**EAST WEST UNIVERSITY**

**FALL 2019**

**Project Name:** Flow Control and Error Control Techniques in the Data Link Layer Protocols (Stop and Wait ARQ using CRC-32).

Course Code: CSE350

Course Title: Data Communication  
Section: 01

**Submitted by:**

Name: Rakibul Hassan RobinID: 2017-1-60-033

Name: Waliul Islam

ID: 2017-1-60-153

Name: Nasif Wasek Fahim

ID: 2017-1-60-037

**Submitted to:**

Dr. Maheen Islam

Assistant Professor

Department of Computer Science and Engineering

East West University

**Date of Submission:** 11 December, 2019

**Introduction:**

Stop-and-wait ARQ, also referred to as alternating bit protocol, is a method in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order. It is the simplest automatic repeat-request (ARQ) mechanism. A stop-and-wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with transmit and receive window sizes equal to one in both cases. After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal. After receiving a valid frame, the receiver sends an ACK.

If the ACK does not reach the sender before a certain time, known as the timeout, the sender sends the same frame again. The timeout countdown is reset after each frame transmission. The above behavior is a basic example of Stop-and-Wait. However, real-life implementations vary to address certain issues of design.

We are also using CRC-32 method to check if the received data is the same as the sender tried to send. Where data size will be long and the Divisor will be 32 bit long Binary code.

Typically the transmitter adds a redundancy check number to the end of each frame. The receiver uses the redundancy check number to check for possible damage. If the receiver sees that the frame is good, it sends an ACK. If the receiver sees that the frame is damaged, the receiver discards it and does not send an ACK—pretending that the frame was completely lost, not merely damaged.

One problem is when the ACK sent by the receiver is damaged or lost. In this case, the sender doesn't receive the ACK, times out, and sends the frame again. Now the receiver has two copies of the same frame, and doesn't know if the second one is a duplicate frame or the next frame of the sequence carrying identical DATA.

Another problem is when the transmission medium has such a long latency that the sender's timeout runs out before the frame reaches the receiver. In this case the sender resends the same packet. Eventually the receiver gets two copies of the same frame, and sends an ACK for each one. The sender, waiting for a single ACK, receives two ACKs, which may cause problems if it assumes that the second ACK is for the next frame in the sequence.

To avoid those issues, we are implementing the Stop and wait ARQ along with CRC-32 checksum.

**Protocol Description**

1. In stop-and-wait ARQ sender sends one frame at a time.
2. It waits for ACK then sends another frame, or resend if there’s any fault sending.
3. After receiving a valid frame, the receiver sends an ACK.
4. In receiver will data will be checked using (CRC-32) if the data is valid or not. After checking, receiver sends ACK if the data is OK.
5. If the ACK does not reach the sender before a certain time, time error will be shown and the frame will be resend.
6. Finally receiver will receive the data.

