

# CS1680

## Physical Layer, Link Layer I

TODAY

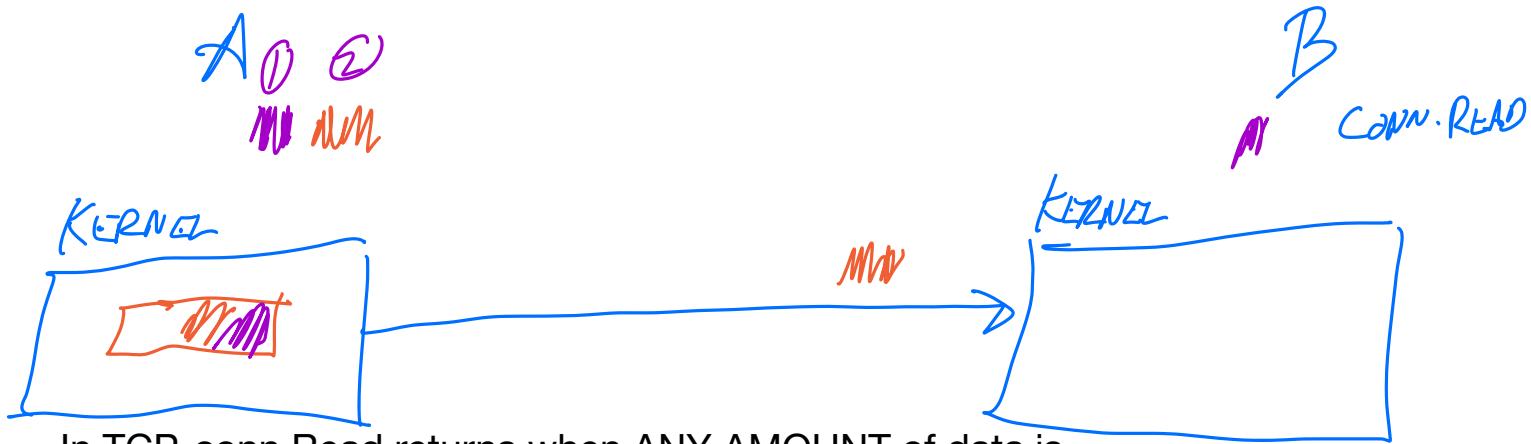
- SOCKETS WRAPUP
- PHYSICAL LAYER

# Administrivia

- Snowcast: milestone due last night
  - I will be reviewing today; look for an announcement for feedback, reference doc, Gradescope
- Snowcast full submission: due Tuesday, Jan 10

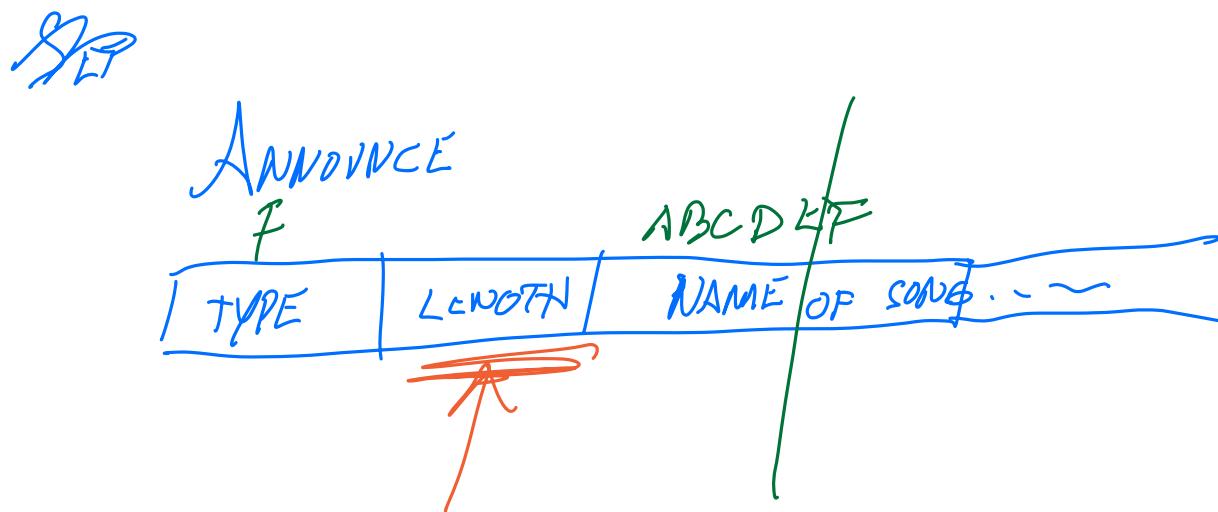
## Last call for registration stuff

- If you received an override, make sure you accept it. [Check now!](#)
- If you want to be enrolled but are not, [email me ASAP](#)

