VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

COMPUTER NETWORKS

Submitted by

AMIT GUPTA (1BM20CS012)

Under the Guidance of
Dr.Shyamala G
Assistant Professor,BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019

October-2022 to Feb-2023

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**COMPUTER NETWORKS**" carried out by **AMIT GUPTA (1BM20CS012),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **COMPUTER NETWORKS - (20CS5PCCON)** work prescribed for the said degree.

Dr. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
professor and Head
Department of CSE
BMSCE ,Bengaluru

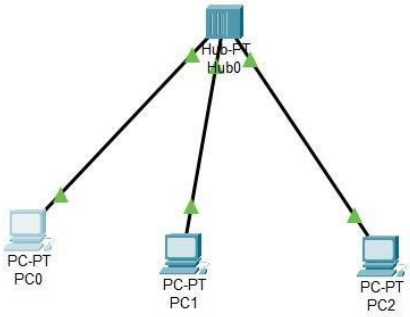
`

Index

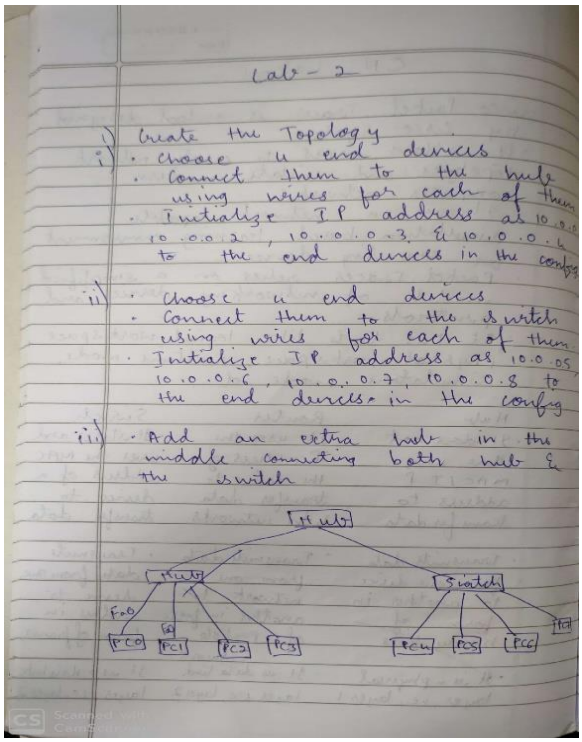
Cycle-1 Experiment No 1 Aim of the program

Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.

