**Name : Shrinivas Hatyalikar**

**Roll No. : 26**

**Div : CS-B**

**Assignment 3**

**Write a program to stimulate Go back N and Selective Repeat Modes of Sliding Window Protocol in peer-to-peer mode and demonstrate the packets captured traces using the Wireshark Packet Analyzer Tool for peer to peer mode.**

**1]Go Back to N:**

**Code:**

1.Client Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock.h>

#define SERVER\_ADDR "127.0.0.1"

#define PORT 9090

void transmission(SOCKET sock, int N, int tf, int \*tt)

{

    int i = 1;

    char buffer[128];

    while (i <= tf)

    {

        int z = 0;

        for (int k = i; k < i + N && k <= tf; k++)

        {

            sprintf(buffer, "%d", k);

            send(sock, buffer, strlen(buffer), 0);

            printf("Sending Frame %d...\n", k);

            (\*tt)++;

        }

        for (int k = i; k < i + N && k <= tf; k++)

        {

            fd\_set readfds;

            struct timeval timeout;

            FD\_ZERO(&readfds);

            FD\_SET(sock, &readfds);

            timeout.tv\_sec = 2; // 2 seconds timeout for acknowledgment

            timeout.tv\_usec = 0;

            int activity = select(sock + 1, &readfds, NULL, NULL, &timeout);

            // If timeout or error, break and retransmit

            if (activity <= 0)

            {

                printf("Timeout for acknowledgment of Frame %d...\n", k);

                break;

            }

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

            int bytesRead = recv(sock, buffer, sizeof(buffer) - 1, 0); // reserve 1 byte for the null terminator

            if (bytesRead > 0)

            {

                buffer[bytesRead] = '\0'; // null-terminate the received data

            }

            int ack = atoi(buffer);

            if (ack == k)

            {

                printf("Acknowledgment for Frame %d...\n", k);

                z++;

            }

            else

            {

                printf("Expected acknowledgment for Frame %d, but received for Frame %d\n", k, ack);

                break;

            }

        }

        i += z;

    }

}

int main()

{

    int tf, N, tt = 0;

    WSADATA wsaData;

    SOCKET sock;

    struct sockaddr\_in serverAddr;

    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0)

    {

        printf("WSAStartup failed. Error: %d\n", WSAGetLastError());

        return 1;

    }

    sock = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sock == INVALID\_SOCKET)

    {

        printf("Socket creation failed. Error: %d\n", WSAGetLastError());

        WSACleanup();

        return 1;

    }

    memset(&serverAddr, 0, sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_addr.s\_addr = inet\_addr(SERVER\_ADDR);

    serverAddr.sin\_port = htons(PORT);

    if (connect(sock, (struct sockaddr \*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR)

    {

        printf("Error connecting to server. Error: %d\n", WSAGetLastError());

        closesocket(sock);

        WSACleanup();

        return 1;

    }

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

    printf("Enter the Total number of frames: ");

    scanf("%d", &tf);

    printf("Enter the Window Size: ");

    scanf("%d", &N);

    transmission(sock, N, tf, &tt);

    printf("Total number of frames which were sent and resent are: %d\n", tt);

    closesocket(sock);

    WSACleanup();

    return 0;

}

2.Server Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock.h>

#include <time.h>

#define PORT 9090

int main()

{

    WSADATA wsaData;

    SOCKET listenSocket, clientSocket;

    struct sockaddr\_in serverAddr, clientAddr;

    int clientAddrSize = sizeof(clientAddr);

    char buffer[128];

    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0)

    {

        printf("WSAStartup failed. Error: %d\n", WSAGetLastError());

        return 1;

    }

    listenSocket = socket(AF\_INET, SOCK\_STREAM, 0);

    if (listenSocket == INVALID\_SOCKET)

    {

        printf("Socket creation failed. Error: %d\n", WSAGetLastError());

        WSACleanup();

        return 1;

    }

    memset(&serverAddr, 0, sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

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

    serverAddr.sin\_port = htons(PORT);

    if (bind(listenSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR)

    {

        printf("Bind failed. Error: %d\n", WSAGetLastError());

        closesocket(listenSocket);

        WSACleanup();

        return 1;

    }

    if (listen(listenSocket, 5) == SOCKET\_ERROR)

    {

        printf("Listen failed. Error: %d\n", WSAGetLastError());

        closesocket(listenSocket);

        WSACleanup();

        return 1;

    }

    printf("Server is listening for connections...\n");

    clientSocket = accept(listenSocket, (struct sockaddr \*)&clientAddr, &clientAddrSize);

    if (clientSocket == INVALID\_SOCKET)

    {

        printf("Accept failed. Error: %d\n", WSAGetLastError());

        closesocket(listenSocket);

        WSACleanup();

        return 1;

    }

    printf("Connection accepted from client.\n");

    srand((unsigned int)time(NULL));

    while (1)

    {

        memset(buffer, 0, sizeof(buffer)); // Clear the buffer

        int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);

        if (bytesReceived <= 0)

        {

            printf("Connection closed or error occurred. Ending session.\n");

            break;

        }

        int frame = atoi(buffer);

        int f = rand() % 2;

        if (f == 0)

        { // Send acknowledgment

            sprintf(buffer, "%d", frame);

            send(clientSocket, buffer, strlen(buffer), 0);

        }

        else

        {

            printf("Simulating frame drop for frame: %d\n", frame);

        }

    }

    closesocket(clientSocket);

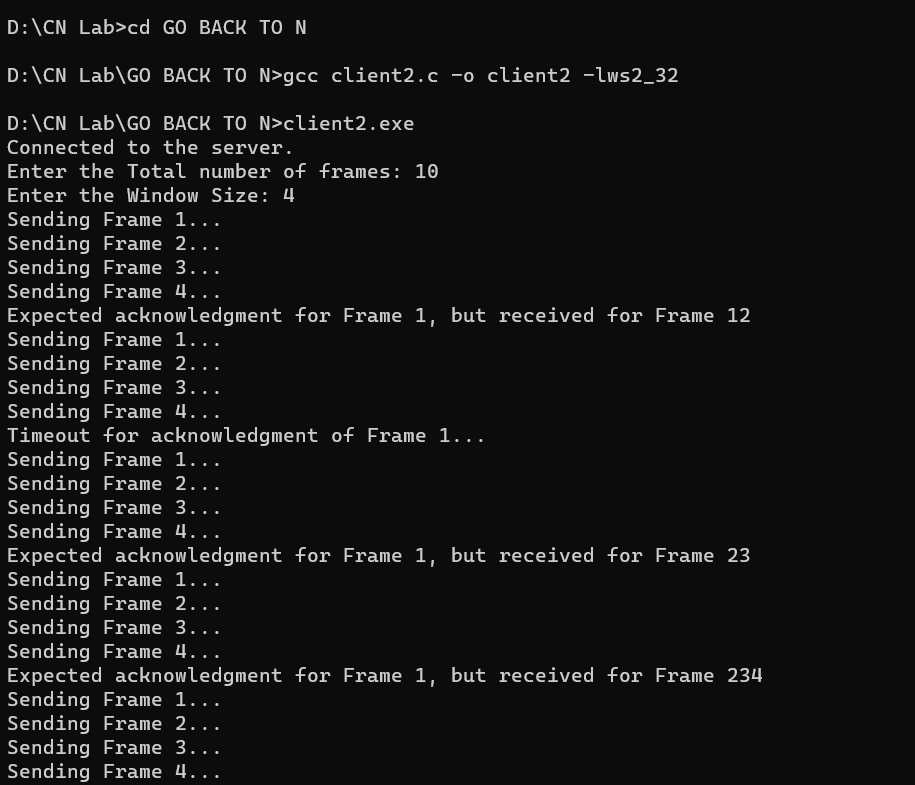
    closesocket(listenSocket);

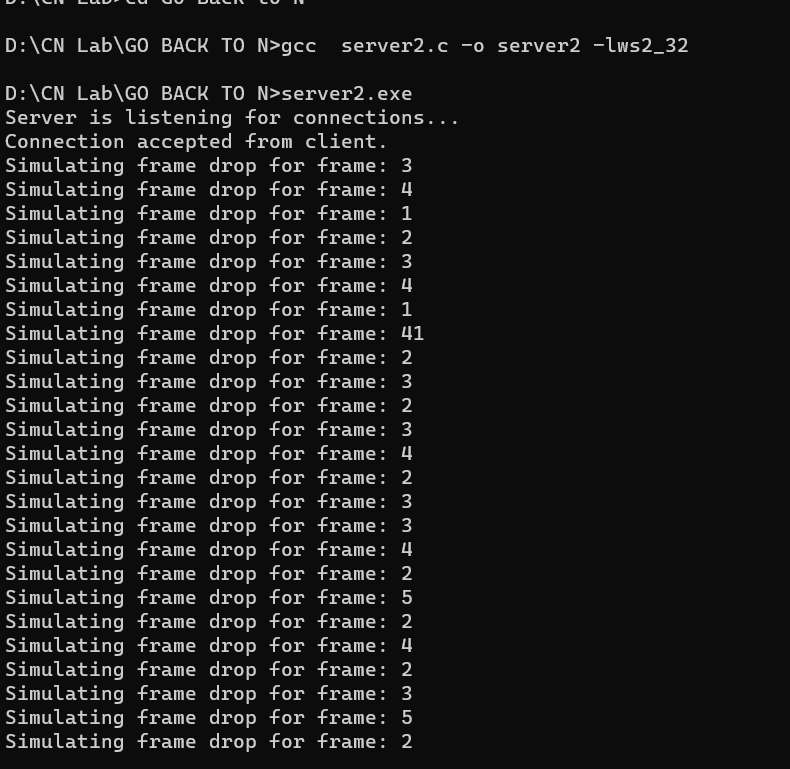
    WSACleanup();

    return 0;

