University of Colorado **Boulder**

CSCI 3403 INTRO TO CYBERSECURITY

Lecture: 10-1

Topic: Networking

Presenter: Matt Niemiec

Announcements

- Grades went out to your emails
  - There was a minor error on Homework 6. Could change up to 1% of your overall total
- Project 3 is posted
  - Please get started early on this!
- Next homework coming later this week
- Upcoming guest lecturers
  - From Proofpoint on 4/9
  - From Rule4 on 4/21
  - From Twitter on 4/23 (Andy Sayler)

Final Exam, current picture

- Will probably happen
- If you get a 65% or higher on the final, the highest of your final and midterm will count for both scores
- Will be online Moodle quiz
- Will be open book, open note, open internet
  - NOT open friend
  - Will be different/harder in some ways to compensate
  - May be required to write basic scripts

- Note: This is subject to change

Exam Extra Credit (Current picture. Will be finalized before next Tuesday)

- Research a topic that won't be discussed in class
- Get credit in one of the following ways
  - Record a 5-minute video explanation of your topic for up to 10%. Details to come soon
  - Present your topic for 5 minutes in recitation for up to 15%. Depending on demand, this may be first-come-first-serve. If you got an 85% or better on the midterm, please leave this for others
  - If you're selected as an outstanding project from recitation, give a 10-minute presentation for up to 25% (total)
- Percentages apply to your higher exam score

Exam Extra Credit Criteria

- Will be graded on at least the following:
  - Interesting topic/information relevant to cybersecurity
  - Quality, professional preparation and presentation
  - Inspires the listener to want to learn more and provides resources to do so
  - Shows insight and depth in research presented in an appropriate manner for the given timeframe

Some Extra Credit Potential Topics

- Network security
  - Wireless security, honeypots, cloud security, SIEM, Tor
- Applied security
  - OWASP Top 10, reverse engineering, penetration testing
- Crypto
  - Common crypto libraries, homomorphic encryption
- Windows
  - Windows/AD security, Windows CLI
- Miscellaneous
  - Ethics in security, auditing, data provenance
- Or anything else! Just run it by Matt or your TA

Technology Recap 3/17 (Old stuff)

- Piazza is used for content-related questions
- Feedback: https://forms.gle/WRUUbPkmFNsa6q3D6
- Instructor/TA email is used for individual circumstances
- cyber@Colorado.edu is used for accommodations/logistical questions
- Moodle is used for assignments, slides, and additional resources

Technology Recap 3/17 (New stuff)

- Calendar is used for holding all Zoom meetings, instructions, and meeting IDs
  - May contain due dates, but not guaranteed
- Lecture Zoom ID: https://cuboulder.zoom.us/j/633893668
  - This and others found in Google Calendar
- Lecture capture folder: https://drive.google.com/drive/folders/1VMrHEigP4AgDwRnRPTsgQS35EAozc19-?usp=sharing

Binary Exploits

Buffer Overflow Attacks

- A buffer overflow (buffer overrun or buffer overwrite) can occur as a result of a programming error when a process attempts to store data beyond the limits of a fixed-sized buffer and consequently overwrites adjacent memory locations

Buffer Overflow Problems

- Extremely difficult to detect
  - Difficult to prove one exists, even with proper functions
- Still very common

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Figure 10.1 Basic Buffer Overflow Example

Memory Address	Before gets(str2)	After gets(str2)	Contains Value of
. . . .	. . . .	. . . .	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
. . . .	. . . .	. . . .	

Figure 10.2 Basic Buffer Overflow Stack Values

Local Variable Ordering

- In general, previous is standard
  - But it isn't guaranteed!
- If you're unsure about stack ordering, check in GDB
- Typically ints/pointers are on top (higher addresses)
  - Followed by arrays towards the bottom
- GCC optimizations don't guarantee anything!

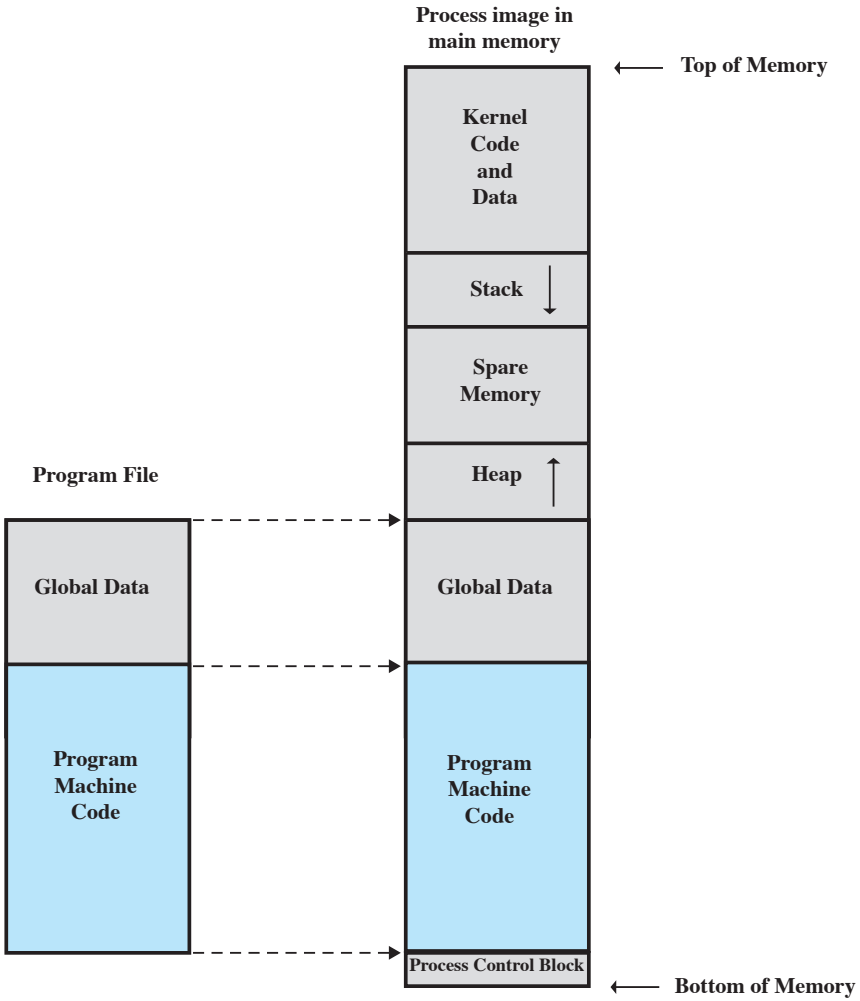


Figure 10.4 Program Loading into Process Memory

Stack Overflow Motivations

- Crash the server (typically easy)
- Change control
- Execute some code (Bomb lab)
- Get a shell (Real prize!)

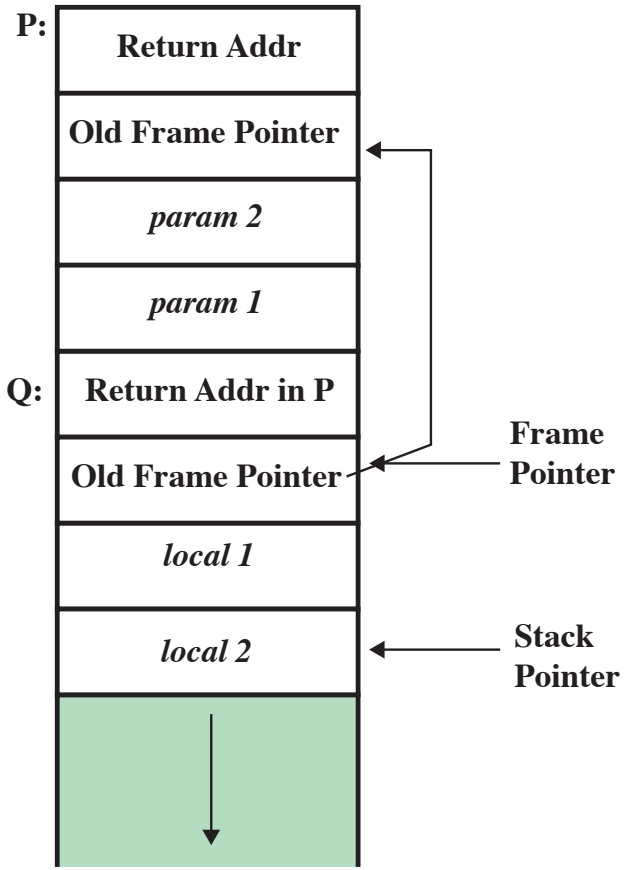


Figure 10.3 Example Stack Frame with Functions P and Q

Getting A Shell

- A shell allows you to become that user
- You can set up persistence
- Should be position-independent
- Should be in envp, not the buffer
- Should involve a NOP sled

