Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: SY	Division: B	Roll No: 272028
Semester: IV	Academic Year: 2022-2023	
Subject Name & Code: Fundamentals of Computer Networks: ADUA22203		
Title of Assignment: Write a program to simulate Go back N and Selective Repeat Modes of Sliding Window Protocol in Peer-to-Peer mode.		

ASSIGNMENT NO. 6

PAGE NO.:
DATE: / /

FCN Assignment no. 6

Name: Siddhesh Dilip Khairnar
Roll no: 272028
PRN no: 2211D398
Batch: B2

* **Problem Definition**: Write a program to simulate Go back N and selective Repeat modes of sliding window Protocol in peer to peer modes.

* **Prerequisite**
1. Data link Layer: Roles, Protocol
2. Java Programming syntax
3. Wireshark Tools.

* **Theory**: –
Data-link layer is responsible for implementation of point-to-point flow and error control mechanism.

- **flow control**: –
1. When a data frame (Layer-2 data) is sent from one host to another over a single medium, it is required that the sender and receiver should work at the same speed.
2. That is, sender sends at a speed on which the receiver can process and accept the data.

Two types of mechanism can be deployed control the flow: –

1) **Stop and wait**: –
This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data

PAGE NO.:
DATE: / /

frame sent is required.

2. Sliding window :-
In this flow control mechanism, both sender and receiver agree on the no. of data-frames after which the acknowledgement should be sent.

- **Error Control**

1. When data frame is transmitted, there is a probability that data frame may be lost in the transit or it is received corrupted.
2. In both cases, the receiver does not receive the current data frames and sender does not know anything about any loss.

- Requirement for error control mechanism

1. error detection
2. Positive Ack
3. Negative Ack
4. Retransmission.

There are 3 types of techniques available which data-link layer may deploy to control the error by Automatic Repeat Request.

1. Stop-and-wait ARQ :-
The following transaction may occur in stop & wait ARQ :-
- The sender maintain a timeout counter.
- When a frames is sent, the sender start the timeout counter.

2. Go-Back-N ARQ :-
Here, both sender and receiver maintain a window. The sending-window size enable the sender to sender multiple frames window receiving the

acknowledgement of previous frame one.

3. Selective - Repeat - ARQ:
Hence, the receiver while keeping track of sequence number, buffer the frames in memory and sends NACk for only frames which is missing or damaged.

Conclusion: Hence, we have studied and understand the concept of Go-back - N and selective Repeat Modes of sliding window protocol and captured packet in wireshark tool.

(c)
Ashwin
21/4/23.

Program and Output:

A) Go-Back-N ARQ:

```
#Go-Back-N ARQ
import random
tf = int(input("Enter the Total number of frames: "))
N = int(input("Enter the Window Size: "))
def transmission(i, N, tf):
    tt = 0
    while i <= tf:
        z = 0
        for k in range(i, min(i + N, tf + 1)):
            print("Sending Frame", k, "...")
            tt += 1
        for k in range(i, min(i + N, tf + 1)):
            f = random.randint(0, 1)
            if not f:
                print("Acknowledgment for Frame", k, "...")
                z += 1
            else:
                print("Timeout!! Frame Number:", k, "Not Received")
                print("Retransmitting window...")
                break
        print()
        i += z
    return tt
i = 1
tt = transmission(i, N, tf)
print("Total number of frames which were sent and resent are ", tt)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS D:\MY FILES\PROGRAM> python -u "d:\MY FILES\PROGRAM\FCN_ASS6.py"
Enter the Total number of frames: 4
Enter the Window Size: 5
Sending Frame 1 ...
Sending Frame 2 ...
Sending Frame 3 ...
Sending Frame 4 ...
Timeout!! Frame Number: 1 Not Received
Retransmitting window...

Sending Frame 1 ...
Sending Frame 2 ...
Sending Frame 3 ...
Sending Frame 4 ...
Timeout!! Frame Number: 1 Not Received
Retransmitting window...

Sending Frame 1 ...
Sending Frame 2 ...
Sending Frame 3 ...
Timeout!! Frame Number: 3 Not Received
Retransmitting window...
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Retransmitting window...

Sending Frame 1 ...
Sending Frame 2 ...
Sending Frame 3 ...
Timeout!! Frame Number: 3 Not Received
Retransmitting window...

Sending Frame 3 ...
Sending Frame 4 ...
Timeout!! Frame Number: 3 Not Received
Retransmitting window...

Sending Frame 3 ...
Sending Frame 4 ...
Acknowledgment for Frame 3 ...
Acknowledgment for Frame 4 ...

Total number of frames which were sent and resent are  18
PS D:\MY FILES\PROGRAM>
```