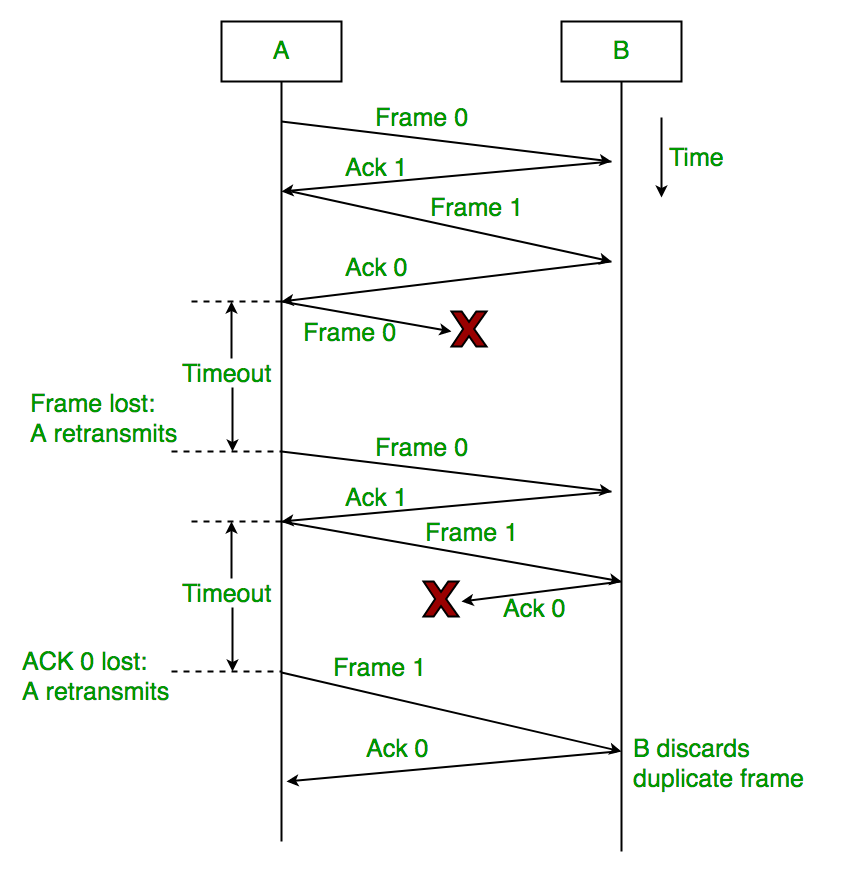


Image Reference: <https://www.geeksforgeeks.org/stop-and-wait-arq/>

**Code in c++ :**

using namespace std;

#include<bits/stdc++.h>

#include<windows.h>

#include<conio.h>

#include<time.h>

#define TIME 500

string errortry(string str);

void opening();

void display();

//Sender work

class sending{

string data;

string divisor = "10001111001100110010101011111110";

string frame;

int st = 0;

public:

void generate\_data(){

for(int i=0;i<1000;i++){

int random=rand()%2;

int f=random;

char c=f+'0';

this->data =data+c;

}

}

void CRCcheck(){

string temp;

int k = this->st;

while(this->st<(k+100)){

temp=temp+(this->data[st]);

this->st++;

}

frame=temp;

int d=this->divisor.size();

for(int i=0;i<d-1;i++){

temp=temp+'0';

}

for(int i=0; i<=temp.size()-d; i++){

for(int j=0; j<d; j++){

if(temp[i+j] == divisor[j])

temp[i+j] = '0';

else

temp[i+j] = '1';

}

while(i==temp.size() && temp[i]==1)

i++;

}

for(int i=temp.size()-d+1; i<temp.size(); i++)

frame += temp[i];

}

string getframe(){

return this->frame;

}

string getdata(){

return this->data;

}

void showdata(){

cout<<this->data<<endl<<endl;

}

};

//Receiver work

class receiving{

string data;

string divisor = "10001111001100110010101011111110";

string frame;

int ACK=0;

public:

int receive\_frame(int i, string f){

string temp;

if(i < ACK){

cout<<"\t\t\t\t\t\tDuplicate Data Received... ";

cout<<"Ignoring One..."<<endl;

Sleep(TIME);

cout<<"\t\t\t\t\t\tAcknowledgment sent for Frame: ";

cout<<ACK<<endl<<endl;

return ACK;

}

else{

this->frame = f;

temp = f;

}

int d = this->divisor.size();

for(int i=0; i<=temp.size()-d; i++){

for(int j=0; j<d; j++){

if(temp[i+j] == divisor[j])

temp[i+j] = '0';

else

temp[i+j] = '1';

}

while(i==temp.size() && temp[i]==1)

i++;

}

string t;

for(int i=temp.size()-d+1; i<temp.size(); i++)

t += temp[i];

for(int i=0;i<t.size();i++){

if(t[i]=='1'){

frame.empty();

cout<<"\t\t\t\t\t\tFrame: "<<ACK+1;

cout<<" is lost..."<<endl;

Sleep(TIME);

cout<<"\t\t\t\t\t\tResend the Data..."<<endl;

Sleep(TIME);

return -1;

}

}

int U=100,L=1;

int random = (rand() % (U-L+1))+L;

if(random<20){

ACK++;

cout<<"\t\t\t\t\t\tReceived..."<<endl;

cout<<"\t\t\t\t\t\tAcknowledgment sent for Frame: ";

cout<<ACK<<endl<<endl;

addData();

return 0;

}

else{

ACK++;

cout<<"\t\t\t\t\t\tReceived..."<<endl;

Sleep(TIME);

cout<<"\t\t\t\t\t\tAcknowledgment sent for Frame: ";

cout<<ACK<<endl<<endl;

addData();

return ACK;

}

}

void addData(){

string d;

for(int i=0;i<100;i++)

d += this->frame[i];

this->data += d;

return;

}

string getdata(){

return this->data;

}

void showdata(){

cout<<this->data<<endl<<endl;

}

};

//main

sending S;

receiving R;

int main(){

time\_t t;

srand(time(&t));

S.generate\_data();

opening();

cout<<"................................................................................................"<<endl<<endl;

cout<<"\tSENDING\t\t\t\t\t\t\t\t\t\tRECEIVING"<<endl;

cout<<"................................................................................................"<<endl;

for(int i=0;i<10;i++){

S.CRCcheck();

cout<<"................................................................................................"<<endl;

RES: string str = S.getframe();

cout<<" Sending Frame: "<<i+1<<endl;

for(int k=0;k<7;k++){

cout<<" . ";

Sleep(TIME);

}

cout<<endl;

string str2 = errortry(str);

int ACK = R.receive\_frame(i,str2);

if(ACK==-1){

Sleep(TIME);

cout<<" Trying to Resend..."<<endl<<endl;

goto RES;

}

else if(ACK==0){

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

cout<<" . "<<endl;

Sleep(4\*TIME);

}

cout<<" Timeout...!!!"<<endl;

Sleep(TIME);

cout<<" No Acknowledgment received for Frame: ";

cout<<i+1<<endl<<endl;

Sleep(TIME);

cout<<" Resending Data..."<<endl<<endl;

goto RES;

}

else{

Sleep(TIME);

cout<<" Acknowledgment received for Frame: ";

cout<<ACK<<endl<<endl;

cout<<"................................................................................................"<<endl;

}

Sleep(TIME);

}

cout<<endl<<endl<<" All Data Received Successfully...!!!";

cout<<endl;

system("pause");

system("cls");

display();

getch();

return 0;

}

string errortry(string str){

int U=115,L=100;

int random = (rand()%(U-L+1))+L;

str[random] = '0';

return str;

}

void opening(){

cout<<"\t\t\t\t ->Randomly Generated Data<-"<<endl<<endl;

S.showdata();

cout<<endl;

cout<<"Press Any Key to Start Sending the Data..."<<endl;

getch();

system("cls");

}

void display(){

cout<<"SENT DATA:"<<endl<<endl;

S.showdata();

cout<<endl;

cout<<"RECEIVED DATA:"<<endl<<endl;

R.showdata();

cout<<endl;

if(S.getdata()==R.getdata())

cout<<"No Error Found, Data received without error."<<endl;

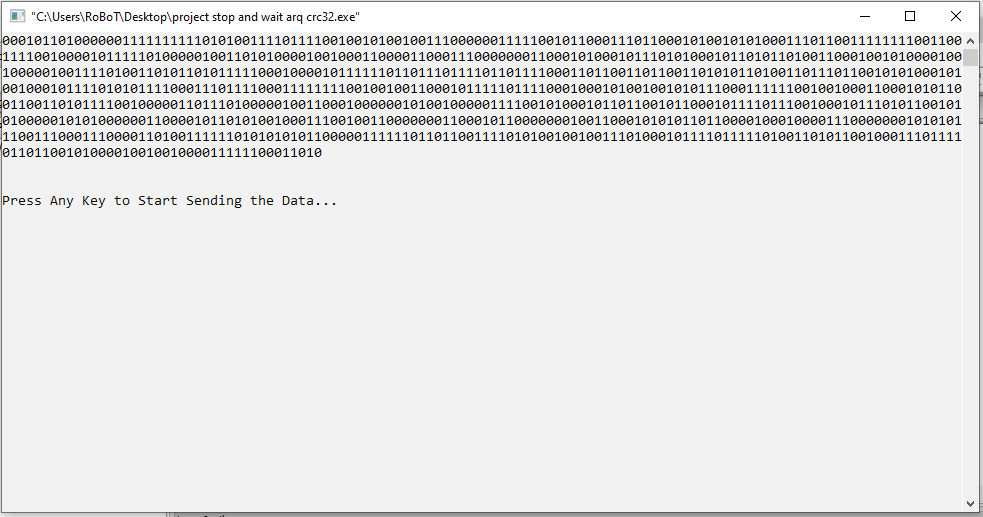
else

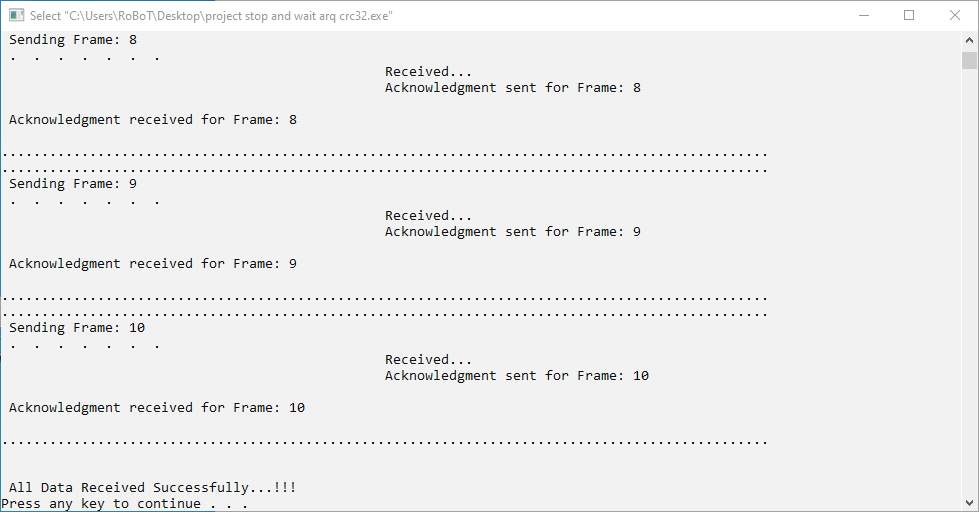
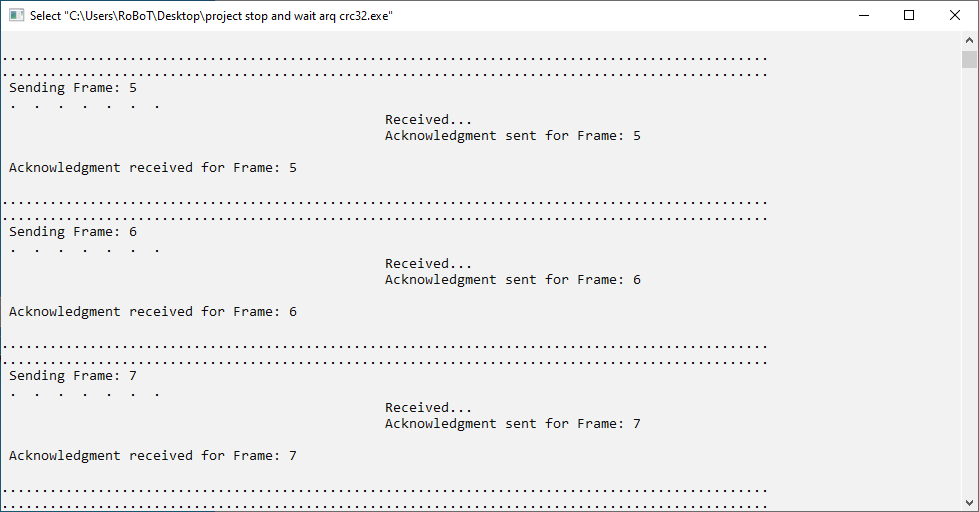
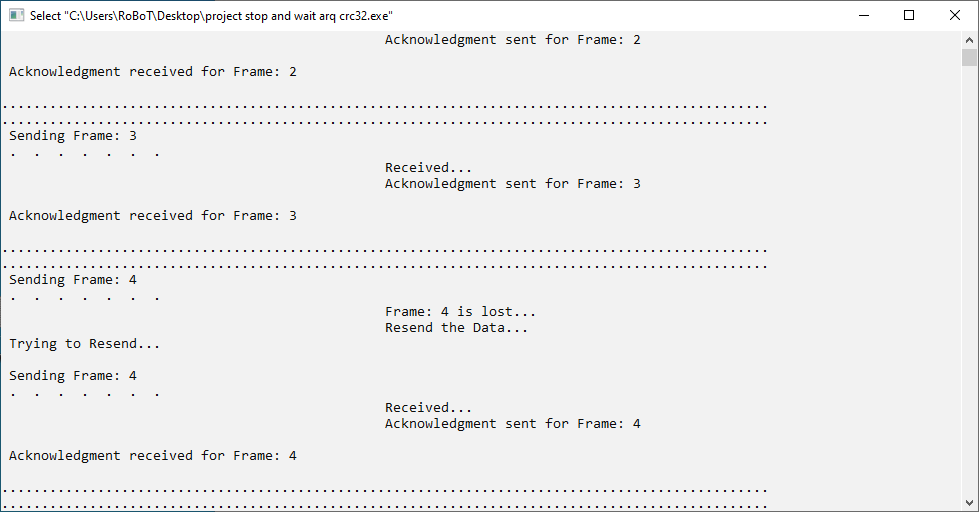
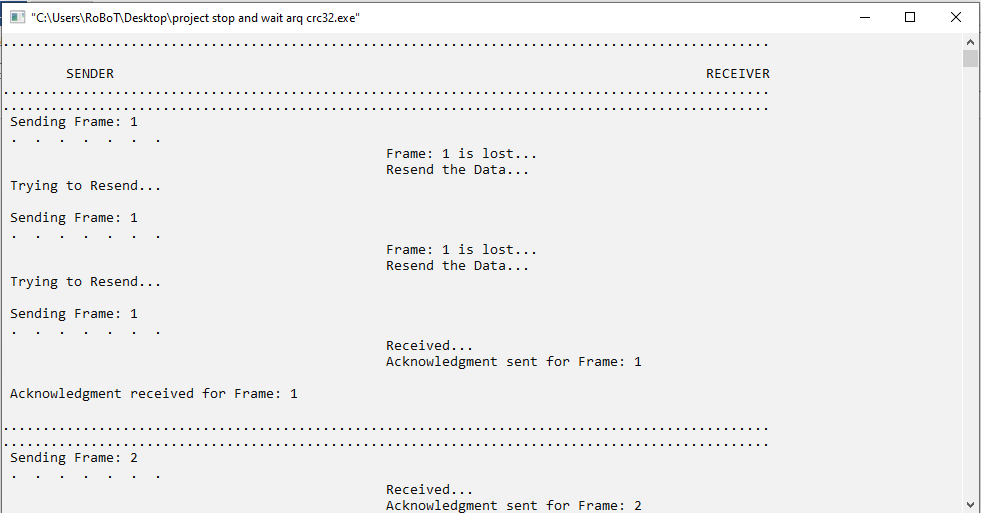
cout<<"Error Found!!"<<endl;

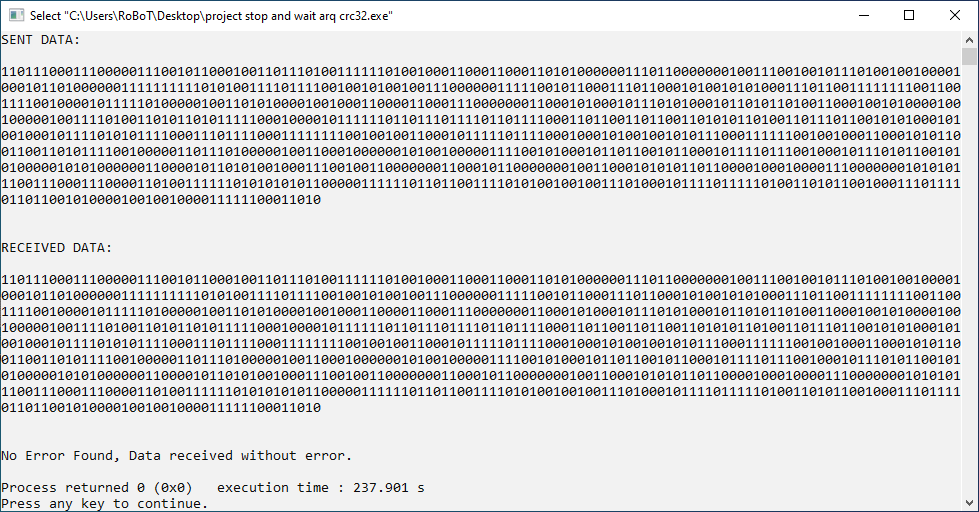
}

**Output:**

Data generated:



Transmission: ****

****

**Conclusion:**

In this project, we have implemented Stop and Wait ARQ method and detected the errors in CRC-32 technique successfully. This project helps us to learn a lot about data transmission from sender side, receiver side, transmission medium, CRC checksum and everything. In future it will help us to do more.