UNIVERSITY OF DELHI

Atma Ram Sanatan Dharma Collage

Computer Networking

Practical Code : BHCS07

Submitted By: Suraj Rai

Collage Roll no. : 21/18012

Submitted To : Ms Uma Ojha

Practical : 1

1.Simulate Cyclic Redundancy Check (CRC) error detection algorithm for channel.

Code:

```
#include <iostream>

#include<stdio.h>

using namespace std;
```

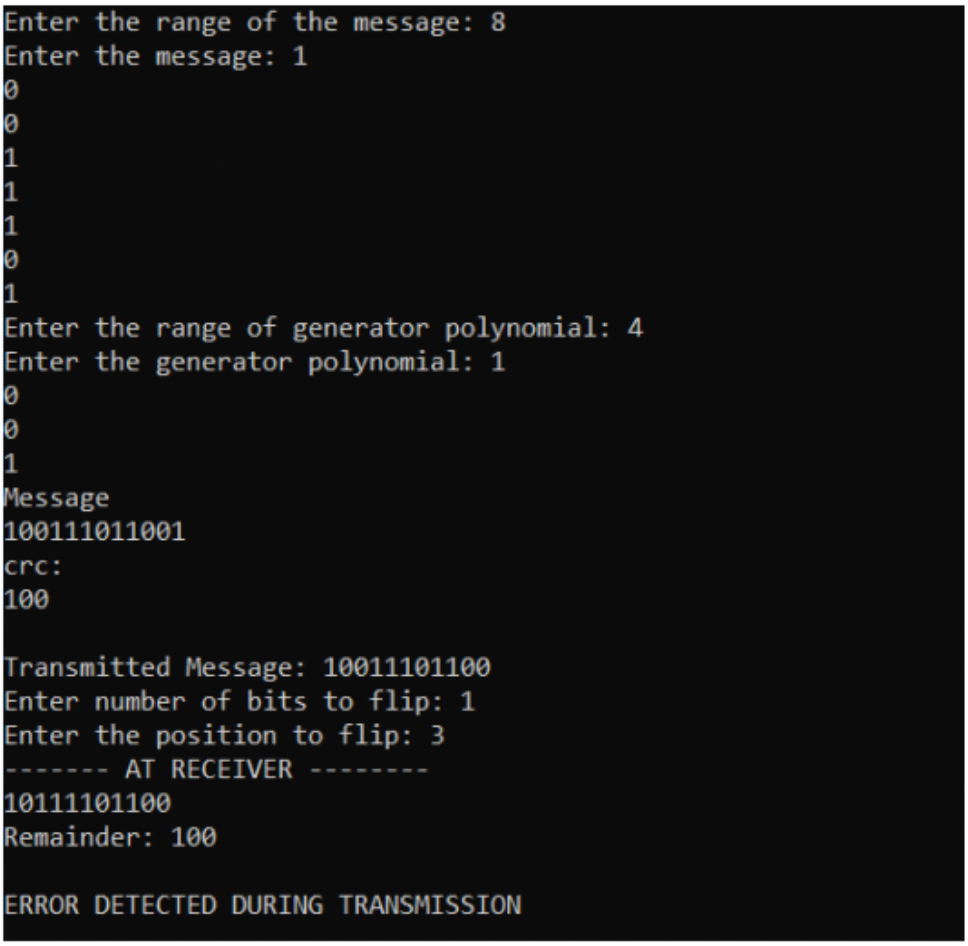
```
int main()
{
int msg[20],gr[20];
int m,n;
cout<<"Enter the range of the message: ";
cin>>n;
cout<<"Enter the message: ";
for(int i=0;i<n;i++)
cin>>msg[i];
cout<<"Enter the range of generator polynomial: ";
cin>>m;
cout<<"Enter the generator polynomial: ";
for(int i=0;i<m;i++)
cin>>gr[i];
cout<<"Message\n";
for(int i=0;i<n;i++)
 cout<<msg[i];
for(int i=0;i<m;i++)
 cout<<gr[i];
int codeword[n+(m-1)];
for(int i=0;i<n;i++)
 codeword[i]=msg[i];
for(int i=n;i<n+(m-1);i++)
 codeword[i]=0;
int temp[n+(m-1)];
for(int i=0;i<n+(m-1);i++)
 temp[i]=codeword[i];
for(int i=0;i<n;i++)
```

```
{
 int j=0,k=i;
 if(temp[k]>=gr[j])
 while(j<m)
 temp[k++]^=gr[j++];
}
int crc[20];
for(int i=0,j=n;i<(m-1);i++,j++)
 crc[i]=temp[j];
cout<<"\ncrc: \n";
for(int i=0;i<(m-1);i++)
 cout<<crc[i];
cout<<"\n";
for(int i=0,j=n;i<(m-1);i++,j++)
 codeword[j]=crc[i];
cout<<"\nTransmitted Message: ";
for(int i=0;i<n+(m-1);i++)
 cout<<codeword[i];
cout<<"\n";
//FOR NOISY CHANNEL
int nbits,pos;
cout<<"Enter number of bits to flip: ";
cin>>nbits;
for(int i=0;i<nbits;i++)
{
 cout<<"Enter the position to flip: ";
 cin>>pos;
 codeword[pos-1]=codeword[pos-1]==0?1:0;
```

```
}
cout<<"------- AT RECEIVER --------\n";
for(int i=0;i<n+(m-1);i++)
 cout<<codeword[i];
 cout<<"\n";
int temp2[n+(m-1)];
for(int i=0;i<n+(m-1);i++)
 temp2[i]=codeword[i];
for(int i=0;i<n;i++)
{
 int j=0,k=i;
 if(temp2[k]>=gr[j])
 while(j<m)
 temp2[k++]^=gr[j++];
}
int rem[20];
 for (int i = n, j = 0; i < n + (m - 1); i++, j++)
 rem[j] = temp2[i];
 cout << "Remainder: ";
 for (int i=0;i<(m-1);i++)
 cout<<rem[i];
 cout<<"\n";
 int flag = 0;
 for (int i = 0; i < (m - 1); i++)
 if (rem[i] != 0)
 flag = 1;
 cout <<"\n";
 if (flag==0)
```

```
 cout << "NO ERROR\n";

 else

 cout << "ERROR DETECTED DURING TRANSMISSION\n";

}
```