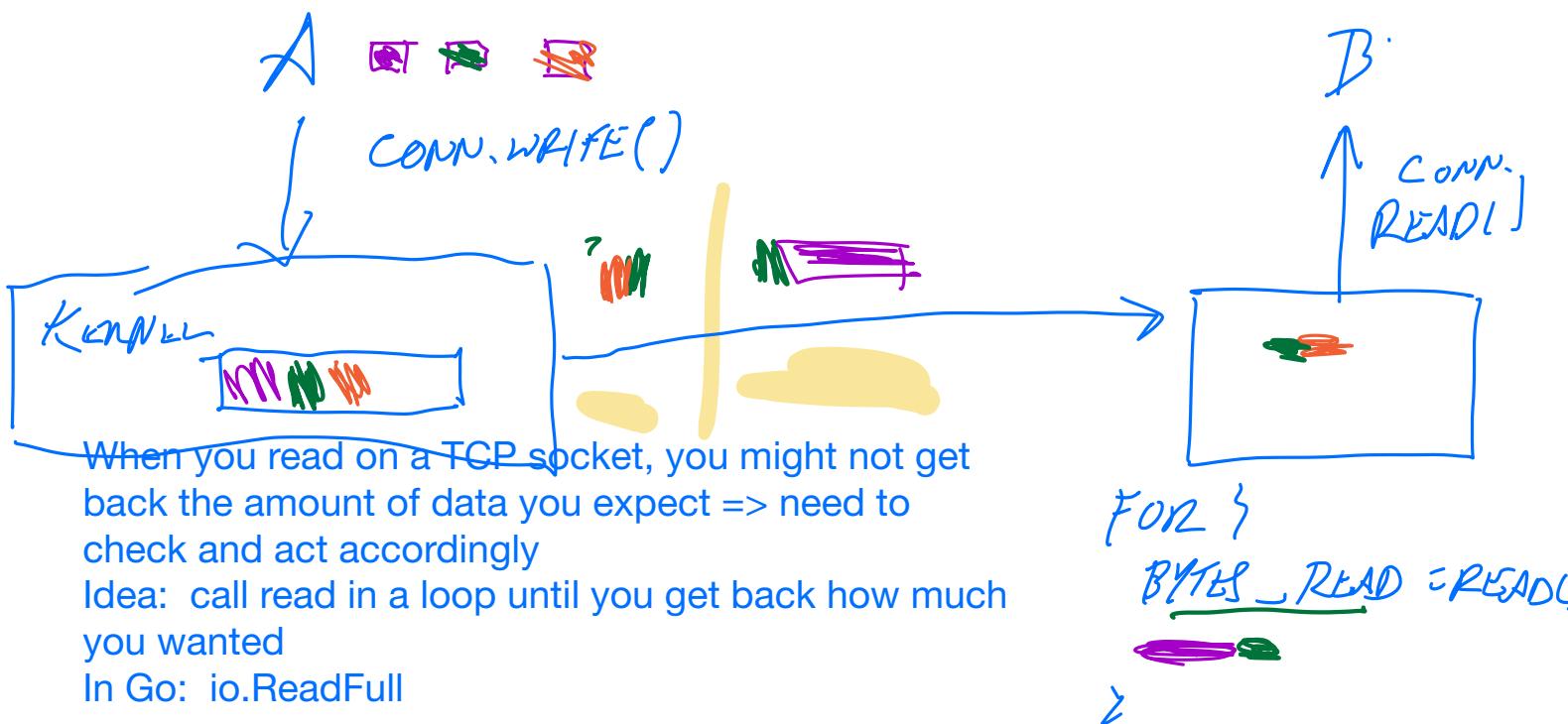


In TCP, conn.Read returns when ANY AMOUNT of data is available  
 => MAY NOT be the amount you want

Solution: keep reading until you get the amount of data that you want io.ReadFull



# SENDING DATA w/ TCP



TCP is designed to provide a STREAM of ordered data => it doesn't care about the separation of individual messages

WHAT HAPPENS IF YOU DO

THE SIZE OF THE MESSAGE?

$\Rightarrow$  PROTOCOL NEEDS TO BE SET UP  
SO THAT YOU CAN ALWAYS  
FIGURE OUT HOW MUCH DATA TO  
READ NEXT.

E6.

TYPE	LENGTH	STRING ..
------	--------	-----------

$\hookrightarrow$  LENGTH OF DATA TO FOLLOW  
MESSAGE TYPE  $\Rightarrow$  SIZE

# Today

- Two more things on sockets
- Physical/Link layer: how to connect two things  
=> Inherent properties of *real* networks

# Layers, Services, Protocols

	Application	Service: user-facing application. Application-defined messages
	Transport	Service: multiplexing applications Reliable byte stream to other node (TCP), Unreliable datagram (UDP)
	Network	Service: move packets to any other node in the network IP: Unreliable, best-effort service model
L2	Link	Service: move frames to other node across link. May add reliability, medium access control
L1	Physical	Service: move bits to other node across link

# Physical Layer (Layer 1)

Specifies three things:

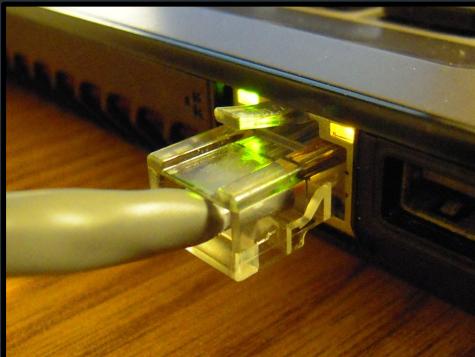
- Physical medium : *WIFI, ETHERNET, CELLULAR..~*
- Signaling/modulation : *How do I SEND 0 OR 1*
- Encoding: *How do you GET USEFUL INFO*



# Physical Layer (Layer 1)

Specifies three things:

- Physical medium: cable, fiber, wireless frequency
- Signaling/modulation: how to transmit/receive
- *Encoding*: how to get meaningful data



