Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA Deemed to be University

MAJOR ASSIGNMENT-01

UNIX Network Programming (CSE 4042)

Problem Statement

Experiment with byte ordering, byte manipulation, and IP address conversion functions in socket programming. Additionally deals with how to work with loopback address and netstat command to realise TCP state transitions.

Part-1

The following experiments helps student to work with TCP day-time-server-client and to learn the usages of address conversion functions `inet_pton`, `ntohs`, `htons` etc.

(1) The below codes, Program 1: and Program 2: for TCP daytimetcpserver and daytimetcpclient to display the date and time in a human-readable format, when the client establishes a TCP connection. Compile and run the code. Run the client code for few times, specifying a different IP address as the command line argument each time and record the responses. Some run cases for the client as;

(a) `./a.out 127.0.0.0 portdisplayedinserver`

(b) `./a.out 127.0.0.1 portdisplayedinserver`

⋮⋮⋮⋮

(c) `./a.out 127.0.0.255 portdisplayedinserver`

Note: make a note of the loopback address.

Program 1: Day time tcp server

```
1  #include<stdio.h>
2  #include<unistd.h>
3  #include<sys/socket.h>
4  #include<sys/types.h>
5  #include<netinet/in.h>
6  #include<stdlib.h>
7  #include<string.h>
8  #include<time.h>
9  #include<arpa/inet.h>
10 #include<time.h>
11 #define MAXLINE 4096
12 int main(int argc, char **argv){
13   int listenfd, connfd,len;
14   struct sockaddr_in servaddr,cliaddr;
15   char buff[MAXLINE];
16   time_t ticks;
17   len=sizeof(servaddr);
18   listenfd = socket(AF_INET, SOCK_STREAM, 0);
19   bzero(&servaddr, sizeof(servaddr));
20   servaddr.sin_family = AF_INET;
21   servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
22   servaddr.sin_port   = htons(0);
23   bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
24   getsockname(listenfd,(struct sockaddr *) &servaddr,&len);
```

```
  printf("Port for client=%d\n",(int)ntohs(servaddr.sin_port));
  listen(listenfd, 1024);
  for ( ; ; ) {
     connfd = accept(listenfd, (struct sockaddr *) &cliaddr,&len);
     ticks = time(NULL);
     snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
     write(connfd, buff, strlen(buff));
     close(connfd);
   }
}
```

Program 2: Day time tcp client

```
#define MAXLINE 4096
int main(int argc, char **argv){
  int sockfd, n,len;
  char recvline[MAXLINE+ 1];
  struct sockaddr_in servaddr;
  len=sizeof(servaddr);
  if (argc != 3){
     perror("usage: a.out  <IP address> <port number>");
     exit(1);
  }
  if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
     perror("socket error");
     exit(2);
  }
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons(atoi(argv[2]));
  if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0){
     perror("inet_pton error for %s\n",argv[1]);
     exit(3);
  }
  if (connect(sockfd, (struct sockaddr *) &servaddr, len) < 0){
     perror("connect error");
     exit(4);
  }
  while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
     recvline[n] = 0;   /* null terminate */
     if (fputs(recvline, stdout) == EOF){
         perror("fputs error");
         exit(5);
     }
 }
  if (n < 0){
     perror("read error");
     exit(6);
  }
  exit(0);
}
```

(2) Modify the first argument to `socket` in Program 2: to be 9999. Compile and run the program. What happens? Find the errno value corresponding to the error that is printed. How can you find more information on this error?

(3) Modify Program 2: by placing a **`counter`** in the `while` loop, counting the number of times `read`

returns a value greater than 0. Print the value of the **`counter`** before terminating. Compile and run your new client.

(4) Modify 1: as follows: First, change the port number assigned to the `sin_port` member from 13 to 9999 instead of 0 and avoid using of `getsockname` function call. Next, change the single call to write into a loop that calls write for each byte of the result string. Compile this modified server and start it running in the background. Next, modify the client from the previous exercise (which prints the `counter` before terminating), changing the port number assigned to the `sin_port` member from 13 to 9999. Start this client, specifying the IP address of the host on which the modified server is running as the command-line argument. What value is printed as the client's counter? If possible, also try to run the client and server on different hosts.

(5) Assume that your server and client is working properly. Now, the client desires to display the details of the server (IP and port of server) to whom it is connected. Make necessary modification in the client code and run to display the requirement. Run your server and client code in different machine to get more concrete answer.

Part-2

Dealing with transport layer protocols and realization of TCP state transition diagram

1. Run your TCP client server program to observe the different states of the TCP connection and termination process.. Observe how the local address and foreign addresses are getting filled up in client side and in server side.

2. Differentiate between TCP and UDP through an implementation.

3. Build the TCP server from the Text Book **`Figures 5.2`** and **`5.3`** and the TCP client from **`Figures 5.4`** and **`5.5`**. Start the server and then start the client. Type in a few lines to verify that the client and server work. Terminate the client by typing your `EOF` character and note the time. Use **`netstat`** on the client host to verify that the client's end of the connection goes through the `TIME_WAIT` state. Execute **`netstat`** every five seconds or so to see when the `TIME_WAIT` state ends. What is the `MSL` for this implementation?

Part-3

Discovering the network topology of your network

1. Few basic commands (**`netstat`**, **`ifconfig`**, **`ping`**, **`ping`** etc.) can be used to discover some details of a network. Go through the following steps to discover information about your network topology.

   (a) **`netstat -i`** provides information on the interfaces. Also try **`netstat -ni`** to print numeric addresses. Test the usages of the commands.

   (b) An another way to determine the interfaces: **`netstat -r`**. Check the output of the command **`netstat -nr`**

   (b) An another way to determine the interfaces: **`netstat -r`**. Check the output of the command.

(c) Type the command **`ifconfig`** and make a observation on the output to get the details of the interfaces. Find the **name** of the ethernet interface or any other interfaces on your machine. Take a note of your IP address, broadcast address, mask, hardware address etc.

(d) Demonstrate the use of **`ping command`**. For example **`ping –b <broadcast address>`** helps to find the IP address of many hosts on the local network.

(e) You are required to try the other command, **`ip`** - show / manipulate routing, devices, policy routing and tunnels. Visit man page for **`ip`** command and perform followings;

(i) **`ifconfig`**

(i) **`ip addr show`**

(i) **`ip link show`**