CSE 461

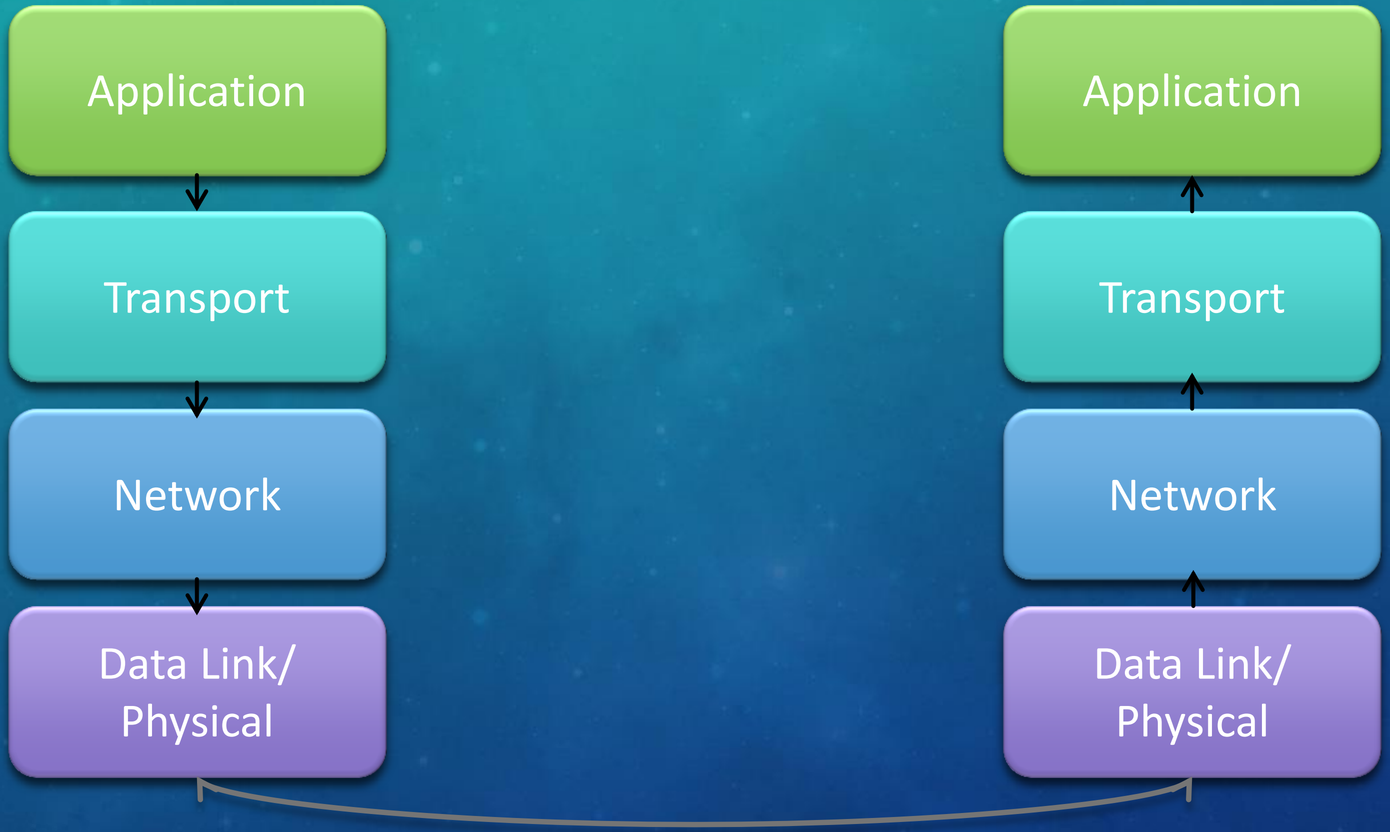
MIDTERM REVIEW

HELP YOURSELF TO SNACKS

MIDTERM OVERVIEW

- 50 minutes
- Closed book, closed notes
- Covers topics in lectures, projects and homeworks
- Not intended to test things you can easily look up
  - If something seems like you could Google it in a second, ask JZ
- Mixture of straightforward questions and conceptual thought questions
- **Bonus Question**: name all three TAs for CSE 461
  - Keith, Patrick, and Nat