Defending Against Buffer Overflows

- Don’t use any unsafe functions!
- Comb code carefully for logic errors
  - E.g. off-by-one errors:

- Don’t use any of the following unsafe functions:

`gets(char *str)`	read line from standard input into str
`sprintf(char *str, char *format, ...)`	create str according to supplied format and variables
`strcat(char *dest, char *src)`	append contents of string src to string dest
`strcpy(char *dest, char *src)`	copy contents of string src to string dest
`vsprintf(char *str, char *fmt, va_list ap)`	create str according to supplied format and variables

Compile/Runtime Defenses

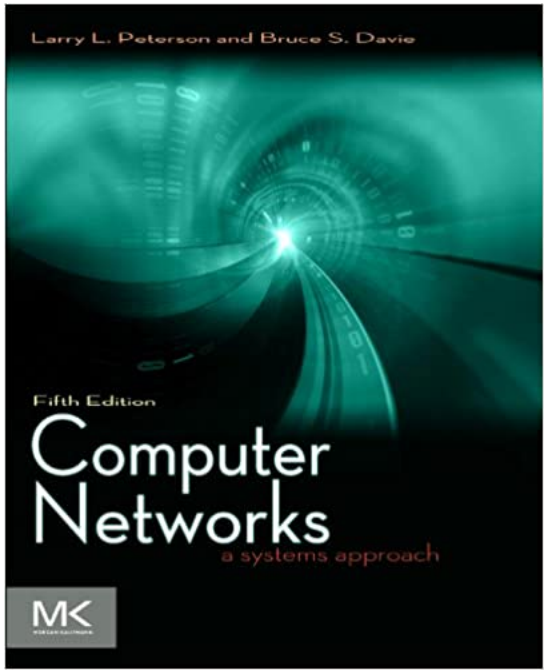
- Stack smashing detection
  - E.g. Canaries
- Stack randomization
- Guard pages
  - Regions of empty virtual address space between critical regions
- Executable address space protection

Other Binary Exploit Tricks

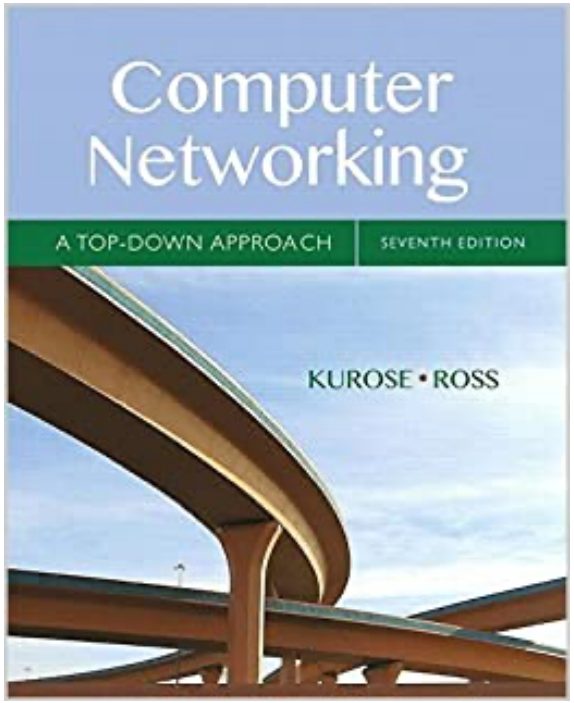
- The *file* command on Linux is your friend
- Remember which way the stack goes
- Spend a little time in GDB
  - Watch recitation video and use a cheat sheet
  - See: https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf
- Don't forget directory traversal
- You may be able to manipulate symlinks
  - See https://unix.stackexchange.com/a/2910

Networking Intro

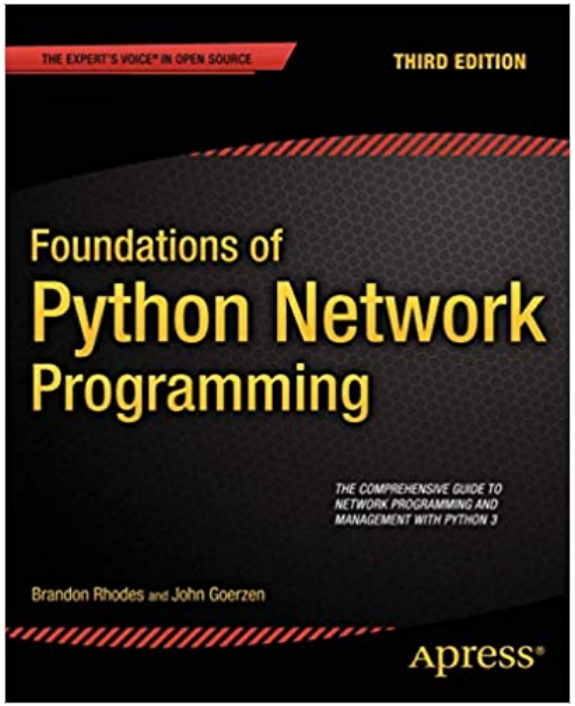
*Slides adapted from Computer Networking: A top-Down Approach and Computer Networks: A Systems Approach



Strongly Recommend



Recommend



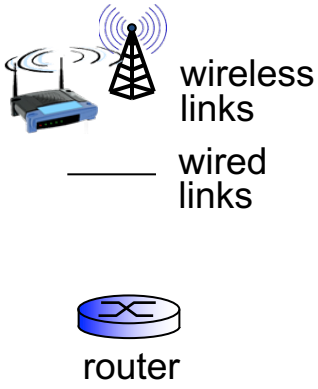
If you're curious

Questions of Networking

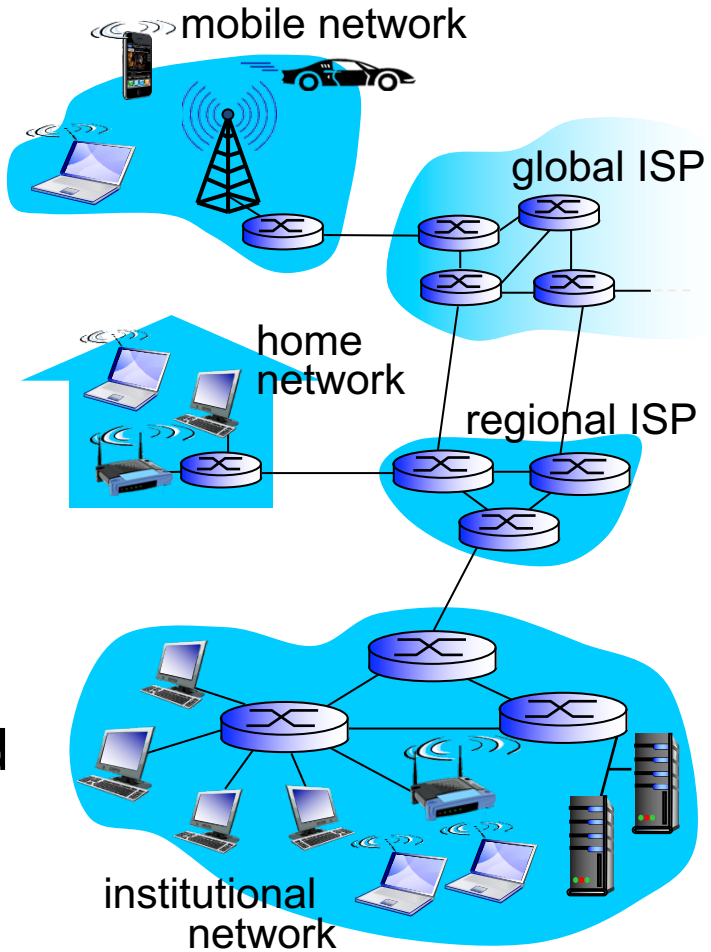
- What is a computer network?
- What is a computer network architecture?
- What's a protocol?
- What do we need to know about the internet to secure it?

- **Our goal:** Learn enough about networking to effectively use and secure computers on the Internet

What's the Internet?

