

# Digital Communication in the Modern World

## First Semester 2005-6

<http://www.cs.huji.ac.il/~com1>  
[com1@cs.huji.ac.il](mailto:com1@cs.huji.ac.il)

*Some of the slides have been borrowed from:*  
*Computer Networking: A Top Down Approach Featuring the Internet,*  
3<sup>rd</sup> edition.  
Jim Kurose, Keith Ross  
Addison-Wesley, 2005.

# Administrative Matters

## Reception hours:

- ❑ Teacher - Danny Bickson: Sunday 11:00-12:00 at Ross 107,  
Phone: 85706  
email: daniel51@cs
- ❑ Assistant - Ariel Daliot: Thursday 13:00-14:00 at Ross 106,  
Phone: 85770  
email: com1@cs or adaliot@cs



# Course Policy and Grading

- This is an elective course giving a credit of 4 points.
- There will be 3 practical exercises and 2 theoretical exercises. Two of the practical exercise must be done in Java, one in C/C++.
- You can write the exercises on whatever platform you choose but they must be able to run on Linux.
- All the exercises can be done in pairs except Ex 1.
- All exercises submission is mandatory. Students that will get less than 55 in the exercises will not be able to take the exam.

# Course Policy and Grading

- The exercises will comprise 60% of the final grade. The final exam will be 40%.
- The weight of each exercise is published in the course home page
- Late submission will result in 3 points penalty per day (not including Friday/Saturday).

# Reserve Duty

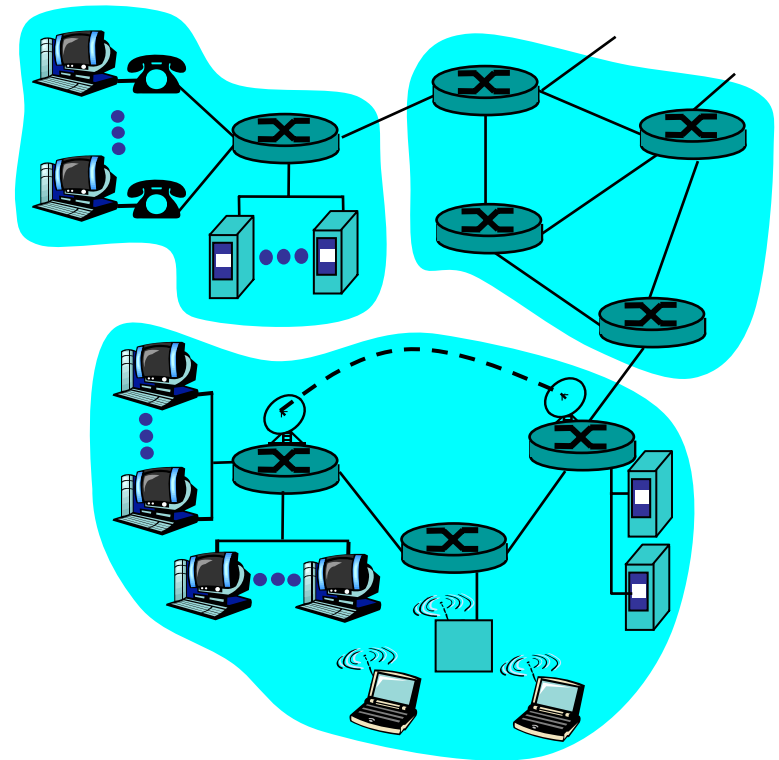
- Students that are in reserve duty for less than three weeks during the semester will be given an appropriate extension for one or more home exercises.
- Students that are in reserve duty for more than three weeks during the semester will be released from one home exercise. Additionally an extension to the other exercises will be given if necessary.

# Exercise submission

- The practical exercises will be submitted electronically.
- For each project create a tar file called *ex#.tar* (i.e. *ex1.tar*). Submit it from the administrative web page.
- Every submitted programming exercise (not the theoretical) automatically gets 3 bonus points for not appealing.

# A general look at network structure

- **network edge:**  
applications and hosts
- **network core:**
  - routers
  - network of networks
- **physical media:**  
communication links



# A Network vs. A Network Algorithm

- Network (Distributed) Algorithm: the steps performed for solving a problem between elements that communicate
- The network is there to serve the network algorithm
- A “good” network vastly eases the design of network algorithms
- The communication properties can make the difference between being able to solve a problem easily and not being able to solve it at all



# Human Analogy

- Similarly, communication between people is a mean or tool for a higher purpose
- Two individual groups wanting to settle an age-old dispute can be greatly facilitated by “good” communication



# Networking Course Objectives

- The “language” that is spoken between two network elements is called a **Protocol** (E.g. TCP/IP, UDP/IP, HTTP, FTP, PPP, ...)
- In this course we focus on the network core and its protocols and not on the distributed algorithms that use the network (*the applications*)
- I.e. focus more on how to efficiently communicate and less on how to efficiently solve problems through communication

# Understanding networking by understanding the Internet

# The Network Edge

## Client-Server paradigm

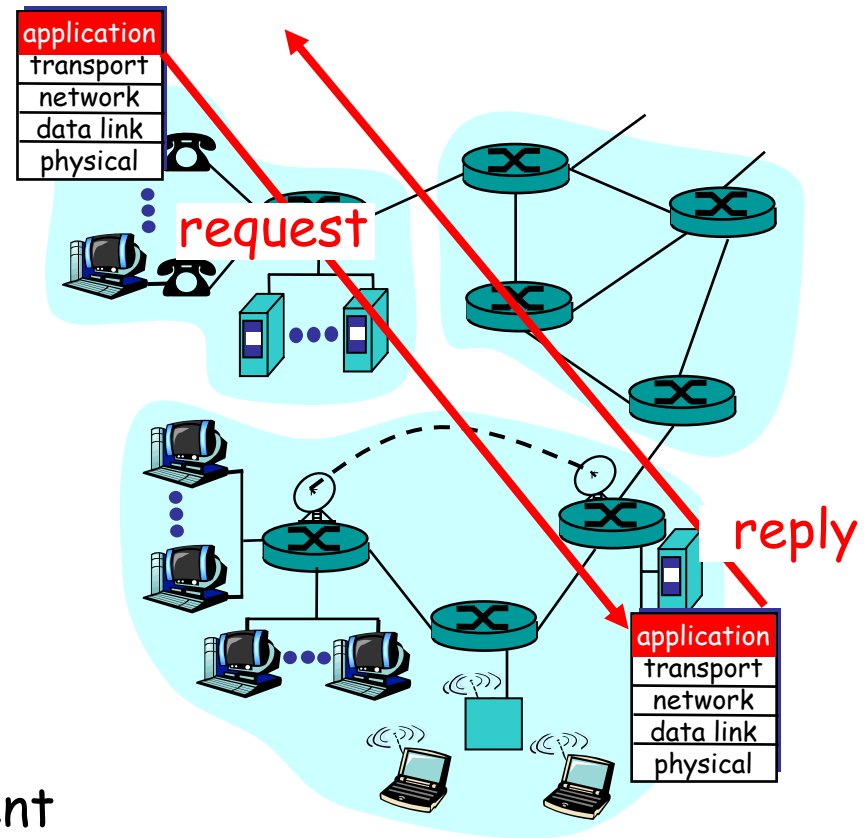
Typical network app has two pieces: *client* and *server*

### Client:

- initiates contact with server ("speaks first")
- typically requests service from server,
- Web: client implemented in browser; e-mail: in mail reader

### Server:

- provides requested service to client
- Web: server sends requested Web page; e-mail: server delivers e-mail



# Classes of protocols between hosts

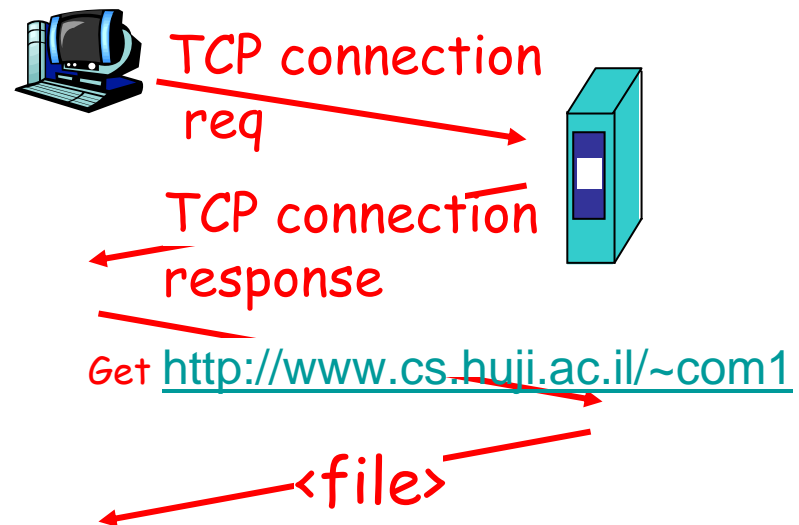
# Connectionless/Connection-oriented protocols

- **Connection-oriented:** The Internet's connection-oriented protocol is **TCP**.

a human protocol



computer network protocol



# TCP - Transmission Control Protocol [RFC 793]

- **Reliable.** In-order byte-stream data transfer. Loss handled by acknowledgements and retransmissions.
- **Flow control.** Sender will not overwhelm receiver.
- **Congestion control.** Senders "slow down" sending rate when network congested.
- Apps using TCP: HTTP (WWW), FTP (File transfer protocols), Telnet (remote login), SMTP (email)

