**Practical 4 – Computer Networks Lab**

**Name:** Neeraj Belsare

**Roll No.:** 79

**Batch:** A4

**PRN:** 202101040133

**Title:**

Sliding Window Mechanism

**Aim:**

Write a program to implement sliding window mechanisms using

1. Stop and Wait ARQ

2. Go Back N ARQ

3. Selective Repeat ARQ

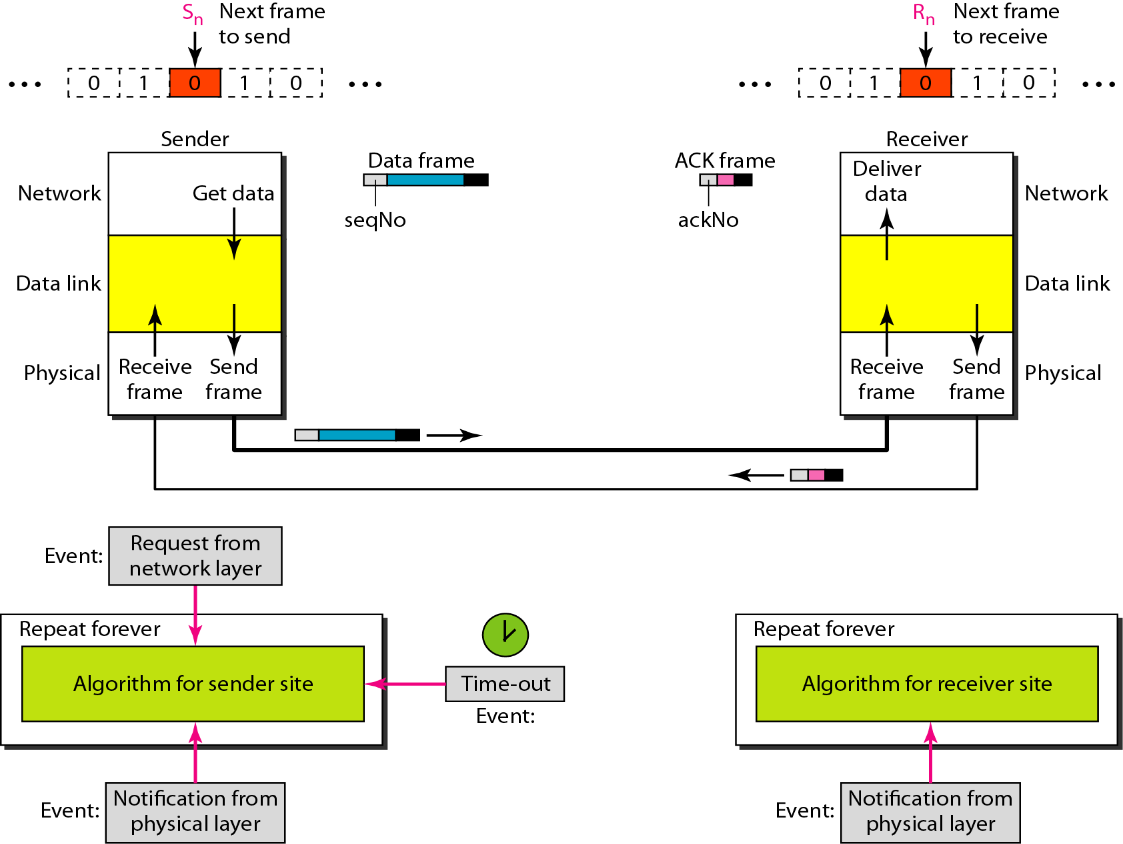
**Theory:**

1. **Stop-and-Wait ARQ**

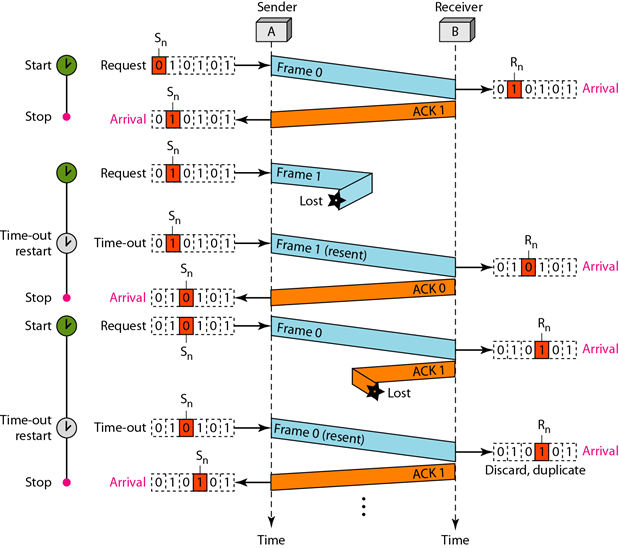
Stop-and-Wait ARQ (Automatic Repeat reQuest) is a simple error control protocol used for reliable data transmission over a communication channel. In Stop-and-Wait ARQ, the sender sends a single data frame to the receiver and waits for an acknowledgement (ACK) from the receiver. If the sender receives the ACK, it knows that the frame has been successfully received, and it can then send the next frame. If the sender does not receive the ACK within a specified timeout indicating an error, it retransmits the same frame.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting the frame when the timer expires.

**Design of the Stop-and-Wait ARQ Protocol**



**Flow Diagram of the Stop-and-Wait ARQ Protocol**



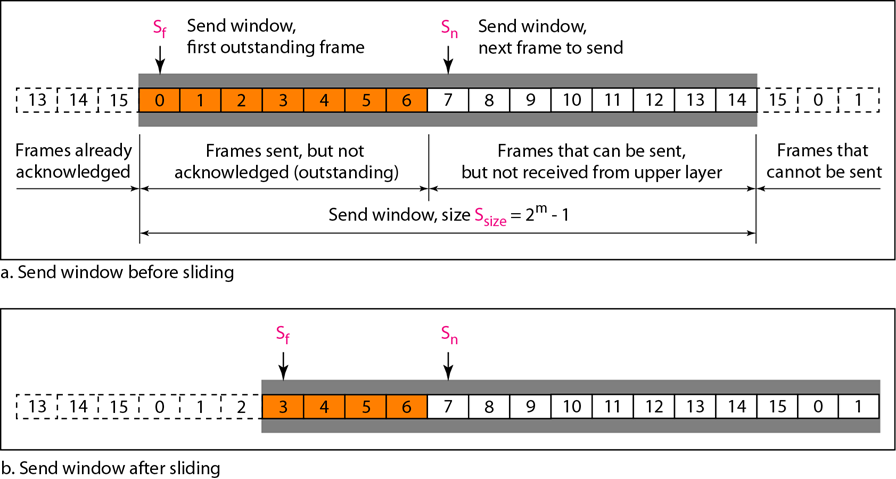
1. **Go Back N ARQ**

Go-Back-N is an Automatic Repeat reQuest (ARQ) protocol for reliable data communication. It ensures data packets are transmitted and received correctly over an unreliable network. If an acknowledgement (ACK) is not received for a sent packet within a certain time window, all packets starting from the unacknowledged packet are retransmitted. The send window is an abstract concept defining an imaginary box of size 2m−1 with three variables: Sf, Sn, and Ssize. The send window slides one or more slots when a valid acknowledgement arrives.

