# CSE 461

FINAL EXAM REVIEW

# HELP YOURSELF TO SNACKS

# FINAL OVERVIEW


- Online “take-home” final
- Starts Sunday; due at noon on Wednesday
- Open book, open notes, open internet, but not open people
- Covers topics in lectures, sections, projects, and textbook/homework
- Questions about the test should be sent to JZ/TAs by e-mail
- When appropriate, we’ll post on Catalyst discussion board (check often!)
- Bonus Question: What are my office hour times?
  - Tuesdays, noon to 2PM (but not anymore!)

# SOME MAJOR TOPICS

- Network layers & encapsulation
- Types of addresses
- Ports and sockets
- HTTP 1.0/1.1/2.0
- TCP vs. UDP
  - Sliding window and sequence numbers
  - Congestion control
- Error handling and checksums
- TLS
- NAT
- BGP, MPLS
- IPv4 vs. IPv6
- Routing algorithms
- All of the projects

# PROBLEMS TO EXPECT ON THE FINAL

- Lots of protocol questions
  - Why does [protocol] do [thing]?
  - Would [protocol] work if it didn't do [thing]?
  - How does [protocol] achieve [goal]?
- Lots of hypothetical, long-form questions to see how you think
- Maybe some multiple-choice
- Coding problem (do individually!)

# NETWORK LAYERS & ENCAPSULATION

# APPLICATION LAYER


- Used by applications
- Protocol is arbitrary

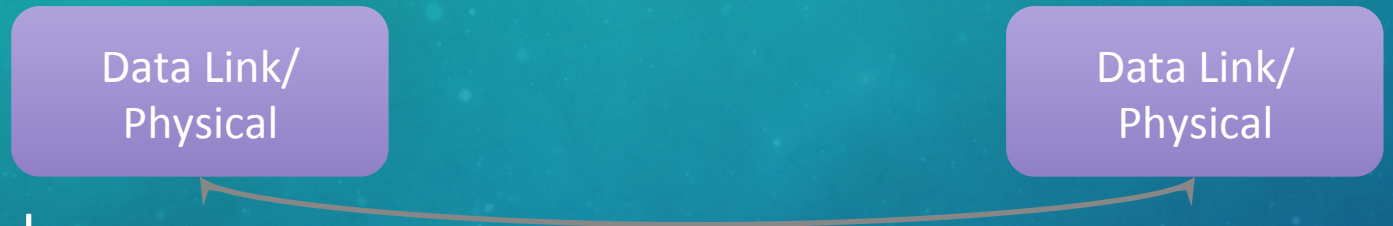
# TRANSPORT LAYER


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

# NETWORK LAYERS & ENCAPSULATION


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

# DATA LINK/PHYSICAL LAYERS




- 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

# Bonus question:

This layered network model is often called the _____ model. (Three-letter acronym)

- OSI (Open Systems Interconnection) model

# ADDRESSING

- MAC addresses
- IP addresses
- Ports
- Sockets (file descriptors)
- Other types?

# MAC ADDRESSES

- 48-bit
- Identify instance of specific network interface hardware
- E.g., 00:0a:95:9d:68:16

# IP ADDRESSES

- 32-bit (in IPv4) or 128-bit (in IPv6)
- Identify a host on a network
- Can change dynamically
- E.g., 173.250.157.38

# PORTS

- 16-bit
- Identify communication channels on a specific host
- Often map to applications

# SOCKETS

- Programming interface for networking
- Most common implementation is Berkeley sockets
- Allows data to be sent with file descriptor-like structures

