

# Lecture 1: Introduction

COMP 411, Fall 2022

Victoria Manfredi

W E S L E Y A N  
U N I V E R S I T Y



**Acknowledgements:** materials adapted from Computer Networking: A Top Down Approach 7<sup>th</sup> edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University and some material from Computer Networks by Tannenbaum and Wetherall.

# Today

## 1. Announcements

- Homework 1 out due Fri. Sept. 16 at 5p.

## 2. Administivia

## 3. Computer networks

- overview

## 4. Building a network

- how to connect devices
- how to connect processes on devices
- how to share resources

# **Administrivia**

# Course webpage (*not* moodle)

## Everything posted here

- <http://vumanfredi.wescreates.wesleyan.edu/teaching/comp411-f22/>

## We'll use Google classroom for announcements, discussion, maybe grades

- I will add you via email

## Grade breakdown

- 20%: project
- 80%: 10 homework assignments, no scores dropped
  - mix of written and (multi-assignment) programming projects

## Late days

- 4 free days, use at most 2 for any assignment
- Once used, you will lose 15% of grade for each 24 hours late

# Getting started

## Python3

- we'll review as needed, see class resources webpage
  - please check you have python3 **installed!**
    - type python3 at terminal prompt
  - **tutorials** and other resources posted on course website

## Python help available

- at SCIC on 1<sup>st</sup> floor of Exley

## vim and python

- create a .vimrc file in your home directory
- put lines in block in .vimrc and save it
- open new terminal and use vim
  - should see color, line numbers, etc.

```
syntax on
filetype indent plugin on
set modeline
set number
autocmd BufWritePre * %s/\s\+$/ei
au BufNewFile,BufRead *.py
\ set tabstop=4
\ set softtabstop=4
\ set shiftwidth=4
\ set textwidth=79
\ set expandtab
\ set autoindent
\ set fileformat=unix
```

# Homework

## 1<sup>st</sup> homework is posted

- warm-up homework: implement tic-tac-toe in python
- 2<sup>nd</sup> homework is to implement distributed tic-tac-toe using sockets

## We'll use Google drive for homework submissions

- Each of you will have directory for this course, with homework subdirectories

## Important!

- put your name **inside** every file!
- file formats: only **.py, pdf, .txt** so my printing script works
  - if I can't print it, I can't grade it :-)
- filename should match what is specified

# Looking forward

## 1<sup>st</sup> few weeks

- high-level overview of components of network
- familiarity with terminology
- covers a lot of material!

## Rest of course

- digging into details of what we talked about in 1<sup>st</sup> few weeks
- will talk about each layer and component in much greater depth
- having had high-level should help give context for details

If you have questions or concerns please come talk to me

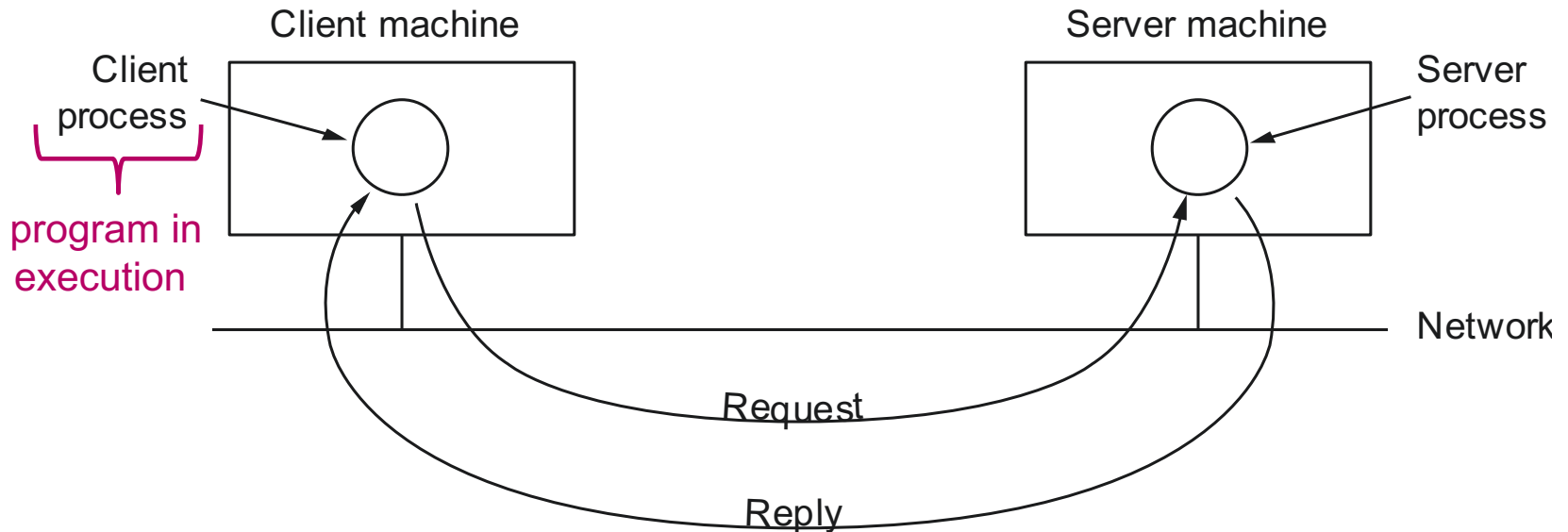
# Computer Networks

## **OVERVIEW**



# What's a computer network?

2 or more **computing devices** able to exchange data



**Necessary network  
functionality**

1. Specify remote machine
2. Connect to it (possibly some handshaking)
3. Transfer data
4. Close connection

# More on processes

## Process: program in execution

- your machine has many processes running on it

“top” command (or type “ps auxwww” in terminal)

```
Processes: 533 total, 3 running, 530 sleeping, 4091 threads 11:45:39
Load Avg: 1.57, 1.96, 2.44 CPU usage: 14.31% user, 14.31% sys, 71.36% idle
SharedLibs: 196M resident, 46M data, 17M linkedit.
MemRegions: 256976 total, 5317M resident, 135M private, 2227M shared.
PhysMem: 15G used (3423M wired), 1203M unused.
VM: 2492G vsize, 627M framework vsize, 52872168(189) swapins, 55781927(0) swapouts.
Networks: packets: 32240950/23G in, 20824902/2706M out.
Disks: 9478634/359G read, 3501804/297G written.
```

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID
65817	screen captur	0.0	00:00.19	6	4	173	10M	444K	0B	65817	1
65816	screen captur	9.7	00:00.36	3	2	58	2548K	20K	0B	432	432
65814	top	8.8	00:01.98	1/1	0	22	4848K	0B	0B	65814	65807

# Killing processes

## Use “ps” to get process id

- type `ps auxwww | grep NAME`

## Use “kill” to terminate process

- `kill processid`
- `kill -9 processid` // nuclear option: don't let process clean up

```
> python3 tictactoe_full.py
```

```
=====
| TicTacToe Game |
=====
```

```
Enter number of rows in TicTacToe board: Terminated: 15
```

```
> ps auxwww | grep python | grep tictactoe
```

```
vmanfredi 12060 0.0 0.0 2419260 7004 s006 S+ 10:51AM 0:0
0.04 /usr/local/Cellar/python/3.7.0/Frameworks/Python.framework/Versions/3
.7/Resources/Python.app/Contents/MacOS/Python tictactoe_full.py
```

```
vmanfredi@ ~ ($) $
```

```
> kill 12060
```

# Distributed system vs. computer network

## Distributed system

- software system built on top of computer network

## Example

- World Wide Web is built on top of Internet

  
Distributed system

  
Computer network

# Why build a computer network?

## User view

### – sharing resources

- hardware: printers, compute servers, cloud computing
- software: word, Matlab
- data: customer records, inventory, financials, p2p file sharing
- information: web-browsing, Wikipedia, search

### – communication

- email, text, voIP, screen share, video conference, social network

### – electronic commerce

- online shopping, banking, business

### – entertainment

- multi-user network games, video streaming

# Why build a computer network?

## Programmer view

- to support distributed applications
  - e.g., web, ftp, ...
- most functionality in software
  - many applications, easy to create
- general-purpose, increasingly faster computers
  - can manage many processes
- new functionality easily added ``inside" network
  - e.g., Content Distribution Net

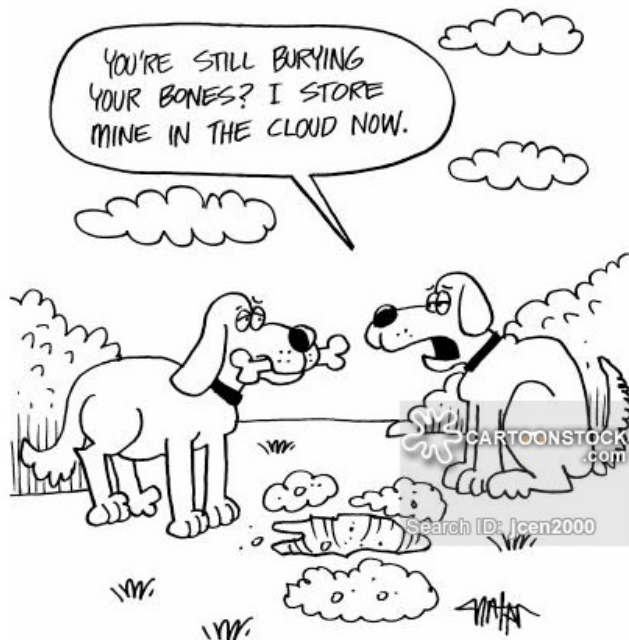
# Why should you care?

## Networks of processes are ubiquitous

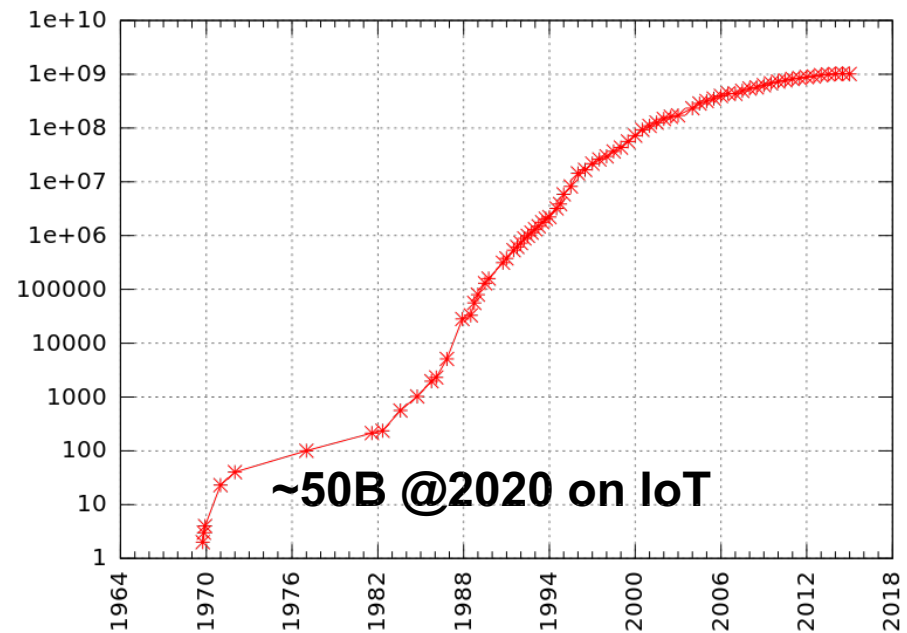
- to support a myriad of distributed applications

## Networks are getting larger and more complex

- need experts in leveraging & managing them



Number of hosts on Internet



By Kopiersperre (Own work) [CC BY-SA 3.0  
(<http://creativecommons.org/licenses/by-sa/3.0>) or GFDL  
(<http://www.gnu.org/copyleft/fdl.html>)], via Wikimedia Commons

# Many “networking” firsts originated not too far away

## First optical (light) “one-if-by-land-and-two-if-by-sea” signals

- used to signal that the British are coming in 1775

## First telegraph (Morse code)

- used by Boston Fire Alarm Telegraph System for reporting fires in 1852



**Paul Revere**

## First transatlantic radio message

- from Nova Scotia to England in 1902



**Guglielmo Marconi**

## First switches and email message

- at BBN in 1967-1972



# How to build a computer network?

1. Need way to connect devices
2. Need way to connect processes on devices
3. Need way to share-resources efficiently

We'll overview general networks today. But in future our focus will primarily be Internet

