**Madhurya Mozumder**

**RA1911028010036**

**CSE-CC**

**I2**

**Experiment 6**

**Half Duplex Chat Using TCP/IP**

Aim:

To To implement a half duplex application, where the Client establishes a connection with the Server. The Client can send and the server well receive messages at the same time.

Procedure:

Server:

* Include the necessary header files.
* Create a socket using socket function with family AF\_INET, type as

SOCK\_STREAM.

* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET, sin\_addr to INADDR\_ANY, sin\_port to dynamically assigned port number.
* Bind the local host address to socket using the bind function.
* Listen on the socket for connection request from the client.
* Accept connection request from the Client using accept function.
* Fork the process to receive message from the client and print it on the console.
* Read message from the console and send it to the client.

Client:

* Include the necessary header files.
* Create a socket using socket function with family AF\_INET, type as

SOCK\_STREAM.

* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET.
* Get the server IP address and the Port number from the console.
* Using gethostbyname function assign it to ahostent structure, and assign it to sin\_addr of the server address structure.
* Request a connection from the server using the connect function.
* Fork the process to receive message from the server and print it on the console.
* Read message from the console and send it to the server.

1. Server

#include<sys/types.h>

#include<stdio.h>

#include<netdb.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<netinet/in.h>

#include<string.h>

int main(int argc,char \*argv[]){ int n,sd,ad;

struct sockaddr\_inservaddr,cliaddr; socklen\_tclilen,servlen; char buff[10000],buff1[10000];

bzero(&servaddr,sizeof(servaddr));

/\*Socket address structure\*/ servaddr.sin\_family=AF\_INET;

servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

servaddr.sin\_port=htons(2500);

/\*TCP socket is created, an Internet socket address structure is filled with wildcard address & server’s well known port\*/ sd=socket(AF\_INET,SOCK\_STREAM,0);

/\*Bind function assigns a local protocol address to the socket\*/ bind(sd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

/\*Listen function specifies the maximum number of connections that kernel should queue for this socket\*/

listen(sd,5);

printf("%s\n","server is running…");

/\*The server to return the next completed connection from the front of the completed connection Queue calls it\*/ ad=accept(sd,(struct sockaddr\*)&cliaddr,&clilen); while(1){

bzero(&buff,sizeof(buff));

/\*Receiving the request from client\*/ recv(ad,buff,sizeof(buff),0);

printf("Receive from the client:%s\n",buff); n=1; while(n==1){

bzero(&buff1,sizeof(buff1)); printf("%s\n","Enter the input data:"); /\*Read the message from client\*/ fgets(buff1,10000,stdin); /\*Sends the message to client\*/ send(ad,buff1,strlen(buff1)+1,0); printf("%s\n","Data sent");

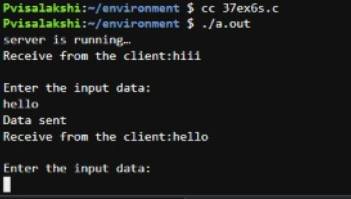
n=n+1;

}

} return 0;

}

Output:



2. Client

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<unistd.h>

#include<stdio.h>

#include<netdb.h>

#include<string.h>

int main(int argc,char \*argv[]){

int n,sd,cd;

struct sockaddr\_inservaddr,cliaddr; socklen\_tservlen,clilen; char buff[10000],buff1[10000];

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET; servaddr.sin\_addr.s\_addr=inet\_addr(argv[1]); servaddr.sin\_port=htons(2500);

sd=socket(AF\_INET,SOCK\_STREAM,0);

cd=connect(sd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

while(1){ bzero(&buff,sizeof(buff));

printf("%s\n","Enter the input data:");

fgets(buff,10000,stdin);

send(sd,buff,strlen(buff)+1,0);

printf("%s\n","Data sent");

n=1;

while(n==1){ bzero(&buff1,sizeof(buff1)); recv(sd,buff1,sizeof(buff1),0); printf("Received from the server:%s\n",buff1);

n=n+1;

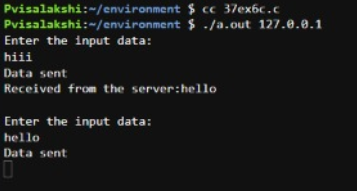
}

}

return 0;

}

Output:



Result:

Thus the chat application full duplex communication is established by sending the request from the client to the server, server gets the message and gives response to the client and prints it.