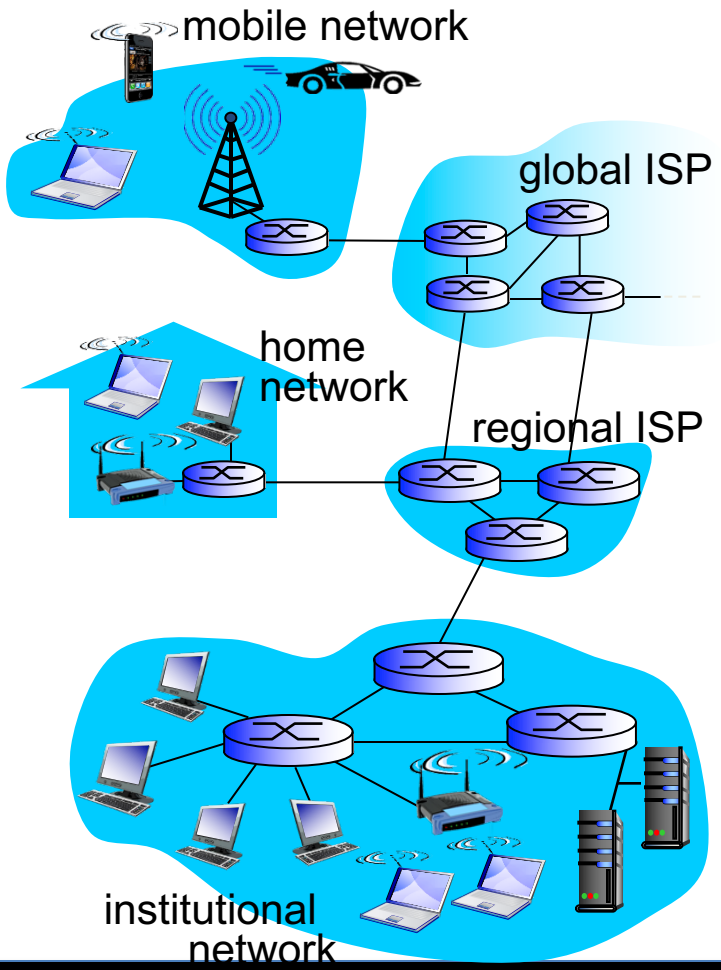


- Millions of connected computing devices:
  - Hosts = end systems
  - Running network apps
- Communication links
  - Fiber, copper, radio, satellite
  - Transmission rate: bandwidth
- Packet switches: forward packets (chunks of data)
  - Routers and switches



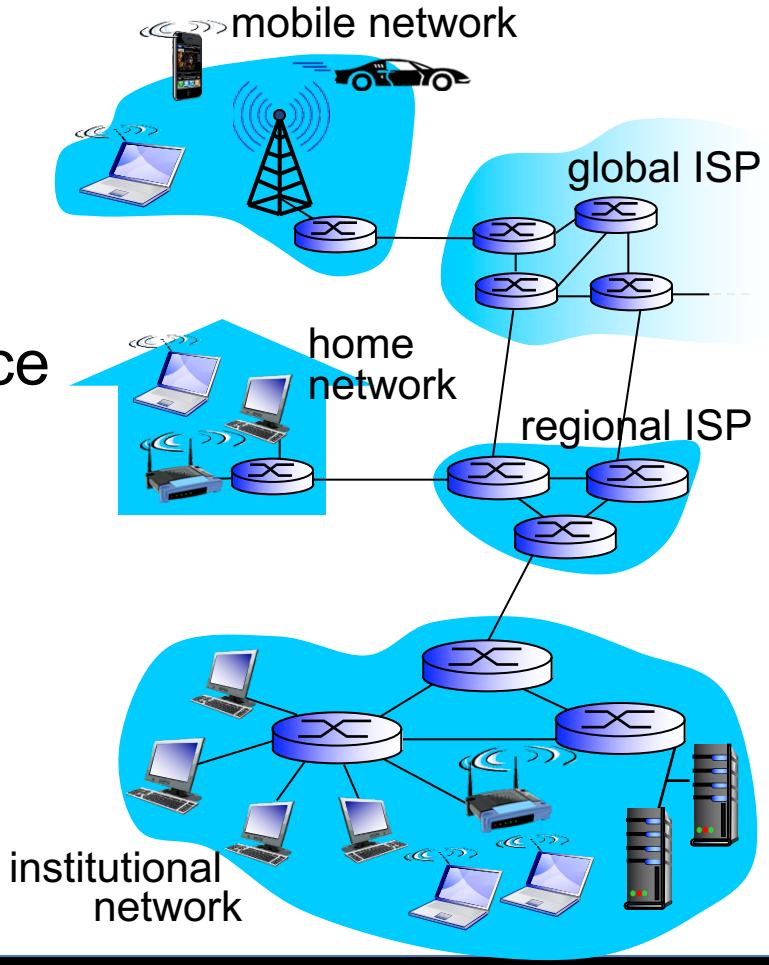
What's the Internet: "nuts and bolts" view

- Internet: "network of networks"
- Interconnected ISPs
- Protocols control sending, receiving of messages
- E.g., TCP, IP, HTTP, Skype, 802.11
- Internet standards
- RFC: Request for comments
- IETF: Internet Engineering Task Force



What's the Internet: A Service view

- Infrastructure that provides services to applications:
- Web, VoIP, email, games, e-commerce, social nets, …
- Provides programming interface to apps
- Hooks that allow sending and receiving app programs to "connect" to Internet
- Provides service options, analogous to postal service



What's a Protocol?

- Human protocols:
  - "What's the time?"
  - "I have a question"
  - Introductions

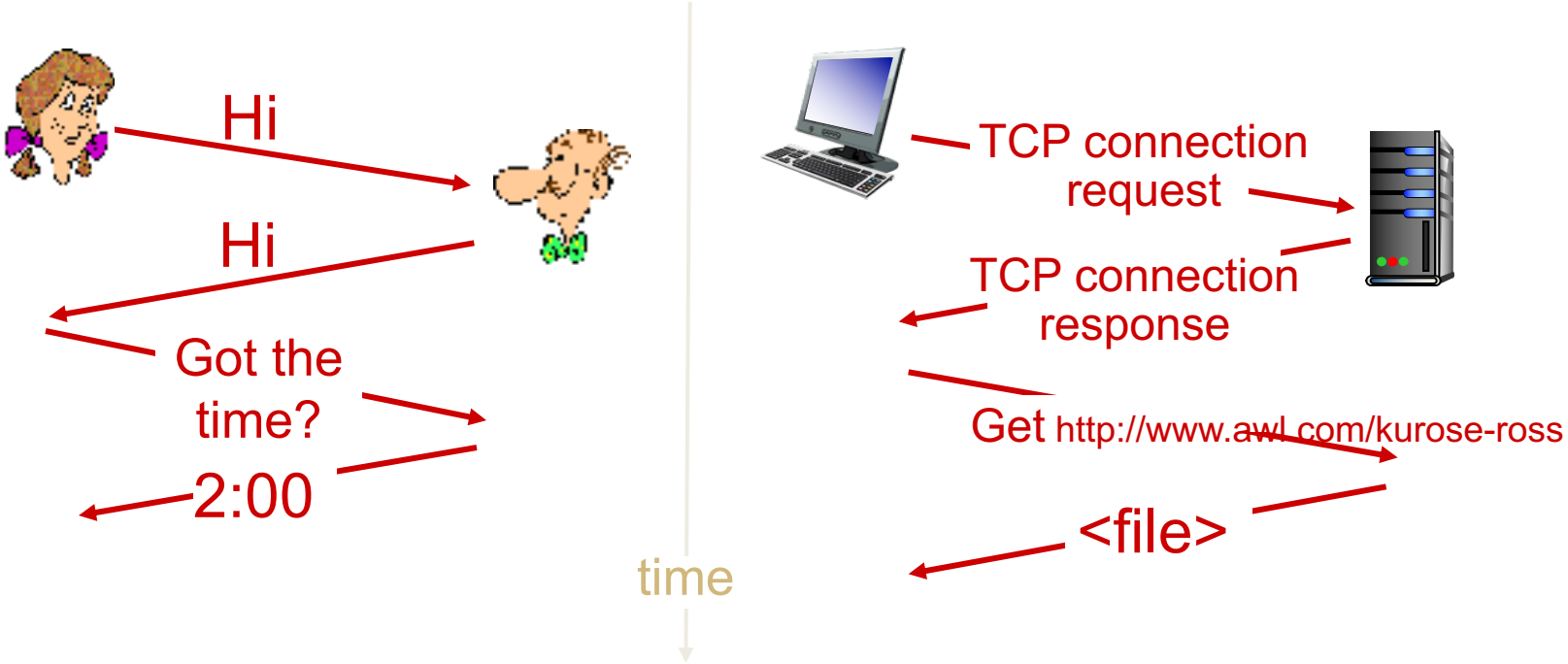
- … Specific messages sent
- … Specific actions taken when messages received, or other events

- Network protocols:
  - Machines rather than humans
  - All communication activity in Internet governed by protocols

Protocols define format, order of messages sent and received among network entities, and actions taken on msg transmission, receipt

What's a Protocol?

