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**

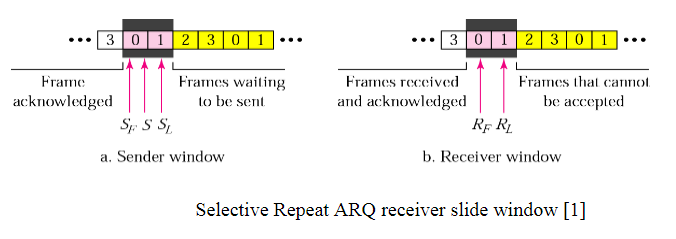
Selective Repeat protocol provides for sending multiple frames depending upon the availability of frames in the sending window, even if it does not receive acknowledgement for any frame in the interim. The maximum number of frames that can be sent depends upon the size of the sending window.

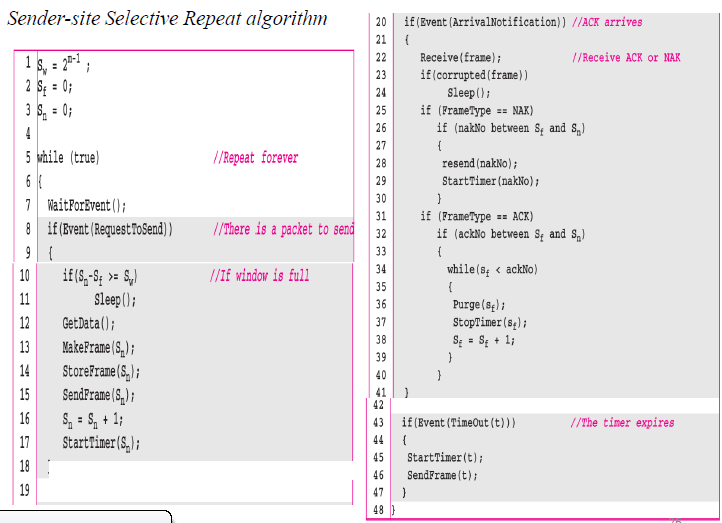
The receiver records the sequence number of the earliest incorrect or un-received frame. It then fills the receiving window with the subsequent frames that it has received. It sends the sequence number of the missing frame along with every acknowledgement frame.

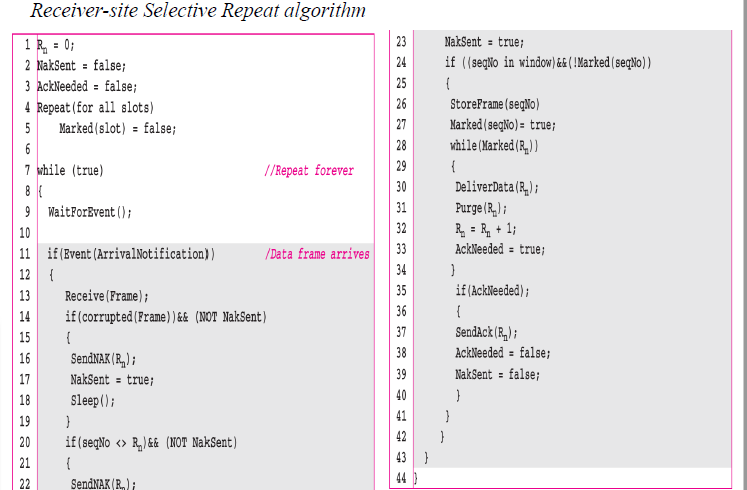
The sender continues to send frames that are in its sending window. Once, it has sent all the frames in the window, it retransmits the frame whose sequence number is given by the acknowledgements. It then continues sending the other frames.

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

SF, SL and S. But the sender sliding window size changed into 2m-1.Receiver sliding window has 2 control variables, RF and RL.