Output:



Practical : 2

2.Simulate and implement stop and wait protocol for noisy channel.

Code:

```
#include<iostream>
```

```
#include <time.h>
#include <cstdlib>
#include<ctime>
#include <unistd.h>
using namespace std;
class timer {
 private:
 unsigned long begTime;
 public:
 void start() {
 begTime = clock();
 }
 unsigned long elapsedTime() {
 return ((unsigned long) clock() - begTime) /
CLOCKS_PER_SEC;
 }
 bool isTimeout(unsigned long seconds) {
 return seconds >= elapsedTime();
 }
};
int main()
{
int frames[] = {1,2,3,4,5,6,7,8,9,10};
unsigned long seconds = 5;
srand(time(NULL));
timer t;
cout<<"Sender has to send frames : ";
for(int i=0;i<10;i++)
```

```
 cout<<frames[i]<<" ";
cout<<endl;
int count = 0;
bool delay = false;
cout<<endl<<"Sender\t\t\t\t\tReceiver"<<endl;
do
{
 bool timeout = false;
 cout<<"Sending Frame : "<<frames[count];
 cout.flush();
 cout<<"\t\t";
 t.start();
 if(rand()%2)
 {
 int to = 24600 + rand()%(64000 - 24600) + 1;
 for(int i=0;i<64000;i++)
 for(int j=0;j<to;j++) {}
 }
 if(t.elapsedTime() <= seconds)
 {
 cout<<"Received Frame : "<<frames[count]<<" ";
 if(delay)
 {
 cout<<"Duplicate";
 delay = false;
 }
 cout<<endl;
 count++;
```

```
}
else
{
cout<<"---"<<endl;
cout<<"Timeout"<<endl;
timeout = true;
}
t.start();
if(rand()%2 || !timeout)
{
int to = 24600 + rand()%(64000 - 24600) + 1;
for(int i=0;i<64000;i++)
for(int j=0;j<to;j++) {}
if(t.elapsedTime() > seconds )
{
cout<<"Delayed Ack"<<endl;
count--;
delay = true;
}
else if(!timeout)
cout<<"Acknowledgement : "<<frames[count]-1<<endl;
}
}while(count!=10);
return 0;
}
```

Output:

```
Sender has to send frames : 1 2 3 4 5 6 7 8 9 10

Sender                                  Receiver
Sending Frame : 1               Received Frame : 1
Delayed Ack
Sending Frame : 1               Received Frame : 1 Duplicate
Delayed Ack
Sending Frame : 1               ---
Timeout
Sending Frame : 1               Received Frame : 1 Duplicate
Delayed Ack
Sending Frame : 1               Received Frame : 1 Duplicate
Acknowledgement : 1
Sending Frame : 2               Received Frame : 2
Delayed Ack
Sending Frame : 2               ---
Timeout
Delayed Ack
Sending Frame : 1               Received Frame : 1 Duplicate
Acknowledgement : 1
Sending Frame : 2               ---
Timeout
Sending Frame : 2               Received Frame : 2
Acknowledgement : 2
Sending Frame : 3               Received Frame : 3
Acknowledgement : 3
Sending Frame : 4               Received Frame : 4
Delayed Ack
Sending Frame : 4               Received Frame : 4 Duplicate
Acknowledgement : 4
Sending Frame : 5               Received Frame : 5
Acknowledgement : 5
Sending Frame : 6               Received Frame : 6
Acknowledgement : 6
Sending Frame : 7               Received Frame : 7
Delayed Ack
Sending Frame : 7               Received Frame : 7 Duplicate
Acknowledgement : 7
Sending Frame : 8               Received Frame : 8
Acknowledgement : 8
Sending Frame : 9               Received Frame : 9
Delayed Ack
Sending Frame : 9               Received Frame : 9 Duplicate
Acknowledgement : 9
Sending Frame : 10              Received Frame : 10
Delayed Ack
Sending Frame : 10              Received Frame : 10 Duplicate
Acknowledgement : 47734

Process returned 0 (0x0)   execution time : 177.609 s
```

Prcatical: 4

3.Simulate and implement selective repeat sliding window protocol.

Code:

```
#include<iostream>
#include<conio.h>
#include<time.h>
#include<math.h>
#define TOT_FRAMES 500
#define FRAMES_SEND 10
using namespace std;
class SelRepeat
{
private:
 int frames;
 int arr[TOT_FRAMES];
 int send[FRAMES_SEND];
 int rcvd[FRAMES_SEND];
 char rcvd_ack[FRAMES_SEND];
 int sw;
 int rw;
public:
void input()
{
 int m,n,i;
 cout<<"Enter the number of bits: ";
 cin>>n;
 m=pow(2,n);
 int t=0;
frames=(m/2);
```

```
for(i=0;i<TOT_FRAMES;i++)
{
arr[i]=t;
t=(t+1)%m;
}
for(i=0;i<frames;i++)
{
send[i]=arr[i];
rcvd[i]=arr[i];
rcvd_ack[i]='n';
}
rw=sw=frames;
sender(m);
}
void sender(int m)
{
for(int i=0;i<frames;i++)
{
if(rcvd_ack[i]=='n')
cout<<"SENDER : Frame "<<send[i]<<" is sent\n";
}
receiver(m);
}
void receiver(int m)
{
time_t t;
int f;
int j;
```

