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

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.

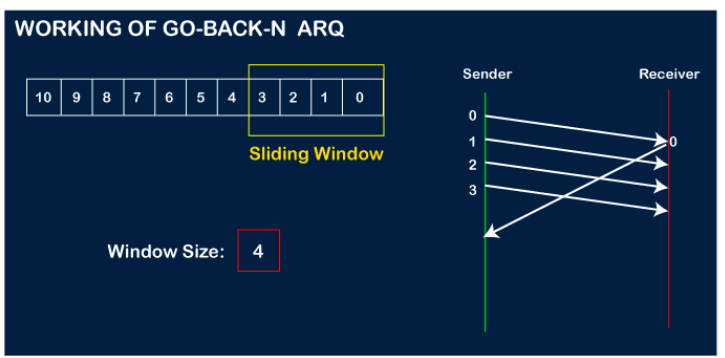
* In Go-Back-N ARQ, the size of the sender window is N and the size of the receiver window is always 1.
* This protocol makes the use of **cumulative acknowledgements** means here the receiver maintains an acknowledgement timer.
* If the receiver receives a corrupted frame, then it silently discards 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.

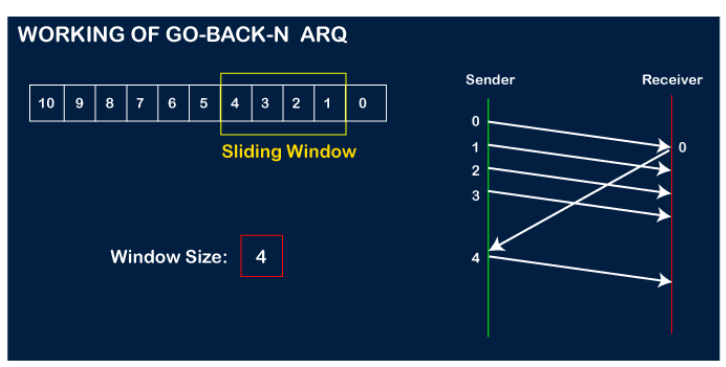
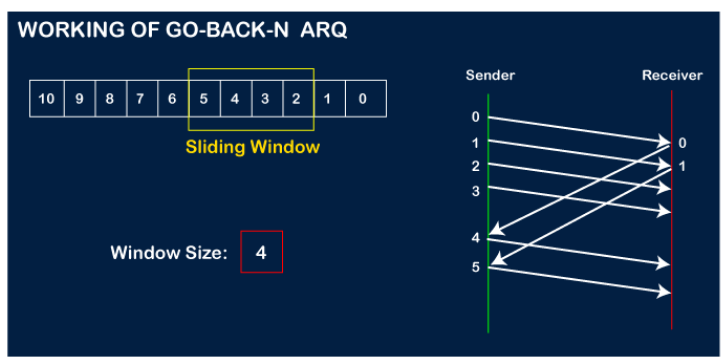
Disadvantages

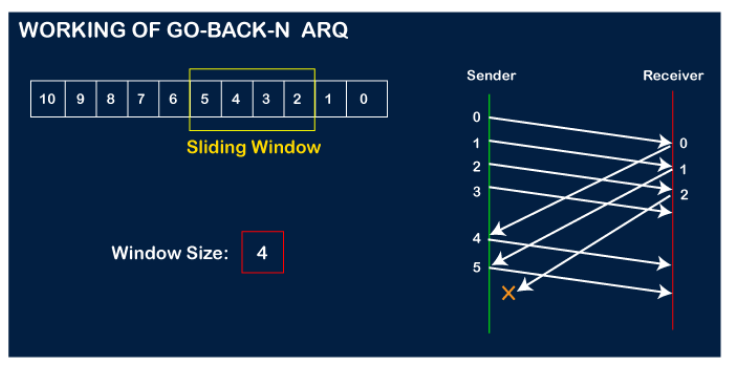
* Timeout timer runs at the receiver side only.
* The transmitter needs to store the last N packets.
* The retransmission of many error-free packets follows an erroneous packet.

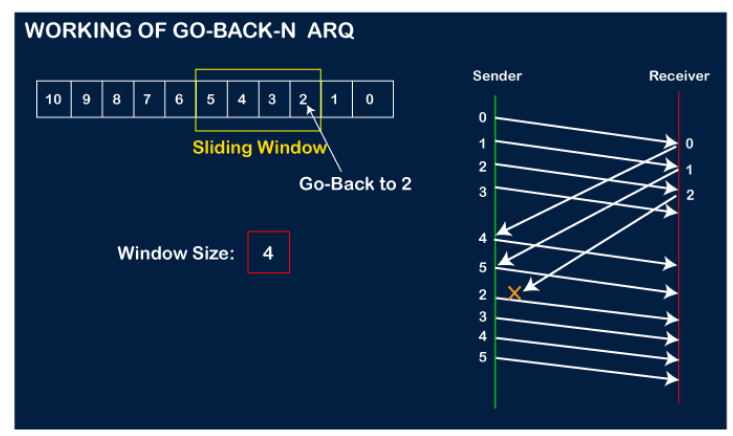
Go-Back-N ARQAlgorithm

1.

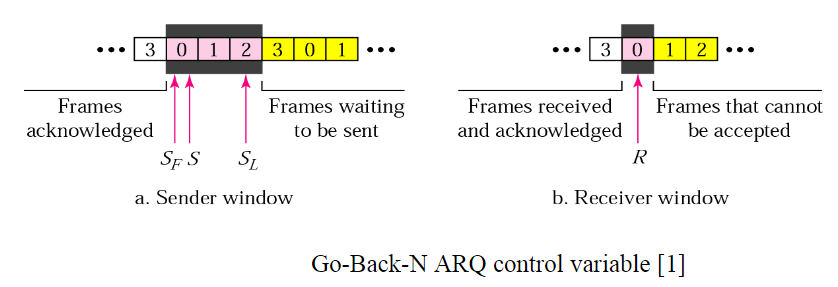
****

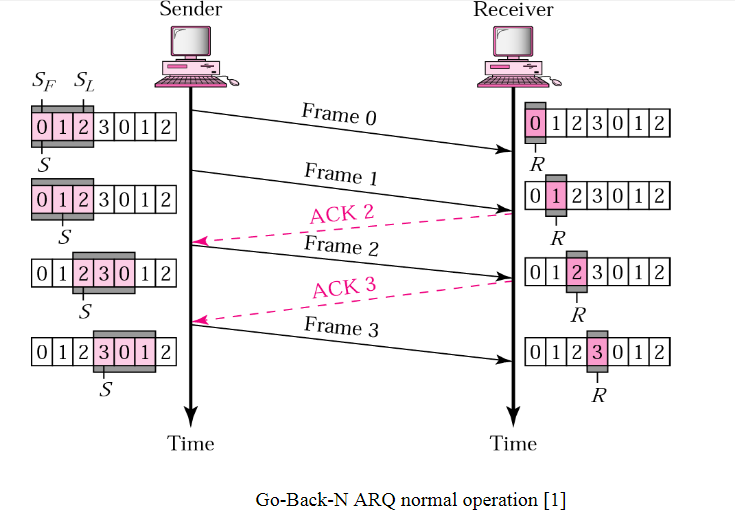
****

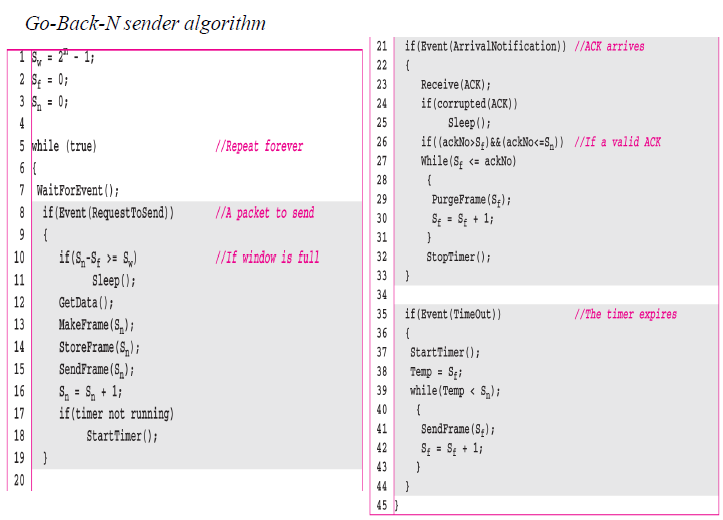
****

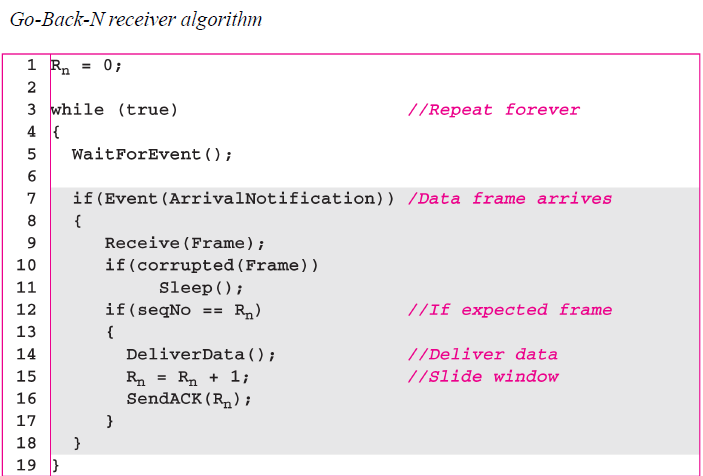
****

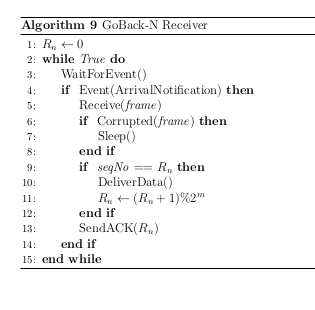
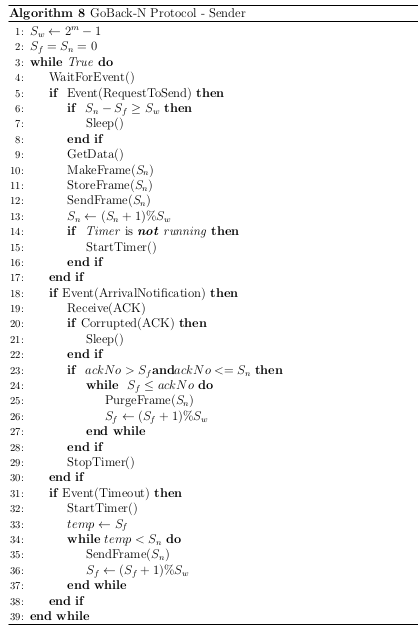
SF is the sequence number of the first frame in the slide window, SL is the sequence number of the last frame in the slide window. R is the sequence number of the excepted frame. W=SL-SF+1=2m-1. Only when R and sequence number of received frame are matched, frame accept, otherwise discard it.