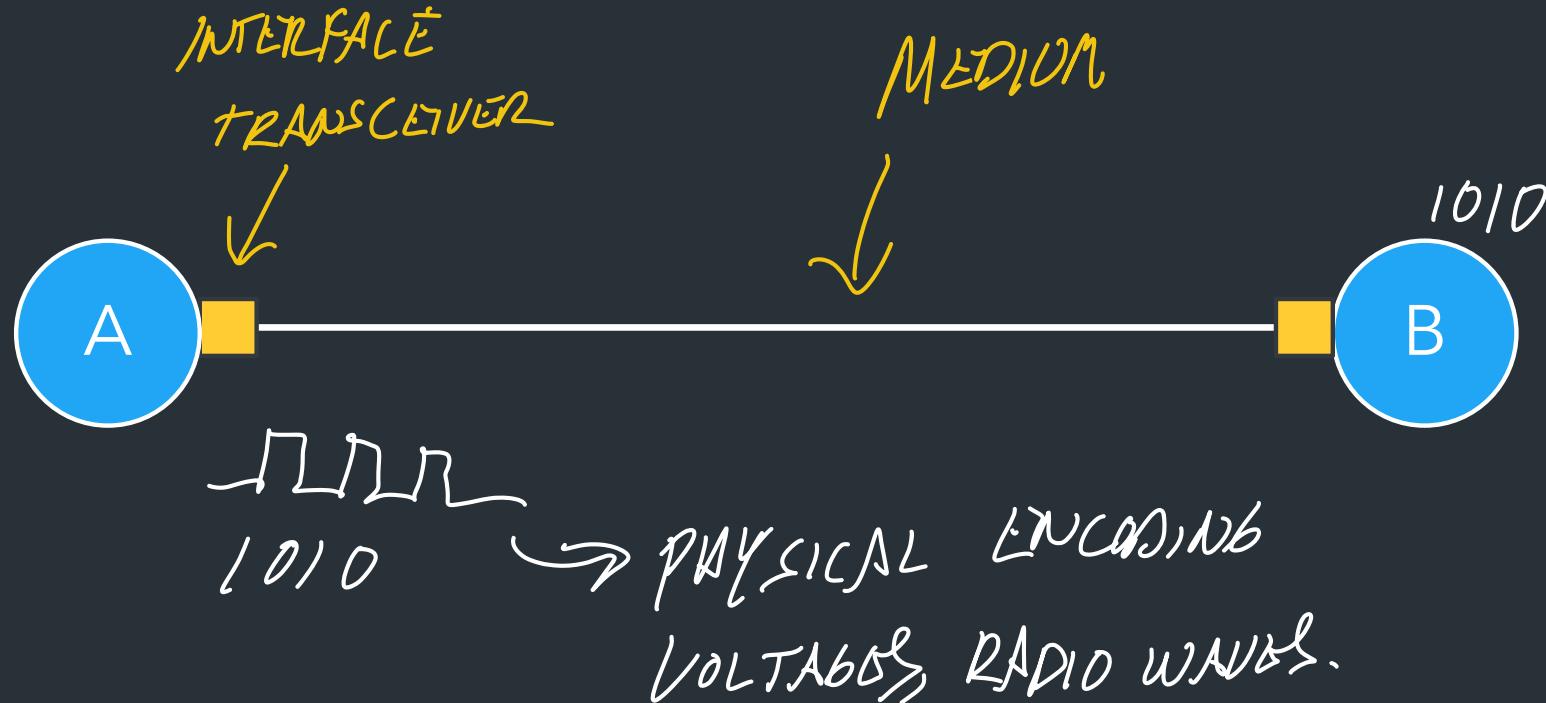
# Why should we care?

*This is the line between electrical engineering and computer science*

*Helpful to understand challenges involved  
=> How design/limitations affect our systems*

Also: Learn important principles we'll use elsewhere

# The main idea



Send/receive data over some kind of medium

Sender encodes message in some format, sends it "over the wire"

Receiver decodes message (knowing the format) => recover the message

# Why is this hard?

- Sharing channel: interference from other devices
- Noise
- Physical distance (attenuation)
- Energy usage
- Security
- ...

=> Every medium has its own characteristics, and problems

*We don't need to know the details.*

*However, there are some key takeways to help understand the challenges and implications*

# Key points

1. All media have fixed bandwidth => fixed "space" to transmit information
2. Sending data takes time! => latency
3. All media have (some) errors => how to deal with them?

# Bandwidth

---

**Bandwidth:** set of frequencies that a channel (medium) can propagate  
wall

(Hz)

PHONE / DIAL UP: 8 kHz

- Creates a fixed “space” in which data can be transmitted  
=>Wires: defined by physical properties  
⇒ Wireless: frequency ranges are regulated

Bandwidth gives an upper bound on throughput => amount of data we can send per time (bits / second)

Bandwidth: frequencies that a channel propagates well  
(Most signals made up of different frequencies)

- Creates a fixed “space” in which data can be transmitted  
=>Wires: defined by physical properties  
⇒ Wireless: frequency ranges are regulated

Upper bound on *throughput*: amount of data we  
can send per time (bits per second)

# UNITED STATES FREQUENCY ALLOCATIONS

## THE RADIO SPECTRUM

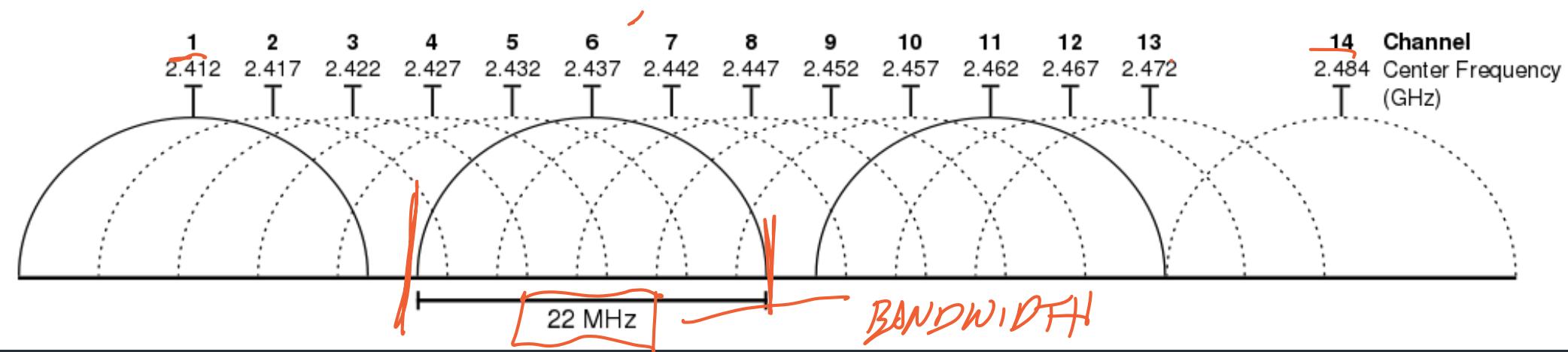


U.S. DEPARTMENT OF COMMERCE  
National Telecommunications and Information Administration  
Office of Spectrum Management  
JANUARY 2006

For the National Telecommunications and Information Administration, U.S. Department of Commerce, Office of Spectrum Management, Washington, D.C. 20585-0001, U.S. Government. All rights reserved. This document contains neither recommendations nor conclusions of the U.S. Government. It is the property of the U.S. Government and is loaned to your agency; it and its contents are not to be distributed outside your agency without the express written consent of the U.S. Government.

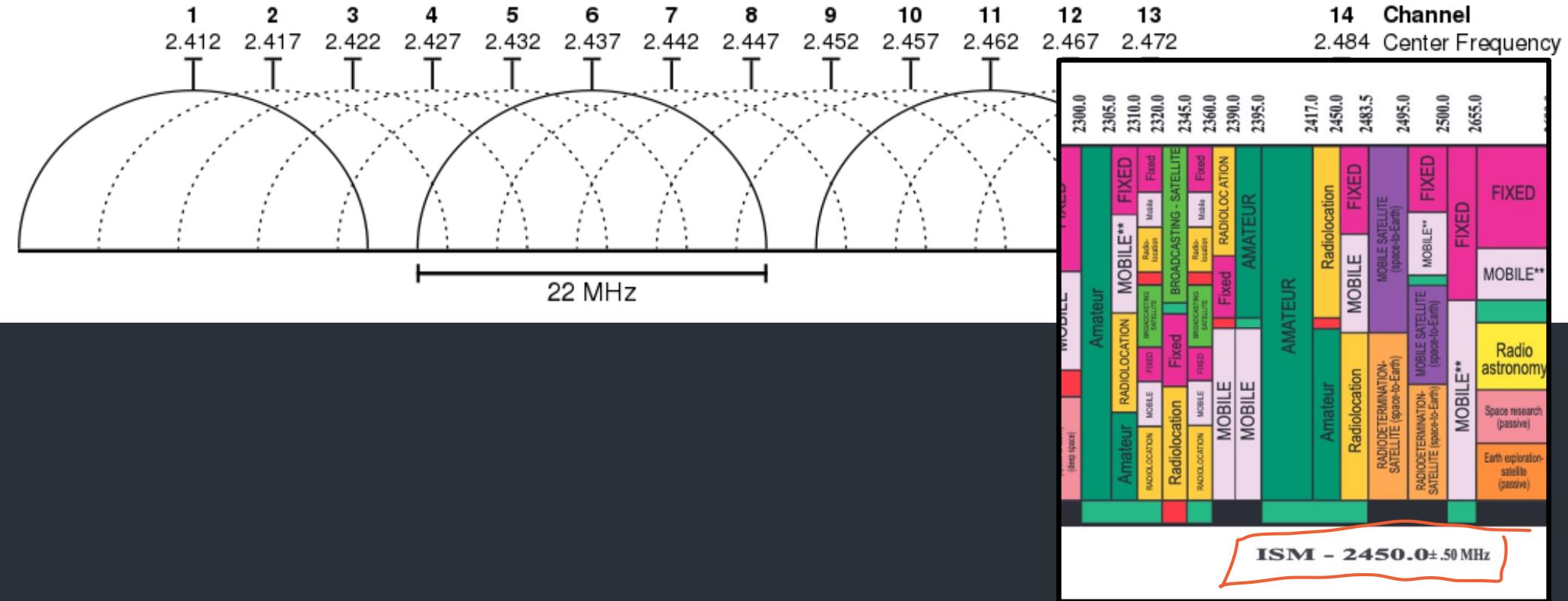
PLEASE NOTE: THE USE AND RELEASE OF THE INFORMATION CONTAINED IN THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF THE U.S. GOVERNMENT'S AGREEMENT WITH YOUR AGENCY.

# Early IEEE 802.11 (Wifi) channel bandwidth



2.46 Hz : "Unlicensed BAND"

# Early IEEE 802.11 (Wifi) channel bandwidth



Wi-Fi generations					V·T·E
Generation	IEEE standard	Adopted	Maximum link rate (Mbit/s)	Radio frequency (GHz)	
Wi-Fi 7	802.11be	(2024)	1376 to 46120	2.4/5/6	
Wi-Fi 6E	802.11ax	2020	574 to 9608 <sup>[41]</sup>	6 <sup>[42]</sup>	
Wi-Fi 6		2019		2.4/5	
Wi-Fi 5	802.11ac	2014	433 to 6933	5 <sup>[43]</sup>	
Wi-Fi 4	802.11n	2008	72 to 600	2.4/5	
(Wi-Fi 3)*	802.11g	2003	6 to 54	2.4	
	802.11a	1999		5	
(Wi-Fi 2)*	802.11b	1999	1 to 11	2.4	
(Wi-Fi 1)*	802.11	1997	1 to 2	2.4	
*(Wi-Fi 1, 2, and 3 are by retroactive inference)					<i>IEEE 802.11</i>