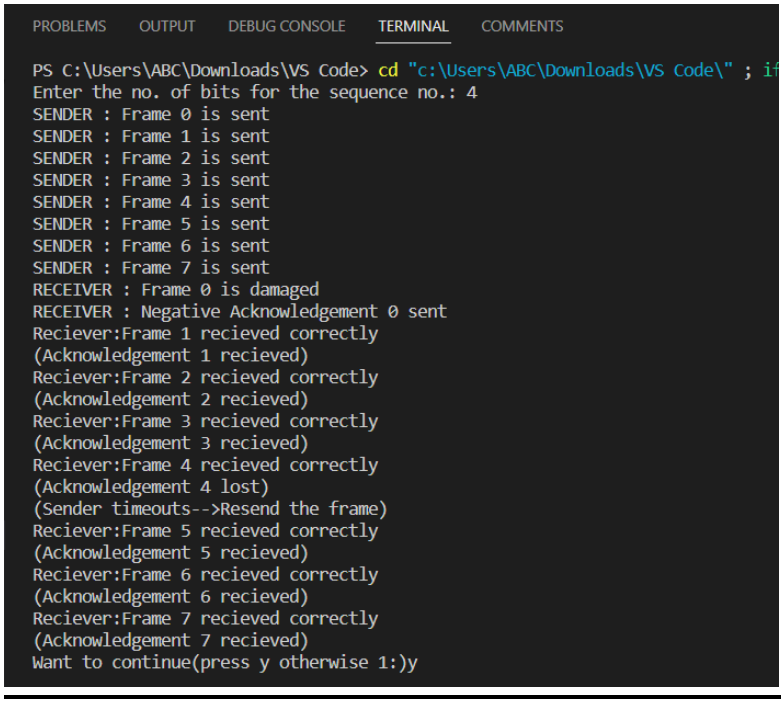
B) Selective Repeat ARQ:

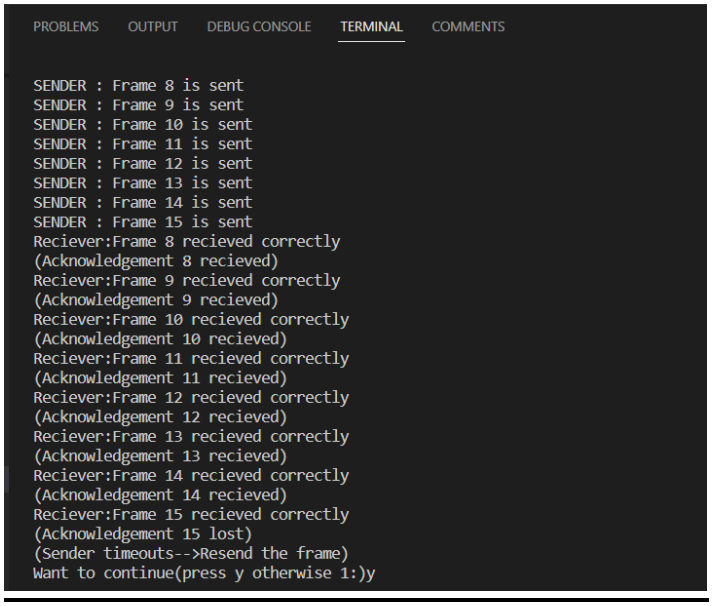
```
// Selective Repeat ARQ
#include <iostream>
using namespace std;
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define TOT_FRAMES 500
#define FRAMES_SEND 10
class sel_repeat
{
private:
    int fr_send_at_instance;
    int arr[TOT_FRAMES];
    int send[FRAMES_SEND];
    int rcvd[FRAMES_SEND];
    char rcvd_ack[FRAMES_SEND];
    int sw;
    int rw; // tells expected frame
public:
    void input();
    void sender(int);
    void receiver(int);
};

void sel_repeat::input()
{
    int n; // no. of bits for the frame
    int m; // no. of frames from n bits int i;
    int i;
```

```
    cout << "Enter the no. of bits for the sequence no.: ";
    cin >> n;
    m = pow(2, n);
    int t = 0;
    fr_send_at_instance = (m / 2);
    for (i = 0; i < TOT_FRAMES; i++)
    {
        arr[i] = t;
        t = (t + 1) % m;
    }
    for (i = 0; i < fr_send_at_instance; i++)
    {
        send[i] = arr[i];
        rcvd[i] = arr[i];
        rcvd_ack[i] = 'n';
    }
    rw = sw = fr_send_at_instance;
    sender(m);
}

void sel_repeat::sender(int m)
{
    for (int i = 0; i < fr_send_at_instance; i++)
    {
        if (rcvd_ack[i] == 'n')
            cout << "SENDER: Frame " << send[i] << " is sent\n";
    }
    receiver(m);
}
void sel_repeat::receiver(int m)
{
    time_t t;
    int f;
    int j;
    int f1;
    int a1;
    char ch;
    srand((unsigned)time(&t));
    for (int i = 0; i < fr_send_at_instance; i++)
    {
        if (rcvd_ack[i] == 'n')
        {
            f = rand() % 10;

            // if f-5 frame is discarded for some reason
            // else frame is correctly recieved

            if (f != 5)
```

```
            {
                for (int j = 0; j < fr_send_at_instance; j++)
                    if (rcvd[j] == send[1])
                    {
                        cout << "Reciever:Frame " << rcvd[j] << " recieved
correctly\n";
                        rcvd[j] = arr[rw];
                        rw = (rw + 1) % m;
                        break;
                    }
                int j;
                if (j == fr_send_at_instance)
                    cout << "Reciever:Duplicate frame " << send[i] << "
discarded\n";
                a1 = rand() % 5;

                // if al--3 then ack is lost
                // else recieved

                if (a1 == 3)
                {
                    cout << "(Acknowledgement " << send[i] << " lost)\n";
                    cout << "(Sender timeouts-->Resend the frame)\n";
                    rcvd_ack[i] = 'n';
                }
                else
                {
                    cout << "(Acknowledgement " << send[i] << " recieved) \n";
                    rcvd_ack[1] = 'p';
                }
            }
            else
            {
                int ld = rand() % 2;

                // if 0 then frame damaged
                // else frame lost

                if (ld == 0)
                {
                    cout << "RECEIVER : Frame " << send[i] << " is damaged\n";
                    cout << "RECEIVER : Negative Acknowledgement " << send[i]
<< " sent \n";
                }
                else
                {
                    cout << "RECEIVER : Frame " << send[i] << " is lost\n";
                    cout << "(SENDER TIMEOUTS-->RESEND THE FRAME)\n";
```

```
                }
                rcvd_ack[i] = 'n';
            }
        }
    }
    for (int j = 0; j < fr_send_at_instance; j++)
    {
        if (rcvd_ack[j] == 'n')
            break;
    }
    int i = 0;
    for (int k = j; k < fr_send_at_instance; k++)
    {
        send[i] = send[k];
        if (rcvd_ack[k] == 'n')
            rcvd_ack[i] = 'n';
        else
            rcvd_ack[i] = 'p';
        i++;
    }
    if (i != fr_send_at_instance)
    {
        for (int k = i; k < fr_send_at_instance; k++)
        {
            send[k] = arr[sw];
            sw = (sw + 1) % m;
            rcvd_ack[k] = 'n';
        }
    }
    cout << "want to continue (press y otherwise 1:)";
    cin >> ch;
    cout << "\n";
    if (ch == 'y')
        sender(m);
    else
        exit(0);
}
int main()
{
    sel_repeat sr;
    sr.input();
}
```





Conclusion: Hence, we have studied and understood the concept of Go back N and Selective Repeat Modes of Sliding Window Protocol in Peer-to-Peer mode.