Ex.No:6	HALF 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 or receive message from the other. There is only a single way communication between them.

TECHNICAL OBJECTIVE:

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.

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

```
#define MAX 1000
#define BACKLOG 5
int main()
{
        char serverMessage[MAX];
        char clientMessage[MAX];
        int  socketDescriptor = socket(AF_INET, SOCK_STREAM, 0);
        struct sockaddr_inserverAddress;
        serverAddress.sin_family = AF_INET;
        serverAddress.sin_port = htons(5214);
        serverAddress.sin_addr.s_addr = INADDR_ANY;
        bind(socketDescriptor, (struct sockaddr*)&serverAddress, sizeof(serverAddress));
        listen(socketDescriptor, BACKLOG);
        int clientSocketDescriptor = accept(socketDescriptor, NULL, NULL);
while (1)
{
        printf("\ntext message here .. :");
        scanf("%s", serverMessage);
        send(clientSocketDescriptor, serverMessage, sizeof(serverMessage) , 0);
        recv(clientSocketDescriptor, &clientMessage, sizeof(clientMessage), 0) ;
        printf("\nCLIENT: %s", clientMessage);

 }
        close(socketDescriptor);
        return 0;
}
```

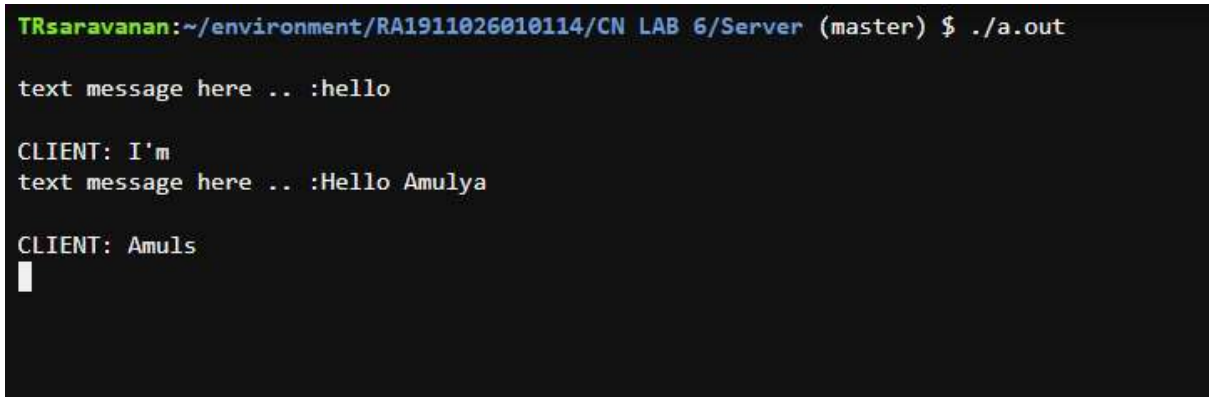
Client:

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include <sys/types.h>
#include <sys/socket.h>
include <netinet/in.h>
#include <unistd.h>
#include "netdb.h"
#include "arpa/inet.h"
#define h_addrh_addr_list[0]
#define PORT 5214
#define MAX 1000
int main(){
char clientResponse[MAX];
int socketDescriptor = socket(AF_INET, SOCK_STREAM, 0);
char hostname[MAX], ipaddress[MAX];
struct hostent *hostIP;
f(gethostname(hostname,sizeof(hostname))==0){
        hostIP = gethostbyname(hostname);
}
else{
        printf("ERROR:FCC4539 IP Address Not ");
}
struct sockaddr_inserverAddress;
serverAddress.sin_family = AF_INET;
```

```
serverAddress.sin_port = htons(PORT);
serverAddress.sin_addr.s_addr = INADDR_ANY;
connect(socketDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
printf("\nLocalhost: %s\n", inet_ntoa(*(struct in_addr*)hostIP->h_addr));
printf("Local Port: %d\n", PORT);
printf("Remote Host: %s\n", inet_ntoa(serverAddress.sin_addr));
while (1)
{
        recv(socketDescriptor, serverResponse, sizeof(serverResponse), 0);
        printf("\nSERVER : %s", serverResponse);
        printf("\ntext message here... :");
        scanf("%s", clientResponse);
        send(socketDescriptor, clientResponse, sizeof(clientResponse), 0);
}
close(socketDescriptor);
return 0;
}
```

Sample Output:

Server:



Client: