VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Lab Assessment-III, SEPTEMBER 2020

B.Tech., Fall-2020-2021

NAME	PRIYAL BHARDWAJ
REG. NO.	18BIT0272
COURSE CODE	ITE3001
COURSE NAME	DATA COMMUNICATION & COMPUTER NETWORKS
SLOT	L15+L16
FACULTY	Prof. DINAKARAN MURUGANANDAM

LA 2 - last question:

Assume that you are following the Hamming code mechanism for data transfer. Consider the dataword is 1011, find the codeword.

CODE:

```
#include<iostream>
#include<cmath>
#include<string>
using namespace std;
 class Hamming
{
 string message;
 int codeword[50],temp[50];
 int n,check;
 char parity;
public:
 Hamming()
 {
     parity = 'E';
     message = "";
     n=check=0;
     for(int i=0;i<50;i++)
     {
         temp[i]=codeword[i]=0;
     }
 }

 void generate()
 {
     do
     {
         cout<<"Enter the message in binary : ";
         cin>>message;
     }while(message.find_first_not_of("01") != string::npos);
     n=message.size();
     cout<<"Odd(O)/Even(E) Parity ? ";
     cin>>parity;
     for(unsigned int i=0;i<message.size();i++)
     {
         if(message[i] == '1')
             temp[i+1]=1;
         else
             temp[i+1]=0;
     }
     computeCode();
 }
```

```
void computeCode()
 {
     check = findr();
     cout<<"Number of Check Bits : "<<check<<endl;
     cout<<"Number of Bits in Codeword : "<<n+check<<endl;
     for(int i=(n+check),j=n;i>0;i--)
     {
         if((i & (i - 1)) != 0)
             codeword[i] = temp[j--];
         else
             codeword[i] = setParity(i);
     }
     cout<<"Parity Bits - ";
     for(int i=0;i<check;i++)
      cout<<"P"<<pow(2,i)<<" : "<<codeword[(int)pow(2,i)]<<"\t";
     cout<<endl;
     cout<<"Codeword :"<<endl;
     for(int i=1;i<=(n+check);i++)
         cout<<codeword[i]<<" ";
     cout<<endl;
 }

int findr()
 {
     for(int i=1;;i++)
     {
         if(n+i+1 <= pow(2,i))
             return i;
     } }

int setParity(int x)
 {
     bool flag = true;
     int bit;
     if(x == 1)
     {
         bit = codeword[x+2];
         for(int j=x+3;j<=(n+check);j++)
         {
             if(j%2)
             {
                 bit ^= codeword[j];
             }
         }
     }
     else
     {
         bit = codeword[x+1];
```

```
        for(int i=x;i<=(n+check);i++)
        {
            if(flag)
            {
                if(i==x || i==x+1)
                    bit = codeword[x+1];
                else
                    bit ^= codeword[i];
            }
            if((i+1)%x == 0)
                flag = !flag;
        }
    }
    if(parity == 'O' || parity == 'o')
        return !bit;
    else
        return bit;
}

void correct()
{
    do
    {
        cout<<"Enter the received codeword : ";
        cin>>message;
    }while(message.find_first_not_of("01") != string::npos);
    for(unsigned int i=0;i<message.size();i++)
    {
        if(message[i] == '1')
            codeword[i+1]=1;
        else
            codeword[i+1]=0;
    }
    detect();
}

void detect()
{
    int position = 0;
    cout<<"Parity Bits - ";
    for(int i=0;i<check;i++)
    {
        bool flag = true;
        int x = pow(2,i);
        int bit = codeword[x];
        if(x == 1)
        {
            for(int j=x+1;j<=(n+check);j++)
```

```
            {
                if(j%2)
                {
                    bit ^= codeword[j];
                }
            }
        }
        else
        {
            for(int k=x+1;k<=(n+check);k++)
            {
                if(flag)
                {
                    bit ^= codeword[k];
                }
                if((k+1)%x == 0)
                    flag = !flag;
            }
        }
        cout<<"P"<<x<<": "<<bit<<"\t";
        if((parity=='E' || parity == 'e') && bit==1)
            position += x;
        if((parity=='O' || parity == 'o') && bit==0)
            position += x;
    }
    cout<<endl<<"Received Codeword :"<<endl;
    for(int i=1;i<=(n+check);i++)
        cout<<codeword[i]<<" ";
    cout<<endl;
    if(position != 0)
    {
        cout<<"Error at bit : "<<position<<endl;
        codeword[position] = !codeword[position];
        cout<<"Corrected Codeword : "<<endl;
        for(int i=1;i<=(n+check);i++)
            cout<<codeword[i]<<" ";
        cout<<endl;
    }
    else
        cout<<"No Error in Received code."<<endl;
    cout<<"Received Message is : ";
    for(int i=1;i<=(n+check);i++)
        if((i & (i - 1)) != 0)
            cout<<codeword[i]<<" ";
    cout<<endl;
 }
};
```

```
int main()
{
 char choice;
 do
 {
     Hamming a;
     cout<<"18BIT0272 - PRIYAL BHARDWAJ"<<endl;
     cout<<"At Sender's side : "<<endl;
     a.generate();
     cout<<endl<<"At Receiver's Side : "<<endl;
     a.correct();
     cout<<endl<<"Enter another code ? (Y/N) : ";
     cin>>choice;
     cout<<endl;
 }while(choice == 'y' || choice == 'Y');
 return 0;
}
```

OUTPUT:

```
"C:\Users\PRIYAL BHARDWAJ\Downloads\dccn2.exe"
18BIT0272 - PRIYAL BHARDWAJ
At Sender's side :
Enter the message in binary : 1011
Odd(O)/Even(E) Parity ? E
Number of Check Bits : 3
Number of Bits in Codeword : 7
Parity Bits - P1 : 0    P2 : 1  P4 : 0
Codeword :
0 1 1 0 0 1 1

At Receiver's Side :
Enter the received codeword : 0110011
Parity Bits - P1: 0     P2: 0   P4: 0
Received Codeword :
0 1 1 0 0 1 1
No Error in Received code.
Received Message is : 1 0 1 1

Enter another code ? (Y/N) : N


Process returned 0 (0x0)   execution time : 84.517 s
```

LA 3 Questions:

1. Implement the following Flow Control Mechanisms using any programming language

Consider a single program for implementing sender and receiver as separate functions.

a. Stop and Wait ARQ Protocol

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define TIMEOUT 5
#define MAX_SEQ 1
#define TOT_PACKETS 10
#define inc(k) if(k<MAX_SEQ) k++; else k=0;
typedef struct
{
int data;
}packet;
typedef struct
{
int kind;
int seq;
int ack;
packet info;
int err;
}frame;
frame DATA;
typedef enum{frame_arrival,err,timeout,no_event} event_type;
void from_network_layer(packet *);
void to_network_layer(packet *);
void to_physical_layer(frame *);
void from_physical_layer(frame *);
void wait_for_event_sender(event_type *);
void wait_for_event_reciever(event_type *);
void reciever();
void sender();
int i=1; //Data to be sent by sender
char turn;
int DISCONNECT=0;
int main()
{
printf("18BIT0231 \n\n");
srand(time(NULL));
while(!DISCONNECT)
{
sender();
usleep(4);
reciever();
}
```

```
return 0;
}
void sender()
{
static int frame_to_send=0;
static frame s;
packet buffer;
event_type event;
static int flag=0;
if(flag==0)
{
from_network_layer(&buffer);
s.info = buffer;
s.seq = frame_to_send;
printf("SENDER : Info = %d Sequence No = %d ",s.info,s.seq);
turn = 'r';
to_physical_layer(&s);
flag = 1;
}
wait_for_event_sender(&event);
if(turn=='s')
{
if(event==frame_arrival)
{
from_network_layer(&buffer);
inc(frame_to_send);
s.info = buffer;
s.seq = frame_to_send;
printf("SENDER : Info = %d Sequence No = %d ",s.info,s.seq);
turn = 'r';
to_physical_layer(&s);
}
if(event==timeout)
{
printf("SENDER : Resending Frame ");
turn = 'r';
to_physical_layer(&s);
} } }
void reciever()
{
static int frame_expected=0;
frame r,s;
event_type event;
wait_for_event_reciever(&event);
if(turn=='r')
{
if(event==frame_arrival)
{
```

```
from_physical_layer(&r);
if(r.seq==frame_expected)
{
to_network_layer(&r.info);
inc(frame_expected);
}
else
printf("RECIEVER : Acknowledgement Resent\n");
turn = 's';
to_physical_layer(&s);
}
if(event==err)
{
printf("RECIEVER : Garbled Frame\n");
turn = 's'; //if frame not recieved
}}}
void from_network_layer(packet *buffer)
{
(*buffer).data = i;
i++;
}
void to_physical_layer(frame *s)
{ // 0 means error
s->err = rand(); //non zero means no error
DATA = *s; //probability of error = 1/4
}
void to_network_layer(packet *buffer)
{
printf("RECIEVER :Packet %d received , Acknowledgement Sent\n",(*buffer).data)
;
if(i>TOT_PACKETS) //if all packets received then disconnect
{
DISCONNECT = 1;
printf("\nDISCONNECTED");
}}
void from_physical_layer(frame *buffer)
{
*buffer = DATA;
}
void wait_for_event_sender(event_type * e)
{
static int timer=0;
if(turn=='s')
{
timer++;
if(timer==TIMEOUT)
{
*e = timeout;
```

```
printf("SENDER : Acknowledgement not received => TIMEOUT\n");
timer = 0;
return;
}
if(DATA.err==0)
*e = err;
else
{
timer = 0;
*e = frame_arrival;
}}}
void wait_for_event_reciever(event_type * e)
{
if(turn=='r')
{
if(DATA.err==0)
*e = err;
else
*e = frame_arrival;
}}
```

OUTPUT:

```
"C:\Users\PRIYAL BHARDWAJ\Downloads\dccn31.exe"
18BIT0272 - PRIYAL BHARDWAJ

Sender : Info = 1 Sequence No = 0 Receiver :Packet 1 received , Acknowledgement Sent
SENDER : Info = 2 Sequence No = 1 Receiver :Packet 2 received , Acknowledgement Sent
SENDER : Info = 3 Sequence No = 0 Receiver :Packet 3 received , Acknowledgement Sent
SENDER : Info = 4 Sequence No = 1 Receiver :Packet 4 received , Acknowledgement Sent
SENDER : Info = 5 Sequence No = 0 Receiver :Packet 5 received , Acknowledgement Sent
SENDER : Info = 6 Sequence No = 1 Receiver :Packet 6 received , Acknowledgement Sent
SENDER : Info = 7 Sequence No = 0 Receiver :Packet 7 received , Acknowledgement Sent
SENDER : Info = 8 Sequence No = 1 Receiver :Packet 8 received , Acknowledgement Sent
SENDER : Info = 9 Sequence No = 0 Receiver :Packet 9 received , Acknowledgement Sent
SENDER : Info = 10 Sequence No = 1 Receiver :Packet 10 received , Acknowledgement Sent

Disconnected
Process returned 0 (0x0)   execution time : 2.573 s
```

b. Go Back N ARQ Protocol

CODE:

```
#include<iostream>
#include<ctime>
#include<cstdlib>
using namespace std;
int main()
{
cout<<"18BIT0272 - PRIYAL BHARDWAJ"<<endl;
int nf,N;
```

```
int no_tr=0;
srand(time(NULL));
cout<<"Enter the number of frames: ";
cin>>nf;
cout<<"Enter the Window Size: ";
cin>>N;
int i=1;
while(i<=nf)
{
 int x=0;
 for(int j=i;j<i+N && j<=nf;j++)
 {
 cout<<"Sent Frame "<<j<<endl;
 no_tr++;
 }
 for(int j=i;j<i+N && j<=nf;j++)
 {
 int flag = rand()%2;
 if(!flag)
 {
 cout<<"Acknowledgment for Frame "<<j<<endl;
 x++;
 }
 else
 { cout<<"Frame "<<j<<" Not Received"<<endl;
 cout<<"Retransmitting Window"<<endl;
 break;
 }
 }
 cout<<endl;
 i+=x;
}
cout<<"Total number of transmissions: "<<no_tr<<endl;
return 0;
}
```

OUTPUT:

PTO

```
"C:\Users\PRIYAL BHARDWAJ\Downloads\dccn32.exe"
18BIT0272 - PRIYAL BHARDWAJ
Enter the number of frames: 4
Enter the Window Size: 2
Sent Frame 1
Sent Frame 2
Acknowledgment for Frame 1
Acknowledgment for Frame 2

Sent Frame 3
Sent Frame 4
Frame 3 Not Received
Retransmitting Window

Sent Frame 3
Sent Frame 4
Acknowledgment for Frame 3
Acknowledgment for Frame 4

Total number of transmissions: 6

Process returned 0 (0x0)   execution time : 41.217 s
```

c. Selective Repeat Protocol

CODE:

```
#include<iostream>
using namespace std;
int main()
{
 int w,i,f,frames[50];
 cout<<"18BIT0272 - PRIYAL BHARDWAJ"<<endl;
 cout<<"Enter window size: ";
 cin>>w;
 cout<<"\nEnter number of frames to transmit: ";
 cin>>f;
 cout<<"\nEnter "<<f<<" frames: ";
 for(i=1;i<=f;i++)
 cin>>frames[i];
 cout<<"\n The frames are sent as follows: \n\n";
 cout<<"After sending "<<w<<" frames at each stage sender waits for
acknowledgement sent by the receiver\n\n";
 for(i=1;i<=f;i++)
 {
 if(i%w==0)
 {
 cout<<frames[i]<<"\n";
```

```
cout<<"Acknowledgement of above frames sent is received by sender\n\n";
}
else
cout<<frames[i]<<" ";
}
if(f%w!=0)
cout<<"\nAcknowledgement of above frames sent is received by sender\n";
return 0; }
```

OUTPUT:

```
"C:\Users\PRIYAL BHARDWAJ\Downloads\dccn33.exe"
18BIT0272 - PRIYAL BHARDWAJ
Enter window size: 6

Enter number of frames to transmit: 10

Enter 10 frames: 9
4
1
8
6
3
11
2
7
12

 The frames are sent as follows:

After sending 6 frames at each stage sender waits for acknowledgement sent by the receiver

9 4 1 8 6 3
Acknowledgement of above frames sent is received by sender

11 2 7 12
Acknowledgement of above frames sent is received by sender

Process returned 0 (0x0)   execution time : 71.931 s
```

****