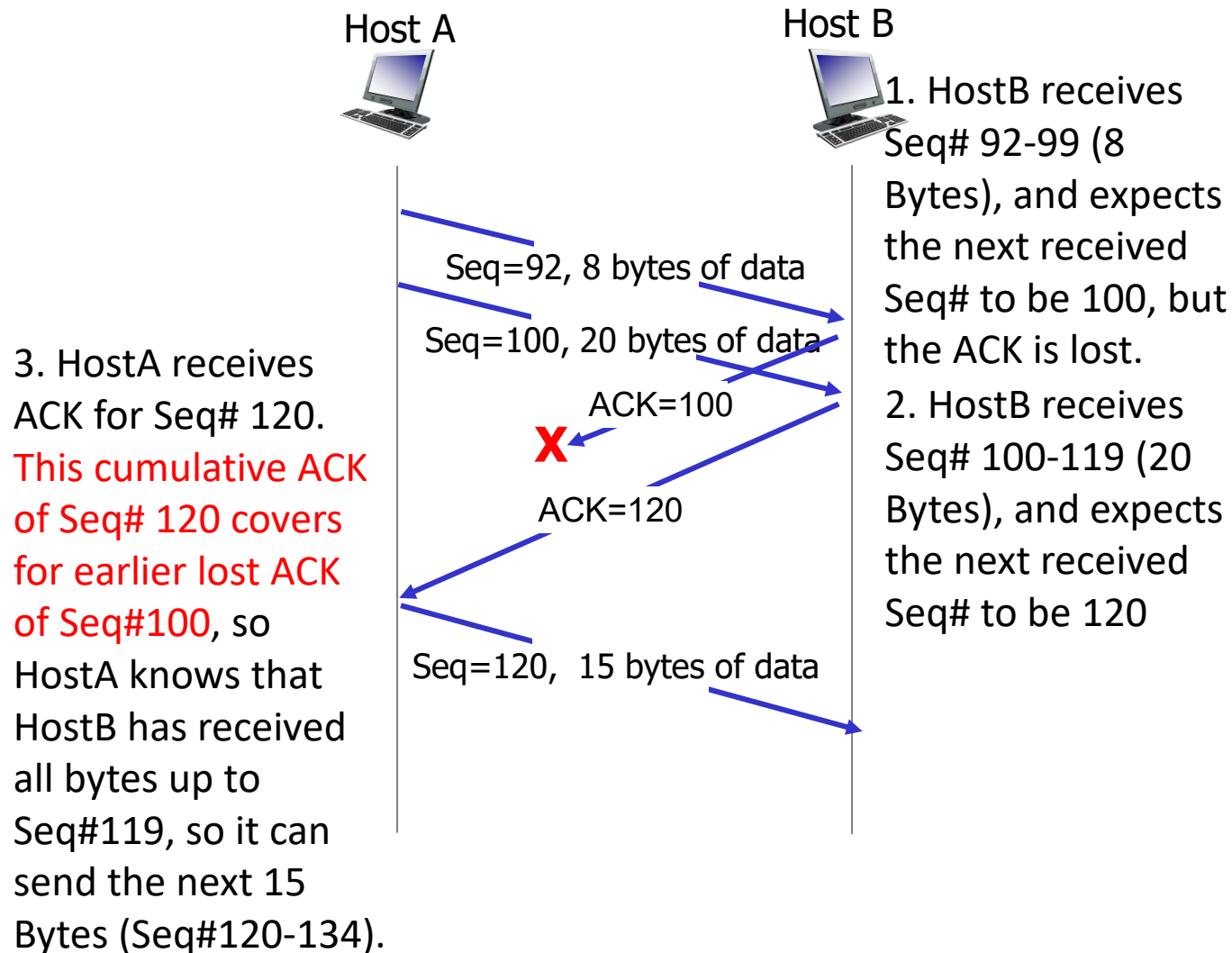
TCP sequence numbers, ACKs



TCP: retransmission scenarios

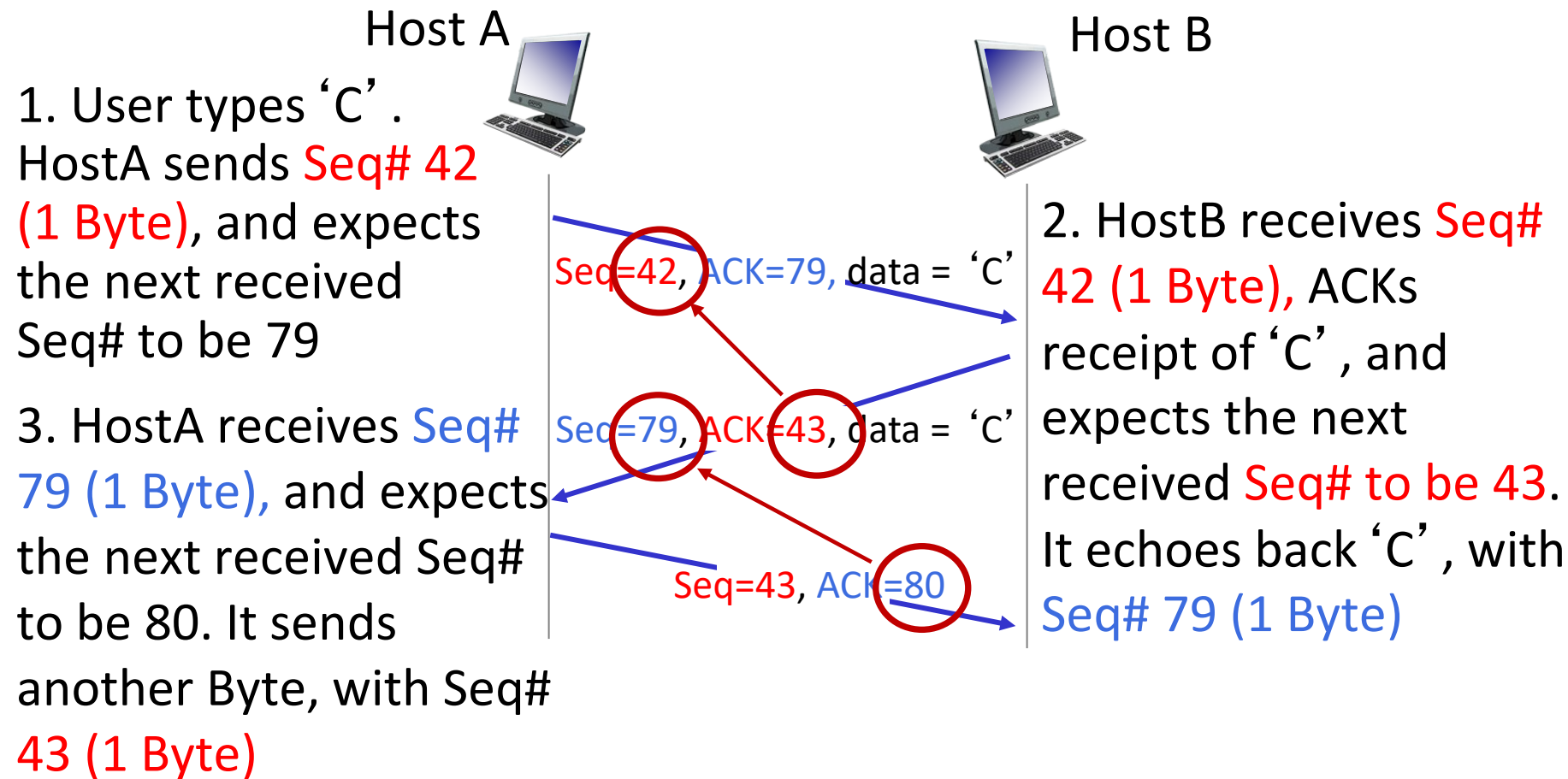


TCP: retransmission scenarios



- Q: what happens if the segment with Seq=92, 8 bytes of data from Host A to Host B gets lost?
- A: Host B will NOT send ACK=120, since a cumulative ACK=120 implies that all previous segments with Seq < 120 have been received

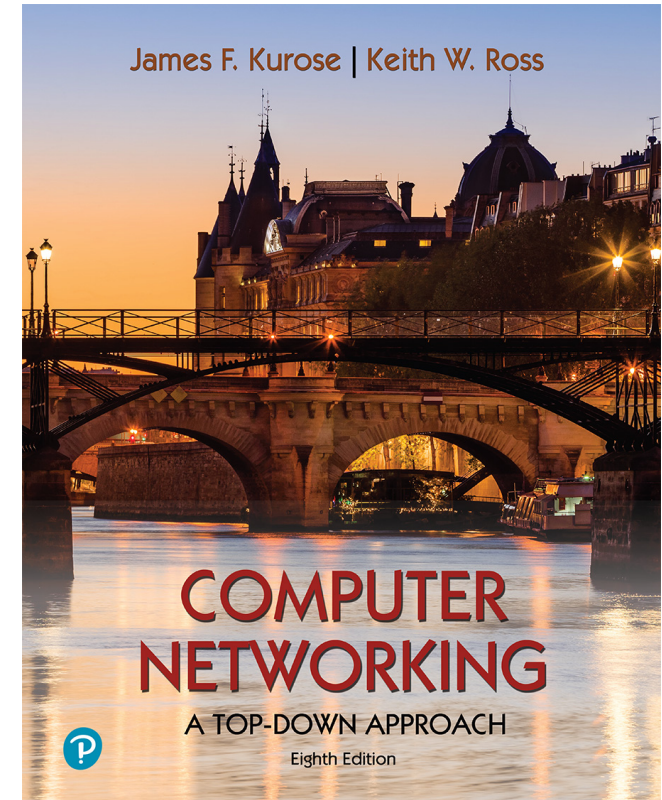