}

**Output:**





**2]Selective Repeat:**

**Code:**

1.Client Code:

#include <stdio.h>

#include <winsock2.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

    WSADATA wsaData;

    SOCKET connection;

    struct sockaddr\_in serverAddr;

    int v[9]; // Array to store data

    int n = 0;

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

        printf("WSAStartup failed.\n");

        return 1;

    }

    // Create a socket

    connection = socket(AF\_INET, SOCK\_STREAM, 0);

    if (connection == INVALID\_SOCKET) {

        printf("Error in socket creation.\n");

        return 1;

    }

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(8011);

    serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); // Use the appropriate server IP address

    // Connect to the server

    if (connect(connection, (struct sockaddr\*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR) {

        printf("Connection failed.\n");

        return 1;

    }

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

    // Read the number of frames from the server

    int p;

    if (recv(connection, (char\*)&p, sizeof(p), 0) == SOCKET\_ERROR) {

        printf("Error receiving number of frames.\n");

        closesocket(connection);

        WSACleanup();

        return 1;

    }

    printf("No of frame is: %d\n", p);

    // Receive frames from the server

    for (int i = 0; i < p; i++) {

        if (recv(connection, (char\*)&v[i], sizeof(v[i]), 0) == SOCKET\_ERROR) {

            printf("Error receiving frame %d.\n", i);

            closesocket(connection);

            WSACleanup();

            return 1;

        }

        printf("Received frame is: %d\n", v[i]);

    }

    v[5] = -1;

    // Request retransmission for frames with value -1

    for (int i = 0; i < p; i++) {

        if (v[i] == -1) {

            printf("Request to retransmit packet no %d again!!\n", i + 1);

            n = i;

            if (send(connection, (char\*)&n, sizeof(n), 0) == SOCKET\_ERROR) {

                printf("Error sending request for retransmission.\n");

                closesocket(connection);

                WSACleanup();

                return 1;

            }

        }

    }

    // Receive the retransmitted frame

    if (recv(connection, (char\*)&v[n], sizeof(v[n]), 0) == SOCKET\_ERROR) {

        printf("Error receiving retransmitted frame.\n");

        closesocket(connection);

        WSACleanup();

        return 1;

    }

    printf("Received frame is: %d\n", v[n]);

    printf("Quitting\n");

    // Clean up and close the socket

    closesocket(connection);

    WSACleanup();

    return 0;

}

2.Server Code:

#include <stdio.h>

#include <winsock2.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

    WSADATA wsaData;

    SOCKET serverSocket, clientSocket;

    struct sockaddr\_in serverAddr, clientAddr;

    int a[] = { 30, 40, 50, 60, 70, 80, 90, 100, 110 };

    int y = sizeof(a) / sizeof(a[0]);

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

        printf("WSAStartup failed.\n");

        return 1;

    }

    serverSocket = socket(AF\_INET, SOCK\_STREAM, 0);

    if (serverSocket == INVALID\_SOCKET) {

        printf("Error in socket creation.\n");

        return 1;

    }

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(8011);

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

    if (bind(serverSocket, (struct sockaddr\*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR) {

        printf("Error in binding.\n");

        return 1;

    }

    if (listen(serverSocket, 10) == 0) {

        printf("Waiting for connection...\n");

    } else {

        printf("Error in listening.\n");

    }

    int addr\_size = sizeof(clientAddr);

    clientSocket = accept(serverSocket, (struct sockaddr\*)&clientAddr, &addr\_size);

    printf("The number of packets sent is: %d\n", y);

    // Send the number of packets to the client

    if (send(clientSocket, (char\*)&y, sizeof(y), 0) == SOCKET\_ERROR) {

        printf("Error sending the number of packets.\n");

        closesocket(clientSocket);

        closesocket(serverSocket);

        WSACleanup();

        return 1;

    }

    // Send the packets to the client

    for (int i = 0; i < y; i++) {

        if (send(clientSocket, (char\*)&a[i], sizeof(a[i]), 0) == SOCKET\_ERROR) {

            printf("Error sending packet %d.\n", i);

            closesocket(clientSocket);

            closesocket(serverSocket);

            WSACleanup();

            return 1;

        }

    }

    int k;

    // Receive the index 'k' from the client

    if (recv(clientSocket, (char\*)&k, sizeof(k), 0) == SOCKET\_ERROR) {

        printf("Error receiving index 'k' from the client.\n");

        closesocket(clientSocket);

        closesocket(serverSocket);

        WSACleanup();

        return 1;

    }

    // Send the packet 'a[k]' to the client

    if (send(clientSocket, (char\*)&a[k], sizeof(a[k]), 0) == SOCKET\_ERROR) {

        printf("Error sending packet 'a[%d]' to the client.\n", k);

        closesocket(clientSocket);

        closesocket(serverSocket);

        WSACleanup();

        return 1;

    }

    printf("Quitting\n");

    // Clean up and close the sockets

    closesocket(clientSocket);

    closesocket(serverSocket);

    WSACleanup();

    return 0;

}

**Output:**



