ICS 451: Today's plan

- Exam Review:
  - protocols
  - client-server networking
  - sockets API in C (and Python), socket types
  - protocol layers: physical, data link, network, transport, application
  - application layer: HTTP, DNS, email
  - transport layer: TCP sequence numbers, acks
  - reliable transmission: ABP, windows, speed
  - assignments: HTTP, DNS

Networking Protocols

- Protocol: an agreement about what an exchange means
- Computer networking protocols: agreement about what received bits mean
  - and what action to take in response
- Distributed systems need protocols
  - no system knows the state of another system
  - except through the protocol
  - “knowing” is limited to appropriate responses

Client-Server Networking

- server waits for incoming requests
  - on TCP, for incoming connections
- client (connects and) sends requests
- server sends reply
- connection may be *cached* for further requests and replies

Sockets API

- C and Python
- client: `socket`, `connect`, `send`/`recv`, `close`
- server: `socket`, `listen`, `bind`, `accept`
  - on accepted socket, `send`/`recv`, `close`
- socket types: STREAM (TCP), DGRAM (UDP)
- UDP: `socket`, `sendto`/`recvfrom`, `close`
- pay attention to return values

Protocol Layers

- physical: getting the bits from one system to another
- data link: collecting bits into frames, Medium Access Control (MAC) and MAC addresses
- network: end-to-end best-effort transmission
- transport: demultiplexing, reliability, connections
- application: everything else!

Application Layer: HTTP

- HTTP: client-server
- request line (`GET` path `HTTP/1.1`)
- response line (`HTTP/1.1 200 OK`)
- `Host: example.com` is mandatory in request
- header ends at empty line, then maybe content
- `Content-Type:`
- `Connection:` close/keep-alive
- `Keep-Alive:` time and requests

Application Layer: DNS

- DNS: client-server, usually over UDP
  - recursive queries make server into client
  - iterative queries: “try this server”
  - ID matches replies to requests
- Header
- Resource Records
  - efficient encoding of repeated domain names
  - TTL for caching
- Query/Response Types: A, AAAA, MX, NS
  - and one class, Internet

Application Layer: email

- Mail User Agent/MUA, Mail Transfer Agent/MTA
- header, empty line, body
  - all in 7-bit ASCII
- `Received:` fields trace (back) message path
- Simple Mail Transfer Protocol, SMTP
  - MUA to MTA and MTA to MTA
- for MTA to MUA, IMAP or POP
- Webmail

Application (Presentation) Layer: MIME

- Content-Type
  - type/specific
  - e.g. image/png
  - often now just application/something
- often:
  - Content-Lenght
  - Content-Transfer-Encoding
    - encodes binary using ASCII characters
- used for both email and HTTP

Reliable Transmission: ABP, windows, throughput

- Alternating-Bit Protocol: single-bit sequence/ack numbers
  - stop-and-wait transmission
- Windows allow transmission of multiple MTUs before waiting
  - different strategies for acknowledgements
  - different strategies for receivers to handle out-of-order packets
- throughput ≤ window / RTT
  - i.e. for full speed, window ≥ bandwidth x delay

TCP

- 32-bit sequence and ack numbers
  - counting bytes, not packets
  - ack is seq (of received packet) + bytes
    - or seq + 1 for SYN or FIN packet
- starting at a random place
- SYN packet to establish connection
  - and availability for connection
  - three-way handshake
- data exchanged after connection is established

Assignments: HTTP, DNS (C)

- Assignment 1 and 2: HTTP client/server in Python and C
  - HTTP, sockets API
- Assignment 3: C pointers, strings, byte ordering
- Assignment 4: building binary headers, sending DNS requests, RRs