GO BACK N ARQ IMPLEMENTATION

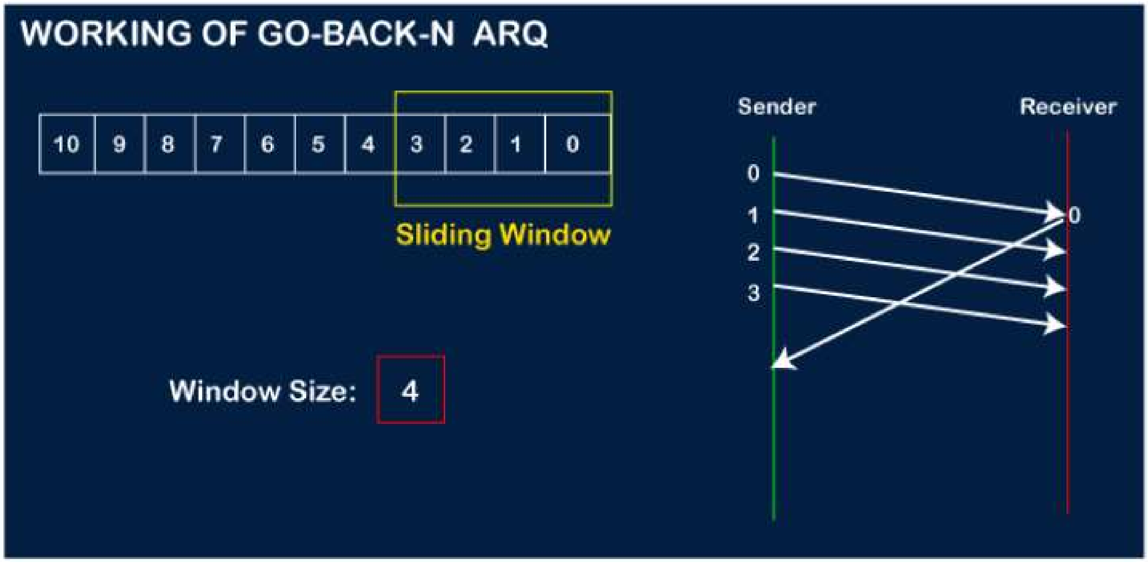
A**im**:- write a program to perform simulation on sliding window protocol using Go-back-N ARQ for noisy channel.

Description:-

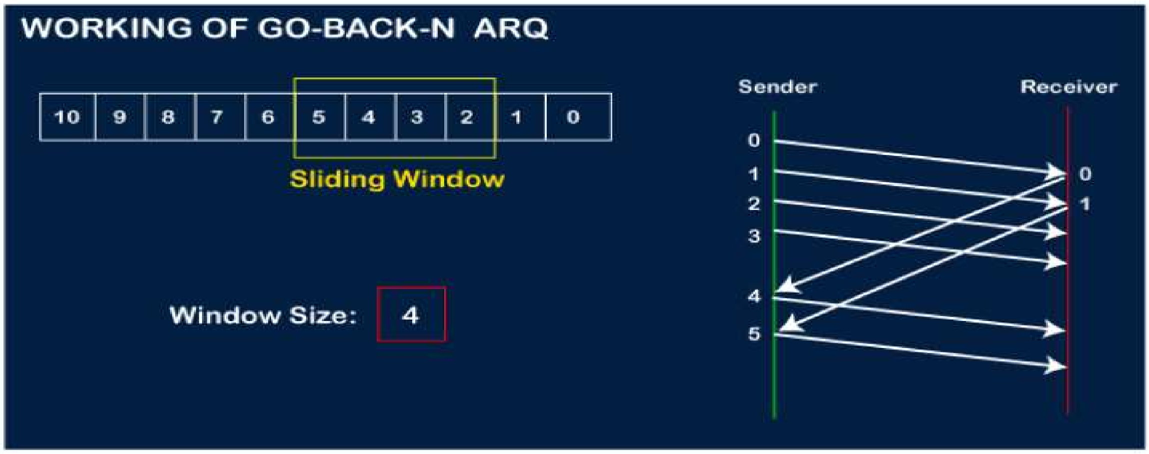
Go-Back-N ARQ is mainly a specific instance of Automatic Repeat Request **(ARQ) protocol** where the sending process continues to send a number of frames as specified by the window size even without receiving an acknowledgement **(ACK) packet** from the receiver. The sender keeps a copy of each frame until the acknowledgement arrives.

This protocol is a practical approach to the sliding window.