# UDP VS. TCP

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

# GETTING ADDRESSES WITH ARP AND DHCP

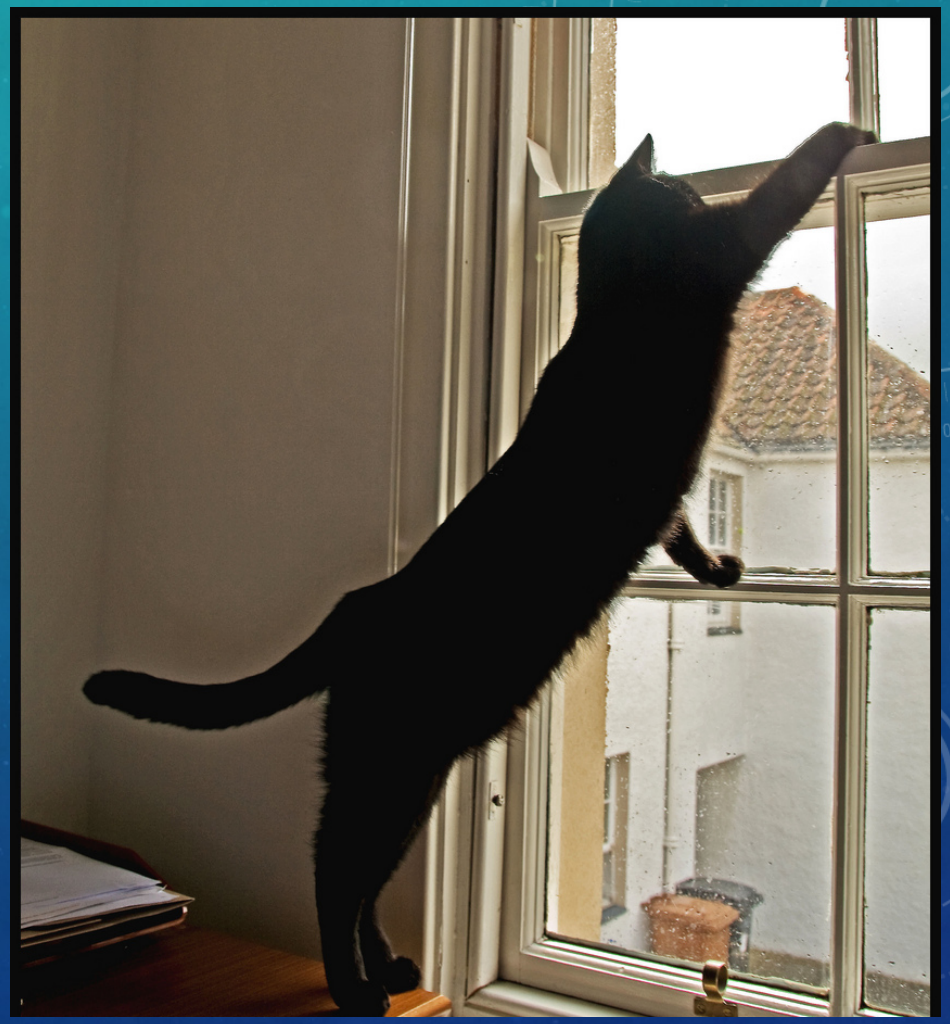
- ARP (Address Resolution Protocol) allows hosts to convert an IP address to a MAC address
- DHCP allows hosts that’ve just joined a network to receive an IP address
- **Bonus Question**: how could a hacker use ARP maliciously?
  - By ARPing IP addresses it hears with its own MAC address, or with non-existent MAC addresses.

# HTTP

- HTTP 1.0
  - Initial connection over TCP acts as a preamble
  - Content-length can designate payload end
    - Bad for streaming
    - Put the content-length in the end of the payload → hard to cache on the receiver side
  - Caching used heavily
- HTTP 1.1
  - Data comes as a stream, chunked into defined lengths (tokenized)
  - Connections are reused, reducing overhead
  - Some pipelining possible, but limited
- HTTP 2.0
  - Reduces latency through header compression
  - Allows asynchronous sending/ multiplexing over a single TCP connection
  - Fixed the head-of-line blocking problem in HTTP 1.x

# TCP SLIDING WINDOW

- All packets within window can be sent without ACKs
- ACKs must be received to move the window
- This is how TCP does “flow control”
- Why do we do this, though?

