TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Lab Report

Of

Distributed System

On

Simple Client Server Implementation

Submitted By:

Santosh Pandey (THA076BCT041)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

June, 2023

Theory:

The client-server architecture is a distributed computing model in which tasks and resources are divided between clients (requestors) and servers (providers). The core components of this architecture include the client, server, and the communication channel that facilitates data exchange between them.

The client refers to a user interface, application, or device that initiates requests for services or resources from the server. Clients can be software applications running on a personal computer, mobile device, or any other networked device.

The server, on the other hand, is a centralized entity that receives and processes client requests, providing the requested services or resources. Servers are typically high-performance computers or systems that are capable of handling multiple client connections simultaneously.

In a client-server architecture, communication between the client and server takes place over a network using predefined protocols. The client sends requests to the server, which processes the requests and sends back the requested data or performs the requested actions. This communication can occur over various protocols such as HTTP, TCP/IP, or WebSocket, depending on the specific requirements of the application

Implementation:

Server.cpp

```
// Chat Server

#define WIN32_LEAN_AND_MEAN
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#include <ws2tcpip.h>
#include <iostream>
#include <string>
#pragma comment(lib, "ws2_32.lib")
int main()
{
    WSADATA wsaData;
    WORD requiredVersion = MAKEWORD(2, 2);
    SOCKET LSocket;
```

```
sockaddr_in Server;

//localHost=gethostbyname("");

//requesting a desired version of winsock dll
int result = WSAStartup(requiredVersion, &wsaData);

//check if the requested version is returned or not
if (result != 0)
{
    std::cout << "WSAStartup failed with error: " << result << ".\n";
    return 1;
}
if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
{
    std::cout << "Could not find a usable version of Winsock.dll\n";
    WSACleanup();
    return 1;
}
else
{
    std::cout << "The Winsock 2.2 dll was found okay\n";
}

//Creating the listening Socket
LSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

//Checking if the Socket was created Successfully or not
if (LSocket == INVALID_SOCKET)
{
    std::cout << "Failed to Create a Scocket, with error: "
    << WSAGetLastError() << ".\n";
    WSACleanup();
    return 1;
}

//Configure the Local adress and port
Server.sin_family = AF_INET;
Server.sin_addr.s_addr = inet_addr("127.0.0.1");
Server.sin_port = htons(8080);

//Bind the socket with Local Address
result = bind(LSocket, (SOCKADDR*)&Server, sizeof(Server));

//Checking if Binding was Successful or not
```

```
    if (result == SOCKET_ERROR)
    {
        std::cout << "Binding Failed with Error: " <<
        WSAGetLastError() << ".\n";
        closesocket(LSocket);
        WSACleanup();
        return 1;
    }
    else
    {
        std::cout << "Binding Successful\n";
    }
    //Listen for incoming Connection
    result = listen(LSocket, SOMAXCONN);
    if (result == SOCKET_ERROR)
    {
        std::cout << "Failed to Listen on Socket with Error: " <<
        WSAGetLastError() << ".\n";
        WSACleanup();
        return 1;
    }
    else
    {
        std::cout << "Listening on Socket...\n";
    }


    sockaddr_in ClientAddr;
    int ClientSize = sizeof(ClientAddr);
    SOCKET ClientSock;

    //Accepting Connection
    std::cout << "Waiting for Client to Connect...\n";
    ClientSock= accept(LSocket, (sockaddr*)&ClientAddr, &ClientSize);
    char ClientName[NI_MAXHOST];
    char ClientPort[NI_MAXSERV];
    if (ClientSock == INVALID_SOCKET)
    {
        std::cout << "Accept Failed with error: " <<
        WSAGetLastError() << ".\n";
        closesocket(LSocket);
        WSACleanup();
        return 1;
    }
    else
```

```
{
    std::cout << "Client Connected...\n";
}

char msg[2048];
int msglen;

//process after connection is made
while (true)
{
    ZeroMemory(msg,2048);
    msglen = recv(ClientSock, msg, 2048, 0);
    if (strcmp(msg, "exit") == 0)
    {
        std::cout << "user left\n";
        break;
    }
    if(msglen!=0)
        std::cout<< "user: " << msg << "\n";

    std::cout << "me: ";
    std::cin.getline(msg, 2048);

    if (strcmp(msg,"exit")==0)
    {
        break;
    }
    else
        send(ClientSock, msg, 2048, 0);

}
//Close the socket
result = closesocket(LSocket);
if (result == SOCKET_ERROR)
{
    std::cout << "Failed to Close the Socket with Error: " <<
    WSAGetLastError() << ".\n";
    WSACleanup();
    return 1;
}
else
{
    std::cout << "Socket Closed Successfully...\n";
}
//done using the Winsock.dll
```

```
    WSACleanup();
}
```