# Connectionless/Connection-oriented protocols

- **Connectionless**

- When someone is making a speech or reading the news on radio; sending e-mails
- The Internet's connectionless protocol is **UDP**



# UDP – User Datagram Protocol

## [RFC 768]

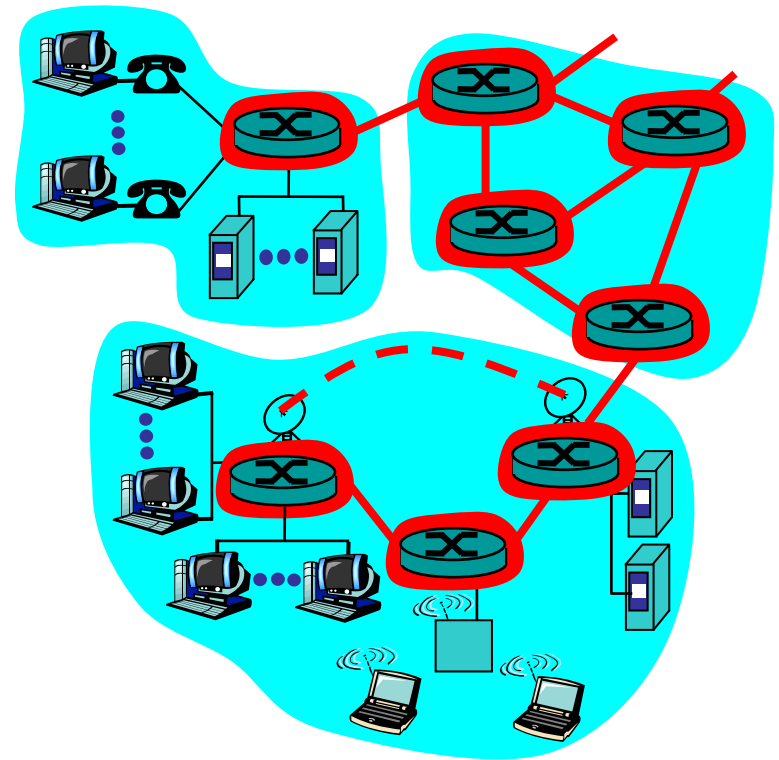
- Unreliable data transfer
- No flow control
- No congestion control
- Apps using UDP: streaming media (real-player), teleconferencing, Internet telephony

# The Network Core

## How is data transferred through net?

# The Network Core

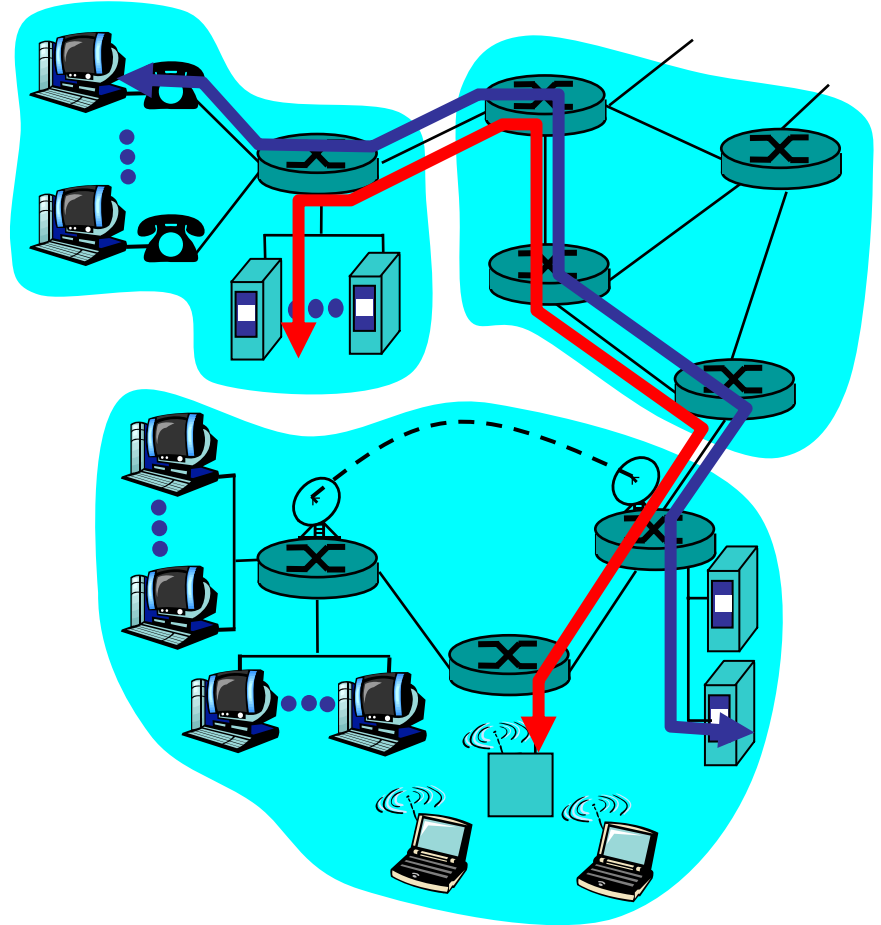
- mesh of interconnected routers
- the fundamental question: how is data transferred through net?
  - **circuit switching**: dedicated circuit per call: telephone net
  - **packet-switching**: data sent through net in discrete "chunks"