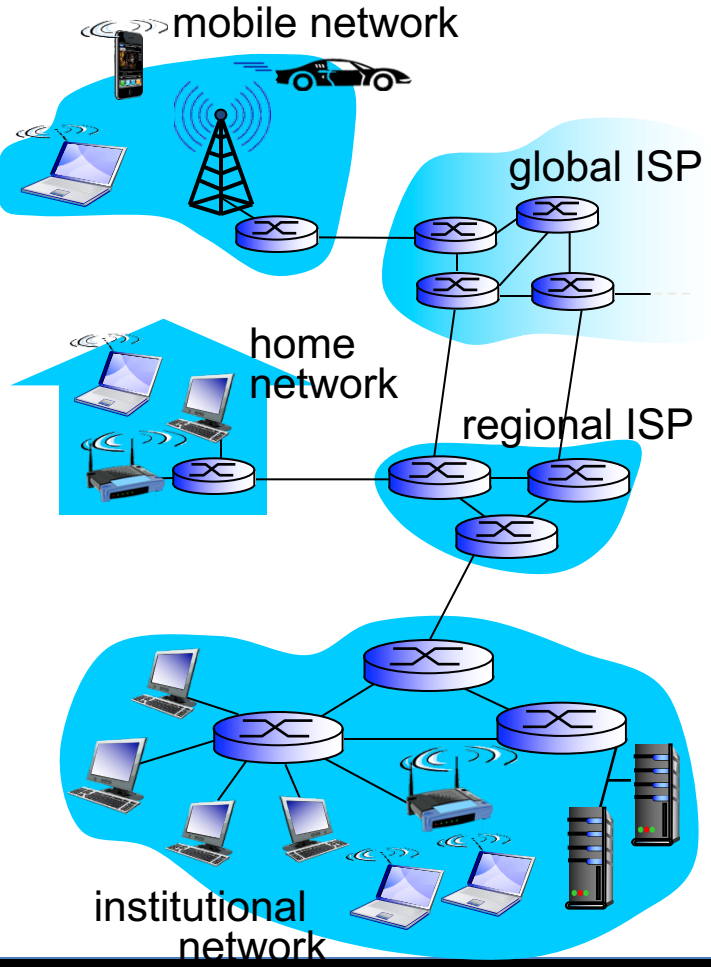
A human protocol and a computer network protocol:



Q: other human protocols?

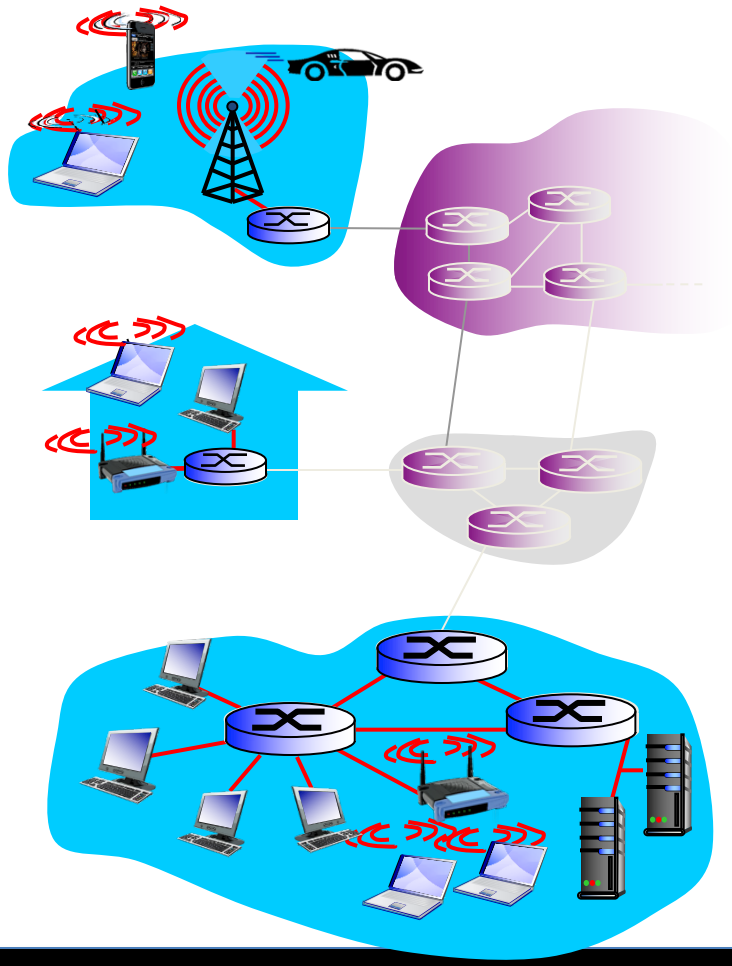
A Closer Look at Network Structure:

- Network edge:
  - Hosts: clients and servers
  - Servers often in data centers
- Access networks, physical media: wired, wireless communication links
- Network core:
  - Interconnected routers
  - Network of networks

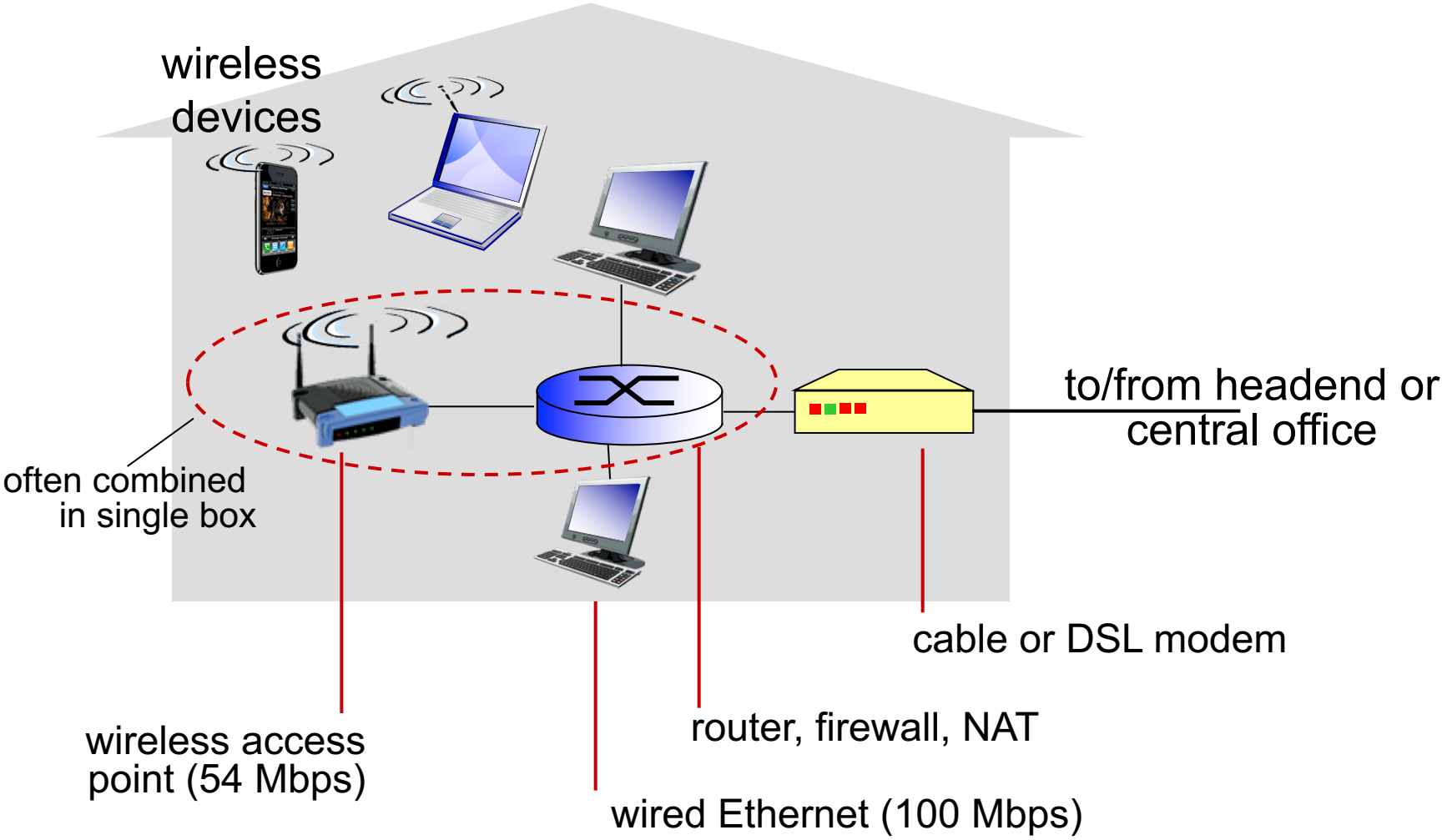


Access Networks and Physical Media

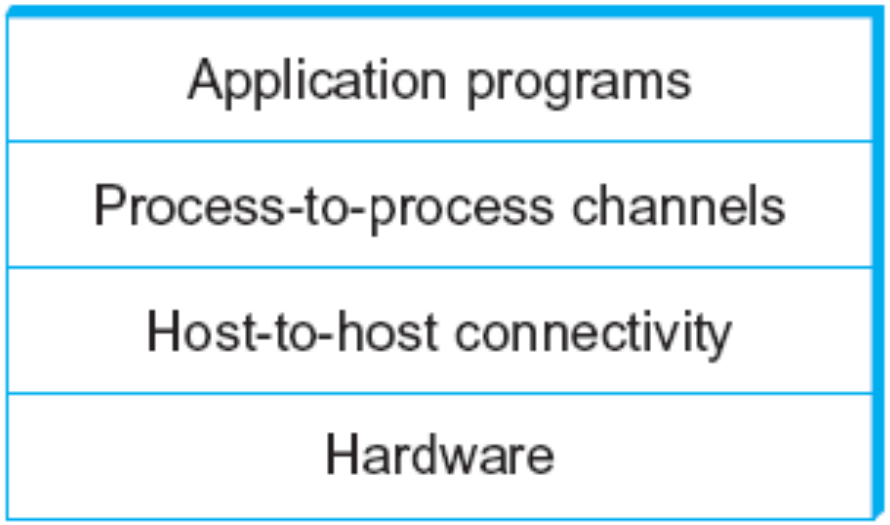
- Q: How to connect end systems to edge router?
  - Residential access nets
  - Institutional access networks (school, company)
  - Mobile access networks
- Keep in mind:
  - Bandwidth (bits per second) of access network?
  - Shared or dedicated?