**Building a Network**

**HOW TO CONNECT DEVICES**

# Building blocks

**Nodes:** laptop, server, router, switch, cell phone, UAV, IoT devices, ...

**Links:** copper wire, coaxial cable, optical fiber, radio, ...

## Telephone lines

Ethernet, up to 10 Gbps



## Cable TV infrastructure

Shared/broadcast medium, more people using simultaneously, less bandwidth each gets



10's of Mbps

## Glass fiber carrying light pulses (bits)

Forms Internet core: carries lots of traffic. Low bit error rate since unaffected by electromagnetic noise



up to 100s of Gbps

Kbps =  $10^3$  bits per second  
Mbps =  $10^6$  bits per second

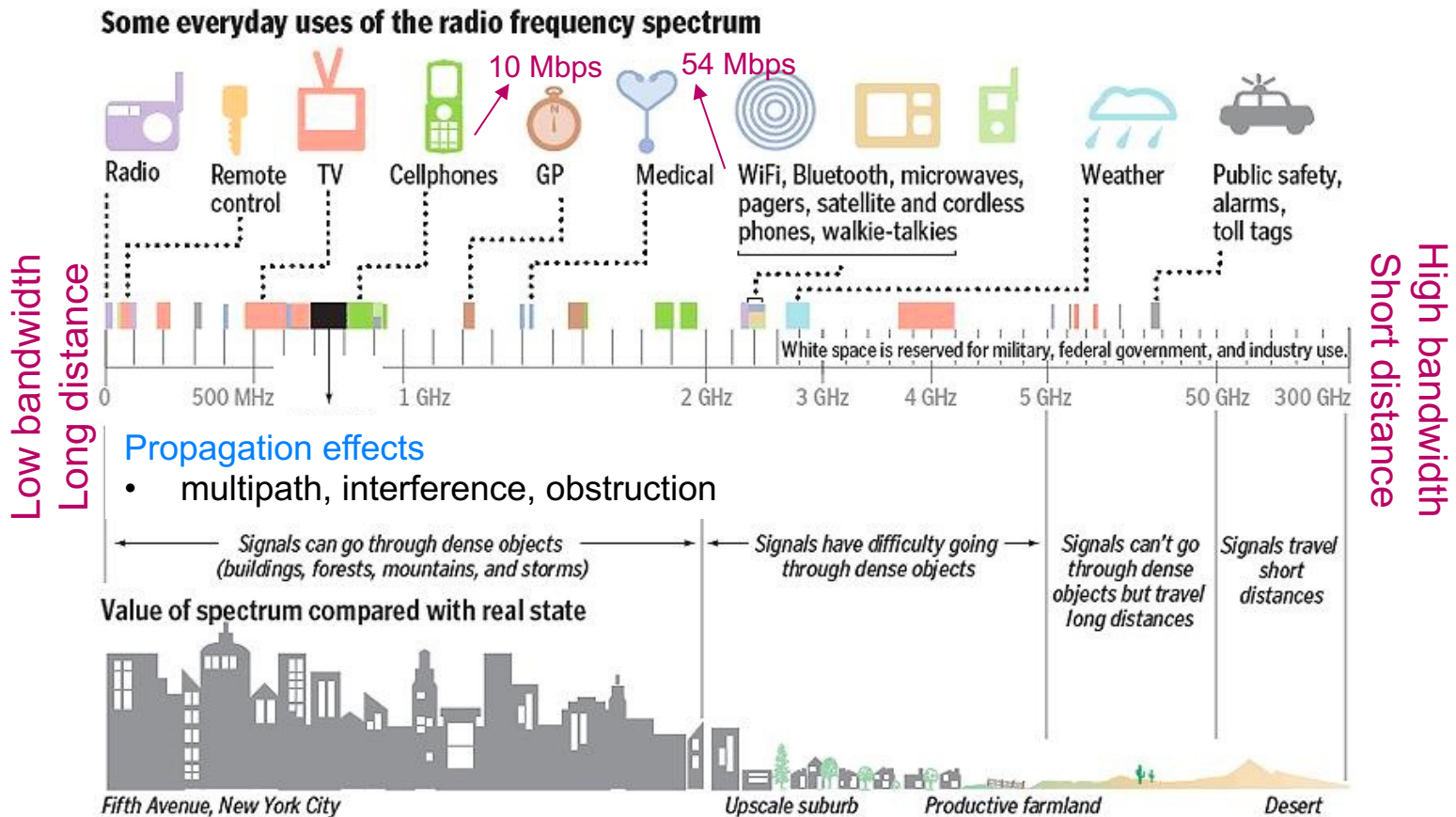
 **Wired link media**

# Building blocks

Nodes: laptop, server, router, switch, cell phone, UAV, IoT device...

Links: copper wire, coaxial cable, optical fiber, radio, ➡ **Wireless**

Signal carried in electromagnetic spectrum } Shared, typically broadcast, medium



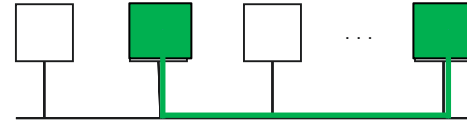
# Connecting devices with direct links

## Point-to-point



E.g., dial-up, Digital Subscriber Line (DSL))

## Multiple access



LAN environment

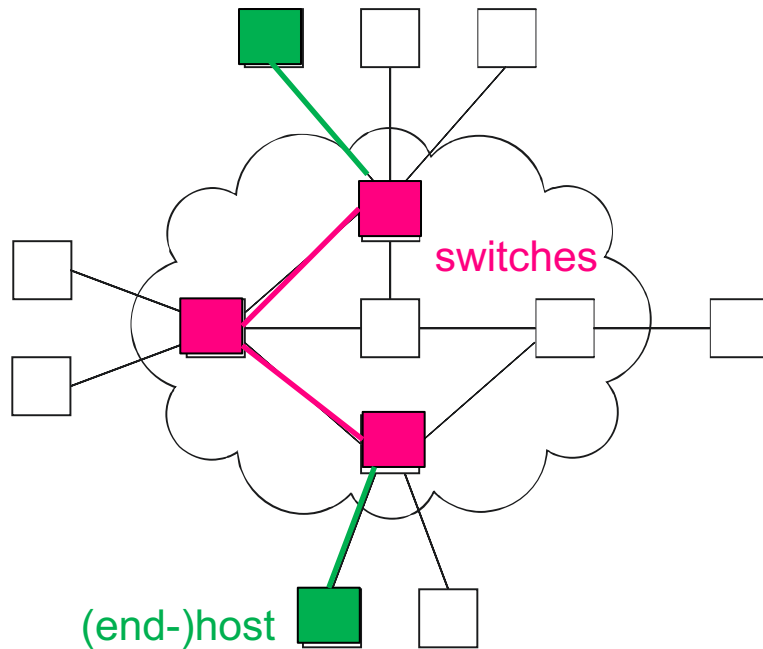
**Need MAC (Medium Access Control) protocol**

to control access to shared medium. E.g., shared Ethernet, Hybrid Fiber Coaxial (HFC) upstream channel, wireless

# Connecting devices with switches and routers

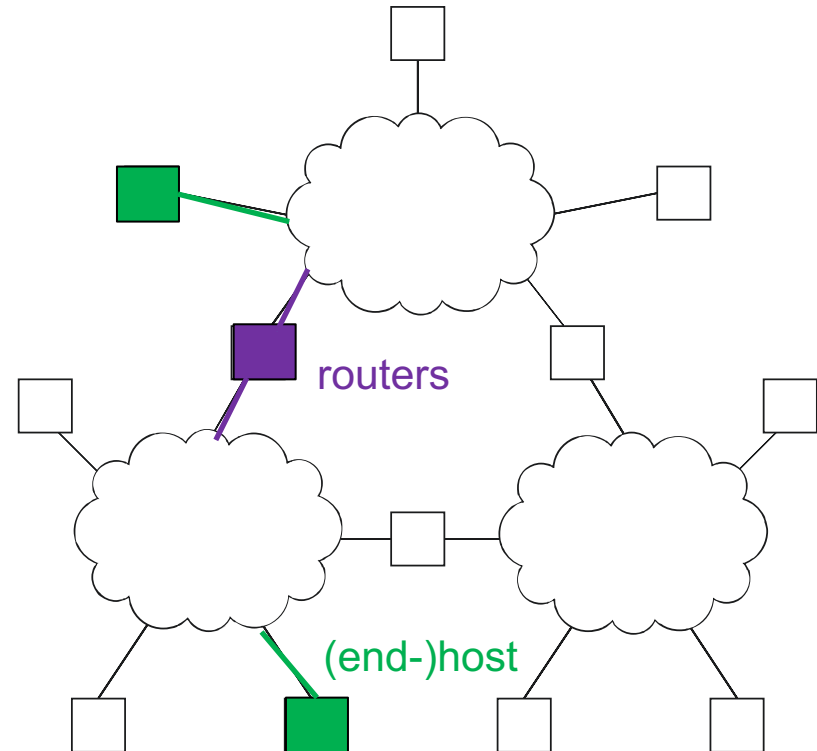
## Indirect connectivity

- switched network



## Internetwork

- routers: connect networks



## A network can be defined recursively

- 2 or more devices connected by a physical link
- 2 or more networks connected by 2 or more devices

# How do devices identify and find each other?

## Addressing

- address is byte-string that identifies device; usually unique

## Routing

- algorithm determining how routers forward messages toward destination device based on address

## Types of addresses

- **unicast**: device-specific
- **broadcast**: all devices on network
- **multicast**: some subset of device on network

# Internet addresses example

Every device on Internet has Internet Protocol (IP) address

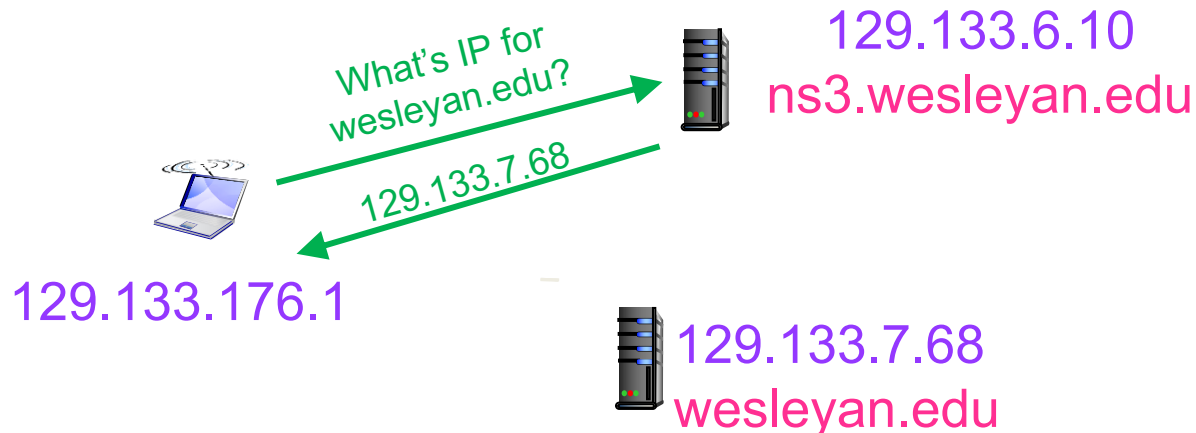
- string of #s interpretable by computer
- assigned when host joins network connected to Internet



129.133.176.1

Some IP addresses are associated with a domain name

- use equivalent of phone book to do mapping



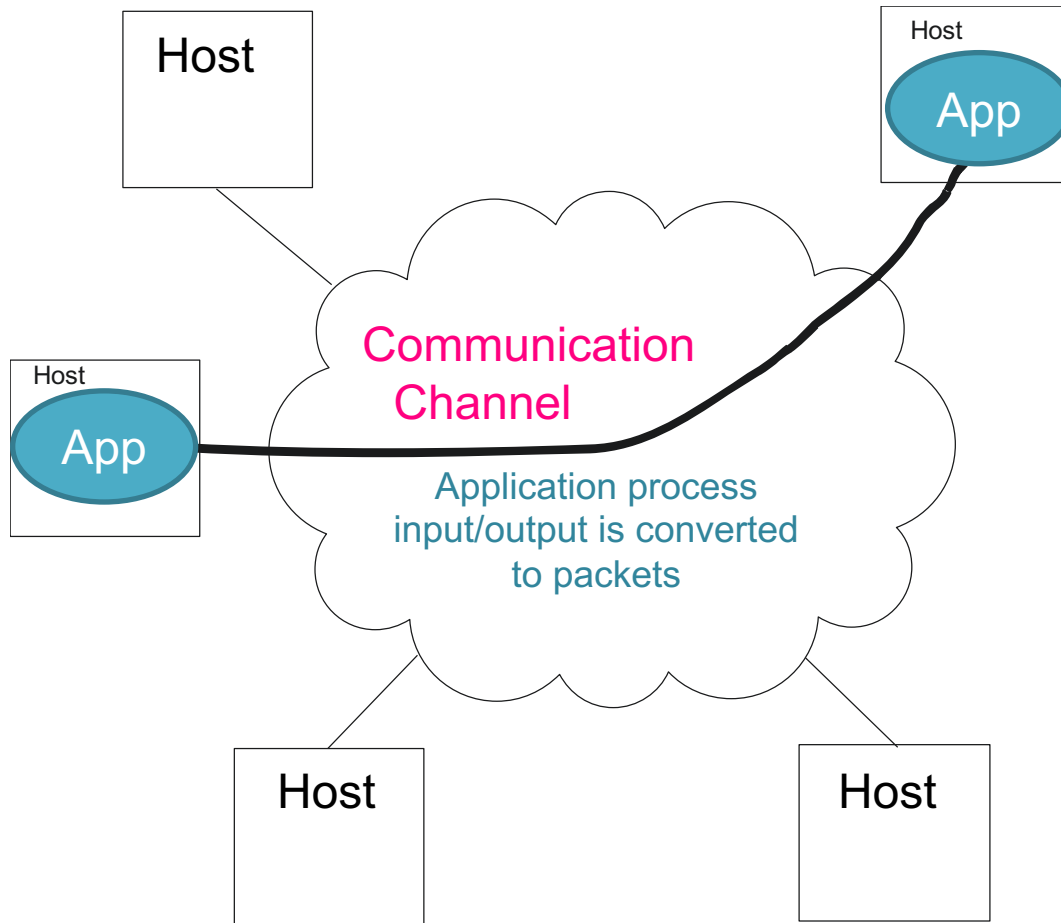


**Building a Network**

**HOW TO CONNECT  
PROCESSES ON DEVICES**

# Processes, not devices, are communicating

How do processes running on different devices communicate?



## Packet

- sequence of bits
- 0101111010100000110...

Header Data

## Bit

- what propagates over links between src/dest

# Typical goals for communication channels

## Reliable

- no loss, no errors, no duplication, in-order
- for file access and digital libraries

## Secure

- privacy, authentication, message integrity

## Delay-bounded

- for real-time voice and video

# What goes wrong in network?

## All sorts of things ...

- bit-level errors (electrical interference)
- packet-level errors (bit errors, congestion)
- link and node failures
- packets are delayed
- packets are delivered out-of-order
- third parties eavesdrop

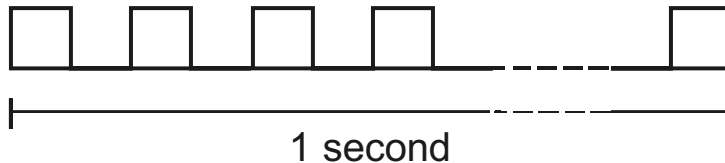
## Communication channel must work even if things go wrong

- key problem
  - fill in gap between what applications expect and what underlying technology provides

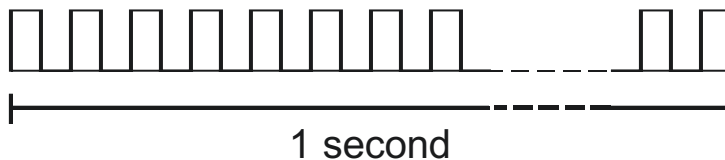
# Quantifying channel performance

## Bit Rate (aka throughput aka capacity)

- amount of data that can be transmitted per unit time
  - link versus end-to-end
- measurement units
  - Kbps =  $10^3$  bits per second
  - Mbps =  $10^6$  bits per second
  - Gbps =  $10^9$  bits per second



1Mbps  
(each bit 1 microseconds wide)