*How to actually send data?*

*(Within a limited bandwidth)*

# How to actually send stuff?

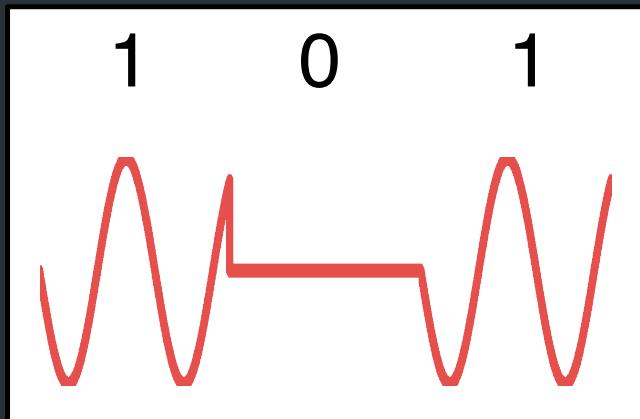
MOdulation: how to vary signal in order to transmit information



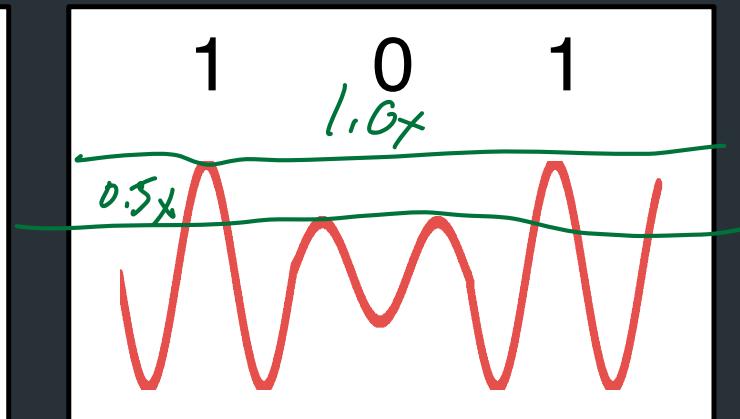
# One way: Use Carriers

Start with a *carrier frequency*, modulate it to encode data:

*OOK: On-Off Keying*



*ASK: Amplitude Shift Keying*



An (early) medium



Example: Bell 103 modem (c. 1960s)

Uses frequency-shift keying to encode data

Transmitting ("originating") side

-1 ("Mark"): 1270 Hz

-0 ("Space"): 1070 Hz

Receiving ("answering") side

- 1 ("Mark"): 2225 Hz

- 0 ("Space"): 2025 Hz



$\Rightarrow 56 \text{ KHz/sec}$

$\Rightarrow 300 \text{ Bits/sec}$

## Example: Bell 103 modem (c. 1960s)

Uses frequency-shift keying to encode data

Transmitting ("originating") side

-1 ("Mark"): 1270 Hz

-0 ("Space"): 1070 Hz

Receiving ("answering") side

- 1 ("Mark"): 2225 Hz

- 0 ("Space"): 2025 Hz

Throughput: 300 bits/second ("baud")



Transceiver: acoustic coupler

*Lots of innovation since then!*

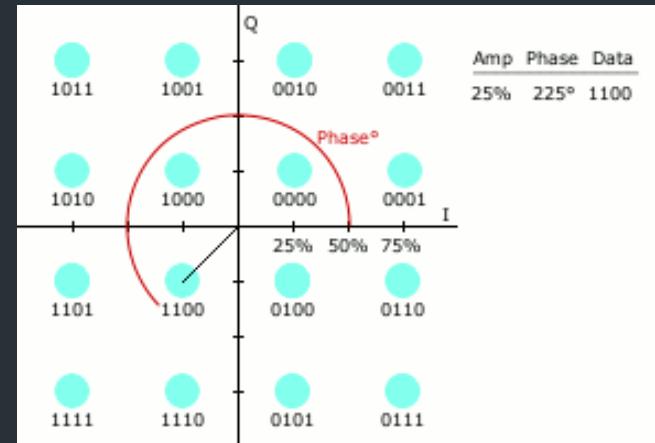
*But still some fundamental limitations  
of the medium...*