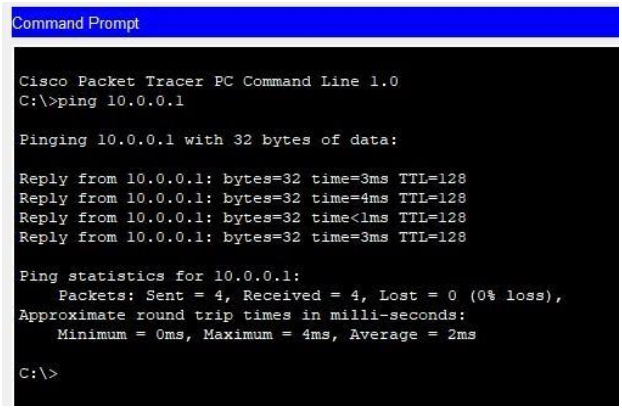
Hub Topology



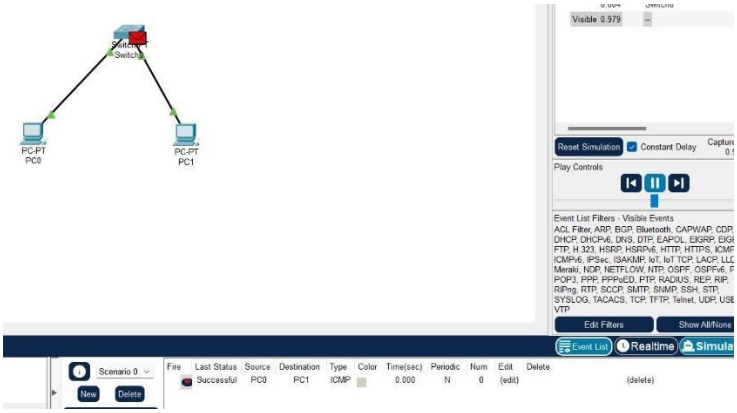
Procedure



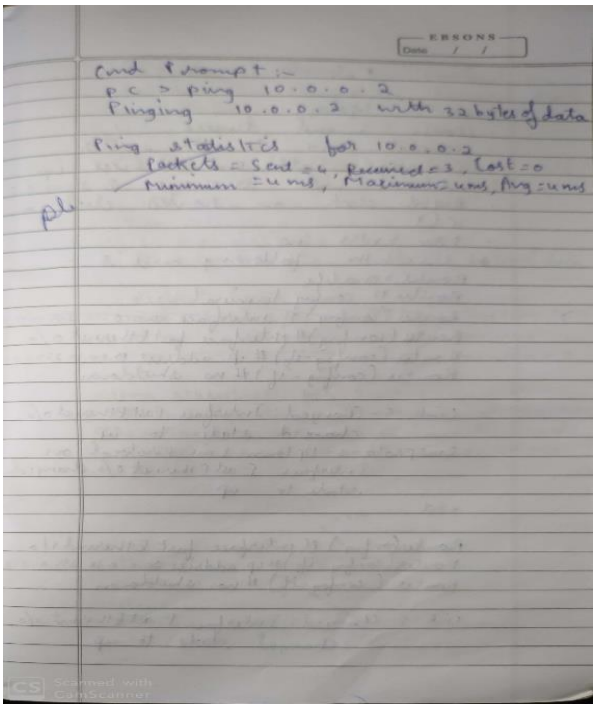
Output



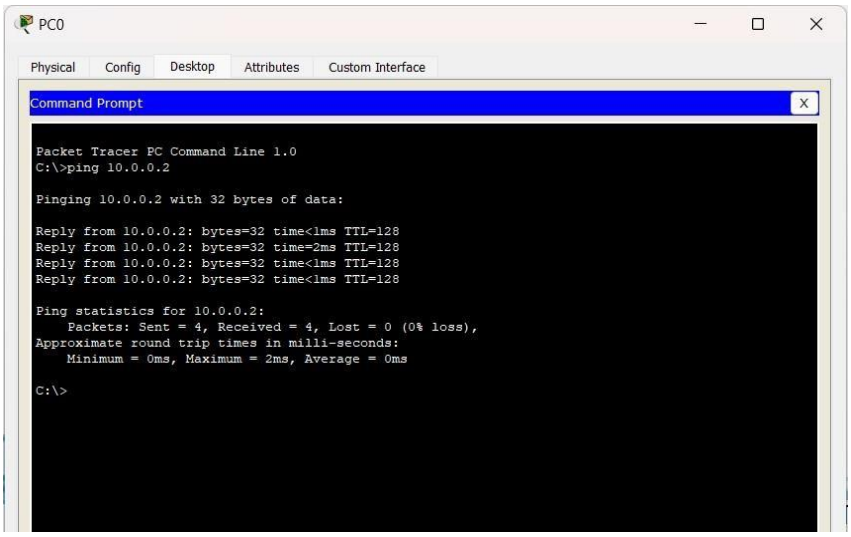
Switch Topology



Procedure



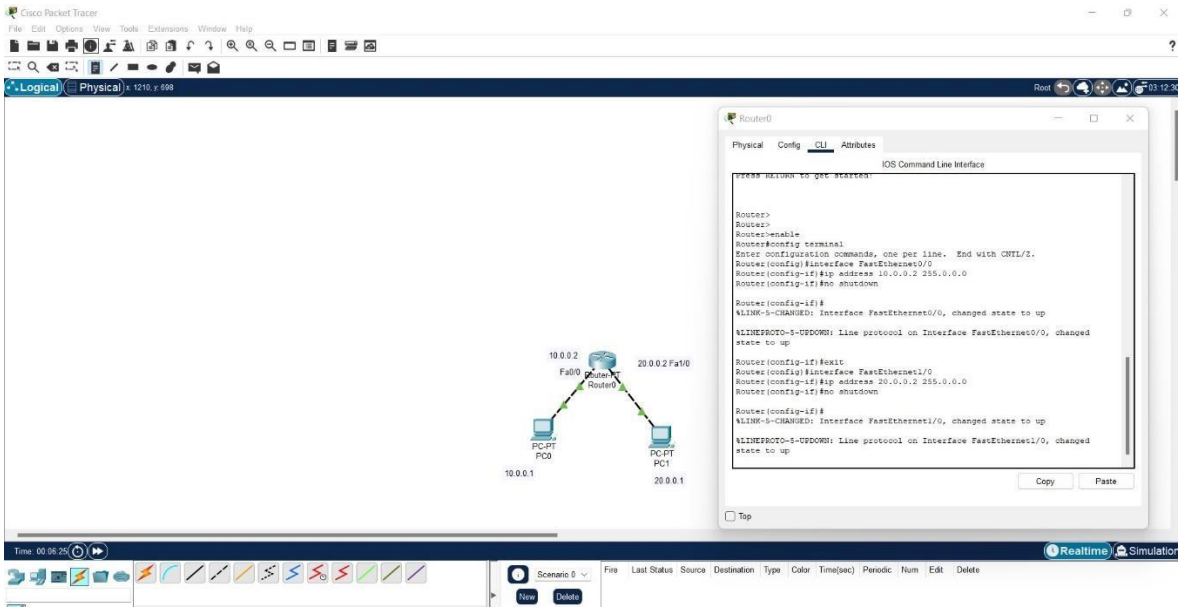
Output



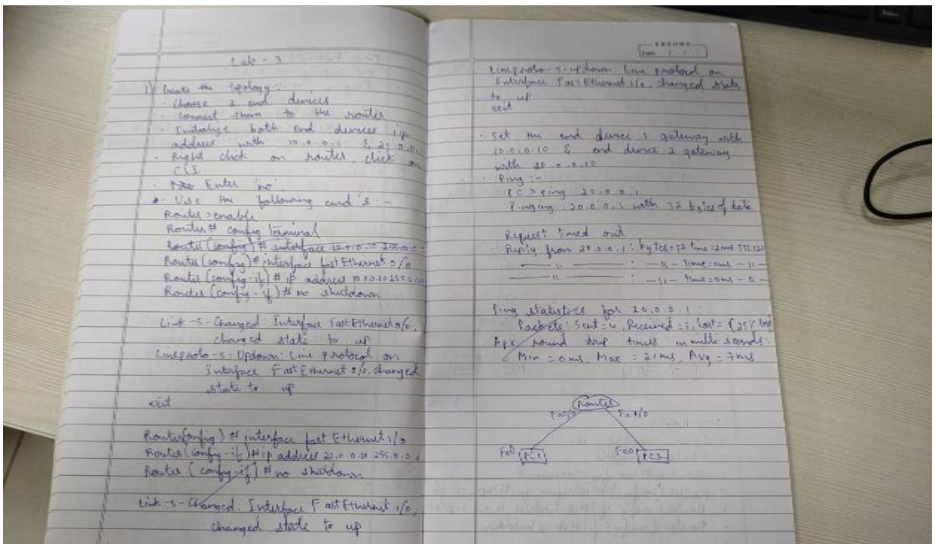
Experiment No 2 Aim of the program

Configuring IP address to Routers in Packet Tracer. Exploring the following messages: Ping Responses, Destination unreachable, Request timed out, Reply.

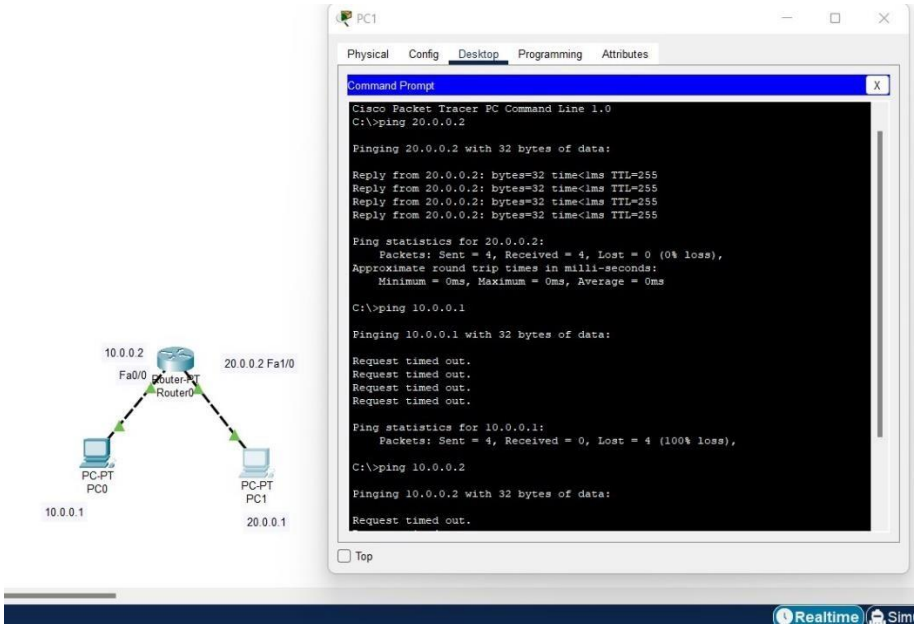
Topology



Procedure



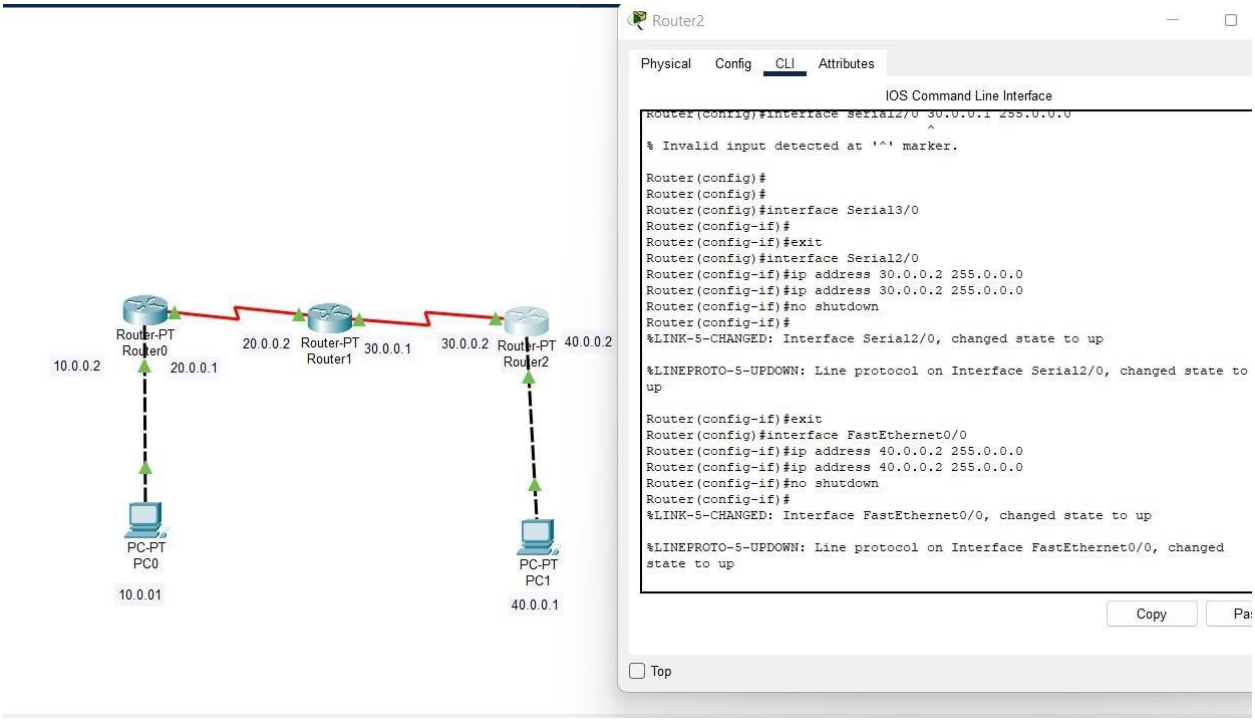
Output


PC1
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Reply from 20.0.0.2: bytes=32 time<1ms TTL=255
Reply from 20.0.0.2: bytes=32 time<1ms TTL=255
Reply from 20.0.0.2: bytes=32 time<1ms TTL=255
Reply from 20.0.0.2: bytes=32 time<1ms TTL=255
Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Request timed out.
Top
10.0.0.2
Fa0/0
Router-PT
Router0
20.0.0.2 Fa1/0
PC-PT
PC0
10.0.0.1
PC-PT
PC1
20.0.0.1
Realtime