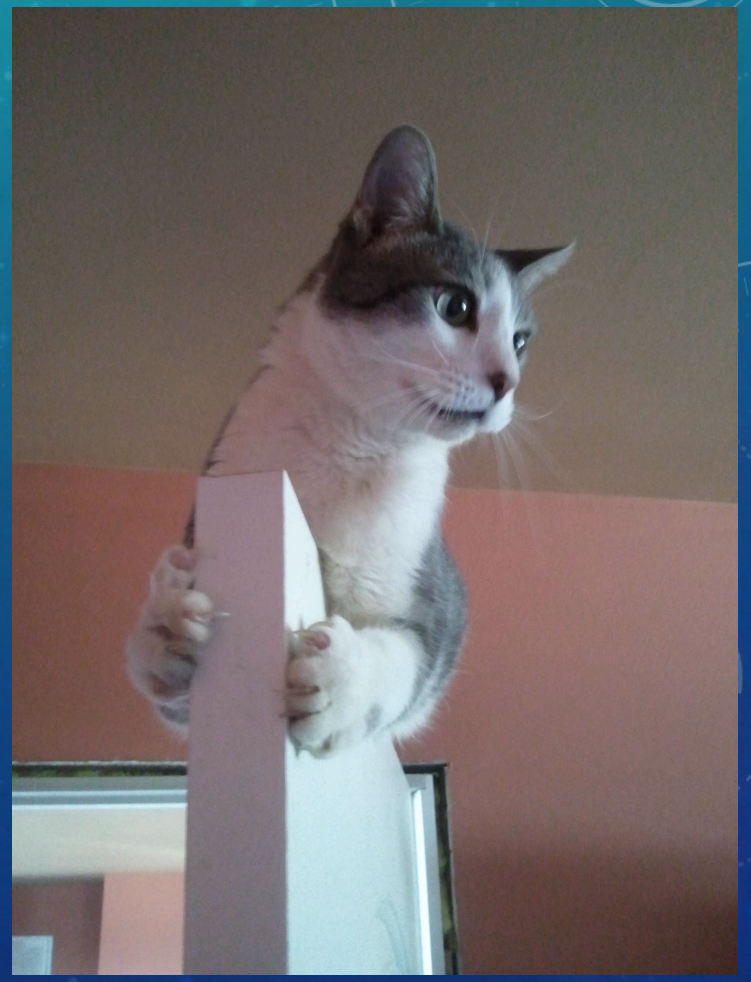


# TCP CONGESTION CONTROL

- Slow start
  - Window doubles in size on each ACK
  - On packet drop, window reduced to half its size, and grows by one segment per ACK

# ERROR HANDLING / INTEGRITY

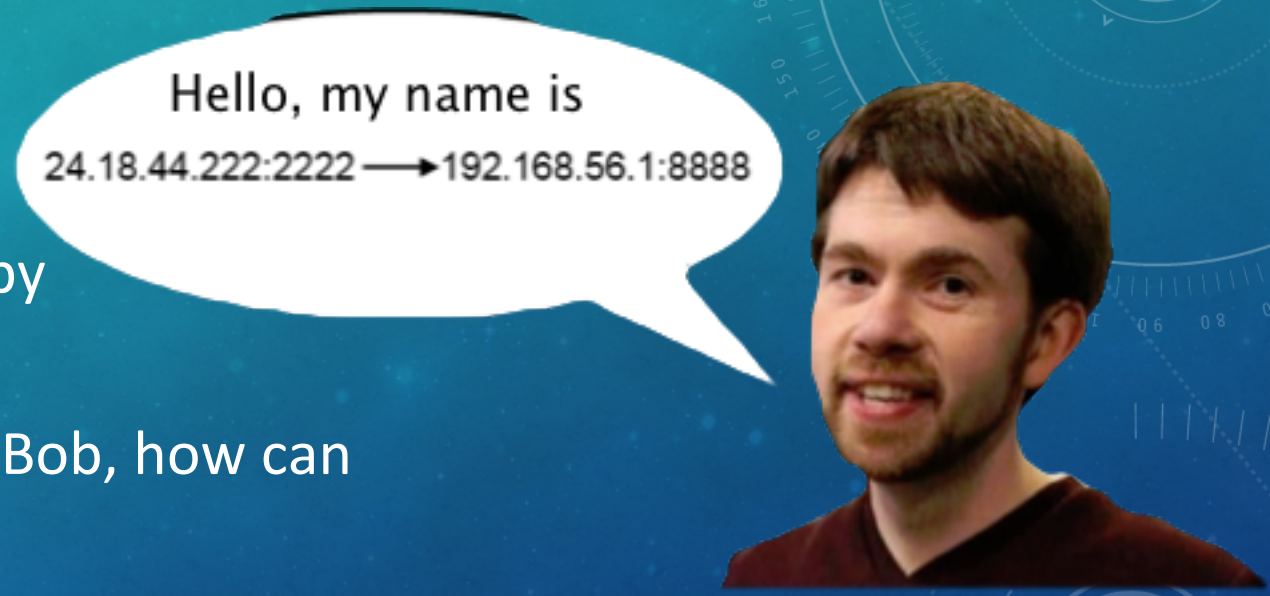
- Parity bits
- Checksums
- CRCs



# TLS/HTTPS

- HTTP CONNECT used to establish a “tunnel” for encrypted communication
- TLS handshake
  - Version data and crypto algorithm choices exchanged
  - Certificates exchanged (for authentication)
  - Secrets exchanged, encrypted using public keys
  - Both hosts generate shared keys and start communicating

# NETWORK ADDRESS TRANSLATION (NAT)



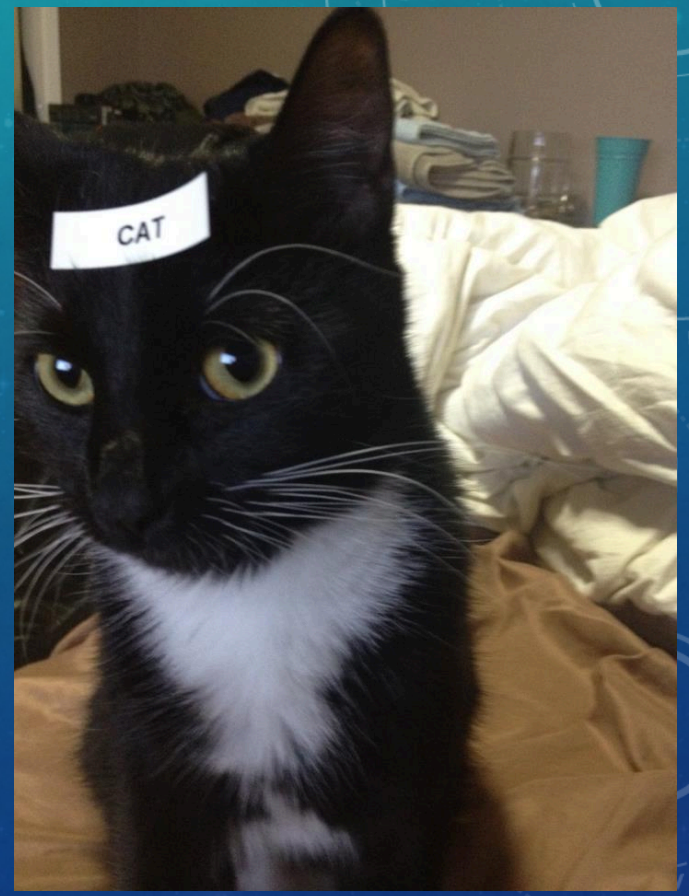

- Allows computers connected to a router with a single IP address to be addressed by an IP address/port pair
- **Bonus Question**: If Alice is connecting to Bob, how can Alice and Bob tell if Alice is on a NAT?
  - Alice sends a packet to Bob with a payload containing a hash of her own source IP address and port. Bob hashes the source IP address and port, and sees if that hash matches the payload hash.

# BORDER GATEWAY PROTOCOL (BGP)

- Deals with routes between “autonomous systems”
- Routers exchange information about routes to nodes
- Routers maintain a shortest path vector for other routers
- BGP must be manually configured; no discovery, and often based on policy

# MPLS

- MultiProtocol Label Switching
- “Labels” are added on top of link layer frames
- Routers look at, remove, and add labels
- Allow routers to route internally within “autonomous systems”
- Traffic class field determines Quality of Service priority
- Violates net neutrality!



# IPV4 VS. IPV6

- 32-bit vs. 128-bit
- Decimal vs. hex representation
- NAT vs. no NAT :(
- Weak multicast support vs. real multicast support
- **Bonus question:** what's the difference between broadcast, multicast, and anycast?
  - Broadcast: goes out to all hosts
  - Multicast: goes out to a subset of hosts
  - Anycast: goes out to a single member of a group of potential receivers

# ROUTING BASICS

- Distributed algorithm to determine efficient paths in a network
- Distributed approach
  - Distance vector algorithm
  - Link-state algorithm
  - Spanning tree algorithm

# CLASSLESS INTER-DOMAIN ROUTING (CIDR)

- Designates subnets within the IP address space
- IP address of subnet followed by the number of bits that are used for routing (32 minus the number of bits free for subnet addressing)
- Note: the first address in a subnet is used as the subnet address (seldom actually used), and the last address is the broadcast address.
- **Bonus Question**: What would the broadcast address be for the subnet 24.18.4.0/24?
  - 24.18.4.255

# DISTANCE VECTOR ALGORITHM

- Uses Bellman-Ford algorithm
- Each node maintains a table of the shortest path to each node through each node
- Nodes send their full routing table to their neighbors only
- When a node receives tables from its neighbors, it recalculates the shortest paths
- Algorithm stops when all of the routing tables have converged
- Requires lots of space for routing table storage

# LINK-STATE ALGORITHM

- Each node floods out packets identifying its neighbors and the metrics for the link with each neighbor (its “link state”)
- Nodes construct a map of network connectivity
- Nodes calculate the shortest path to every possible destination (usually with Dijkstra’s algorithm)
- These paths are put into a routing table
- Requires more processing power/logic to calculate link properties/metrics

# SPANNING TREE ALGORITHM

- Creates a loop-free connected graph for bridges to communicate
- Node with lowest MAC is root; low MAC addresses break ties
- Each node remembers shortest path to the root it sees

# SPANNING TREE POEM: “ALGORHYME” (BY RADIA PERLMAN, INVENTOR OF SPANNING TREE PROTOCOL)

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.
-Radia Perlman

# PROJECT 0: UDP COMMUNICATION

- How to send basic packets across the internet
- How to use the “names” for applications on the network (IP and port)

# PROJECT 1: REGISTRATION SERVER

- Like a DNS service (lets you specify a destination with a convenient name and gives you metadata about that)
- Useful discovery mechanism
- Protocol headers and parsing

# PROJECT 2: HTTP PROXY

- What an HTTP proxy does and how it works
- What HTTP packet format looks like (generally)
- Basic flow of HTTP request-response exchange
- Why we manipulated packets to turn off keep-alive
- Think about how an application can invisibly sit in the “middle” of a connection

# PROJECT 3: TOR61

- How Tor (or Tor61) works
- How nodes establish connections to each other
- How nodes extends work
- How Tor provides privacy
- Virtual circuits and protocol translation
- Possible causes for deadlock in networked systems like Tor

# THOUGHT PROBLEM: NETWORK LAG TROUBLESHOOTING

- Developer wants to improve laggy multiplayer game
- We do a packet capture and see that many packets aren't reaching the destination
- We also see hundreds of packets per second
- What's the likely problem, and what are some ways we can fix it?

# THOUGHT PROBLEM: TOR DESIGN

Why or how does Tor (or Tor61):

- Use streams?
  - What if we can't spare that header space? Is there an alternative?
- Prevent cycles from occurring?
  - Could a malicious router create infinite cycles in the network?

# THOUGHT PROBLEM: NETWORK PROGRAMMING

What makes network programming (and distributed programming) so much harder from standard local programming?

# THOUGHT PROBLEM: TCP RATE LIMITING

Suppose you wanted to try to speed up the rate at which servers send TCP data back to your machine, and you were willing to modify your machine's TCP implementation to do it. What could you do that would cause servers to send data back to you faster than they would if you used a legitimate TCP implementation on your machine?

# SAMPLE PROBLEM: CIDR

What is the maximum number of hosts that a subnet can handle, if its network prefix is 192.168.176.0/20 ?

# SAMPLE PROBLEM: CIDR (SOLUTION)

The routing prefix is 20, which means that there are 12 bits of data that is unmasked and can be used for the network. 12 bits of data can provide 2^12, or 4096, different hosts. (Or 4094, if you don’t count the first and last addresses.)

# TRIVIA COMPETITION (JUST FOR FUN)

- 16 questions total
- People with the top scores get prizes

# QUESTION 1

- What are the three non-overlapping 802.11 channels supported by most wireless devices?

# QUESTION 2

Which one of following is the correct sequence of functions to be called for an elementary server? Select one.

- A) Socket(), connect(), write(), read(), close()
- B) Socket(), bind(), listen(), accept(), read(), write(), close()
- C) Socket(), bind(), accept(), listen(), write(), read(), close()
- D) Socket(), listen(), accept(), read(), write(), close()

# QUESTION 3

- What was the first video game console with a built-in modem?

# QUESTION 4

- For most organizations, a class A network, with 16 million addresses is too big, and a class C network, with 256 addresses is too small. A class B network, with 65536 addresses is just right. What is the name for this problem?

# QUESTION 5

- To the nearest second, what is the RTT for a laser pulse from the Earth to the Moon and back?

# QUESTION 6

- What does HTTP Status 418 mean? It was developed for the facetious “Hyper Text Coffee Pot Control Protocol,” but never implemented.

# QUESTION 7

- Which combination of family and type of socket is used to create a UDP socket? (AF_XXXX, SOCK_XXXXX)

# QUESTIONS 8 THROUGH 16: ABBREVIATIONS

8. TCP
9. RFC
10. CRC
11. DHCP
12. WEP
13. CIDR
14. ICMP
15. DSL
16. STUN

# FINISHED!

- How did you do?

# ANSWERS

1. 1, 6, and 11
2. B
3. Dreamcast
4. Three Bears Problem
5. 3s (actually ~2.56s)
6. I'm a Teapot
7. AF_INET, SOCK_DGRAM
8. Transmission Control Protocol
9. Request for Comments
10. Cyclic Redundancy Check
11. Dynamic Host Configuration Protocol
12. Wired Equivalent Privacy
13. Classless Inter-Domain Routing
14. Internet Control Message Protocol
15. Digital Subscriber Line
16. Session Traversal Utilities for NAT

# JOKES

- How do you catch an ether bunny?
  - With an ether net!
- What did the ARP request order at McDonald's?
  - A big MAC
- Did you hear about the network engineer who broke both his arms?
  - He had to get a multi-cast.
- HTTP Error 413: That’s what she said
- Why did the IPv4 addresses take a nap?
  - They were exhausted!
- I'd tell you a joke about CIDR, but you're too classy.

# BEYOND 461: MORE COOL NETWORK STUFF (1)

- Learn how distributed systems work
  - 2013's CSE 552 is online:
  - http://courses.cs.washington.edu/courses/csep552/13sp/
  - http://courses.cs.washington.edu/courses/csep552/13sp/video/
  - BitTorrent is a really cool protocol; look it up!
- Build a wireless sensor network
  - This book is pretty good: http://shop.oreilly.com/product/9780596807740.do

# BEYOND 461: MORE COOL NETWORK STUFF (2)

- Learn about network security
  - Penetration testing for fun and profit!
- Learn game networking! Excellent talk about networking in FPSes by network programmer of HALO Reach:
  - http://www.gdcvault.com/play/1014345/I-Shot-You-First-Networking
- Reverse-engineer your wireless devices:
  - Capture all the packets!

# BEYOND 461: MORE COOL NETWORK STUFF (3)

Fun stuff I've done in networks:

- Communicating with a robotic arm from Unity game engine
- Simulations that send network traffic to Dragon spacecraft to simulate dynamics of launch, orbit, and ISS docking
- Lunar rover to ground station communication

Go out and do fun network stuff yourself!!

# ANY QUESTIONS?

P.S. MAKE SURE YOU FILL OUT COURSE EVALUATIONS!!