2 Mbps  
(each bit 0.5 microseconds wide)

# Quantifying channel performance

## Delay

- **time to send packet** from host A to host B
  - example: 24 milliseconds (ms)
  - sometimes interested in round-trip time (RTT)
    - include time to get reply back from host B
- components
  - **Total Delay** = **Processing** + **Propagation** + **Transmission** + **Queue**
  - **Propagation** Delay = Distance / SpeedOfLight
  - **Transmission** Delay = Packet length / Bit Rate
- speed of light
  - $3.0 \times 10^8$  meters/second in a vacuum
  - $2.3 \times 10^8$  meters/second in a cable
  - $2.0 \times 10^8$  meters/second in a fiber

**Building a Network**

**HOW TO SHARE RESOURCES**

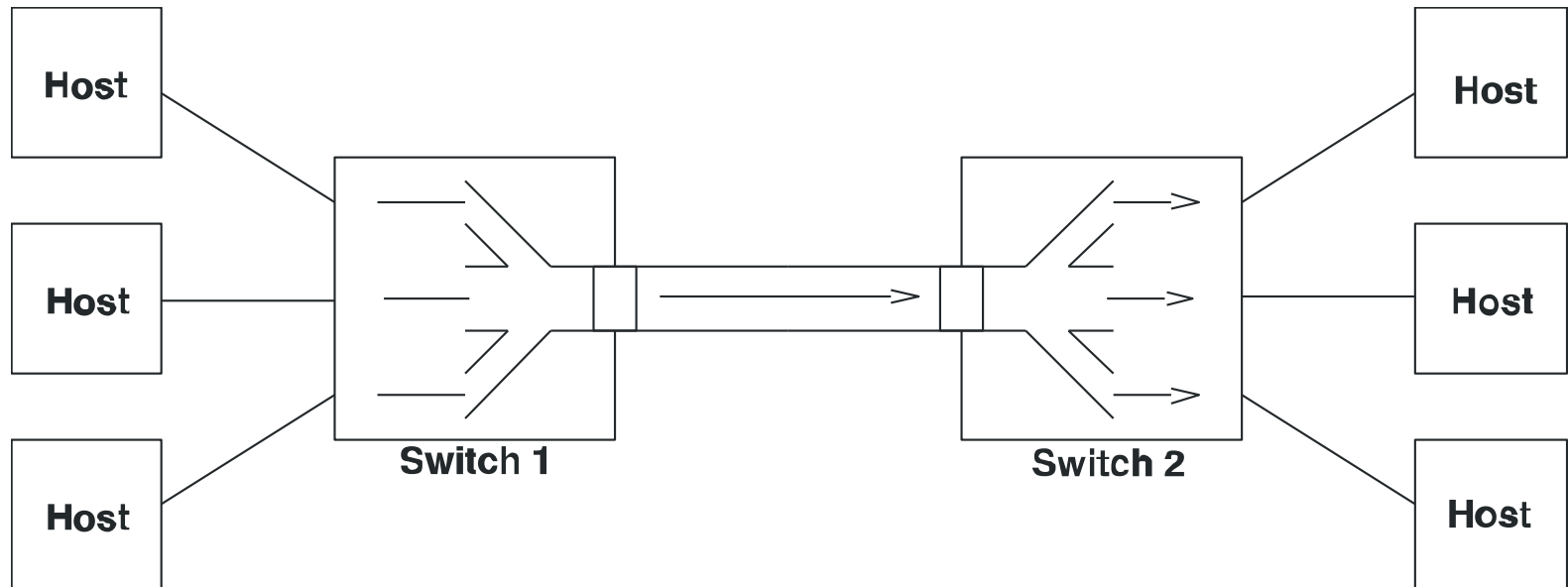
# Sharing network resources

## Devices and links

- must be shared (**multiplexed**) among multiple users

## Common Multiplexing Strategies

- **Frequency-Division Multiplexing (FDM)**: pre-assign frequencies
- **Time-Division Multiplexing (TDM)**: pre-assign time slots





# Multiplexing strategy used on Internet

## Statistical Multiplexing

- time-division, but **on demand** rather than fixed (no waste)
  - reschedule link on per-packet basis
  - packets from different sources interleaved on link
- buffer overflow causing **packet drops** (loss), is called **congestion**