**sr.c**

#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]);

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[10000] = {-1};

int i ;

for(i = 1 ; i <= frame\_size ; i++)

{

int flag = rand() % 2;

if(flag)

{

printf(" Frame[%d] with value %d Acknowledged !!! \n\n", i , frames[i]);

nt++;

}

else

{

printf(" Frame[%d] with value %d Not Acknowledged !!! \n\n", i , frames[i]);

left[k++] = frames[i];

nt++;

}

if(i % window\_size == 0)

{

for(int x = 0 ; x < k ; x++)

{

printf(" Frame[%d] with value %d Retransmitted \n\n", x , left[x]);

nt++;

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]);

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

**Sender Side Algorithm for Selective Repeat ARQ Protocol**

begin

   frame s; //s denotes frame to be sent

   frame t; //t is temporary frame

   S\_window = power(2,m-1); //Assign maximum window size

   SeqFirst = 0; // Sequence number of first frame in window

   SeqN = 0; // Sequence number of Nth frame window

   while (true) //check repeatedly

      do

         Wait\_For\_Event(); //wait for availability of packet

         if ( Event(Request\_For\_Transfer)) then

            //check if window is full

            if (SeqN–SeqFirst >= S\_window) then

               doNothing();

            end if;

            Get\_Data\_From\_Network\_Layer();

            s = Make\_Frame();

            s.seq = SeqN;

            Store\_Copy\_Frame(s);

            Send\_Frame(s);

            Start\_Timer(s);

            SeqN = SeqN + 1;

         end if;

         if ( Event(Frame\_Arrival) then

            r = Receive\_Acknowledgement();

            //Resend frame whose sequence number is with ACK

            if ( r.type = NAK) then

               if ( NAK\_No > SeqFirst && NAK\_No < SeqN ) then

                  Retransmit( s.seq(NAK\_No));

                  Start\_Timer(s);

               end if

             //Remove frames from sending window with positive ACK

               else if ( r.type = ACK ) then

                  Remove\_Frame(s.seq(SeqFirst));

                  Stop\_Timer(s);

                  SeqFirst = SeqFirst + 1;

               end if

         end if

         // Resend frame if acknowledgement haven’t been received

         if ( Event(Time\_Out)) then

            Start\_Timer(s);

            Retransmit\_Frame(s);

         end if

end

**Receiver Side Algorithm for Selective Repeat ARQ Protocol**

Begin

   frame f;

   RSeqNo = 0; // Initialise sequence number of expected frame

   NAKsent = false;

   ACK = false;

   For each slot in receive\_window

   Mark(slot)=false;

   while (true) //check repeatedly

      do

         Wait\_For\_Event(); //wait for arrival of frame

         if ( Event(Frame\_Arrival) then

            Receive\_Frame\_From\_Physical\_Layer();

            if ( Corrupted ( f.SeqNo ) AND NAKsent = false) then

               SendNAK(f.SeqNo);

               NAKsent = true;

            end if

            if ( f.SeqNo != RSeqNo AND NAKsent = false ) then

               SendNAK(f.SeqNo);

               NAKsent = true;

               if ( f.SeqNo is in receive\_window ) then

                  if ( Mark(RSeqNo) = false ) then

                     Store\_frame(f.SeqNo);

                     Mark(RSeqNo) = true;

                  end if

               end if

               else

               while ( Mark(RSeqNo))

                  Extract\_Data(RSeqNo);

                  Deliver\_Data\_To\_Network\_Layer();

                  RSeqNo = RSeqNo + 1;

                  Send\_ACK(RSeqNo);

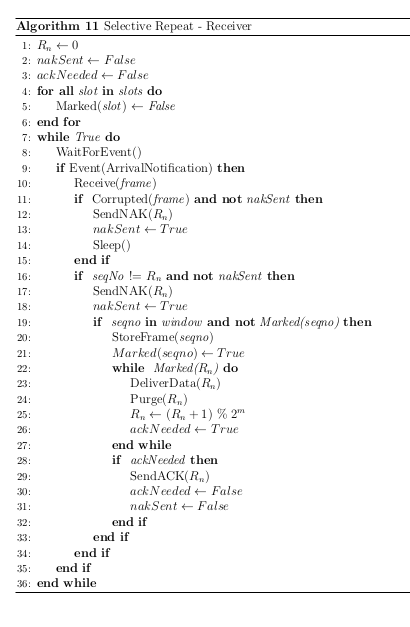
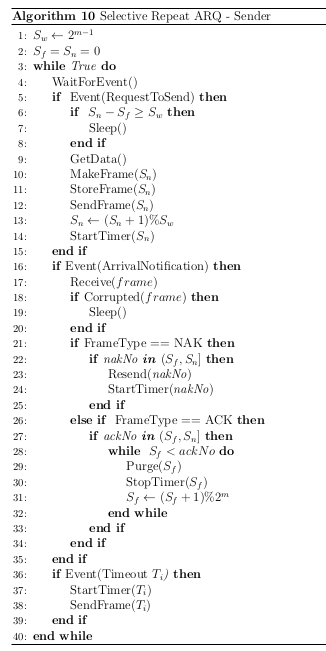
               end while

            end if

         end if

   end while

end



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. |
| 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 acknowledgment from the receiver. Once acknowledged, sender sends another frame to the receiver. If acknowledgment is not received then frame/packet is retransmitted.

**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 send window is much smaller; it is 2^(m−1) . Second, the receiver window has the same size as the sender window.

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. |
| 7 | Retransmissions | In Stop and Wait protocol, in case of packet drop,number of retransmition is 1. | In GoBackN protocol, in case of packet drop,numbers of retransmitions are N. | In Selective Repeat protocol, in case of packet drop,number of retransmition is 1. |

**Selective-Repeat Client/Server Implementation in C**

**srclient.c**

#include<time.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<sys/time.h>

#include<sys/wait.h>

#include<string.h>

#include<unistd.h>

#include<arpa/inet.h>

int isfaulty(){ //simulating corruption of message

int d=rand()%4;

return (d>2);

}

int main() {

srand(time(0));

int c\_sock;

c\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in client;

memset(&client, 0, sizeof(client));

client.sin\_family = AF\_INET;

client.sin\_port = htons(9009);

client.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if(connect(c\_sock, (struct sockaddr\*)&client, sizeof(client)) == -1) {

printf("Connection failed");

return 0;

}

printf("\n\tClient -with individual acknowledgement scheme\n\n");

char msg1[50]="acknowledgement of";

char msg3[50]="negative ack ";

char msg2[50];

char buff[100];

int count=-1,flag=1;

while(count<8){

bzero(buff,sizeof(buff));

bzero(msg2,sizeof(msg2));

if(count==7&&flag==1){

printf("here\n"); //simulate loss

flag=0;

read(c\_sock,buff,sizeof(buff));

continue;

}

int n = read(c\_sock, buff, sizeof(buff));

char i=buff[strlen(buff)-1];

printf("Message received from server : %s \n",buff);

int isfault=isfaulty();

printf("corruption status : %d \n",isfault);

printf("Response/acknowledgement sent for message \n");

if(isfault)

strcpy(msg2,msg3);

else{

strcpy(msg2,msg1);

count++;}

msg2[strlen(msg2)]=i;

write(c\_sock,msg2, sizeof(msg2));

}

**srserver.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<sys/time.h>

#include<netinet/in.h>

#include<string.h>

#include<unistd.h>

#include<arpa/inet.h>

#include<fcntl.h>

void rsendd(int ch,int c\_sock){

char buff2[60];

bzero(buff2,sizeof(buff2));

strcpy(buff2,"reserver message :");

buff2[strlen(buff2)]=(ch)+'0';

buff2[strlen(buff2)]='\0';

printf("Resending Message to client :%s \n",buff2);

write(c\_sock, buff2, sizeof(buff2));

usleep(1000);

}

int main() {

int s\_sock, c\_sock;

s\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in server, other;

memset(&server, 0, sizeof(server));

memset(&other, 0, sizeof(other));

server.sin\_family = AF\_INET;

server.sin\_port = htons(9009);

server.sin\_addr.s\_addr = INADDR\_ANY;

socklen\_t add;

if(bind(s\_sock, (struct sockaddr\*)&server, sizeof(server)) == -1) {

printf("Binding failed\n");

return 0;

