

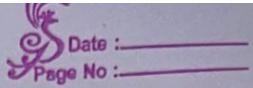
EXPERIMENT-12

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

2) Write a program for error decoding code using CRC-CCITT (16 bits)

```
#include <stdio.h>
#include <string.h>
#define CRC_POLY 0x1021
unsigned short calculate_CRC(unsigned char *data, int length)
{
    unsigned short crc = 0xFFFF;
    for(i=0; i<length; i++)
    {
        crc = (unsigned short) data[i] << 8;
        for(j=0; j<8; j++)
        {
            if(crc & 0x8000)
                crc = (crc << 1) ^ CRC_POLY;
            else
                crc <<= 1;
        }
        return crc;
    }
}

int main()
{
    char data[100];
    printf("Enter data:");
    scanf("%s", data);
    int dataLength = strlen(data);
    unsigned short checksum = calculate_CRC(data, dataLength);
    printf("Calculated CRC: 0x%04X", checksum);
    unsigned short receivedChecksum;
    printf("Enter received CRC:");
}
```



```

SCcmf C"\.hx", & received checksum);
if (received checksum == checksum)
    printf(" Data is error-free\n");
else
    printf(" Data contains errors\n");
}

```

Output:

Enter frame bits: 1011

Message after appendix 16m : 1011 0000 0000 0000

Generata: 10001000000100001

Quotient: 1011

Toronto forecast: 1011 1011 0001 0110 1011

Enter toute 1011 1011 0001 0110 1011

~~left standard :- 0000 0000 0000 0000~~

~~Dist. 11 emasidized~~

Dan is een -je

[A blank horizontal line for a signature.]

810

2/9/22

OUTPUT

```
Enter the frame bits:1011
Message after appending 16 zeros:10110000000000000000
generator:1000100000100001

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

last remainder:0000000000000000

Received frame is correct
Process returned 0 (0x0)   execution time : 14.468 s
Press any key to continue.
```

EXPERIMENT-14

Q) Write a program for congestion control using Leaky bucket algorithm

g) Write a program for congestion control using leaky bucket.

```
#include < stdio.h>
int main()
{
    int incoming, outgoing, buck_size, m, store = 0;
    printf("Enter bucket size, outgoing rate & no. of inputs");
    scanf("%d %d %d", &buck_size, &outgoing, &m);
    while (m != 0)
    {
        printf("Enter incoming packet size: ");
        incoming = atoi(stdin);
        if (incoming <= (buck_size - store))
        {
            store = incoming;
            printf("bucket buffer size=%d count=%d", store,
                buck_size);
        }
        else
        {
            printf("dropped %d no. of packets\n", incoming - incoming - (buck_size - store));
            printf("bucket buffer size=%d count=%d; store=%d", buck_size);
            store = buck_size;
        }
        store = store + outgoing;
        printf("After outgoing %d packet left out %d in
            buffer, store = %d, buck_size = %d", outgoing, store, buck_size);
    }
}
```



Date : _____

Page No. _____

Output1:

Enter bucket size, outgoing rate of net
20 10 2

Enter incoming packet size = 30

incoming packet = 30

Drops 10 no of packets

bucket buffer size 0 cont of 20

After cont going 10 packets left 20 in buffer

Enter incoming packet size: 10

incoming packet size 10

bucket buffer size 10 cont of 20

After cont going 10 packets left 20 in buffer

Output2: Enter bucket size, outgoing 2 no of 11 ps.

① 8
② 10

2/9/23

OUTPUT

```
Enter bucket size, outgoing rate and no of inputs: 20 10 2
Enter the incoming packet size : 30
Incoming packet size 30
Dropped 10 no of packets
Bucket buffer size 0 out of 20
After outgoing 10 packets left out of 20 in buffer
Enter the incoming packet size : 10
Incoming packet size 10
Bucket buffer size 20 out of 20
After outgoing 10 packets left out of 20 in buffer

Process returned 0 (0x0)    execution time : 22.003 s
Press any key to continue.
```

EXPERIMENT-15

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

- 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

Server

- 1) A Server has a bind() method which binds to specific IP and port so that it can listen to incoming request on that IP port and port.
- 2) A server has a listen() method which puts the Server into listening mode. This allows server to take incoming connections.
- 3) Server has accept() and close() method accept() initiates a connection accept() - with client. close() - closes the connection with the Client

