VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

VENKATA SATWIK POTULA (1BM20CS183)

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 **VENKATA SATWIK POTULA (1BM20CS183),** who is a 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. Nandhini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

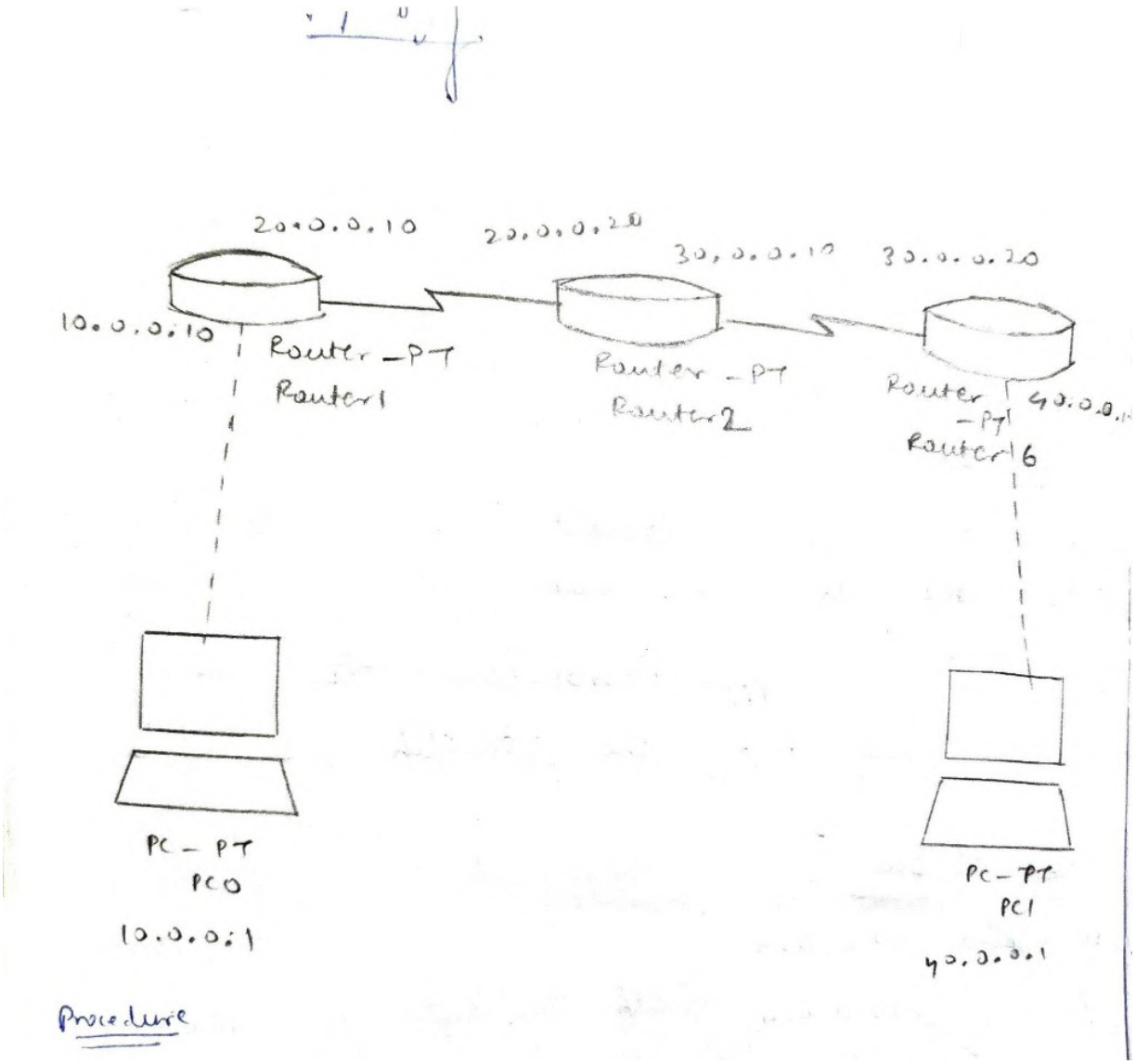
`

Index

Experiment No. 1

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

Topology:



Procedure:

Procedure :-

→ A hub is a dumb device i.e. it is not intelligent. So, whatever data is sent from a sender to the hub is then broadcasted to all devices.

→ Time it takes to send from sender to receiver is about 0.002 seconds.

→ A default type connection is used.

→ Device and hub is connected using a copper straight cable.

→ It will broadcast even to devices which do not require the data.

Output:

for Hub:

```
PC>ping 10.0.0.2 (input)
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=10ms TTL=128
Reply from 10.0.0.2: bytes=32 time=4ms TTL=128
Reply from 10.0.0.2: bytes=32 time=4ms TTL=128
Reply from 10.0.0.2: bytes=32 time=4ms TTL=128
Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 10ms, Average = 5ms
```