Access Net: Home Network

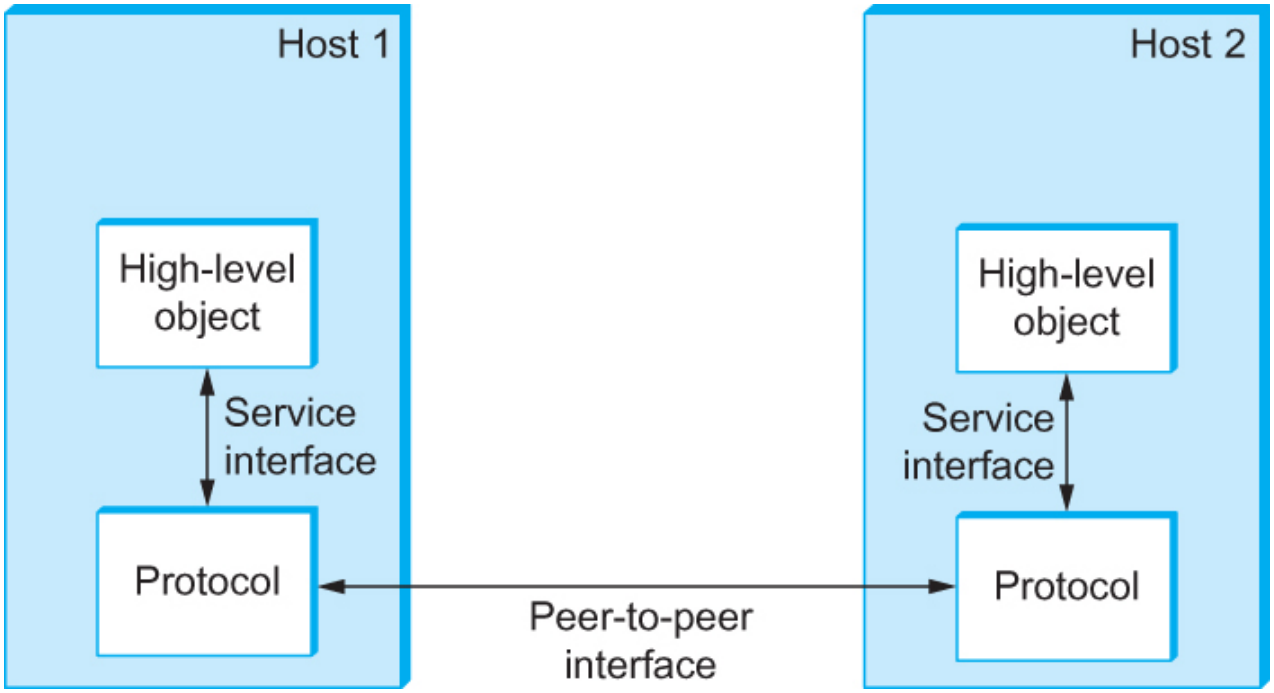


General Layers of a Network



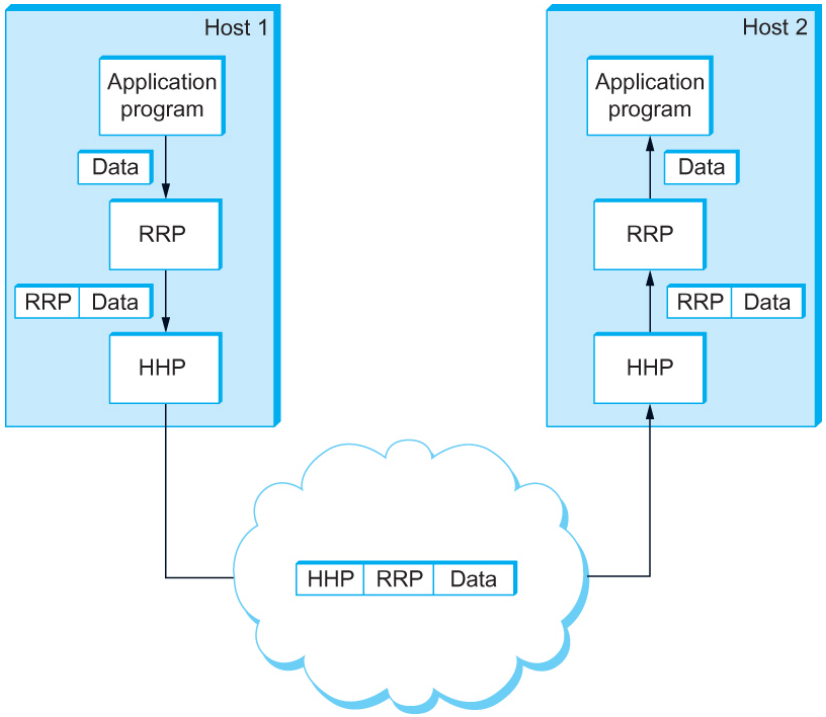
Example of a layered network system

Interfaces

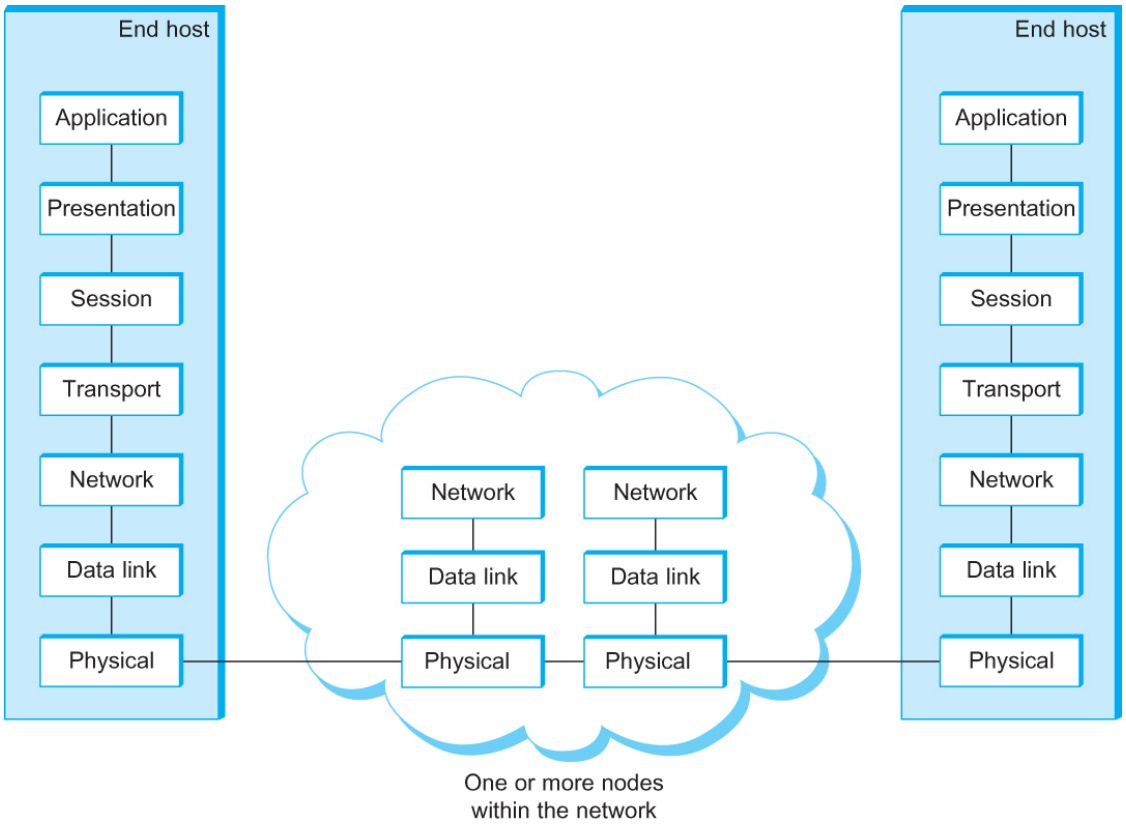


Service and Peer Interfaces

Encapsulation



High-level messages are encapsulated inside of low-level messages



The OSI 7-layer Model

OSI – Open Systems Interconnection

Layer 1: Physical Layer

*Slides adapted from Computer Networking: A top-Down Approach

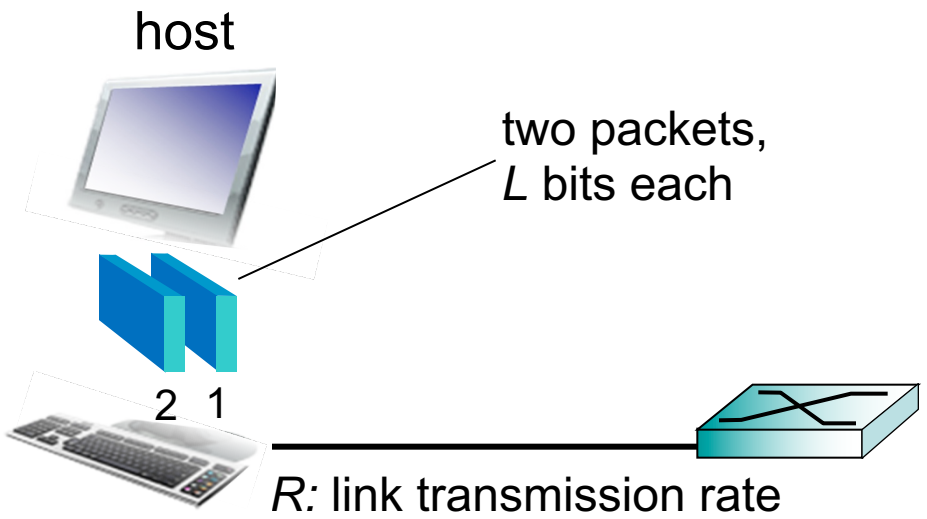
Physical Layer

- Sends 1s and 0s across a physical/wireless link
- Ensure reliability
- Placing binary data on a link is called *encoding*
  - We will know that encoding exists, but not deal with it directly
- Provides framing
  - Gives metadata in order to better transmit (e.g. checksums, error detection/correction, number of bytes being transmitted in frame)

- Example: I want to send you 60 bytes of data. Here

Host: Sends Packets of Data

- Host sending function:
  - Takes application message
  - Breaks into smaller chunks, known as packets, of length L bits
  - Transmits packet into access network at transmission rate R



- Link transmission rate, aka link capacity, aka link bandwidth

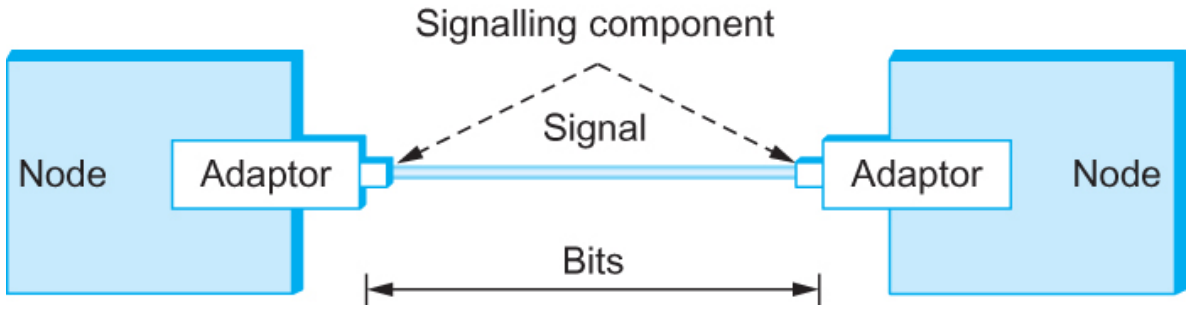
$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Physical Media

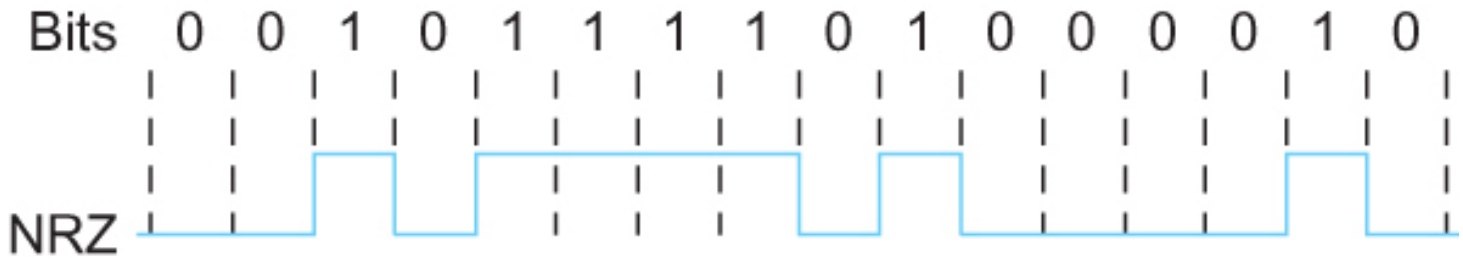
- Bit: propagates between transmitter/receiver pairs
- Physical link: what lies between transmitter & receiver
- Guided media:
  - Signals propagate in solid media: copper, fiber, coax
- Unguided media:
  - Signals propagate freely, e.g., radio

- Twisted pair (TP)
- Two insulated copper wires
  - Category 5: 100 Mbps, 1 Gbps Ethernet
  - Category 6: 10Gbps

Encoding: Most Basic



Signals travel between signaling components; bits flow between adaptors

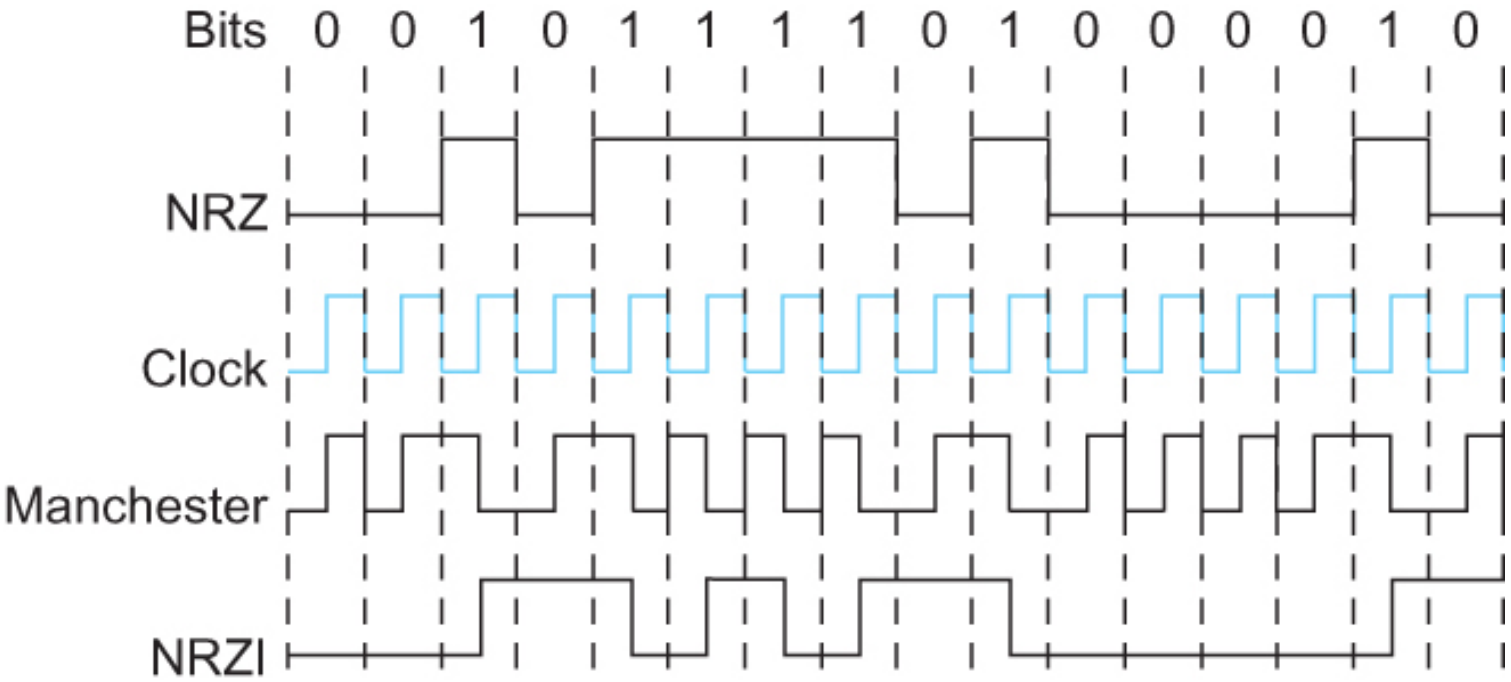


NRZ encoding of a bit stream

NRZ Problems

- Baseline wander
  - NRZ keeps an average of the signals it's seen so far. Receiving many 1s or 0s will skew the average, making it difficult to compare incoming signals
- Clock recovery
  - Clock cycles are used to know how long a bit is. Clocks must be perfectly in sync to work. With long streams of 1s or 0s, it's difficult to know exactly how many bits were transmitted