# This can get more complex...

Lots of engineering you can do

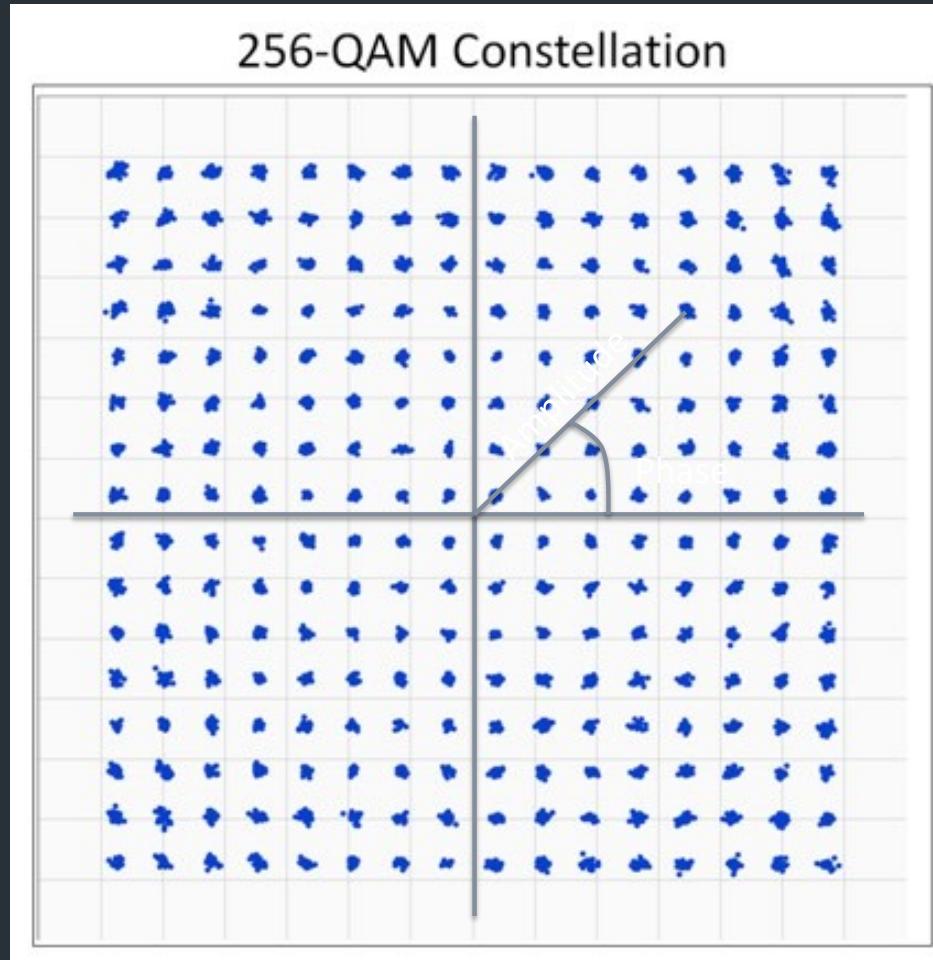
- Multiple carriers/frequencies
- Adjust amplitude, phase
- Clever ways to avoid errors
- ...

QAM: Quadrature Amplitude Modulation



A good animation on Wikipedia

# Example: Quadrature Amplitude Modulation (QAM)



# Modulation schemes in action

- <https://www.youtube.com/watch?v=vvr9AMWEU-c>

# Sounds great, right?

- Problem: noise limits the number of modulation levels (M)



# Sounds great, right?

- Problem: noise limits the number of modulation levels (M)

Shannon's Law:  $C = B \log_2(1 + S/N)$

- C: channel capacity (throughput) in bits/second
- B : bandwidth in Hz
- S, N: average signal, noise power

The amount of data we can fit in a channel is limited by the bandwidth, and the amount of noise in the medium

# Sounds great, right?

- Problem: noise limits the number of modulation levels (M)

Shannon's Law:  $C = B \log_2(1 + S/N)$

- C: channel capacity (throughput) in bits/second
- B : bandwidth in Hz
- S, N: average signal, noise power

**Takeaway:** fundamental limit on how much data we can fit into a fixed channel, based on noise

=> For any medium, designers create encodings to try and maximize throughput

Medium	Bandwidth	Throughput
Dialup	8 kHz	56 Kbit/s
Early Wifi (802.11g)	20 MHz	54 Mbit/s
Modern Wifi (802.11ax)	20-40 MHz	Up to 9 Gbps
Ethernet	62.5 MHz (1Gbps version)	1Gbit/s (common) Up to 100Gbps
3G cellular	Depends on carrier	2 Mbit/s
5G cellular	Depends on carrier	> 1 Gbps

=> Does this mean wifi is the best?

$10^{-9}$   
RANGE  
INFLUENCE

$10^{-12}$

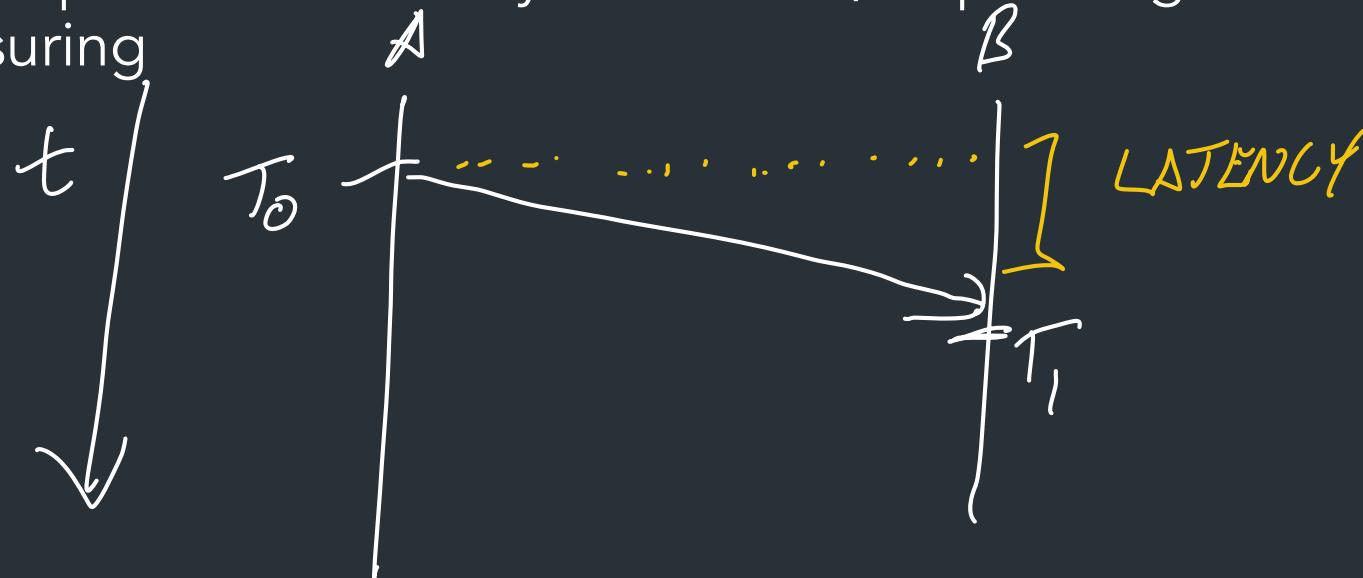
**Next few pages are extra material we'll discuss later!**