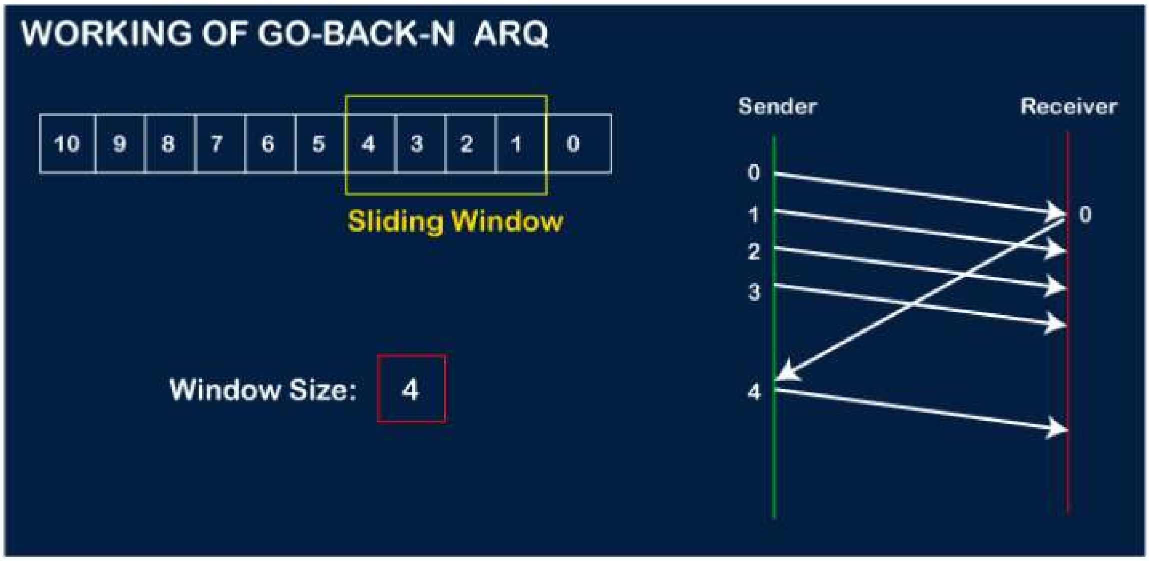
- In Go-Back-N ARQ, the size of the sender window is N and the size of the receiver window is always 1.
- This protocol makes the use of **cumulative acknowledgements** means here the receiver maintains an acknowledgement timer.
- If the receiver receives a corrupted frame, then it silently discards that corrupted frame and the correct frame is retransmitted by the sender after the timeout timer expires.
- In case if the receiver receives the out of order frame then it simply discards all the frames.
- In case if the sender does not receive any acknowledgement then the frames in the entire window will be retransmitted again.



