 **Ramaiah**

**Institute of**

**Technology**

**Certificate**

*This report is submitted for the evaluation of Assignment component for the subject "DATA COMMUNICTION" with the subject code CS44 during the term Jan-May 2020.*

Submitted by

**1MS18CS066 Manish M**

**1MS18CS083 Nikhil RV**

Signature of Faculty

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

www.msrit.edu

# Jan-May 2020

**Problem Statement**

Write a C/C++/Java program to implement Selective Repeat ARQ in noisy channel. The sender should send more than one data frames(within window size) and start timer. If it receives an acknowledgement, then it should stop the timer, move the sending window and send the next set of frame/frames. If the sender receives negative acknowledgement or if the timer times out for a particular frame, it should retransmit those frame/frames that it expects. If receiver receives frame out of order, it should buffer them and send a negative acknowledgement for the lost frame/frames.

**Introduction**

**Selective Repeat Automatic Repeat Request/ Selective Repeat ARQ:**

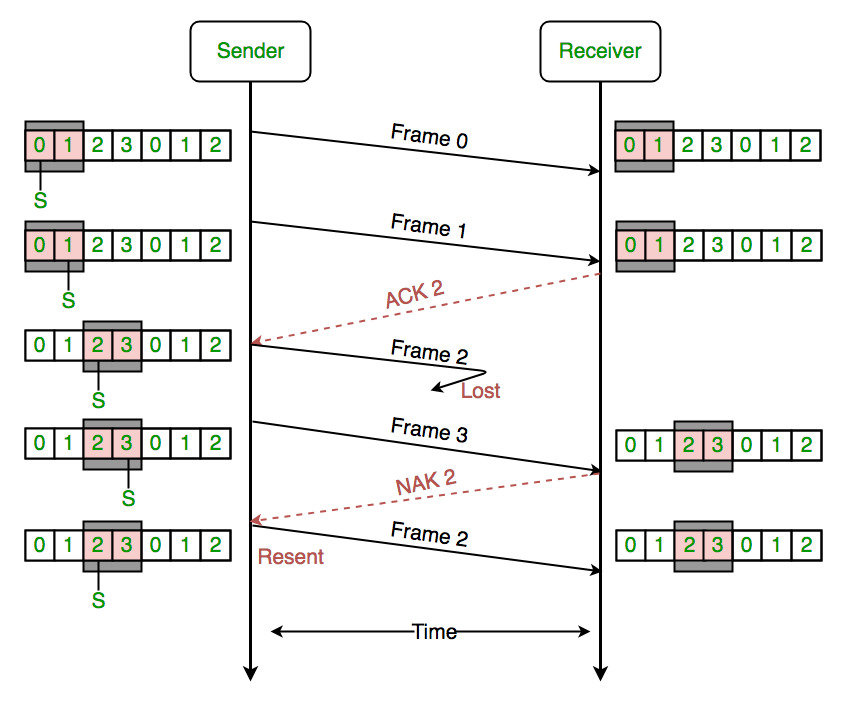
In a noisy link, a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is a mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called ‘Selective Repeat ARQ’. It is more efficient for noisy links but the processing at the receiver is more complex.

**Windows:**

The Selective Repeat protocol uses two windows: a send window and a receive window. The size of the send window is 2^(m-1). For example if m = 4, then the sequence numbers go from 0 to 15 but the size of the window is just 8. The size of the receive window is the same as the size of the send window.

**Sliding Window Protocol:**

Selective Repeat ARQ uses ‘Sliding window protocol’. A sliding window protocol is a feature of packet-based [data transmission](https://en.wikipedia.org/wiki/Data_transmission) [protocols](https://en.wikipedia.org/wiki/Protocol_(computing)). Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the [data link layer](https://en.wikipedia.org/wiki/Data_link_layer) ([OSI layer 2](https://en.wikipedia.org/wiki/OSI_model#Layer_2:_Data_Link_Layer)) as well as in the [Transmission Control Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) (TCP). They are also used to improve efficiency when the channel may include high [latency](https://en.wikipedia.org/wiki/Network_delay). When the receiver receives the packets in order, it sends an ‘ACK’(acknowledgement) signal but if it does not receive the packets in order, it sends a ‘NAK’(negative acknowledgement).