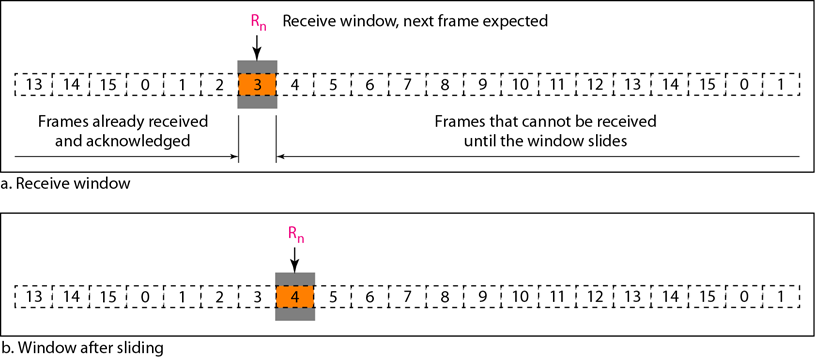
Sf is Send window, first outstanding frame (frame sent but not yet acknowledged). Sn is Send window, next frame to send. Ssize is Send window size = 2m - 1.

The receive window is an abstract concept defining an imaginary box of size 1 with a single variable Rn. The window slides when a correct frame has arrived; it slides one slot at a time. Rn is next frame to receive.

