

# 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 : 05**

**PROBLEM STATEMENT**

**Problem Statement:** Develop a client server using TCP Berkeley socket primitives to transfer a file 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\_IP "127.0.0.1"

#define PORT 12345

#define BUFFER\_SIZE 1024

int main() {

WSADATA wsa;

SOCKET client\_socket;

struct sockaddr\_in server\_addr; char buffer[BUFFER\_SIZE];

// Initialize Winsock

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) { perror("Error initializing Winsock"); return 1;

}

// Create socket

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

perror("Error creating socket"); WSACleanup();

return 1;

}

server\_addr.sin\_family = AF\_INET; server\_addr.sin\_port = htons(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("Error connecting to server"); closesocket(client\_socket); WSACleanup();

return 1;

}

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

// File transfer

FILE \*file = fopen("client\_file.txt", "wb"); if (file == NULL) {

perror("Error opening file"); closesocket(client\_socket); WSACleanup();

return 1;

}

while (1) {

int bytes\_received = recv(client\_socket, buffer, sizeof(buffer), 0);

if (bytes\_received <= 0) break;

fwrite(buffer, 1, bytes\_received, file);

}

fclose(file); closesocket(client\_socket); WSACleanup();

printf("File received successfully.\n"); return 0;

}

# Server.c

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

#define PORT 12345

#define BUFFER\_SIZE 1024

int main() {

WSADATA wsa;

SOCKET server\_socket, client\_socket;

struct sockaddr\_in server\_addr, client\_addr; int client\_addr\_len = sizeof(client\_addr); char buffer[BUFFER\_SIZE];

// Initialize Winsock

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) { perror("Error initializing Winsock"); return 1;

}

// Create socket

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

perror("Error creating socket"); WSACleanup();

return 1;

}

// Bind

server\_addr.sin\_family = AF\_INET; server\_addr.sin\_port = htons(PORT); server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

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

perror("Error binding"); closesocket(server\_socket); WSACleanup();

return 1;

}

// Listen

if (listen(server\_socket, 5) == SOCKET\_ERROR) { perror("Error listening");

closesocket(server\_socket); WSACleanup();

return 1;

}

printf("Server listening on port %d...\n", PORT);

// Accept a connection

client\_socket = accept(server\_socket, (struct sockaddr

\*)&client\_addr, &client\_addr\_len);

if (client\_socket == INVALID\_SOCKET) { perror("Error accepting connection"); closesocket(server\_socket); WSACleanup();

return 1;

}

printf("Client connected.\n");

// File transfer

FILE \*file = fopen("server\_file.txt", "rb"); if (file == NULL) {

perror("Error opening file"); closesocket(client\_socket); closesocket(server\_socket); WSACleanup();

return 1;

}

while (1) {

int bytes\_read = fread(buffer, 1, sizeof(buffer), file); if (bytes\_read <= 0)

break;

int bytes\_sent = send(client\_socket, buffer, bytes\_read, 0); if (bytes\_sent == SOCKET\_ERROR) {

perror("Error sending data"); fclose(file); closesocket(client\_socket); closesocket(server\_socket); WSACleanup();

return 1;

}

}

fclose(file); closesocket(client\_socket);

closesocket(server\_socket); WSACleanup();

printf("File sent successfully.\n"); return 0;

}

# Wireshark -