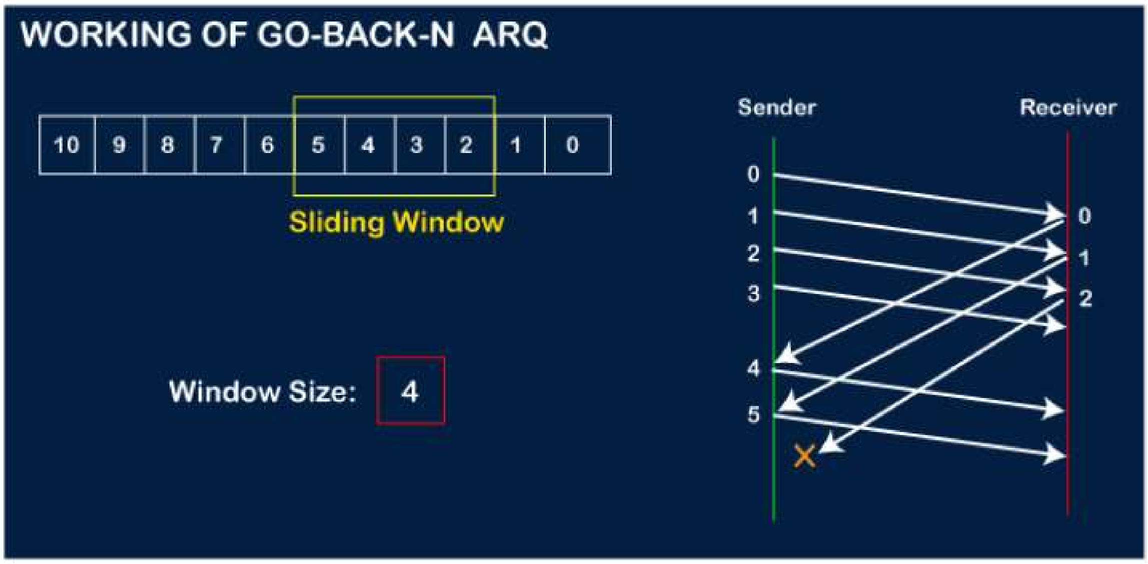
WORKING OF GO-BACK-N ARQ

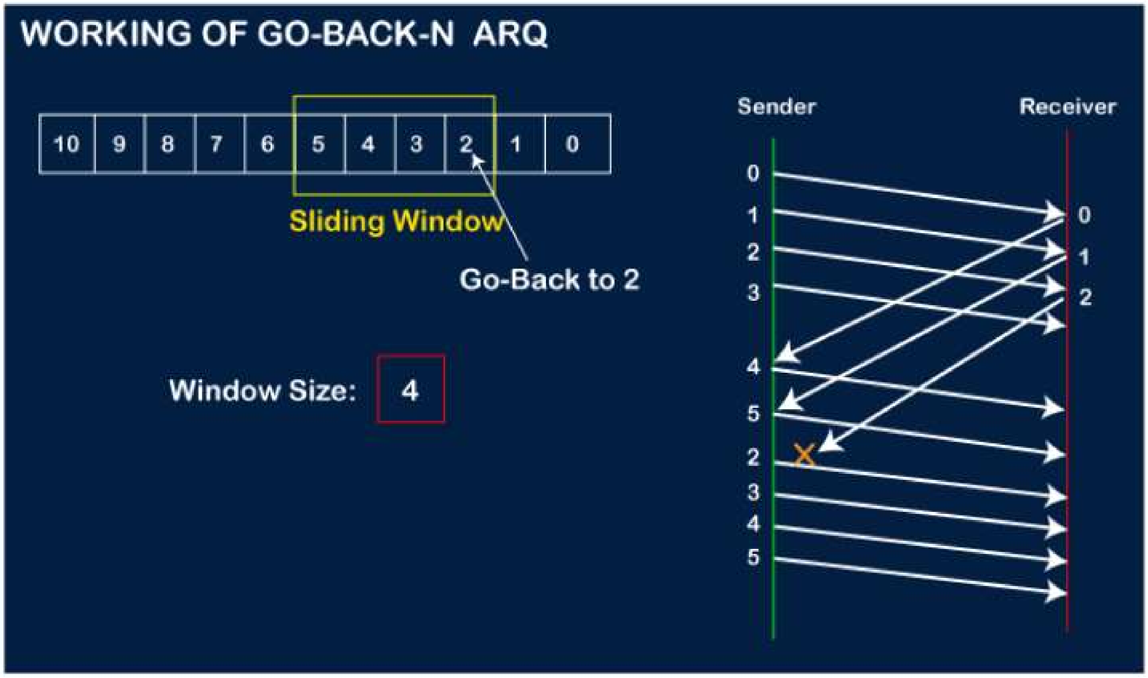


WORKING OF GO-BACK-N ARQ

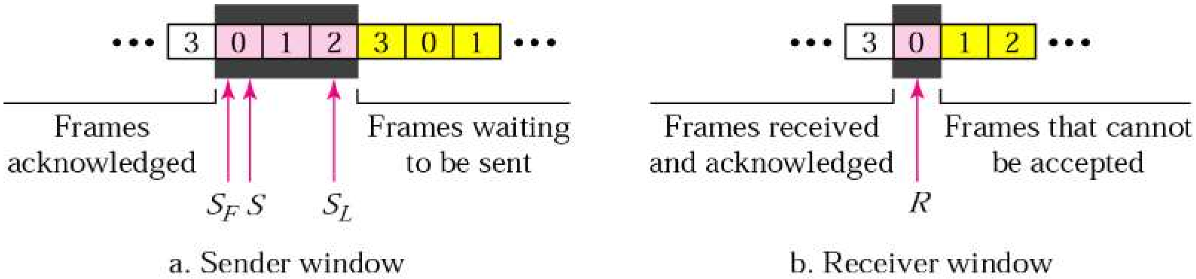


WORKING OF GO-BACK-N ARQ


WORKING OF GO-BACK-N ARQ

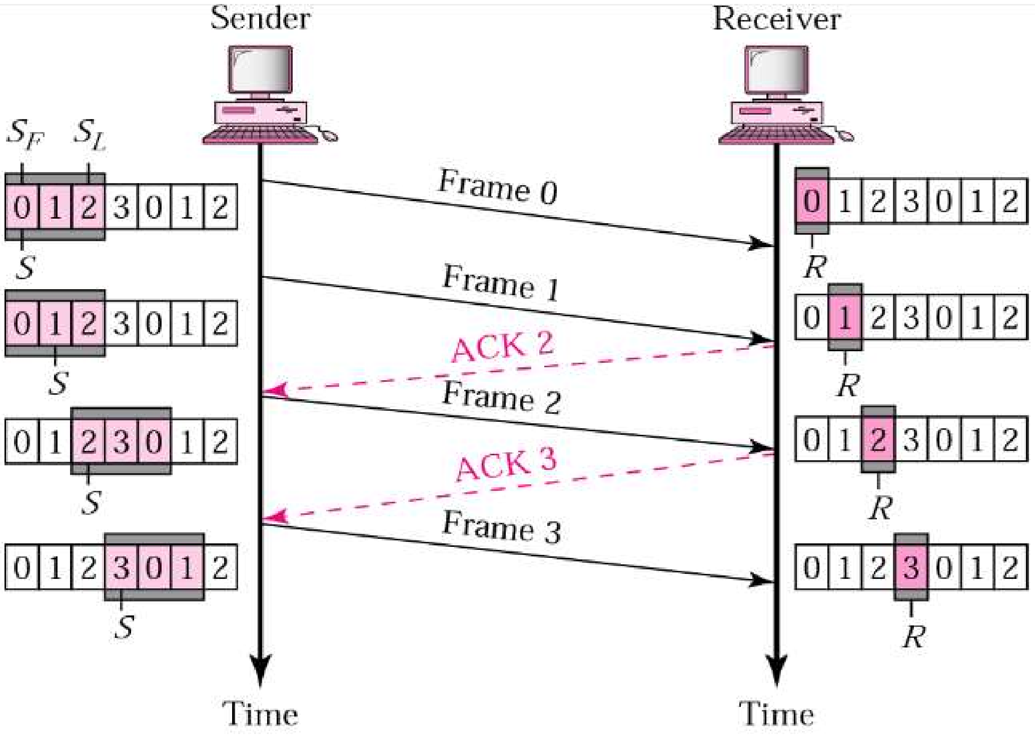

WORKING OF GO-BACK-N ARQ

S_F is the sequence number of the first frame in the sliding window, S_L is the sequence number of the last frame in the sliding window. R is the sequence number of the expected frame. $W=S_L-S_F+1=N$ itself, indicating the number of frames. Only when R and sequence number of received frame are matched, frame is accepted, otherwise discard frame. Receiver window size is 1. Consequently, the size of the sending window is 2^n-1. Thus in order to accommodate a sending window size of 2^n-1, an n-bit sequence number is chosen. The maximum window size = **2^3 - 1 = 7** i.e window will carry frames from 0 to 6 which are 7 in number. If window size is 3, we use a 2-bit sequence number to transmit frames, i.e 2^2-1=3.



3
0
1
2
3
0
1
Frames acknowledged
Frames waiting to be sent
S_F S
S_L
a. Sender window
3
0
1
2
Frames received and acknowledged
Frames that cannot be accepted
R
b. Receiver window

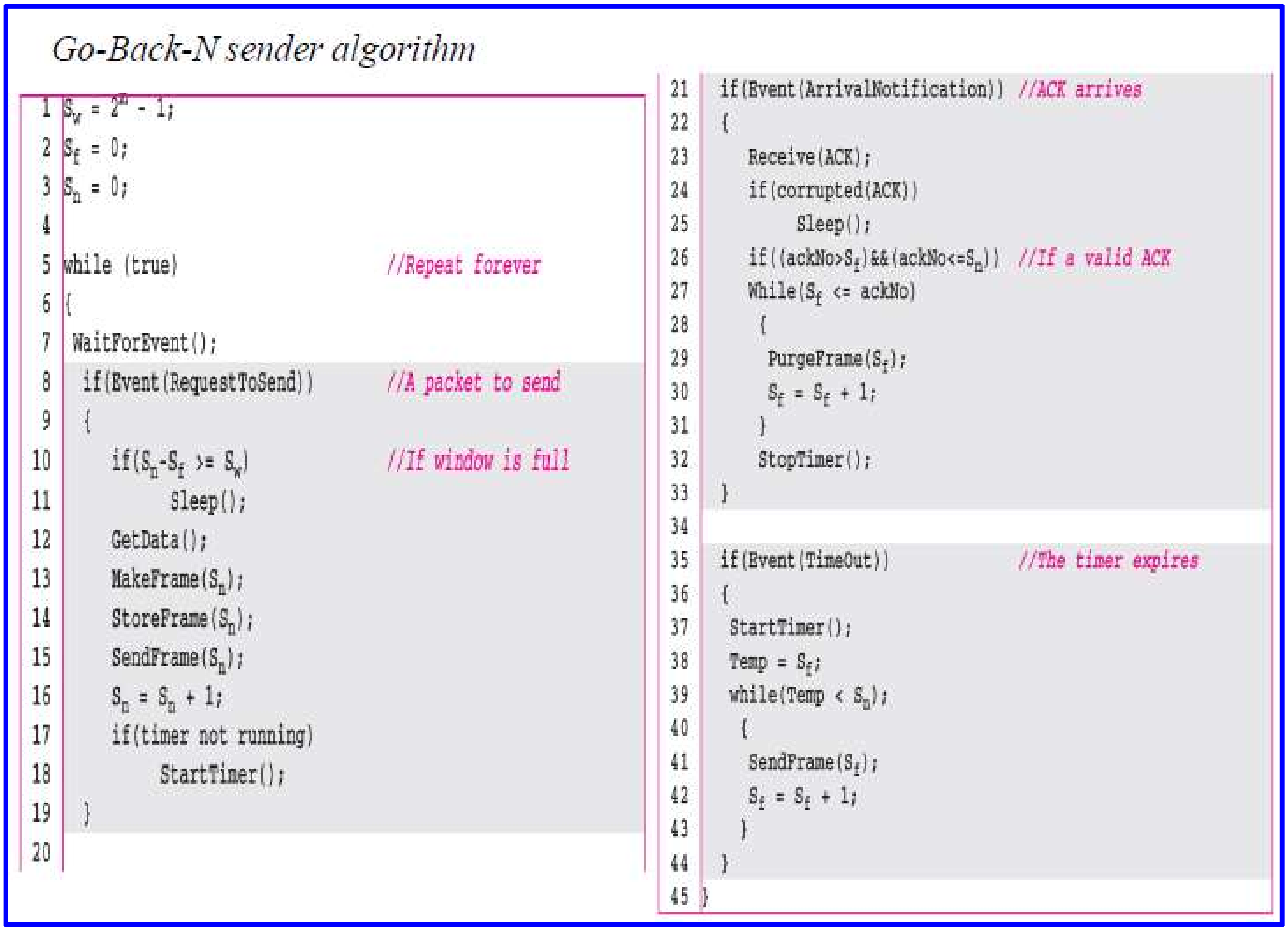
Go-Back-N ARQ control variable [1]



Go-Back-N ARQ normal operation [1]

Frame 0 &1 send, ACK 1 & 2 back to sender. Frame 2 send, ACK 3 back to sender.