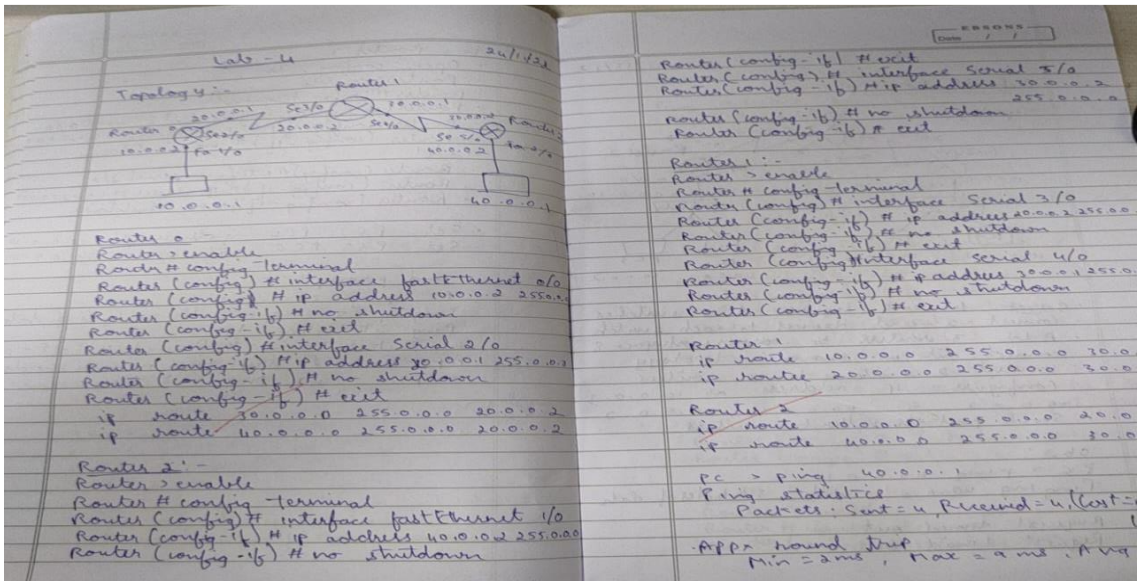
Experiment No 3 Aim of the program

Configuring default route to the Router

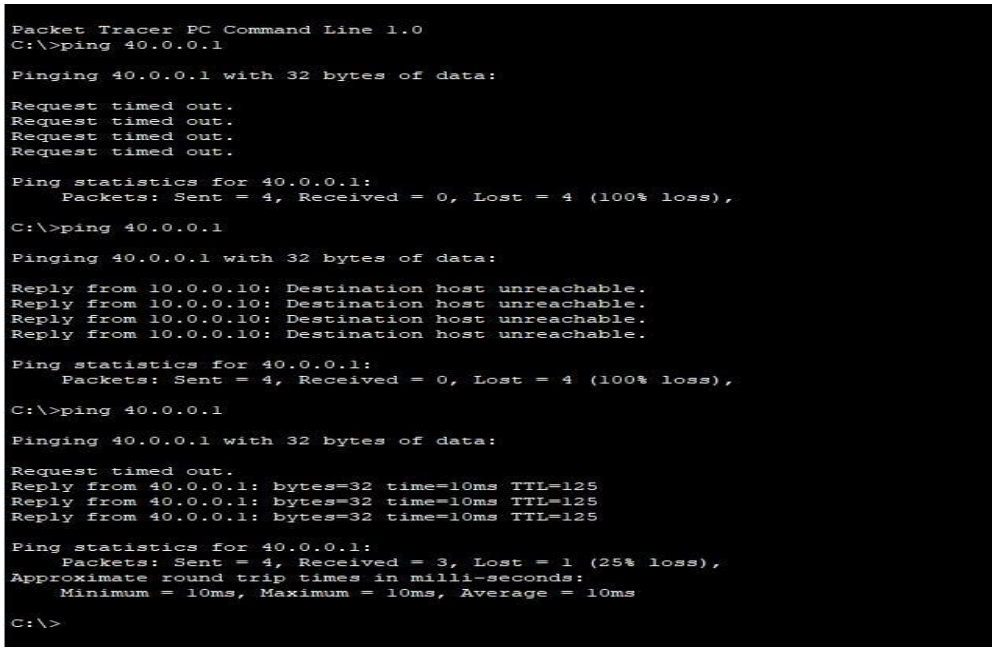
Topology



Procedure



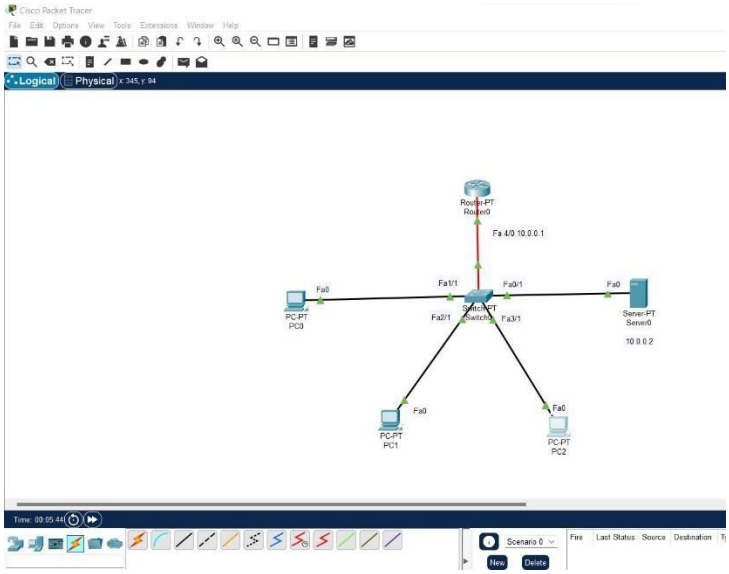
Output


Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 10ms, Maximum = 10ms, Average = 10ms
C:\>

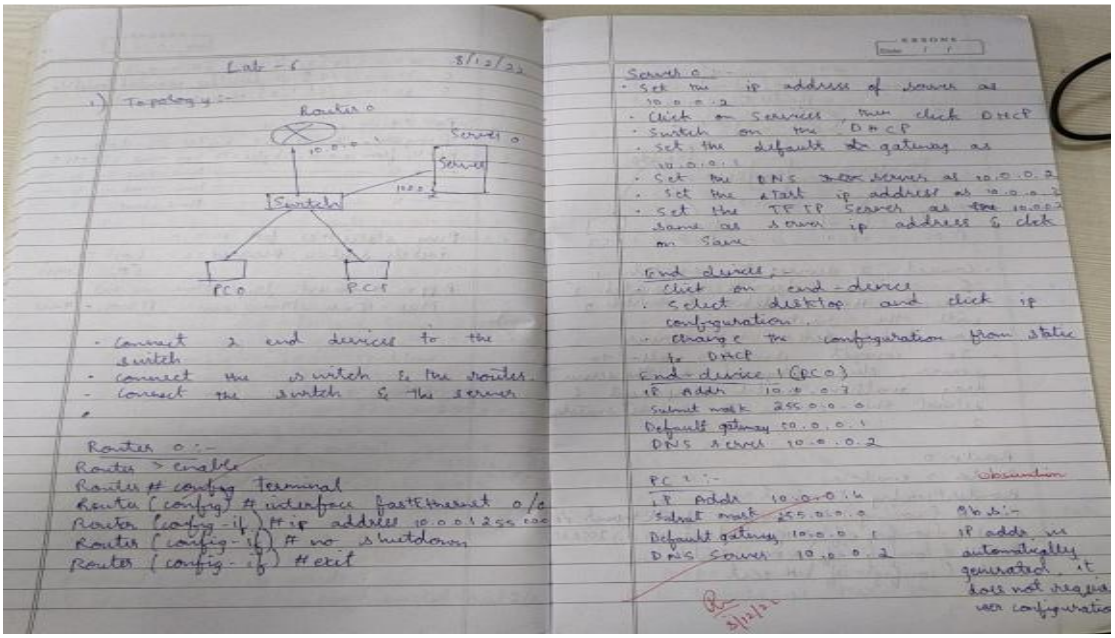
Experiment No 4 Aim of the program

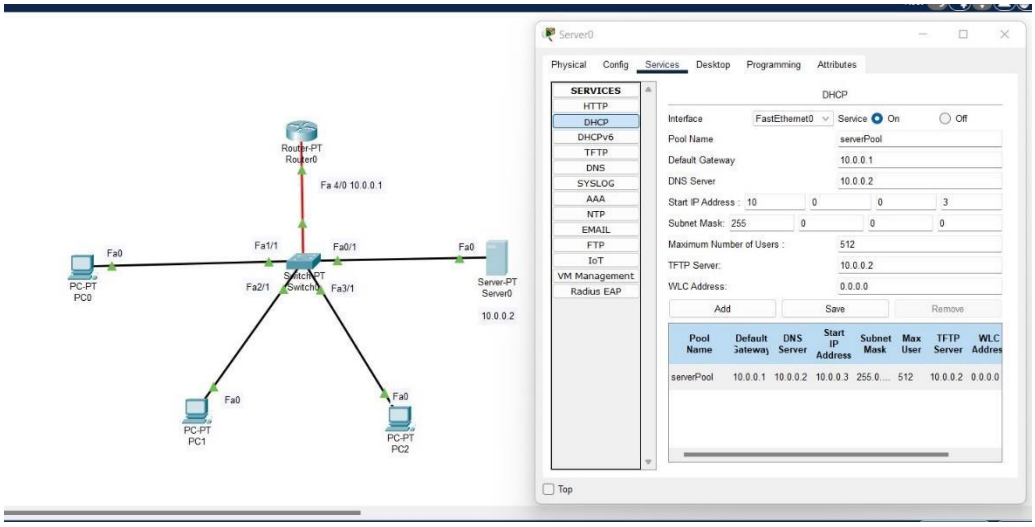
Configuring DHCP within a LAN in a packet Tracer