# Latency

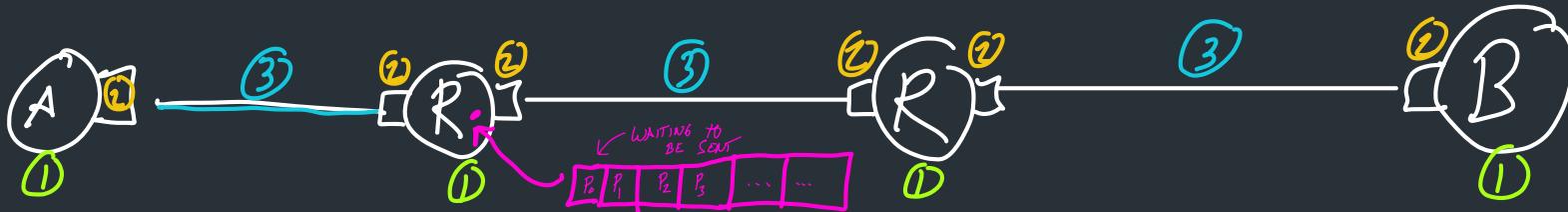
---

# Sending data takes time!

- Latency: time between sending data and when data arrives (somewhere)  
Sending data is not instantaneous (for many reasons)
- Multiple components => many definitions, depending on what we're measuring

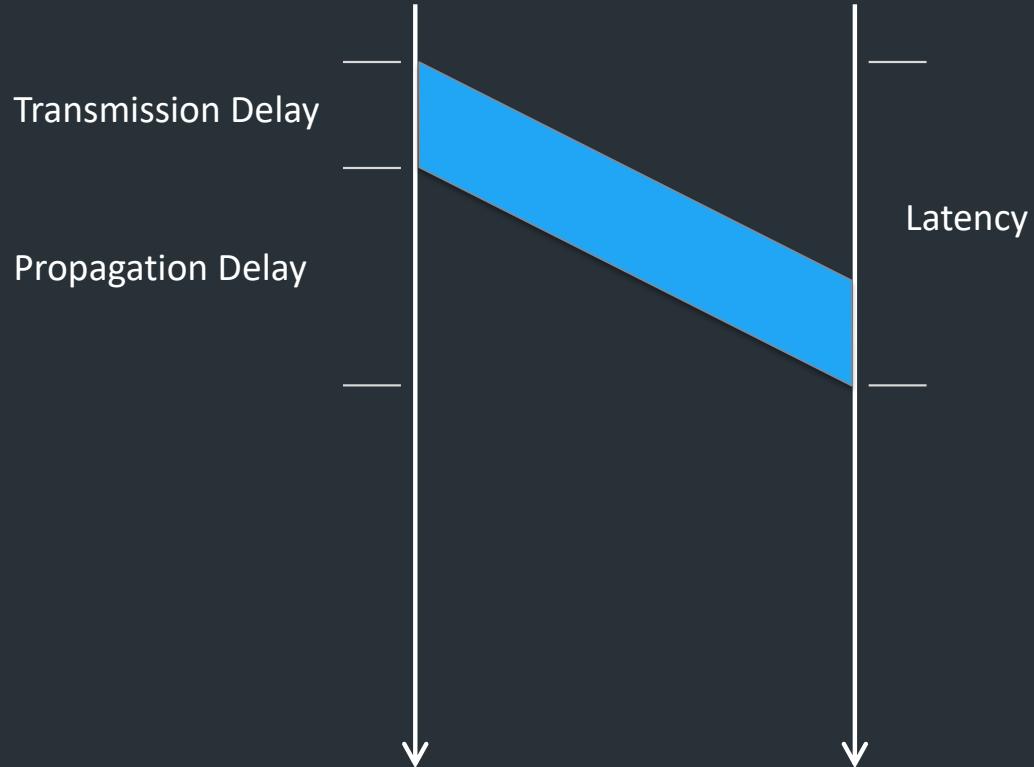


# How to think about latency



- ① Processing delay: Time for data to be sent by a node  
=> Time for OS to decide to send data
- ② Transmission delay: Time for transceiver to actually send bits on wire
- ③ Propagation delay: Time for data to propagate across wire => speed of light
- ④ Queuing delay: time data spends in buffers waiting to be sent (due to congestion, etc.)