Algorithm to be written in record



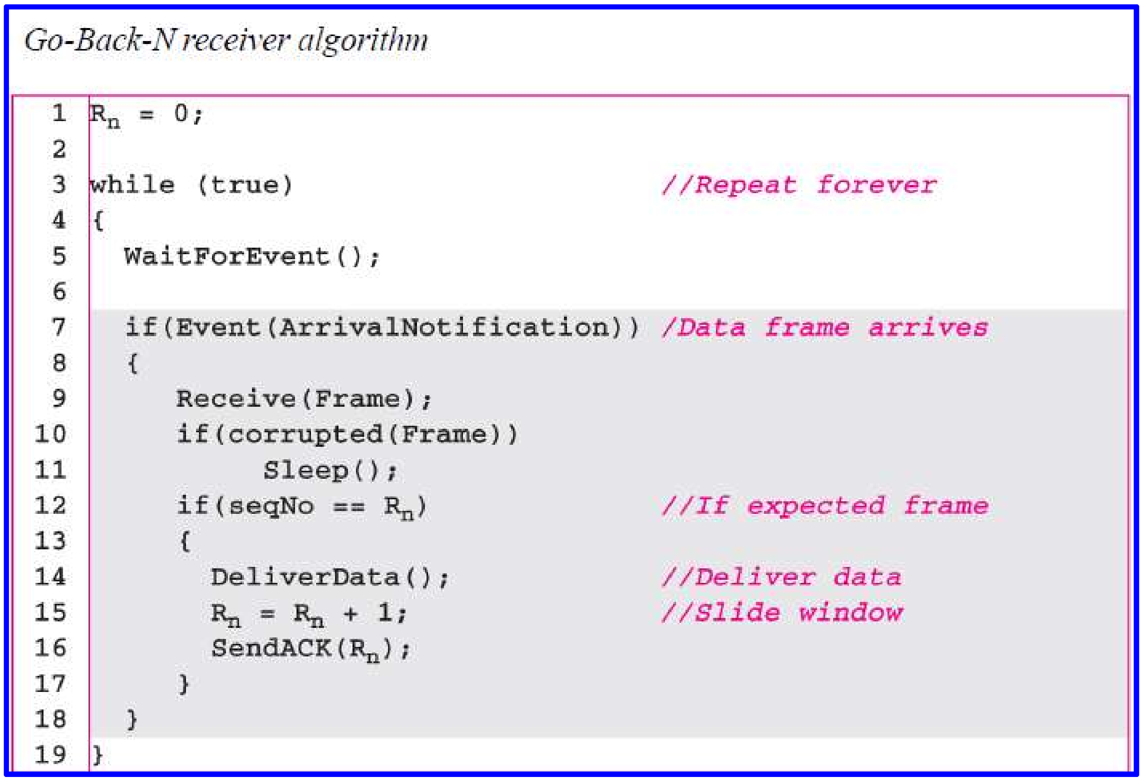
Go-Back-N sender algorithm

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

while (true)                          //Repeat forever
{
 WaitForEvent();
  if(Event(RequestToSend))            //A packet to send
  {
     if($S_n$-$S_f$ >= $S_w$)                //If window is full
           Sleep();
     GetData();
     MakeFrame($S_n$);
     StoreFrame($S_n$);
     SendFrame($S_n$);
     $S_n = S_n + 1$;
     if(timer not running)
          StartTimer();
  }

  if(Event(ArrivalNotification)) //ACK arrives
  {
     Receive(ACK);
     if(corrupted(ACK))
          Sleep();
     if((ackNo>$S_f$)&&(ackNo<=$S_n$))  //If a valid ACK
     While($S_f$ <= ackNo)
      {
       PurgeFrame($S_f$);
       $S_f = S_f + 1$;
      }
      StopTimer();
  }

  if(Event(TimeOut))                 //The timer expires
  {
   StartTimer();
   Temp = $S_f$;
   while(Temp < $S_n$);
    {
     SendFrame($S_f$);
     $S_f = S_f + 1$;
    }
  }
}
```



Go-Back-N receiver algorithm

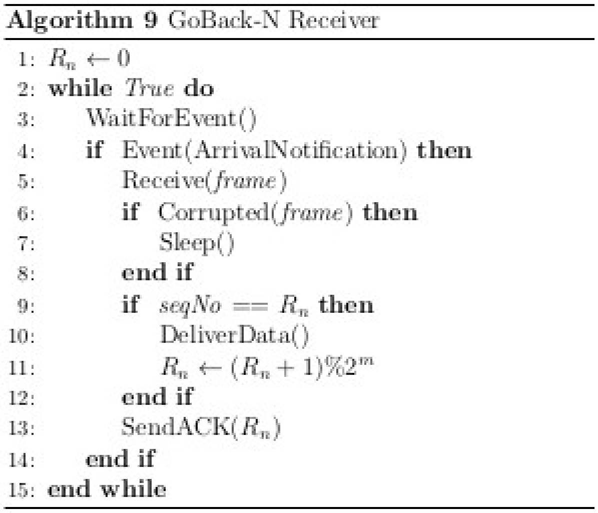
```
$R_n$ = 0;

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

  if(Event(ArrivalNotification)) /Data frame arrives
  {
     Receive(Frame);
     if(corrupted(Frame))
          Sleep();
     if(seqNo == $R_n$)                   //If expected frame
     {
       DeliverData();                     //Deliver data
       $R_n$ = $R_n$ + 1;                 //Slide window
       SendACK($R_n$);
     }
  }
}
```

Algorithm 8 GoBack-N Protocol - 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($S_n$)
        $S_n \leftarrow (S_n + 1)\%S_w$
        if $Timer$ is not $running$ then
            StartTimer()
        end if
    end if
    if Event(ArrivalNotification) then
        Receive(ACK)
        if Corrupted(ACK) then
            Sleep()
        end if
        if $ackNo > S_f$and$ackNo <= S_n$ then
            while $S_f \leq ackNo$ do
                PurgeFrame($S_n$)
                $S_f \leftarrow (S_f + 1)\%S_w$
            end while
        end if
        StopTimer()
    end if
    if Event(Timeout) then
        StartTimer()
        $temp \leftarrow S_f$
        while $temp < S_n$ do
            SendFrame($S_n$)
            $S_f \leftarrow (S_f + 1)\%S_w$
        end while
    end if
end while
```



Algorithm 9 GoBack-N Receiver

```
$R_n \leftarrow 0$
while True do
    WaitForEvent()
    if Event(ArrivalNotification) then
        Receive(frame)
        if Corrupted(frame) then
            Sleep()
        end if
        if $seqNo == R_n$ then
            DeliverData()
            $R_n \leftarrow (R_n + 1)\%2^m$
        end if
        SendACK($R_n$)
    end if
end while
```

GoBackN.c

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>

int main()

{

 int nf,N;

 int tr=0;

 srand(time(NULL));

 printf("Enter the number of frames : ");

 scanf("%d",&nf);

 printf("Enter the Window Size : ");

 scanf("%d",&N);

 int i=1;

 while(i<=nf)

 {
  int x=0;

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

  {

    printf("Sent Frame %d \n", j);

    tr++; //After each frame is send, increment tr by 1 to track total number of transmissions

  }

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

  {

    int flag = rand()%2; //lost frame set as frame 2

    if(!flag)

      {

        printf("%d : Acknowledged! \n", j);

        x++; //After acknowledging frame, increment x indicating success of frame transmission

      }

    else

      {   printf("Frame %d Not Received \n", j);

        printf("Retransmitting Window \n");

        break;

      }

  }

  printf("\n");

  i+=x; //i updates number of successful transfers - all acknowledged frames

}

printf("Total number of transmissions : %d \n", tr);

return 0; }
```

Output

```
gcc goBackN.c
net@inlab:~$ ./a.out
Enter the number of
frames : 5
Enter the Window Size : 2
Sent Frame 1
Sent Frame 2
1 : Acknowledged!
Frame 2 Not Received
Retransmitting Window

Sent Frame 2
Sent Frame 3
2 : Acknowledged!
3 : Acknowledged!

Sent Frame 4
Sent Frame 5
4 : Acknowledged!
5 : Acknowledged!

Total number of
transmissions : 6
```

Go-Back N Client/Server Implementation in C

gbns.c

```
#include <stdio.h>

#include <stdlib.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>
```

```
#include <fcntl.h>

#include<string.h>

#include<unistd.h>

void itoa(int number, char numberString[])


{ numberString[0] = (char)(number + 48);

//integer to ascii conversion by adding 48 and then typecasting to character

  numberString[1] = '\0';

}


int main()

{

int sockfd, newSockFd, size, windowStart = 1, windowCurrent = 1, windowEnd = 4, oldWindowStart, flag;

char buffer[100];

socklen_t len;

struct sockaddr_in server, client;

server.sin_family = AF_INET;

server.sin_port = 3033;

server.sin_addr.s_addr = INADDR_ANY;

sockfd = socket(AF_INET, SOCK_STREAM, 0);


printf("\nStarting up...");

int k;

k=bind(sockfd, (struct sockaddr *)&server, sizeof(server)); //bind socket with ip addr of server

if(k==-1)
```

```
 printf("Error in binding");


len = sizeof(client);


listen(sockfd,1);

//listen to 1 active connection to client

newSockFd = accept(sockfd, (struct sockaddr *)&client,&len); //accept client connection

recv(newSockFd, buffer, 100, 0);//receive message from client

fcntl(newSockFd,F_SETFL,O_NONBLOCK);//It allows the program to place a read or a write lock.

printf("\n Received a request from client. Sending packets one by one...");

do

{

if(windowCurrent != windowEnd)

{


itoa(windowCurrent, buffer); //convert current window no to ascii and copy to buffer

send(newSockFd, buffer, 100, 0); //send window no to server through newSockFd

printf("\nPacket Sent: %d\n",windowCurrent); //show which window number the transmitted frame
was

windowCurrent++; //After sending packet, increment the current window

}


recv(newSockFd, buffer, 100, 0); // receive message from Client


if(buffer[0]=='R') //check if buffer contained retransmission request packet denoting 'R'

{
```

```
//resend packet number in buffer[1]
printf("\n** Received a RETRANSMIT packet.\n Resending packet no. %c...", buffer[1]);
itoa((atoi(&buffer[1])), buffer); //copy packet number as ascii value to buffer
send(newSockFd, buffer, 100, 0); //send packet number to client
windowCurrent = atoi(&buffer[0]); //note down the window number of retransmitted frame
windowCurrent++; //increment window after retransmission
}

else if(buffer[0] == 'A')  //check if incoming buffer contained acknowledgement denoted by ‘A’
{
oldWindowStart = windowStart; //initialize 1 as window starting index
// update the new window no based on acknowledgement from receiver
windowStart = atoi(&buffer[1]) + 1; windowEnd += (windowStart - oldWindowStart);
//print on screen which ACK was received
printf("\n** Received ACK %c. Moving window boundary.",buffer[1]);
}
}
 while(windowCurrent!= 10);
close(sockfd);
close(newSockFd);
printf("\nSending Complete. Sockets closed.Exiting...\n");
return(0);
}
```

<u>gbnc.c</u>

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include<unistd.h>

int main()

{

int sockfd, newSockFd, size, firstTime = 1, currentPacket, wait = 3;

char data[100], digit[2];

struct sockaddr_in client;

sockfd = socket(AF_INET, SOCK_STREAM,0);


client.sin_family = AF_INET;

client.sin_port = 3033;

client.sin_addr.s_addr = INADDR_ANY;


printf("\nStarting up...");

size = sizeof(client);

printf("\nEstablishing Connection to Server...");

connect(sockfd, (struct sockaddr *)&client, size);


sprintf(data, "REQUEST");

send(sockfd, data, strlen(data), 0); //send REQUEST message to server
```

```
do
{
recv(sockfd, data, 100, 0); //receive data from server
currentPacket = atoi(data); //note current packet number
printf("\nGot packet: %d", currentPacket);
if(currentPacket == 3 && firstTime)
{ //issue a retransmission after receiving packets until packet 3
printf("\n*** Simulation: Packet data corrupted or incomplete.");
printf("\n*** Sending RETRANSMIT for packet 1.");
memset(&data, 0, sizeof(data)); //clear buffer data
sprintf(data,"R1"); //Code for message requesting retransmission is R1 to retransmit packet 1
send(sockfd, data, strlen(data), 0);
//send R1 message to server
firstTime =0;
}
else
{ wait--; //wait time is initialized as 3ms. we can reduce wait time till 0
if(!wait)
{
printf("\n*** Packet Accepted -> Sending ACK");
wait = 3; //after accepting packet, reset wait time as 3ms
sprintf(data, "A");
digit[0] = (char)(currentPacket + 48); //convert packet number to ascii value
digit[1] = '\0';
strcat(data, digit); //concatenate A and packet number together for each packet
send(sockfd, data,strlen(data),0); //send acknowledgement to server
}
}
}
while(currentPacket != 9);
printf("\nAll packets received...Exiting.");
 close(sockfd);
return(0);
}
```

SERVER SIDE OUTPUT – FIRST RUN SERVER

```
labb04@labb04:~/Desktop$ gcc gbns1.c -o s1
labb04@labb04:~/Desktop$ ./s1

Starting up...
Recieved a request from client. Sending packets one by one...
Packet Sent: 1

Packet Sent: 2

Packet Sent: 3

** Received a RETRANSMIT packet.
 Resending packet no. 1...
Packet Sent: 2
Packet Sent: 3


** Received ACK 1. Moving window boundary.

Packet Sent: 4


** Received ACK 4. Moving window boundary.

Packet Sent: 5


Packet Sent: 6


Packet Sent: 7


** Received ACK 7. Moving window boundary.

Packet Sent: 8


Packet Sent: 9


Sending Complete. Sockets closed.Exiting…
```

CLIENT SIDE OUTPUT

```
labb04@labb04:~/Desktop$ gcc gbnc.c -o c1

labb04@labb04:~/Desktop$ ./c1


Starting up...

Establishing Connection...
```

```
Got packet: 1

Got packet: 2

Got packet: 3

*** Simulation: Packet data corrupted or incomplete.

*** Sending RETRANSMIT for packet 1.

Got packet: 1

*** Packet Accepted -> Sending ACK

Got packet: 2

Got packet: 3

Got packet: 4

*** Packet Accepted -> Sending ACK

Got packet: 5

Got packet: 6

Got packet: 7

*** Packet Accepted -> Sending ACK

Got packet: 8

Got packet: 9

All packets received...Exiting.
```