****

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







**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++;

}

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

{

int flag = rand()%2;

if(!flag)

{

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

x++;

}

else

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

printf("Retransmitting Window \n");

break;

}

}

printf("\n");

i+=x;

}

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**  **gbnclient.c** |

#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 main() {

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 msg2[50];

char buff[100];

int flag=1,flg=1;

for(int i=0;i<=9;i++) {

flg=1;

bzero(buff,sizeof(buff));

bzero(msg2,sizeof(msg2));

if(i==8&&flag==1){

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

flag=0;

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

}

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

if(buff[strlen(buff)-1]!=i+'0'){ //out of order

printf("Discarded as out of order \n");

i--;

}

else{

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

printf("Acknowledgement sent for message \n");

strcpy(msg2,msg1);

msg2[strlen(msg2)]=i+'0';

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

}

}

close(c\_sock);

return 0;

}

**gbnserver.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>

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 Go back n (n=3) used to send 10 messages \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 i=-1;

qq:

i=i+1;

bzero(buff,sizeof(buff));

char buff2[60];

bzero(buff2,sizeof(buff2));

strcpy(buff2,"server message :");

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

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

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

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

usleep(1000);

i=i+1;

bzero(buff2,sizeof(buff2));

strcpy(buff2,msg);

buff2[strlen(msg)]=i+'0';

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

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

i=i+1;

usleep(1000);

qqq:

bzero(buff2,sizeof(buff2));

strcpy(buff2,msg);

buff2[strlen(msg)]=i+'0';

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

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

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("Going back from %d:timeout \n",i);

i=i-3;

goto qq;}

else{

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

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

i++;

if(i<=9)

goto qqq;

}

qq2:

FD\_ZERO(&set2);

FD\_SET(c\_sock, &set2);

timeout2.tv\_sec = 3;

timeout2.tv\_usec = 0;

rv2 = select(c\_sock + 1, &set2, NULL, NULL, &timeout2);

if(rv2 == -1)

perror("select error "); // an error accured

else if(rv2 == 0){

printf("Going back from %d:timeout on last 2\n",i-1);

i=i-2;

bzero(buff2,sizeof(buff2));

strcpy(buff2,msg);

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

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

usleep(1000);

bzero(buff2,sizeof(buff2));

i++;

strcpy(buff2,msg);

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

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

goto qq2;} // a timeout occured

else{

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

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

bzero(buff,sizeof(buff));

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

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

}

close(c\_sock);

close(s\_sock);

return 0;

}

**Output**

gcc gbserver.c -o s

cca@labb30:~$ ./s

Server Up

Go back n (n=3) used to send 10 messages

Message sent to client :server message :0

Message sent to client :server message :1

Message sent to client :server message :2

Message from Client: acknowledgement of0

Message sent to client :server message :3

Message from Client: acknowledgement of1

Message sent to client :server message :4

Message from Client: acknowledgement of2

Message sent to client :server message :5

Message from Client: acknowledgement of3

Message sent to client :server message :6

Going back from 6:timeout

Message sent to client :server message :4

Message sent to client :server message :5

Message sent to client :server message :6

Message from Client: acknowledgement of4

Message sent to client :server message :7

Message from Client: acknowledgement of5

Message sent to client :server message :8

Message from Client: acknowledgement of6

Message sent to client :server message :9

Message from Client: acknowledgement of7

Going back from 9:timeout on last 2

Message from Client: acknowledgement of8

Message from Client: acknowledgement of9

gcc gbclient.c -o c

cca@labb30:~$ ./c

Client -with individual acknowledgement scheme

Message received from server : server message :0 0

Acknowledgement sent for message

Message received from server : server message :1 1

Acknowledgement sent for message

Message received from server : server message :2 2

Acknowledgement sent for message

Message received from server : server message :3 3

Acknowledgement sent for message

Discarded as out of order

Discarded as out of order

Discarded as out of order

Message received from server : server message :4 4

Acknowledgement sent for message

Message received from server : server message :5 5

Acknowledgement sent for message

Message received from server : server message :6 6

Acknowledgement sent for message

Message received from server : server message :7 7

Acknowledgement sent for message

here

Discarded as out of order

Message received from server : server message :8 8

Acknowledgement sent for message

Message received from server : server message :9 9

Acknowledgement sent for message