CSCI-1680
Sockets and network programming

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

Administrivia

- Container setup: fill out form by TONIGHT
  - Whether or not you have it working

Snowcast is out!

- Gearup Today 9/14 5-7pm CIT368 (+Zoom, recorded)
  - Look at the notes!
- Milestone due by Tuesday, 9/19 by 11:59pm EDT
  - Warmup + design doc

Topics for Today

- Working with sockets
- TCP & UDP
- Building a protocol

Sockets: Communication Between Machines

- Network sockets are file descriptors too
- Datagram sockets (eg. UDP): unreliable message delivery
  - Send atomic messages, which may be reordered or lost
- Stream sockets (TCP): bi-directional pipes
  - *Stream* of bytes written on one end, read on another
  - Reads may not return full amount requested, must re-read

System calls for using TCP

Client

Server

`socket` – make socket

`bind` – assign address, port

`listen` – listen for clients

`socket` – make socket

`bind*` – assign address

`connect` – connect to listening socket

`accept` – accept connection

- This call to bind is optional, connect can choose address & port.

Socket Naming

- TCP & UDP name *communication endpoints*
  - IP address specifies host (128.148.32.110)
  - 16-bit port number demultiplexes within host
  - Well-known services listen on standard ports (*e.g.* ssh – 22, http – 80, mail – 25)
  - Clients connect from arbitrary ports to well known ports
- A connection is named by 5 components
  - Protocol, local IP, local port, remote IP, remote port

Dealing with Data

- Many messages are binary data sent with precise formats

- Data usually sent in Network byte order (Big Endian)
  - Remember to always convert!
  - In C, this is htons(), htonl(), ntohs(), ntohl()