CSCI-1680
Sockets and network programming

Nick DeMarinis

Administrivia

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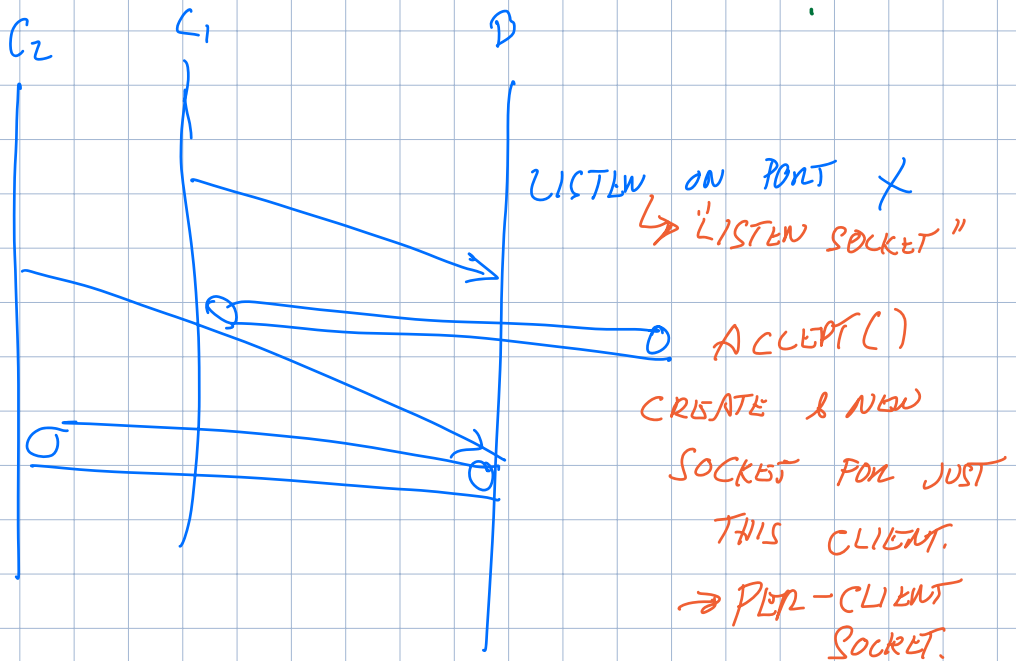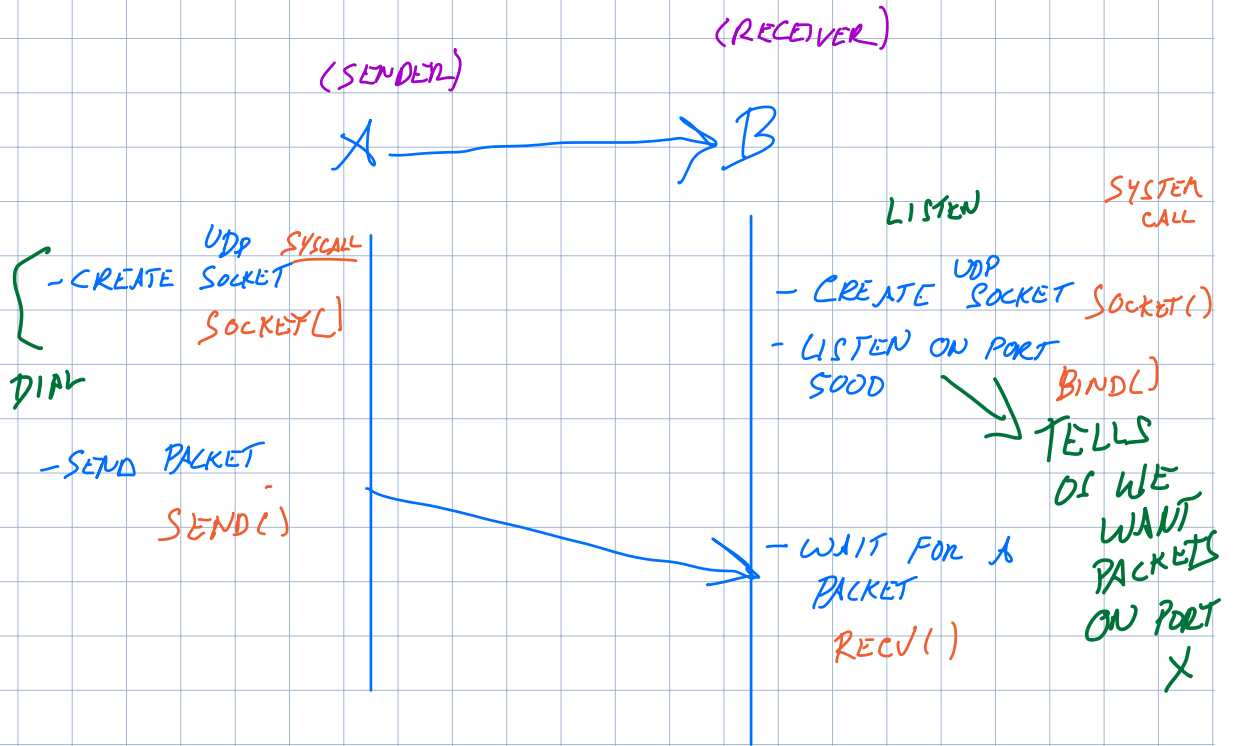 - Whether or not you have it working

<u>Snowcast is out!</u>
- 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
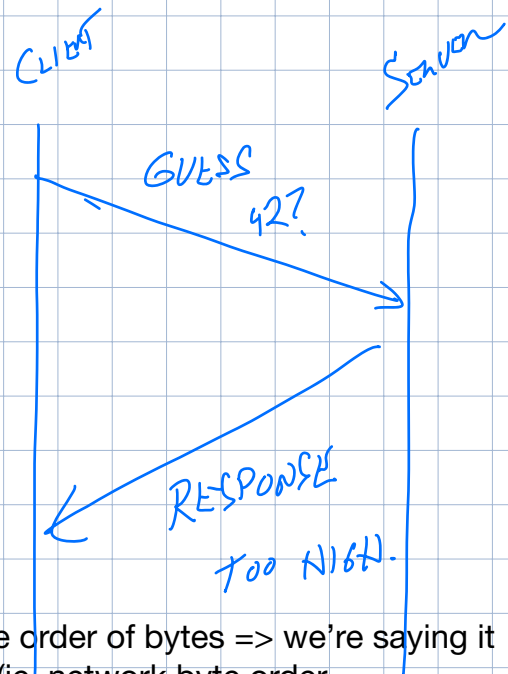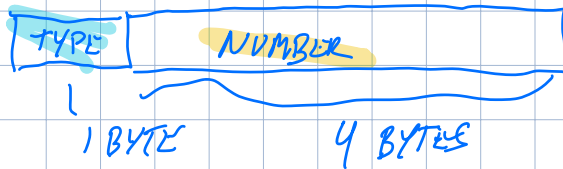
- Working with sockets
- TCP & UDP
- Building a protocol

UDP EXAMPLE

(SENDER)   (RECEIVER)

A ────────────▶ B

                                        LISTEN   SYSTEM
                                                         CALL

UDP SYSCALL   UDP
{ - CREATE SOCKET   - CREATE SOCKET   SOCKET()
    SOCKET()   - LISTEN ON PORT
DIAL   5000   BIND()

                                                        TELLS
- SEND PACKET   OS WE
    SEND()   WANT
                                     - WAIT FOR A   PACKETS
                                       PACKET   ON PORT
                                       RECV()   X

C₂   C₁   D

                         LISTEN ON PORT X
                              ↳ "LISTEN SOCKET"

                         O   ACCEPT()

                             CREATE & NEW
                             SOCKET FOR JUST
                             THIS CLIENT.
                             ⇒ PER-CLIENT
                                 SOCKET.

Client-server example: Guessing game
Server picks a random number
Clients connect and can guess numbers
Server responds with too high, too low, or correct
First client to respond wins, restarts game

As the designers, we get to decide on the
format for how messages are exchanged
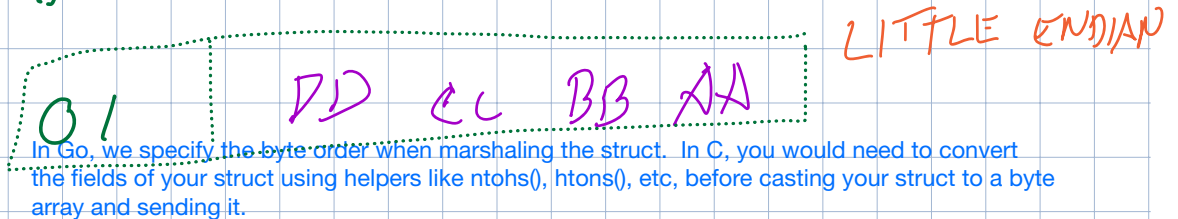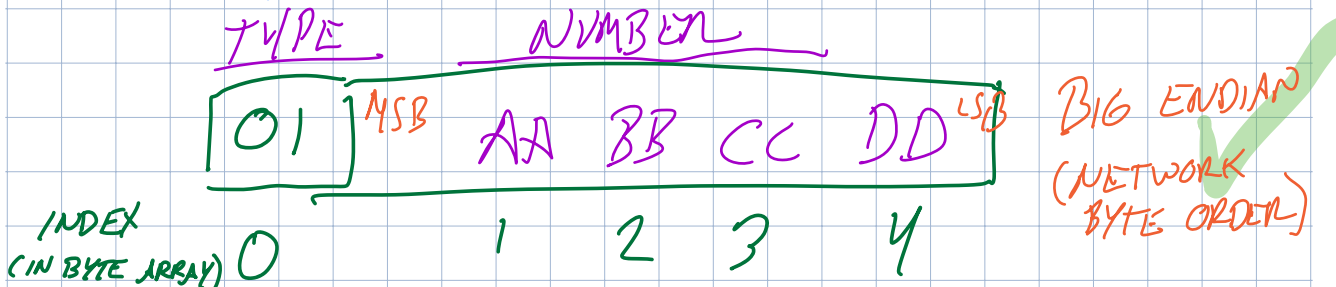Here's our format. In this version, every
message is 5 bytes:

CLIENT   SERVER

GUESS
42?

RESPONSE
TOO HIGH.

TYPE | NUMBER

1 BYTE   4 BYTES

Protocol must give the order of bytes => we're saying it
should be big endian (ie, network byte order

GUESS
TYPE = 0
NUMBER = GUESS

RESPONSE:
TYPE = 1
NUMBER = { 1 TOO HIGH
          { 0 CORRECT!
          { -1 TOO LOW

When we format the message as a byte array, we order each field as in the picture above: first the type, then the
number. For multi-byte data like integers, our protocol needs to specify the byte order (ie, the endianness) used to
send the data "over the wire". In our protocol, we'll use big endian, or "network byte order." If our guess were the
number 0xaabbccdd, we'd format it like this:

TYPE   NUMBER

01 MSB   AA BB CC DD LSB   BIG ENDIAN
                                      (NETWORK
INDEX   BYTE ORDER)
(IN BYTE ARRAY) 0   1   2   3   4

                                      LITTLE ENDIAN

01   DD CC BB AA

In Go, we specify the byte order when marshaling the struct. In C, you would need to convert
the fields of your struct using helpers like ntohs(), htons(), etc, before casting your struct to a byte
array and sending it.

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()