SELECTIVE REPEAT ARQ IMPLEMENTATION

Selective repeat protocol, also called Selective Repeat ARQ (Automatic Repeat request), is a data link layer protocol that uses sliding window method for reliable delivery of data frames. Here, only the erroneous or lost frames are retransmitted, while the good frames are received and buffered.

It uses two windows of equal size: a sending window that stores the frames to be sent and a receiving window that stores the frames receive by the receiver. The size is half the maximum sequence number of the frame. For example, if the sequence number is from 0 – 15, the window size will be 8.

Working Principle

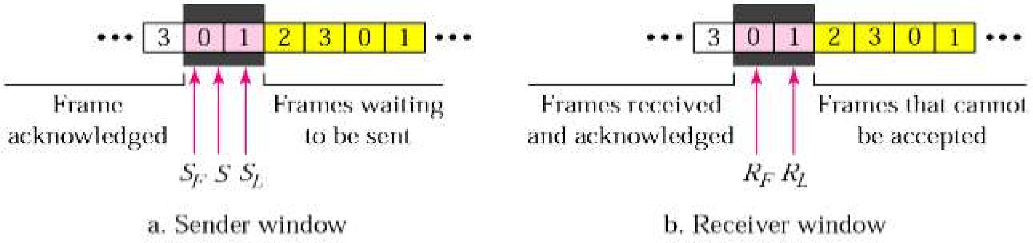
Selective Repeat protocol provides for sending multiple frames depending upon the availability of frames in the sending window, even if it does not receive acknowledgement for any frame in the interim. The maximum number of frames that can be sent depends upon the size of the sending window.

The receiver records the sequence number of the earliest incorrect or un-received frame. It then fills the receiving window with the subsequent frames that it has received. It sends the sequence number of the missing frame along with every acknowledgement frame.

The sender continues to send frames that are in its sending window. Once, it has sent all the frames in the window, it retransmits the frame whose sequence number is given by the acknowledgements. It then continues sending the other frames.

The control variables in Selective Repeat ARQ are same as in Go-Back-N ARQ:

S_F, S_L and S. But the sender sliding window size changed into 2^{m-1}.Receiver sliding window has 2 control variables, R_F and R_L.