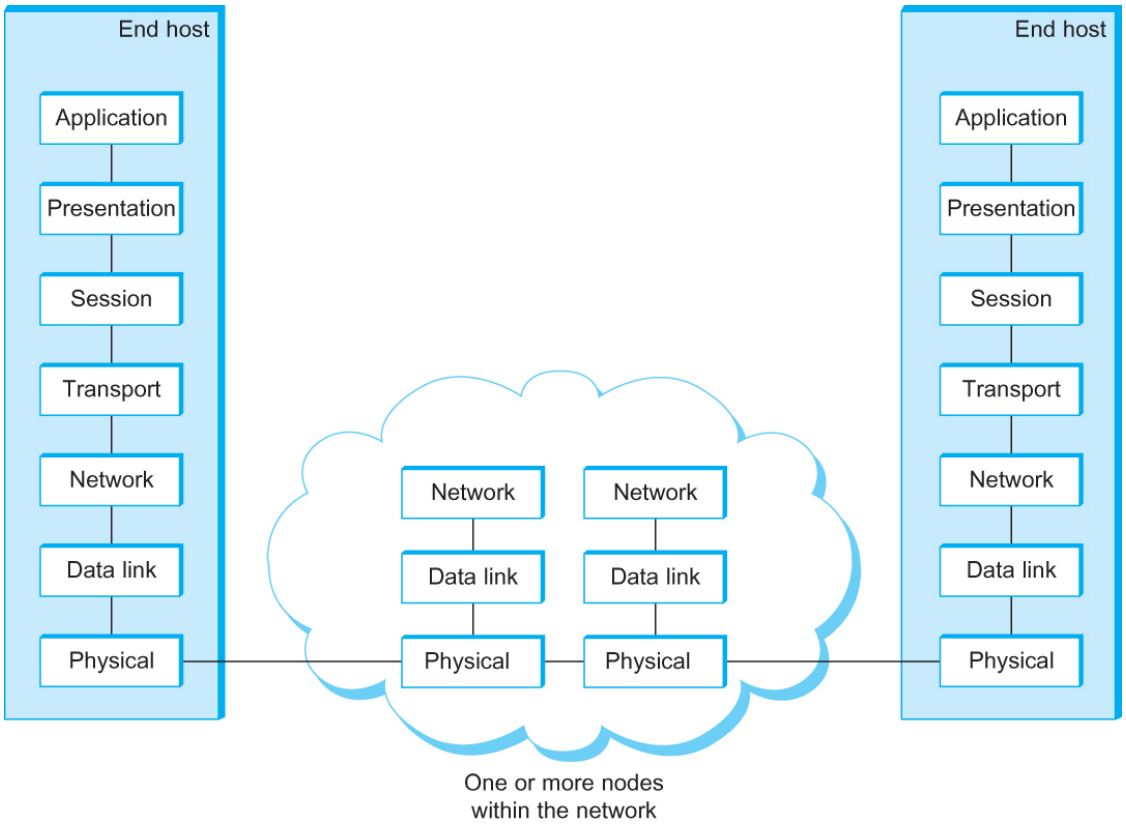
On Your Own: Find the Patterns



Different encoding strategies

Layer 2: Data Link Layer

*Slides adapted from Computer Networking: A top-Down Approach



The OSI 7-layer Model

OSI – Open Systems Interconnection

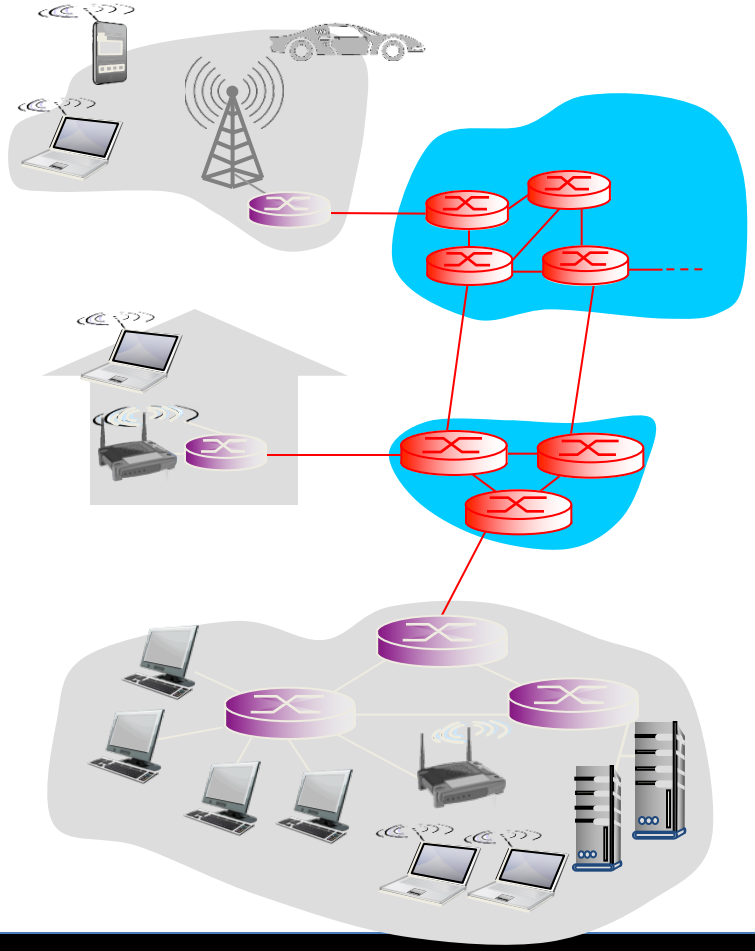
Data Link Layer

- Provides some basic frame distribution
- Introduces *switching* and *bridging*, but not yet routing
  - We won't get caught up with these
- Begins providing scalability
- Transfers are performed based on the physical address of the recipient
  - Media Access Control (MAC)

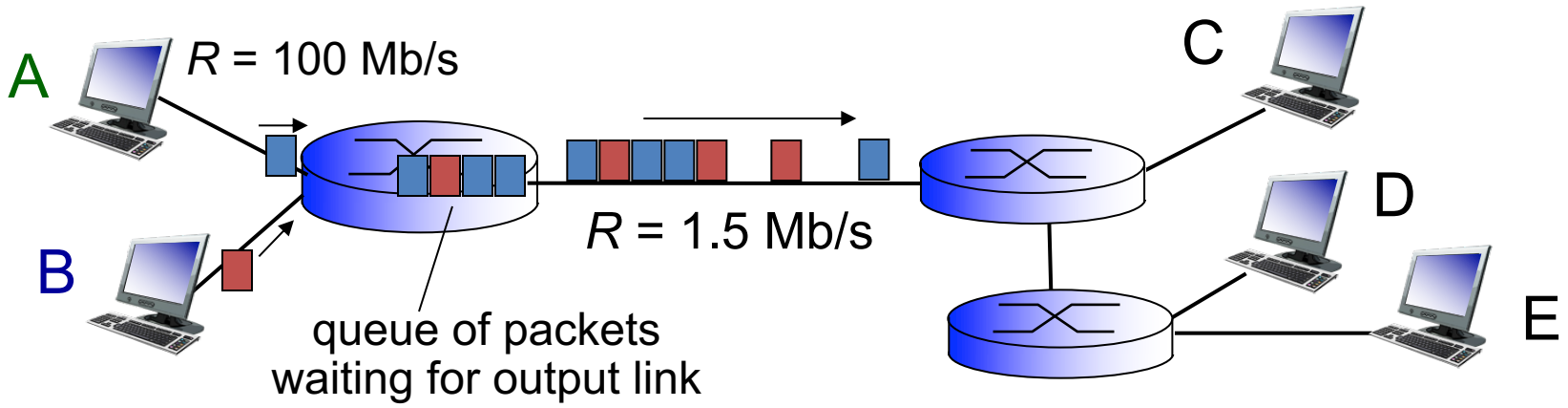
- Example: Hello! You are directly connected to Bob, or know exactly how to get to him. Please deliver these 60 bytes of data to him