**Example for ‘Selective Repeat ARQ’ for a window size of 2**

**Code**

/\* Program in C++ to implement Selective Repeat ARQ in noisy channel

  Done by:

  Manish M - 1MS18CS066

  Nikhil RV - 1MS18CS083

\*/

#include<iostream>

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<time.h>

#include<Windows.h>

using namespace::std;

void Send\_frames(int,int);

int Receive();

void Resend\_lost\_frame(int);

void Resend\_frames\_due\_to\_timeout(int,int);

int main()

{

    srand(time(0));

    int i,window\_size=8,number\_of\_frames,remaining\_number\_of\_frames;

    bool timeout = false;

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

    cin>>number\_of\_frames;

    cout<<"Window size is : "<<window\_size;

    cout<<"\n-----------------------------------\n\n";

    remaining\_number\_of\_frames = number\_of\_frames;

    int start\_frame = 0;

    while (remaining\_number\_of\_frames!=0)

    {

        timeout = false;

        if(remaining\_number\_of\_frames < window\_size) {

            window\_size = remaining\_number\_of\_frames;

        }

        cout<<"\nWindow is: "<<start\_frame<<" to "<<start\_frame+window\_size;

        Send\_frames(start\_frame,window\_size);

        receive:

        clock\_t time\_start = clock();

        int x = Receive();     //Receiver should receive the frames of the window

        clock\_t time\_end = clock();

        if((time\_end - time\_start)/CLOCKS\_PER\_SEC > 5)

        {

            if(!timeout) {

                cout<<"\n\tTimeout...\n"<<"\tResending window again";   //Timeout

                Resend\_frames\_due\_to\_timeout(start\_frame,window\_size);

            }

            timeout = true;

            goto receive;

        }

        else if(x == 0) {

            int frame\_to\_be\_resent = rand()%(window\_size) + start\_frame;

            cout<<"\n\tNAK "<<frame\_to\_be\_resent;       //Negative Acknowledgement

            cout<<"\nTime taken = "<<(double)(time\_end - time\_start)/CLOCKS\_PER\_SEC<<" secs";

            Resend\_lost\_frame(frame\_to\_be\_resent);

            timeout = true;

            goto receive;

        }

        else {

            cout<<"\n\tACK";    //Acknowledged

            cout<<"\nTime taken = "<<(double)(time\_end - time\_start)/CLOCKS\_PER\_SEC<<" secs";

        }

        start\_frame += 8;

        remaining\_number\_of\_frames -= window\_size;

    }

    return 0;

}

//Function to send the frames

void Send\_frames(int start\_frame,int window\_size) {

    for(int i=start\_frame;i<start\_frame+window\_size;i++) {

        cout<<"\nSending frame " <<i;

        Sleep(500);

    }

}

// Function that simulates the receiver

int Receive() {

    int n = rand()%6+1;     //Generate a random number between 1 and 6

    Sleep(n\*1000);          //Sleep for 1,2,3,4,5 or 6 seconds

    return rand()%2;    //0 or 1 randomly where 1=>ACK and 0=>NAK

}

//Function to resend the frames due to timeout

void Resend\_frames\_due\_to\_timeout(int start\_frame, int window\_size) {

    cout<<"\nResending the frames "<<start\_frame<<" to "<<start\_frame+window\_size;

    for(int i=start\_frame;i<start\_frame+window\_size;i++){

        cout<<"\nSending frame " <<i;

        Sleep(500);

    }

}

//Function to resend the frame lost (NAK)

void Resend\_lost\_frame(int frame\_number) {

    cout<<"\nResending the frame "<<frame\_number;

    Sleep(1000);

}

**Results Snapshots**

1. **Number of frames is 5**



1. **Number of frames is 11**



1. **Number of frames is 16**