TCP sequence numbers, ACKs



simple telnet scenario

Chapter 4

Network Layer: Data Plane



Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

Packet Scheduling: FCFS

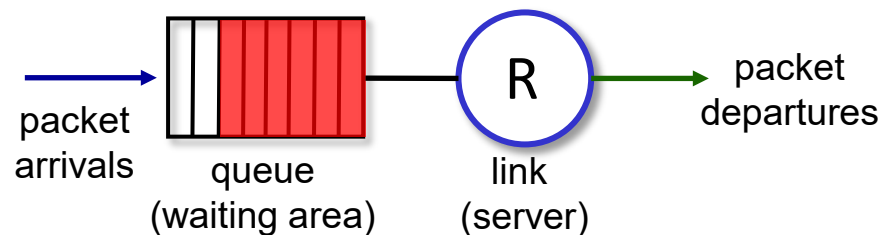
packet scheduling: deciding which packet to send next on link

- 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?

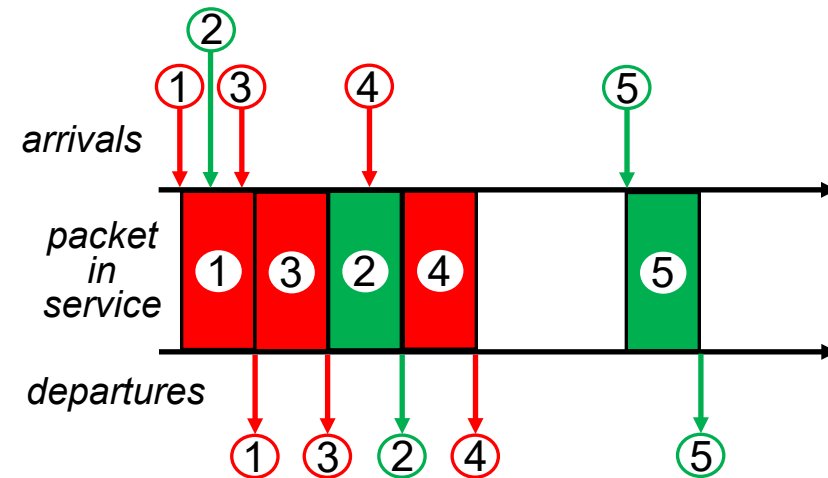
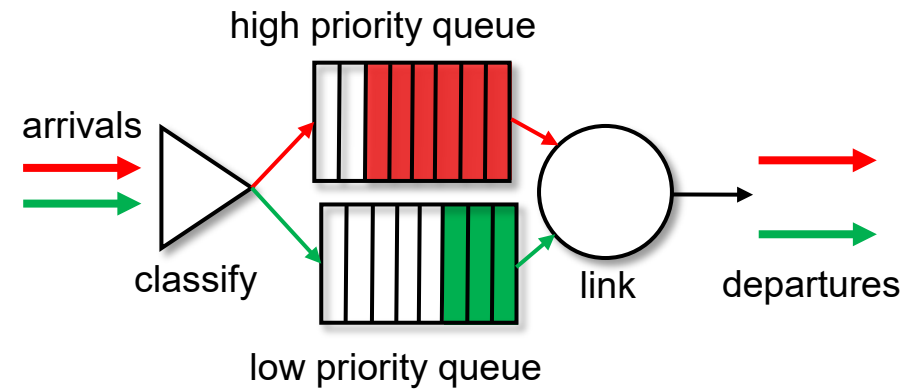
Abstraction: queue



