# Eng 67: Lab 4

## Lab 4 Learning Objectives:

1. To expand your understanding of Internet programming and addressing.
2. How to create a program using the TCP protocol.
3. Expand knowledge of C and POSIX programming with timers and string pattern matching.

**Background:** You should be familiar with the following (new functions in shown in bold):

- *sockaddr* structure – specifying socket addresses
- *sockaddr_in* structure – a convenient form of sockaddr
- AF_INET – a constant specifying the Internet protocol family
- memset() – used to initialize a sockaddr_in structure
- inet_addr() – used to generate an IP address from a string
- htons() – converts a short to network byte order
- socket() – used to create a socket
- close() – closes a socket
- htonl(INADDR_ANY) – allow use of *any* incoming IP at the server
- bind() – assign a PORT to a socket at the server

- **connect()** – connect to a TCP server from a client
- **send()** and **recv()** – send and recv TCP messages
- **listen()** – listen for a connection on a PORT at the server
- **accept()** – get a new socket for each client connection at the server
- **alarm()** – set a SIGALRM timeout before a blocking call

## TASKS

1. Modify your UDP *client* to timeout using an alarm() if the server does not respond within 5 seconds and cause a failure i.e. printing a message "Server Not Available", closing the connection, and

terminating the program. Test your new client by running the client *without* the server.

2. Modify your UDP client to retry three times before failing.

3. Copy and subsequently modify your UDP client and server to use TCP rather than UDP; your new code should communicate with the send() and recv() functions, rather than sendto() and recvfrom().

4. **ENGS 115:** Modify your TCP server to incorporate multi-threading such that each client connection is handled by a separate POSIX thread at the server.

## TOPICS

**Research the following questions:**

- How can you define a MACRO with multiple arguments using the **#define** statement in C?
- What is the Message Passing Interface (MPI) ?
- What programming issues is MPI supposed to solve?
- What do the following functions do:
    - MPI_Init
    - MPI_Finalize
    - MPI_Comm_rank
    - MPI_Comm_size
    - MPI_Send
    - MPI_Recv
- Open MPI is the version of MPI that you will be using:
    - How do you compile a program to use MPI?
    - How do you initiate execution of a program on 5 computers?
    - How do you pass command line arguments to the program?