BITS Pilani - Hyderabad Campus
CS F303 (Computer Networks)
Second Semester 2023-24, Lab Sheet 4
Socket Programming over TCP

1. Overview

In this lab, we will see how to use sockets and various functions that are used in socket programming. Sockets can be used with both UDP or TCP protocol. In this lab, we will see socket programming over TCP. A basic architecture of using sockets over TCP is shown in Figure 1.

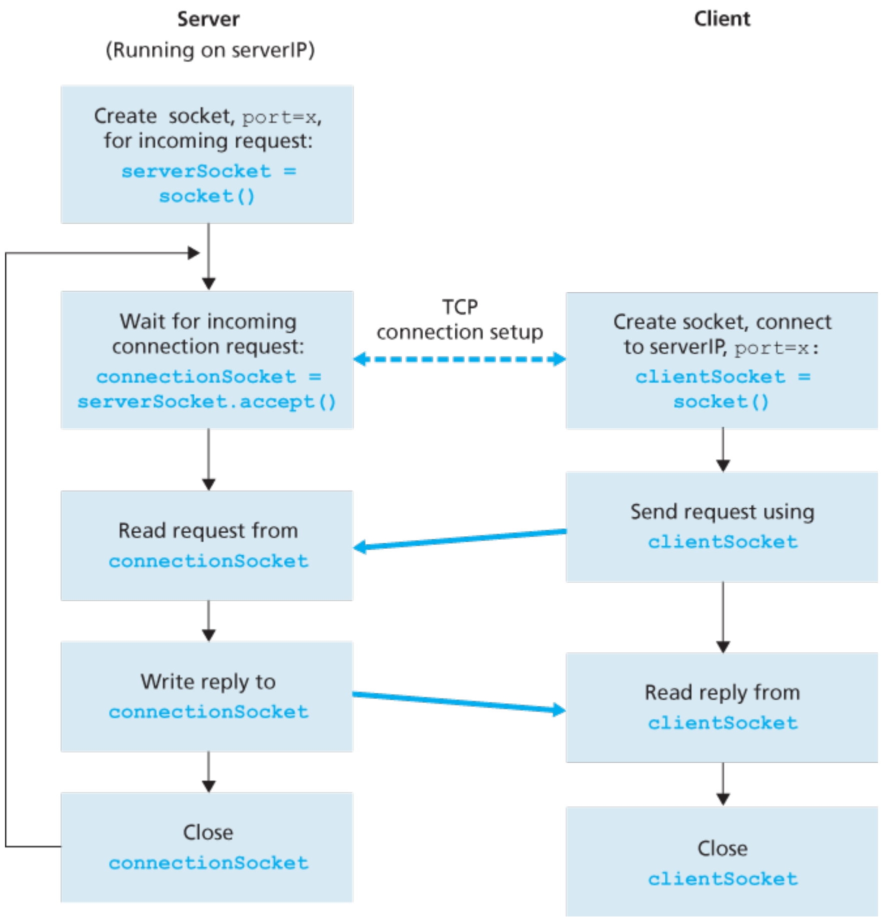
Server
(Running on serverIP)
Client
Create socket, port=x, for incoming request:
serverSocket = socket()
Wait for incoming connection request:
connectionSocket = serverSocket.accept()
TCP connection setup
Create socket, connect to serverIP, port=x:
clientSocket = socket()
Read request from connectionSocket
Send request using clientSocket
Write reply to connectionSocket
Read reply from clientSocket
Close connectionSocket
Close clientSocket

Figure 1 Socket Programming over TCP

Following structures can be utilized while implementing sockets:

- struct sockaddr_in server; // IPv4 AF_INET sockets:
- struct sockaddr_in

```
{

short sin_family; // e.g. AF_INET, AF_INET6

unsigned short sin_port; // e.g. htons(3490)

struct in_addr sin_addr; // see struct in_addr, below

char sin_zero[8]; // zero this if you want to

};
```

- struct in_addr

```
{

unsigned long s_addr; // load with inet_pton()

};
```

- struct sockaddr

```
{

unsigned short sa_family; // address family, AF_xxx

char sa_data[14]; // 14 bytes of protocol address

};
```

2. Functions used at client side

At client side, following steps should be taken:

- Create a socket using the socket() function;
- Connect the socket to the address of the server using the connect() function.
- Send and receive data by means of the read() and write() functions.

The socket programming implementation at client side include creating a socket, connecting to a server, sending data and receiving reply as explained follows.

- **Create a socket:**

```
int socket_desc;
socket_desc = socket(AF_INET , SOCK_STREAM , 0);
```

- **Connect to a server:**

```
server.sin_addr.s_addr = inet_addr("74.125.235.20");

server.sin_family = AF_INET;

server.sin_port = htons( 80 );

 //Connect to remote server

connect(socket_desc , (struct sockaddr *)&server , sizeof(server))
```

- **Send Data:**

```
message = "GET / HTTP/1.1\r\n\r\n";

send(socket_desc , message , strlen(message) , 0)
```

- **Receive Reply**

```
recv(socket_desc, server_reply , 2000 , 0)
```

- **Close the Socket**

```
close(socket_desc);
```

3. Functions used at Server side

At server side, following steps should be taken:

- Create a socket with the socket() function.
- Bind the socket to an address using the bind() function; · Listen for connections with the listen() function.

- Accept a connection with the accept() function system call. This call typically blocks until a client connects with the server.
- Send and receive data by means of send() and receive().

The socket programming implementation at server side include creating a socket, binding to an address, listening for incoming connection and sending to the client.

- **create a socket**

Same like client-side

- **Bind to an address(and port):**

```
// Prepare the sockaddr_in structure

server.sin_family = AF_INET;

server.sin_addr.s_addr = INADDR_ANY;

server.sin_port = htons( 8888 );

 //Bind

bind(socket_desc,(struct sockaddr *)&server , sizeof(server))
```

- **Listen for incoming connections and accept connection:**

```
 struct sockaddr_in server , client;

 listen(socket_desc , 3); int c = sizeof(struct sockaddr_in);

 //Accept and incoming connection
```

```
accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
```

- **Send to client:**

```
://Reply to the client

message = "Hello Client , I have received your connection. But I have to go

now, bye\n"; write(new_socket , message , strlen(message));
```

4. Practice Exercise

Write a C programme for chat application using socket programming.

TCP Server:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int connfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
                bzero(buff, MAX);
                // read the message from client and copy it in buffer read(connfd, buff,
                sizeof(buff));
                // print buffer which contains the client contents
                printf("From client: %s\t To client : ", buff);
                bzero(buff, MAX);
                n = 0;
                // copy server message in the buffer
                while ((buff[n++] = getchar()) != '\n')
                        ;

                // and send that buffer to client
                write(connfd, buff, sizeof(buff));

                // if msg contains "Exit" then server exit and chat ended. if
                (strncmp("exit", buff, 4) == 0) {
                        printf("Server Exit...\n");
                        break;
                }
        }
}
// Driver function
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;

        // socket create and verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                printf("socket creation failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));
```

```
// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification if
((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
}
else
        printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0)
{
        printf("Listen failed...\n");
        exit(0);
}
else
        printf("Server listening..\n");
len = sizeof(cli);

// Accept the data packet from client and
verification connfd = accept(sockfd, (SA*)&cli,
&len);
if (connfd < 0)
{
        printf("server accept failed...\n");
        exit(0);
}
else
        printf("server accept the client...\n");

// Function for chatting between client and server func(connfd);
```

```
        // After chatting close the socket
        close(sockfd);
}
```

TCP Client:

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
        char buff[MAX];
        int n;
        for (;;) {
                bzero(buff, sizeof(buff));
                printf("Enter the string : ");
                n = 0;
                while ((buff[n++] = getchar()) != '\n') ;
                write(sockfd, buff, sizeof(buff));
                bzero(buff, sizeof(buff));
                read(sockfd, buff, sizeof(buff));
                printf("From Server : %s", buff);
                if ((strncmp(buff, "exit", 4)) == 0)
                {
                        printf("Client Exit...\n");
                        break;
                }
        }
}

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0); if
     (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    // function for chat
    func(sockfd);

    // close the socket
    close(sockfd);
}
```

Compilation:

Server side:

```
gcc server.c -o server
 ./server
```
Client side:
```
gcc client.c -o client
 ./client
```

Output:

Server side:
```
Socket successfully created..

Socket successfully binded..

Server listening..

 server accept the client...

 From client: hi

  To client : hello

 From client: exit

  To client : exit

Server Exit...
```

Client side:
```
Socket successfully created..

connected to the server..

 Enter the string : hi

 From Server : hello

 Enter the string : exit

 From Server : exit

Client Exit...
```

5. Lab Exercise

Write a C program with TCP socket programming to transfer a file from a client to the server. The server should reply with a message “File transfer complete” to the client when the file transfer process is completed.