Topology

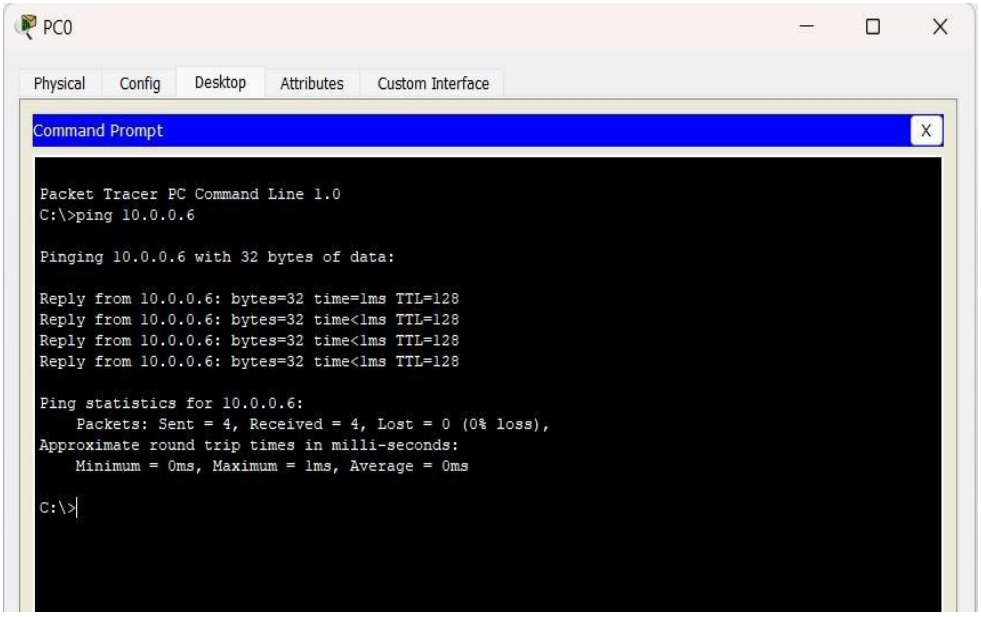


Procedure




Router-PT
Router0
Fa 4/0 10.0.0.1
Fa0
PC-PT
PC0
Fa1/1
Fa0/1
Fa0
Fa2/1
Switch-PT
Switch0
Fa3/1
Server-PT
Server0
10.0.0.2
Fa0
PC-PT
PC1
Fa0
PC-PT
PC2
Server0
Physical
Config
Services
Desktop
Programming
Attributes
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoT
VM Management
Radius EAP
DHCP
Interface
FastEthernet0
Service
On
Off
Pool Name
serverPool
Default Gateway
10.0.0.1
DNS Server
10.0.0.2
Start IP Address :
10
0
0
3
Subnet Mask:
255
0
0
0
Maximum Number of Users :
512
TFTP Server:
10.0.0.2
WLC Address:
0.0.0.0
Add
Save
Remove
Pool Name
Default Gateway
DNS Server
Start IP Address
Subnet Mask
Max User
TFTP Server
WLC Addres
serverPool
10.0.0.1
10.0.0.2
10.0.0.3
255.0....
512
10.0.0.2
0.0.0.0
Top

Output

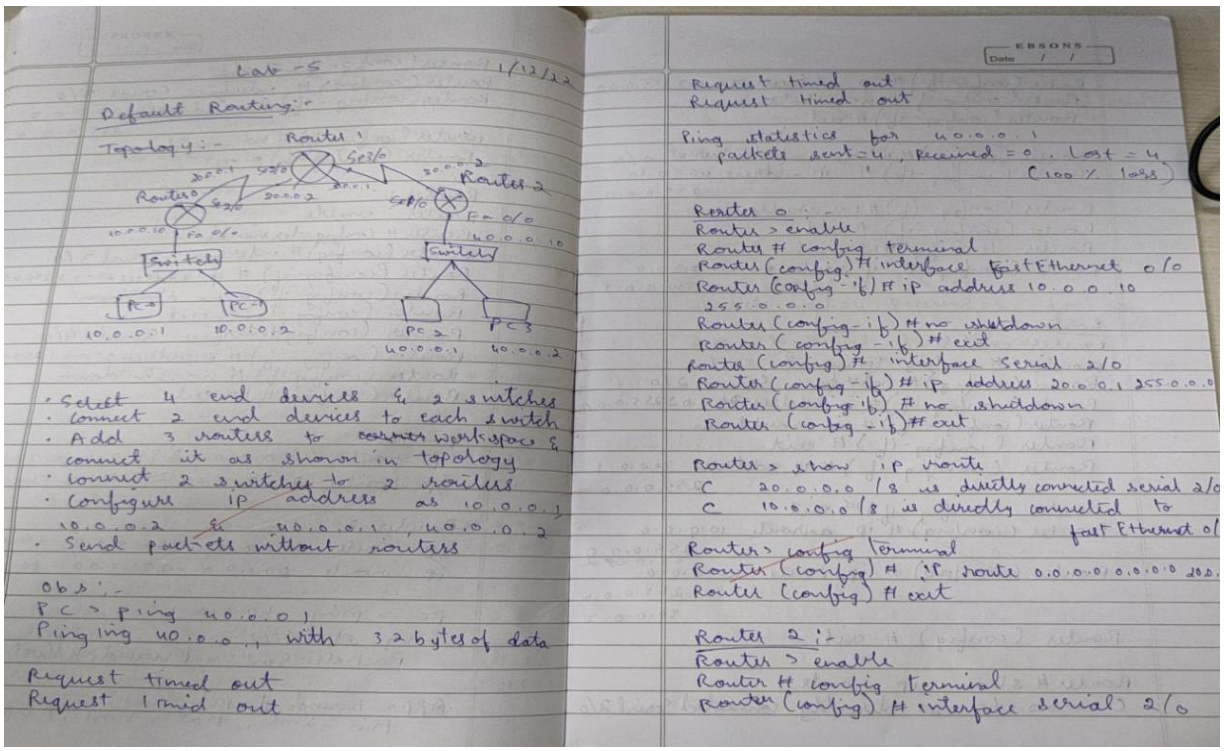

PC0
Physical
Config
Desktop
Attributes
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.6
Pinging 10.0.0.6 with 32 bytes of data:
Reply from 10.0.0.6: bytes=32 time=1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Ping statistics for 10.0.0.6:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
C:\>

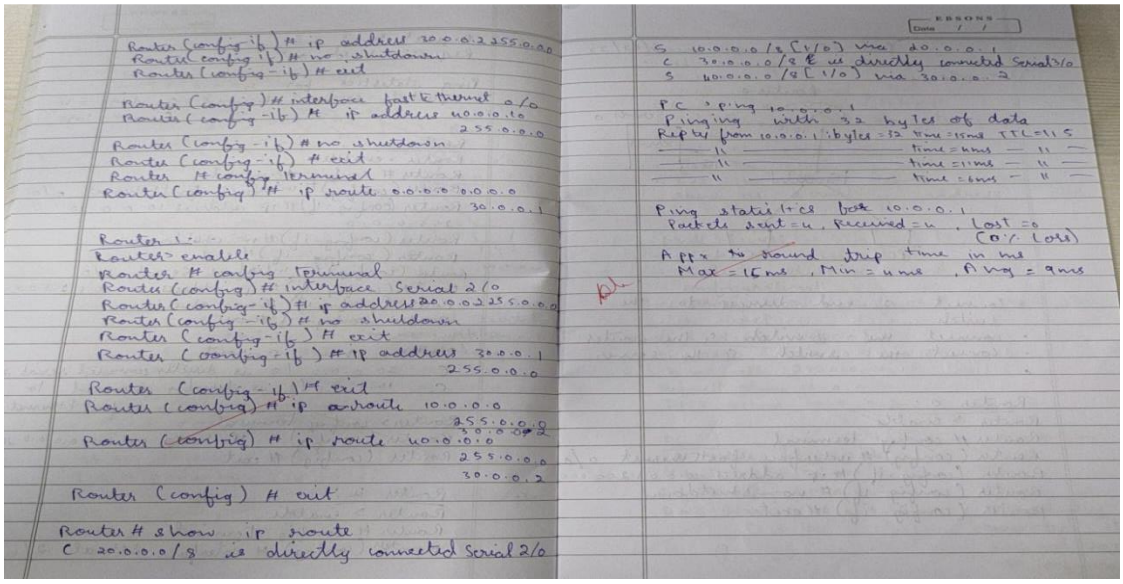
Experiment No 5 Aim of the program

Configuring default router to router

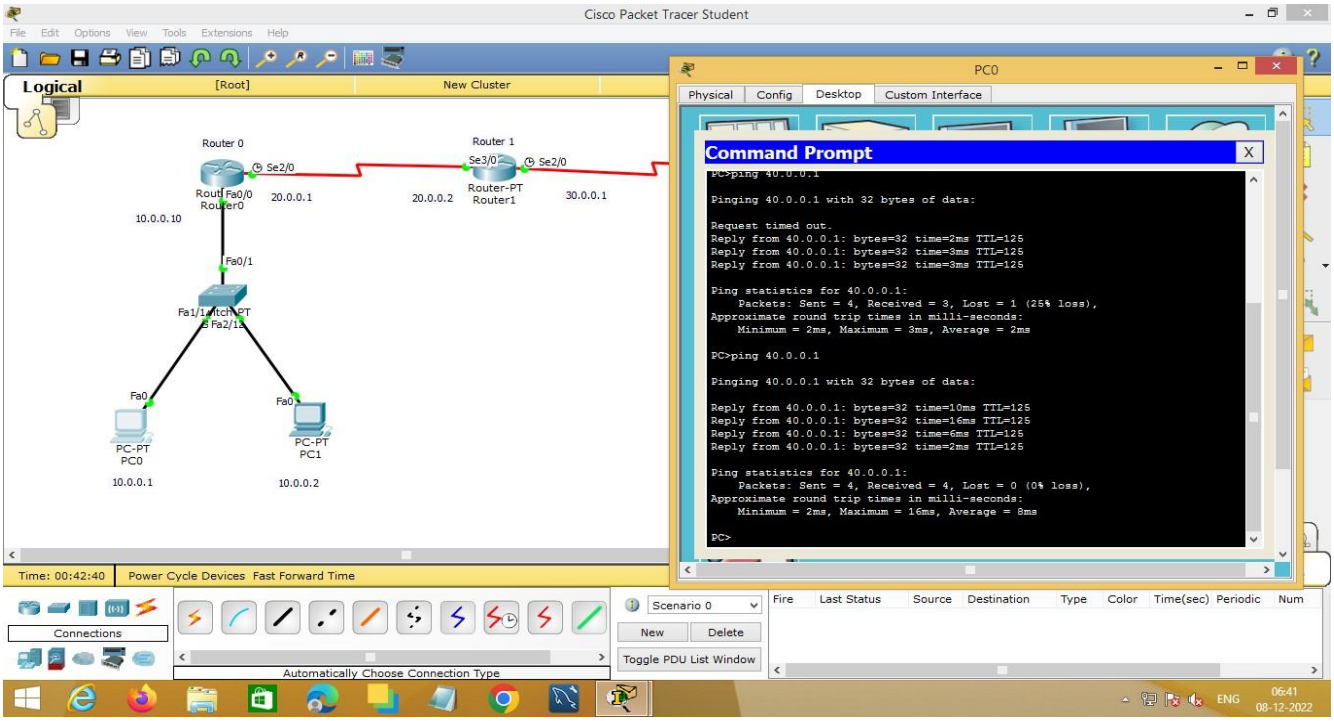
Topology

Procedure




Router (config-if) # ip address 20.0.0.2 255.0.0.0
Router (config-if) # no shutdown
Router (config-if) # exit
Router (config) # interface fast E thernet 0/0
Router (config-if) # ip address 40.0.0.10
255.0.0.0
Router (config-if) # no shutdown
Router (config-if) # exit
Router # config terminal
Router (config) # ip route 0.0.0.0 0.0.0.0
30.0.0.1
Router 1:
Router> enable
Router # config terminal
Router (config) # interface Serial 2/0
Router (config-if) # ip address 30.0.0.2 255.0.0.0
Router (config-if) # no shutdown
Router (config-if) # exit
Router (config-if) # ip address 30.0.0.1
255.0.0.0
Router (config-if) # exit
Router (config) # ip route 10.0.0.0
255.0.0.0 30.0.0.2
Router (config) # ip route 40.0.0.0
255.0.0.0
30.0.0.2
Router (config) # exit
Router # show ip route
C 20.0.0.0/8 is directly connected Serial 2/0
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 30.0.0.0/8 is directly connected Serial 2/0
S 40.0.0.0/8 [1/0] via 30.0.0.2
PC > ping 40.0.0.1
Pinging with 32 bytes of data
Reply from 40.0.0.1: bytes=32 time=15ms TTL=115
time=4ms
time=11ms
time=6ms
Ping statistics for 40.0.0.1
Packets sent = 4, Received = 4, Lost = 0
(0% Loss)
Approx round trip time in ms
Max = 15ms, Min = 4ms, Ping = 9ms

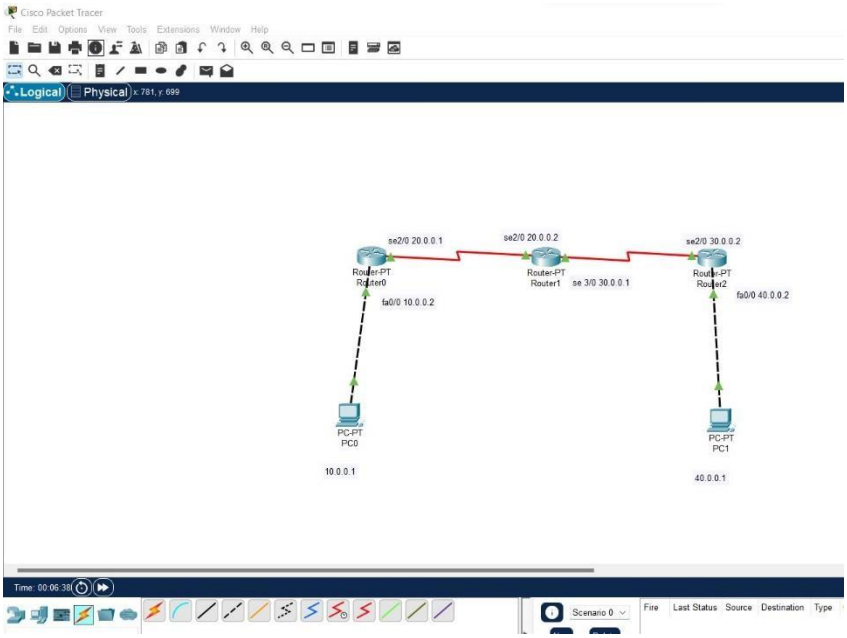
Output

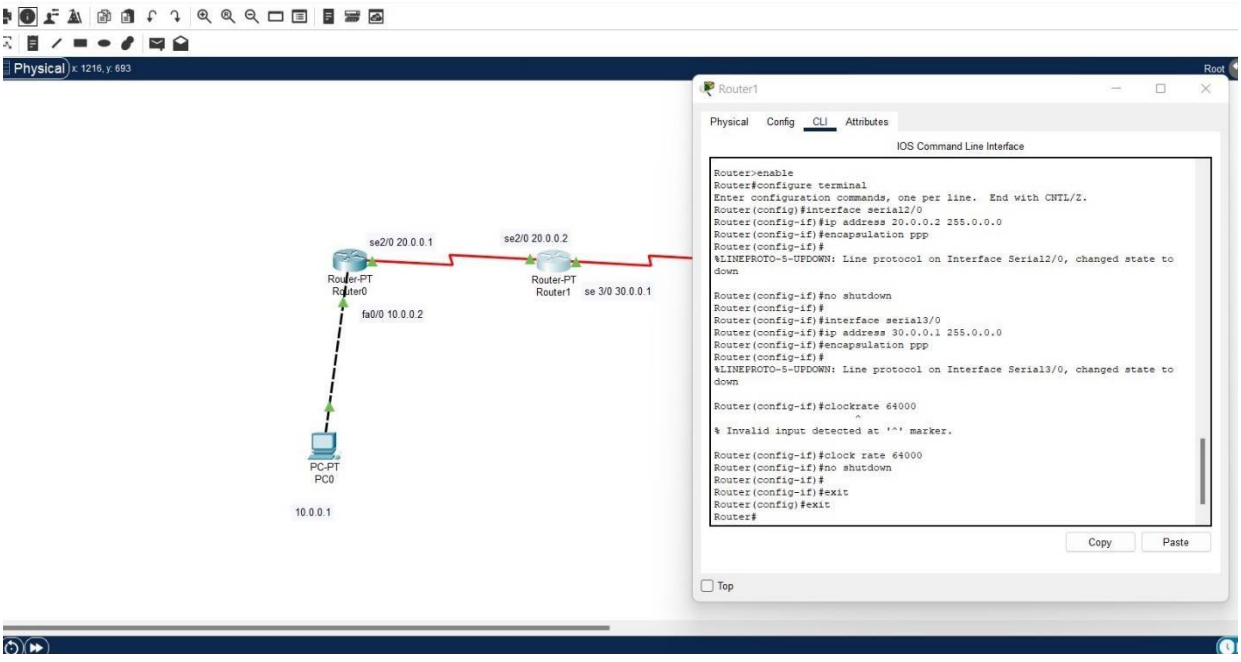

Cisco Packet Tracer Student
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Router 0
Router 1
Se2/0
Se3/0
Router-PT
Router1
Router0
Fa0/0
20.0.0.1
20.0.0.2
30.0.0.1
10.0.0.10
Fa0/1
Fa1/1
Switch-PT
Fa2/1
Fa0
PC-PT
PC0
PC1
10.0.0.1
10.0.0.2
PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=3ms TTL=125
Reply from 40.0.0.1: bytes=32 time=3ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 3ms, Average = 2ms
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
Reply from 40.0.0.1: bytes=32 time=16ms TTL=125
Reply from 40.0.0.1: bytes=32 time=6ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 16ms, Average = 8ms
PC>
Time: 00:42:40
Power Cycle Devices Fast Forward Time
Connections
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
ENG
06:41
08-12-2022

Experiment No 6 Aim of the program

Configuring RIP Routing Protocol in Routers

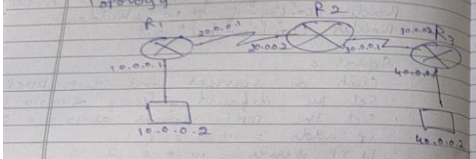
Topology





Lab - 7 15/12/22

Topology



R1 :-
R1(config) # interface fastEthernet 0/0
ip address 10.0.0.1 255.0.0.0
no shutdown

R1(config) # interface serial 2/0
ip address 20.0.0.1 255.0.0.0
encapsulation PPP
clock rate 64000
no shutdown

R2 :-
R2(config) # interface serial 2/0
ip address 20.0.0.2 255.0.0.0
encapsulation PPP
no shutdown

R2(config) # interface serial 3/0
ip address 30.0.0.1 255.0.0.0
encapsulation PPP
clock rate 64000
no shutdown

R3 :-
R3(config) # interface serial 2/0
ip address 30.0.0.2 255.0.0.0
encapsulation PPP
no shutdown

R3(config) # interface fastEthernet 0/0
ip address 40.0.0.1 255.0.0.0
no shutdown

R1 :-
R1(config) # router rip
R1(config-route) # network 10.0.0.0
network 20.0.0.0
exit

R2 :-
R2(config) # router rip
R2(config-route) # network 20.0.0.0
network 30.0.0.0
exit

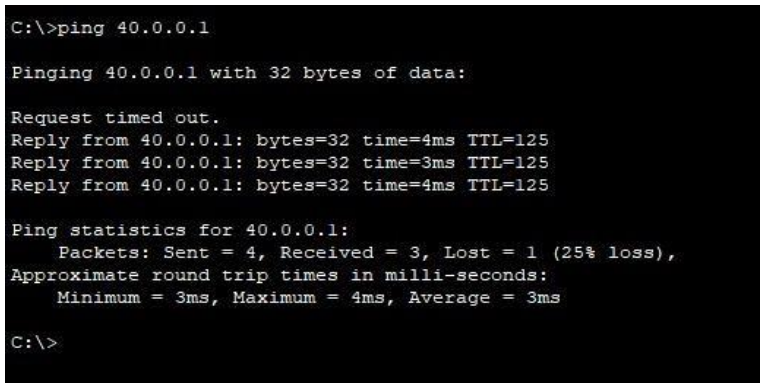
R3 :-
R3(config) # router rip
R3(config-route) # network 30.0.0.0
network 40.0.0.0
exit

Obs :-
PC > ping 40.0.0.2
Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2 : bytes = 32 time = 0ms TTL = 125
Reply from 40.0.0.2 : bytes = 32 time = 10ms TTL = 125
Reply from 40.0.0.2 : bytes = 32 time = 12ms TTL = 125
Reply from 40.0.0.2 : bytes = 32 time = 12ms TTL = 125

Ping statistics for 40.0.0.2 :
Packets : Sent = 4, Received = 4, Loss = 0
Approx round trip in ms
Min = 0ms, Max = 12ms, Avg = 10ms

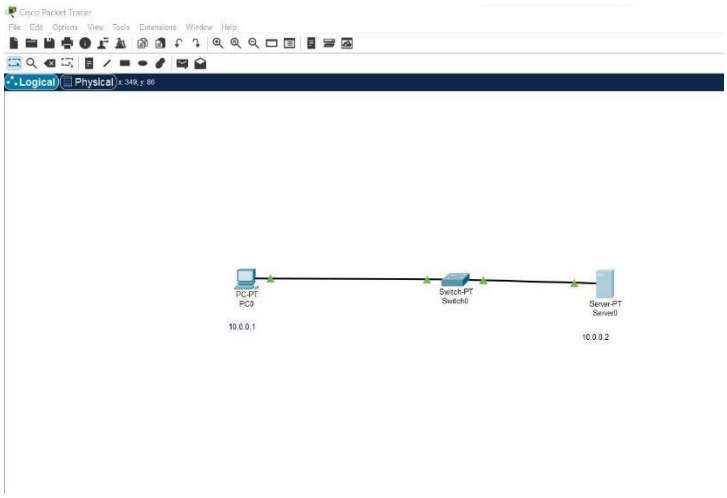
Output


C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125
Reply from 40.0.0.1: bytes=32 time=3ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 4ms, Average = 3ms
C:\>

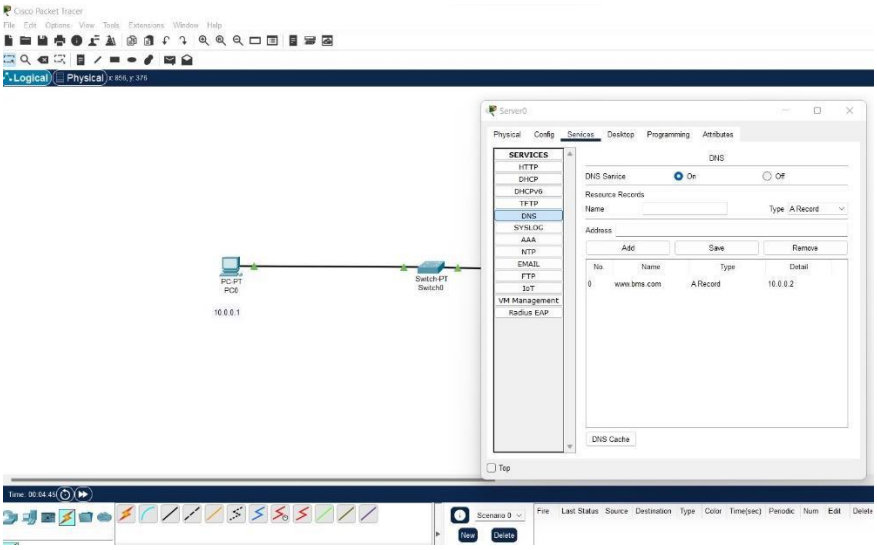
Experiment No 7 Aim of the program

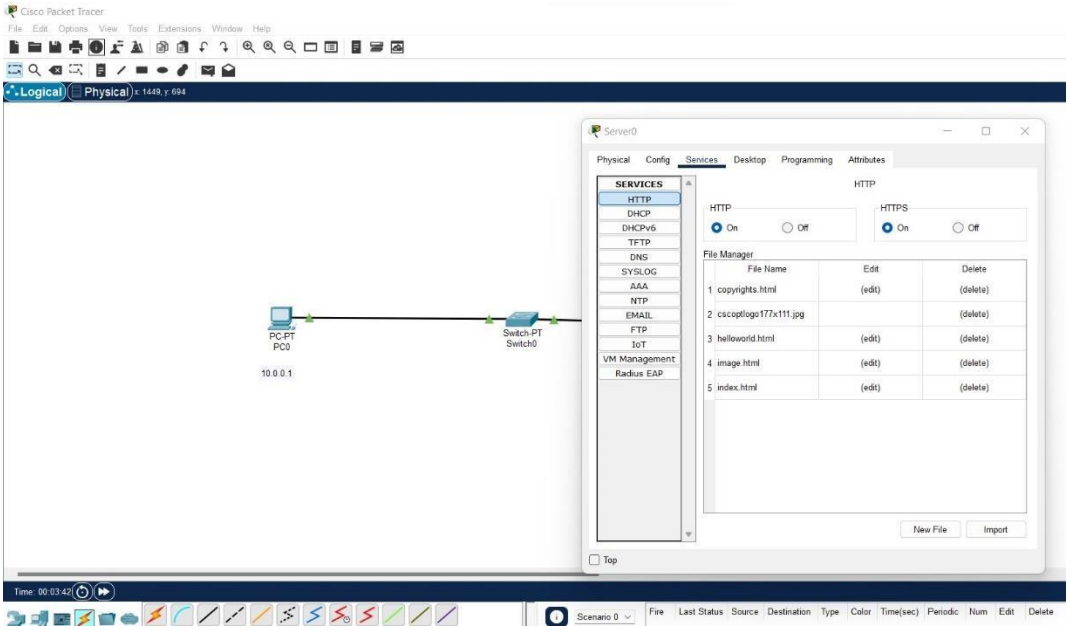
Demonstration of WEB server and DNS using Packet Tracer

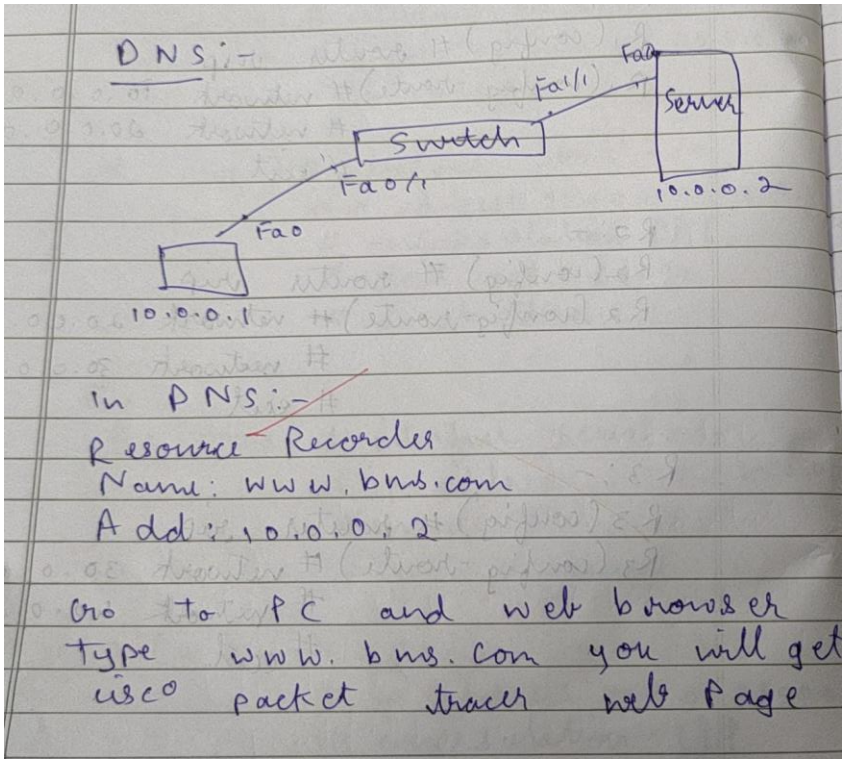
Topology

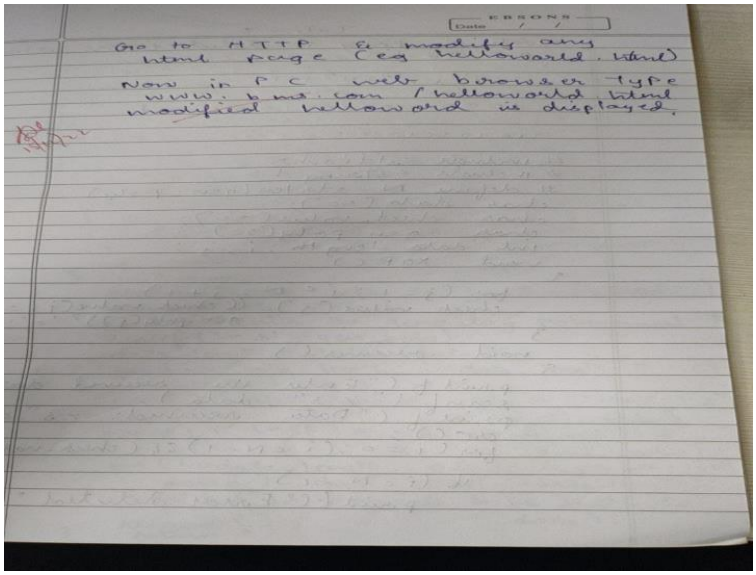


Procedure

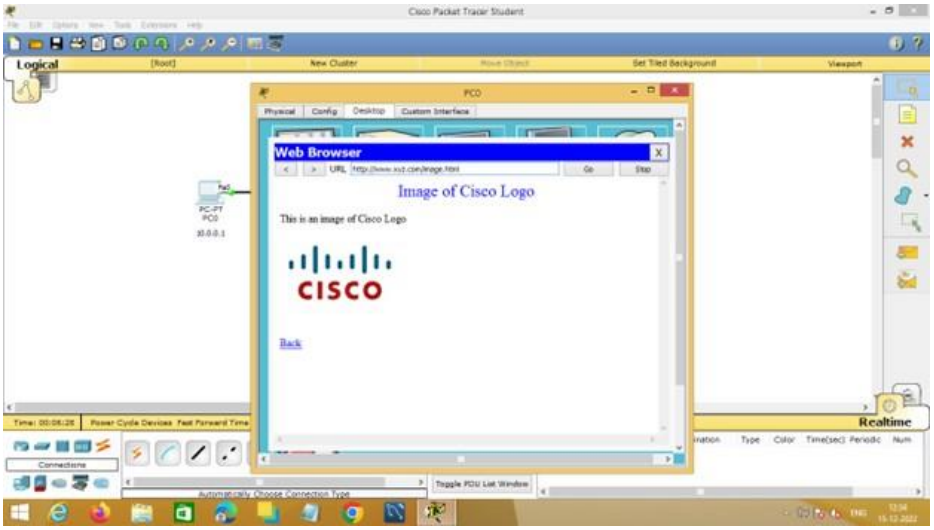



Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical
Physical
Server0
Physical Config Services Desktop Programming Attributes
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoT
VM Management
Radius EAP
HTTP
HTTP
On
Off
HTTPS
On
Off
File Manager
File Name
Edit
Delete
1 copyrights.html (edit) (delete)
2 cscoptlogo177x111.jpg (delete)
3 helloworld.html (edit) (delete)
4 image.html (edit) (delete)
5 index.html (edit) (delete)
New File
Import
Top
PC-PT
PC0
10.0.0.1
Switch-PT
Switch0
Time: 00:03:42
Scenario 0
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete


DNS:-
Fa0
Fa1/1
Server
Switch
Fa0/1
10.0.0.2
Fa0
10.0.0.1
In DNS:-
Resource Recorder
Name: www.bns.com
Add: 10.0.0.2
Go to PC and web browser
type www.bns.com you will get
cisco packet tracer web page


Go to HTTP & modify any
html page (eg helloworld.html)
Now in PC web browser type
www.bms.com/helloworld.html
modified helloworld is displayed.

Output


Cisco Packet Tracer Student
Logical
PC0
Physical
Config
Desktop
Custom Interface
Web Browser
URL
Go
Stop
Image of Cisco Logo
This is an image of Cisco Logo
CISCO
Back
Realtime

Cycle-2 Experiment No 1 Aim of the Experiment

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

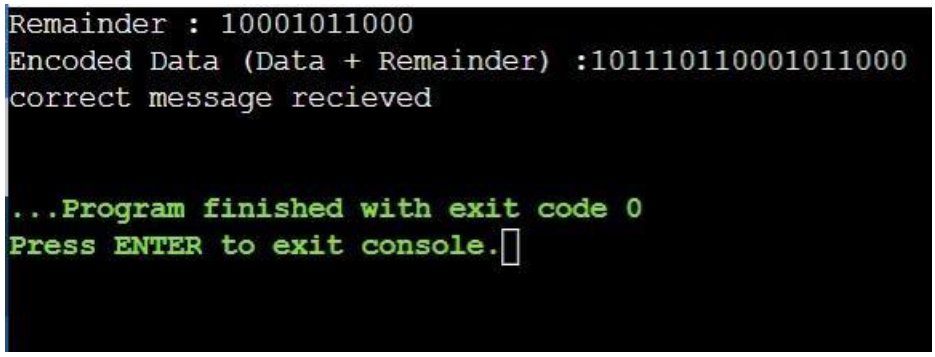
Code

```
#include<bits/stdc++.h> using namespace
std; void receiver(string
data, string key);

string xor1(string a, string b)
{

        string result = "";

        int n = b.length();

        for(int i = 1; i < n; i++)
        {
                if (a[i] == b[i])
                        result += "0";
else
                        result += "1";
        }
        return result;
}

string mod2div(string dividend, string divisor)
{
        int pick = divisor.length();
```

```
        string tmp = dividend.substr(0, pick);

        int n = dividend.length();

         while (pick < n)
        {
                if (tmp[0] == '1')                              tmp =
xor1(divisor, tmp) + dividend[pick];
                else
                        tmp = xor1(std::string(pick, '0'), tmp) +
                                dividend[pick];

                 pick += 1;
        }
        if (tmp[0] == '1')
 tmp = xor1(divisor, tmp);
        else
                 tmp = xor1(std::string(pick, '0'), tmp);

        return tmp;
}

void encodeData(string data, string key)
{
         int l_key = key.length();

            string appended_data = (data +std::string(l_key - 1, '0'));

         string remainder = mod2div(appended_data, key);
```

```
        string codeword = data + remainder;
cout << "Remainder : "
                << remainder << "\n";
          cout << "Encoded Data (Data + Remainder) :"
                << codeword << "\n";
         receiver(codeword, key);
}
void receiver(string data, string key)
{       string currxor = mod2div(data.substr(0, key.size()), key);
int curr = key.size();      while (curr != data.size())
        {
                if (currxor.size() != key.size())
                {
                        currxor.push_back(data[curr++]);
                }
                else
                {
                        currxor = mod2div(currxor, key);
                }
        }
          if (currxor.size() == key.size())
        {
                currxor = mod2div(currxor, key);
        }
        if (currxor.find('1') != string::npos)
        {       cout << "there is some error in data" << endl;
        }
        else
```

```
        {
                cout << "correct message recieved" << endl;
        }
} int
main()
{
        string data = "1011101";
 string key = "100010000001";

         encodeData(data, key);

        return 0;
}
```



Experiment No 2 Aim of the Experiment

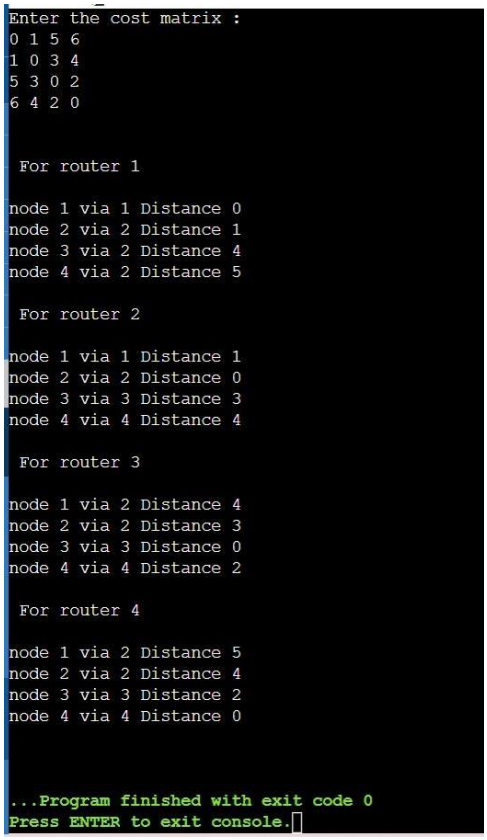
Write a program for distance vector algorithm to find suitable path for transmission.

Code

```
# include<stdio.h>
                    struct node
                    {
          unsigned dist[20];           unsigned
from[20];
      }rt[10];      int main()
                    {
          int costmat[20][20];         int nodes,i,j,k,count=0;
          printf("\nEnter the number of nodes : ");
scanf("%d",&nodes);//Enter the nodes          printf("\nEnter the
cost matrix :\n");      for(i=0;i<nodes;i++)
                        {
                            for(j=0;j<nodes;j++)
                            {
                  scanf("%d",&costmat[i][j]);
costmat[i][i]=0;
                                rt[i].dist[j]=costmat[i][j];//initialise the distance equal
                    to cost matrix
                                rt[i].from[j]=j;
                            }
                        }
                            do
                            {
                                count=0;
                                for(i=0;i<nodes;i++)//We choose arbitary vertex k and
                    we calculate the direct distance from the node i to k using the
                    cost
                    matrix
                  //and add the distance from k to node j
for(j=0;j<nodes;j++)                    for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                                    {//We calculate the minimum distance
                                        rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];


                           rt[i].from[j]=k;
count++;
                                    }
              }while(count!=0);             for(i=0;i<nodes;i++)
                                {
```

```
                    printf("\n\n For router %d\n",i+1);
 for(j=0;j<nodes;j++)
                                    {
                                        printf("\t\nnode %d via %d Distance %d
                      ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
                                    }
                                }
           printf("\n\n");       getch();


                      }
```

Output



Experiment No 3 Aim of the Experiment

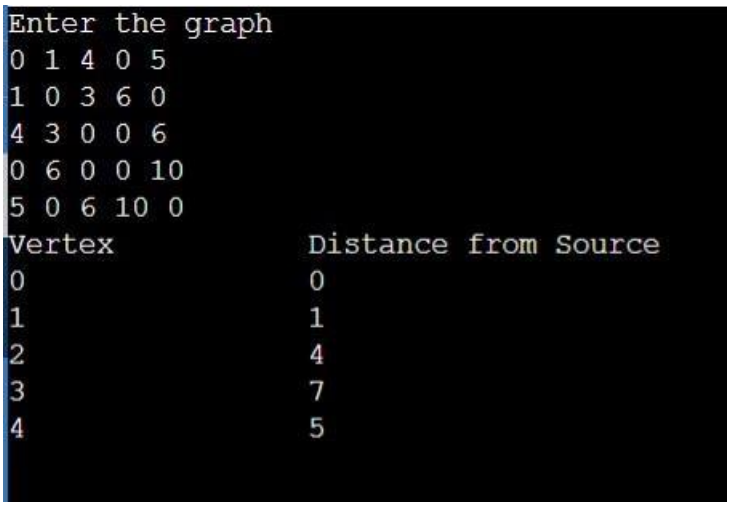
Implement Dijkstra's algorithm to compute the shortest path for a given topology.

Code

```
# include<bits/stdc++.h>
      #include <limits.h>   #include <stdio.h>
using namespace std;


                          #define V 5



                          int minDistance(int dist[], bool Test[])
      {                         int min = INT_MAX, min_index;


          for (int v = 0; v < V; v++)             if ( Test[v] == false &&
dist[v] <= min)                  min = dist[v], min_index = v;


                               return min_index;
                                                                          }


                          void printSolution(int dist[])
                          {
          printf("Vertex \t\t Distance from Source\n");             for
(int i = 0; i < V; i++)             printf("%d \t\t %d\n", i, dist[i]);
                          }



                          void dijkstra(int graph[V][V], int src)
                          {
          int dist[V];          bool Test[V];
          for (int i = 0; i < V; i++)             dist[i] = INT_MAX,
Test[i] = false;                        dist[src] = 0;


                        for (int count = 0; count < V - 1; count++) {
                            int u = minDistance(dist, Test);



                                  Test[u] = true;


                            for (int v = 0; v < V; v++)
```

```
                                if (!Test[v] && graph[u][v] && dist[u] != INT_MAX
&& dist[u] + graph[u][v] < dist[v])                         dist[v] = dist[u] +
graph[u][v];
                            }



                            printSolution(dist);
                                                                        }


                        int main()

                                                                        {
                            int graph[V][V] ;
     cout<<"Enter the graph "<<endl;      for(int i = 0; i<V; i++)
                            {
             for(int j = 0; j<V; j++)                cin>>graph[i][j];
                            }
                                                                  dijkstra(graph, 0);


                            return 0;
                        }
```

Output



Experiment No 4 Aim of the Experiment

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

Server:

```
from socket import * serverName = '' serverPort = 12530

serverSocket = socket(AF_INET,SOCK_STREAM)

serverSocket.bind((serverName,serverPort))

serverSocket.listen(1) print("The server is ready to

receive") while 1:

   connectionSocket, addr = serverSocket.accept()     sentence

= connectionSocket.recv(1024).decode()     try:

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

connectionSocket.send(l.encode())

file.close()     except Exception as e:

message = "No such file exist"

connectionSocket.send(message.encode())     connectionSocket.close()
```

Client:

```
from socket import * serverName =

'192.168.1.104' serverPort = 12530  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 ('From Server:',

filecontents) clientSocket.close()
```

Experiment No 5 Aim of the Experiment

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

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)


    file=open(sentence,"r")     l=file.read(2048)


    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
print("sent back to client",l)     file.close() Client:  from
socket import * serverName = "127.0.0.1" serverPort
= 12000 clientSocket = socket(AF_INET,
SOCK_DGRAM)


sentence = input("Enter file name") clientSocket.sendto(bytes(sentence,"utf-8"),(serverName,
serverPort)) filecontents,serverAddress = clientSocket.recvfrom(2048) print ('From Server:', filecontents)


clientSocket.close()
```