**CSL 332 NETWORKING LAB  
CYCLE 1**

**I.** Familiarize the basic Linux commands that allow our system to communicate in Internet by performing the following experiments

1. View the configuration including addresses of your computer’s network interfaces

2. Test the network connectivity between your computer and several other computers

3. View the active TCP connections in the computer after visiting a website.

4. Find the hardware/MAC address of another computer in the network using ARP.

5. View the network configuration files

**II.** Familiarize the basic commands in Windows OS that allow our system to communicate in Internet by performing the following experiments

1. View the configuration including addresses of your computer’s network interfaces

2. Test the network connectivity between your computer and several other computers

3. View the active TCP connections in the computer after visiting a website.

4. Find the hardware/MAC address of another computer in the network using ARP.

5. View the network configuration files

**III.** Write the system calls used for creating sockets and transferring data between two nodes.

**IV.** Write a program to find the maximum, minimum and average of an array of integers using socket programming.

**V.** (a) Create three programs, two of which are clients to a single server. Client1 will send a string to the server process using datagram socket and stream socket. The server will reverse the string and send the result to Client2. Client2 prints the reversed string it receives and then all the processes terminate.

(b) Follow the same procedure as in part a except that the data type of the message should be integer and the server should square the integer before transmitting it to Client2.

(c) Write a socket program to enable Client1 to send a float value to the server. The server process should increase the value of the number it receives by a power of 1.5. The server should print both the value it receives and the value that it sends. Client2 should print the value it receives from the server.

**VI. Dining Philosophers Problem**

a. Create five child processes for philosophers, and five child processes for each of the chopsticks.

b. Initiate a pair of sockets for each connection to a server

c. Avoid possible deadlock

d. Close and unlink the sockets.

**Description**: Use the fork( ) system call to create a child process. The parent should do nothing but create ten child processes. The requests for granting and releasing chopsticks should be done by each child process representing a philosopher through communication with the two processes representing the philosopher’s left and right hand chopsticks.TCP/IP should be used for communication between philosophers and chopstick processes. The datagram type of sockets should be employed between the parent and child processes including both the philosopher and chopstick processes. Use write( ) and read() system calls with the connection-oriented communication. A solution that prevents deadlock or starvation should be implemented. For example the RightLeftDP algorithm (Lynch,1996) is a suitable algorithm. Five child processes are to be created for the five philosophers. The philosophers change their states among “thinking”, “waiting”, and “eating” until all philosophers fulfil their eating requirements(60 seconds total eating time). Print the status of each philosopher whenever there is a change in the status.

**VII. Multi user chat server** using TCP

Write the client and server programs, where the server can exchange text with many client processes. A client process starts the communication with an input "start". After this, the client process waits for the answer from the server. If server permits, it can further send any text message (with restriction of not more than 25 words in a day). The communication goes on in this way until the client process sends the message " stop " to the server.

**VIII.** Implement a simple web proxy server that accepts HTTP requests and forwarding to remote servers and returning data to the client using TCP.

**IX.** Implement a Concurrent Time Server application using UDP to execute the program at a remote server. Client sends a time request to the server, server sends its system time back to the client. Client displays the result.

**ADVANCED EXERCISES**

1. Send a C structure that includes data of type character, integer and float from client1 to the server. The server should change the values so that client2 receives a structure with entirely different data. It is not permitted that the data should be converted to any other data type before transmission.

2. Repeat 1 except that the machine architectures of the clients and server should be different. You are not permitted to convert the structures to stings before transmission. This is an exercise in the use of big endian and little endian representations of data. Explain what changes had to be made in order to accommodate the differences in data representations.

3. A haiku is a three-line poem in which the first line contains five syllables, the second line contains seven syllables, and the third line contains five syllables. Write a haiku server that listens to port 5575. When a client connects to this port, the server responds with a haiku.