# Network Core: Circuit Switching

End-to-end resources reserved for "call"

- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required



# Network Core: Circuit Switching

network resources  
(e.g., bandwidth)

divided into  
"pieces"

- pieces allocated to calls
- resource piece *idle* if not used by owning call (*no sharing*)

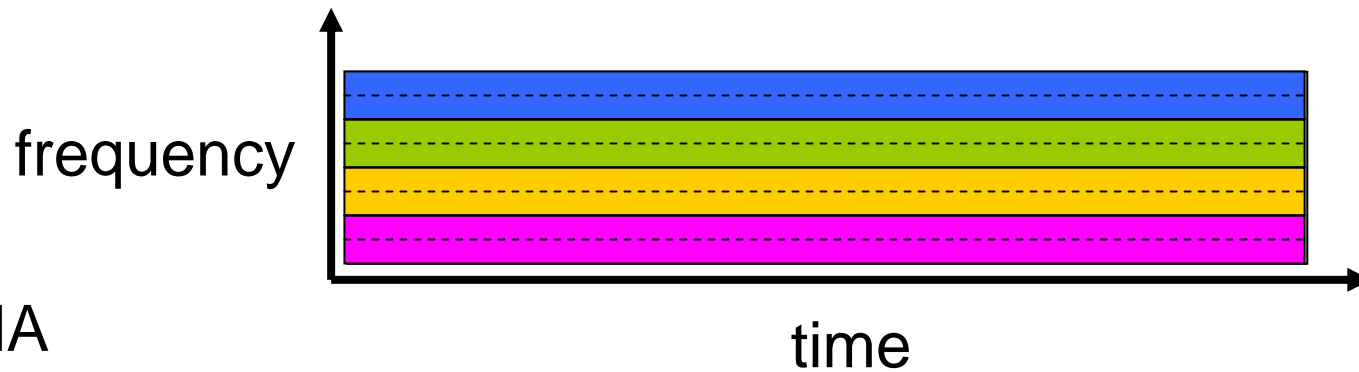
- dividing the link bandwidth into "pieces" through
  - frequency division
  - time division

# Circuit Switching: FDMA and TDMA

FDMA

Example:

4 users

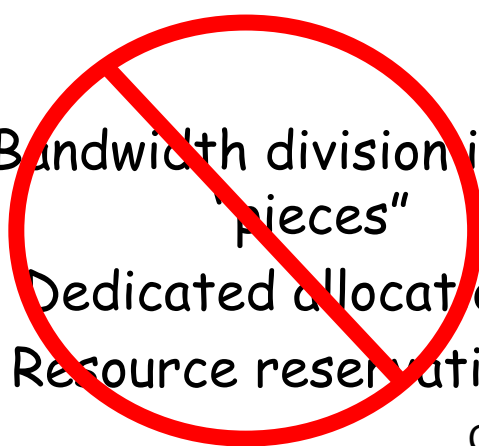


# Network Core: Packet Switching

each end-end data stream  
divided into *packets*

- user A, B packets *share* network resources
- each packet uses full link bandwidth
- resources used *as needed*

Bandwidth division into  
"pieces"  
Dedicated allocation  
Resource reservation



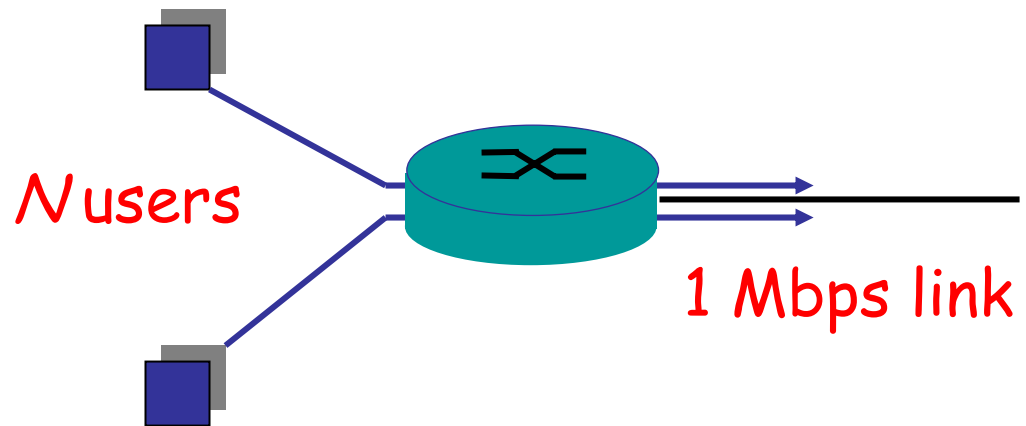
resource contention:

- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time
  - transmit over link
  - wait turn at next link

# Packet switching versus circuit switching

Packet switching allows more users to use network!