NETWORK LAYERS & ENCAPSULATION



APPLICATION LAYER

Application
Application

- Used by applications
- Protocol is arbitrary

TRANSPORT LAYER

- Involves packaging of data for transport
- UDP/TCP and ports

NETWORK LAYERS & ENCAPSULATION

Network

Network

- Handles issues related to routing on the network
- Data treated as packets

DATA LINK/PHYSICAL LAYERS

Data Link/ Physical
Data Link/ Physical

- Data link layer
  - Puts data onto the actual line
  - Error-correcting codes to account for line noise are in the data link layer
    - At this level, data consists of frames
- Physical layer
  - Actual electrical or wireless oscillations

ADDRESSING

- MAC addresses
- IP addresses
- Ports
- Sockets

MAC ADDRESSES

- 48-bit
- Identify instance of specific network interface hardware
- **Bonus Question** : can you guess what device 40:f4:07:a0:21:07 is?
  - Nintendo Wii U NIC (40:f4:07 belongs to Nintendo)

IP ADDRESSES

- 32-bit (in IPv4) or 128-bit (in IPv6)
- Identify a host on a network
- Can change dynamically
- **Bonus Question** : do you know what 8.8.8.8 is?
    - Google DNS IP address (8.8.8.0-8.8.8.255 belongs to Google)

PORTS

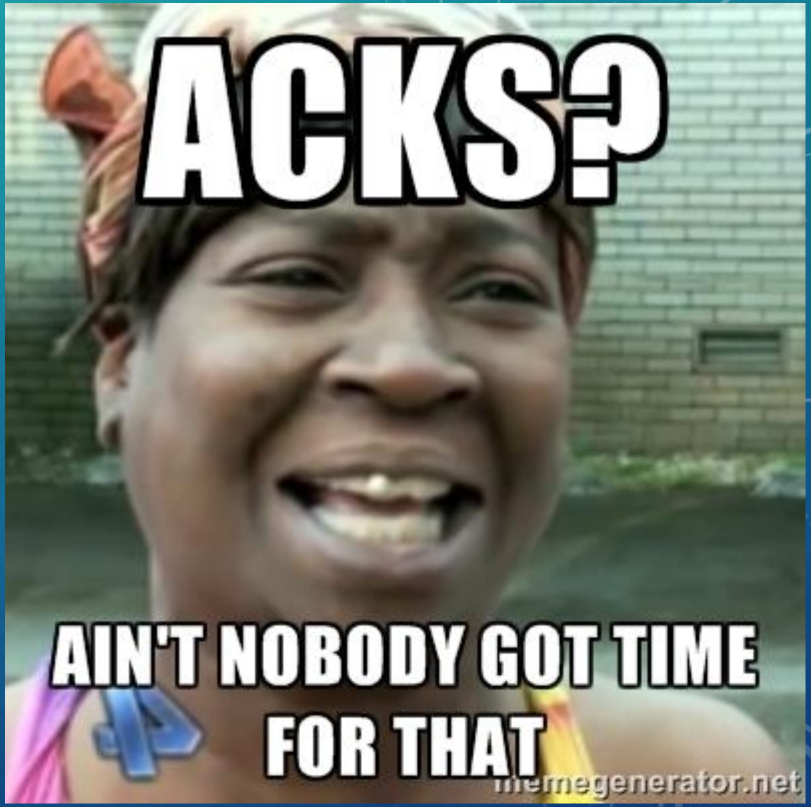
- 16-bit
- Identify communication channels on a specific host
- Often map to applications
- **Bonus Question** : what application uses port 1214?
    - Kazaa

SOCKETS

- Programming interface for networking
- Most common implementation is Berkeley sockets
- Allows data to be sent with file descriptor-like structures
- **Bonus Question** : name two valid *type* arguments you can specify for a socket
    - SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, SOCK_RAW

UDP VS. TCP

UDP	TCP
Unreliable	Reliable
Connection-less	Connection-oriented
No acknowledgements	Acknowledgements
No flow control	Sliding window
No sequence numbers	Sequence numbers



UDP

METRICS

- Bandwidth
- Latency
- Throughput, goodput
- Channel utilization
- Shannon’s theorem
- Nyquist rate

