

# Bansilal Ramnath Agarwal Charitable Trust’s

# Vishwakarma Institute of Technology

#### (An Autonomous Institute affiliated to Savitribai Phule Pune University)

**COMPUTER NETWORKS LAB ASSIGNMENTS**

|  |  |
| --- | --- |
| NAME | ABBAS MADHVASWALA |
| PRN | 12110285 |
| DIVISION | CS-C |
| ROLL.NO | 13 |
| BATCH | B3 |

**LAB ASSIGNMENT NO : 06**

**PROBLEM STATEMENT**

**Problem Statement:** Develop a client server using UDP Berkeley socket primitives for chat application in peer to peer and client server mode.

Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

# Client.c

#include <stdio.h> #include <stdlib.h> #include <string.h> #include <winsock2.h>

#define SERVER\_PORT 5600 int main() {

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) { perror("WSAStartup failed");

return 1;

}

int client\_socket;

struct sockaddr\_in server\_addr;

char server\_ip[] = "127.0.0.1"; // Change this to the server's IP address

// Create a socket

client\_socket = socket(AF\_INET, SOCK\_STREAM, 0); if (client\_socket == INVALID\_SOCKET) {

perror("Socket creation failed"); WSACleanup();

return 1;

}

// Prepare the server address structure server\_addr.sin\_family = AF\_INET; server\_addr.sin\_port = htons(SERVER\_PORT); server\_addr.sin\_addr.s\_addr = inet\_addr(server\_ip);

// Connect to the server

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

perror("Connection failed"); closesocket(client\_socket); WSACleanup();

return 1;

}

printf("Connected to the server.\n");

char buffer[1024]; int bytes\_received;

while (1) {

printf("Enter a message (type 'bye' to exit): "); fgets(buffer, sizeof(buffer), stdin);

// Send the user input to the server send(client\_socket, buffer, strlen(buffer), 0);

// Check if the user wants to exit if (strcmp(buffer, "bye\n") == 0) {

break;

}

// Clear the buffer

memset(buffer, 0, sizeof(buffer));

// Receive data from the server

bytes\_received = recv(client\_socket, buffer, sizeof(buffer), 0); if (bytes\_received == SOCKET\_ERROR) {

perror("Connection closed or error"); break;

}

printf("Server response: %s", buffer);

}

// Close the socket and clean up Winsock closesocket(client\_socket); WSACleanup();

return 0;

}

# Server.c

#include <stdio.h> #include <stdlib.h> #include <string.h>

#include <winsock2.h> // Include the Winsock2 header #define PORT 5600

int main() {

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) { perror("WSAStartup failed");

return 1;

}

int server\_socket, client\_socket;

struct sockaddr\_in server\_addr, client\_addr; int client\_addr\_len = sizeof(client\_addr);

// Create a socket

server\_socket = socket(AF\_INET, SOCK\_STREAM, 0); if (server\_socket == INVALID\_SOCKET) {

perror("Socket creation failed"); WSACleanup(); // Clean up Winsock return 1;

}

// Prepare the server address structure server\_addr.sin\_family = AF\_INET; server\_addr.sin\_addr.s\_addr = INADDR\_ANY; server\_addr.sin\_port = htons(PORT);

// Bind the socket to the specified port

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

perror("Binding failed"); closesocket(server\_socket); WSACleanup(); // Clean up Winsock return 1;

}

// Listen for incoming connections

if (listen(server\_socket, 5) == SOCKET\_ERROR) {

perror("Listening failed"); closesocket(server\_socket); WSACleanup(); // Clean up Winsock return 1;

}

printf("Waiting for a client...\n");

// Accept a client connection

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

if (client\_socket == INVALID\_SOCKET) { perror("Accepting client connection failed"); closesocket(server\_socket);

WSACleanup(); // Clean up Winsock return 1;

}

printf("Client connected\n");

char buffer[1024]; int bytes\_received;

while (1) {

// Receive data from the client

bytes\_received = recv(client\_socket, buffer, sizeof(buffer), 0); if (bytes\_received <= 0) {

perror("Connection closed or error"); break;

}

buffer[bytes\_received] = '\0'; printf("Received: %s", buffer);

// Check if the client wants to exit if (strcmp(buffer, "bye") == 0) {

break;

}

}

// Close sockets and clean up Winsock closesocket(client\_socket); closesocket(server\_socket); WSACleanup();

return 0;

}

# Wireshark -