- 1 Mbit link
- each user:
  - uses 100 kbps when "active"
  - active 10% of time
- circuit-switching:
  - 10 users
- packet switching:
  - with 35 users, probability > 10 active less than 0.0004





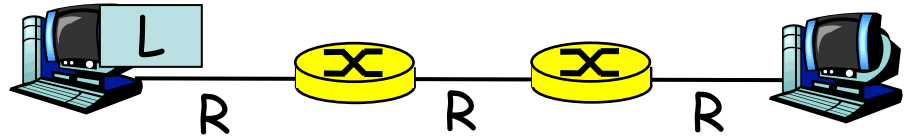
# Packet switching versus circuit switching

## Is packet switching a winner?

- Great for bursty data
  - resource sharing
  - simpler, no call setup
- **Excessive congestion:** packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- **Q: How to provide circuit-like behavior?**
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (chapter 6 - Kurose)

# Packet-switching: store-and-forward

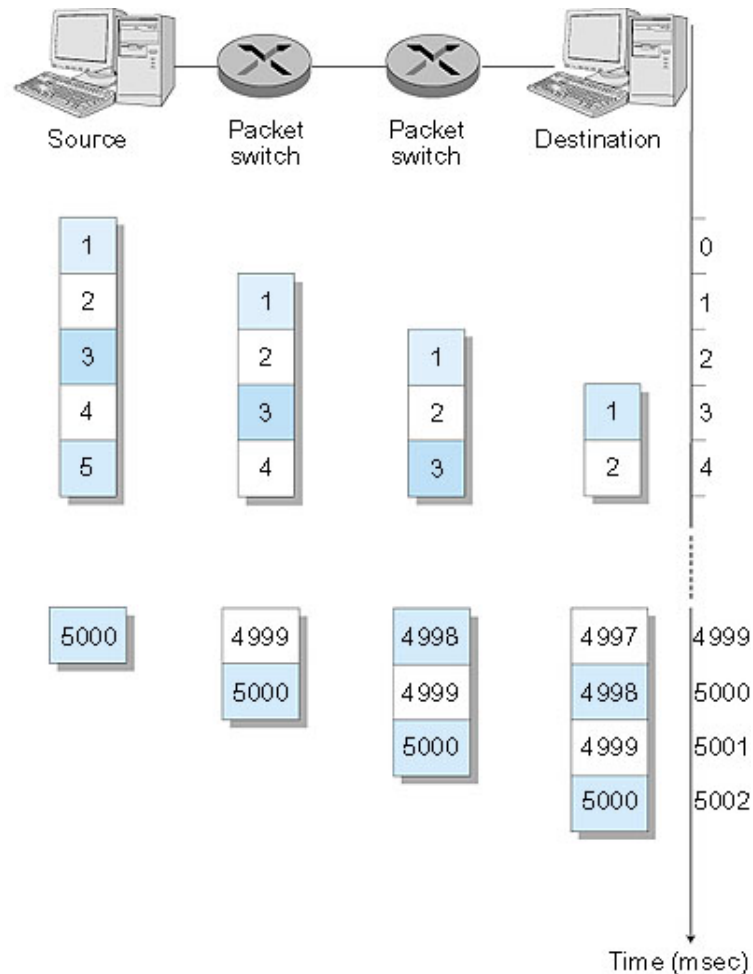
- Entire packet must arrive at router before it can be transmitted on next link: *store and forward*
- Takes  $L/R$  seconds to transmit (push out) packet of  $L$  bits on to a link of  $R$  bps
- $\text{delay} = 3L/R$



## Example:

- $L = 7.5$  Mbits
- $R = 1.5$  Mbps
- ➔  $\text{delay} = 15$  sec

# Packet Switching: Message Segmenting



Now break up the message into 5000 packets

- Each packet 1,500 bits
- 1 msec to transmit packet on one link
- *pipelining*: each link works in parallel
- Delay reduced from 15 sec to 5.002 sec

# Reliability of Data Transfer

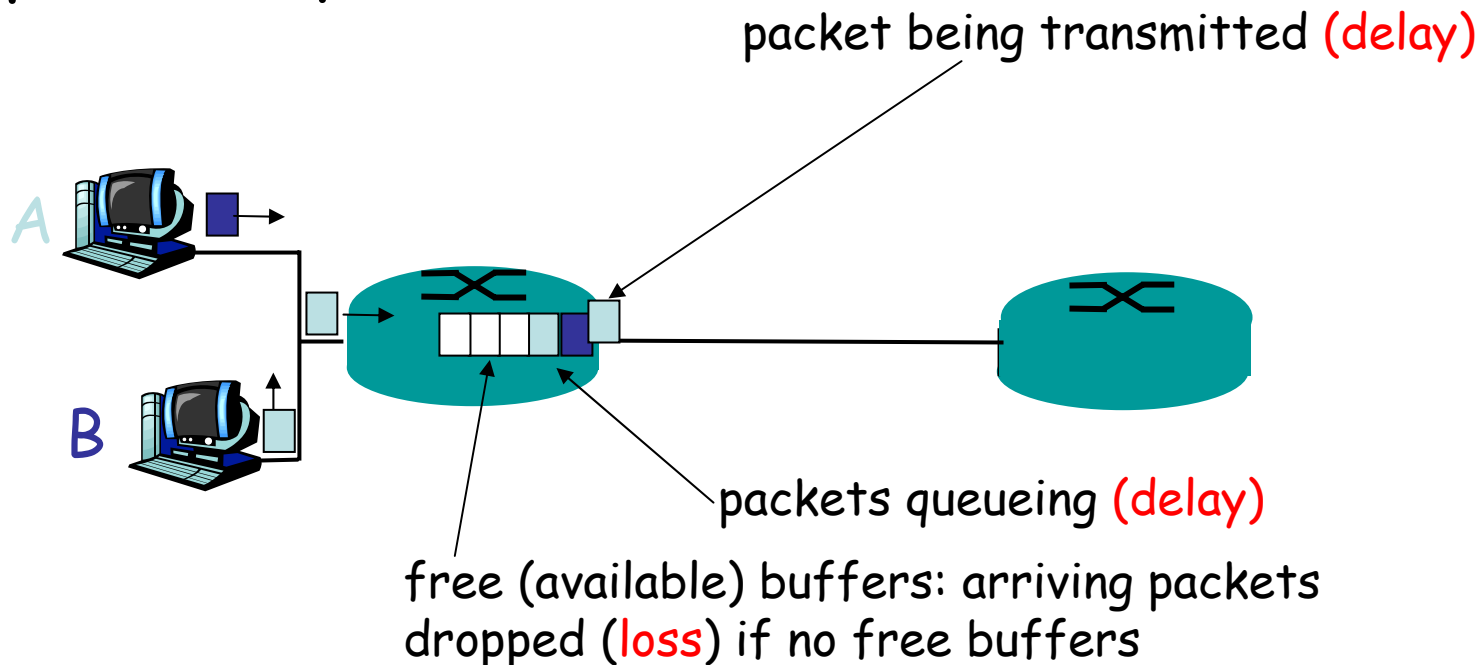
## Delay and Loss



# How do loss and delay occur?

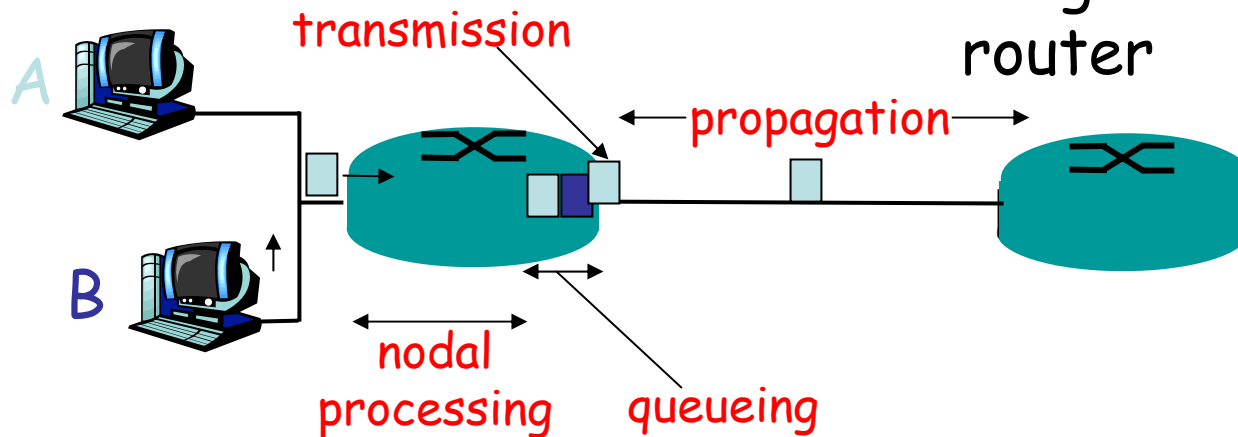
packets *queue* in router buffers

- packet arrival rate to link exceeds output link capacity
- packets queue, wait for turn