FREQUENCY & BANDWIDTH

- Frequency: rate of an oscillation
- Bandwidth: measures the width of a range of frequencies
- Bandwidth = $\text{freq}_{\text{upper}}$ - $\text{freq}_{\text{lower}}$
- Human hearing bandwidth: ~20kHz (20kHz - 20 Hz)
- “Bandwidth” and “bitrate” are often used interchangeably; this is a different definition
- **Bonus Question**: what’s the frequency range and bandwidth of 802.11 b/g?
  - 2.4 GHz to 2.5 GHz; 100 MHz

LATENCY

- Time between source and destination
- Shortest possible latency bounded by *c*
- Ping can measure round-trip latency
- Why might latency vary between ping tests?
- **Bonus Question**: what's the round-trip time to Voyager 1, at 19 billion km from Earth? Light speed is 300,000 km/s.
  - 126,666 s, or just over 35 hours

THROUGHPUT & GOODPUT

- Throughput: measures how much data can be sent in a given time period
- E.g., 100 Gbps
- Goodput: excludes protocol bits and retransmitted data packets
- What factors might cause goodput < throughput?
    - Protocol overhead
    - Dropped or corrupted packets
    - Flow control

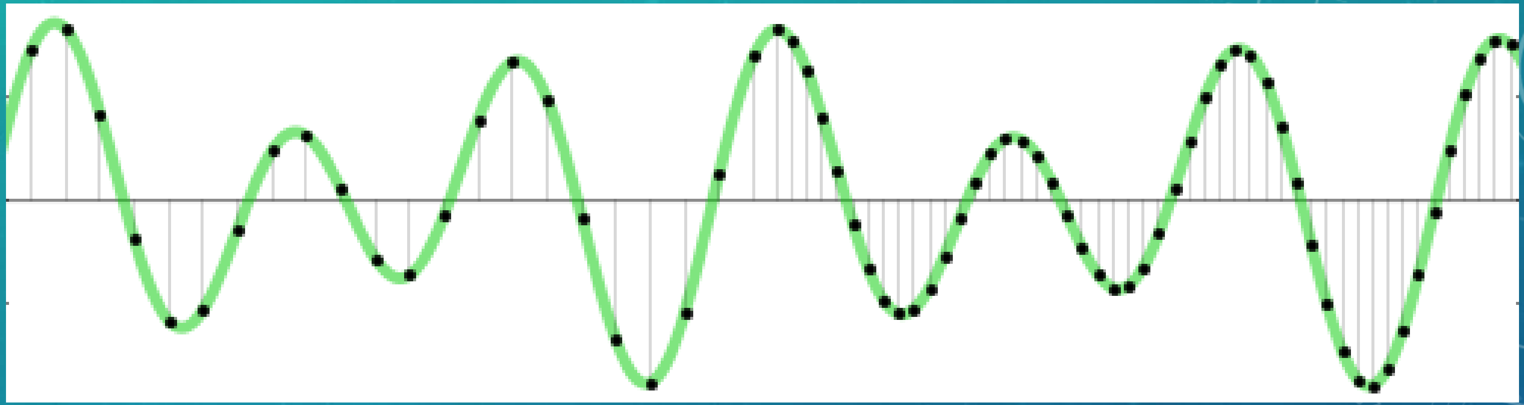
CHANNEL UTILIZATION

- Calculates how much of the channel is being used
- Bandwidth-delay product = throughput * latency
    - Kind of like: Volume of a pipe = area * length
- Channel utilization = (data in flight at any given time) / (bandwidth-delay product)
- If you're using stop-and-wait and only sending 1KB at a time over a 1MBps channel with latency of 10s, what's the channel utilization?
    - 1 / (1024 * 10) = ~0.01%

SHANNON THEOREM

- Tells about maximum bitrate in the presence of noise
- Capacity = bandwidth * $\log_2(1 + \text{signal/noise})$
- $C = B \log_2(1 + S/N)$
- What are the implications of this?

NYQUIST RATE



- To recover a waveform, the sampling rate must be at least two times the highest frequency
- Telephone sampling rate is 8kHz; what are the implications of this?
- What sampling rate would be required to recover all frequencies audible by humans? (Up to 20kHZ)
  - Audio CDs use 44.1kHz sampling rates for this reason

NETWORK-RELATED STORY TIME

"At this point the only solution that might work would be to create a different thread and initialize the network on that thread, still this will take us a significant amount of time and significant resources... We're afraid that the thread creation will also have a blocking impact and also the network initialization will take even longer to complete…"

HTTP

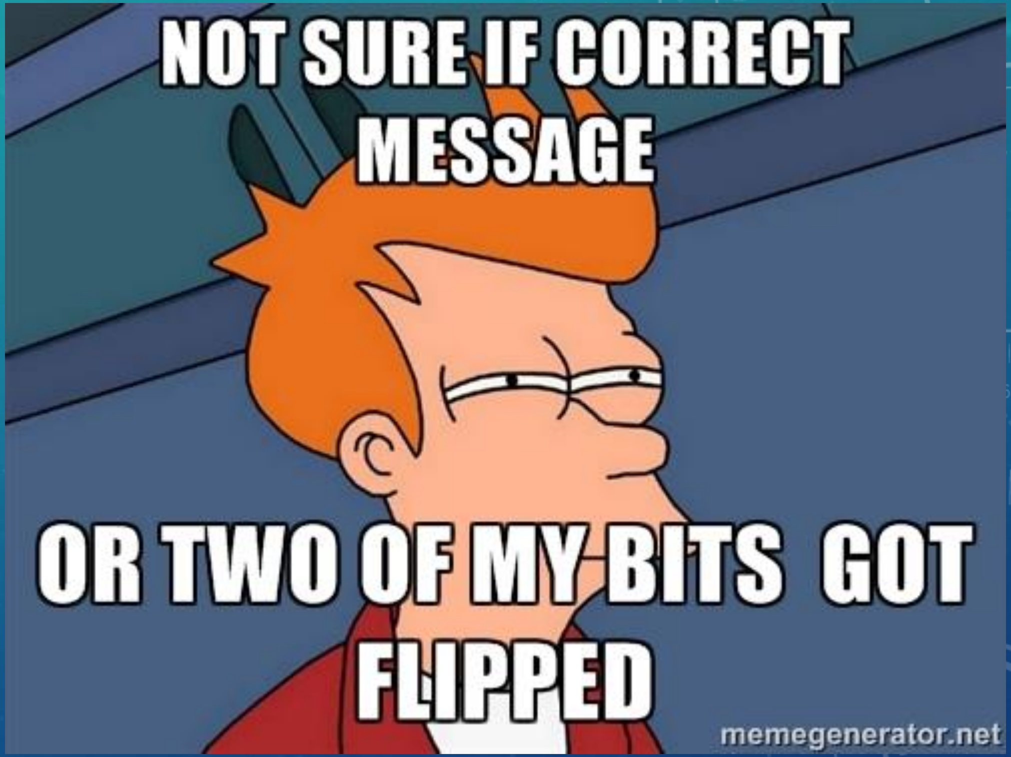
- HTTP 1.0
  - Initial connection over TCP acts as a preamble
  - Content-length can designate payload end
    - Bad for streaming
    - Alternative: drop the connection!
  - Caching used heavily
- HTTP 1.1
  - Data comes as a stream, chunked into defined lengths
  - Connections are reused, reducing overhead
  - Some pipelining possible, but limited (HOL blocking)
- HTTP 2.0
  - Reduces latency through compression
  - Allows asynchronous sending/multiplexing

ERROR HANDLING

- Parity bits
- Hamming codes
- Checksums
- CRCs
- Bursty errors and error locality
- MITM attacks
- Cryptographic hashing
- Digital signatures

PARITY BITS

- Bits check parity on a set of bits
- Even parity: bits add to 0
- Odd parity: bits add to 1
- Multiple parity bits (on odd bits/ on even bits, etc.) can increase effectiveness
- What parity bit would need to go in the x to achieve even parity?
  - 0010101x



HAMMING CODES

- An extension of bit parity, where parity check bits are in “powers of two” positions

  -
    ```
    Bit string: 0   0   1   0   0   0   0   1   0   0   1
    Bit number: 1   2   3   4   5   6   7   8   9   10  11
    Par/msg:    p   p   m   p   m   m   m   p   m   m   m
    ```
- Each data bit is checked (with even parity) by check bits that make up its “power of two” sum
  - E.g., for data bit 7, add to sum for parity bits 4 + 2 + 1
  - Possible to recover from single errors by reading check bits in reverse order and putting a 1 for each bit that is incorrect (this is the “error syndrome”)
- Hamming distance: minimum number of bit flips necessary to change one string into another

CHECKSUMS

- Several algorithms:
  - One studied in class added all words in data as unsigned numbers, allowing to overflow
  - Sum was then compared to check data integrity
- What are the possible problems with this? How many bit flips does it take to break it?
  - Susceptible to lots of different types of bitflip errors
  - Only takes 2

CRCS

- Like a checksum, but better
- Somewhat complex polynomial algorithm
- Good for detecting bursty errors

SUPER-BASIC SECURITY

- Man-in-the-middle attacks
- Cryptographic hashing
  - Very, very hard to reverse
- Digital signatures
  - Public/private key encryption

OTHER TOPICS

- End-to-end argument
  - Packet ordering and flow control shouldn't be done by the network; it's wasted work
  - What's the counterargument?
- Sliding window protocol
  - Receiver must receive a certain minimum number of segments before sender can send new data
  - Used in TCP
- Cumulative ACKing
  - ACKing a sequence number means you've received **all** data preceding that sequence number

PROJECT 1 QUESTIONS

- Could you use cumulative ACKs without breaking the protocol? Would it be useful?
- Why use headers?
- Why does the registration server need to be idempotent?
  - Idempotent: can be applied multiple times without changing the result beyond the initial application

SAMPLE EXAM QUESTIONS

SAMPLE QUESTION

Ethernet and 802.11 both support multiple data rates. When an Ethernet cable is plugged into the device, it communicates with the other end, chooses a rate to use, and then sticks with it. 802.11 devices, however, continuously talk to the AP to choose a specific rate to use for the next short while, adjusting that rate up and down as they please.

Why is it a good idea for 802.11 to repeatedly choose transmission rates? Why is it **not** a good idea for Ethernet to do this? (Discuss with someone near you.)

ANSWER

The 802.11 signal-to-noise ratio can change dramatically over time, which strongly affects the possible transmission rates. Unless 802.11 adapted, it would have to choose between wide coverage at low rates and high rates at low coverage. Dynamic adaptation lets it try to achieve both. Ethernet operates in a much more constrained environment, with strict limits on signal quality imposed by the specification. The environment does not change dynamically.

SAMPLE QUESTION

A sender and receiver are using the sliding window protocol, with cumulative ACKs. The receiver has a bug where it repeats the last ACK it sent whenever it hasn't received a data frame in the last 10 msec. The connection is bad enough that some data and ACK frames may be lost. Will the sender and receiver achieve reliable transmission? Will this protocol fail? ? (Discuss with someone near you.)

ANSWER

They'll achieve reliable transmission, assuming no bugs other than those described. The extra ACKs will simply repeat a previously sent ACK. They won't affect the sender's window; the sender will view them as duplicate ACKs and drop them.

SAMPLE QUESTION

In Project 1 your code sent UDP packets from a client to a server. If we were to look at the bits actually being carried on the wire (assuming we're using a wired network), we'd find the destination IP address and port were part of the bits being sent. Did the bits on the wire also carry a destination MAC address, or not?

ANSWER

Yes. All data delivery happens at the link layer, which uses MAC addresses. Higher level protocols, like UDP, are encapsulated in link layer frames.

SAMPLE QUESTION

Sliding window protocols specify a sending window and a receiving window. Can it ever be useful for the sending window to be larger than the receiving window? Briefly explain your answer.

ANSWER

Yes, possibly. The sending window limits the amount of data that can be in flight but unacknowledged. The receiver window is a limit on the memory available for buffering and reordering. If the receiver is capable of consuming each incoming frame basically as it arrives, it could have a very small receive window. In that case, the send window could be larger than the receive, to allow frames to be in flight (on the wire).

ADDITIONAL STUDY SUGGESTIONS

- Read through your project 0 & 1 code and diagram what it's doing
- Review HW problems; do similar problems
- Watch David Wetherall's Coursera course videos
- Review old midterms
  - Available on CSE site (but cover somewhat different material than what ours will)

ANY QUESTIONS?