}

printf("\tServer Up\n Selective repeat scheme\n\n");

listen(s\_sock, 10);

add = sizeof(other);

c\_sock = accept(s\_sock, (struct sockaddr\*)&other, &add);

time\_t t1,t2;

char msg[50]="server message :";

char buff[50];

int flag=0;

fd\_set set1,set2,set3;

struct timeval timeout1,timeout2,timeout3;

int rv1,rv2,rv3;

int tot=0;

int ok[20];

memset(ok,0,sizeof(ok));

while(tot<9){

int toti=tot;

for(int j=(0+toti);j<(3+toti);j++){

bzero(buff,sizeof(buff));

char buff2[60];

bzero(buff2,sizeof(buff2));

strcpy(buff2,"server message :");

buff2[strlen(buff2)]=(j)+'0';

buff2[strlen(buff2)]='\0';

printf("Message sent to client :%s \t%d\t%d\n",buff2,tot,j);

write(c\_sock, buff2, sizeof(buff2));

usleep(1000);

}

for(int k=0+toti;k<(toti+3);k++){

qq:

FD\_ZERO(&set1);

FD\_SET(c\_sock, &set1);

timeout1.tv\_sec = 2;

timeout1.tv\_usec = 0;

rv1 = select(c\_sock + 1, &set1, NULL, NULL, &timeout1);

if(rv1 == -1)

perror("select error ");

else if(rv1 == 0){

printf("Timeout for message :%d \n",k);

rsendd(k,c\_sock);

goto qq;} // a timeout occured

else{

read(c\_sock, buff, sizeof(buff));

printf("Message from Client: %s\n", buff);

if(buff[0]=='n'){

printf(" corrupt message acknowledgement (msg %d) \n",buff[strlen(buff)-1]-'0');

rsendd((buff[strlen(buff)-1]-'0'),c\_sock);

goto qq;}

else

tot++;

}

}

}

close(c\_sock);

close(s\_sock);

return 0;

}

gcc srserver.c -o s

cca@labb30:~$ ./s

Server Up

Selective repeat scheme

Message sent to client :server message :0 0 0

Message sent to client :server message :1 0 1

Message sent to client :server message :2 0 2

Message from Client: acknowledgement of0

Message from Client: acknowledgement of1

Message from Client: acknowledgement of2

Message sent to client :server message :3 3 3

Message sent to client :server message :4 3 4

Message sent to client :server message :5 3 5

Message from Client: negative ack 3

corrupt message awk (msg 3)

Resending Message to client :reserver message :3

Message from Client: acknowledgement of4

Message from Client: negative ack 5

corrupt message awk (msg 5)

Resending Message to client :reserver message :5

Message from Client: acknowledgement of3

Message from Client: acknowledgement of5

Message sent to client :server message :6 6 6

Message sent to client :server message :7 6 7

Message sent to client :server message :8 6 8

Message from Client: negative ack 6

corrupt message awk (msg 6)

Resending Message to client :reserver message :6

Message from Client: acknowledgement of7

Message from Client: acknowledgement of8

Timeout for message :8

Resending Message to client :reserver message :8

Message from Client: acknowledgement of8

cca@labb30:~$ gcc srclient.c -o c

cca@labb30:~$ ./c

Client -with individual acknowledgement scheme

Message received from server : server message :0

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :1

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :2

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :3

corruption status : 1

Response/acknowledgement sent for message

Message received from server : server message :4

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :5

corruption status : 1

Response/acknowledgement sent for message

Message received from server : reserver message :3

corruption status : 0

Response/acknowledgement sent for message

Message received from server : reserver message :5

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :6

corruption status : 1

Response/acknowledgement sent for message

Message received from server : server message :7

corruption status : 0

Response/acknowledgement sent for message

Message received from server : server message :8

corruption status : 0

Response/acknowledgement sent for message

here

Message received from server : reserver message :8

corruption status : 0

Response/acknowledgement sent for message