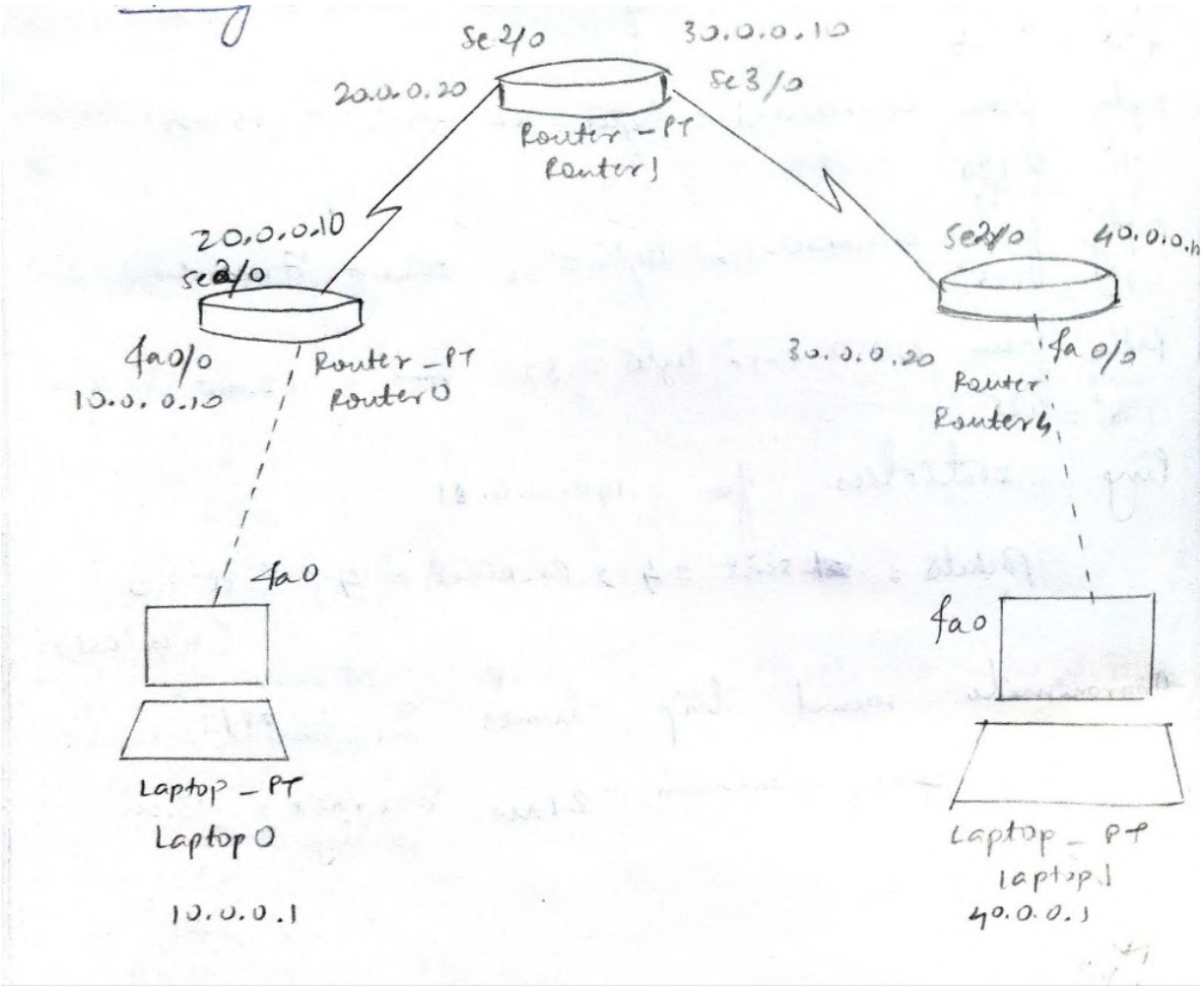
for Switch:

```
PC>ping 10.0.0.71
Pinging 10.0.0.71 with 32 bytes of data:
Reply from 10.0.0.71: bytes=32 time=0ms TTL=128
Reply from 10.0.0.71: bytes=32 time=0ms TTL=128
Reply from 10.0.0.71: bytes=32 time=0ms TTL=128
Reply from 10.0.0.71: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.71:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

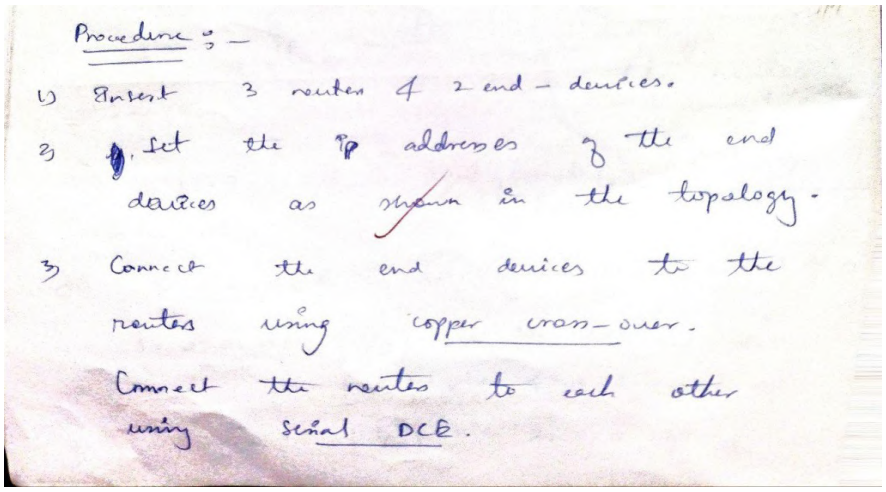
Experiment No. 2

Aim: Configuring IP address to routers in Packet Tracer. Explore the following messages. Ping Responses, Destination unreachable, Request Timed out, Reply

Topology:



Procedure:



Procedure :-

1) Insert 3 routers & 2 end-devices.
2) Set the IP addresses of the end devices as shown in the topology.
3) Connect the end devices to the routers using copper cross-over.
   Connect the routers to each other using serial DCE.

4) Click on the router and go to the CLI.

5) Type in the following commands

Router > enable

Router # config t

~~Router (config) # ip route 0.0.0.0 0.0.0.0 30.0.0.20~~

→ interface fast ethernet 0/0

Router # (config-if)

→ ip address 10.0.0.10 255.0.0.0

Router 2

→ enable

Router #

→ config t

Router # (config)

→ interface serial 2/0

Router # (config-if)

→ ip address 20.0.0.10 255.0.0.0

6) Similarly configure the other routers using the same commands.

---

6) Once done, we will have to implement using the same commands used for static routing, except, we let the computer decide how to establish the connection with the end devices and nodes connected to the other routers.

A

⇒ following are the commands used

Route #

→ config t

Router # (config)

→ ip route 0.0.0.0 0.0.0.0 20.0.0.20

Router # (config)

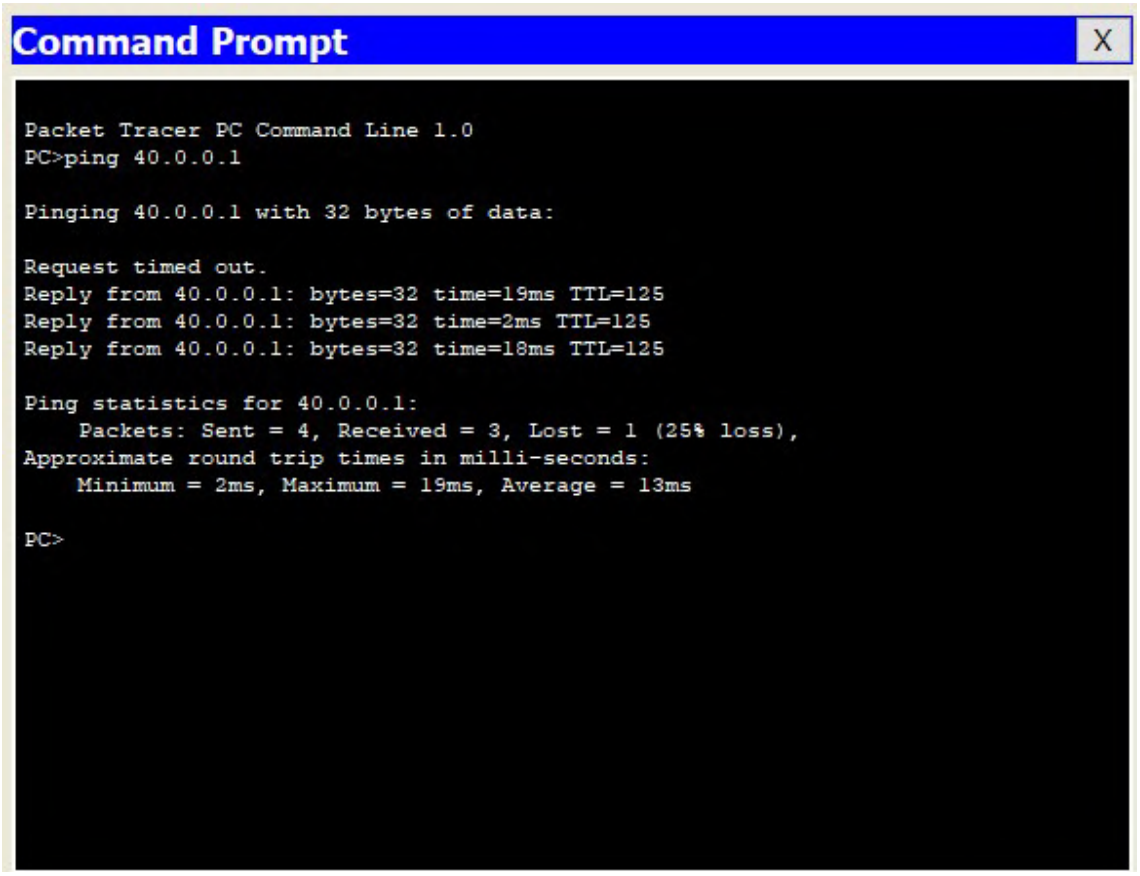
→ exit

Router #

→ show ip route

Similarly add ip route to the other routers using the default address 0.0.0.0

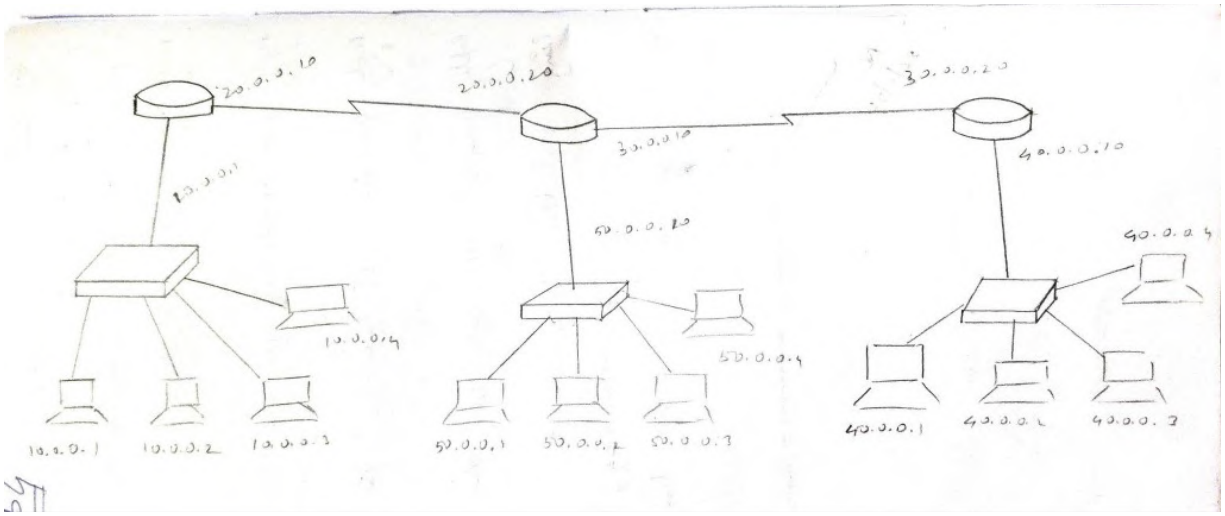
Output:


Command Prompt
X
Packet Tracer PC Command Line 1.0
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=19ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=18ms 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 = 19ms, Average = 13ms
PC>

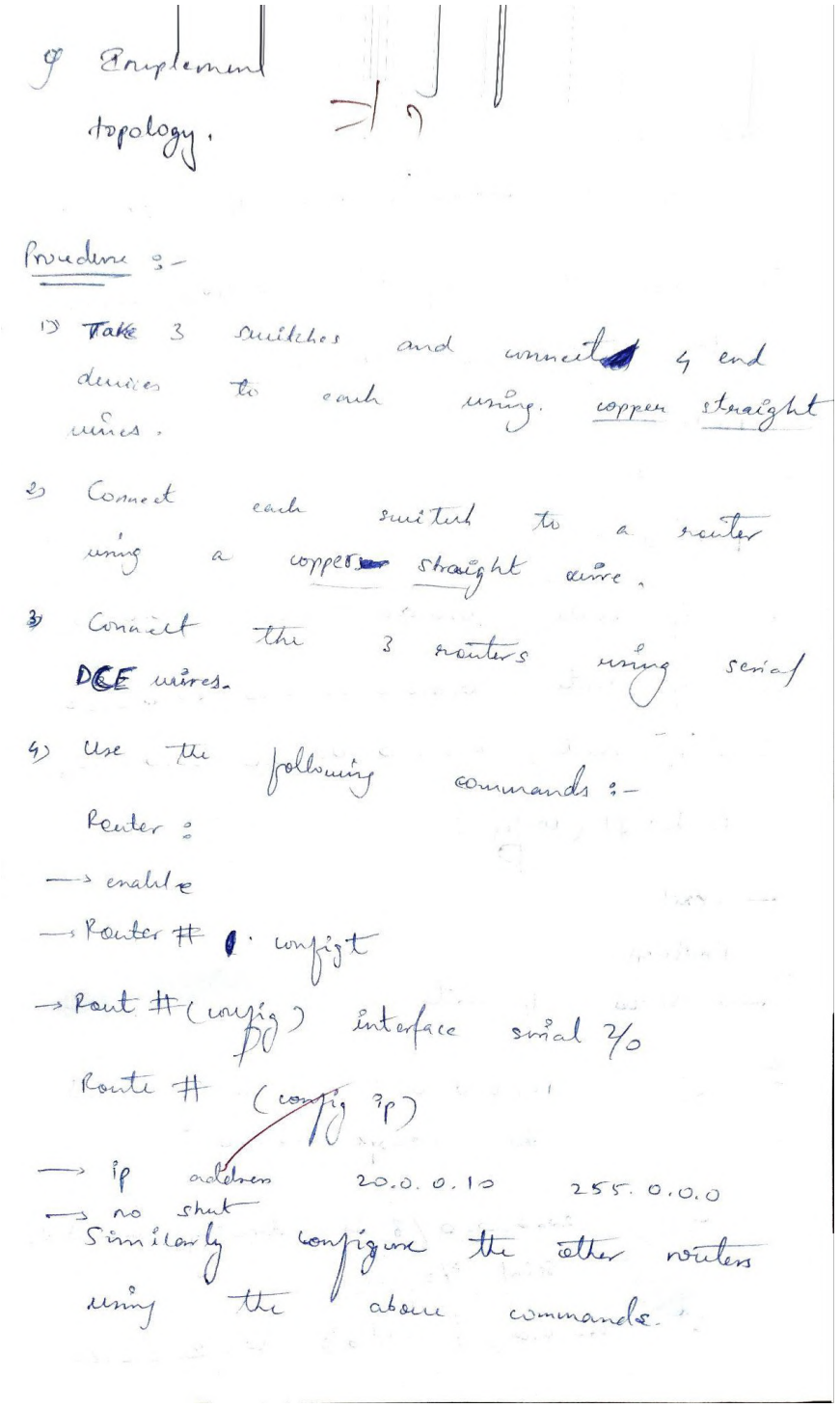
Experiment No. 3

Aim: Configuring default route to the router

Topology:



Procedure:



9 Implement topology.

Procedure :-

1) Take 3 switches and connect 4 end devices to each using copper straight wires.

2) Connect each switch to a router using a copper straight wire.

3) Connect the 3 routers using serial DCE wires.

4) Use the following commands :-

Router :

→ enable

→ Router # configt

→ Rout #(config) interface serial 2/0

Route # (config ip)

→ ip address 20.0.0.10 255.0.0.0

→ no shut

Similarly configure the other routers using the above commands.

2

5) Set the gateways in each end device as equal to their corresponding ~~switch~~ router's ip address.

6) Implement default routing using the following commands:-

Router #

→ config t

Router # (config)

→ ip route 0.0.0.0 0.0.0.0 20.0.0.20

→ ip route 0.0.0.0 0.0.0.0 30.0.0.20

→ ip route 0.0.0.0 0.0.0.0 30.0.0.10

Router # (config)

→ cntl

Router #

→ show ip route

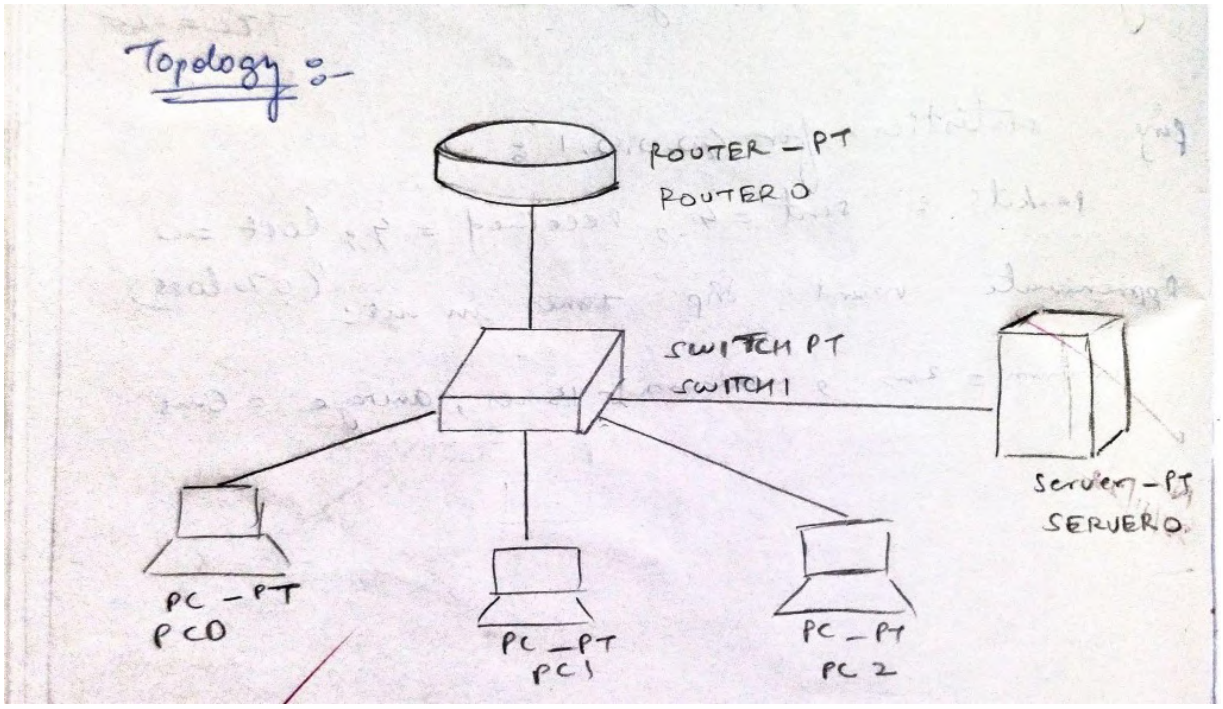
Output:

```
Packet Tracer PC Command Line 1.0
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=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 = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 6ms, Average = 3ms

PC>
```

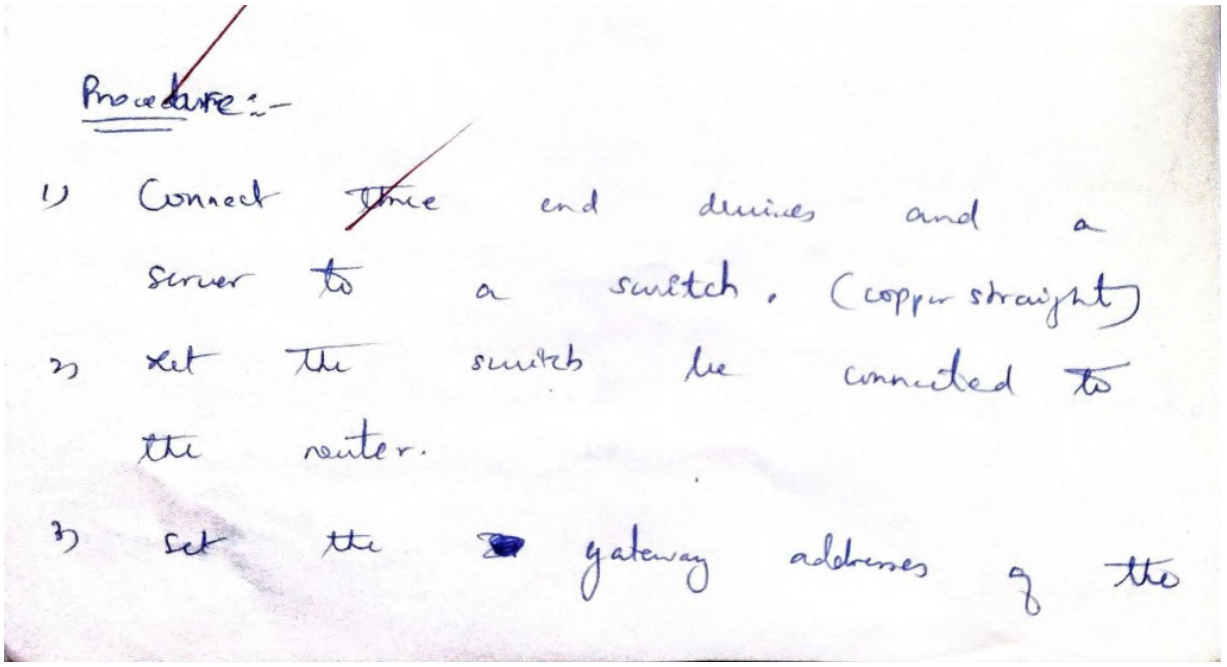
Experiment No. 4

Aim: Configuring DHCP route to the Router

Topology:



Procedure:



Procedure :-

1) Connect three end devices and a server to a switch. (copper straight)
2) Let the switch be connected to the router.
3) Set the gateway addresses of the

end devices as the IP address of the server. (10.0.0.1)

4) The server can be configured as follows :-

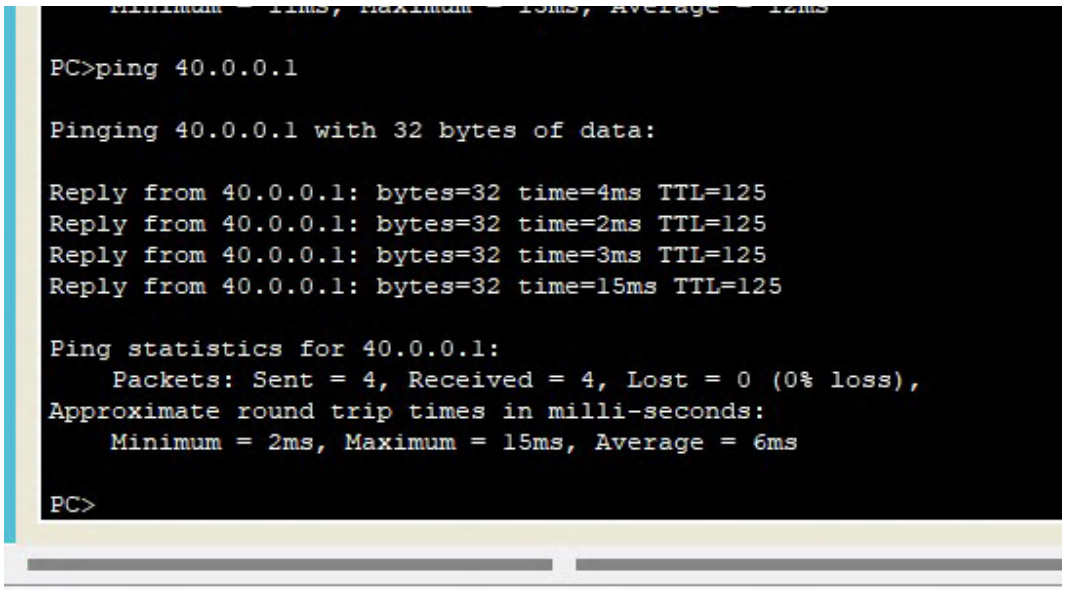
In Server 0's settings,

1) Go to desktop > IP configuration
2) Set the ip configuration as DHCP.
3) Go to services > DHCP and change the following values :-

POOL Name	serverPool
Default gateway	0.0.0.0
DNS server	0.0.0.0
Start IP address	10.0.0.2
Subnet mask	255.0.0.0

5) Now, go to each end device and set the IP addresses to DHCP.

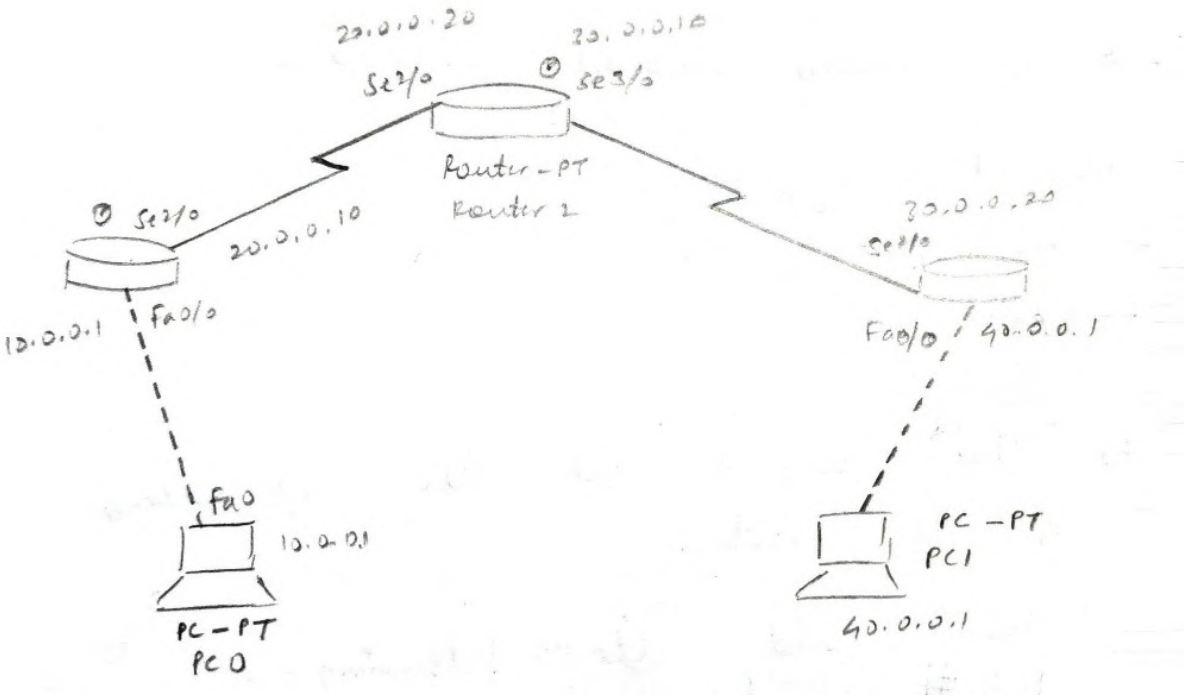
Output:


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=4ms TTL=125
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=15ms 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 = 15ms, Average = 6ms
PC>

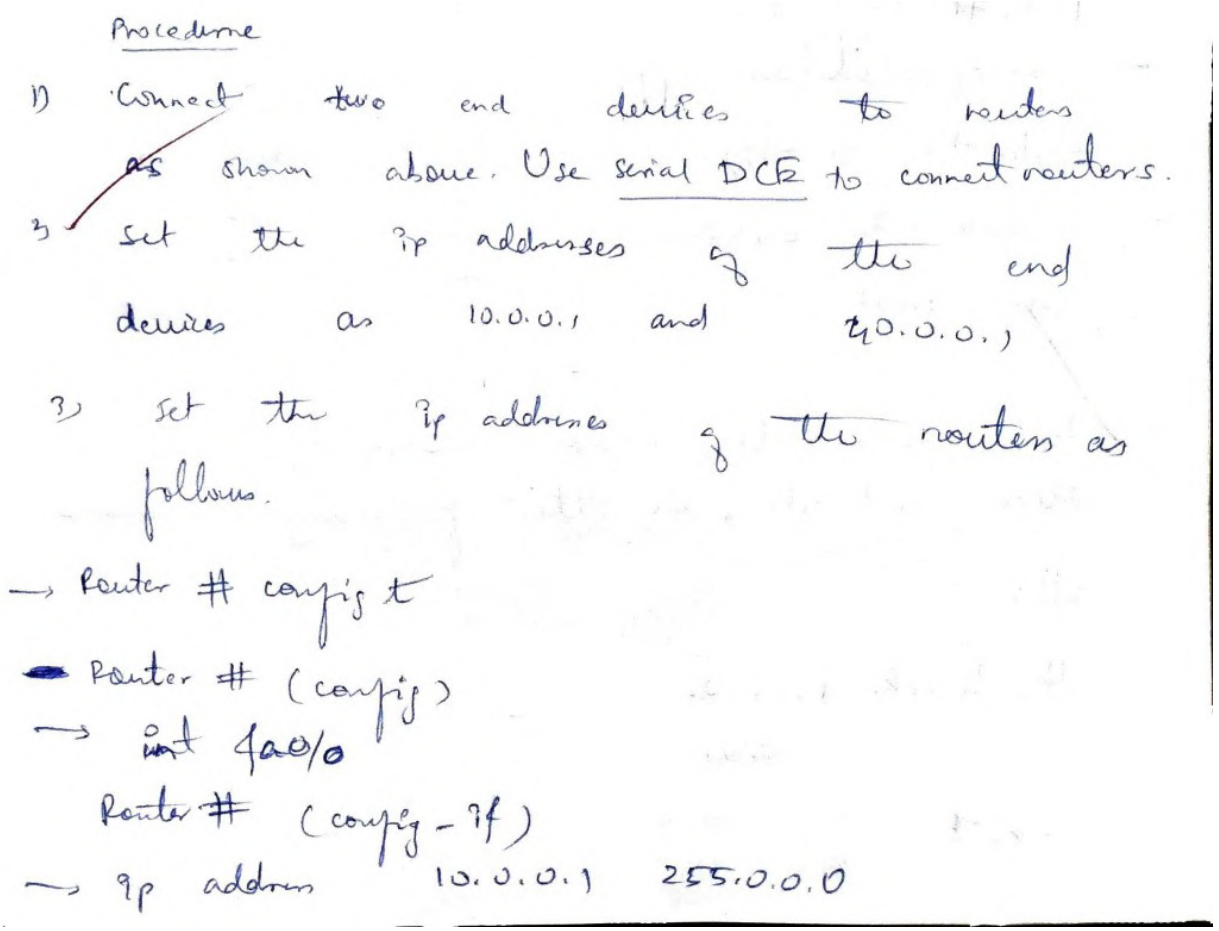
Experiment No. 5

Aim: Configuring RIP Routing Protocol in Routers

Topology:



Procedure:



Procedure

1) Connect two end devices to routers as shown above. Use serial DCE to connect routers.

2) Set the ip addresses of the end devices as 10.0.0.1 and 20.0.0.1

3) Set the ip address of the routers as follows.

→ Router # config t

Router # (config)

→ int fa0/0

Router # (config-if)

→ ip addrss 10.0.0.1 255.0.0.0

→ no shut

A → exit

Router # (config)

→ int se 2/0

Router # (config-if)

→ ip address 20.0.0.10 255.0.0.0

Router # (config-if)

→ no shut

→ exit

4) This would set the ip address of the routers.

Then add the following :-

Router # (config-if)

→ encapsulation PPP

Router # (config-if)

→ clock rate 64,000

→ no shut

} for serial interfaces only.

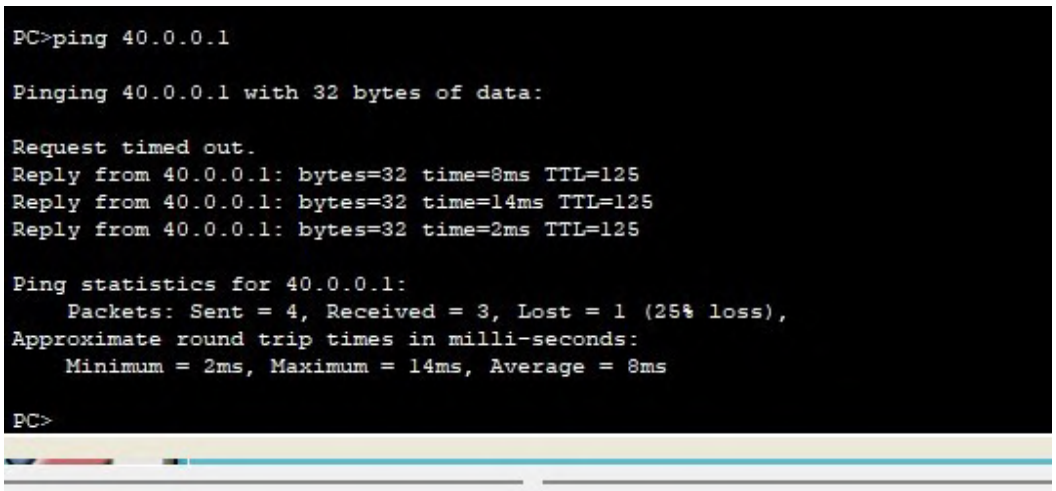
5) To make the router aware of the other networks, use the following commands

router rip

network 10.0.0.0

network 20.0.0.0

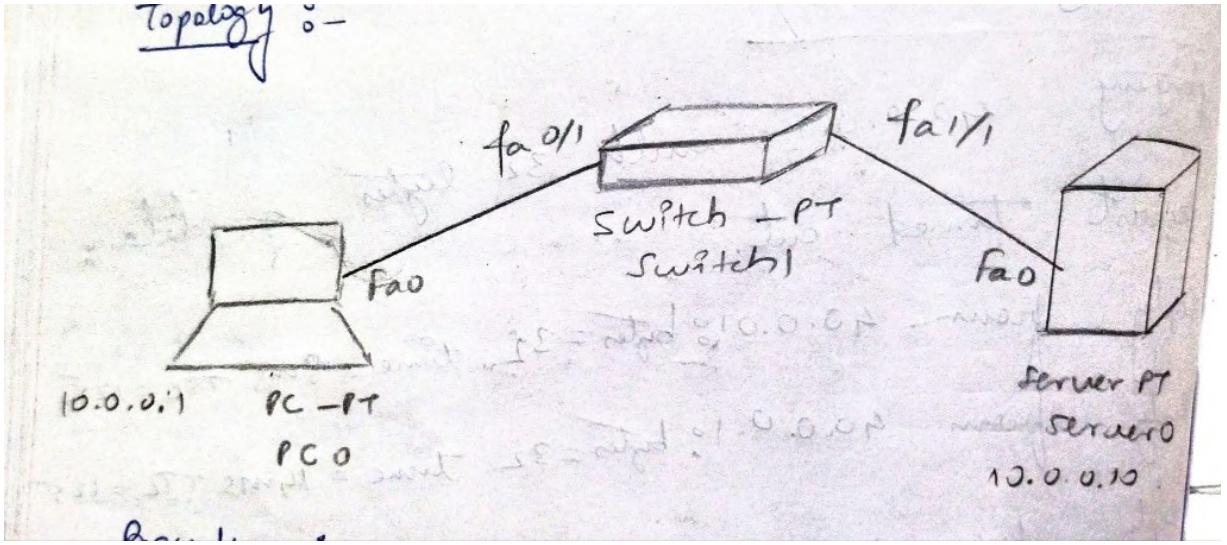
Output:


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=8ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms 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 = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 14ms, Average = 8ms
PC>

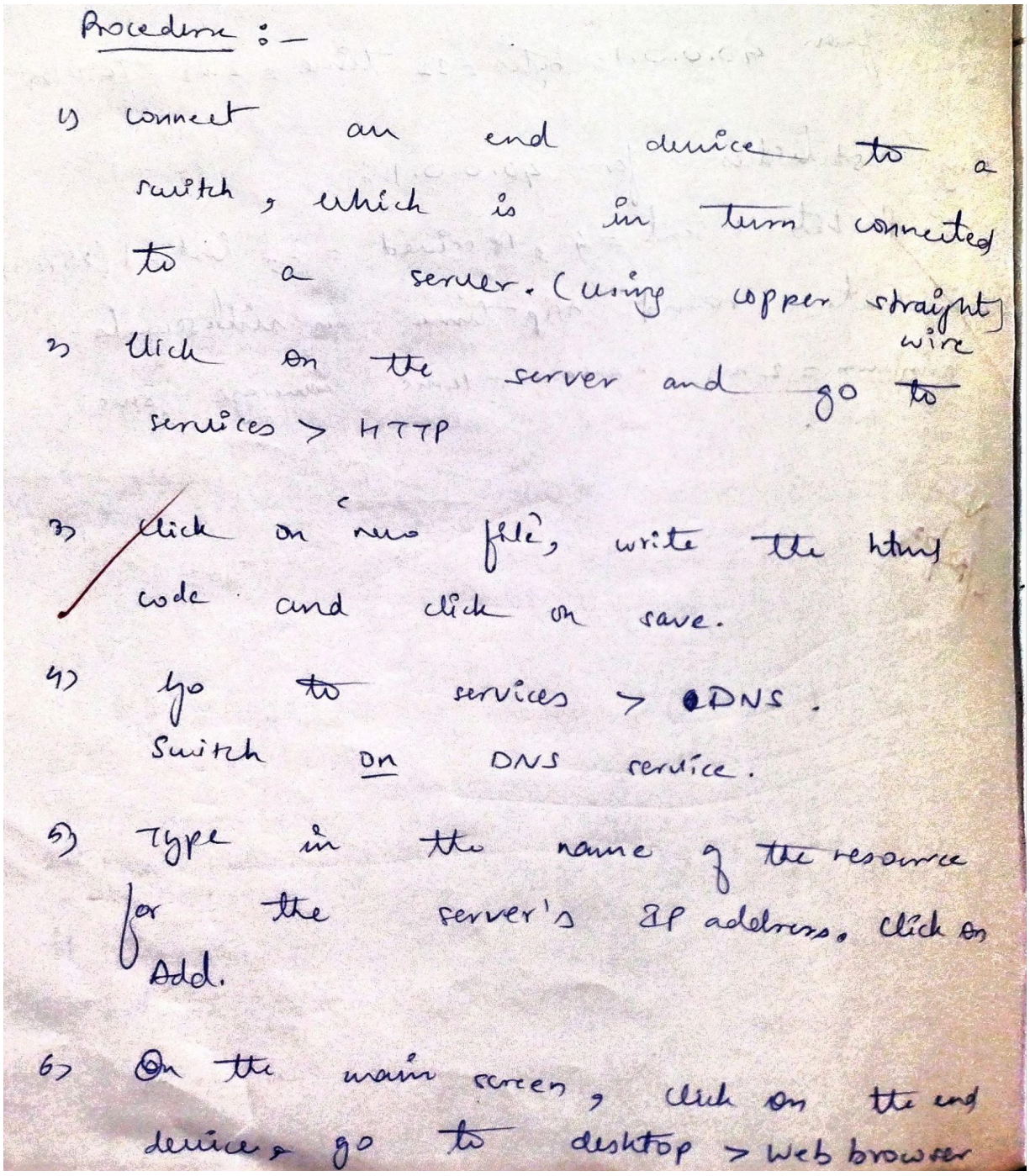
Experiment No. 6

Aim: Demonstration of WEB server and DNS using Packet Tracer

Topology:



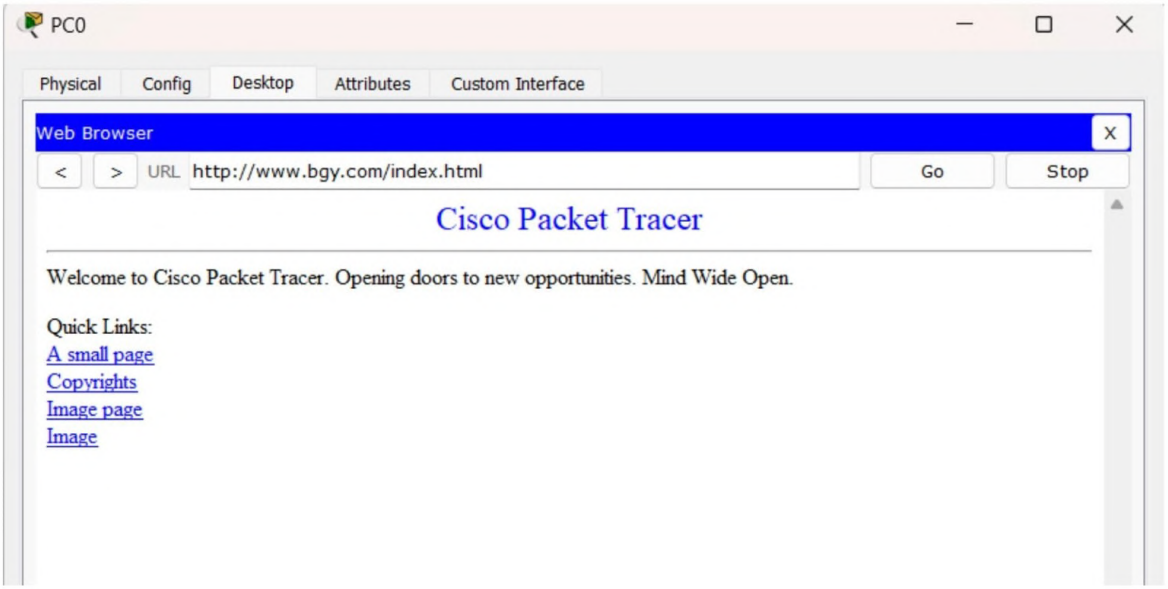
Procedure:



Procedure :-

1) Connect an end device to a switch, which is in turn connected to a server. (using copper straight wire)
2) Click on the server and go to services > HTTP
3) Click on new file, write the html code and click on save.
4) Go to services > DNS. Switch ON DNS service.
5) Type in the name of the resource for the server's IP address. Click on Add.
6) On the main screen, click on the end device, go to desktop > Web browser

Output:


PC0
Physical
Config
Desktop
Attributes
Custom Interface
Web Browser
X
<
>
URL
http://www.bgy.com/index.html
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image

Experiment No. 7

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

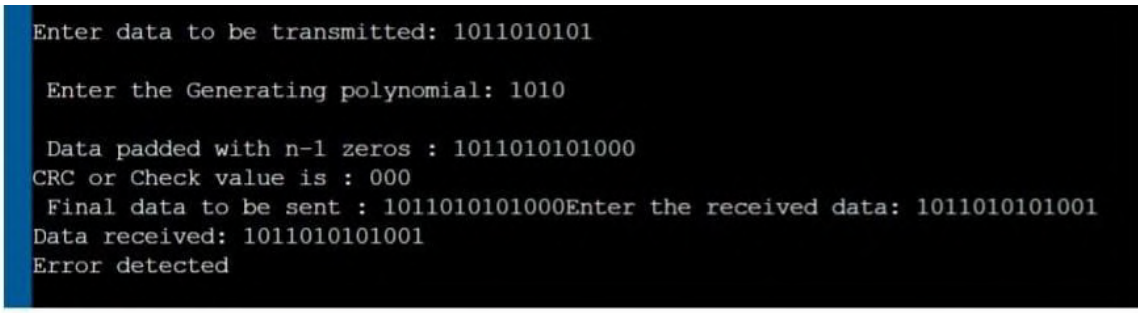
Code:

```
import socket


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)


def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0 : pick]


    while pick < len(dividend):


            if tmp[0] == '1':


                    tmp = xor(divisor, tmp) + dividend[pick]
```

```
            else: # If leftmost bit is '0'

                    tmp = xor('0'*pick, tmp) + dividend[pick]
            pick += 1

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

    checkword = tmp
    return checkword

def encodeData(data, key):

    l_key = len(key)

    appended_data = data + '0'*(l_key-1)
    remainder = mod2div(appended_data, key)

    codeword = data + remainder
    return codeword

s = socket.socket()

port = 12345
s.connect(('127.0.0.1', port))
```

```
input_string = input("Enter data you want to send->")
#s.sendall(input_string)
data =(''.join(format(ord(x), 'b') for x in input_string))
print("Entered data in binary format :",data)
key = "1001"

ans = encodeData(data,key)
print("Encoded data to be sent to server in binary format :",ans)
s.sendto(ans.encode(),('127.0.0.1', 12345))


print("Received feedback from server :",s.recv(1024).decode())

s.close()
```

Output:



Experiment No. 8

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

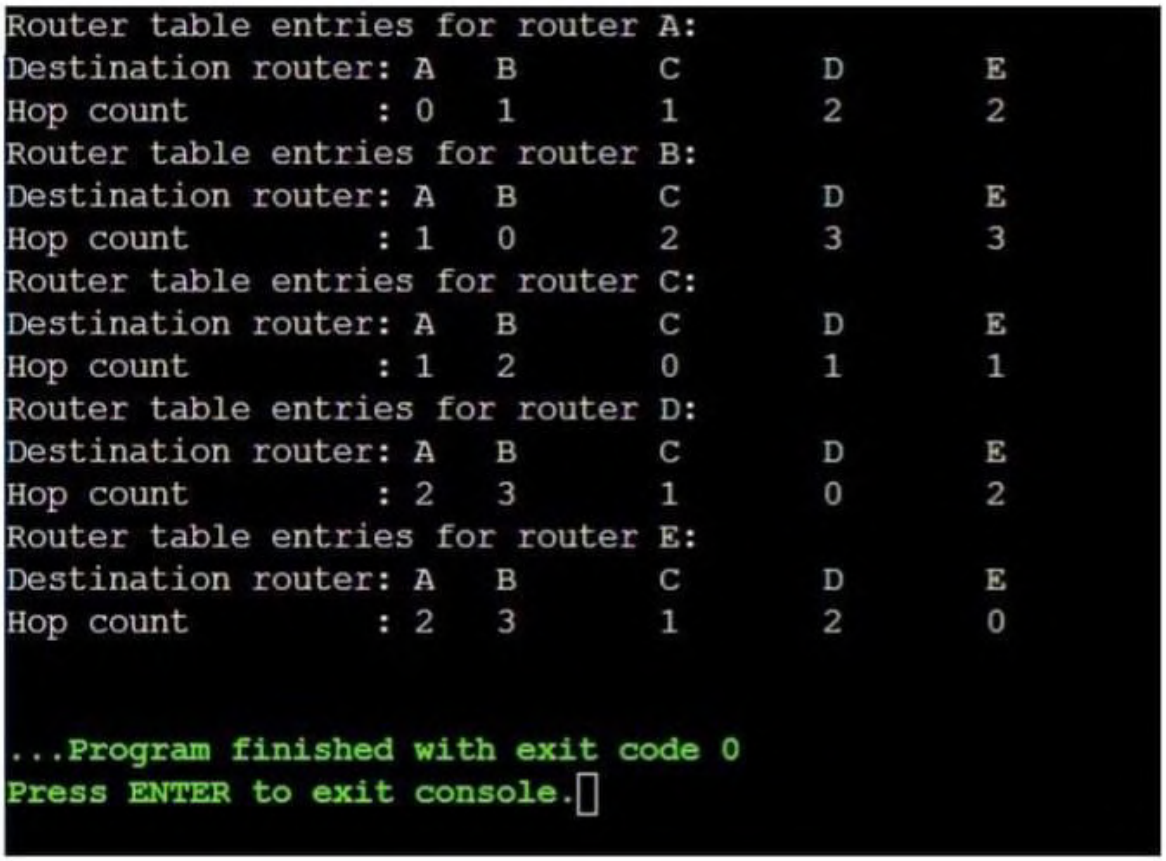
Code:

```
#include <stdio.h>
int n, c[10][10], src;
void dijkstras();

void main()
{
        int i, j;
        printf("\n Enter the no of nodes/devices:\t");
        scanf("%d", &n);

        printf("\nEnter the distance matrix:\n");
        for (i = 1; i <= n; i++)
        {
        for (j = 1; j <= n; j++)
        {
        scanf("%d", &c[i][j]);
        }
        }

        printf("\n Enter the source node:\t");
        scanf("%d", &src);
        dijkstras();
}
```

```
void dijkstras()
{
        int i, j, u, min = 0, count = 0;
        int dist[10], vis[10];

        for (i = 1; i <= n; i++)
        dist[j] = c[src][j];

        for (i = 1; i <= n; i++)
        vis[i] = 0;

        while (count != n)
        {
        min = 9999;

        for (i = 1; i <= n; i++)
        {
        if (dist[i] < min && vis[i] != 1)
        {
                min = dist[i];
                u = i;
        }
        }

        vis[u] = 1;
        count++;
```

```
        for (i = 1; i <= n; i++)
        {
        if (c[u][i] + min < dist[i] && vis[i] != 1)
        {
                dist[i] = c[u][i] + min;
        }
        }

        printf("Shortest Distance is:");
        for (i = 1; i <= n; i++)
        {
        printf("\n %d--------->%d=%d", src, i, dist[i]);
        }
        }
}
```

Output:


Router table entries for router A:
Destination router: A B C D E
Hop count : 0 1 1 2 2
Router table entries for router B:
Destination router: A B C D E
Hop count : 1 0 2 3 3
Router table entries for router C:
Destination router: A B C D E
Hop count : 1 2 0 1 1
Router table entries for router D:
Destination router: A B C D E
Hop count : 2 3 1 0 2
Router table entries for router E:
Destination router: A B C D E
Hop count : 2 3 1 2 0
...Program finished with exit code 0
Press ENTER to exit console.

Experiment No. 9

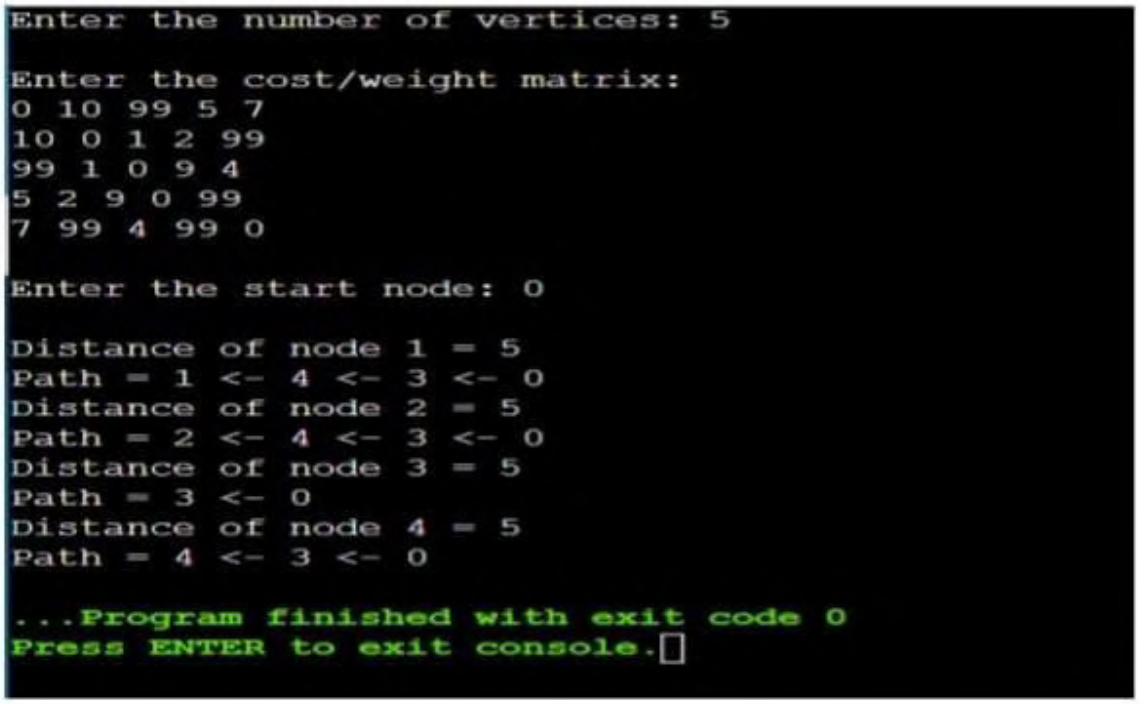
Aim: Write a program for distance vector algorithm to find suitable path for transmission.

Code:

```
// Online C compiler to run C program online
#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");
    return 0;
}
```

Output:


Enter the number of vertices: 5
Enter the cost/weight matrix:
0 10 99 5 7
10 0 1 2 99
99 1 0 9 4
5 2 9 0 99
7 99 4 99 0
Enter the start node: 0
Distance of node 1 = 5
Path = 1 <- 4 <- 3 <- 0
Distance of node 2 = 5
Path = 2 <- 4 <- 3 <- 0
Distance of node 3 = 5
Path = 3 <- 0
Distance of node 4 = 5
Path = 4 <- 3 <- 0
...Program finished with exit code 0
Press ENTER to exit console.

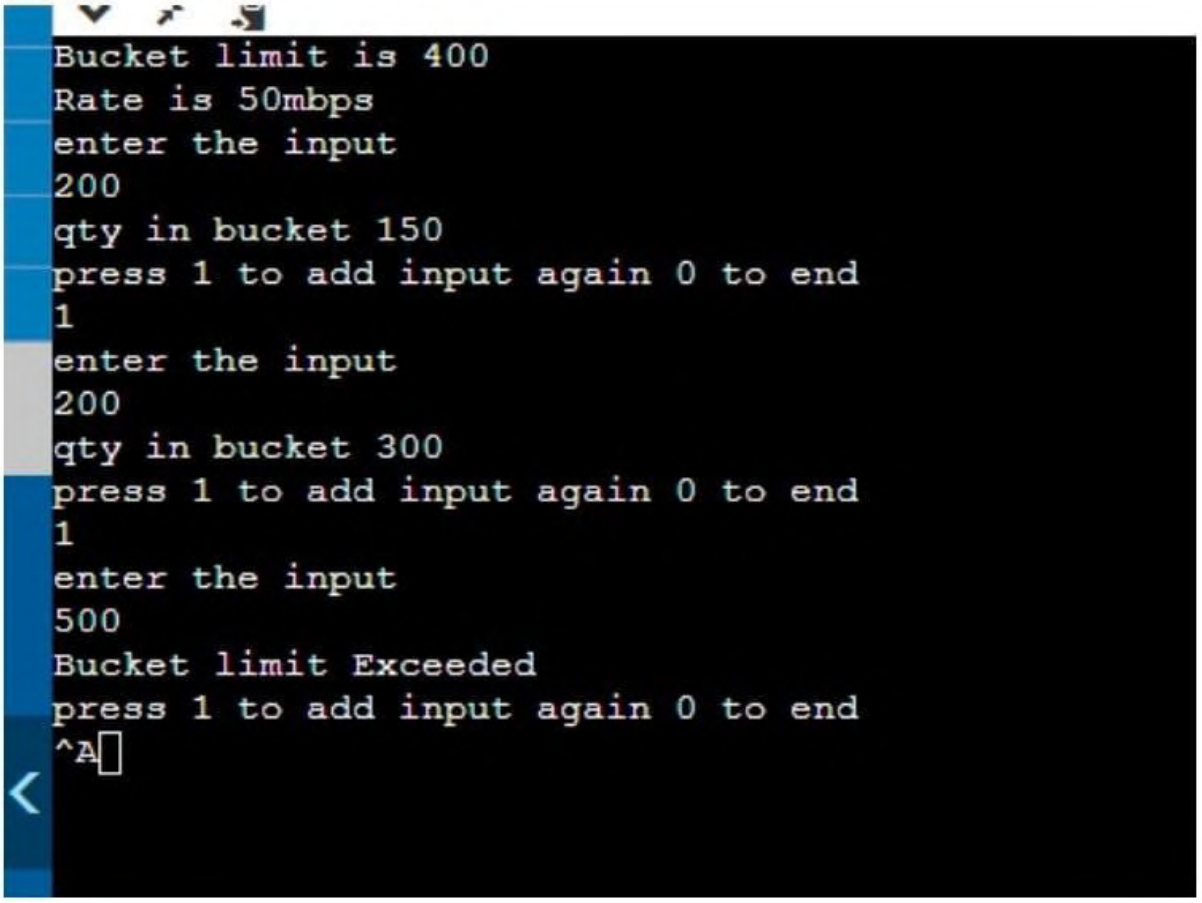
Experiment No. 10

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

```
capacity = int(input("Enter the bucket size"))
outrate = int(input("Enter the outflow rate"))

buffer = 0
n = 0
while n < 5:
        inflow = int(input("Enter the momentary inflow"))
        buffer += inflow
        if buffer >= capacity:
        print("Bucket full")
        break
        buffer -= outrate
        if buffer < 0:
        buffer = 0
        print(buffer)
        n+=1
```

Output:


Bucket limit is 400
Rate is 50mbps
enter the input
200
qty in bucket 150
press 1 to add input again 0 to end
1
enter the input
200
qty in bucket 300
press 1 to add input again 0 to end
1
enter the input
500
Bucket limit Exceeded
press 1 to add input again 0 to end
^A

Experiment No. 11

Aim: 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:

```
from socket import *
serverName = 'DESKTOP-3FLLP14'
serverPort = 12533
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()
from socket import *
```

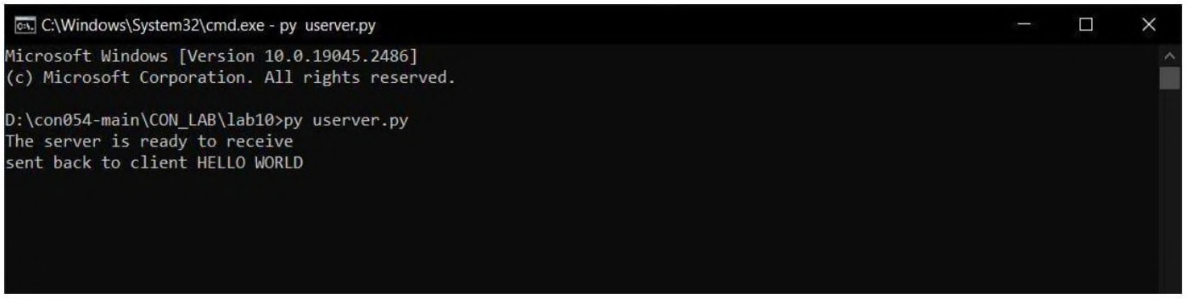
```
serverName='DESKTOP-3FLLP14'
serverPort = 12533
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print ("The server is ready to receive")
while 1:
        connectionSocket, addr = serverSocket.accept()
```

```
        sentence = connectionSocket.recv(1024).decode()

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

        connectionSocket.send(l.encode())
        file.close()
        connectionSocket.close()
```

```
        Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import socket
>>> socket.gethostname()
'DESKTOP-3FLLP14'
>>>
========== RESTART: C:/satwik/lab11/servertcp.py ==========
The server is ready to receive
```

Output:


C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py uclient.py
Enter file name: try.txt
From Server: b'HELLO WORLD\n\n'
D:\con054-main\CON_LAB\lab10>


C:\Windows\System32\cmd.exe - py userver.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py userver.py
The server is ready to receive
sent back to client HELLO WORLD

Experiment No. 12

Aim: 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:

```
from socket import *
serverName = VivoBook4854-PC
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name")

clientSocket.sento(bytes(sentence, "utf-8"), (serverName, serverPort))
filecontents, serverAddress = clientSocket.recvfrom(2048)
print('From server:' , filecontents)
clientSocket.close()
```

```
from socket import *
serverPort = 12001
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()
```

Output: