Bansilal Ramnath Agarwal Charitable Trust’s

# Vishwakarma Institute of Technology, Pune

(*An Autonomous Institute of Savitribai Phule Pune University*)



# Department of Artificial Intelligence and Data Science

|  |  |
| --- | --- |
| **Class/ Division** | SY-AIDS-A |
| **Name** | Archit Bagad |
| **Roll No.** | 25 |
| **GR No** | 12311951 |
| **Batch** | 03 |
| **Subject** | Computer Network |

**Lab Assignment 7**

**Implement Socket programming to create Client and Server to send Hello message. (Implement TCP and UDP Socket)**

1. **TCP**

**Code:**

1. **tcpServer.cpp**

#include <iostream>

#include <string>

#ifdef \_WIN32

    #include <winsock2.h>

    #pragma comment(lib, "Ws2\_32.lib")

#else

    #include <unistd.h>

    #include <arpa/inet.h>

    #include <sys/socket.h>

#endif

int main() {

    #ifdef \_WIN32

        WSADATA wsaData;

        if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

            std::cerr << "WSAStartup failed" << std::endl;

            return 1;

        }

    #endif

    SOCKET server\_fd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

    if (server\_fd == INVALID\_SOCKET) {

        std::cerr << "Socket creation failed" << std::endl;

        #ifdef \_WIN32

            WSACleanup();

        #endif

        return 1;

    }

    sockaddr\_in server\_addr{};

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

    server\_addr.sin\_port = htons(65432);

    if (bind(server\_fd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == SOCKET\_ERROR) {

        std::cerr << "Bind failed" << std::endl;

        #ifdef \_WIN32

            closesocket(server\_fd);

            WSACleanup();

        #endif

        return 1;

    }

    if (listen(server\_fd, 1) == SOCKET\_ERROR) {

        std::cerr << "Listen failed" << std::endl;

        #ifdef \_WIN32

            closesocket(server\_fd);

            WSACleanup();

        #endif

        return 1;

    }

    std::cout << "TCP Server listening on port 65432" << std::endl;

    while (true) {

        sockaddr\_in client\_addr{};

        int client\_len = sizeof(client\_addr);

        std::cout << "Waiting for client connection..." << std::endl;

        SOCKET client\_fd = accept(server\_fd, (struct sockaddr\*)&client\_addr, &client\_len);

        if (client\_fd == INVALID\_SOCKET) {

            std::cerr << "Accept failed" << std::endl;

            continue;

        }

        std::cout << "Client connected!" << std::endl;

        char buffer[1024] = {0};

        int bytes\_received = recv(client\_fd, buffer, 1024, 0);

        if (bytes\_received > 0) {

            std::cout << "Received: " << buffer << std::endl;

        } else {

            std::cout << "No data received or connection closed" << std::endl;

        }

        std::string response = "Hello from TCP Server!";

        send(client\_fd, response.c\_str(), response.length(), 0);

        std::cout << "Response sent to client" << std::endl;

        closesocket(client\_fd);

    }

    #ifdef \_WIN32

        closesocket(server\_fd);

        WSACleanup();

    #endif

    return 0;

}

1. **tcpClient.cpp**

#include <iostream>

#include <string>

#ifdef \_WIN32

    #include <winsock2.h>

    #pragma comment(lib, "Ws2\_32.lib")

#else

    #include <unistd.h>

    #include <arpa/inet.h>

    #include <sys/socket.h>

#endif

int main() {

    #ifdef \_WIN32

        WSADATA wsaData;

        if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

            std::cerr << "WSAStartup failed" << std::endl;

            return 1;

        }

    #endif

    SOCKET sock = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

    if (sock == INVALID\_SOCKET) {

        std::cerr << "Socket creation failed" << std::endl;

        #ifdef \_WIN32

            WSACleanup();

        #endif

        return 1;

    }

    sockaddr\_in server\_addr{};

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(65432);

    server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    std::cout << "Attempting to connect to server..." << std::endl;

    if (connect(sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == SOCKET\_ERROR) {

        std::cerr << "Connection failed" << std::endl;

        closesocket(sock);

        WSACleanup();

        return 1;

    }

    std::cout << "Connected to server!" << std::endl;

    std::string message = "Hello from TCP Client!";

    send(sock, message.c\_str(), message.length(), 0);

    std::cout << "Message sent to server" << std::endl;

    char buffer[1024] = {0};

    int bytes\_received = recv(sock, buffer, 1024, 0);

    if (bytes\_received > 0) {

        std::cout << "Server response: " << buffer << std::endl;

    } else {

        std::cout << "No response received" << std::endl;

    }

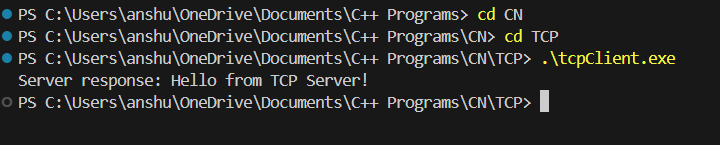
    closesocket(sock);

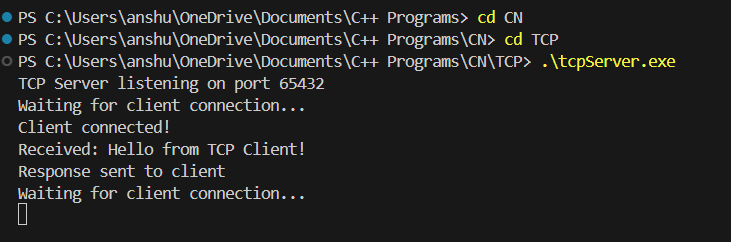
    WSACleanup();

    return 0;

}

**Output:**

****

****

1. **UDP**

**Code:**

1. **udpServer.cpp**

#include <iostream>

#include <string>

#include <winsock2.h>

// #include <ws2tcpip.h>  // Not needed for inet\_ntoa

#pragma comment(lib, "Ws2\_32.lib")

#define PORT 65433

int main() {

    // Initialize Winsock

    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

        std::cerr << "WSAStartup failed: " << WSAGetLastError() << std::endl;

        return 1;

    }

    // Create UDP socket

    SOCKET server\_fd = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

    if (server\_fd == INVALID\_SOCKET) {

        std::cerr << "Socket creation failed: " << WSAGetLastError() << std::endl;

        WSACleanup();

        return 1;

    }

    // Enable address reuse

    int opt = 1;

    if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR, (const char\*)&opt, sizeof(opt)) == SOCKET\_ERROR) {

        std::cerr << "Setsockopt failed: " << WSAGetLastError() << std::endl;

        closesocket(server\_fd);

        WSACleanup();

        return 1;

    }

    // Set up server address structure

    sockaddr\_in server\_addr{};

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

    server\_addr.sin\_port = htons(PORT);

    // Bind socket

    if (bind(server\_fd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == SOCKET\_ERROR) {

        std::cerr << "Bind failed: " << WSAGetLastError() << std::endl;

        closesocket(server\_fd);

        WSACleanup();

        return 1;

    }

    std::cout << "UDP Server listening on port " << PORT << " (Ctrl+C to exit)" << std::endl;

    while (true) {

        sockaddr\_in client\_addr{};

        int client\_len = sizeof(client\_addr);

        char buffer[1024] = {0};

        std::cout << "Waiting for message..." << std::endl;

        int bytes\_received = recvfrom(server\_fd, buffer, 1023, 0,

                                    (struct sockaddr\*)&client\_addr, &client\_len);

        if (bytes\_received == SOCKET\_ERROR) {

            std::cerr << "Receive failed: " << WSAGetLastError() << std::endl;

            continue;

        }

        buffer[bytes\_received] = '\0';  // Ensure null termination

        // Use inet\_ntoa instead of inet\_ntop

        std::string client\_ip = inet\_ntoa(client\_addr.sin\_addr);

        std::cout << "Received from " << client\_ip << ":" << ntohs(client\_addr.sin\_port)

                  << ": " << buffer << std::endl;

        // Send response

        std::string response = "Hello from UDP Server!";

        if (sendto(server\_fd, response.c\_str(), response.length(), 0,

                  (struct sockaddr\*)&client\_addr, client\_len) == SOCKET\_ERROR) {

            std::cerr << "Send failed: " << WSAGetLastError() << std::endl;

            continue;

        }

        std::cout << "Response sent" << std::endl;

    }

    closesocket(server\_fd);

    WSACleanup();

    return 0;

}

1. **udpClient.cpp**

#include <iostream>

#include <string>

#include <winsock2.h>

#pragma comment(lib, "Ws2\_32.lib")

#define PORT 65433

int main() {

    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

        std::cerr << "WSAStartup failed: " << WSAGetLastError() << std::endl;

        return 1;

    }

    SOCKET sock = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

    if (sock == INVALID\_SOCKET) {

        std::cerr << "Socket creation failed: " << WSAGetLastError() << std::endl;

        WSACleanup();

        return 1;

    }

    sockaddr\_in server\_addr{};

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(PORT);

    const char\* server\_ip = "127.0.0.1";

    // Use inet\_addr instead of inet\_pton

    server\_addr.sin\_addr.s\_addr = inet\_addr(server\_ip);

    if (server\_addr.sin\_addr.s\_addr == INADDR\_NONE) {

        std::cerr << "Invalid address: " << WSAGetLastError() << std::endl;

        closesocket(sock);

        WSACleanup();

        return 1;

    }

    std::string message = "Hello from UDP Client!";

    int server\_len = sizeof(server\_addr);

    std::cout << "Sending message to " << server\_ip << ":" << PORT << "..." << std::endl;

    if (sendto(sock, message.c\_str(), message.length(), 0,

              (struct sockaddr\*)&server\_addr, server\_len) == SOCKET\_ERROR) {

        std::cerr << "Send failed: " << WSAGetLastError() << std::endl;

        closesocket(sock);

        WSACleanup();

        return 1;

    }

    std::cout << "Message sent" << std::endl;

    char buffer[1024] = {0};

    int bytes\_received = recvfrom(sock, buffer, 1023, 0,

                                (struct sockaddr\*)&server\_addr, &server\_len);

    if (bytes\_received == SOCKET\_ERROR) {

        std::cerr << "Receive failed: " << WSAGetLastError() << std::endl;

    } else {

        buffer[bytes\_received] = '\0';

        std::cout << "Server response: " << buffer << std::endl;

    }

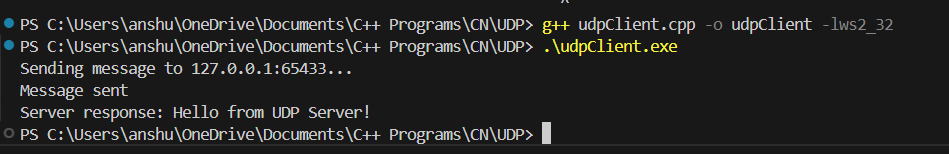
    closesocket(sock);

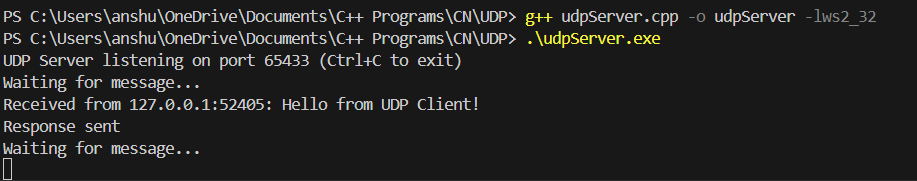
    WSACleanup();

    return 0;

}

**Output:**

****

****