CYCLE-2

-Shreesha H Shetty

1BM21CS209

1.Write a program for error detecting code using CRC-CCITT(16-bits).

Code:

```
#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];
void caltrans(int);
void crc(int);
void calram();
void shiftl();
int main() {
        int n,i=0; char ch,flag=0;
        printf("Enter the frame bits:");
        while((ch=getc(stdin))!='\n')
                m[i++]=ch;
        n=i;
        for(i=0;i<16;i++)
                m[n++]='0';
        m[n]='\0';
        printf("Message after appending 16 zeros:%s",m);
        for(i=0;i<=16;i++)
                g[i]='0';
        g[0]=g[4]=g[11]=g[16]='1';
        g[17]='\0';
        printf("\ngenerator:%s\n",g);
        crc(n);
        printf("\n\nquotient:%s",q);
        caltrans(n);
        printf("\ntransmitted frame:%s",m);
        printf("\nEnter transmitted frame:");
        scanf("\n%s",m);
        printf("CRC checking\n");
```

```
        crc(n);
        printf("\n\nlast remainder:%s",r);
        for(i=0;i<16;i++)
                if(r[i]!='0')
                        flag=1;
                else
                        continue;
        if(flag==1)
                printf("Error during transmission");
        else
                printf("\n\nReceived freme is correct");
}

void crc(int n){
        int i,j;
        for(i=0;i<n;i++)
                temp[i]=m[i];
        for(i=0;i<16;i++)
                r[i]=m[i];
        for(i=0;i<n-16;i++){
                if(r[0]=='1'){
                        q[i]='1';
                        calram();
                }
                else{
                        q[i]='0';
                        shiftl();
                }
                r[16]=m[17+i];
                r[17]='\0';
                for(j=0;j<=17;j++)
                        temp[j]=r[j];
        }
        q[n-16]='\0';
}
```

```
void calram() {
        int i,j;
        for(i=1;i<=16;i++)
                r[i-1]=((int)temp[i]-48)^((int)g[i]-48)+48;
}

void shiftl(){
        int i;
        for(i=1;i<=16;i++)
                r[i-1]=r[i];
}

void caltrans(int n) {
        int i,k=0;
        for(i=n-16;i<n;i++)
                m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
        m[i]='\0';
}
```

Output:

```
Enter the frame bits:1011
Message after appending 16 zeros:10110000000000000000
generator:10001000000100001


quotient:1011
transmitted frame:10111011000101101011
Enter transmitted freme:10111011000101101011
CRC checking


last remainder:0000000000000000

Received freme is correct
```

```
Enter the frame bits:1011
Message after appending 16 zeros:10110000000000000000
generator:10001000000100001


quotient:1011
transmitted frame:10111011000101101011
Enter transmitted freme:101
CRC checking


last remainder:0001000000100001 Error during transmission
```

DOMS | Page No.
Date / /

Q Write a program for error detecting code using CRC-CCITT (16-bits).

→

```
#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];

void caltrans(int);
void crc(int);
void calram();
void shift1();

int main(){
    int n, i=0;
    char ch, flag=0;
    printf("Enter the frame bits:");
    while((ch=getc(stdin))!='\n')
        m[i++]=ch;
    n=i;
    for(i=0;i<16;i++)
        m[n++]='0';
    m[n]='\0';
    printf("Message after appending 16 zeros: %s",m);
    for(i=0;i<=16;i++)
        g[i]='0';
    g[0]=g[4]=g[11]=g[16]='1';
    g[17]='\0';
```

DOMS | Page No.
Date / /

```
    printf("\n generator: %s\n",g);
    crc(n);
    printf("\n quotient: %s",q);
    caltrans(n);
    printf("\n transmitted frame: %s",m);
    scanf("%s",m);
    printf("CRC checking\n");
    crc(n);
    printf("\n last remainder: %s",r);
    for(i=0;i<16;i++)
        if(r[i]!='0')
            flag=1;
        else
            continue;
    if(flag==1)
        printf("Error during transmission");
    else
        printf("\n Received frame is correct");
}
```

```
void crc(int n){
    int i,j;
    for(i=0;i<n;i++)
        temp[i]=m[i];
    for(i=0;i<16;i++)
        r[i]=m[i];
    for(i=0;i<n-16;i++){
        if(r[0]=='1'){
            q[i]='1';
            calram();
        }
        else{
            q[i]='0';
            shiftl();
        }
        r[16]=m[17+i];
        r[17]='\0';
        for(j=0;j<17;j++)
            temp[j]=r[j];
    }
    q[n-16]='\0';
}

void calram(){
    int i,j;
    for(i=1;i<=16;i++)
        r[i-1]=((int)temp[i]-48)^((int)g[i]-48)+48;
}
```

```
void shiftl(){
    int i;
    for(i=1;i<=16;i++)
        r[i-1]=r[i];
}
void caltrans(int n){
    int i, k=0;
    for(i=n-16;i<n;i++)
        m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
    m[i]='\0';
}
```

Output:

```
Enter the frame bits : 1011
Message after appending 16 zeros: 1011000000000000
Generator: 1000100000100001
Transmitted frame: 10111011000101101011
Enter transmitted frame: 10111011000101101011
CRC checking
Last remainder: 0000 0000 0000 0000
Received frame is correct.
```

2.Write a program for congestion control using Leaky bucket algorithm.

Code:

```
#include<stdio.h>

int main(){
	int incoming, outgoing, buck_size, n, store = 0;
	printf("Enter bucket size, outgoing rate and no of inputs: ");
	scanf("%d %d %d", &buck_size, &outgoing, &n);
	while (n != 0){
		printf("Enter the incoming packet size : ");
		scanf("%d", &incoming);
		printf("Incoming packet size %d\n", incoming);
		if (incoming <= (buck_size - store)){
			store += incoming;
			printf("Bucket buffer size %d out of %d\n", store, buck_size);
		}
		else {
			printf("Dropped %d no of packets\n", incoming - (buck_size - store));
			printf("Bucket buffer size %d out of %d\n", store, buck_size);
			store = buck_size;
		}
		store = store - outgoing;
		printf("After outgoing %d packets left out of %d in buffer\n", store, buck_size);
		n--;
	}
}
```

Output:

```
Enter bucket size, outgoing rate and no of inputs: 8 6 4
Enter the incoming packet size : 3
Incoming packet size 3
Bucket buffer size 3 out of 8
After outgoing -3 packets left out of 8 in buffer
Enter the incoming packet size : 3
Incoming packet size 3
Bucket buffer size 0 out of 8
After outgoing -6 packets left out of 8 in buffer
Enter the incoming packet size : 4
Incoming packet size 4
Bucket buffer size -2 out of 8
After outgoing -8 packets left out of 8 in buffer
Enter the incoming packet size : 3
Incoming packet size 3
Bucket buffer size -5 out of 8
After outgoing -11 packets left out of 8 in buffer
```

CYCLE-2

30-08-2023

congestion control

1. Write a program for ~~error detecting code~~ using ~~CRC~~ ~~CRC-16 CRC-32~~ Leaky bucket algorithm.

→

```
# include <stdio.h>

int main(){
    int incoming, outgoing, buck_size, n, store = 0;
    printf("Enter bucket size, outgoing rate and no of input
    scanf("%d %d %d", &buck_size, &outgoing, &n);
    while (n!=0){
        printf("Enter the incoming packet size: ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
        if (incoming <= (buck_size - store)){
            store += incoming;
            printf("Bucket buffer size %d out of %d\n",
                store, buck_size);
        }
        else{
            printf("Dropped %d no of packets\n",
                incoming - (buck_size - store));
            printf("Bucket buffer size %d out of %d\n", store,
                buck_size);
        }
        store = buck_size;
        store = store - outgoing;
        printf("After outgoing %d packets left out of %d
            in buffer\n", store, buck_size);
        n--;
    }
}
```

Output:

```
Enter bucket size, outgoing rate and no of inputs: 8 6 4
Enter the incoming packet size: 3
Incoming packet size: 3

Bucket buffer size 3 out of 8
After outgoing -3 packets left out of 8 in buffer

Enter the incoming packet size: 3
Incoming packet size 3
Bucket buffer size 0 out of 8
After outgoing - 6 packets left out of 8 in buffer

Enter the incoming packet size: 4
Incoming packet size 4
Bucket buffer size -2 out of 8
After outgoing - 8 packets left out of 8 in buffer

Enter the incoming packet size: 3
Incoming packet size 3
Bucket buffer size -5 out of 8
After outgoing -11 packets left out of 8 in buffer.
```

3. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

ClientTCP.py

```
from socket import *

serverName = '127.0.0.1'

serverPort = 12000

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))

sentence = input("\nEnter file name: ")

clientSocket.send(sentence.encode())

filecontents = clientSocket.recv(1024).decode()

print ('\nFrom Server:\n')

print(filecontents)

clientSocket.close()
```

ServerTCP.py

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
        print ("The server is ready to receive") connectionSocket, addr =
        serverSocket.accept() sentence = connectionSocket.recv(1024).decode()

        file=open(sentence,"r") l=file.read(1024)

        connectionSocket.send(l.encode())

        print ('\nSent contents of ' + sentence)

        file.close()

        connectionSocket.close()
```

Output:

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ServerTCP.py
The server is ready to receive
```

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ClientTCP.py

Enter file name: ServerTCP.py

From Server:

from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
```

```
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)

    connectionSocket.send(l.encode())
    print('\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()


Process finished with exit code 0
```

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ServerTCP.py
The server is ready to receive

Sent contents of ServerTCP.py
The server is ready to receive
```

DOMS | Page No. | Date / /

3. Using TCP/IP sockets, write a client-server program to make client sending the filename and the server to send back the contents of the requested file if present.

→

Client TCP.py:

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Enter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('In From Server: \n')
print(filecontents)
clientSocket.close()
```

DOMS | Page No. | Date / /

Server TCP.py:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 6 1:
    print("The server is ready to receive")
    connectionSocket,addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence,"r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("\n Sent contents of "+sentence)
    file.close()
    connection Socket.close()
```

DOMS | Page No.
Date / /

Output:

→ When you run Server TCP.py
The server is ready to receive

→ When you run Client TCP.py
Enter file name : Server TCP.py
From Server:
(The file from Server TCP.py will be copied and displayed here)

→ In Server TCP.py
The server is ready to receive
Sent contents of Server TCP.py
The server is ready to receive.

4.Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

ClientUDP.py

```
from socket import *
serverName ="127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name: ")

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))

filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
# for i in filecontents:
    # print(str(i), end = '')
clientSocket.close()

clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r") con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ('\nSent contents of ', end = ' ')
    print (sentence)
    # for i in sentence:
```

```
# print (str(i), end = '')
file.close()
```

Output:

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ServerUDP.py
The server is ready to receive
```

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ClientUDP.py

Enter file name:  ServerUDP.py

Reply from Server:

from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    con = file.read(2048)
```

```
    serverSocket.sendto(bytes(con, "utf-8"), clientAddress)

    print('\nSent contents of ', end=' ')
    print(sentence)
    #  for i in sentence:
    #  print (str(i), end = '')
    file.close()


Process finished with exit code 0
```

```
/Users/mac/PycharmProjects/cn-lab/venv/bin/python /Users/mac/PycharmProjects/cn-lab/ServerUDP.py
The server is ready to receive

Sent contents of  ServerUDP.py
```

DOMS | Page No. | Date / /

4. Using UDP Sockets, write a client-server program to make client sending the filename and the server to send back the contents of the requested file if present.

→

ClientUDP.py :

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name: ")
clientSocket.sendto(bytes(sentence, "utf-8"),
                    (serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print('\n Reply from Server: \n')
print(fileContents.decode("utf-8"))
# for i in fileContents:
    # print(str(i), end = '')
clientSocket.close()
clientSocket.close()
```

DOMS | Page No. | Date / /

Server UDP.py:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    con = file.read(2048)
        serverSocket.sendto(bytes(con, "utf-8"),
                            clientAddress)
    print("\n Sent contents of ', end = ' ')
    print(sentence)
    # for i in sentence
        #print(str(i), end = '')
    file.close()
```

DOMS | Page No.
Date / /

Output:

→ When you run ServerUDP.py
The server is ready to receive

→ When you run ClientUDP.py
Enter file name : ServerUDP.py
Reply from Server:
(The files from ServerUDP.py will be copied and displayed here)

→ In ServerUDP.py
The server is ready to receive
Sent contents of serverUDP.py
The server is ready to receive.

5.Tool Exploration -Wireshark

5. Wireshark:

Wireshark captures network packets from various interfaces. It enables us to captures specific type of traffic band protocols, source and destination address and all keywords within packet payload.

It's real time monitoring capability is invaluable for observing ongoing network activities. This features helps in diluting sudden traffic spikes and unusual protocol behaviour.

In command prompt: type config
> ipconfig
window Ip configuration

Ethernet adapter Ethernet:
Connection specific DNS suffix:
Link local IPV6 address . . . : fe80:e587:e9fd
: 4edb%2
IPV4 address . . . . . : 10.129.2.84
Subnet mask . . . . . : 255.255.0.0
Default gateway . . . . . : 10.129.0.11

It shows us IpV6, IPV4 addresses, subnet mask and default gateway.
The selected protocol appears at the bottom with source and destination addresses and all keywords written packet payload.

When user clicks on these keywords for example: destination address, the destination address will be highlighted.

For ex: When we select a TCP protocol it shows:
Source port : 5228
Destination port : 58545
Sequence number : 1
Sequence number (raw): 424758453
Acknowlegement Number : 1
Acknowlegement Number: 2