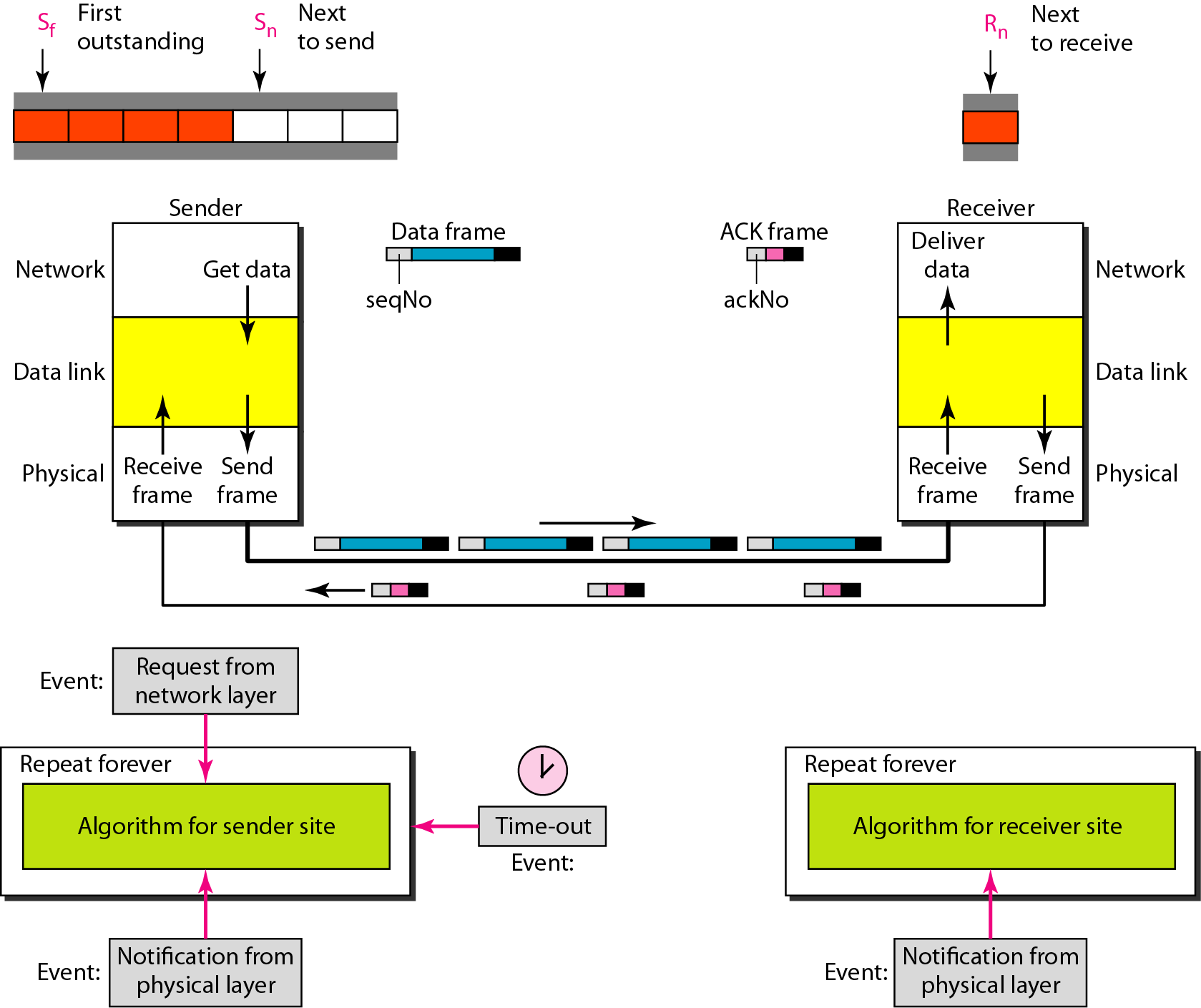
**Send window for Go-Back-N**



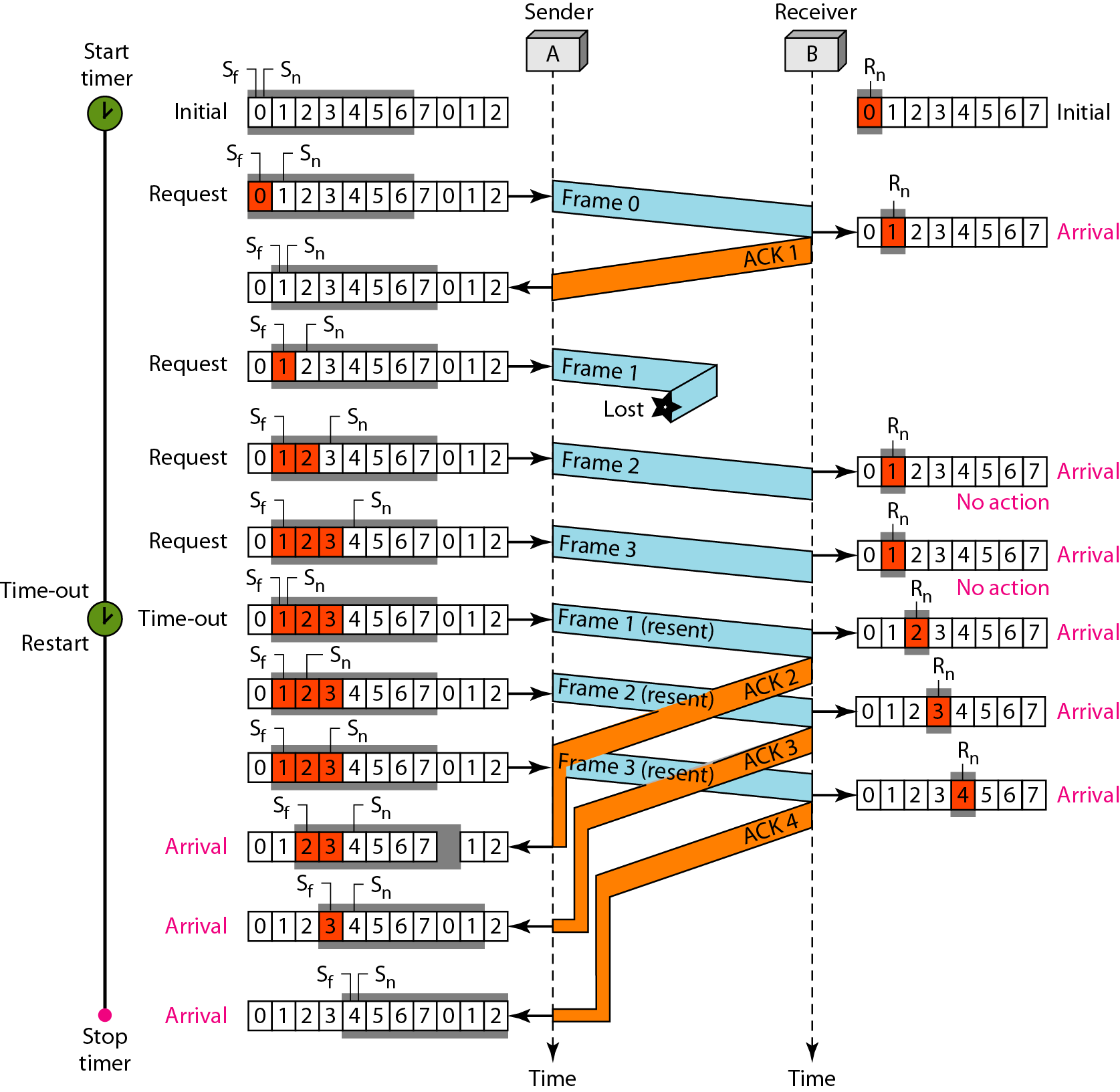
**Receive window for Go-Back-N ARQ**



**Design of Go-Back-N ARQ**

****

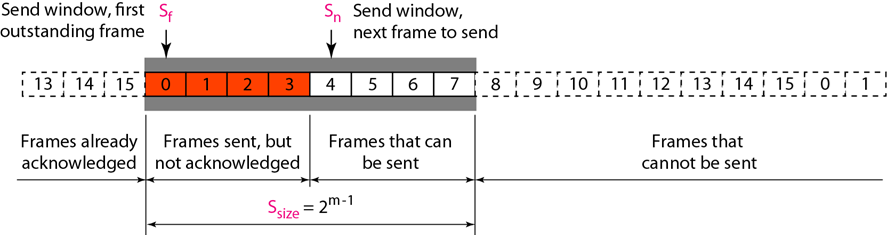
**Flow Diagram of the Go-Back-N ARQ Protocol**

****

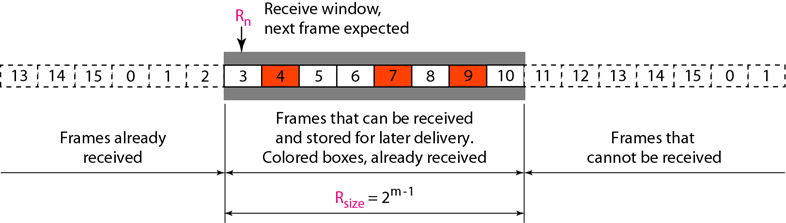
1. **Selective Repeat ARQ**

Selective Repeat is another Automatic Repeat reQuest (ARQ) protocol used for reliable data communication. Unlike Go-Back-N, Selective Repeat allows the receiver to individually acknowledge correctly received frames, enabling the sender to retransmit only the frames that were not received correctly.

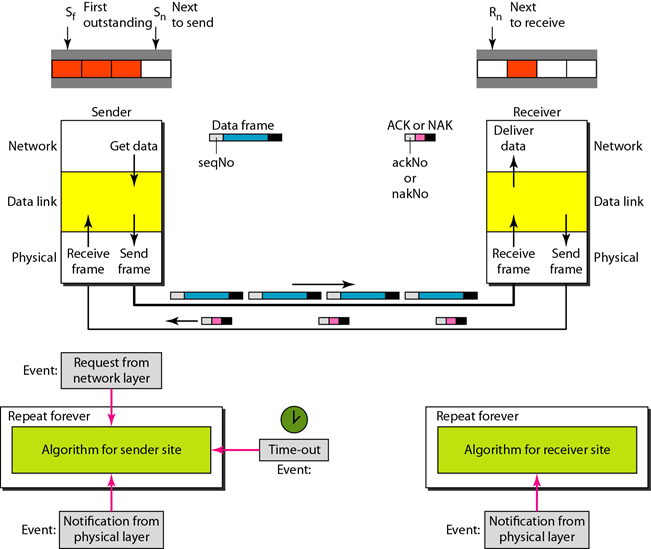
**Sender window for Selective ARQ**



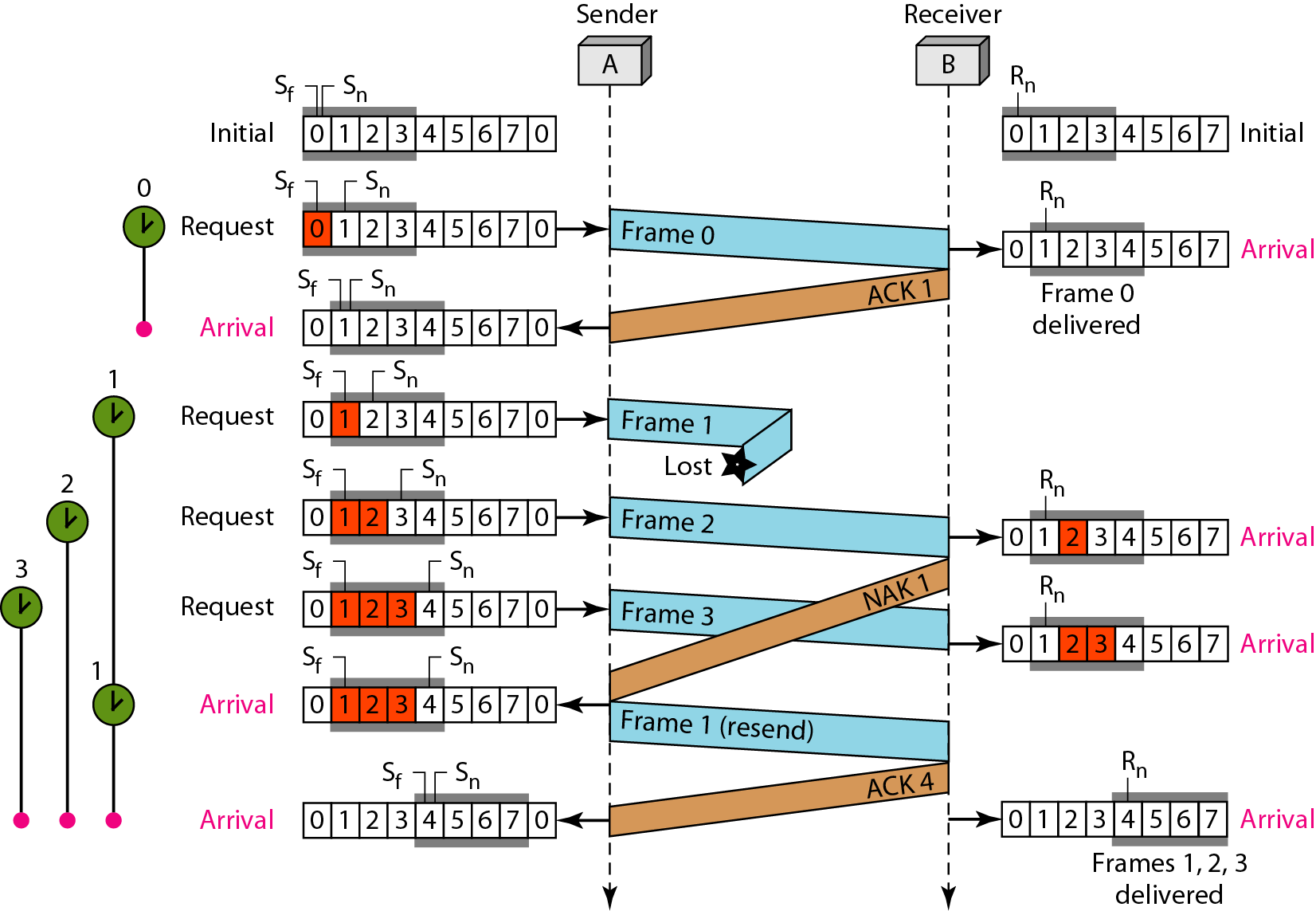
**Receiver window for Selective ARQ**



**Design for Selective ARQ**



**Flow Diagram of the Selective Repeat ARQ Protocol**



**Procedure/code:**

#include<iostream>

#include<cstdlib>

#include<ctime>

#include<unistd.h>

#include<math.h>

#define TOT\_FRAMES 500

#define FRAMES\_SEND 10

using namespace std;

class stop\_wait\_arq {

public:

void execute() {

srand(time(0));

int frameNumber = 1, n;

int receiverAck;

cout << "Enter the number of frames: " << endl;

cin >> n;

while (frameNumber <= n) {

sleep(1);

cout << "Sender: Sending Frame " << frameNumber << endl;

sleep(1);

receiverAck = rand() % 2;

if (receiverAck == 1) {

cout << "Sender: Frame " << frameNumber << " Acknowledged by Receiver" << endl;

frameNumber++;

} else {

cout << "Sender: Frame " << frameNumber << " Not Acknowledged, Resending..." << endl;

}

sleep(1);

}

}

};

class go\_back\_n\_arq {

public:

void execute() {

int n,N;

int no = 0;

srand(time(NULL));

cout<<"Enter the number of frames: ";

cin>>n;

cout<<"Enter the window size: ";

cin>>N;

int i=1;

while(i<=n) {

int x=0;

for(int j = i; j < i+N && j <= n; j++) {

cout << "Sent frame " << j << endl;

sleep(1);

no++;

}

for(int j = i; j < i+N && j <= n; j++) {

int flag = rand() % 2;

if(!flag) {

cout << "Acknowledgment for frame " << j << " received." << endl;

sleep(1);

x++;

} else {

cout << "Frame " << j << " not received." << endl;

sleep(1);

cout << "Retransmitting window" << endl;

break;

}

}

cout<<endl;

i+=x;

}

cout << "Total number of transmissions: " << no << endl;

}

};

class selective\_repeat\_arq {

private:

int f,N;

public:

void execute() {

printf("Enter the number of bits for the sequence number: ");

scanf("%d", &N);

f=pow(2, N-1);

for(int j = 0; j < f; j++) {

printf("\nSender: Frame %d is sent.", j);

sleep(1);

}

for(int i=0; i<f; i++) {

printf("\n");

int flag = rand() % 2;

if(!flag) {

printf("\nReceiver: Frame %d received correctly.\nAcknowledgement for Frame %d received.", i, i);

sleep(1);

} else {

printf("\nReceiver: Frame %d received correctly.\nAcknowledgement for Frame %d lost.",i,i);

printf("\nSender timeout.\nResending the frame...");

sleep(1);

}

}

}

};

int main() {

selective\_repeat\_arq sr;

stop\_wait\_arq st;

go\_back\_n\_arq gb;

int ch=0;

do {

cout<<"\n1.Stop and Wait ARQ\n2.Go Back N ARQ\n3.Selective Repeat ARQ\n4.Exit\nEnter your choice: ";

cin>>ch;

switch(ch) {

case 1:

st.execute();

break;

case 2:

gb.execute();

break;

case 3:

sr.execute();

break;

default:

cout<<"\nInvalid Choice. Please try again!";

