6. Create a socket (UDP) Write a program to implement simple client-server application using UDP.

```
import socket
import time
host "127.0.0.1"
post = 9999
addr = (host,post)
clint = socket.scoket(socket.AF_INET, socket.sock_DGRAM)
message=(input("Enter a messege :")).encode()
clint.swdto(message.addr)
while True:
        data,addr-client.recvfrom(1024)
        data = data.decode()
        print(data)
```

```
import socket
import time
s_address = "127.0.0.1"
post_address = 9999
server = socket.scoket(socket.AF_INET, socket.sock_DGRAM)
server.bind((s_address,post_addresss))
message,address = server.recvfrom(1024)
```

```
print(message)
while True:
    meaasge , address = server.recvfrom(1024)
                            message = meaasge.decode()
    print(message)
                            t=time.localtime()
                            curret_time = time.strftime("%H : %M : %S",t)
                            current_time = current_time.encode()
                            server.sendto(current_time,address)
```

```
import socket
import time
s_address = "13.0.0.219"
post_address = 9999
server = socket.scoket(socket.AF_INET, socket.sock_DGRAM)
server.bind((s_address,post_addresss))
message,address = server.recvfrom(1024)
message = message.decode()
print(message)
```

```
while True :
        t=time.localtime()
        curret_time = time.strftime("%H : %M : %S",t)
        current_time = current_time.encode()
        server.sendto(current_time,address)
        time.sleep(5)
```

```
import socket
import time
s_address = "13.0.0.219"
post_address = 9999
addr=(host,post)
server = socket.scoket(socket.AF_INET, socket.sock_DGRAM)
message=(input("Enter a message")).encode()
clint.sendto(message,addr)
while True :
        data,addr = clint.recvfrom(1024)
```

```
data=data.decode()
print(data)
```

7) Simulation of ARP Web page downloading TCP Module Implementation--

Source___

```
#include<stdio.h>
#include<sys/types.h>
```

```
#include<sys/shm.h>

#include<string.h>

main()

{
int shmid,a,i;

Char *ptr,*shmptr;

Shmid_mget(300.10,IPC_CREAT 10666);

Shmptr_mat(shmid,NULL,0);

Ptr_shmptr;

for(i=0;i<3;i++)

{
puts(“Enter the name:”);

scanf(“%s”,ptr);

a=strlen(ptr);

printf(“String length :%d”,a);

ptr[a]=’ ‘;

puts(“Enter ip:”);

ptr=ptr+a+1;

}

ptr[strlen(ptr)]=’\0’;

printf(“\n ARP table at server side is=\n%s”,shmptr);

shmdt(shmptr);

}
```

Client___

```
#include<stdio.h>

#include<string.h>
```

```
#include<sys/types.h>

#include<sys/shm.h>

main()

{
int shmid,a;

char *ptr,*shmptr;

char ptr2[51],ip[12],mac[26];

shmid=shmget(3000,10,0666);

shmptr=shmat(shmid,NULL,0);

puts(“The ARP table is::”);

printf(“%s”,shmptr);

printf(“\n 1.ARP\n 2.RARP \n 3.Exit\n);

scanf(“%d”,&a);

switch(a)

{

case 1:
  puts(“Enter ip address”);

  scnaf(“%s”,ip);

  ptr=str str(shmptr,ip);

 ptr-=8;

 scanf(ptr,”%s%s”,ptr2);

 printf(“Mac address is:%s”,ptr2);

  break;

case 2:

    puts(“Enter the mac address”);

    scanf(“%s”,mac);

    ptr=str str(shmptr,mac);

    scanf(ptr,”%*s %s”,ptr2);
```

```
  printf(“%s”,ptr2);
 break;
case 3:
   exit(1);
}
}
```

8. Implementation of RMI Write a socket program for implementation of client program in c language and server program in java language. Write a client server socket programming, server stores a TXT file consisting of State Name and Capital city. Client will send ’State Name’ as request, and server should respond with matching ’Capital City’ and vice-versa, if found else adds this entry to the TXT file. Connection should terminate only if client terminates it. I/P : West Bengal O/P : Kolkata OR I/P : Kolkata O/P : West Bengal

Server

```
import java.rmi.*;
import java.rmi.server.*;
public class remoServer extends Unicast RemoteObject implements remoInter
{
public remoServer() throws RemoteException{
super();
}
public string message() throws RemoteException{
return “Hi”;
}
public static void main(String args[])
{
try
```

```
}
Remoserver r=new remoserver();
Naming.rebind(“TestServer”,r);
System.out.println(“the Server Is Ready”);
}
catch(Exception e)
{
}
}
}
```

Client

```
import java.rmi.*;
Import java.rmi.registry.*;
public class remoclient
{
public static void main(String args[]){
try{
remoInter s=(remoInter);
Naming.lookUp(“TestServer”);
System.out.println(s.message[]);
}
catch(Exception e)
{
}
}
```

}

11. Implement on a data set of characters the three CRC polynomials CRC 12, CRC 16 and CRC CCIP

--Implement on a dataset of characters the Three CRC polynomials CRC 12, CRC 16 and CRC CCIP

```
#include <stdio.h>
int gen [4], genl, tel, rim[4];
main(){
int i,j, fr[8],dupfr[11],recfr[11],then,flug;
int frl=8, genl= 4;
printf("Enter frame");
for (i=0; i<frl;i++)
{

scanf("%d",&fr[i]);
dupfr[i]=fr[i];
}
printf("Enter generator;");
for (i=0; i<genl;i++)
scanf("%d", &gen [i]);
then = frl + genl-1;
for(i=frl;i<then;i++)
{
dupfr[i]=0;
}
remainder(dupfr);
for(i=0;i<frl;i++)
```

```
{
recfr[i]=fr[i];
}
remainder=(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
        flag++;
}
if(flag==0)
        printf("Frame received correctly");
else
  printf("The received Frame is wrong");}

remainder(int fr[])
{
if(fr[k] ==1)
{
k1=k;
for(i=0;j=k;l<genl;i++;j++)
{
rem[i]=fr[j].gen[i];
}
for(i=0i<gencl;i++){
fr[k1]=rem[i];
k1++;
```

}}}}

9.. Case study of different routing Algorithm:-

A routing algorithm is a routing protocol determined by network layer for transmitting data packets

form source to destination. The algorithm determines The best or least cost path for data transmission from sender / source to receiver/destination.

Types of Routing Algorithms

static or Dynamic Routing

Distributed or Centralized

Single Path Flat or Multipath

Flat or Hierarchical

Intra Domain or Inter Domain

Link State our Distance Vector.

Further Routes: can be grouped into two categories…

Non Adaptive routing:

-Once the pathway to the destination has been selected, the router sent us the packets for that destination along that one Route. The Routing decisions are not mode, based on the condition on network topology of the network.

Adaptive Routing:-

A router may select a new route for each packet in response to changes in the condition & topology of the networks.

Shortest path routing →It comes under Adaptive Routing

Eg. Dijkstra Algo.

Distance Vector Routing ->It comes under non-adaptive routing.

Eg. Bellman Found.

Flooding Algo -it is an non adaptive algorithm or static algorithm.

10. Implement the data link layer framing methods such as character count, character stuffing and bit stuffing Write a client server socket programming: write a client-server Cyclic redundancy check (CRC) program steps to follow:

(a) Client sends a message to server with an appended CRC.

(b) Server checks the data for any error and accepts it.

(c) Server replies with 'good data' / 'bad data' depending upon there is no error on with error. (Server and client will share a common divisor of 16 bit CRC)

→

```
#include<stdio.h>

#include<string.h>

main(){

int a[20],b[30],i,j,k,count,n;

printf("Enter frame length");

scanf("%d",&n);

printf("Enter input frame(0's and 1's only):);

for(i=0;i<n;i++)

scanf("%d",&a[i]);

int i=0,count=1,j=0;

while(i<n){

if(a[i]==1)

{

b[j]=a[i];
```

```
for(k=i+1;a[k]==1 && k<n && count<j;k++)
{
j++;
b[j]=a[k];
count++;
if(count==5)
{
j++;
b[j]=0;
}
i=k;
}
}
else
{
b[j]=a[i];
}
i++;
j++;
}
printf("After stuffing the frame is:");
for(i=0;i<j;i++)
printf("%d",b[i]);
}
```

CRC SENDER:-

```
#crc sender
data=input('enter input')
divisor = "1101"


p = len(divisor)


divident = data + '0'*(p-1)


tmp = divident[0 : p]


def xor(a, b):


    result = []


    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')


    return ''.join(result)



while p < len(divident):
```

```
    if tmp[0] == '1':

        tmp = xor(divisor , tmp) + divident[p]
    else:



        tmp = xor('0'*p , tmp) + divident[p]

    p += 1



if tmp[0] == '1':
    tmp = xor(divisor , tmp)
else:
    tmp = xor('0'*p , tmp)

checkword = tmp
print('enter the checkword:',checkword)

codeword = data + checkword

print('enter the codeddata:',codeword)
```

CRC RECEIVER:-

```
#crc receiver
```

```
data=input('enter input')
divisor = "1101"

p = len(divisor)

divident = data

tmp = divident[0 : p]

def xor(a, b):

    result = []

    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)



while p < len(divident):

    if tmp[0] == '1':
```

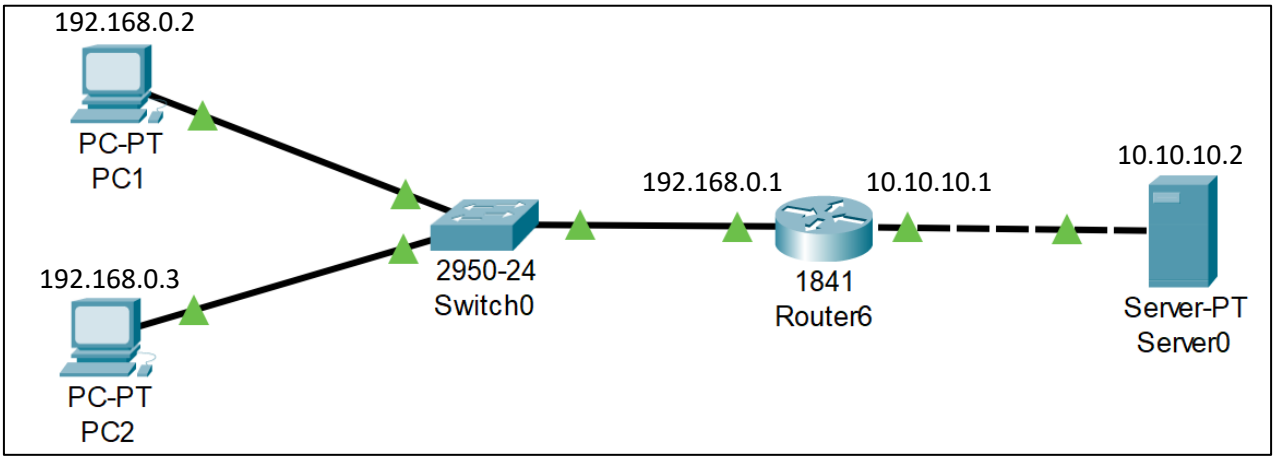
```
                tmp = xor(divisor , tmp) + divident[p]

        else:

                tmp = xor('0'*p , tmp) + divident[p]



        p += 1



if tmp[0] == '1':
        tmp = xor(divisor , tmp)
else:
        tmp = xor('0'*pick , tmp)

checkword = tmp
print('the checkword:',checkword )

if(int(checkword,2)==0):
  print('The data is error free:')

else:
  print('The data is error aded:')
```

# Assignment - 03

- **FTP Server Configuration:**

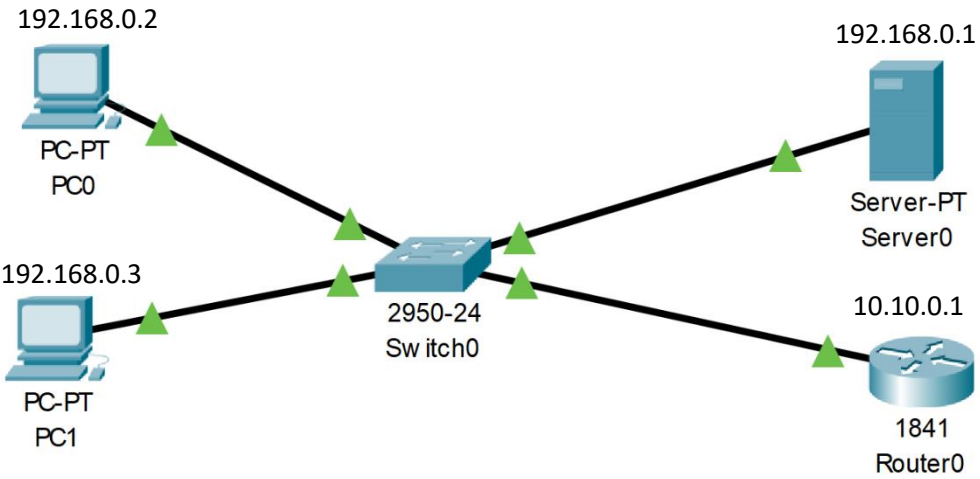
**Steps:**

1) Open Cisco Packet Tracer and select 2 End Devices (PC device), 1 Switch, 1 Router, 1 Server.
2) Now Connect all the devices using the auto connection. 
3) Then configure the IP addresses as per the diagram.




4) Now just wait for some time to let all the connection status turns green.
5) Now we have achieved a connection where a class C IP address is being translated to class A IP Address.
6) Go to one of the PC devices and on Desktop tab select CMD.
7) Now we need to check the connection to the server by `C:\>ping 10.10.10.2`
8) If reply is coming then it means the server is properly configured and connected.
9) Go to the Server → Services →FTP.
10) Put on the FTP service and give username and password and click on ADD.
11) Come back to PC device and open the CMD and type `C:\>ftp 10.10.10.2`
12) It will ask for username and password. Provide the username and password configured earlier.
13) Once the connection is established exit rom the CMD and go to Text Editor and make a new text file.
14) Save the new text file and return to cmd and type `ftp>put filenme.txt`
15) This will send the text file from the PC device (192.168.0.2) to Server (10.10.10.2).
16) Now to verify that the file has been transferred to the server, so type `ftp>dir`
17) You will see your Filename in the list.
18) Now to get a file from server to PC type `ftp>get filename.txt`
19) Now exit from FTP type ctrl+C, then type dir to check that the file is there in the PC or not.
20) So we have successfully send and got a file from a server using FTP protocol.

- **DHCP Server Configuration:**
  1) Open Cisco Packet Tracer and select 2 End Devices (PC device), 1 Switch, 1 Router, 1 Server.
  2) Now Connect all the devices using the auto connection.
  3) Then configure the IP addresses as per the diagram.




  4) Now just wait for some time to let all the connection status turns green.
  5) Now go to the server → Desktop → IP Configuration and set the IP 192.168.0.1.
  6) Go to Services → DHCP and set Default Gateway 192.168.0.1 and DNS Server 10.0.0.1.
  7) Set the Start IP 192.168.0.0 and Max Users 256
  8) Now go to every PC device → Desktop → IP Configuration and set it to DHCP.
  9) Now all the PC will have a DHCP address.

# Assignment - 04

**1) Write a program to find the IP address of the system.**

```
import socket
print("Host Name:", socket.gethostname(), "\nIp
Address:", socket.gethostbyname(socket.gethostname()))
```

**2) Write a socket program for implementation of echo.**

**Server side code:**

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
(c, cip) = s.accept()
c.send(c.recv(1024))
s.close()
```

**Client side code:**

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
data = input()
c.send(data.encode())
dataFromServer = c.recv(1024)
print(dataFromServer.decode())
c.close()
```

**3) Write a client-server application for chat using TCP.**

**Server side code:**

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
while True:
    (c, cip) = s.accept()
    data = c.recv(1024).decode()
    print("Client:",data)
    data = input("Enter Text: ")
    c.send(data.encode())
```

**Client side code:**

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
while True:
    data = input("Enter Text: ")
    c.send(data.encode())
    data = c.recv(1024).decode()
    print("Server:",data)
```

4) **Write a program using client server socket programming: Client needs to authenticate itself by entering a server defined string as a password (like OTP) and then to say Hi to server. Server replies with a Hello. Press any key to exit.**

**Server side code:**

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
(c, cip) = s.accept()
c.send("Enter OTP:".encode())
otp = c.recv(1024).decode()
if otp == '8894':
    c.send("You are Authenticated".encode())
    data = c.recv(1024).decode()
    print("Client:",data)
    data = input("Enter Text: ")
    c.send(data.encode())
else:
    c.send("You are Authenticated".encode())
s.close()
```

**Client side code:**

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
data = c.recv(1024).decode()
print(data, end=" ")
otp = input()
c.send(otp.encode())
data = c.recv(1024).decode()
print(data)
```

```
if data == "You are Authenticated":
    data = input("Enter Text: ")
    c.send(data.encode())
    data = c.recv(1024).decode()
    print("Server:",data)
else:
    c.close()
```

# Assignment - 05

1) **Write a program to Perform File Transfer in Client & Server Using TCP/IP.**

**Server side code:**

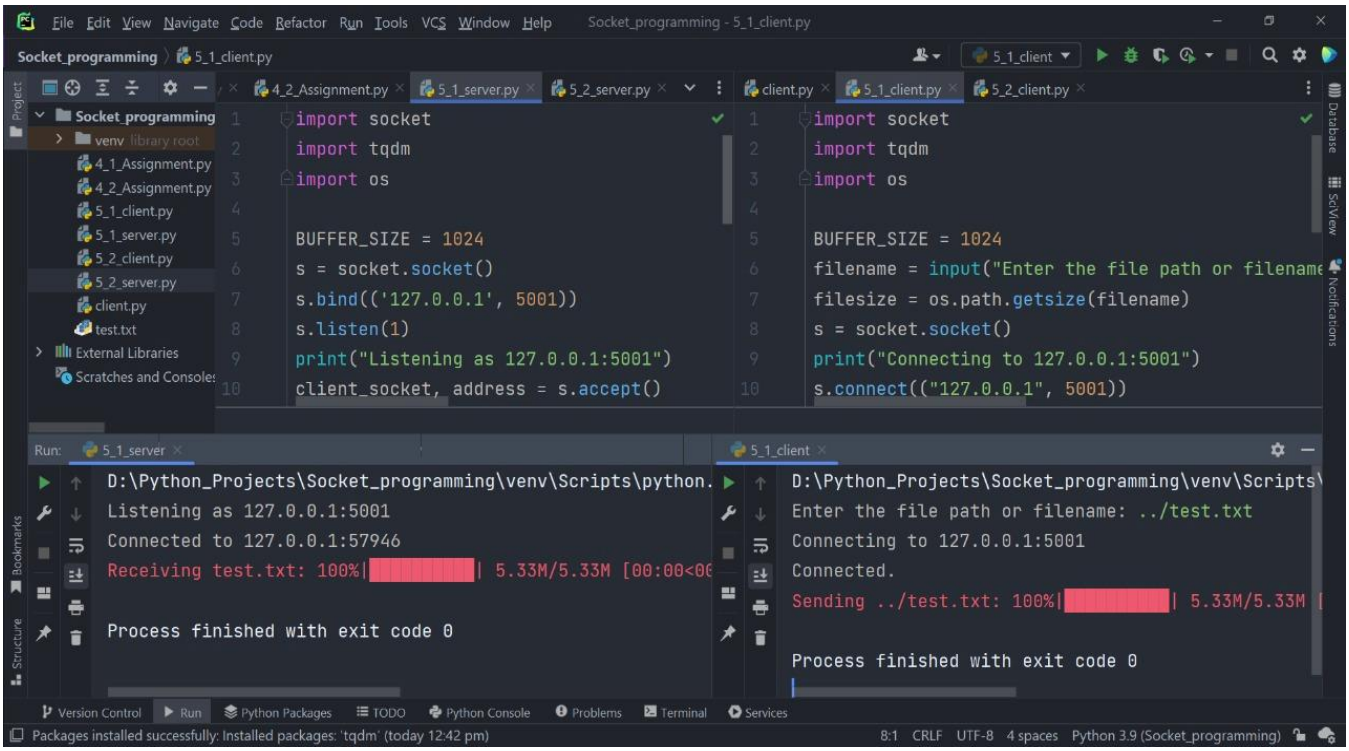
```
import socket
import tqdm
import os

BUFFER_SIZE = 1024
s = socket.socket()
s.bind(('127.0.0.1', 5001))
s.listen(1)
print("Listening as 127.0.0.1:5001")
client_socket, address = s.accept()
print(f"Connected to {address[0]}:{str(address[1])}")
filename, filesize =
client_socket.recv(BUFFER_SIZE).decode().split('||')
filename = os.path.basename(filename)
filesize = int(filesize)
progress = tqdm.tqdm(range(filesize), f"Receiving
{filename}", unit="B", unit_scale=True,
unit_divisor=1024)
with open(filename, "wb") as f:
    while True:
        bytes_read = client_socket.recv(BUFFER_SIZE)
        if not bytes_read:
            break
        f.write(bytes_read)
        progress.update(len(bytes_read))
client_socket.close()
s.close()
```

**Client side code:**

```
import socket
import tqdm
import os

BUFFER_SIZE = 1024
filename = input("Enter the file path or filename: ")
filesize = os.path.getsize(filename)
s = socket.socket()
print("Connecting to 127.0.0.1:5001")
s.connect(("127.0.0.1", 5001))
print("Connected.")
s.send(f"{filename}||{filesize}".encode())
```

```
progress = tqdm.tqdm(range(filesize), f"Sending
{filename}", unit="B", unit_scale=True,
unit_divisor=1024)
with open(filename, "rb") as f:
    while True:
        bytes_read = f.read(BUFFER_SIZE)
        if not bytes_read:
            break
        s.sendall(bytes_read)
        progress.update(len(bytes_read))
s.close()
```

**Output:**




**2) Write a program to implement Remote Command Execution (RCE)**

**Server side code:**

```
import socket
import os

BUFFER_SIZE = 4096
s = socket.socket()
s.bind(('127.0.0.1', 5001))
s.listen(1)
print("Listening as 127.0.0.1:5001")
client_socket, address = s.accept()
print(f"{address[0]}:{str(address[1])} is Connected to
terminal")
while True:
    print("\nClient@Server>>", end=" ")
```

```
    command = client_socket.recv(BUFFER_SIZE).decode()
    print(command)
    if command == 'exit':
        break
    os.system(command)
print("Closing remote connection with client")
client_socket.close()
s.close()
```

**Client side code:**

```
import socket

BUFFER_SIZE = 4096
s = socket.socket()
print("Connecting to 127.0.0.1:5001")
s.connect(("127.0.0.1", 5001))
print("Connected to Server Terminal")
while True:
    command = input("\nServer>> ")
    s.sendall(command.encode())
    if command == 'exit':
        break
print("Closing remote connection with Server")
s.close()
```

**Output:**