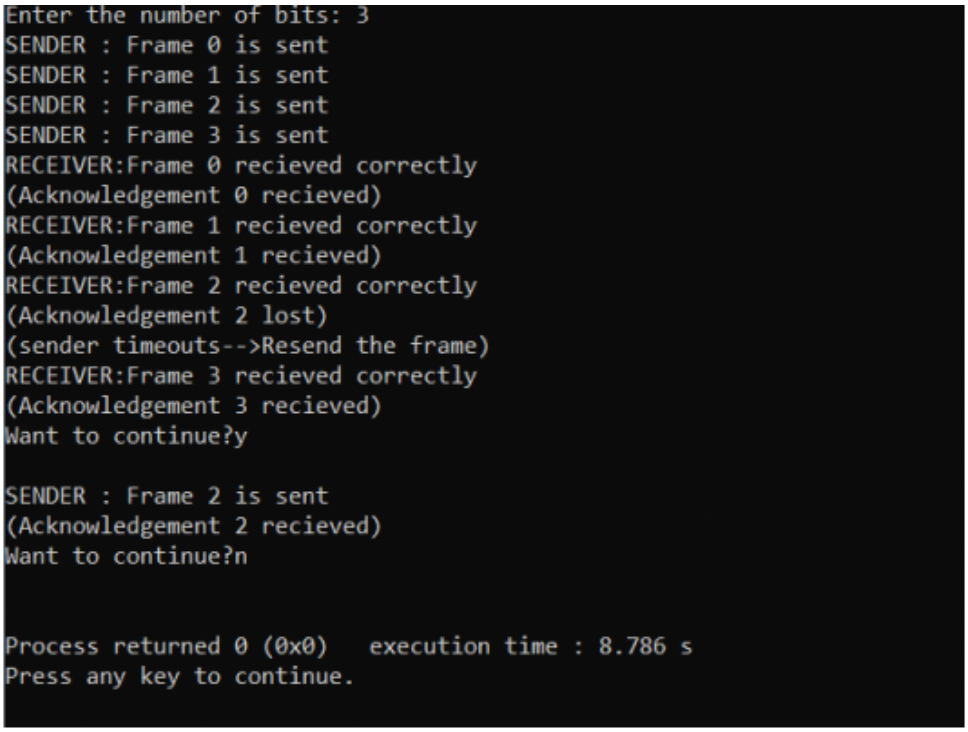
```
int f1;
int a1;
char ch;
srand((unsigned)time(&t));
for(int i=0;i<frames;i++)
{
if(rcvd_ack[i]=='n')
{
f=rand()%10;
if(f!=5)//if f=5 frame is discarded for some reason
 //else frame is correctly recieved
{
for(int j=0;j<frames;j++)
if(rcvd[j]==send[i])
{
cout<<"RECEIVER:Frame "<<rcvd[j]<<" recieved correctly\n";
rcvd[j]=arr[rw];
rw=(rw+1)%m;
break;
}
int j;
if(j==frames)
cout<<"RECEIVER:Duplicate Frame "<<send[i]<<"discarded\n";
a1=rand()%5;
if(a1==3)//if al==3 then ack is lost
//else recieved
{
cout<<"(Acknowledgement "<<send[i]<<" lost)\n";
```

```
cout<<"(sender timeouts-->Resend the frame)\n";
rcvd_ack[i]='n';
}
else
{
cout<<"(Acknowledgement "<<send[i]<<" recieved)\n";
rcvd_ack[i]='p';
}
}
else
{int ld=rand()%2;
//if =0 then frame damaged
//else frame lost
if(ld==0)
{
cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";
cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<"
sent\n";
}
else
{
cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";
cout<<"SENDER TIMEOUT-->RESEND THE FRAME\n";
}
rcvd_ack[i]='n';
}
}
}
```

```
for(int j=0;j<frames;j++)
{
if(rcvd_ack[j]=='n')
break;
}
int i=0;
for(int k=j;k<frames;k++)
{
send[i]=send[k];
if(rcvd_ack[k]=='n')
 rcvd_ack[i]='n';
else
 rcvd_ack[i]='p';
i++;
}
if(i!=frames)
{
for(int k=i;k<frames;k++)
 {
 send[k]=arr[sw];
 sw=(sw+1)%m;
 rcvd_ack[k]='n';
 }
}
cout<<"Want to continue?";
cin>>ch;
cout<<"\n";
if(ch=='y')
```

```
 sender(m);

else

 exit(0);

}};

int main()

{

SelRepeat obj;;

obj.input();

}
```

Output:



Practical : 5

4. Simulate and implement distance vector routing algorithm.

Code:

```
#include<iostream>
```

```
#include<conio.h>
using namespace std;
struct node
{
 unsigned dist[20];
 unsigned from[20];
}dvr[10];
int main()
{
 int cost[20][20];
 int i,j,k,nodes,count=0;
 cout<<"\nEnter the number of nodes: ";
 cin>>nodes;
 cout<<"\nEnter the cost matrix: \n";
 for(i=0;i<nodes;i++)
 {
 for(j=0;j<nodes;j++)
 {
 cin>>cost[i][j];
 cost[i][i]=0;
 dvr[i].dist[j]=cost[i][j];//initializing distance
equal to cost matrix
 dvr[i].from[j]=j;
 }
 }
 do
 {
 count=0;
```

```
 for(i=0;i<nodes;i++)
 for(j=0;j<nodes;j++)
 for(k=0;k<nodes;k++)
 if(dvr[i].dist[j]>cost[i][k]+dvr[k].dist[j])
 {//calculate the minimum distance
dvr[i].dist[j]=dvr[i].dist[k]+dvr[k].dist[j];
 dvr[i].from[j]=k;
 count++;
 }
 }while(count!=0);
 for(i=0;i<nodes;i++)
 {
 cout<<"\nFor router: "<<i+1;
 for(j=0;j<nodes;j++)
 {
 cout<<"\t\n node "<<j+1<<" via
"<<dvr[i].from[j]+1<<" Distance "<<dvr[i].dist[j];
 }
 }
 cout<<endl;
 getch();
}
```

Output:

```
Enter the number of nodes: 3

Enter the cost matrix:
0
2
7
2
0
1
7
1
0

For router: 1
 node 1 via 1 Distance 0
 node 2 via 2 Distance 2
 node 3 via 2 Distance 3
For router: 2
 node 1 via 1 Distance 2
 node 2 via 2 Distance 0
 node 3 via 3 Distance 1
For router: 3
 node 1 via 2 Distance 3
 node 2 via 2 Distance 1
 node 3 via 3 Distance 0
```

Practical : 5

5. Simulate and implement Dijkstra algorithm for shortest path routing.

Code:

```
#include <iostream>

#include <iomanip>

#define MAX_NODES 20

using namespace std;

class Set

{

public:
```

```
int edge;
int vertex;
int path[MAX_NODES];
int dist[MAX_NODES];
int adjMatrix[MAX_NODES][MAX_NODES];
void input(int v, int e)
{
edge = e;
vertex = v;
for (int i = 0; i < v; i++)// initializing the adjacency
matrix
for (int j = 0; j < v; j++)
adjMatrix[i][j] = 0;
int src, dest, weight;
for (int i = 0; i < edge; i++)
{
cout << "\nEDGE " << (i + 1)
<< "\n------\n";
cout << "Enter Source: ";
cin >> src;
cout << "Enter Destination: ";
cin >> dest;
cout << "Enter Weight: ";
cin >> weight;
adjMatrix[src - 1][dest - 1] = weight;
adjMatrix[dest - 1][src - 1] = weight;
}
}
```

```
void display()
{
for (int i = 0; i < vertex; i++)
{
for (int j = 0; j < vertex; j++)
cout << setw(5) << adjMatrix[i][j] << " ";
cout <<"\n";
}
}
void dijkstra(int src)
{
bool visited[MAX_NODES];
for (int i = 0; i < vertex; i++)
{
visited[i] = false;
dist[i] = INT_MAX;
}
path[src] = -1;//source node
dist[src] = 0;
for (int i = 0; i < vertex- 1; i++)
{
int u = minDist(visited);//nearest node
visited[u] = true;
for (int v = 0; v < vertex; v++)
if (visited[v] == false && adjMatrix[u][v] && dist[u] !=
INT_MAX && dist[u] + adjMatrix[u][v] < dist[v])
{
path[v] = u;
```

```
dist[v] = dist[u] + adjMatrix[u][v];
}
}
cout << "\nDestn Node \t Distance \t Shortest
Path";//displaying
cout << "\n---------- \t -------- \t ----------";
for (int i = 0; i < vertex; i++)
{
cout <<"\n"<< (i + 1)<< " \t\t " << dist[i]<< " \t\t " <<
(src + 1);
printShortestPath(i);
}
}
int minDist(bool *visited)
{
int min = INT_MAX, min_index;
for (int v = 0; v < vertex; v++)
if (visited[v] == false && dist[v] <= min)
{
min = dist[v];
min_index = v;
}
return min_index;
}
void printShortestPath(int node)
{
if (path[node] == -1)
return;
```

```
 printShortestPath(path[node]);
 cout << " -> " << (node + 1);
 }
};
int main()
{
 int ver, ed;
 Set s;
 cout << "Enter total number of Nodes: ";
 cin >> ver;
 cout << "Enter number of Edges: ";
 cin >> ed;
 s.input(ver, ed);
 cout<<"\nGRAPH\n";
 cout<<"--------\n";
 s.display();
 cout <<"\n";
 cout << "Enter Source Node: ";
 cin >> ver;
 s.dijkstra(ver - 1);
 return 0;
}
```