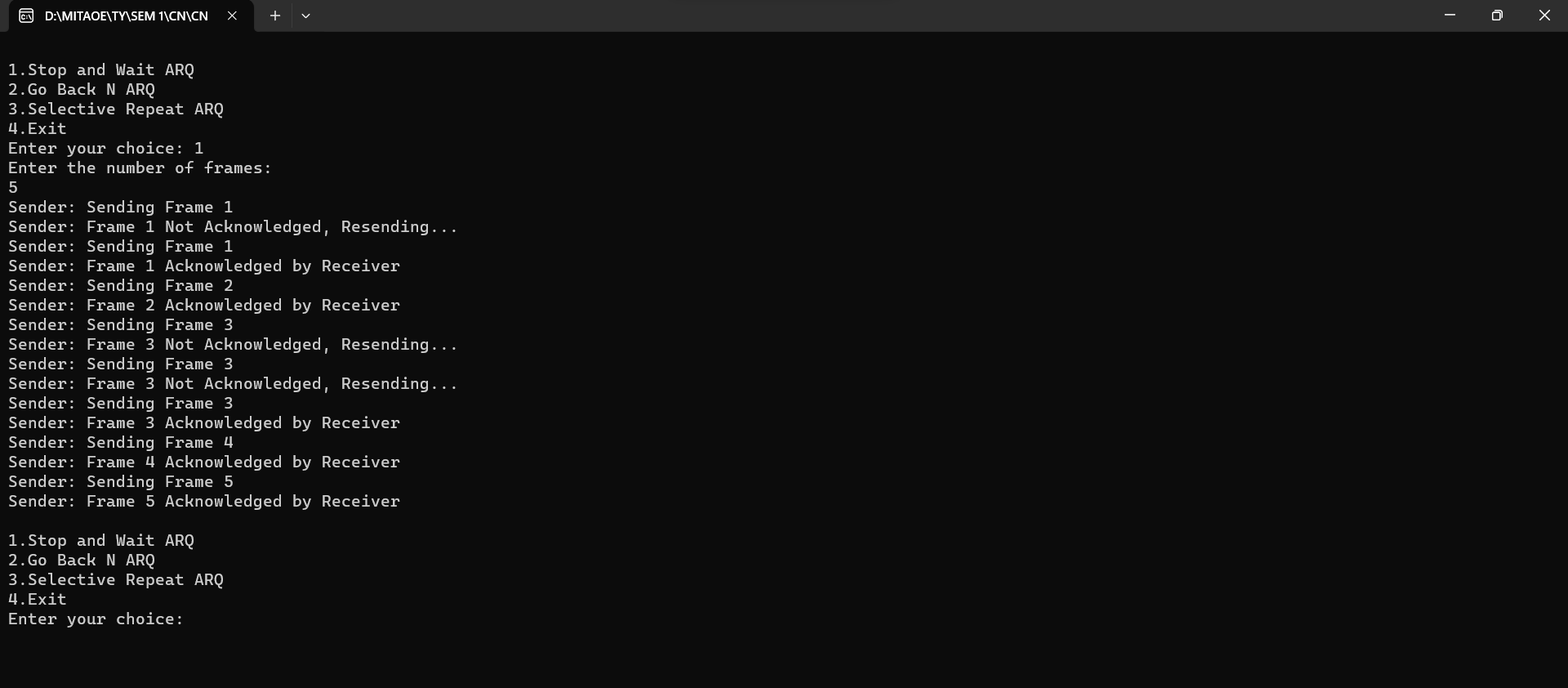
}

} while(ch!=4);

