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.

Ex.No:7	FULL DUPLEX CHAT USING TCP/IP
Date:	

Aim:

There are two hosts, Client and Server. Both the Client and the Server exchange message i.e. they send messages to and receive message from the other. There is a two way communication between them.

TECHNICAL OBJECTIVE:

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

Algorithm :

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 a hostent 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.

Codes:

Server:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
```

```
int main(int argc,char *argv[])
{
        int clientSocketDescriptor,socketDescriptor;
        struct sockaddr_inserverAddress,clientAddress;
        socklen_tclientLength;
        char recvBuffer[8000],sendBuffer[8000];
        pid_tcpid;
        bzero(&serverAddress,sizeof(serverAddress));
        /*Socket address structure*/
        serverAddress.sin_family=AF_INET;
        serverAddress.sin_addr.s_addr=htonl(INADDR_ANY);
        serverAddress.sin_port=htons(9652);
        /*TCP socket is created, an Internet socket address structure is filled with
        wildcard address & server's well known port*/
        socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
        /*Bind function assigns a local protocol address to the socket*/
        bind(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
        /*Listen function specifies the maximum number of connections that kernel should queue
        for this socket*/
        listen(socketDescriptor,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*/
        clientSocketDescriptor=accept(socketDescriptor,(struct
        sockaddr*)&clientAddress,&clientLength);
        /*Fork system call is used to create a new process*/
        cpid=fork();
        if(cpid==0)
        {
                while(1)
                {
                        bzero(&recvBuffer,sizeof(recvBuffer));
                        /*Receiving the request from client*/
                        recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);
                        printf("\nCLIENT : %s\n",recvBuffer);
                }
        }
        else
        {
        while(1)
        {
                bzero(&sendBuffer,sizeof(sendBuffer));
                printf("\nType a message here ...  ");
                /*Read the message from client*/
```

```
                fgets(sendBuffer,80000,stdin);
                /*Sends the message to client*/
                send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
                printf("\nMessage sent !\n");
        }
        }
        return 0;
        }
```

Client:

```
        #include "stdio.h"
        #include "stdlib.h"
        #include "string.h"
        //headers for socket and related functions
        #include <sys/types.h>
        #include <sys/socket.h>
        //for including structures which will store information needed
        #include <netinet/in.h>
        #include <unistd.h>
        //for gethostbyname
        #include "netdb.h"
        #include "arpa/inet.h"
        int main()
        {
                int socketDescriptor;
                struct sockaddr_inserverAddress;
                char sendBuffer[8000],recvBuffer[8000];
                pid_tcpid;
                bzero(&serverAddress,sizeof(serverAddress));
                serverAddress.sin_family=AF_INET;
                serverAddress.sin_addr.s_addr=inet_addr("127.0.0.1");
                serverAddress.sin_port=htons(9652);
                /*Creating a socket, assigning IP address and port number for that socket*/
                socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
                /*Connect establishes connection with the server using server IP address*/
                connect(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
                /*Fork is used to create a new process*/
                cpid=fork();
                if(cpid==0)
                {
                        while(1)
                        {
                                bzero(&sendBuffer,sizeof(sendBuffer));
                                printf("\nType a message here ...  ");
                                /*This function is used to read from server*/
                                fgets(sendBuffer,80000,stdin);
```