Output:

```
Enter total number of Nodes: 4
Enter number of Edges: 5

EDGE 1
------
Enter Source: 1
Enter Destination: 2
Enter Weight: 10

EDGE 2
------
Enter Source: 1
Enter Destination: 3
Enter Weight: 2

EDGE 3
------
Enter Source: 1
Enter Destination: 5
Enter Weight: 100

EDGE 4
------
Enter Source: 2
Enter Destination: 4
Enter Weight: 3

EDGE 5
------
Enter Source: 3
Enter Destination: 2
Enter Weight: 5

GRAPH
--------
    0    10     2     0
   10     0     5     3
    2     5     0     0
    0     3     0     0

Enter Source Node: 1

Destn Node       Distance        Shortest Path
----------       --------        ----------
1                0               1
2                7               1 -> 3 -> 2
3                2               1 -> 3
4                10              1 -> 3 -> 2 -> 4
Process returned 0 (0x0)   execution time : 109.897 s
Press any key to continue.
```

Practical : 3

6.simulate and emplement Go-Back sliding window protocol.

code:

```
#include<bits/stdc++.h>
#include<ctime>
#define ll long long int
using namespace std;
void transmission(ll & i, ll & N, ll & tf, ll & tt) {
  while (i <= tf) {
    int z = 0;
    for (int k = i; k < i + N && k <= tf; k++) {
      cout << "Sending Frame " << k << "..." << endl;
      tt++;
    }
    for (int k = i; k < i + N && k <= tf; k++) {
      int f = rand() % 2;
      if (!f) {
        cout << "Acknowledgment for Frame " << k << "..." << endl;
        z++;
      } else {
        cout << "Timeout!! Frame Number : " << k << " Not Received" << endl;
        cout << "Retransmitting Window..." << endl;
        break;
      }
    }
    cout << "\n";
    i = i + z;
  }
}
int main() {
  ll tf, N, tt = 0;
```

```
  srand(time(NULL));
  cout << "Enter the Total number of frames : ";
  cin >> tf;
  cout << "Enter the Window Size : ";
  cin >> N;
  ll i = 1;
  transmission(i, N, tf, tt);
  cout << "Total number of frames which were sent and resent are : " << tt <<
    endl;
  return 0;
}
```

output :

Enter the Total number of frames : 12
Enter the Window Size : 4
Sending Frame 1...
Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Timeout!! Frame Number : 1 Not Received
Retransmitting Window...

Sending Frame 1...
Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Acknowledgment for Frame 1...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...

Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...

Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Acknowledgment for Frame 2...
Acknowledgment for Frame 3...
Acknowledgment for Frame 4...
Timeout!! Frame Number : 5 Not Received
Retransmitting Window...

Sending Frame 5...
Sending Frame 6...
Sending Frame 7...
Sending Frame 8...
Timeout!! Frame Number : 5 Not Received
Retransmitting Window...

Sending Frame 5...
Sending Frame 6...
Sending Frame 7...
Sending Frame 8...
Acknowledgment for Frame 5...
Timeout!! Frame Number : 6 Not Received
Retransmitting Window...

Sending Frame 6...
Sending Frame 7...
Sending Frame 8...
Sending Frame 9...
Acknowledgment for Frame 6...
Timeout!! Frame Number : 7 Not Received
Retransmitting Window...

Sending Frame 7...
Sending Frame 8...
Sending Frame 9...
Sending Frame 10...
Acknowledgment for Frame 7...
Acknowledgment for Frame 8...
Acknowledgment for Frame 9...
Acknowledgment for Frame 10...

Sending Frame 11...
Sending Frame 12...
Timeout!! Frame Number : 11 Not Received
Retransmitting Window...

Sending Frame 11...
Sending Frame 12...
Timeout!! Frame Number : 11 Not Received
Retransmitting Window...

Sending Frame 11...
Sending Frame 12...
Acknowledgment for Frame 11...
Acknowledgment for Frame 12...

Total number of frames transmitted(sent + resent) are : 38