# How to think about latency

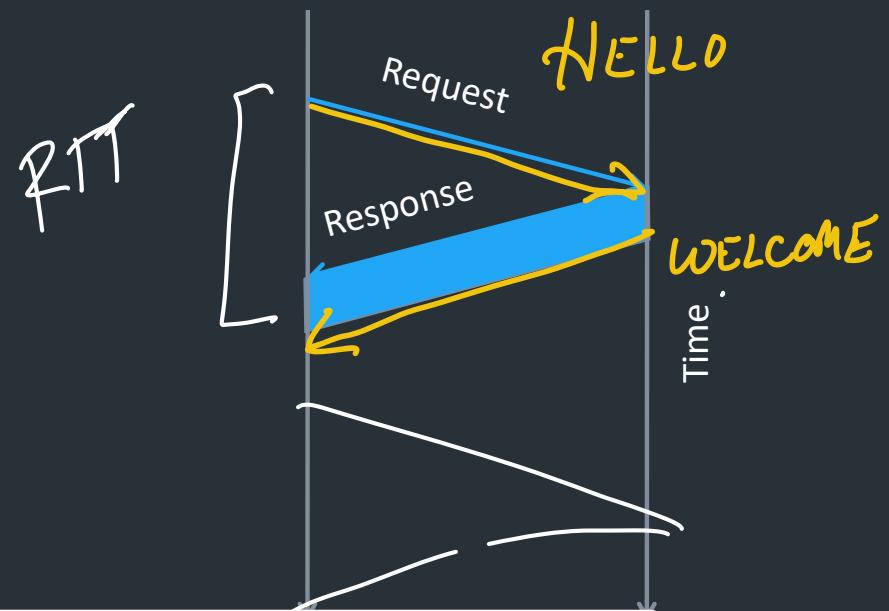


# How to think about latency

- Processing delay at the node: per message computation
- Queuing delay: time spent waiting in buffers
- Transmission delay: sending out the actual data
  - Size/Bandwidth
- Propagation delay: time for bits to actually go out on the wire
  - Upper bound?
  - Depends on media, ultimate upper bound is speed of light

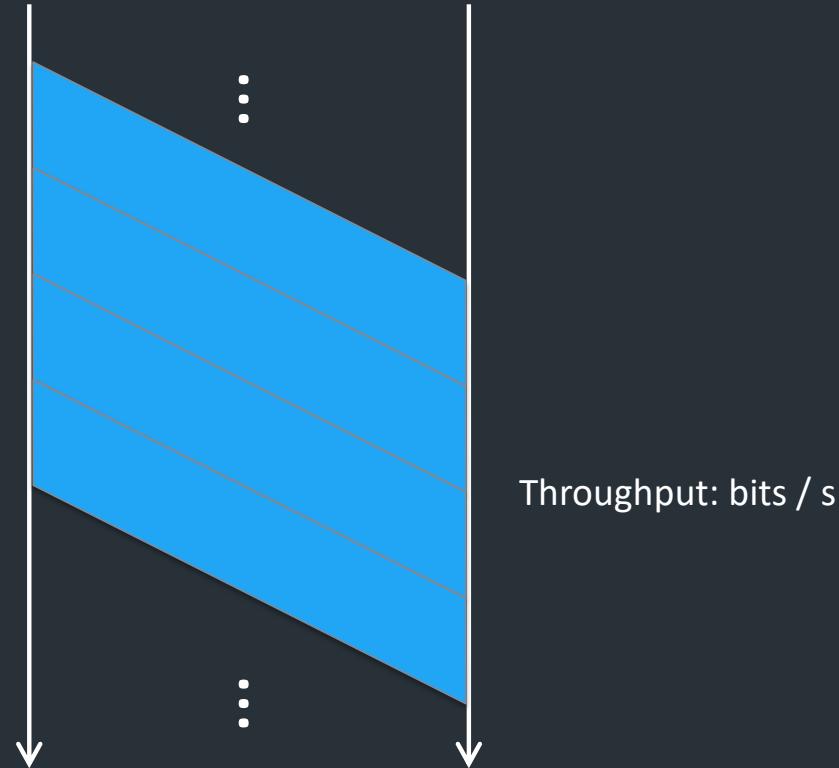
Round trip time (RTT): time between request and response

When we design protocols,  
can think about performance  
based on number of RTTs



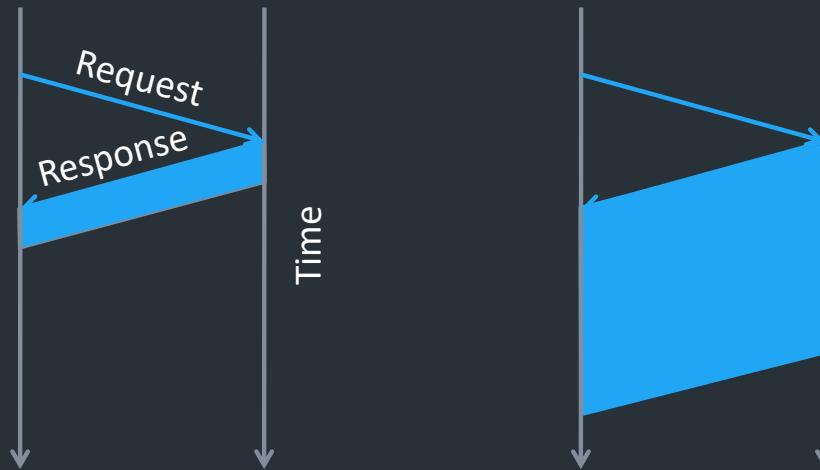
=> Not just about the physical layer!

# Sending Frames Across



# Which matters most, bandwidth or delay?

- How much data can we send during one RTT?
- *E.g.*, send request, receive file



Often: For small transfers, latency more important,  
for bulk, throughput more important

# Performance Metrics

- Throughput: Number of bits received/unit of time
  - e.g. 100 Mbps
- Goodput: *Useful* bits received per unit of time
- Latency: How long for message to cross network
- Jitter: Variation in latency

*Dealing with errors*

# Error Detection

- Basic idea: use a checksum
  - Compute small check value, like a hash of packet

# Error Detection

- Basic idea: use a checksum
  - Compute small check value, like a hash of packet
- Good checksum algorithms
  - Want several properties, e.g., detect any single-bit error
  - Details later

=> Not all protocols do this. Why?

# Approximation of a Square Wave