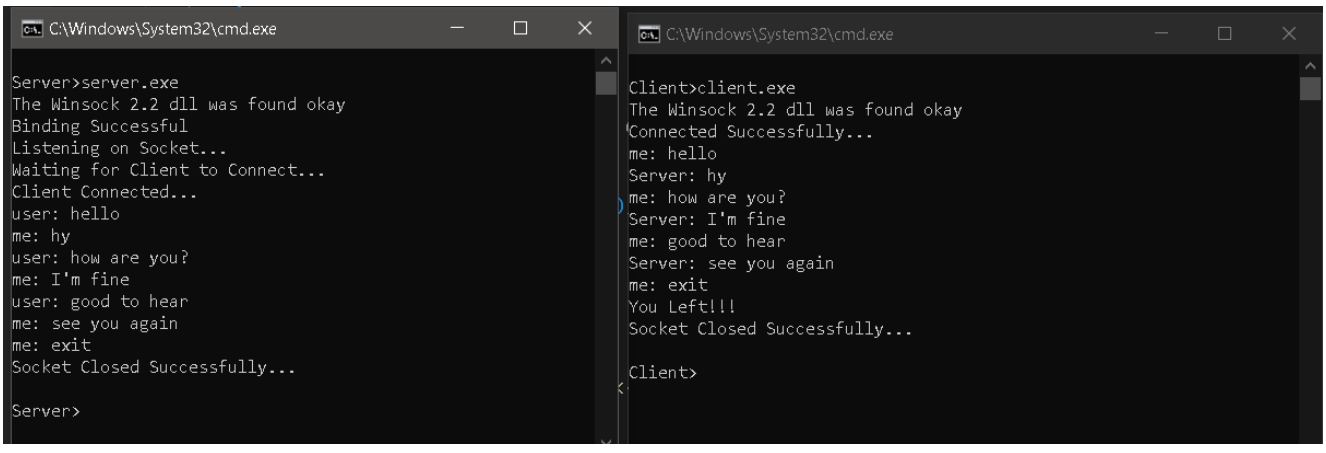
Client.cpp

```
#define WIN32_LEAN_AND_MEAN
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#include<WS2tcpip.h>
#include <iostream>
#include <string>

#pragma comment(lib,"ws2_32.lib")

int main()
{
    WSADATA wsaData;
    WORD Version = MAKEWORD(2, 2);
    SOCKET CSocket;
    sockaddr_in ServerInfo;

    int result = WSAStartup(Version, &wsaData);
    //check if the requested version is returned or not
    if (result != 0)
    {
        std::cout << "WSAStartup failed with error: " << result << ".\n";
        return 1;
    }
    if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
    {
        std::cout << "Could not find a usable version of Winsock.dll\n";
        WSACleanup();
        return 1;
    }
    else
    {
        std::cout << "The Winsock 2.2 dll was found okay\n";
    }

    //creating socket to connect with server
    CSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);


    //setting server info to connect to server
```

```
    ServerInfo.sin_family = AF_INET;
    ServerInfo.sin_addr.s_addr = inet_addr("127.0.0.1");
    ServerInfo.sin_port = htons(8080);

    result = connect(CSocket, (SOCKADDR*)&ServerInfo, sizeof(ServerInfo));

    //Checking if Connection is established Successfully or not
    if (result == SOCKET_ERROR)
    {
        std::cout << "Connecting Failed with Error: " <<
        WSAGetLastError() << ".\n";
        closesocket(CSocket);
        WSACleanup();
        return 1;
    }
    else
    {
        std::cout << "Connected Successfully...\n";
    }


    char msg[2048];
    int msglen;

    //process after connection is made
    while (true)
    {
        //sending msg
        ZeroMemory(msg, 2048);
        std::cout << "me: ";
        std::cin.getline(msg, 2048);
        if (strcmp(msg, "exit") == 0)
        {
            std::cout << "You Left!!!\n";
            break;
        }
        else
            send(CSocket, msg, strlen(msg), 0);
        //receiving msg
        ZeroMemory(msg, 2048);
        msglen = recv(CSocket, msg, 2048, 0);
        if (strcmp(msg, "exit") == 0)
        {
            std::cout << "Server Closed\n";
            break;
```

```
        }
        if (msglen != 0)
            std::cout << "Server: " << msg << "\n";
    }


    result = closesocket(CSocket);
    if (result == SOCKET_ERROR)
    {
        std::cout << "Failed to Close the Socket with Error: " <<
        WSAGetLastError() << ".\n";
        WSACleanup();
        return 1;
    }
    else
    {
        std::cout << "Socket Closed Successfully...\n";
    }
    //done using the Winsock.dll
    WSACleanup();

    return 0;
}
```

Output:



Conclusion:

In this Lab we implemented a simple Chat Server and Chat Client in C++ Programming Language and understood the working of Client-Server Application.