Scheduling policies: priority

Priority scheduling:

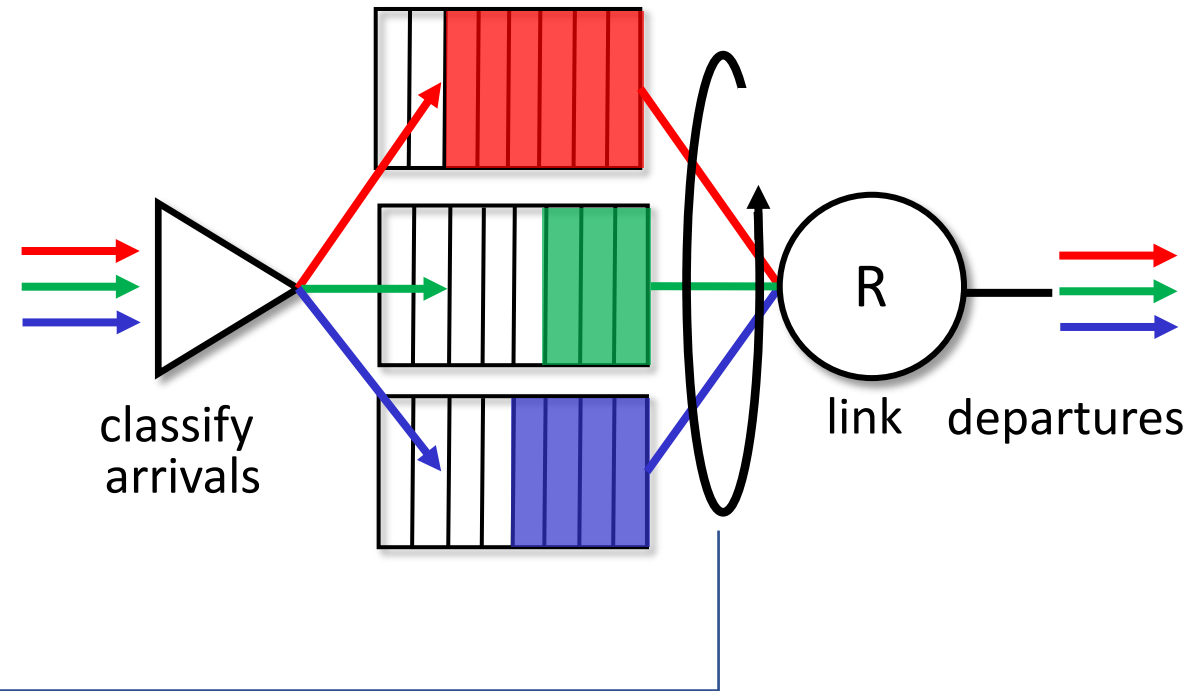
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



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



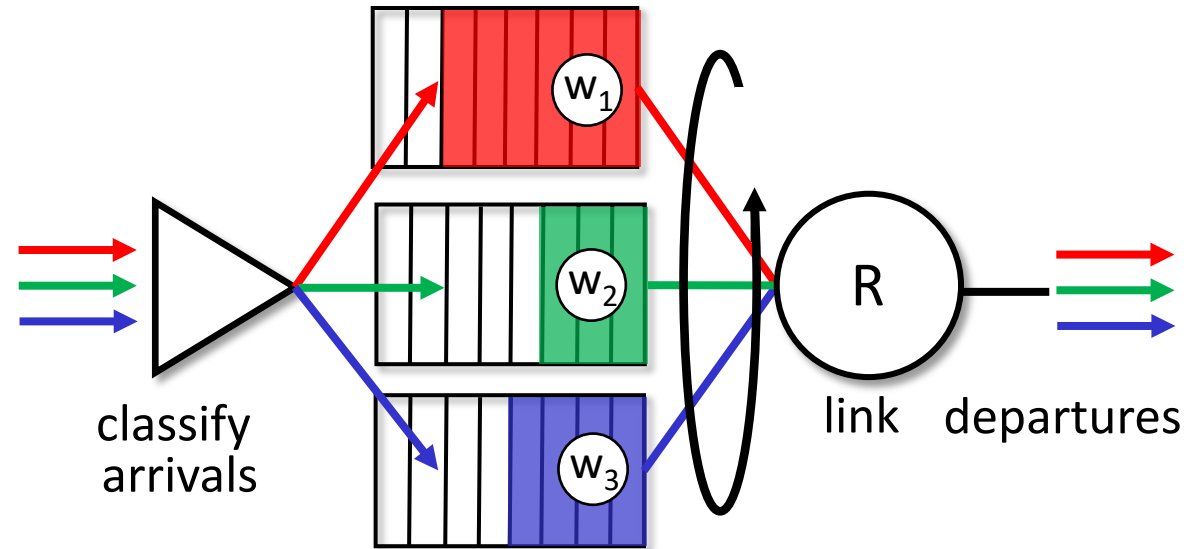
Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

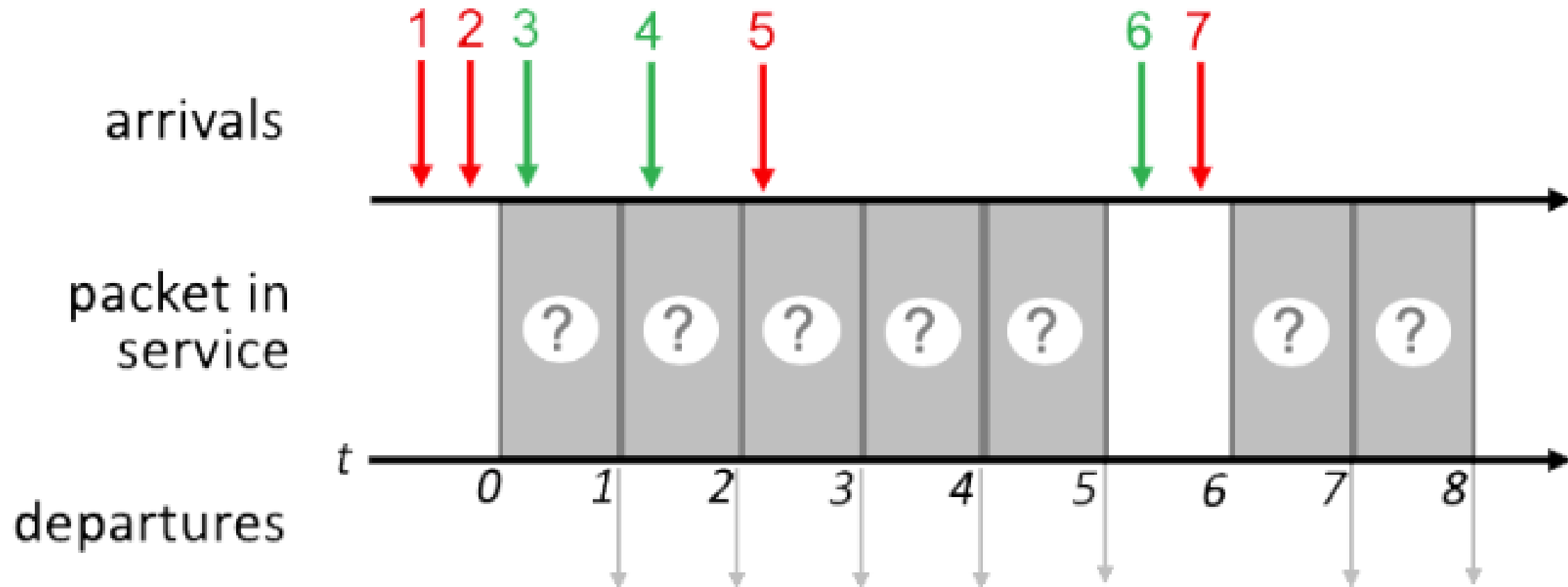
- generalized Round Robin
- each class, i , has weight, w_i , and gets weighted amount of service in each cycle:

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

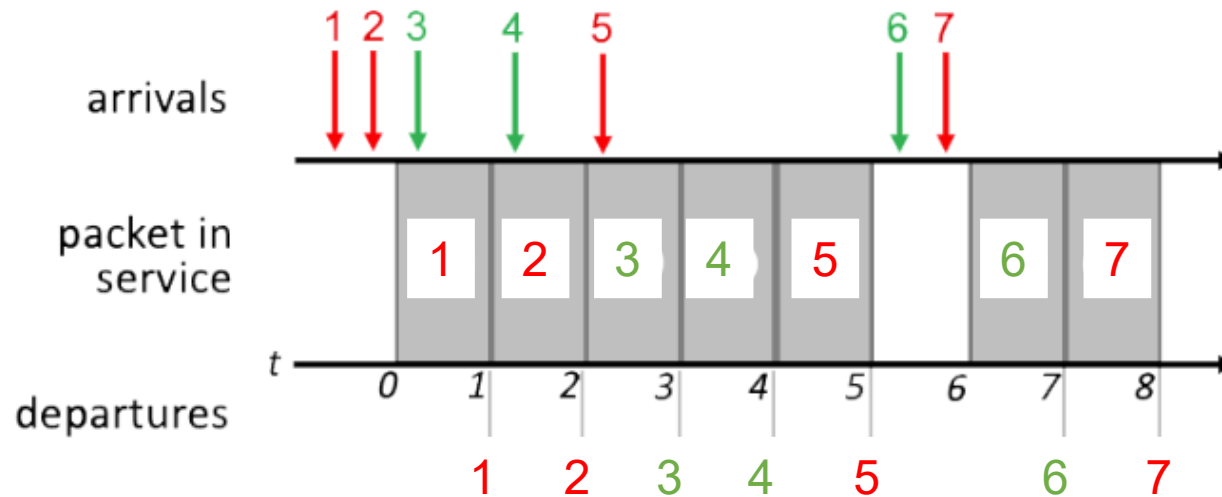
- minimum bandwidth guarantee (per-traffic-class)



Quiz



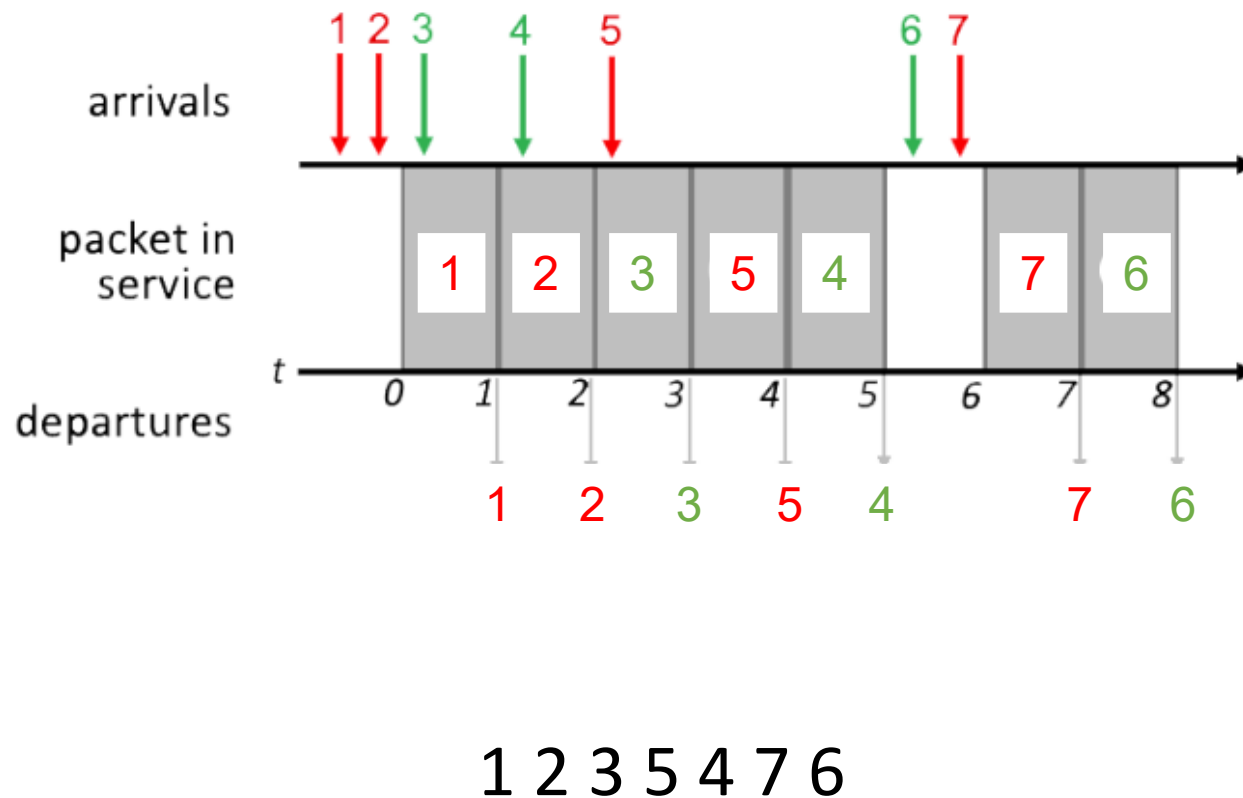
FCFS Scheduling



- Transmit order the same as packet arrival order of 1 2 3 4 5 6 7

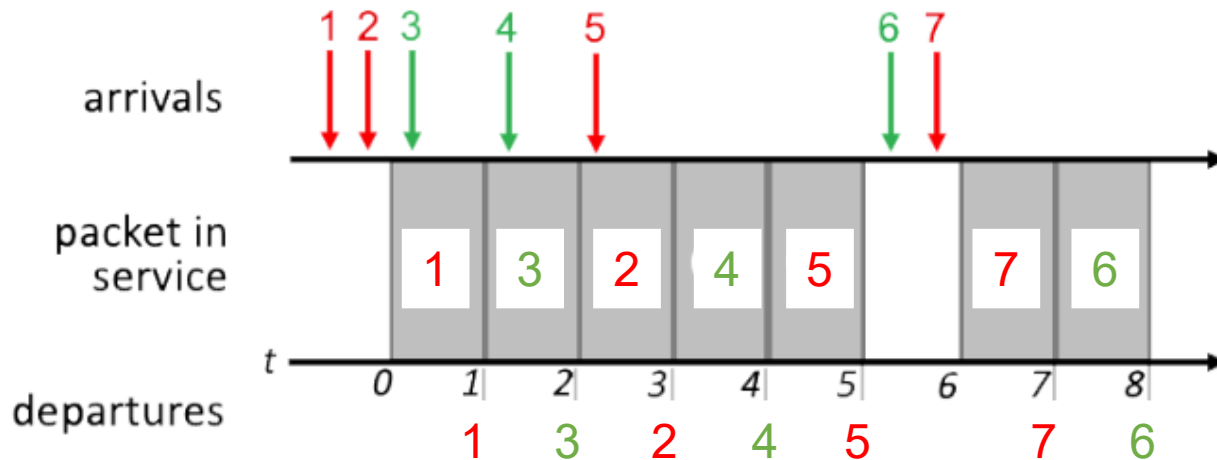
1 2 3 4 5 6 7

Priority Scheduling



- Time 0: 1, 2 in queue, transmit 1
 - FCFS within same priority
- Time 1: 2, 3 in queue, transmit 2
- Time 2: 3, 4 in queue, transmit 3
 - FCFS within same priority
- Time 3: 4, 5 in queue, transmit 5
- Time 4: 4 in queue, transmit 4
- Time 6: 6, 7 in queue, transmit 7
- Time 7: 6 in queue, transmit 6

Round Robin Scheduling

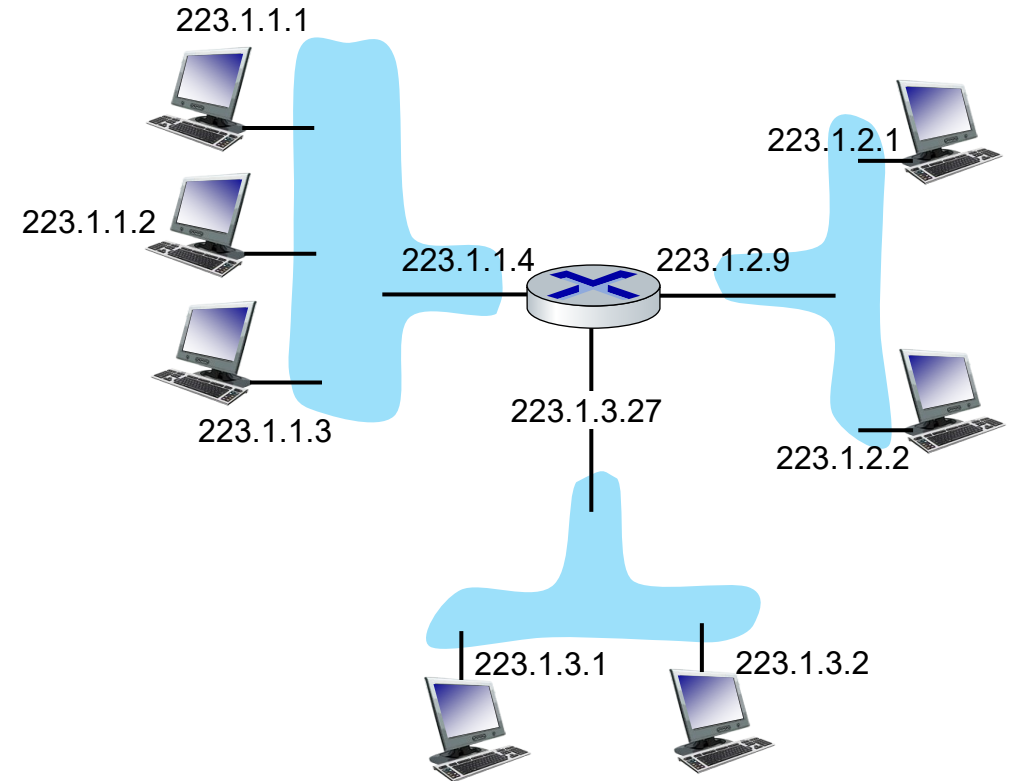


1 3 2 4 5 7 6

- **Red** starts each round if there are both red and green packets ready to transmit after an empty slot, i.e., (**red**, **green**) in each round
- Time 0-1: 1st round: (**1**, **3**)
- Time 2-3: 2nd round: (**2**, **4**)
- Time 4-5: 3rd round: (**5**, **null**)
 - No green packets ready
- Time 6-7: 4th round: (**7**, **6**)

IP addressing: introduction

- **IP address:** 32-bit identifier associated with each host or router *interface*
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



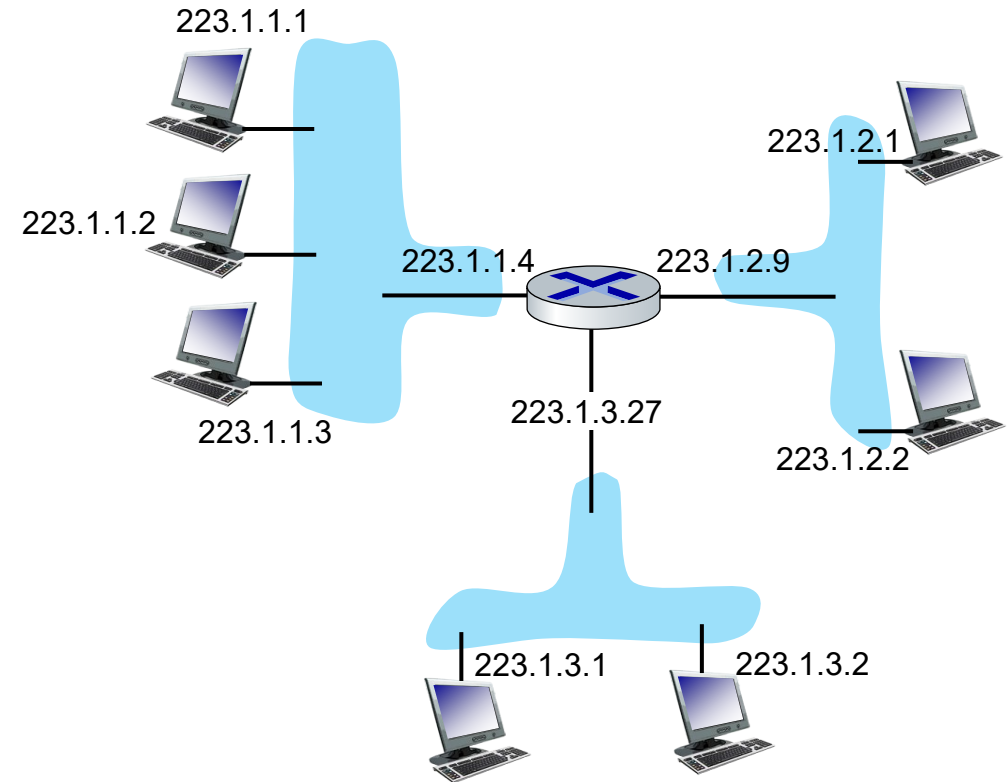
dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223
1
1
1

IP addressing: introduction

- **IP address:** 32-bit identifier associated with each host or router *interface*
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223
1
1
1

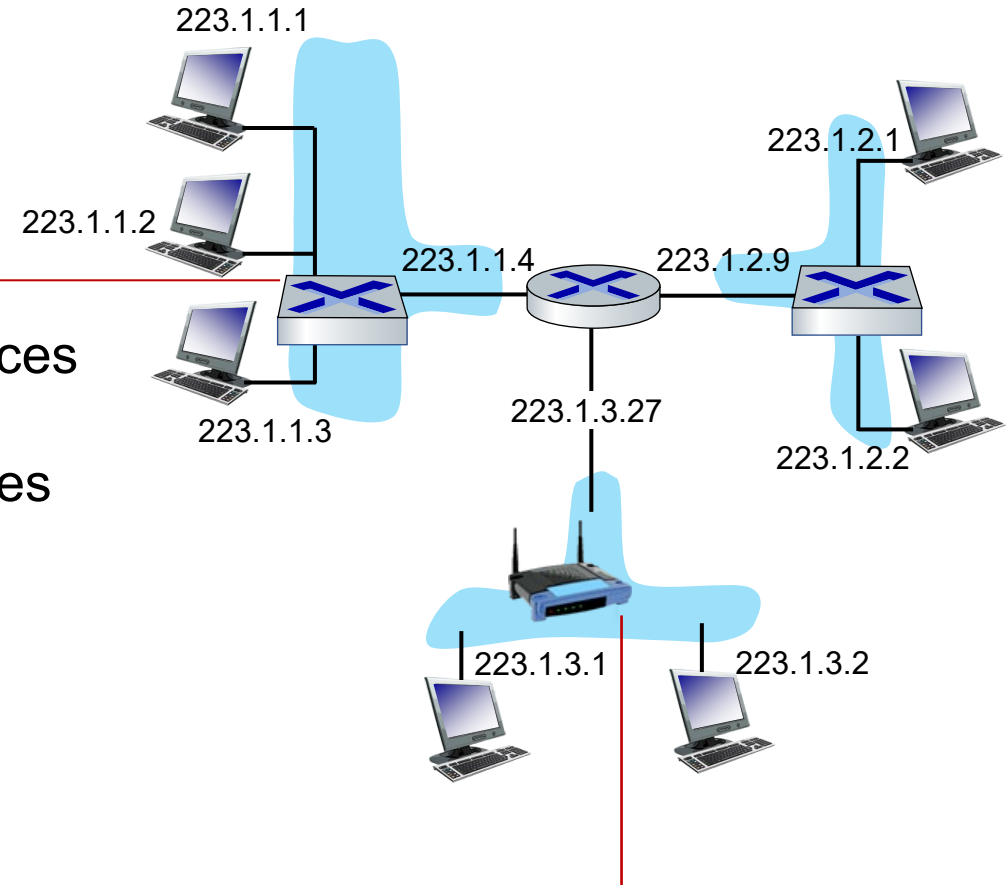
IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapters 6, 7

For now: don't need to worry about how one interface is connected to another (with no intervening router)

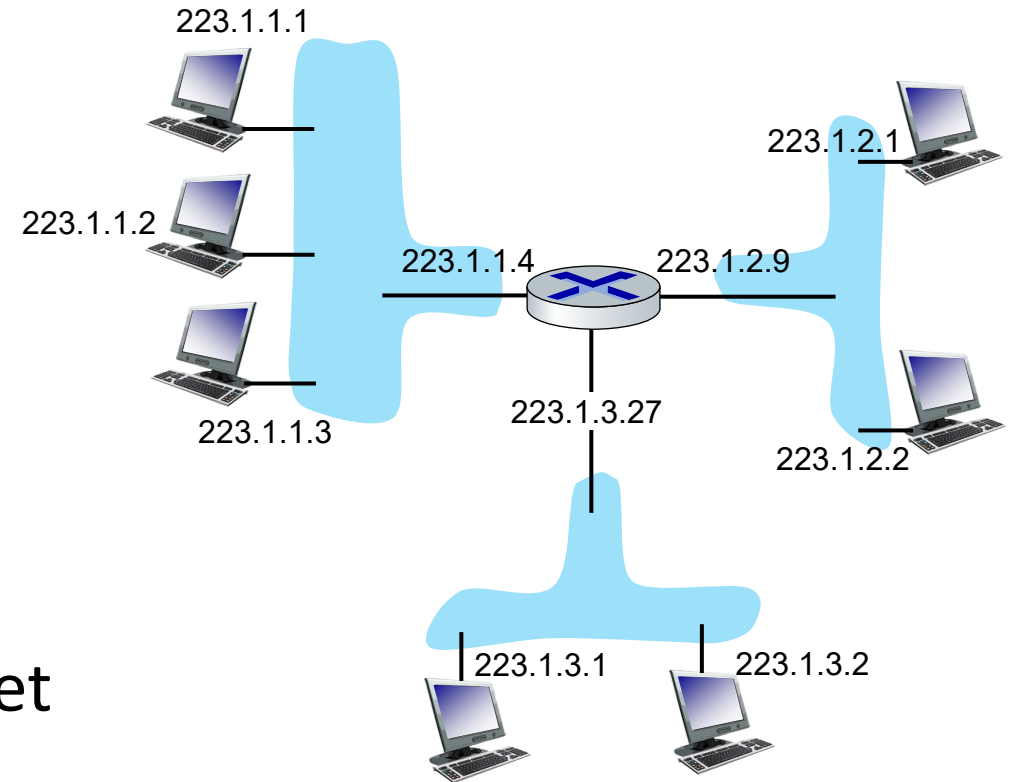
A: wired Ethernet interfaces connected by Ethernet switches



A: wireless WiFi interfaces connected by WiFi base station

Subnets

- *What's a subnet ?*
 - device interfaces that can physically reach each other **without passing through an intervening router**
- IP addresses have structure:
 - **subnet part**: devices in same subnet have common high order bits
 - **host part**: **remaining** low order bits

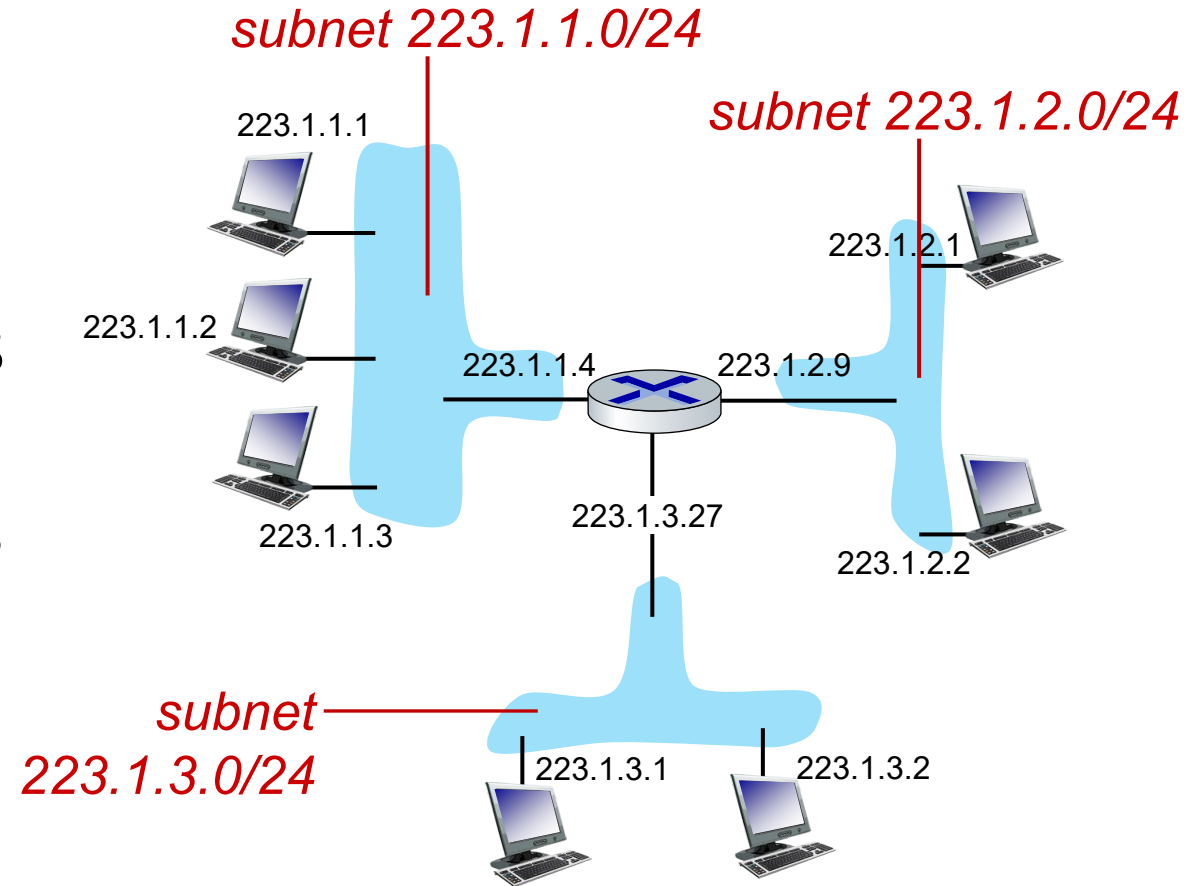


network consisting of 3 subnets

Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*

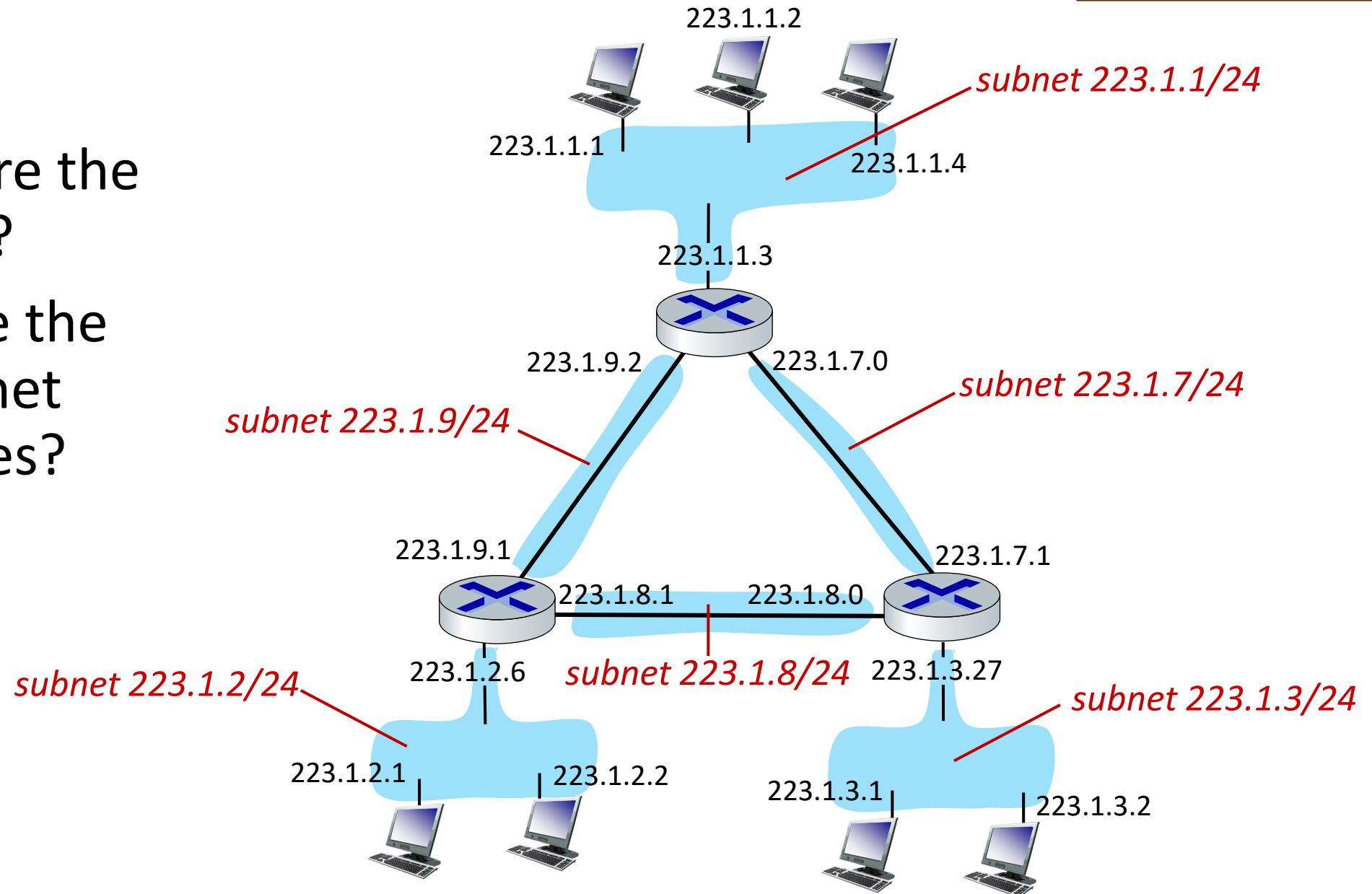


subnet mask: /24

(high-order 24 bits: subnet part of IP address)

Subnets

- where are the subnets?
- what are the /24 subnet addresses?



IP addressing: CIDR

CIDR: Classless **I**nter**D**omain **R**outing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address