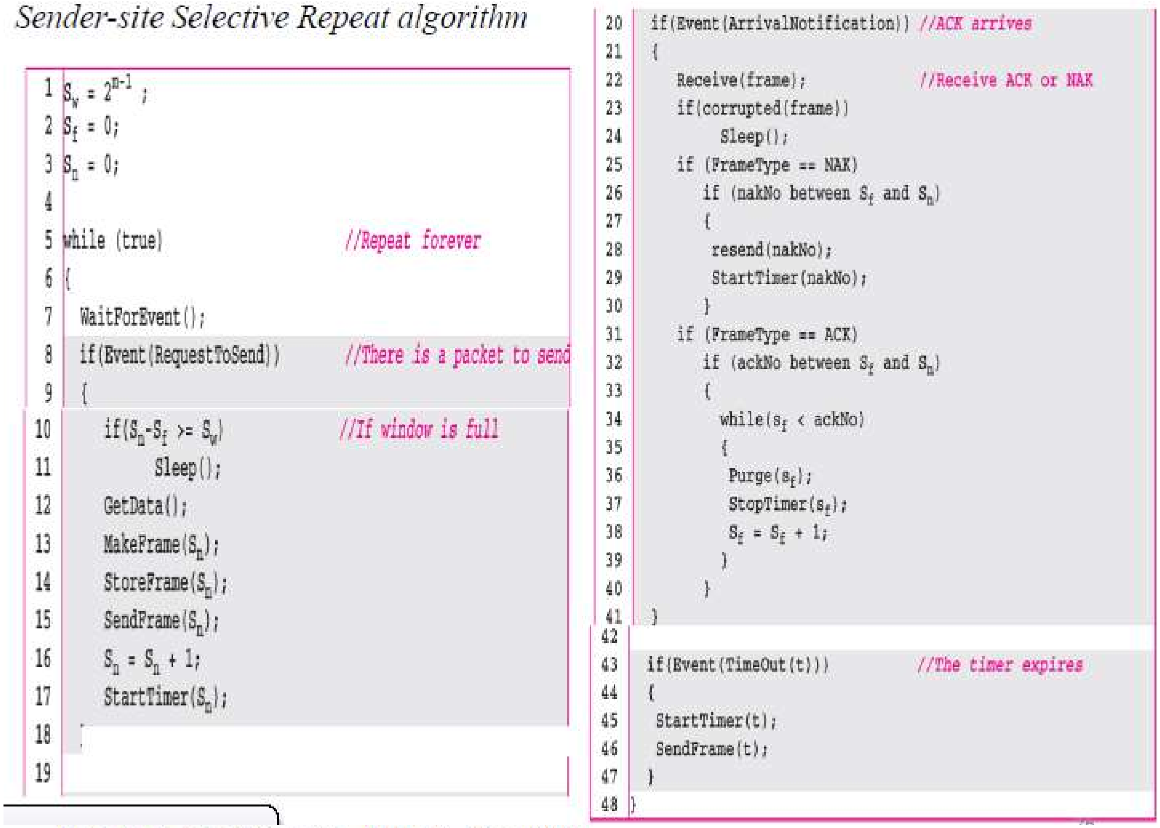

3
0
1
2
3
0
1
Frame acknowledged
Frames waiting to be sent
S_F S S_L
3
0
1
2
3
0
1
Frames received and acknowledged
Frames that cannot be accepted
R_F R_L

a. Sender window

b. Receiver window

Selective Repeat ARQ receiver slide window [1]

Sender-site Selective Repeat algorithm



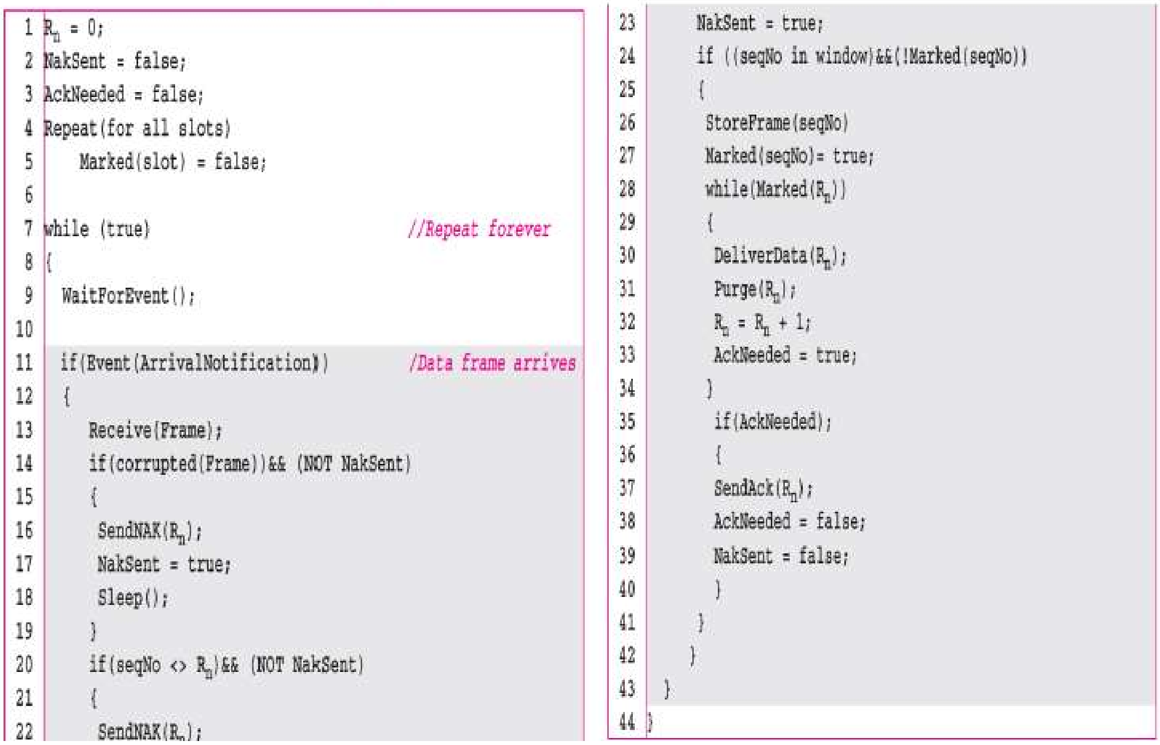
```
$S_w = 2^{m-1}$ ;
$S_f$ = 0;
$S_n$ = 0;

while (true)                              //Repeat forever
{
  WaitForEvent();
  if(Event(RequestToSend))                //There is a packet to send
  {
     if($S_n$-$S_f$ >= $S_w$)                  //If window is full
           Sleep();
     GetData();
     MakeFrame($S_n$);
     StoreFrame($S_n$);
     SendFrame($S_n$);
     $S_n$ = $S_n$ + 1;
     StartTimer($S_n$);
  }

  if(Event(ArrivalNotification)) //ACK arrives
  {
     Receive(frame);              //Receive ACK or NAK
     if(corrupted(frame))
          Sleep();
     if (FrameType == NAK)
        if (nakNo between $S_f$ and $S_n$)
        {
         resend(nakNo);
         StartTimer(nakNo);
        }
     if (FrameType == ACK)
        if (ackNo between $S_f$ and $S_n$)
        {
          while($s_f$ < ackNo)
          {
           Purge($s_f$);
           StopTimer($s_f$);
           $S_f$ = $S_f$ + 1;
          }
        }
  }

  if(Event(TimeOut(t)))          //The timer expires
  {
   StartTimer(t);
   SendFrame(t);
  }
}
```

Receiver-site Selective Repeat algorithm



```
$R_n$ = 0;
NakSent = false;
AckNeeded = false;
Repeat(for all slots)
    Marked(slot) = false;

while (true)                                    //Repeat forever
{
  WaitForEvent();

  if(Event(ArrivalNotification))                /Data frame arrives
  {
     Receive(Frame);
     if(corrupted(Frame))&& (NOT NakSent)
     {
      SendNAK($R_n$);
      NakSent = true;
      Sleep();
     }
     if(seqNo <> $R_n$)&& (NOT NakSent)
     {
      SendNAK($R_n$);
      NakSent = true;
      if ((seqNo in window)&&(!Marked(seqNo))
      {
       StoreFrame(seqNo)
       Marked(seqNo)= true;
       while(Marked($R_n$))
       {
        DeliverData($R_n$);
        Purge($R_n$);
        $R_n$ = $R_n$ + 1;
        AckNeeded = true;
       }
        if(AckNeeded);
        {
        SendAck($R_n$);
        AckNeeded = false;
        NakSent = false;
        }
      }
     }
  }
}
```

Algorithm 10 Selective Repeat ARQ - Sender

```
$S_w \leftarrow 2^{m-1}$
$S_f = S_n = 0$
while $True$ do
    WaitForEvent()
    if Event(RequestToSend) then
        if $S_n - S_f \geq S_w$ then
            Sleep()
        end if
        GetData()
        MakeFrame($S_n$)
        StoreFrame($S_n$)
        SendFrame($S_n$)
        $S_n \leftarrow (S_n + 1)\%S_w$
        StartTimer($S_n$)
    end if
    if Event(ArrivalNotification) then
        Receive($frame$)
        if Corrupted($frame$) then
            Sleep()
        end if
        if FrameType == NAK then
            if $nakNo$ in $(S_f, S_n]$ then
                Resend($nakNo$)
                StartTimer($nakNo$)
            end if
        else if FrameType == ACK then
            if $ackNo$ in $(S_f, S_n]$ then
                while $S_f < ackNo$ do
                    Purge($S_f$)
                    StopTimer($S_f$)
                    $S_f \leftarrow (S_f + 1)\%2^m$
                end while
            end if
        end if
    end if
    if Event(Timeout $T_i$) then
        StartTimer($T_i$)
        SendFrame($T_i$)
    end if
end while
```

Algorithm 11 Selective Repeat - Receiver

```
$R_n \leftarrow 0$
$nakSent \leftarrow False$
$ackNeeded \leftarrow False$
for all $slot$ in $slots$ do
    Marked($slot$) $\leftarrow$ $False$
end for
while $True$ do
    WaitForEvent()
    if Event(ArrivalNotification) then
        Receive($frame$)
        if Corrupted($frame$) and not $nakSent$ then
            SendNAK($R_n$)
            $nakSent \leftarrow True$
            Sleep()
        end if
        if $seqNo$ != $R_n$ and not $nakSent$ then
            SendNAK($R_n$)
            $nakSent \leftarrow True$
            if $seqno$ in $window$ and not $Marked(seqno)$ then
                StoreFrame($seqno$)
                $Marked(seqno) \leftarrow True$
                while $Marked(R_n)$ do
                    DeliverData($R_n$)
                    Purge($R_n$)
                    $R_n \leftarrow (R_n + 1)$ % $2^m$
                    $ackNeeded \leftarrow True$
                end while
                if $ackNeeded$ then
                    SendACK($R_n$)
                    $ackNeeded \leftarrow False$
                    $nakSent \leftarrow False$
                end if
            end if
        end if
    end if
end while
```

sr.c
The overall program counts acknowledgments and retransmissions

```
#include<stdio.h>

#include<stdlib.h>

int input(int a[] , int frame_size)
{
   printf("\n\n Input \n\n");
   for(int i = 1 ; i <= frame_size ; i++)
   {
        printf(" Enter Value For Frame[%d] : " , i);
        scanf("%d",&a[i]);
        printf("\n");
   }
   printf("\n\n");
   return 1;
}
int display(int a[] , int frame_size)
{
   printf("\n\n Display \n\n");
   for(int i = 1 ; i <= frame_size ; i++)
   {
        printf(" Frame[%d] : %d " , i , a[i]);
//List all frames with sequence numbers
        printf("\n");
   }
   printf("\n\n");
   return 1;
}
int selective_repeat(int frames[] , int window_size , int frame_size)
{
        int nt =0;

        int k = 0;
```

```
     int left[10] = {-1};

          int i ;

          for(i = 1 ; i <= frame_size ; i++)
    {
          int flag = rand() % 2;
//Set frame 2 to be dropped during transmission

          if(flag) //success
          {
               printf(" Frame[%d] with value %d Acknowledged !!!
\n\n", i , frames[i]);
               nt++; //Increment nt when frame is acknowledged
          }
          else
         //failure
          {
printf(" Frame[%d] with value %d Not Acknowledged !!! \n\n", i ,
frames[i]);

               left[k++] = frames[i];
//Note frame number not acknowledged and store that frame number
in left 1-D array

               nt++; //count retransmission
          }
          if(i % window_size == 0)
          { //Loop for retransmitting lost frame
               for(int x = 0 ; x < k ; x++)
               {
                    printf(" Frame[%d] with value %d Retransmitted
\n\n", x , left[x]);

                    nt++; //increment nt for retransmitted frame
```

```
                                printf(" Frame[%d] with value %d Acknowledged
on Second Attempt \n\n", x , left[x]);

                        }
                        k = 0;
                }
        }
         for(i = 0 ; i < k ; i++)
         {
             printf(" Frame[%d] with value %d Retransmitted \n\n", i ,
left[i]);
//print retransmitted frame details
                nt++;

                printf(" Frame[%d] with value %d Acknowledged on
Second Attempt \n\n", i , left[i]);
         }
         printf(" Total Transmissions :  %d \n\n", nt);
                return 0;
}
int main()
{
         int frames[50];

         int window_size;

         int frame_size;

         printf("\n\n Selective Repeat \n\n");

                printf(" Enter Window Size : ");

                scanf("%d",&window_size);

                printf(" Enter Number Of Frames To Be Transmitted : ");

                scanf("%d",&frame_size);
                input(frames , frame_size);

                display(frames , frame_size);

                selective_repeat(frames , window_size , frame_size);

                return 0;

}
```

```
net@inlab:~$ gedit sr.c
net@inlab:~$ gcc sr.c
net@inlab:~$ ./a.out


 Selective Repeat

 Enter Window Size : 5
 Enter Number Of Frames To Be Transmitted : 10


 Input

 Enter Value For Frame[1] : 1

 Enter Value For Frame[2] : 2

 Enter Value For Frame[3] : 3

 Enter Value For Frame[4] : 4

 Enter Value For Frame[5] : 5

 Enter Value For Frame[6] : 6

 Enter Value For Frame[7] : 7
 Enter Value For Frame[8] : 8

 Enter Value For Frame[9] : 9

 Enter Value For Frame[10] : 10




 Display

 Frame[1] : 1
 Frame[2] : 2
 Frame[3] : 3
 Frame[4] : 4
 Frame[5] : 5
 Frame[6] : 6
 Frame[7] : 7
 Frame[8] : 8
 Frame[9] : 9
 Frame[10] : 10


 Frame[1] with value 1 Acknowledged !!!

 Frame[2] with value 2 Not Acknowledged !!!

 Frame[3] with value 3 Acknowledged !!!

 Frame[4] with value 4 Acknowledged !!!

 Frame[5] with value 5 Acknowledged !!!

 Frame[0] with value 2 Retransmitted
```

```
Frame[0] with value 2 Acknowledged on Second Attempt

Frame[6] with value 6 Acknowledged !!!

Frame[7] with value 7 Not Acknowledged !!!

Frame[8] with value 8 Not Acknowledged !!!

Frame[9] with value 9 Acknowledged !!!

Frame[10] with value 10 Acknowledged !!!

Frame[0] with value 7 Retransmitted

Frame[0] with value 7 Acknowledged on Second Attempt

Frame[1] with value 8 Retransmitted

Frame[1] with value 8 Acknowledged on Second Attempt

Total Transmissions :  13
```

Both Go-Back-N and Selective Repeat protocols are sliding window protocols.

Following are the important differences between Go-Back-N and Selective Repeat Protocols.

Sr. No.	Key	Go-Back-N	Selective Repeat
1	Definition	In Go-Back-N if a sent frame is found suspected or damaged then all the frames are retransmitted till the last packet.	In Selective Repeat, only the suspected or damaged frames are retransmitted.
2	Sender Window Size	Sender Window is of size N.	Sender Window size is same as N.

Sr. No.	Key	Go-Back-N	Selective Repeat
3	Receiver Window Size	Receiver Window Size is 1.	Receiver Window Size is N.
4	Complexity	Go-Back-N is easier to implement.	In Selective Repeat, receiver window needs to sort the frames.
5	Efficiency	Efficiency of Go-Back-N = N / (1 + 2a).	Efficiency of Selective Repeat = N / (1 + 2a).
6	Acknowledgement	Acknowledgement type is cumulative.	Acknowledgement type is individual.

Stop and Wait protocol

Stop and Wait protocol is a protocol for flow control mechanism. In this protocol, sender sends one frame at a time and waits for acknowledgement from the receiver. Once acknowledged, sender sends another frame to the receiver. If acknowledgment is not received then frame/packet is retransmitted after timeout.

GoBackN protocol

GoBackN is also a protocol for flow control mechanism. In this protocol, sender sends n frames at a time and wait for cumulative acknowledgment. If acknowledgment is not received then entire frames are retransmitted again.

Selective Repeat protocol

Selective Repeat is also a protocol for flow control mechanism. In this protocol, sender sends n frames at a time and wait for acknowledgment of packets received in particular order. If acknowledgment is not received then lost packets are transmitted again which is based on receiver acknowledgment. Receiver maintains a buffer of lost packets.First, the size of the sender window is much smaller; it is 2^(m–1) . Second, the receiver window has the same size as the sender window, i.e 2^(m-1).

Following are some of the important differences between Stop and Wait protocol and Sliding Window protocol.

Sr. No.	Key	Stop and Wait protocol	GoBackN protocol	Selective Repeat protocol
1	Sender window size	In Stop and Wait protocol, Sender window size is 1.	In GoBackN protocol, Sender window size is N.	In Selective Repeat protocol, Sender window size is N.
2	Receiver Window size	In Stop and Wait protocol, Receiver window size is 1.	In GoBackN protocol, Receiver window size is 1.	In Selective Repeat protocol, Receiver window size is N.
3	Minimum Sequence Number	In Stop and Wait protocol, Minimum Sequence Number is 2.	In GoBackN protocol, Minimum Sequence Number is N+1 where N is number of packets sent.	In Selective Repeat protocol, Minimum Sequence Number is 2N where N is number of packets sent.
4	Efficiency	In Stop and Wait protocol, Efficiency formular is 1/(1+2*a) where a is ratio of propagation delay vs transmission delay.	In GoBackN protocol, Efficiency formular is N/(1+2*a) where a is ratio of propagation delay vs transmission delay and N is number of packets sent.	In Selective Repeat protocol, Efficiency formular is N/(1+2*a) where a is ratio of propagation delay vs transmission delay and N is number of packets sent.
5	Acknowledgement Type	In Stop and Wait protocol, Acknowledgement type is individual.	In GoBackN protocol, Acknowledgement type is cumulative.	In Selective Repeat protocol, Acknowledgement type is individual.
6	Supported Order	In Stop and Wait protocol, no specific order is needed at receiver end.	In GoBackN protocol, in-order delivery only are accepted at receiver end.	In Selective Repeat protocol, out-of-order deliveries also can be accepted at receiver end.

Sr. No.	Key	Stop and Wait protocol	GoBackN protocol	Selective Repeat protocol
7	Retransmissions	In Stop and Wait protocol, in case of packet drop,number of retransmission is 1.	In GoBackN protocol, in case of packet drop,numbers of retransmissions are N.	In Selective Repeat protocol, in case of packet drop,number of retransmission is 1.

Selective-Repeat Client/Server Implementation in C

CLIENT SIDE PROGRAM – src.c

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
//structure definition for accepting the packets.
struct frame
{
int packet[40];
};
//structure definition for constructing the acknowledgement frame
struct ack
{
int acknowledge[40];
};
```

```
int main()

{

int clientsocket;

struct sockaddr_in serveraddr;

socklen_t len;

struct hostent *server;

struct frame f1;

int windowsize,totalpackets,totalframes,i=0,j=0,framesreceived=0,k,l,buffer;

struct ack acknowledgement;

char req[50];

clientsocket=socket(AF_INET,SOCK_DGRAM,0);

bzero((char*)&serveraddr,sizeof(serveraddr)); //pad the socket address variable

serveraddr.sin_family=AF_INET;

serveraddr.sin_port=htons(5018);

server=gethostbyname("127.0.0.1"); //initialize sockaddr_in structure members

bcopy((char*)server->h_addr,(char*)&serveraddr.sin_addr.s_addr, sizeof(server->h_addr));

//bcopy() copies n bytes from source to destination

//establishing the connection.

printf("sending request to the server\n");

sendto(clientsocket,"HI IAM CLIENT",sizeof("HI IAM CLIENT"),0,(struct
sockaddr*)&serveraddr,sizeof(serveraddr)); //Send the message Hi I am client to server address

printf("\nWaiting for reply\n");

recvfrom(clientsocket,req,sizeof(req),0,(struct sockaddr*)&serveraddr,&len);

printf("\n The server has to send :\t%s\n",req);

//Server has to send REQUEST FOR WINDOWSIZE

//accepting window size from the user.
```

```
printf("\nenter the window size\n");
scanf("%d",&windowsize);
//sending the window size.
printf("\n sending window size\n");
sendto(clientsocket,(char*)&windowsize,sizeof(windowsize),0,(struct sockaddr*)&serveraddr,sizeof(serveraddr)); //Sending window size from client socket to server address
//collecting details from server.
printf("\n waiting for the server response\n");
recvfrom(clientsocket,(char*)&totalpackets,sizeof(totalpackets),0,(struct sockaddr*)&serveraddr,&len);
//Receive the total number of frames from server through clientsocket.
printf("\nTotal packets are :\t%d\n",totalpackets);
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//Send received packet count
//receive total number of frames
recvfrom(clientsocket,(char*)&totalframes,sizeof(totalframes),0,(struct sockaddr*)&serveraddr,&len);
printf("\n total number of frames or windows are:\t%d\n",totalframes);
//send received frame count message to server
sendto(clientsocket,"RECEIVED",sizeof("RECEIVED"),0,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
//starting the process.
printf("\n starting the process of receiving\n");
while(i<totalpackets)
{
//initializing the receiver buffer.
printf("\nInitializing the received buffer\n");
printf("\nThe expected frame is %d with packets:",framesreceived);
j=0;
```

```
buffer=i;
while(j<windowsize && i<totalpackets)
{
printf("%d",i);
i++;
j++;
}
printf("\nwaiting for the frame\n");
//accepting the frame from serveraddr via clientsocket.
recvfrom(clientsocket,(char*)&f1,sizeof(f1),0,(struct sockaddr*)&serveraddr,&len);
printf("\n received frame %d\n\n enter -1 to send negative acknowledgement for the following packets \n",framesreceived);
//constructing the acknowledgement frame.
j=0;
l=buffer;
k=0;
while(j<windowsize && l<totalpackets)
{
printf("\npacket:%d\n",f1.packet[j]);
//accepting acknowledgement from the user.
scanf("%d",&acknowledgement.acknowledge[j]);
if(acknowledgement.acknowledge[j]==-1)
{
if(k==0)
{
i=f1.packet[j]; //noting the frame with negative acknowledgement
k=1;
}
}
j++;
l++;
}
framesreceived++;
//sending acknowledgement to the server via clientsocket.
sendto(clientsocket,(char*)&acknowledgement,sizeof(acknowledgement),0,(struct
sockaddr*)&serveraddr,sizeof(serveraddr));
}
printf("\nall frames received successfully\n closing connection with the server\n");
close(clientsocket);
}
```

SERVER SIDE PROGRAM – srs.c

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<unistd.h>
#include<arpa/inet.h>
//structure definition for designing the packet.
struct frame
{
int packet[40];
```

```
};
//structure definition for accepting the acknowledgement.
struct ack
{
int acknowledge[40];
};
int main()
{
int serversocket;
struct sockaddr_in serveraddr,clientaddr;
socklen_t len;
struct frame f1;
int windowsize,totalpackets,totalframes,i=0,j=0,framesend=0,k,l,buffer;
struct ack acknowledgement;
char req[50];
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serversocket=socket(AF_INET,SOCK_DGRAM,0);
bzero((char*)&serveraddr,sizeof(serveraddr));
serveraddr.sin_family=AF_INET;
serveraddr.sin_port=htons(5018);
serveraddr.sin_addr.s_addr=INADDR_ANY;
bind(serversocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
bzero((char*)&clientaddr,sizeof(clientaddr));
len=sizeof(clientaddr);
```

```
//connection establishment.
printf("\nwaiting for client connection");
recvfrom(serversocket,req,sizeof(req),0,(struct sockaddr*)&clientaddr,&len);
//Receive message, HI IAM CLIENT
printf("\nThe client connection obtained\t%s\n",req);
//sending request for windowsize.
printf("\nSending request for window size\n");
sendto(serversocket,"REQUEST FOR WINDOWSIZE",sizeof("REQUEST FOR WINDOWSIZE"),0,
(struct sockaddr*)&clientaddr,sizeof(clientaddr));
//obtaining windowsize.
printf("Waiting for the window size\n");
recvfrom(serversocket,(char*)&windowsize,sizeof(windowsize),0,(struct sockaddr*)&clientaddr,&len);
printf("\nThe window size obtained as:\t %d \n",windowsize);
printf("\nObtaining packets from network layer \n");
printf("\nTotal packets obtained :%d\n",(totalpackets=windowsize*5));
printf("\nTotal frames or windows to be transmitted :%d\n",(totalframes=5));
//sending details to client.
printf("\nSending total number of packets \n");
sendto(serversocket,(char*)&totalpackets,sizeof(totalpackets),0,(struct
sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,(struct sockaddr*)&clientaddr,&len);
//receive request from client address via serversocket
printf("\nSending total number of frames \n");
sendto(serversocket,(char*)&totalframes,sizeof(totalframes),0,(struct
sockaddr*)&clientaddr,sizeof(clientaddr));
recvfrom(serversocket,req,sizeof(req),0,(struct sockaddr*)&clientaddr,&len);
```

```
printf("\n Press enter to start the process \n");

fgets(req,2,stdin);

//starting the process of sending

while(i<totalpackets)

{

//initialising the transmit buffer.

bzero((char*)&f1,sizeof(f1));

printf("\nInitializing the transmit buffer \n");

printf("\n The frame to be send is %d with packets:",framesend);

buffer=i;

j=0;

//Builting the frame.

while(j<windowsize && i<totalpackets)

{

printf("%d",i);

f1.packet[j]=i;

j++;

i++;

}

printf("sending frame %d\n",framesend);

//sending the frame.

sendto(serversocket,(char*)&f1,sizeof(f1),0,

(struct sockaddr*)&clientaddr,sizeof(clientaddr));

//Waiting for the acknowledgement.

printf("Waiting for the acknowledgment\n");
```

```
recvfrom(serversocket,(char*)&acknowledgement,sizeof(acknowledgement),0,(struct
sockaddr*)&clientaddr,&len);

//Checking acknowledgement of each packet.

j=0;

k=0;

l=buffer;

while(j<windowsize && l<totalpackets)

{

if(acknowledgement.acknowledge[j]==-1)

{

printf("\nnegative acknowledgement received for packet:%d \n",f1.packet[j]);

printf("\nRetransmitting from packet:%d \n",f1.packet[j]);

i=f1.packet[j];

i=f1.packet[j];

k=l;

break;

}

j++;

l++;

}

if(k==0)

{

printf("\n Positive acknowledgement received for all packets,within the frame:%d \n",framesend);

}

framesend++;

printf("\n press enter to proceed \n");
```

```
fgets(req,2,stdin);

}

printf("\nAll frames sends successfully\n Closing connection with the client \n");

close(serversocket);

}
```

CLIENT SIDE OUTPUT

```
gcc src.c -o client

labb04@labb04:~/Desktop$ ./client

sending request to the server


Waiting for reply


 The server has to send :            REQUEST FOR WINDOWSIZE


enter the window size

3


 sending window size


 waiting for the server response


Total packets are :          15


 total number of frames or windows are:            5
```

```
starting the process of receiving

Initializing the received buffer

The expected frame is 0 with packets:012
waiting for the frame

 received frame 0

 enter -1 to send negative acknowledgement for the following packets

packet:0
0

packet:1
0

packet:2
-1

Initializing the received buffer

The expected frame is 1 with packets:234
waiting for the frame
```

```
 received frame 1

 enter -1 to send negative acknowledgement for the following packets

packet:2
0

packet:3
0

packet:4
0

Initializing the received buffer

The expected frame is 2 with packets:567
waiting for the frame

 received frame 2

 enter -1 to send negative acknowledgement for the following packets

packet:5
0
```

```
packet:6
0


packet:7
0


Initializing the received buffer


The expected frame is 3 with packets:8910
waiting for the frame


 received frame 3


 enter -1 to send negative acknowledgement for the following packets


packet:8
0


packet:9
0


packet:10
0


Initializing the received buffer
```

```
The expected frame is 4 with packets:111213

waiting for the frame


 received frame 4


 enter -1 to send negative acknowledgement for the following packets


packet:11

0


packet:12

0


packet:13

0


Initializing the received buffer


The expected frame is 5 with packets:14

waiting for the frame


 received frame 5


 enter -1 to send negative acknowledgement for the following packets
```

```
packet:14

0


all frames received successfully

 closing connection with the server
```

SERVER SIDE OUTPUT

```
gcc srs.c -o server

labb04@labb04:~/Desktop$ ./server


waiting for client connection

The client connection obtained  HI IAM CLIENT


Sending request for window size

Waiting for the window size


The window size obtained as:      3


Obtaining packets from network layer


Total packets obtained :15


Total frames or windows to be transmitted :5
```

```
Sending total number of packets

Sending total number of frames

 Press enter to start the process



Initializing the transmit buffer

 The frame to be send is 0 with packets:012sending frame 0
Waiting for the acknowlegment

negative acknowledgement received for packet:2

Retransmitting from packet:2

 press enter to proceed



Initializing the transmit buffer

 The frame to be send is 1 with packets:234sending frame 1
Waiting for the acknowlegment

 Positive acknowledgement received for all packets,within the frame:1
 press enter to proceed



Initializing the transmit buffer


 The frame to be send is 2 with packets:567sending frame 2
Waiting for the acknowlegment


 Positive acknowledgement received for all packets,within the frame:2


 press enter to proceed



Initializing the transmit buffer


 The frame to be send is 3 with packets:8910sending frame 3
Waiting for the acknowlegment


 Positive acknowledgement received for all packets,within the frame:3


 press enter to proceed



Initializing the transmit buffer
```

```
 The frame to be send is 4 with packets:111213sending frame 4
Waiting for the acknowlegment

 Positive acknowledgement received for all packets,within the frame:4

 press enter to proceed


Initializing the transmit buffer

 The frame to be send is 5 with packets:14sending frame 5
Waiting for the acknowlegment

 Positive acknowledgement received for all packets,within the frame:5

 press enter to proceed


All frames sends successfully
 Closing connection with the client
labb04@labb04:~/Desktop$
```