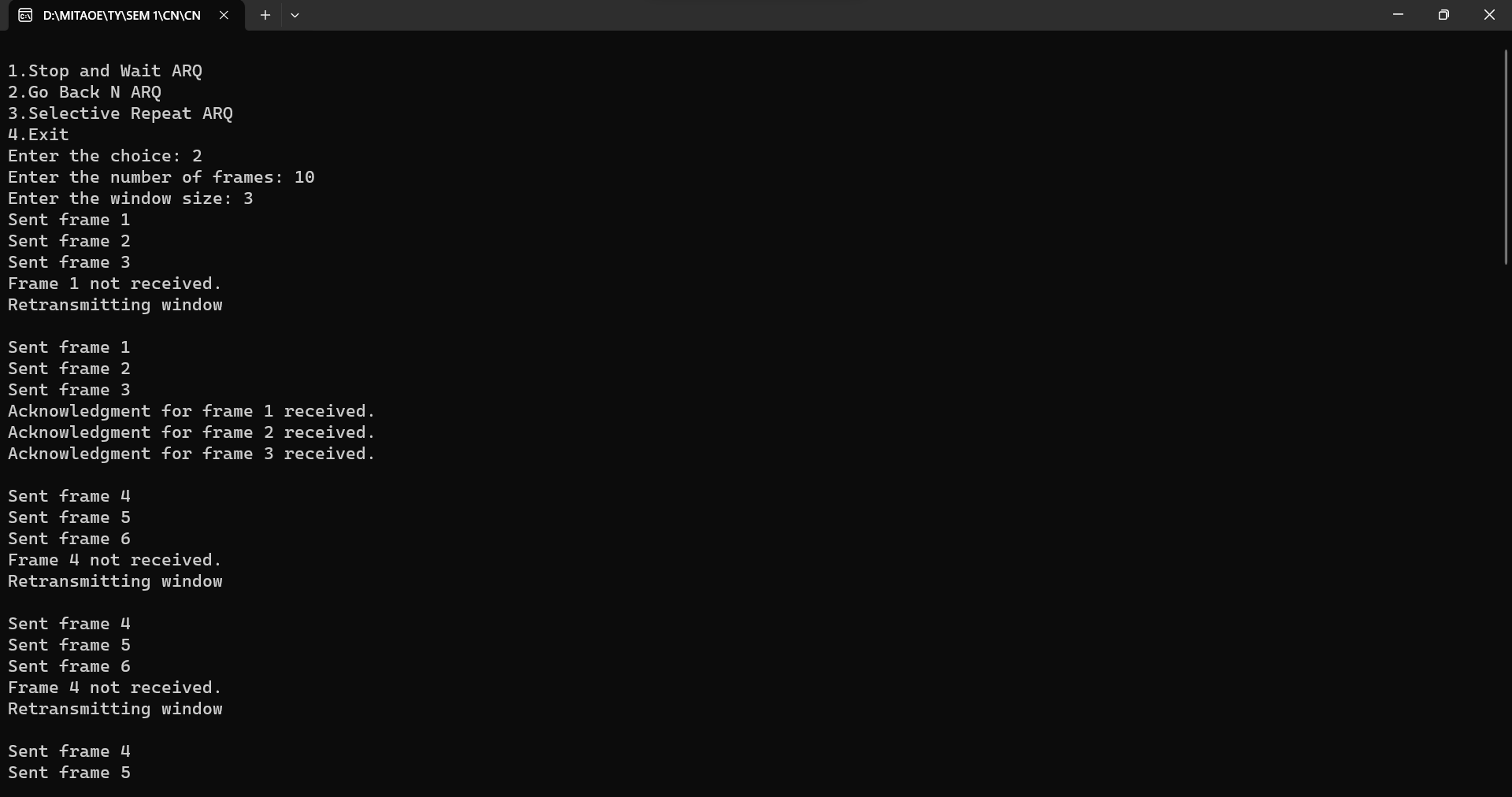
}

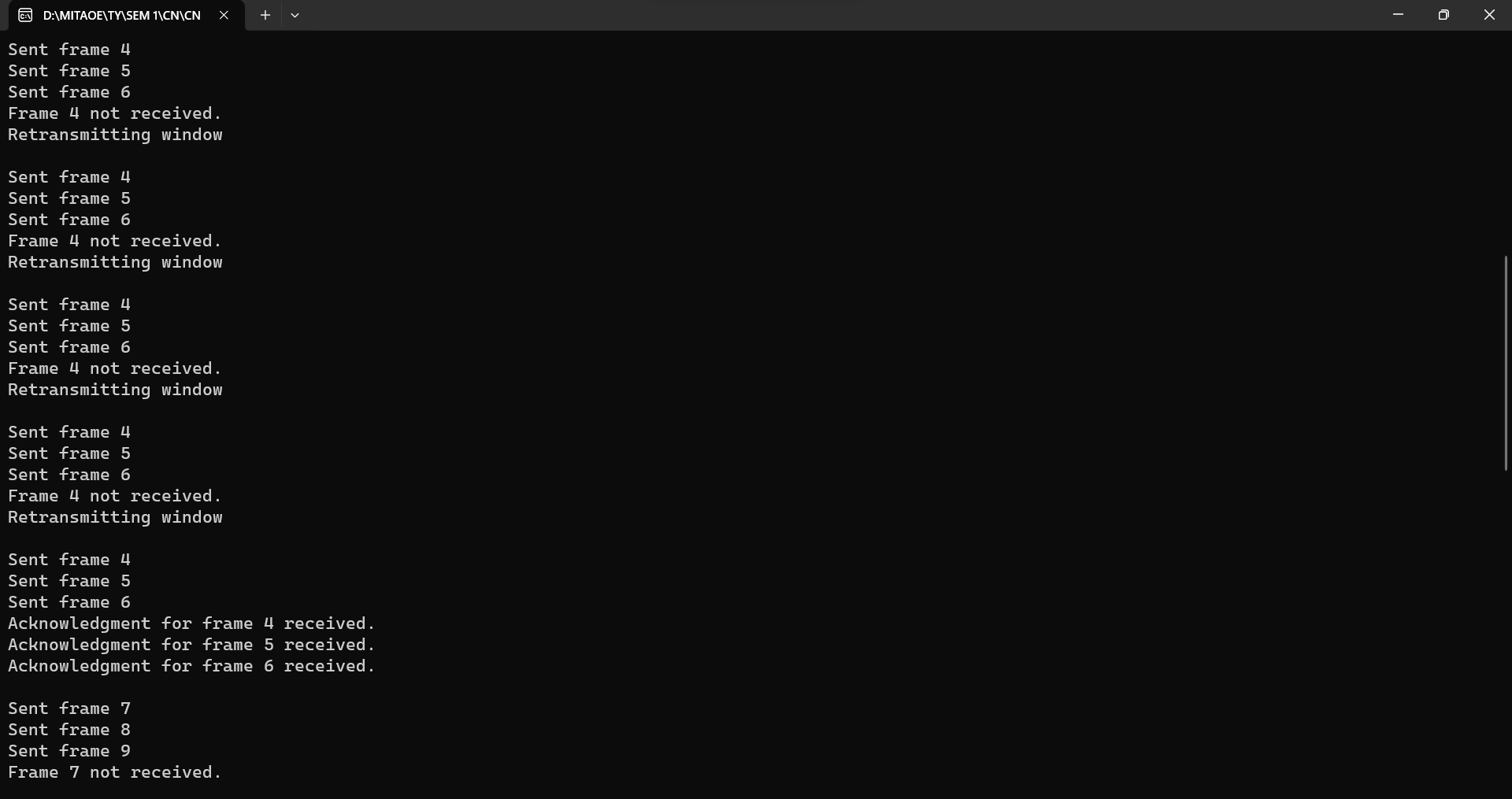
**Output:**

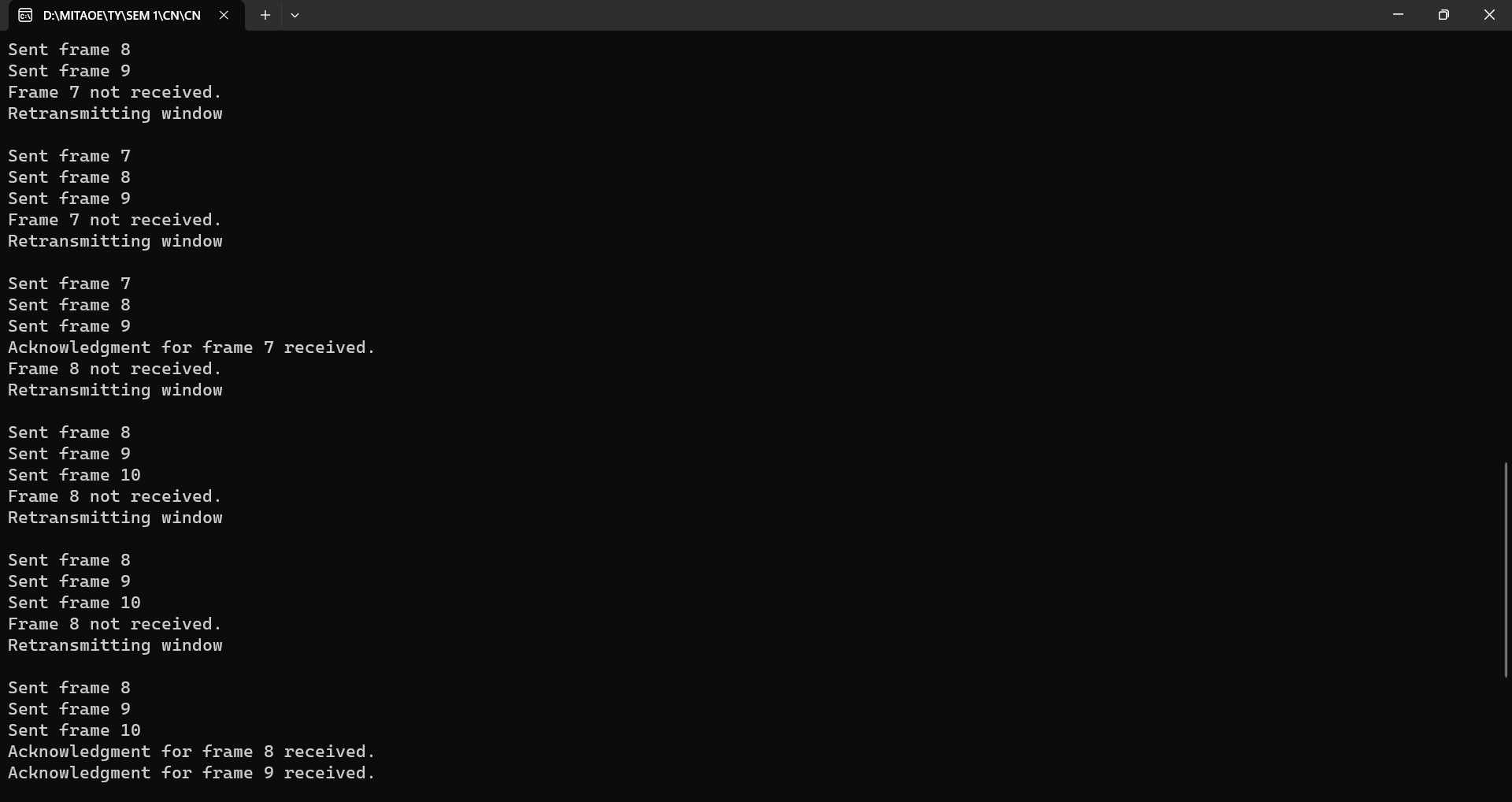
1. **Stop-and-Wait ARQ:**

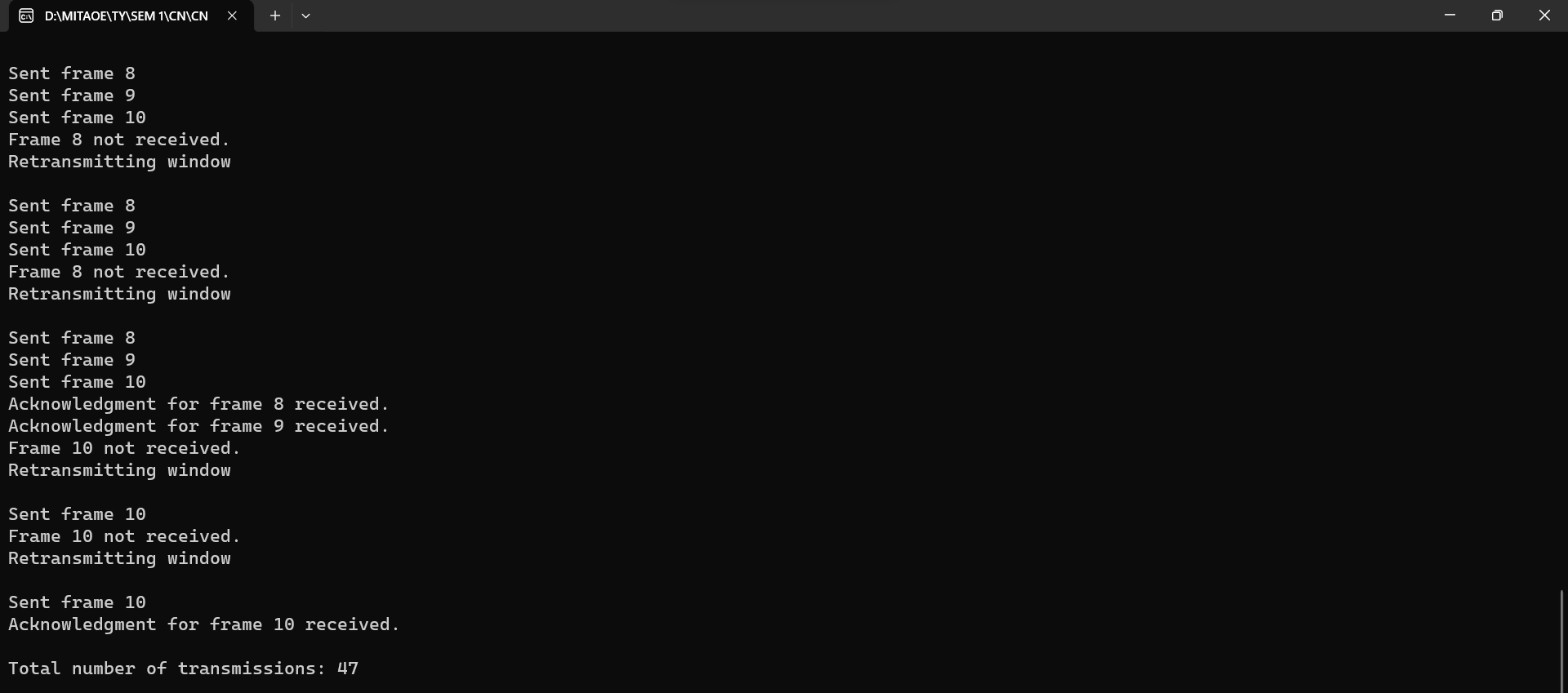
****

1. **Go-Back-N ARQ:**

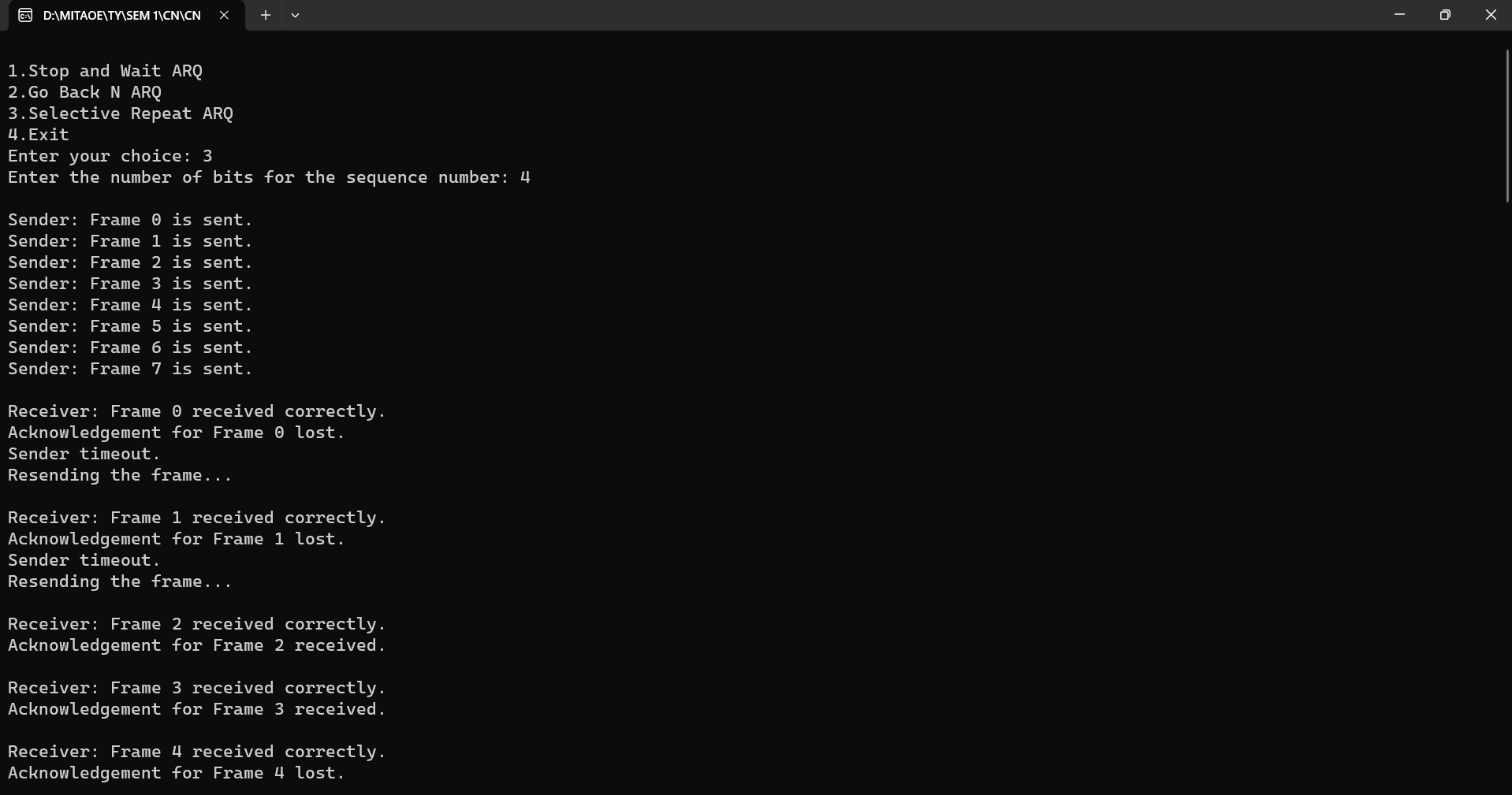
****

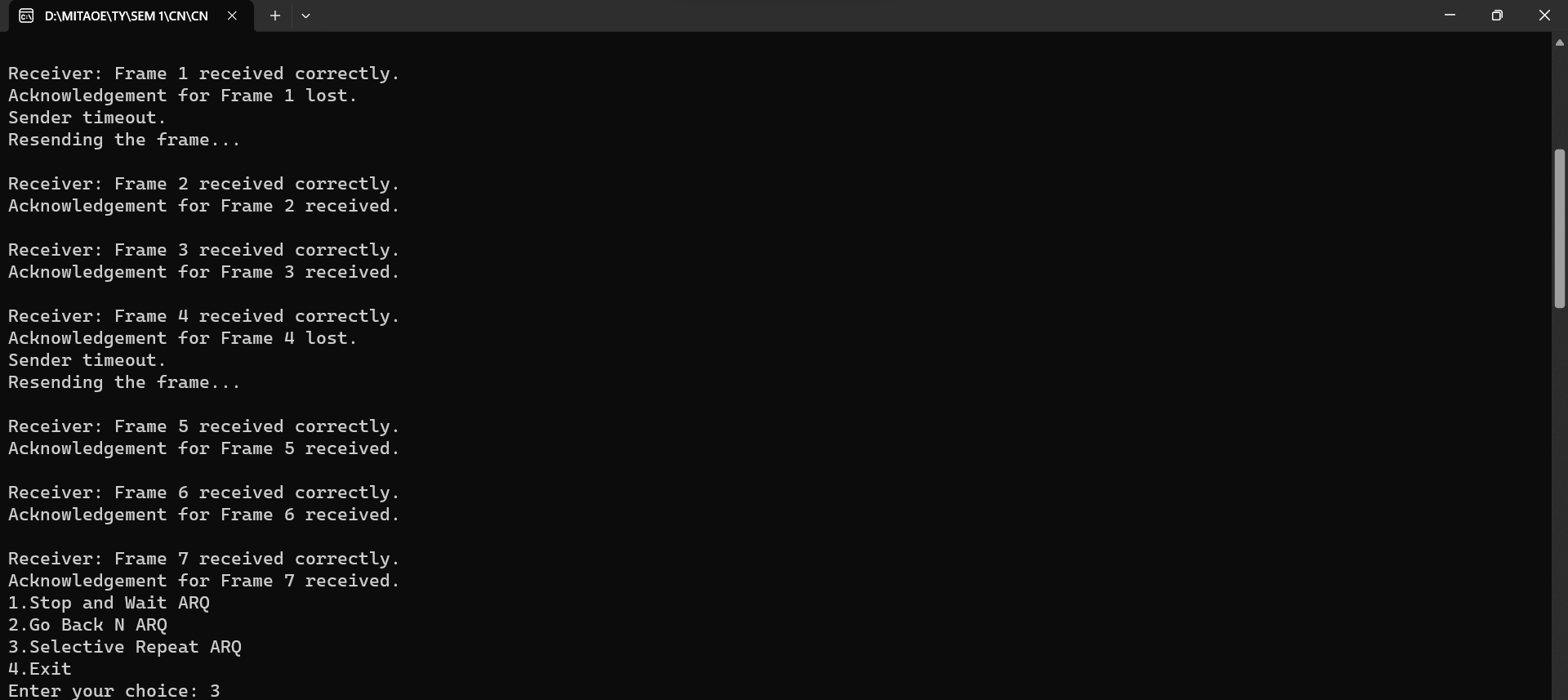
****

****

****

1. **Selective Repeat ARQ:**

****

****

**Conclusion:**

In this assignment, three key Automatic Repeat reQuest (ARQ) protocols have been studied and implemented: Stop-and-Wait, Go-Back-N, and Selective Repeat.

**Stop-and-Wait ARQ:**

Simple, but inefficient due to its one-frame-at-a-time approach.

**Go-Back-N ARQ:**

Allows multiple frames in flight, but inefficiency arises from retransmitting all unacknowledged frames.

**Selective Repeat ARQ:**

Efficient, allowing selective acknowledgement and buffering of out-of-order frames.

Understanding these protocols is vital for designing reliable communication systems, each catering to specific network conditions and trade-offs between efficiency and complexity.