Assignment 4

**Computer Network**

**Name of the student: Swaraj Phand**

**Roll No. 7**

**PRN- 12211292**

**Class- TY – D -B1**

**Div: C Batch: 3**

**Date of performance: 24/01/2024**

**CODE-**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <thread> // For std::this\_thread::sleep\_for

#include <chrono> // For std::chrono::seconds

#define TOTAL\_PACKETS 10

#define WINDOW\_SIZE 4

#define LOSS\_RATE 20 // Percentage of packet loss

enum Protocol {

GO\_BACK\_N,

SELECTIVE\_REPEAT

};

class SlidingWindow {

private:

int base;

int nextSeqNum;

int protocol;

std::vector<int> window;

std::vector<bool> ackReceived;

public:

SlidingWindow(int protocolType) : base(0), nextSeqNum(0), protocol(protocolType) {

window.resize(TOTAL\_PACKETS, -1);

ackReceived.resize(TOTAL\_PACKETS, false);

}

void sendPackets() {

while (base < TOTAL\_PACKETS) {

for (int i = 0; i < WINDOW\_SIZE && nextSeqNum < TOTAL\_PACKETS; i++) {

if (window[base + i] == -1) {

sendPacket(nextSeqNum);

nextSeqNum++;

}

}

receiveAcks();

slideWindow();

std::this\_thread::sleep\_for(std::chrono::seconds(1)); // Simulating delay can also use other //libraries

}

}

private:

void sendPacket(int seqNum) {

window[seqNum] = seqNum;

std::cout << "Sent packet: " << seqNum << std::endl;

}

void receiveAcks() {

for (int i = base; i < base + WINDOW\_SIZE && i < TOTAL\_PACKETS; i++) {

if (ackReceived[i] == false) {

if (simulatePacketLoss()) {

std::cout << "Packet " << i << " lost.\n";

} else {

ackReceived[i] = true;

std::cout << "Received ACK for packet: " << i << std::endl;

}

}

}

}

void slideWindow() {

if (protocol == GO\_BACK\_N) {

for (int i = base; i < base + WINDOW\_SIZE && i < TOTAL\_PACKETS; i++) {

if (ackReceived[i] == false) {

nextSeqNum = i;

break;

} else {

base++;

}

}

} else if (protocol == SELECTIVE\_REPEAT) {

while (ackReceived[base] == true && base < TOTAL\_PACKETS) {

base++;

}

}

}

bool simulatePacketLoss() {

return (rand() % 100) < LOSS\_RATE;

}

};

int main() {

srand(time(0));

std::cout << "Go-Back-N Protocol Simulation:\n";

SlidingWindow gbn(GO\_BACK\_N);

gbn.sendPackets();

std::cout << "\nSelective Repeat Protocol Simulation:\n";

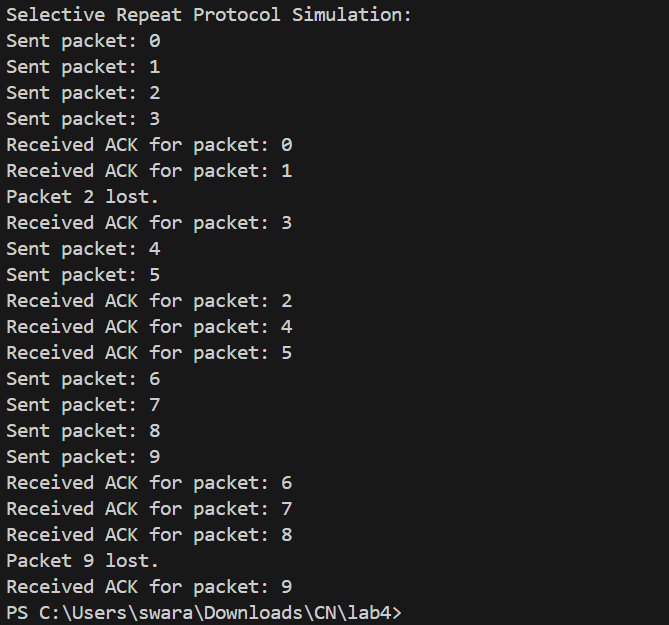
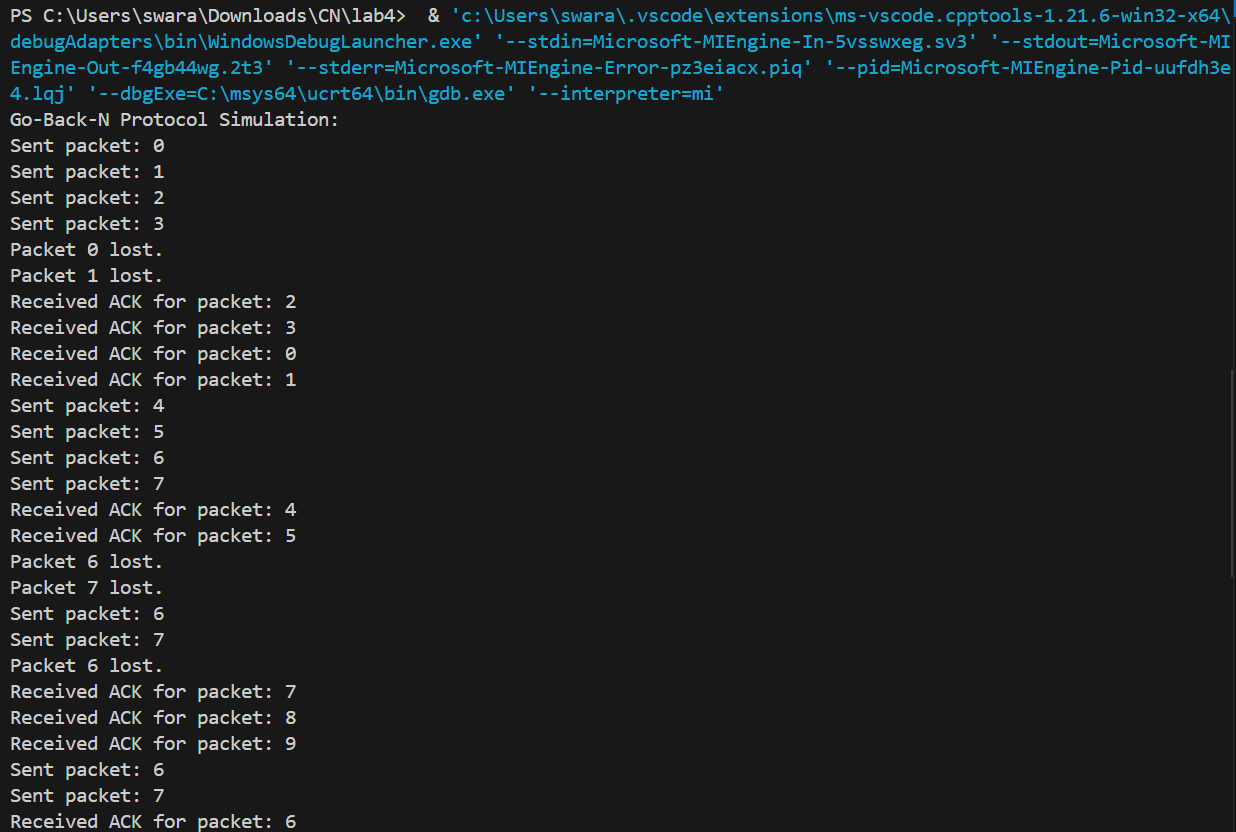
SlidingWindow sr(SELECTIVE\_REPEAT);

sr.sendPackets();

return 0;

}

**OUTPUT –**

****