Review of Course – 472

Client/Server:

```
Connection oriented  TCP
Connectionless       UDP
Iterative: accept - (read) - reply - close
Concurrent: Multiple processes -- fork
            Single process -- select
Programming: port numbers, well-known-ports,
             hostnames, ip numbers
Network system calls: htonl(s), read, write,
             sendto, recvfrom, send, recv.
Underlying calls: socket, bind,
             connect, accept, listen
Data structures: sockadder_in, sockaddr_un,
  hostent, servent
```

Broadcast:

```
Enabling UDP broadcast.
Disabling replies only accepted from (no connect).
Broadcast addresses.
```

Multicast:

```
Enabling UDP multicast (Multicast Registration).
Multicast addresses.
```

Nonblocking IO:

Retries, waits, handling EAGAIN

Industrial strength programs:

Backgrounding.

Logging.

Signal handling (config files).

RPC:

Build, split, create interface methodology.

1 parameter, 1 return value (dummies)

pack call unpack / unpack call pack return

Programming: program and procedure numbers, rpcgen

Underlying: xdr

Portmapper: relating a program number to a port.

CORBA:

Build classes, user interface and main separately.

Create interface definition file.

Build CORBA version of main.

Modify identifiers in Classes.

Programming: stringified remote references,
idl compiler

ORB: advertises and finds remote references

SSL:

Building an encrypted connection on top of a socket

Know the SSL routines for client and server

Understand what a certificate is and that they are stored in files (we used pem format)

Tasks

Understand the basic system calls for networking.
`socket`, `connect`, `accept`...(networking)
`gethostby`, `inet_ntoa`...(information handling)

Build a UDP client/server pair.

Build a TCP client/server pair.
Use an iterative server.
Use a concurrent server (fork).
Use a concurrent server (threads).
Undertand the TCP reassembly loop.

Tasks

Build a concurrent connection-oriented single-process server (`chatd`).
Understand the use of the `select` call.

Build a multi-input TCP client (`chatc`)
Understand the use of the `select` call.

Build a multiprotocol server.
Open TCP and UDP sockets.
Use a `select` and `if` statements.

Build a multiservice server.
Use an array to hold the TCP/UDP designations, the socket number and the function address; base the if statements on the contents of the array.

Build a broadcast client/server.
Enabling UDP sockets for broadcast (client and server).
Sending a broadcast.

Build a multicast client/server.
Enabling UDP sockets for multicast (client and server).
Multicast registration.

Tasks

Using preallocation in a server.
When to use preallocation (frequent connects).
How preallocated servers behave (the client hang).

Remote procedure calls.
Based on using XDR for data transport.
Building a `.x` file based on what values went in and out of the function.
Building client-side interface (cif) and server-side interface (sif) files/functions; with pack, call, unpack (cif); and unpack, call, pack, return (sif).
Returning the address of a static.

Standard techniques (tricks of the trade).
Auto-finding a server using broadcasts.
Auto-backgrounding
Syslog for logging
HUP for reloading.

How to use non-blocking I/O (handling `EAGAIN`).
Understanding the system network buffers,network buffer size and how they can impact programs (hang).

How to use local domain sockets.
Fast, private, network like communication for programs.

Tasks

Understand the Common Object Request Broker Architecture (CORBA).
Specifying the idl file and understanding it's format.
Understanding how to structure a normal program that uses classes so it can be easily transformed into a CORBA program.
Knowing where the network boiler plate should be placed.

Convert a normal socket based client or server into a client or server that uses an encrypted connection. (A more secure client or server.)