# Four sources of packet delay

- 1. nodal processing:
  - check bit errors
  - determine output link
- 2. queuing
  - time waiting at output link for transmission
  - depends on congestion level of router



# Delay in packet-switched networks

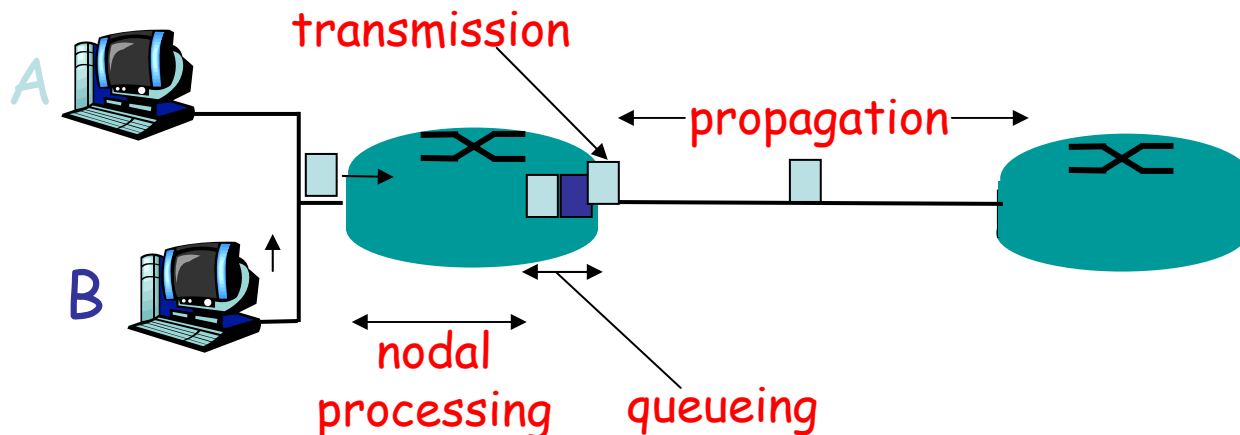
## 3. Transmission delay:

- $R$  = link bandwidth (bps)
- $L$  = packet length (bits)
- time to send bits into link =  $L/R$

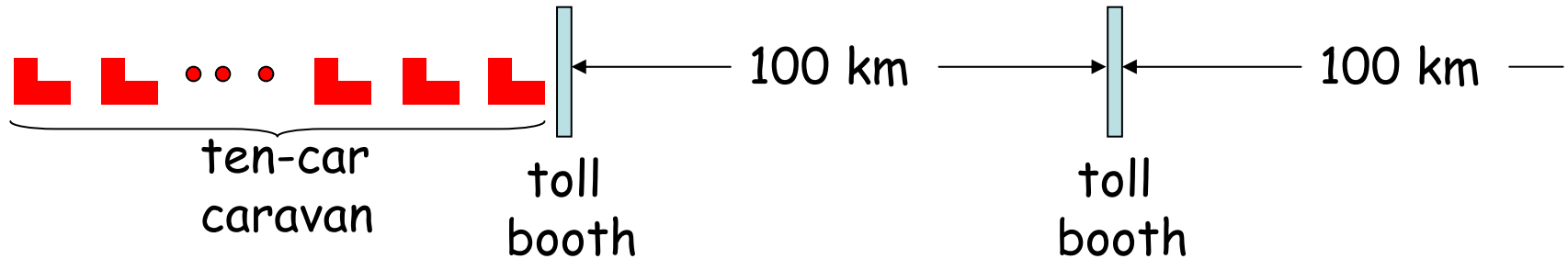
## 4. Propagation delay:

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

**Note:**  $s$  and  $R$  are very different quantities!



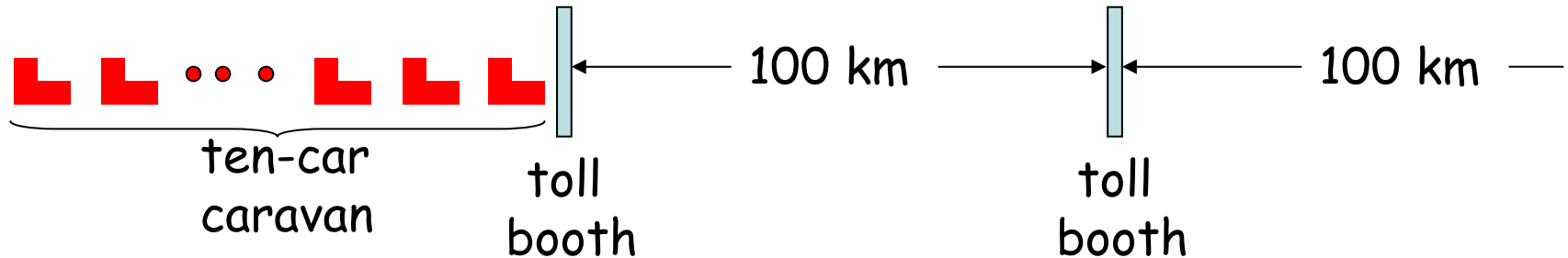
# Caravan analogy



- Cars "propagate" at 100 km/hr
- Toll booth takes 12 sec to service a car (transmission time)
- car~bit; caravan ~ packet
- Q: How long until caravan is lined up before 2nd toll booth?
- Time to "push" entire caravan through toll booth onto highway =  $12 \times 10 = 120$  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



# Caravan analogy (more)



- Cars now "propagate" at 1000 km/hr
- Toll booth now takes 1 min to service a car
- **Q: Will cars arrive to 2nd booth before all cars serviced at 1st booth?**
- **Yes!** After 7 min, 1st car at 2nd booth and 3 cars still at 1st booth.
- 1st bit of packet can arrive at 2nd router before packet is fully transmitted at 1st router!

# Nodal delay

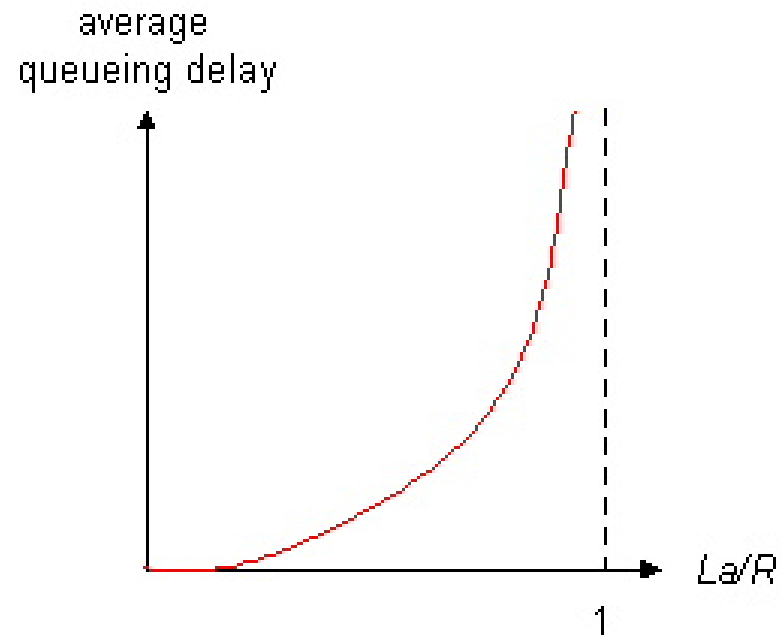
$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

- $d_{\text{proc}}$  = processing delay
  - typically a few microsecs or less
- $d_{\text{queue}}$  = queuing delay
  - depends on congestion
- $d_{\text{trans}}$  = transmission delay
  - $= L/R$ , significant for low-speed links
- $d_{\text{prop}}$  = propagation delay
  - a few microsecs to hundreds of msecs

# Queueing delay (revisited)

- $R$ =link bandwidth (bps)
- $L$ =packet length (bits)
- $a$ =average packet arrival rate

traffic intensity =  $L \cdot a / R$



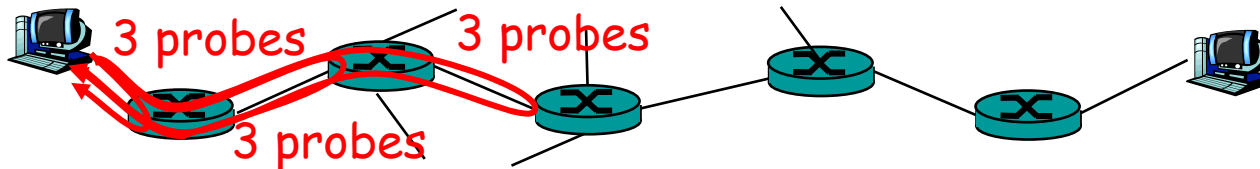
- $La/R \sim 0$ : average queueing delay small
- $La/R \rightarrow 1$ : delays become large
- $La/R > 1$ : more "work" arriving than can be serviced, average delay infinite!

# Packet loss

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

# "Real" Internet delays and routes


- What do "real" Internet delay & loss look like?
- Traceroute program: provides delay measurement from source to router along end-to-end Internet path towards destination. For all  $i$ :
  - sends three packets that will reach router  $i$  on path towards destination
  - router  $i$  will return packets to sender
  - sender times interval between transmission and reply.



# "Real" Internet delays and routes


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

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



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

trans-oceanic  
link



\* means no reponse (probe lost, router not replying)