The Network Core

- Mesh of interconnected routers
- Packet-switching: hosts break application-layer messages into packets
  - Forward packets from one router to the next, across links on path from source to destination
  - Each packet transmitted at full link capacity



Packet Switching: Queueing Delay, Loss

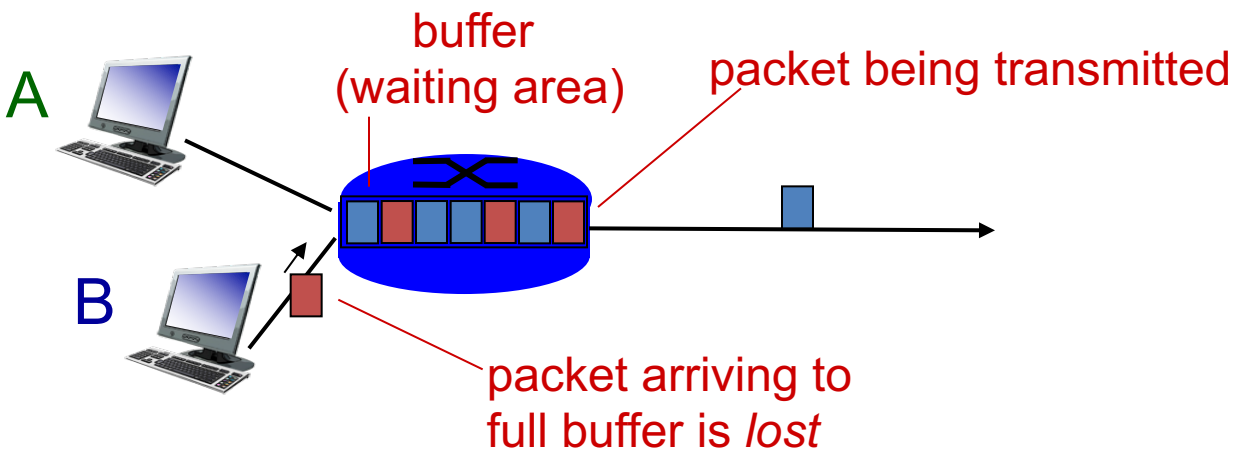


Queuing and loss:

- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - Packets will queue, wait to be transmitted on link
  - Packets can be dropped (lost) if memory (buffer) fills up

Packet Loss

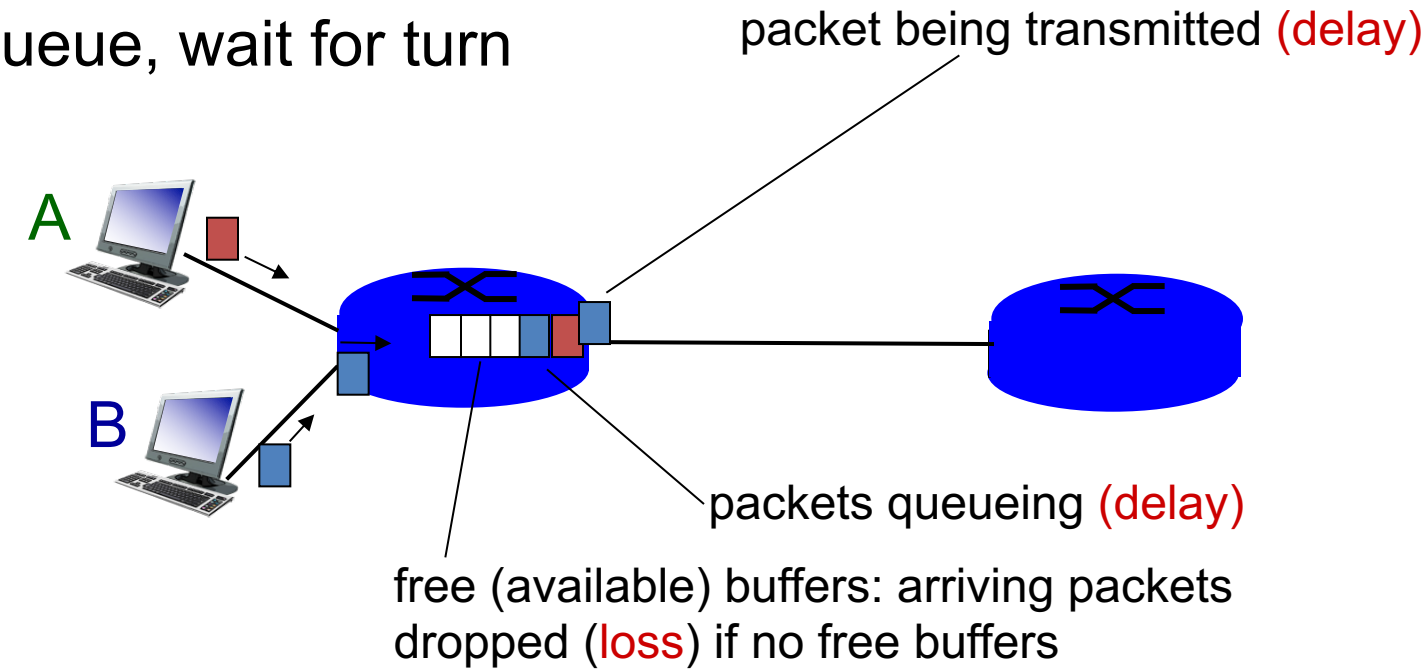
- Queue (aka buffer) preceding link in buffer has finite capacity
- Packet arriving to full queue dropped (aka lost)
- Lost packet may be retransmitted by previous node, by source end system, or not at all



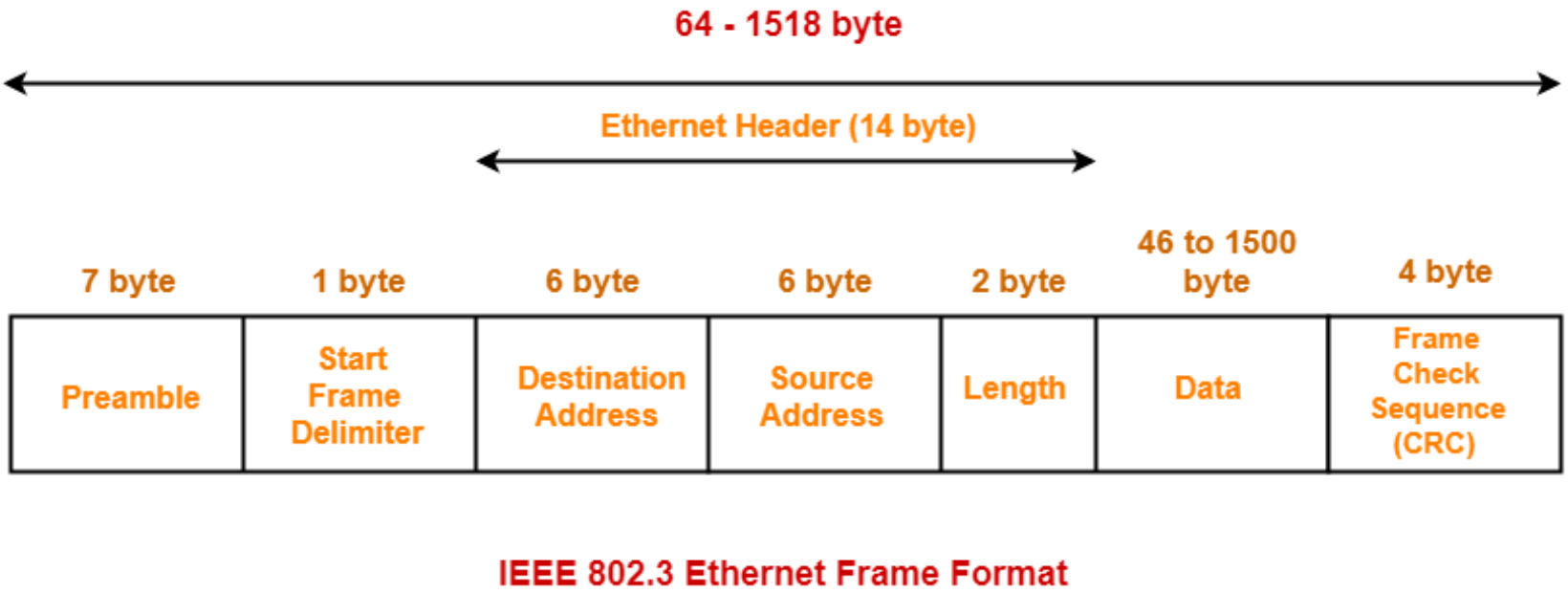
How do Loss and Delay Occur?

Packets queue in router buffers

- Packet arrival rate to link (temporarily) exceeds output link capacity
- Packets queue, wait for turn



Current Transmission Example



IEEE 802.3 Ethernet Frame Format