****

**Project Report**

**On**

**Go Back n Sliding Window as Flow Control & CRC-32**

**As Error Checking Technique**

Course Code: CSE350

Course Title: Data Communication

Section: 02

**Submitted To**

Dr. Maheen Islam

Assistant Professor

Department of Computer Science and Engineering

East West University

**Submitted By**

Md Mashrur Rahman

2015-2-60-020

Md Mustafizur Rahman

2015-2-60-100

Kowshik Ahmed

2016-2-60-094

**Date: 04-04-2018**

**Project Statement:**

We are given a project on sending and receiving data frames in data layer protocol and find if there is any error in the data frames or not. We will use Go Back n Sliding Window to send and receive data and CRC-32 to check if there is any error or not.

**Overview:**

There are different types of flow control techniques. Among them we will use **Go Back n Sliding Window for** this project which is another method. With this flow control technique we can send multiple frames at a time considering the window size. It is a packet based data transmission protocol. In Go Back n sliding window, we have copies of all transmitted frames until the receiver send an acknowledgement. Both acknowledgement and negative acknowledgement can be received by the sender during the transmission. ACK is a signal of successful data transmission sent by receiver so when we get ACK sender will send next frames considering the previous frames are successfully received by the receiver. But if there is any error or damage or lost in the frame, the receiver will send a NAK for a particular frame. So the rest of the frames of the window will be discorded. That time sender will resend the frames until it gets an ACK.

Now to check, if there is any damaged frame while transmitting data we will use **CRC-32**. CRC-32 is Cyclic Redundancy Check with 16 bit character length that is an [error-detecting code](https://en.wikipedia.org/wiki/Error_detection_and_correction) commonly used in digital [networks](https://en.wikipedia.org/wiki/Telecommunications_network) and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a [polynomial division](https://en.wikipedia.org/wiki/Polynomial_long_division) of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRC-32 is a polynomial function generator error checking technique where the value of parity is fixed.

Therefore, in our project we will use this two methods for sending and receiving data in digital networking and storage devices and data link layer protocols.

**Working Procedure:**

Transmit Frames using Go Back n Sliding Window ARQ

Receiver

Sender

Error

No Error

Receiver sends NAK to sender and sender resends the frames

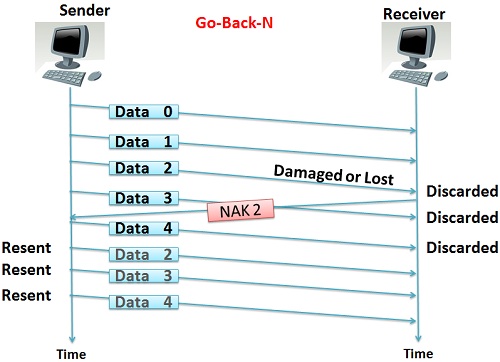
Receiver sends ACK to sender and sender sends next frames

**Fig: Flow chart of working procedure**

**Go Back n Sliding Window:**

1. When any damaged frame is received, receiver sends a negative acknowledgement frame (NAK) with the number of the frame which is damaged.
2. After receiving NAK from receiver, sender retransmits the frames.
3. When any frame is lost, receiving NAK the sender retransmits all the frames back from the frame of NAK.
4. But when the ACK is lost, then the sender again retransmits all the frames back since the last ACK is sent.
5. In this technique, the receiver does not have any sorting method so receiver discards all the duplicate frames that are retransmitted.

**Like-**

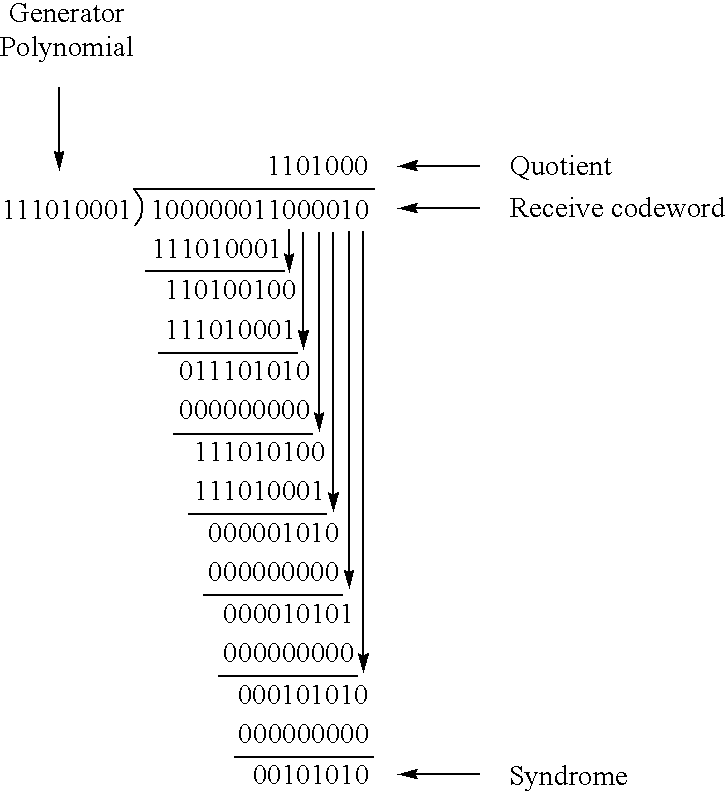
****

**Fig1: Go Back n Sliding Window**

**CRC – 32:**

1. As the value of divisor is fixed so we have to divide the frame first.
2. Then if Frame Check Sequence is not 0, then the data is corrupted. Error is present.
3. Now it will follow the Go Back n Sliding Window method to procced in the next step.
4. But if Frame Check sequence is 0, then the data is error free. So receiver will send an ACK to the sender.

**Like-**

****

**Fig: CRC implementation**

**Conclusion:**

In this project, we have implemented data sending and receiving with Go Back n Sliding Window method and detected the errors in CRC-32 technique successfully. This project was helpful to understand about the transmission of data in different protocols more comprehensively.

**Reference:**

1. <https://www.google.com/search?rlz=1C1CHBD_enBD742BD742&biw=1366&bih=623&tbm=isch&sa=1&q=crc+16+example&oq=crc+16+e&gs_l=psy-ab.3.0.0i24k1.461763.480114.0.481946.54.20.0.0.0.0.526.2931.1j13j1j1j0j1.17.0....0...1.1.64.psy-ab..43.8.1678.0..0j0i67k1j0i30k1j0i10i24k1.9HINM39dLSk#imgdii=fzLkhuThnz_haM:&imgrc=V0xnqEC8HepssM>
2. <https://www.google.com/search?rlz=1C1CHBD_enBD742BD742&biw=1366&bih=623&tbm=isch&sa=1&q=go+back+n+sliding+window+damage+frame+example&oq=go+back+n+sliding+window+damage+frame+example&gs_l=psy-ab.3...24299.28066.0.29903.13.13.0.0.0.0.143.1609.0j13.13.0....0...1.1.64.psy-ab..0.0.0.xMFnlwD2pMQ#imgrc=25lbz7VqoDKUDM>
3. <https://en.wikipedia.org/wiki/Go-Back-N_ARQ>
4. <https://en.wikipedia.org/wiki/Cyclic_redundancy_check>

**Code:**

#include<bits/stdc++.h>

#include<fstream>

using namespace std ;

int frame\_count,frame\_size,data\_size,frame\_data\_size=66,divisor\_size=33,buffer[3],buffer\_used=0,buffer\_size=3,frame\_s[100],lost=-1,alost=-1,inf=99999999999,ra;

char frame[100][100],divisor[33],frame\_r[100][100];

int count\_data\_lenght()

{

ofstream data;

data.open("data.txt");

for (int i=0 ; i <500; i++)

{

int a = rand() ;

a = a%2 ;

data << a << " " ;

}

// data << 1 << " " << 1<< " " << 0 ;

data.close();

ifstream data\_file;

data\_file.open("data.txt");

int counter=0;

int x ;

while(!data\_file.eof())

{

data\_file >> x ;

counter++ ;

}

data\_file.close();

data\_size = counter-1 ;

cout << "count\_data\_lenght check " << data\_size<<endl ;

return data\_size ;

}

int array\_size()

{

count\_data\_lenght();

frame\_count =(data\_size/frame\_data\_size) ;

//cout << "array size check " << frame\_count<< endl ;

return frame\_count ;}

void input\_from\_file( )

{

ifstream data\_file;

data\_file.open("data.txt");

array\_size() ;

int d =data\_size,n ;

for(int i=0 ; i<=frame\_count ; i++)

{

if(i%4==0)

{

frame[i][0] ='0' ;

frame[i][1] ='0' ;

}

if(i%4==1)

{

frame[i][0] ='0' ;

frame[i][1] ='1' ;

}

if(i%4==2)

{

frame[i][0] ='1' ;

frame[i][1] ='0' ;

}

if(i%4==3)

{

frame[i][0] ='1' ;

frame[i][1] ='1' ;

}

n=d;

if(d>frame\_data\_size)

n=frame\_data\_size;

//cout << n <<endl;

for(int j=1 ; j<=n ; j++)

{

char x ;

data\_file >> x ;

frame[i][j] = x ;

//cout << x <<endl ;

}

frame\_s[i]=n+32;

d = d-frame\_data\_size ;}

data\_file.close();}

void print\_data()

{

int d =data\_size,n ;

for(int i=0 ; i<=frame\_count ; i++)

{

cout << "frame " << i << ": " ;

// n=d;

//if(d>frame\_data\_size)

// n=frame\_data\_size;

for(int j=0 ; j<=99 ; j++)

{

cout << frame[i][j] ;

}

// d = d-frame\_data\_size ;

cout << endl;

}

}

void divisior\_generate()

{

for(int i=0 ; i<=divisor\_size ; i++)

{

if(i==0 || i==1|| i==2|| i==4 || i==6|| i==7 ||i==10 || i==11|| i==12|| i==16 || i==22|| i==23|| i==26||i==32)

{

divisor[32-i] = '1' ;

}

else

divisor[32-i] = '0';

}

cout << "divisor : ";

for(int i=0 ; i<=divisor\_size ; i++)

cout << divisor[i];

cout << endl;

}

int check\_error(char f[100],int s)

{

// cout << "checking : "<<endl;

char crc[100];

for(int i=0 ; i<=99 ; i++)

{

crc[i] = f[i] ;

}

int d =data\_size,n ;

for(int j=2; j<=frame\_data\_size+1 ; j++)

{

if(divisor[0]==crc[j])

{

for(int k=0 ; k<divisor\_size ; k++)

{

if(divisor[k]==crc[k+j])

{

crc[k+j] = '0';

// cout << "a";

}

else

{

crc[k+j] = '1' ;

}

}

}

}

for(int i=2; i<s; i++)

{

//cout <<crc[i] ;

if(crc[i]=='1')

return 1 ;

}

//cout <<endl;

return 0 ;

}

void crc\_genarte()

{

char crc[100],fcs[31];

int d =data\_size,n ;

for(int i=0 ; i<=frame\_count ; i++)

{

for(int i=0 ; i<=99 ; i++)

{

crc[i] = '0' ;

}

n=d;

if(d>frame\_data\_size)

n=frame\_data\_size;

for(int j=2 ; j<=n ; j++)

{

crc[j] = frame[i][j];}

d = d-frame\_data\_size ;

//cout << "frame " << i <<endl ;

//for(int j=1; j<=n+32 ; j++)

// cout << crc[j];

// cout<<endl ;

for(int j=2; j<=n+1 ; j++)

{

if(divisor[0]==crc[j])

{

for(int k=0 ; k<divisor\_size ; k++)

{

if(divisor[k]==crc[k+j])

{

crc[k+j] = '0';

// cout << "a";

}

else

{

crc[k+j] = '1' ;

// cout << "b" ;

}

}

}

// for(int j=1; j<=n+32 ; j++)

// cout << crc[j];

// cout<<endl ;

}

// for(int j=1; j<=n+32 ; j++)

// cout << crc[j];

// cout<<endl ;

int a =n+32;

for(int j=0; j<divisor\_size-1 ; j++)

{

frame[i][a] = crc [a] ;

a-- ;

}

//cout <<endl<<endl;

}

}

void transmitter()

{

cout << "\t\t\tGENARATING FRAME WITH FSC CODE " <<endl ;

input\_from\_file();

//cout << "BASIC INFO "

divisior\_generate();

crc\_genarte();

}

int reciver()

{

int i ,l=-1,j,flag=0;

for( i=0 ; i<buffer\_used ; i++)

{

cout << "frame " << buffer[i] ;

if(lost==buffer[i])

{

l=1 ;

j=i;

cout << " not recieved"<<endl;

continue;

}

else

{

cout << " recieved "<<endl;

}

}

if(i==buffer\_size&&l==-1)

cout << "buffer is full " <<endl ;

cout <<endl << "PROCESSING " <<endl ;

for(int i=0 ; i<buffer\_used ; i++)

{

cout << "frame " << buffer[i] ;

if(lost==buffer[i])

{

flag=1;

}

if(flag == 1 )

{

cout <<" discarded" ;

}

if(check\_error(frame\_r[buffer[i]],frame\_s[buffer[i]])==1 && flag==0&&l==-1)

{

cout << " error" ;

j = i ;

flag = 1 ;

}

if(flag==0)

cout << " no error" ;

cout <<endl ;

}

if(flag==1)

{

cout << "NACK " << buffer[j]<<endl; ;

return -buffer[j];

}

if(alost==1)

{

cout << "ACK lost " <<endl;

return inf ;

}

//cout << "asfdasf " <<buffer[i] <<endl ;

return buffer[i-1]+1;

}

void random(int f)

{

lost = -1 ;

alost = -1 ;

srand((int)time(NULL));

int x = rand();

//cout <<x <<endl;

x = x%4;

if (x==0) //0 no error

for(int i=0 ; i<=frame\_count ; i++)

for(int j=0 ; j<=frame\_s[i] ; j++)

frame\_r[i][j] = frame[i][j] ;

if(x==1) //1 error

{

//cout <<"x=1"<<endl;

for(int i=0 ; i<=frame\_count ; i++)

for(int j=0 ; j<=frame\_s[i] ; j++)

frame\_r[i][j] = frame[i][j] ;

int y=rand();

y = y%frame\_count;

if(y>=f)

{

int z=rand()\*rand();

z=z%frame\_data\_size ;

if(z>1)

{

// cout << "check "<<endl;

if(frame\_r[y][z] =='1')

frame\_r[y][z] = '0';

else

frame\_r[y][z] = '1' ;

}

// cout << "a";

}

}

if(x==2) //2 lost frame

{

for(int i=0 ; i<=frame\_count ; i++)

for(int j=0 ; j<=frame\_s[i] ; j++)

frame\_r[i][j] = frame[i][j] ;

int y=rand();

y = y%frame\_count;

if(y<=frame\_count)

{

lost=y;

// cout << lost <<endl;

}

}

if(x==3) //3 ack lost

{

for(int i=0 ; i<=frame\_count ; i++)

for(int j=0 ; j<=frame\_s[i] ; j++)

frame\_r[i][j] = frame[i][j] ;

alost = 1 ;

}

}

void transmission()

{

cout << "\t\t\tTRANSIMISSION" <<endl ;

int ACK=0;

while(ACK<=frame\_count)

{

if(ra==1)

random(ACK);

buffer\_used=0;

for(int i=0 ; i<buffer\_size ; i++)

{

if(ACK+i>frame\_count)

break ;

buffer[i] = ACK+i;

// cout << buffer[i] <<endl;

buffer\_used++;

}

int ack = reciver();

if(ack==inf)

{

ACK=ACK ;

}

if(ack>0&&ack<inf)

{

ACK=ack;

cout << "ACK " << ACK <<endl ;

}

if(ack<=0)

{

ack = ack\*-1 ;

ACK=ack;

}

cout << endl ;

}

cout << "\t\t\tTransmission complete "<<endl<<endl ;

}

int main()

{

ra=inf;

cout << "0=no error\t1=random error : " ;

cin >> ra ;

transmitter();

print\_data();

// cout << buffer[2] ;

transmission();

/\*for(int i=0; i<=6 ; i++)

{

cout << endl << "frame " << i ;

if(check\_error(frame[i],99)==1)

cout << "error"<<endl;

else

cout << "no error" <<endl ;

}\*/

}

**Output:**