Step1: Open idle, in that in file, open new file and write the following code and save as Server.py

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 recve")
    ConnectionSocket, ClientAddress = ServerSocket.accept()
```

```
file = open('sentence', 'r')
l = file.read(1024)
connectionSocket.send(l.encode())
print("In Sent content : " + sentence)
file.close()
connectionSocket.close()
```

Step2: Run the file server.py

O/P \Rightarrow The server is ready to send. This shows that server is working.

Client:

Step1: Make a socket object

Step2: Establish a connection with server and firstly we will receive data from the server and close the function.

Step3: Open idle and open new file and write the following code and save as "client.py"

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
ClientSocket = socket(AF_INET, SOCK_STREAM)
ClientSocket.connect((ServerName, ServerPort))
sentence = input("In Enter file name : ")
ClientSocket.send(sentence.encode())
fileContent = ClientSocket.recv(1024).decode()
print("In Received from server : \n" + fileContent)
```

Print (full code)

ClientSocket.Close()

Run the file client.py

Client O/P = enter file name . server.py

Server O/P = The service is ready to receive
the contents of same . py.
The server is ready to receive.

Q) Using UDP Sockets , write a Client . Server . Program to
make client sending the file name and the server to
Send back the location of our file by present.

10/10

2/9/23

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

The screenshot shows a Windows desktop environment with several windows open:

- A Microsoft Edge browser window titled "Inbox (254) - raaghbir.cs21@bm...". It displays a Google Docs document titled "SocketProgramming_2023.docx".
- An "IDLE Shell 3.10.8" window for the server script. The code is as follows:

```
>>> Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> === RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/client.py ==

Enter file name: server.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()

>>>
>>>
```

- An "IDLE Shell 3.10.8" window for the client script. The code is as follows:

```
>>> Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> === RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/client.py ==

Enter file name: server.py

From Server:

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()
```

- A desktop taskbar at the bottom showing various icons including File Explorer, Control Panel, and Jupyter Notebook.

EXPERIMENT-16

Q) 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

Q) 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.

Here, like in TCP/IP we create Socket object and bind it to the specified port and server will be constantly listening when the Client sends request is received accordingly.

Server UDP.py

```
from socket import *
Server port = 12000
Server Socket = Socket (AF_INET, SOCK_DGRAM)
Server Socket.bind(("127.0.0.1", Server Port))
print("The Server is ready to receive")
while True:
    Sentence, Client Address = Server Socket.recvfrom(2048)
    Sentence = Sentence.decode("utf-8")
    file = open(Sentence, "r")
    Content = file.read()
    file.close()
    Server Socket.sendto(Content, Client Address)
    print("In Sent Content of", End="")
    print(Sentence)
# Print Content
```

Client UDP.py

```
from socket import *
Server Name = ("127.0.0.1",
```

sentence = input("\n Enter file name: ")
 Client socket . sendto (bytes (sentence , "utf-8") , (serverName , serverPort))
 file contents , server Address = clientSocket . recvfrom (2048)
 print (" \n Reply from Server : \n ")
 print (filecontents . decode ("utf-8"))

O/P (Client)

Enter the file name: Server UDP.py

O/P (Server)

The server is ready to receive sent port of
Server UDP.py.

10/10

2/9/23

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 (" \n Reply 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()
```

The screenshot shows a Windows desktop environment with several open windows. In the top taskbar, there are tabs for 'Inbox (254)', 'docs.google.com/document/d/...', 'C:\LAB - Google Drive', and 'SocketProgramming_2023.docx'. The desktop background is a green landscape image. On the left, a vertical sidebar displays icons for 'IBM18CS01...', 'Control Panel', 'Unstable', 'Control Panel', 'Unstable', 'Jupyter T...', 'Untitled...', 'K Nearest Neigh...', 'Cisco Packet Tracer Studio...', 'Jupyter I...', 'New Text Document', and 'Inbox (254)'. The main workspace contains four windows: 1) A large window titled 'SocketProgramming_2023.docx' showing Python code for a server using the 'socket' module. 2) An 'IDLE Shell 3.10.8' window showing the same server code running. 3) A smaller window titled 'server.py' showing the server code. 4) Another 'IDLE Shell 3.10.8' window titled 'client.py' showing Python code for a client using the 'socket' module. The system tray at the bottom right shows the date as '01-09-2023' and various system icons.