VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

VIBHA VENKATESH SHANBHAG (1BM20CS184)

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 “LAB COURSE **COMPUTER NETWORKS**” carried out by **VIBHA VENKATESH SHANBHAG (1BM20CS184),** 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. Nandhini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

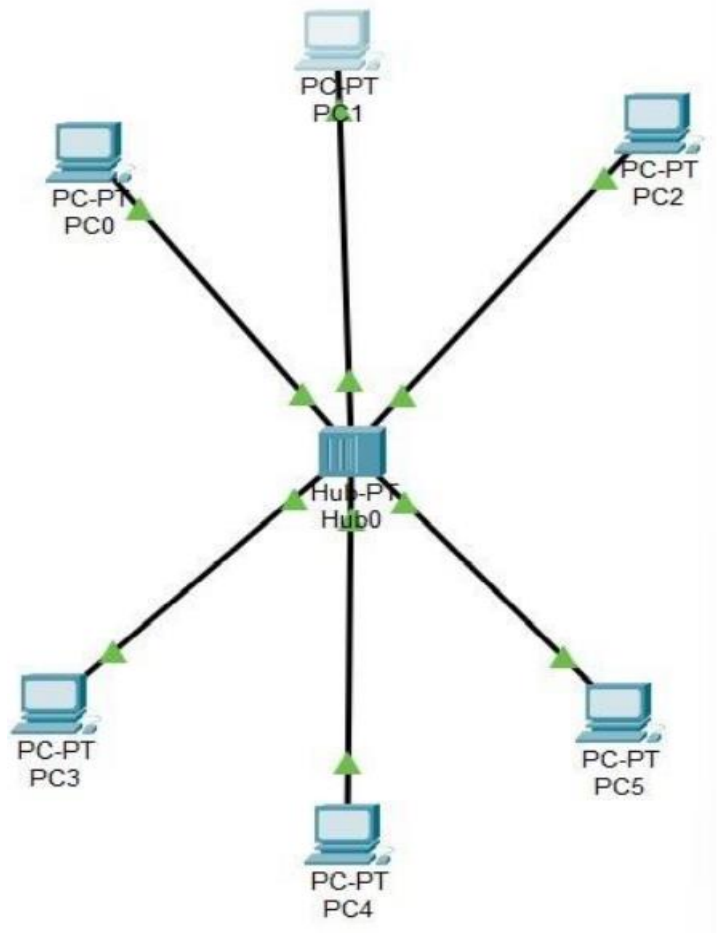
Index

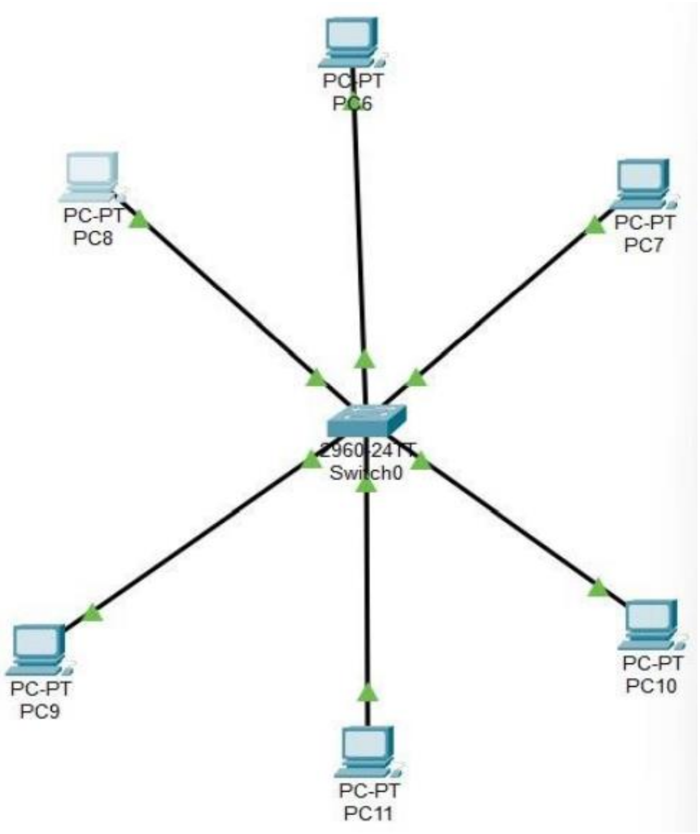
Cycle-1

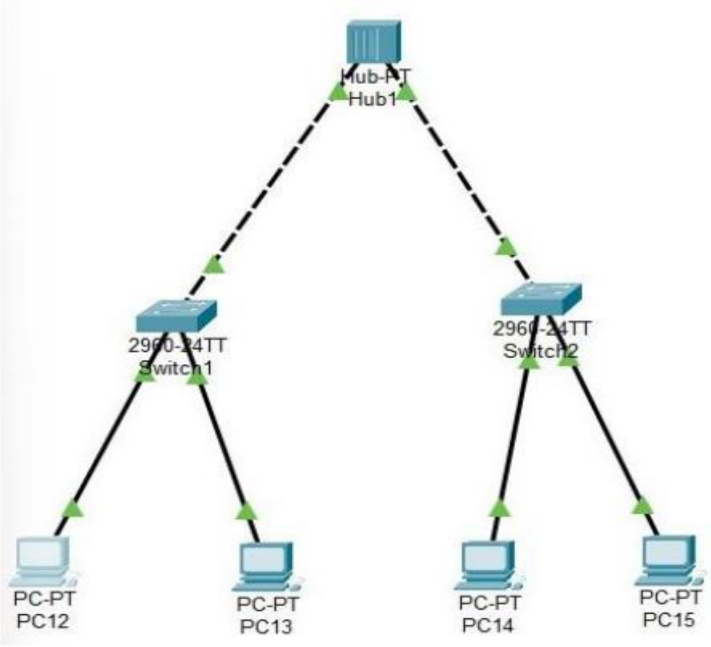
Experiment No - 1

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

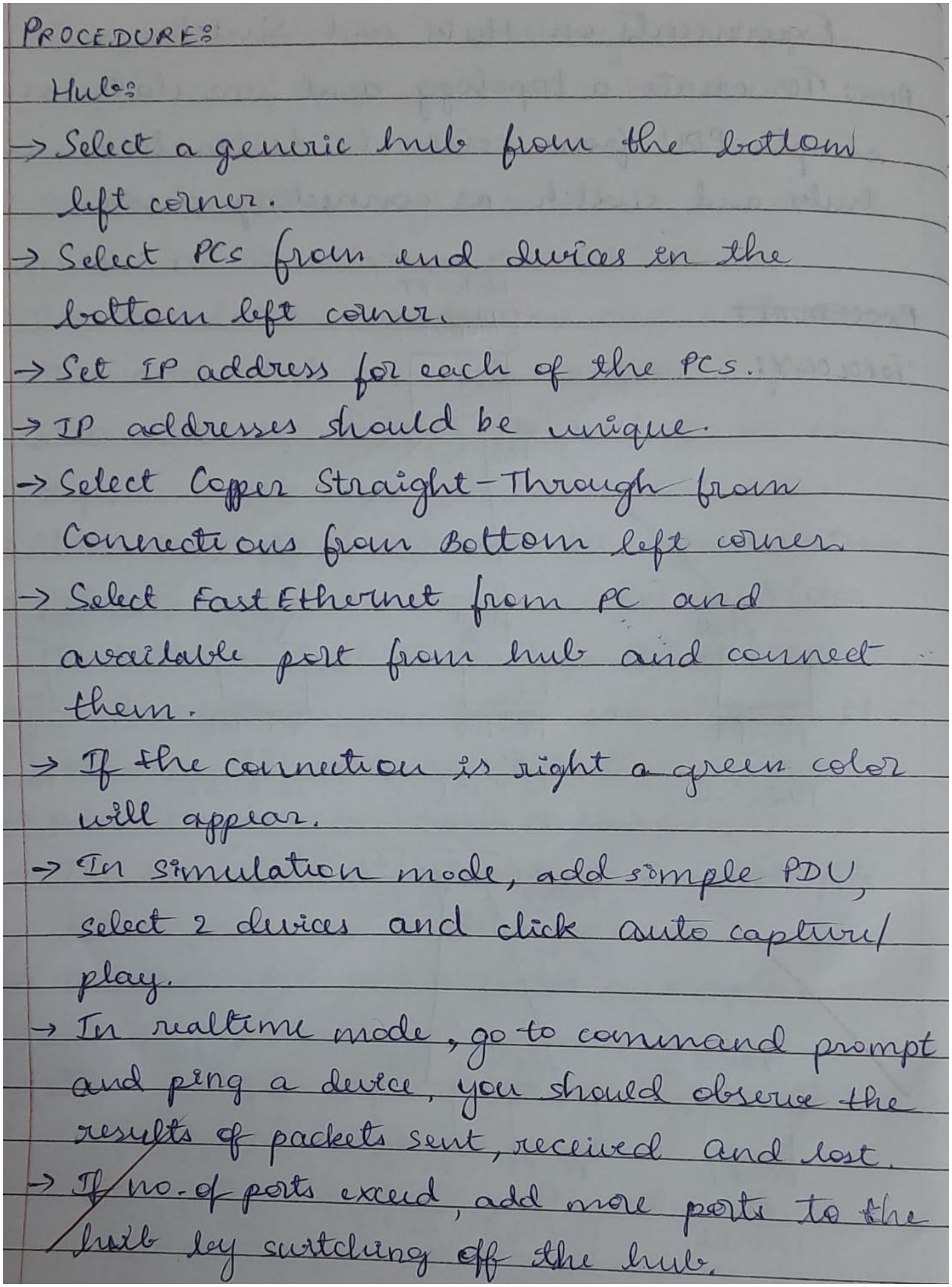
Topology:




PC-PT
PC6
PC-PT
PC8
PC-PT
PC7
2960-24TT
Switch0
PC-PT
PC9
PC-PT
PC10
PC-PT
PC11


Hub-PT
Hub1
2960-24TT
Switch1
2960-24TT
Switch2
PC-PT
PC12
PC-PT
PC13
PC-PT
PC14
PC-PT
PC15

Procedure:



PROCEDURE:

Hub:

-> Select a generic hub from the bottom left corner.

-> Select PCs from end devices in the bottom left corner.

-> Set IP address for each of the PCs.

-> IP addresses should be unique.

-> Select Copper Straight-Through from Connections from Bottom left corner.

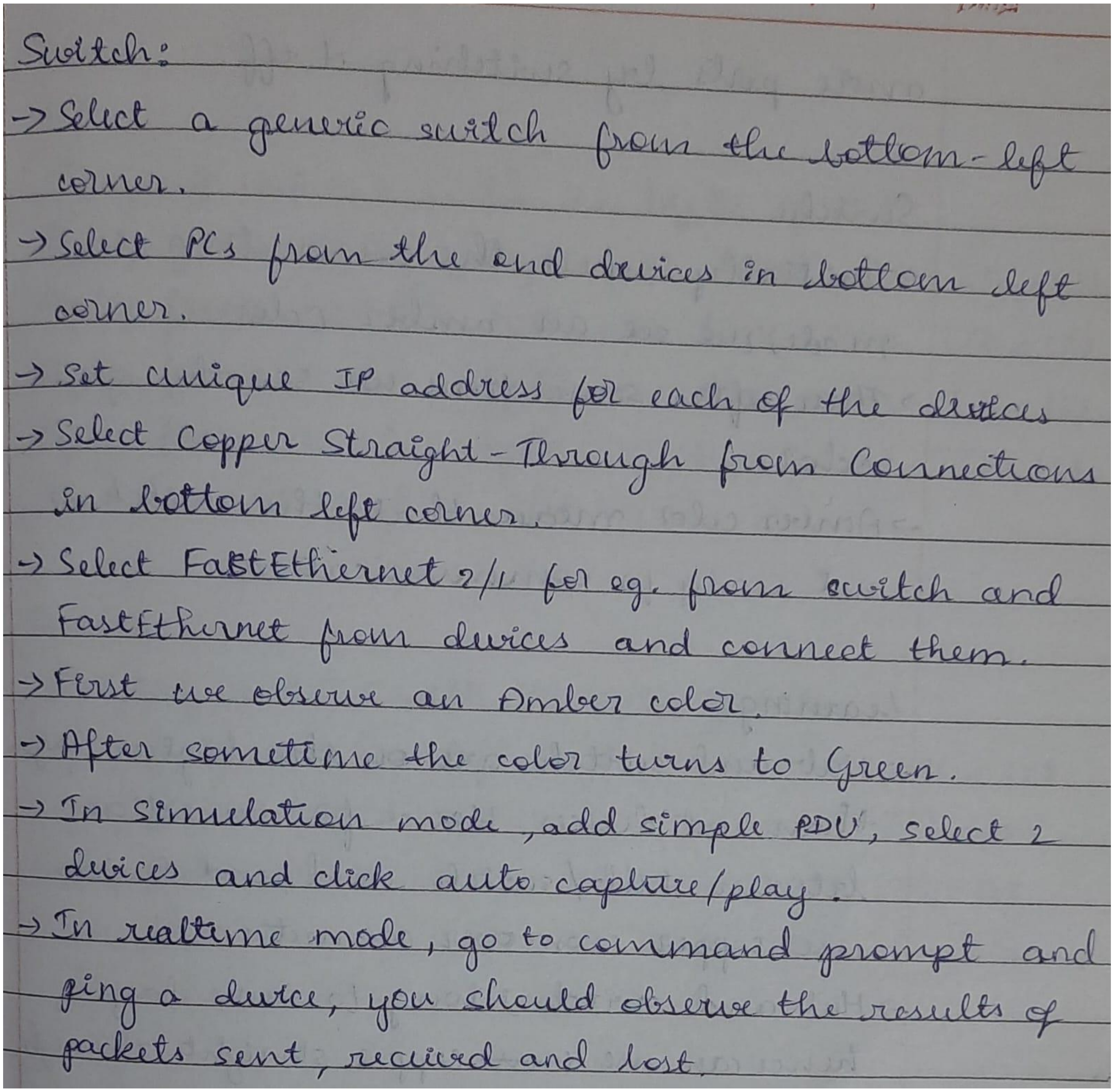
-> Select FastEthernet from PC and available port from hub and connect them.

-> If the connection is right a green color will appear.

-> In simulation mode, add simple PDU, select 2 devices and click auto capture/play.

-> In realtime mode, go to command prompt and ping a device, you should observe the results of packets sent, received and lost.

-> If no. of ports exceed, add more ports to the hub by switching off the hub.



Switch:

-> Select a generic switch from the bottom-left corner.

-> Select PCs from the end devices in bottom left corner.

-> Set unique IP address for each of the devices

-> Select Copper Straight-Through from Connections in bottom left corner.

-> Select FastEthernet 2/1 for eg. from switch and FastEthernet from devices and connect them.

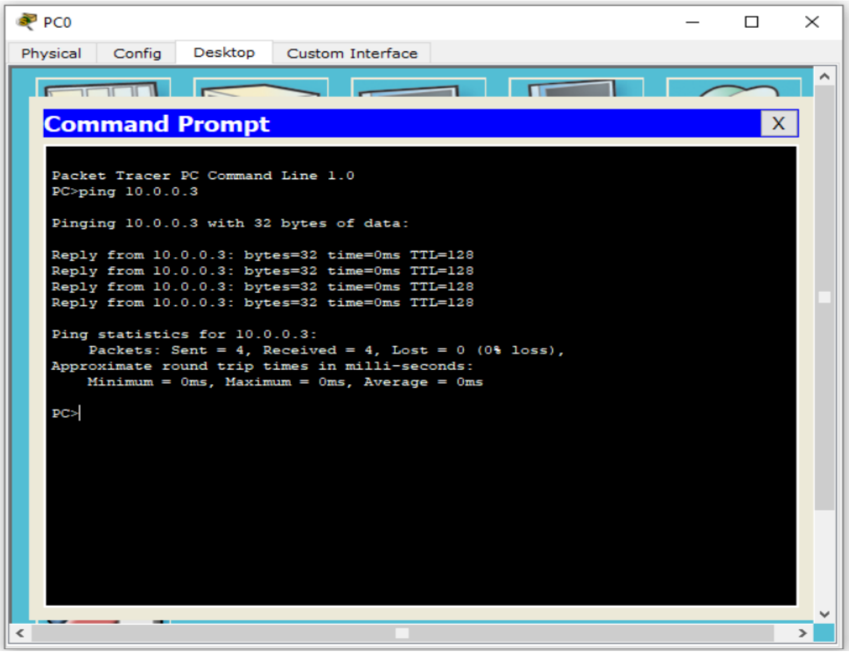
-> First we observe an Amber color.

-> After sometime the color turns to Green.

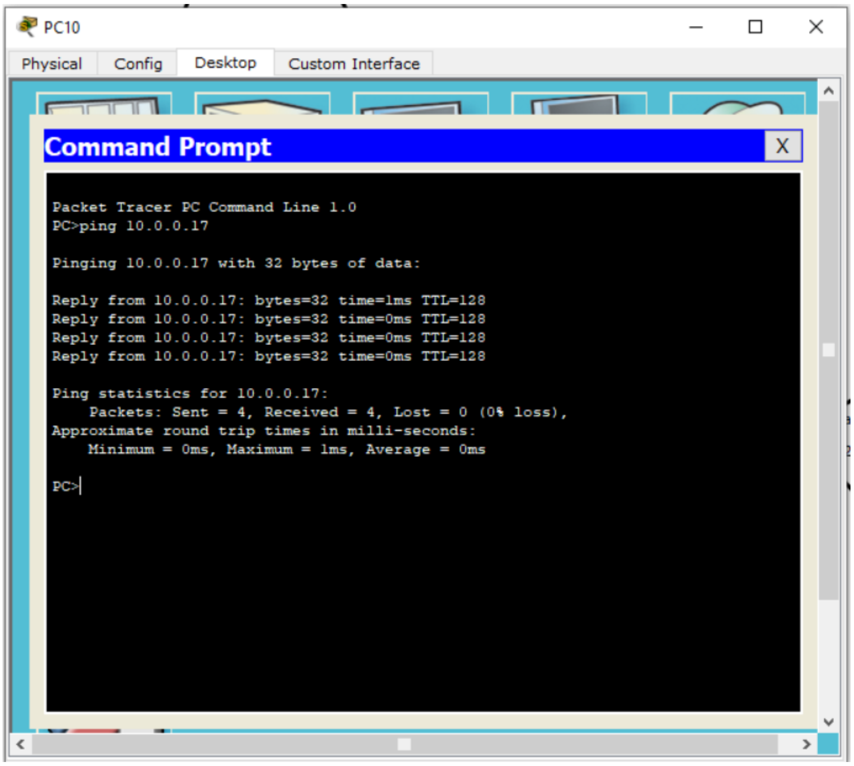
-> In simulation mode, add simple PDU, select 2 devices and click auto capture/play.

-> In realtime mode, go to command prompt and ping a device, you should observe the results of packets sent, received and lost.

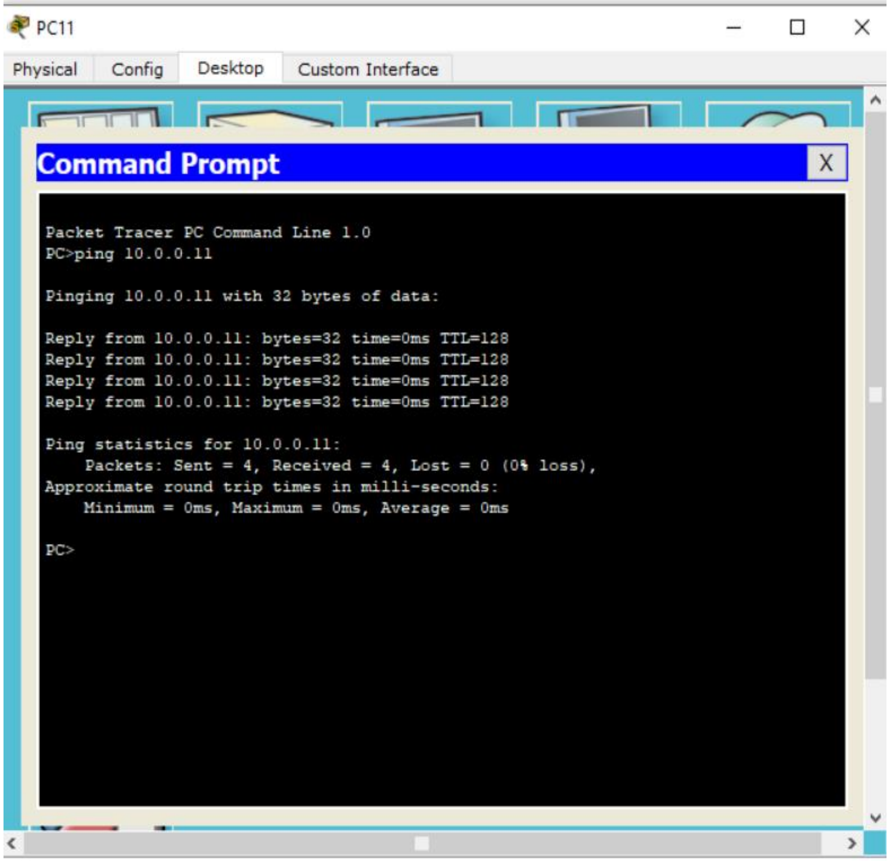
Output:



```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>
```



```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.17

Pinging 10.0.0.17 with 32 bytes of data:

Reply from 10.0.0.17: bytes=32 time=1ms TTL=128
Reply from 10.0.0.17: bytes=32 time=0ms TTL=128
Reply from 10.0.0.17: bytes=32 time=0ms TTL=128
Reply from 10.0.0.17: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.17:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>
```

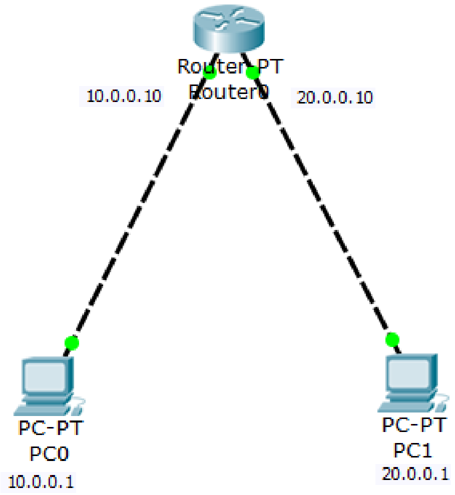


```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.11

Pinging 10.0.0.11 with 32 bytes of data:

Reply from 10.0.0.11: bytes=32 time=0ms TTL=128
Reply from 10.0.0.11: bytes=32 time=0ms TTL=128
Reply from 10.0.0.11: bytes=32 time=0ms TTL=128
Reply from 10.0.0.11: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>
```

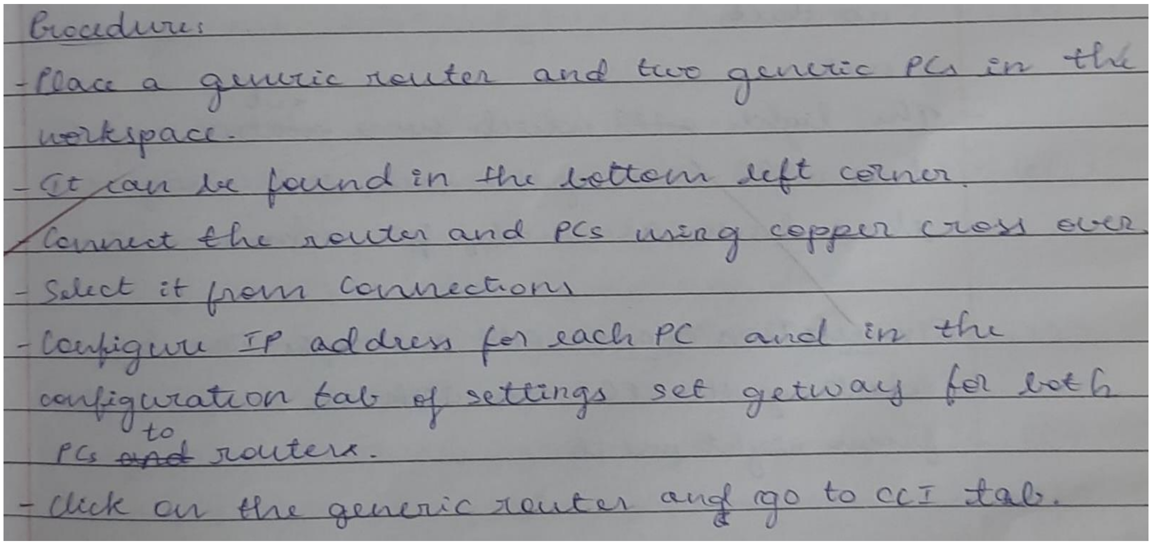
Experiment No – 2

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

Topology:

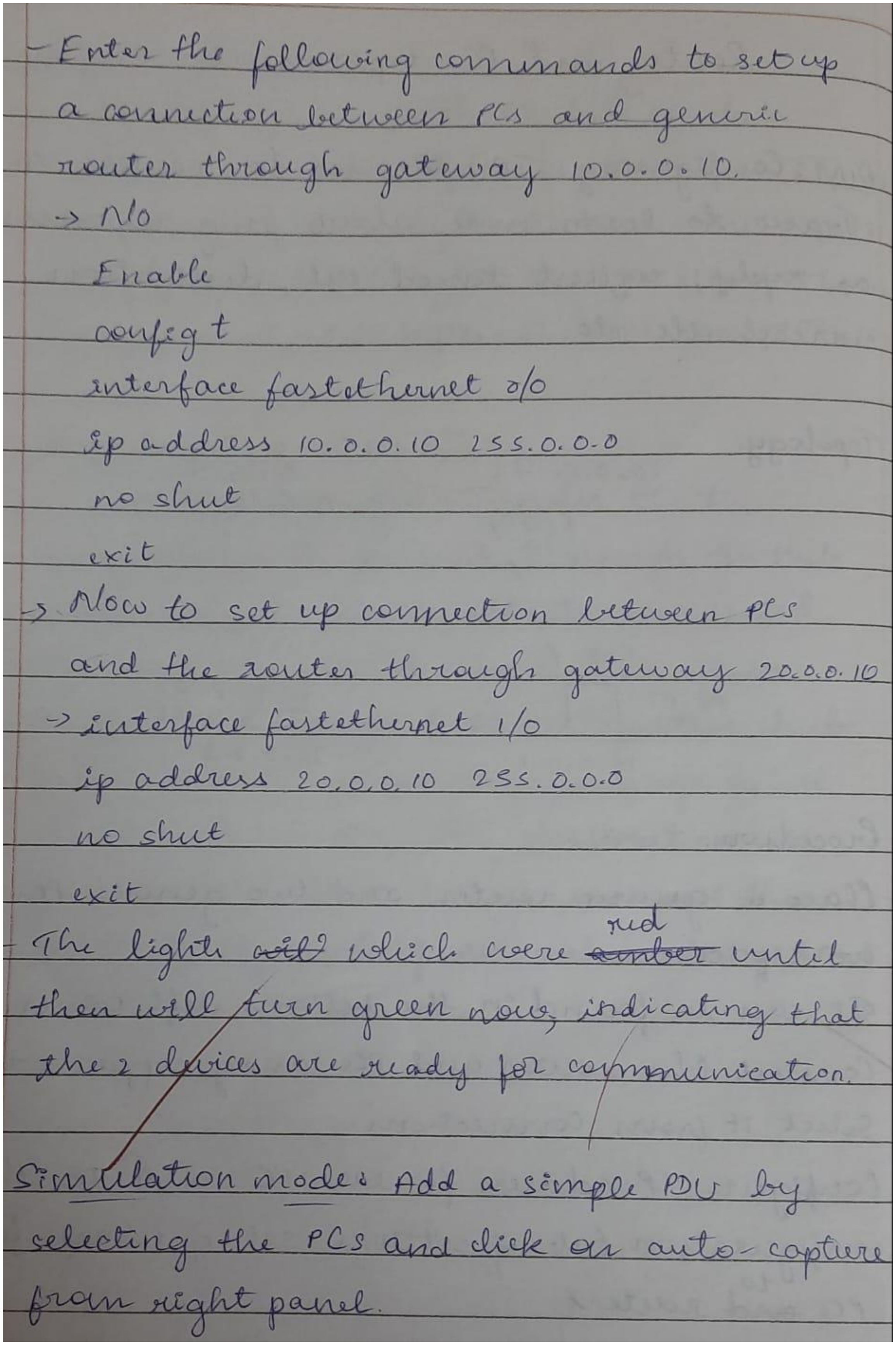


Procedure:



Procedure:

- Place a generic router and two generic PCs in the workspace.
- It can be found in the bottom left corner.
- Connect the router and PCs using copper cross over.
- Select it from connections
- Configure IP address for each PC and in the configuration tab of settings set getway for both PCs ~~and~~ to routers.
- Click on the generic router and go to CCI tab.



- Enter the following commands to set up a connection between PCs and generic router through gateway 10.0.0.10.

```
-> N/o
   Enable
   config t
   interface fastethernet 0/0
   ip address 10.0.0.10 255.0.0.0
   no shut
   exit
```

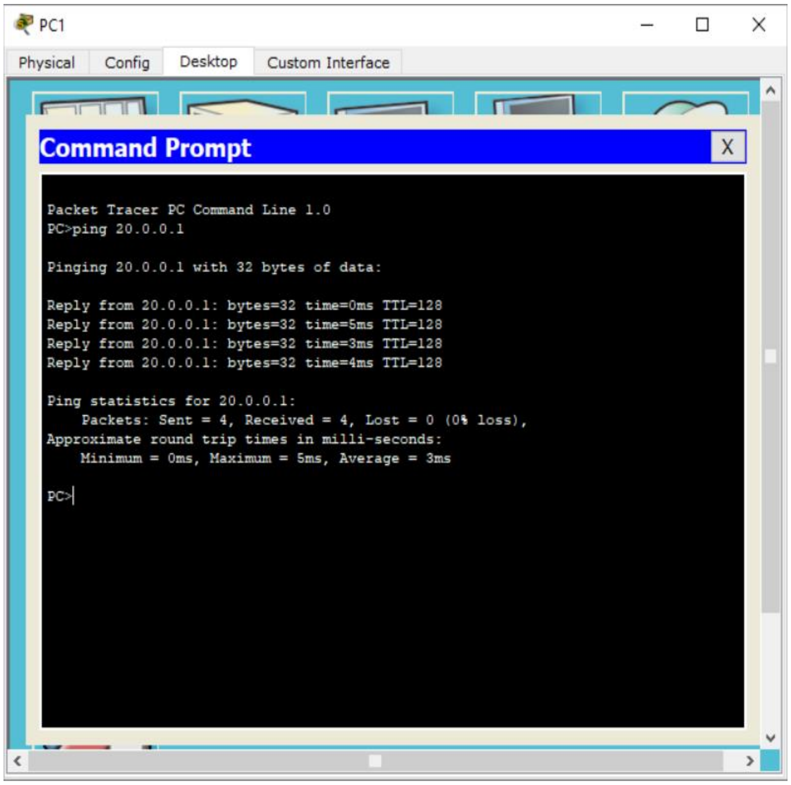
-> Now to set up connection between PCs and the router through gateway 20.0.0.10

```
-> interface fastethernet 1/0
   ip address 20.0.0.10 255.0.0.0
   no shut
   exit
```

- The light ~~will~~ which were ~~amber~~ red until then will turn green now, indicating that the 2 devices are ready for communication.

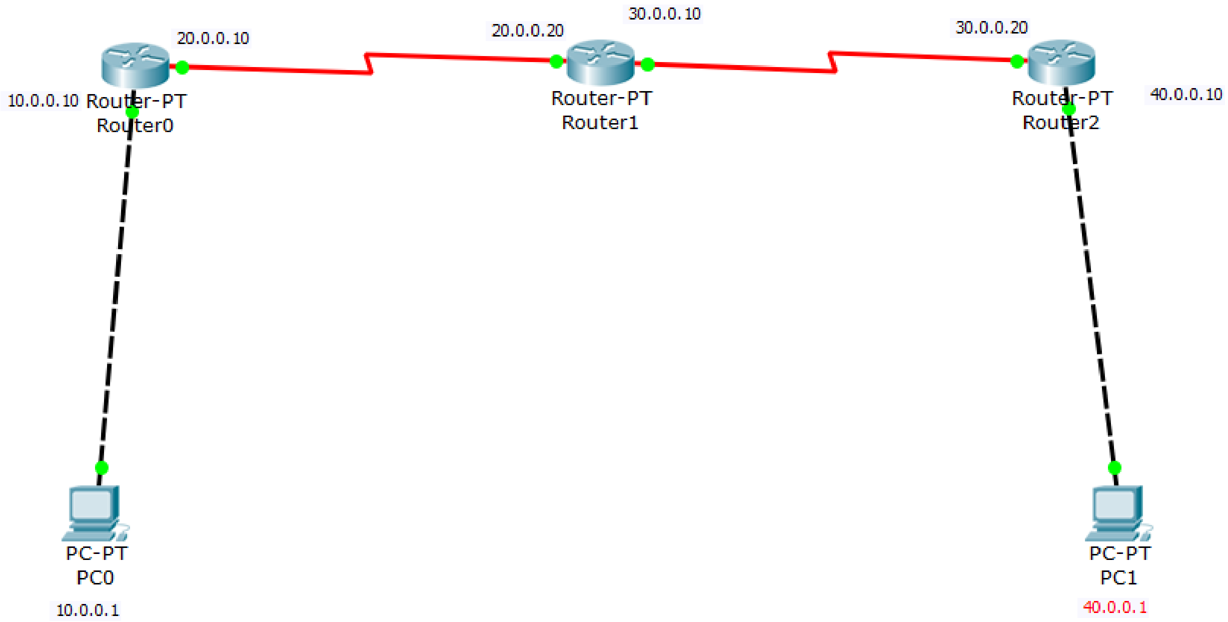
Simulation mode: Add a simple PDU by selecting the PCs and click on auto-capture from right panel.

Output:

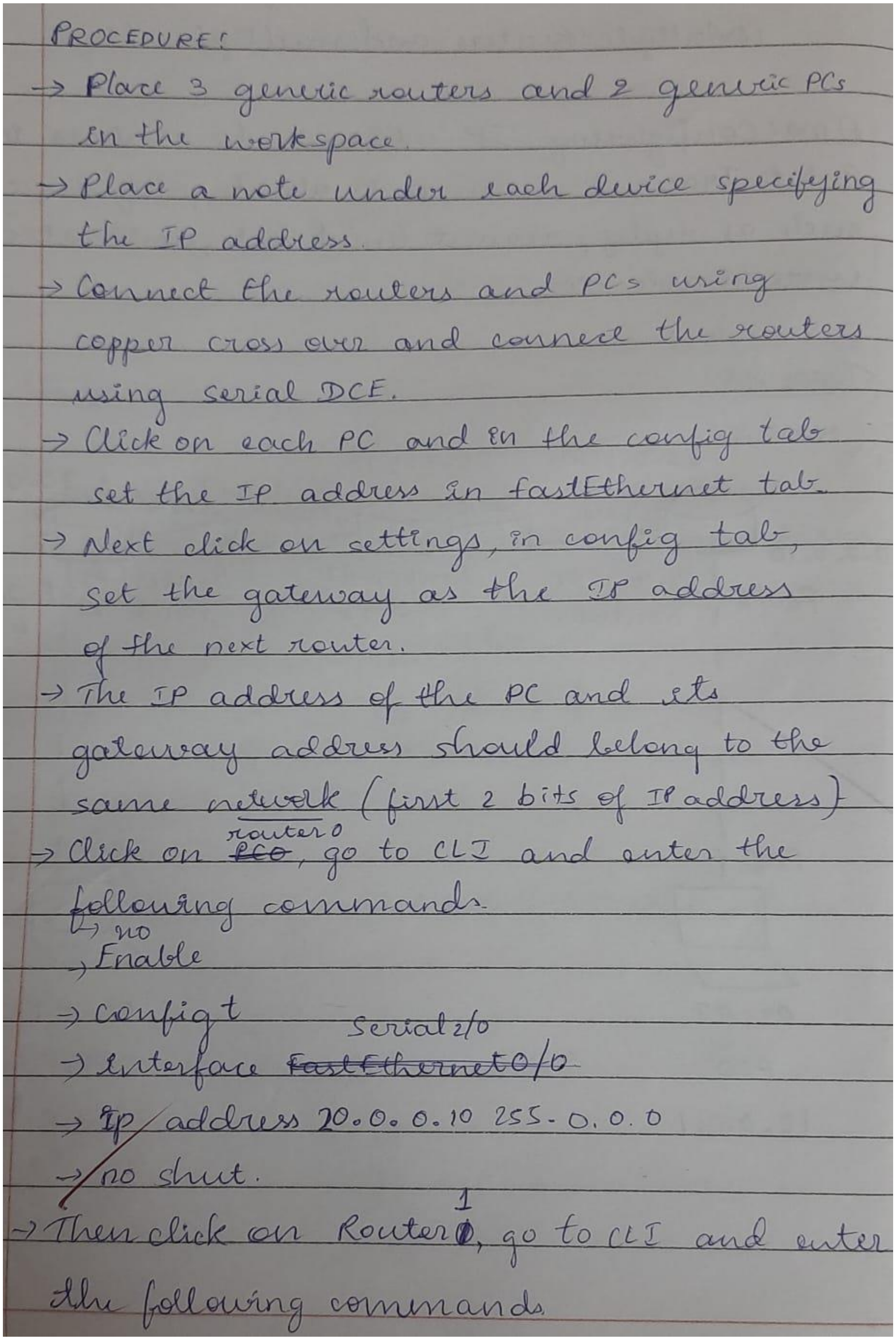


```
Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=128
Reply from 20.0.0.1: bytes=32 time=5ms TTL=128
Reply from 20.0.0.1: bytes=32 time=3ms TTL=128
Reply from 20.0.0.1: bytes=32 time=4ms TTL=128

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 5ms, Average = 3ms

PC>
```

Topology:



Procedure:



PROCEDURE:

→ Place 3 generic routers and 2 generic PCs in the workspace.

→ Place a note under each device specifying the IP address.

→ Connect the routers and PCs using copper cross over and connect the routers using serial DCE.

→ Click on each PC and in the config tab set the IP address in fastEthernet tab.

→ Next click on settings, in config tab, set the gateway as the IP address of the next router.

→ The IP address of the PC and its gateway address should belong to the same network (first 2 bits of IP address)

→ Click on ~~PC0~~ router0, go to CLI and enter the following commands.

↳ no

→ Enable

→ config t

→ interface ~~FastEthernet~~0/0 Serial 2/0

→ ip address 20.0.0.10 255.0.0.0

→ no shut.

→ Then click on Router~~0~~1, go to CLI and enter the following commands.

→ no

→ Enable

→ config t

→ interface ~~FastEthernet0/0~~ Serial 2/0

→ ip address 20.0.0.20 255.0.0.0

→ no shut

→ After doing this the lights which were red until now, will turn green between these 2 routers indicating that they are ready for communication.

To connect 1 PC and 1 Router:

→ Since IP address of the PC is already configured go to router0, open to CLI, enter the following commands

→ Enable

→ config t

→ Interface FastEthernet0/0

→ ip address 10.0.0.10 255.0.0.0

→ no shut.

→ The red light turns green between the PC and the router.

Teaching router0 of network 30

→ Enable

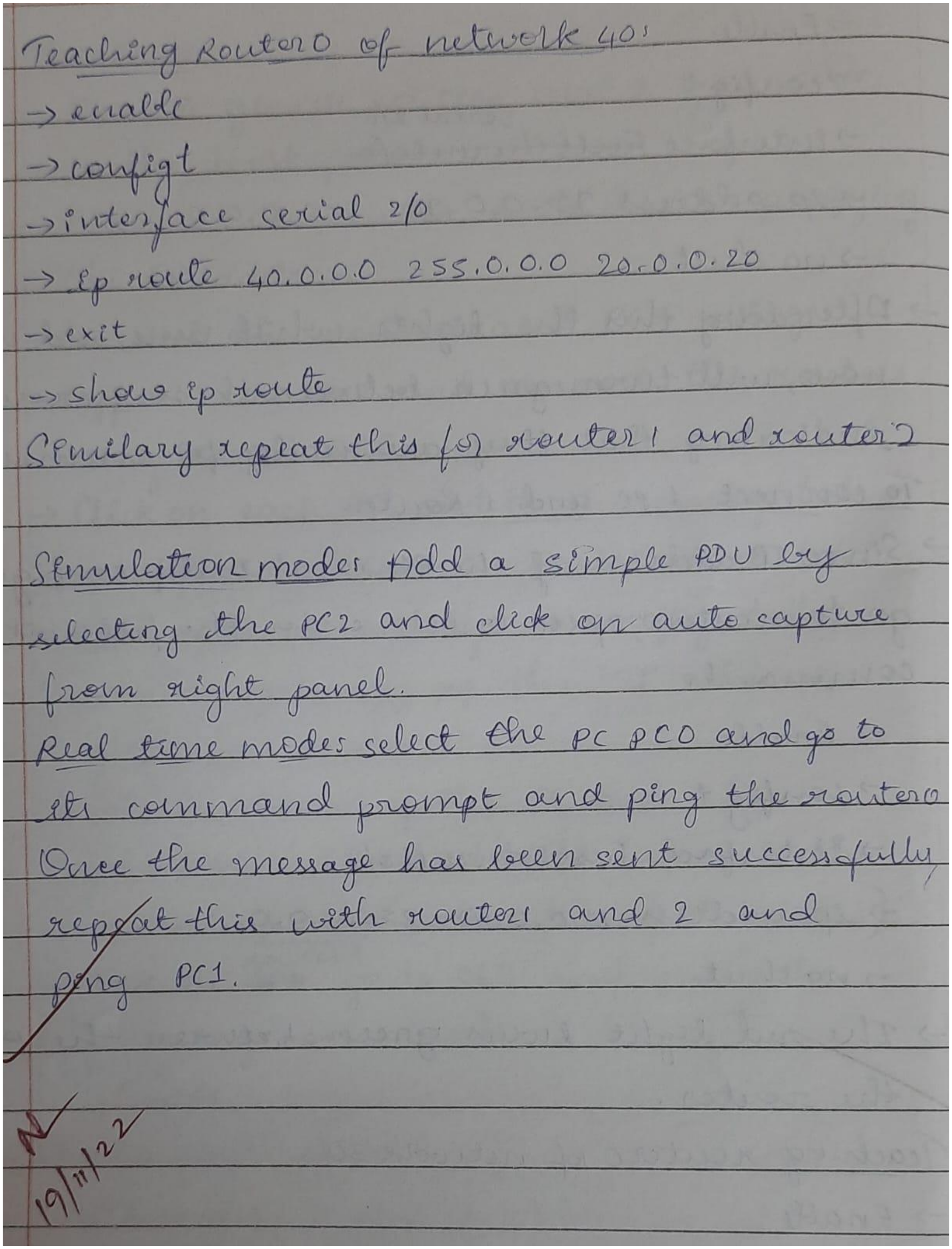
→ configt

→ interface serial 2/0

→ ip route 30.0.0.0 255.0.00 20.0.0.20

→ exit

→ show ip route



Teaching Router0 of network 40:

-> enable

-> config t

-> interface serial 2/0

-> ip route 40.0.0.0 255.0.0.0 20.0.0.20

-> exit

-> show ip route

Similary repeat this for router1 and router2

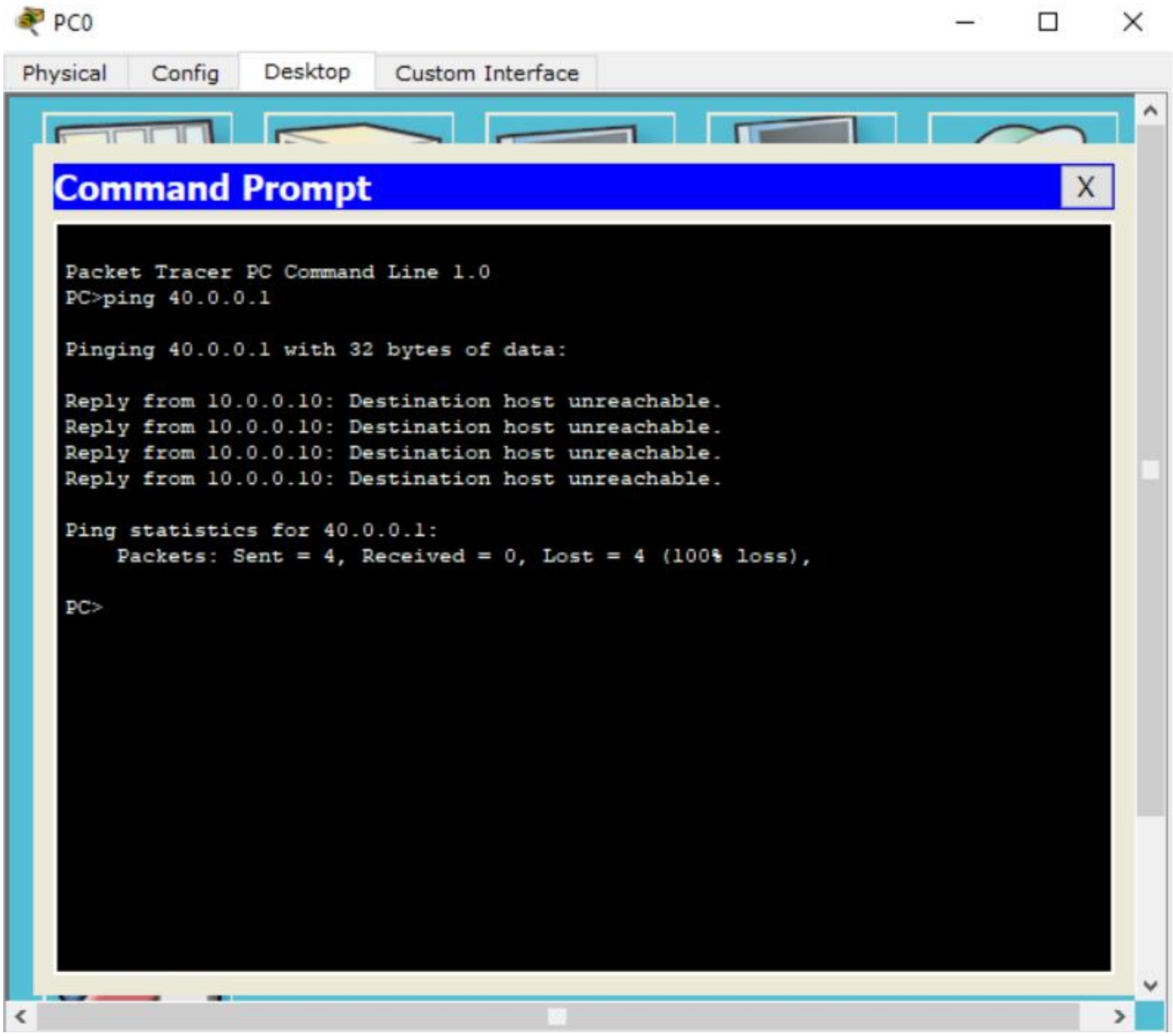
Simulation mode: Add a simple PDU by selecting the PC2 and click on auto capture from right panel.

Real time mode: select the PC PC0 and go to its command prompt and ping the router0

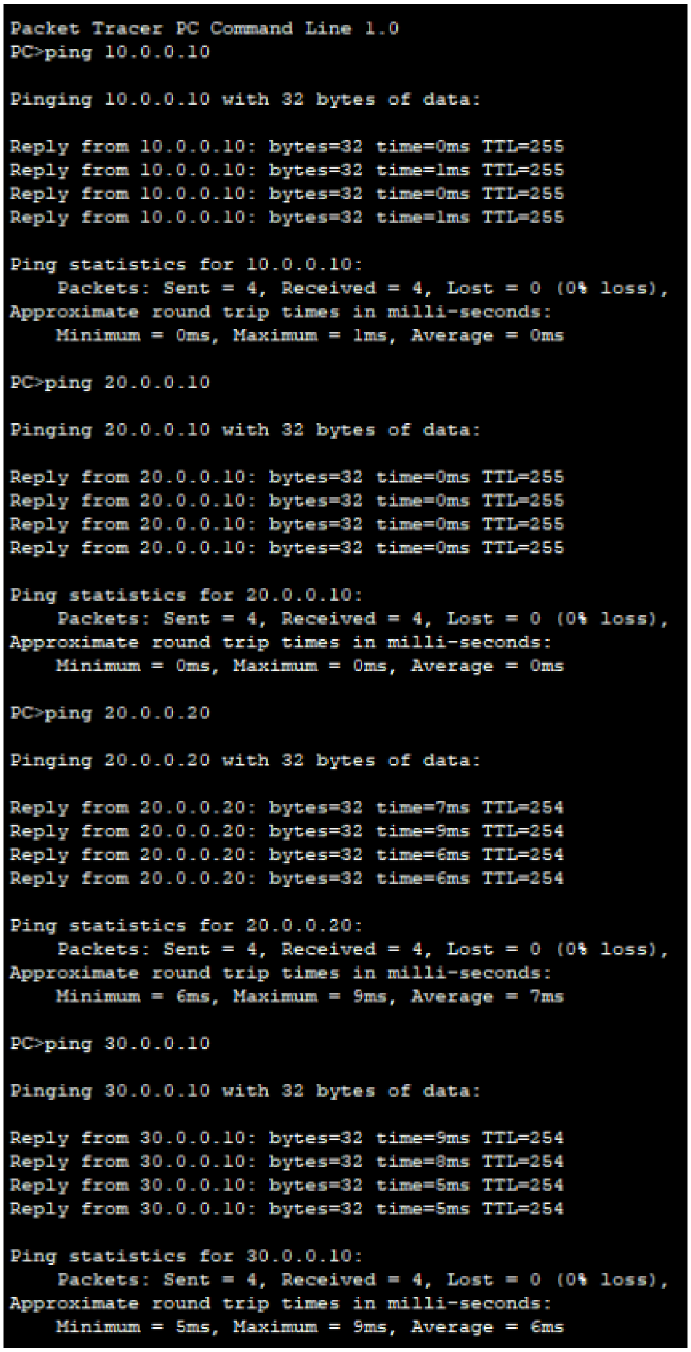
Once the message has been sent successfully repeat this with router1 and 2 and ping PC1.

19/11/22

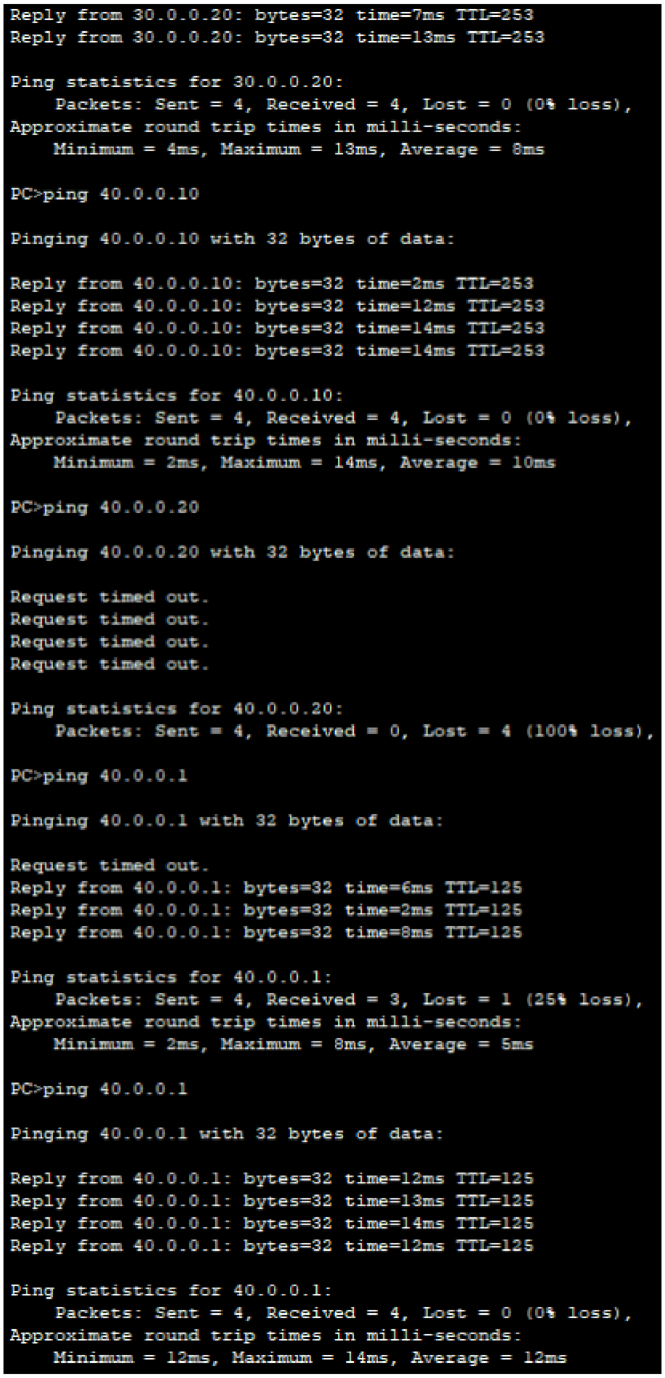
Output:



```
Packet Tracer PC Command Line 1.0
PC>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),

PC>
```



```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.10

Pinging 10.0.0.10 with 32 bytes of data:

Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Reply from 10.0.0.10: bytes=32 time=1ms TTL=255
Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Reply from 10.0.0.10: bytes=32 time=1ms TTL=255

Ping statistics for 10.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>ping 20.0.0.10

Pinging 20.0.0.10 with 32 bytes of data:

Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255

Ping statistics for 20.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>ping 20.0.0.20

Pinging 20.0.0.20 with 32 bytes of data:

Reply from 20.0.0.20: bytes=32 time=7ms TTL=254
Reply from 20.0.0.20: bytes=32 time=9ms TTL=254
Reply from 20.0.0.20: bytes=32 time=6ms TTL=254
Reply from 20.0.0.20: bytes=32 time=6ms TTL=254

Ping statistics for 20.0.0.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 9ms, Average = 7ms

PC>ping 30.0.0.10

Pinging 30.0.0.10 with 32 bytes of data:

Reply from 30.0.0.10: bytes=32 time=9ms TTL=254
Reply from 30.0.0.10: bytes=32 time=8ms TTL=254
Reply from 30.0.0.10: bytes=32 time=5ms TTL=254
Reply from 30.0.0.10: bytes=32 time=5ms TTL=254

Ping statistics for 30.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 9ms, Average = 6ms
```

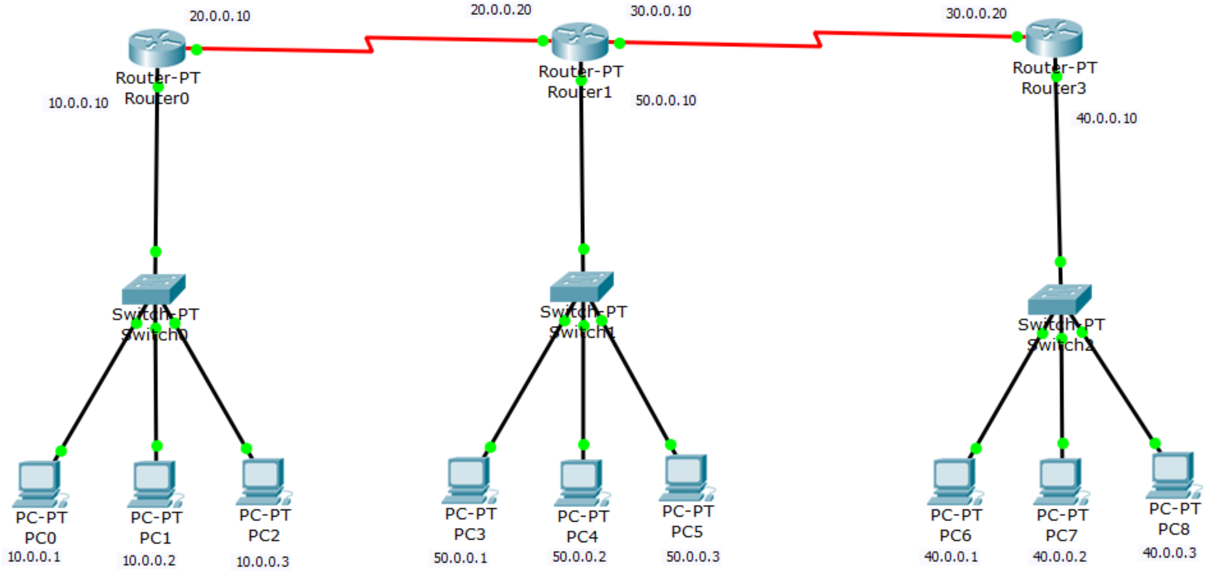


```
Reply from 30.0.0.20: bytes=32 time=7ms TTL=253
Reply from 30.0.0.20: bytes=32 time=13ms TTL=253

Ping statistics for 30.0.0.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 13ms, Average = 8ms

PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=2ms TTL=253
Reply from 40.0.0.10: bytes=32 time=12ms TTL=253
Reply from 40.0.0.10: bytes=32 time=14ms TTL=253
Reply from 40.0.0.10: bytes=32 time=14ms TTL=253

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 14ms, Average = 10ms

PC>ping 40.0.0.20

Pinging 40.0.0.20 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 40.0.0.20:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

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=6ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=8ms 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 = 8ms, Average = 5ms

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=12ms TTL=125
Reply from 40.0.0.1: bytes=32 time=13ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms TTL=125
Reply from 40.0.0.1: bytes=32 time=12ms 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 = 12ms, Maximum = 14ms, Average = 12ms
```

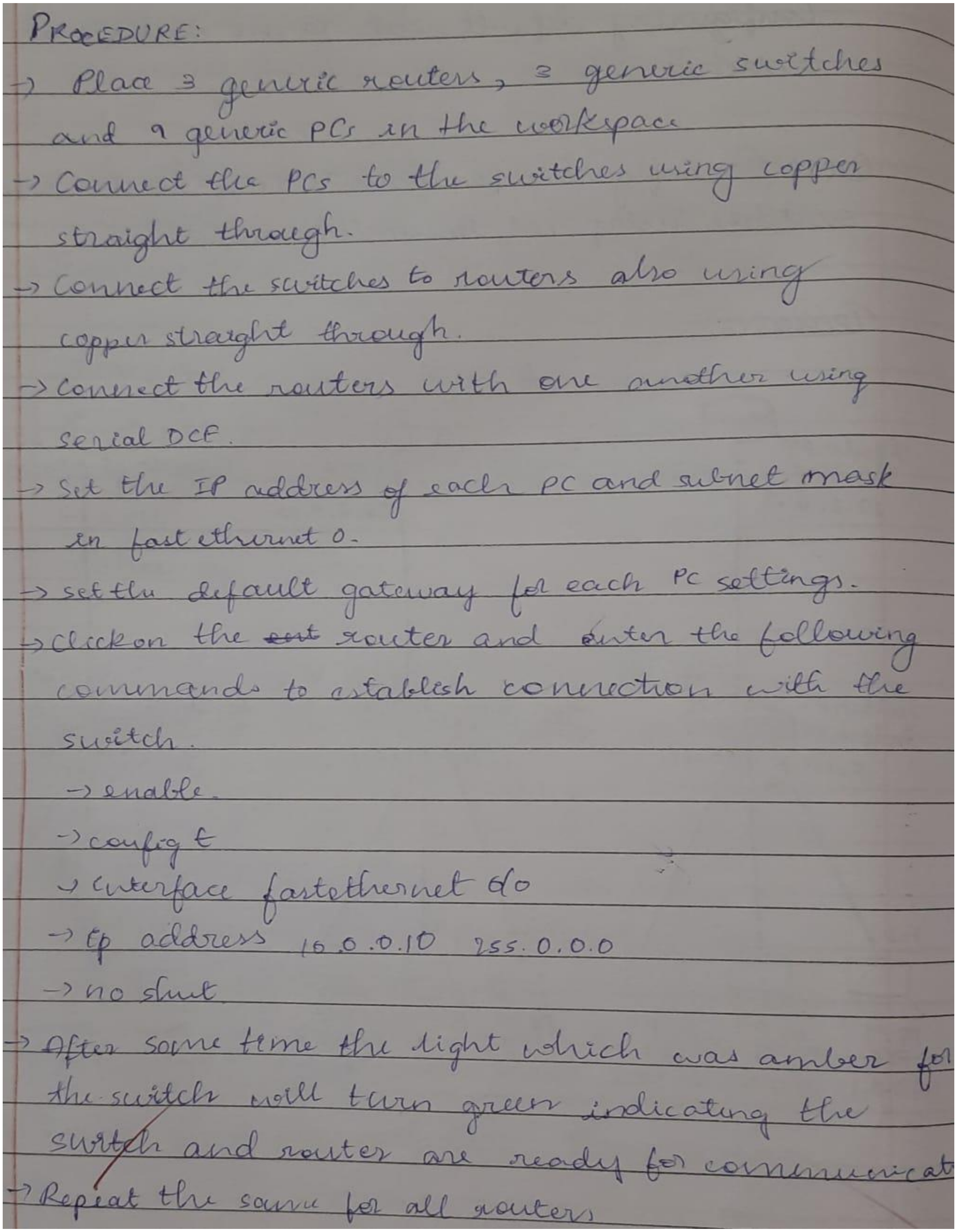
Experiment No – 3

Aim: Configuring default route to the Router.

Topology:



Procedure:



PROCEDURE:

-> Place 3 generic routers, 3 generic switches and 9 generic PCs in the workspace

-> Connect the PCs to the switches using copper straight through.

-> Connect the switches to routers also using copper straight through.

-> Connect the routers with one another using serial DCE.

-> Set the IP address of each PC and subnet mask in fast ethernet 0.

-> set the default gateway for each PC settings.

-> Click on the ~~ent~~ router and enter the following commands to establish connection with the switch.

-> enable.

-> config t

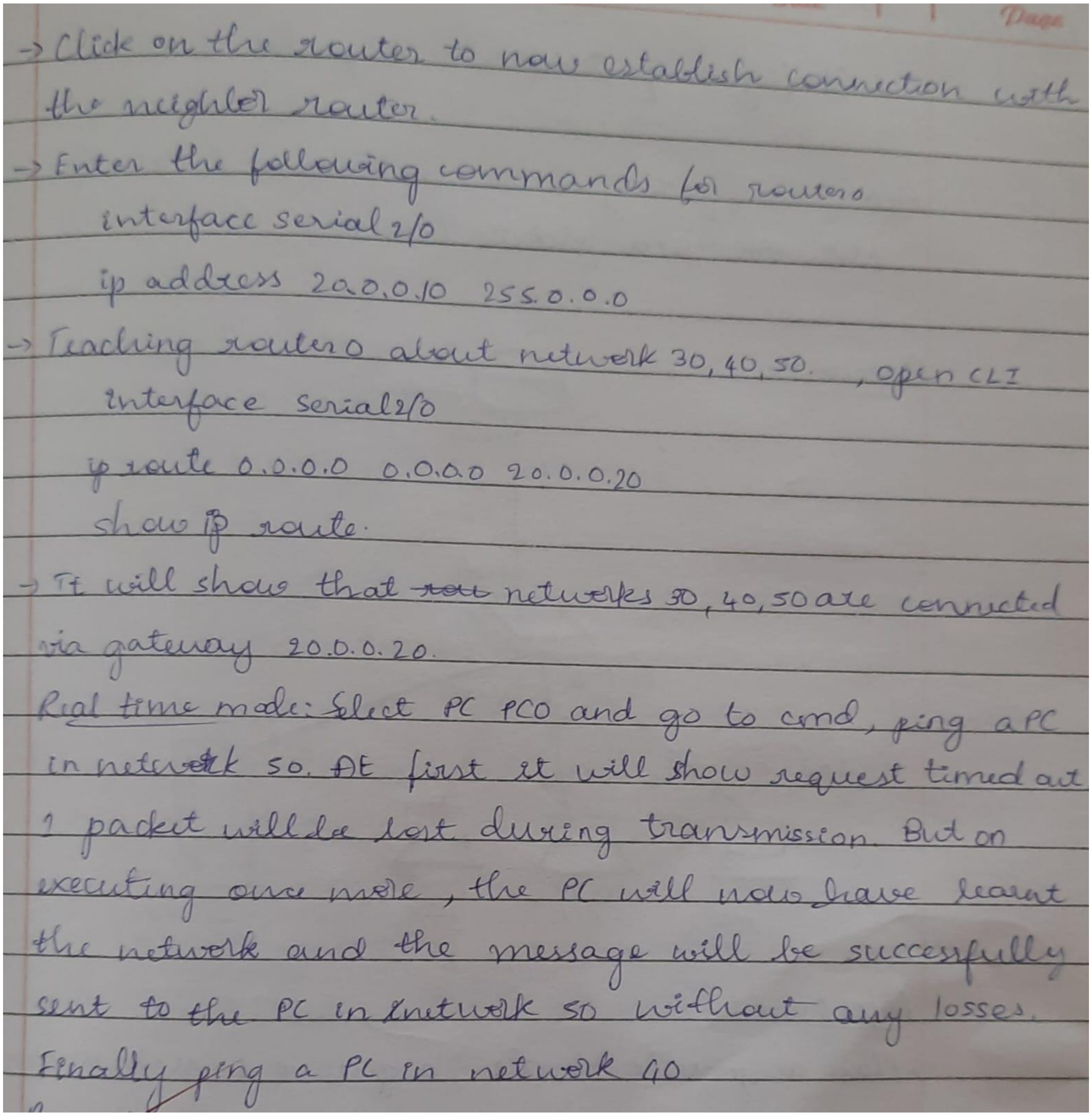
-> interface fastethernet 0/0

-> ip address 10.0.0.10 255.0.0.0

-> no shut

-> After some time the light which was amber for the switch will turn green indicating the switch and router are ready for communicat

-> Repeat the same for all routers



-> Click on the router to now establish connection with the neighbor router.

-> Enter the following commands for router0

```
interface serial 2/0
ip address 20.0.0.10 255.0.0.0
```

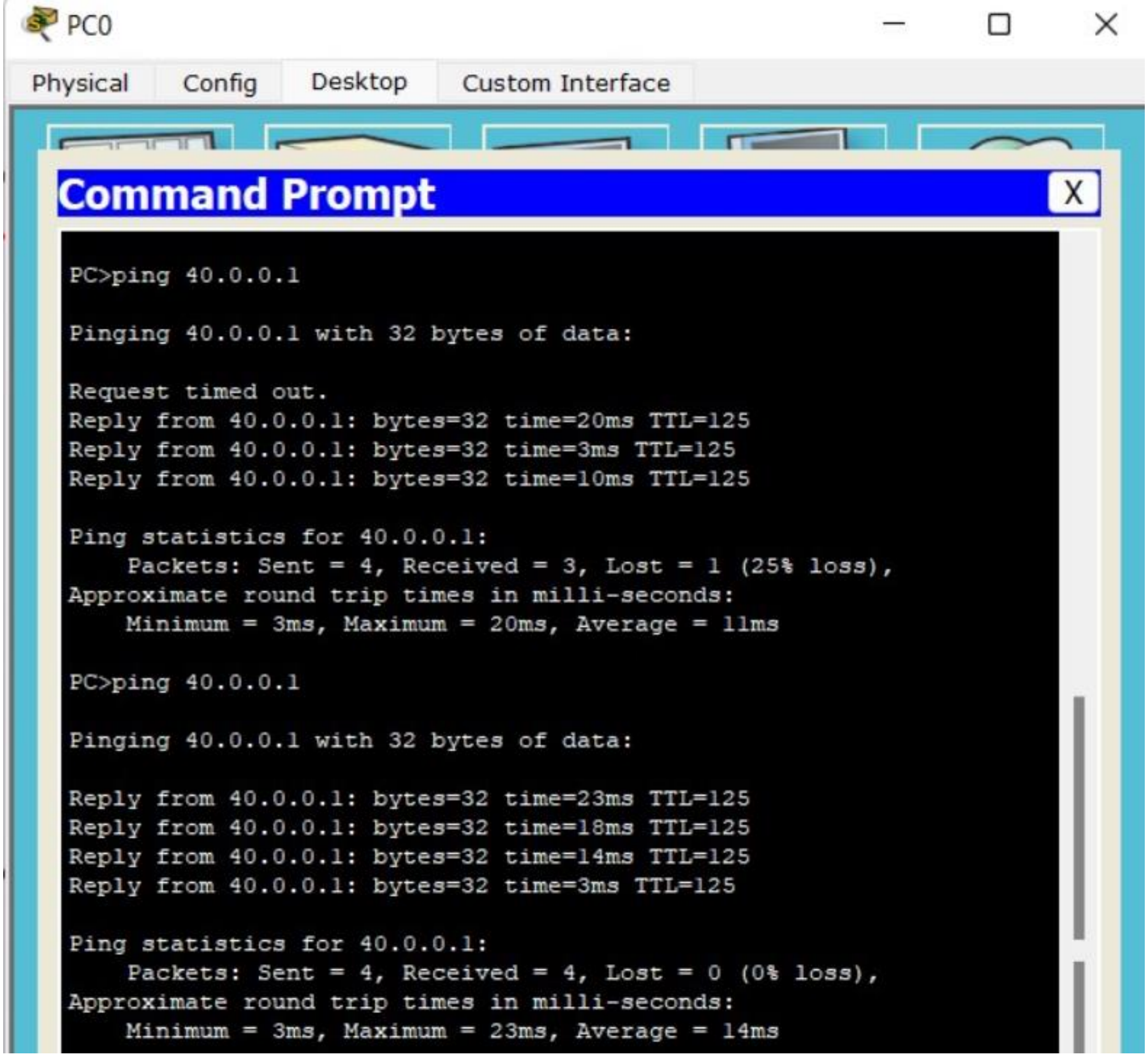
-> Teaching router0 about network 30, 40, 50. , open CLI

```
interface serial2/0
ip route 0.0.0.0 0.0.0.0 20.0.0.20
show ip route.
```

-> It will show that ~~net~~ networks 30, 40, 50 are connected via gateway 20.0.0.20.

Real time mode: Select PC PC0 and go to cmd, ping a PC in network 50. At first it will show request timed out 1 packet will be lost during transmission. But on executing once more, the PC will now have learnt the network and the message will be successfully sent to the PC in Enetwork 50 without any losses. Finally ping a PC in network 40.

Output:

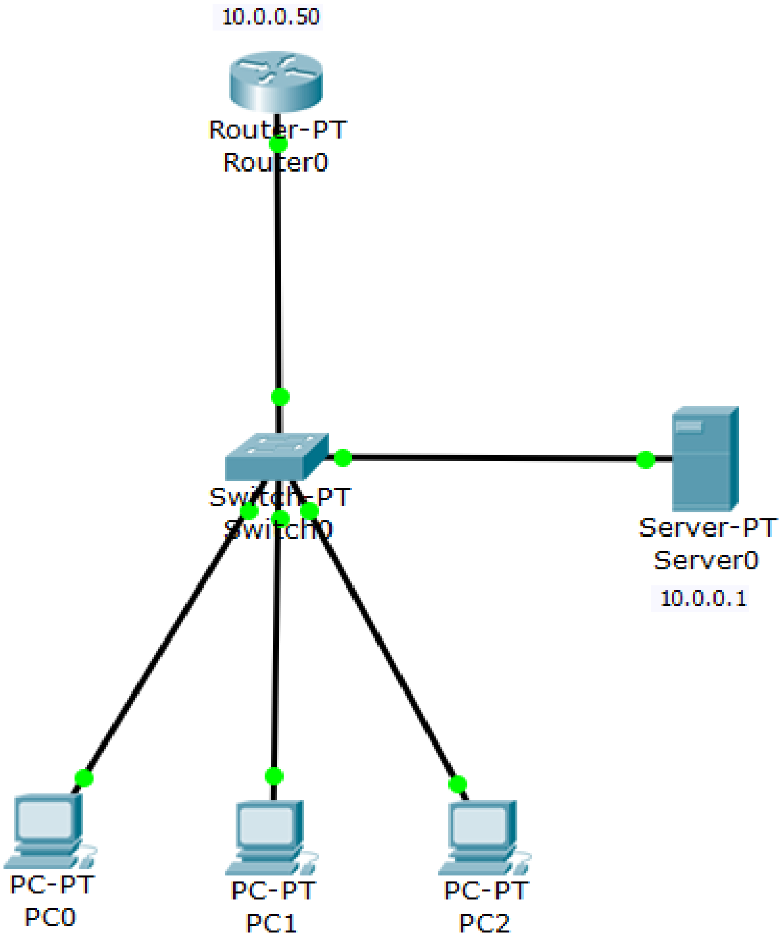


```
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.1: bytes=32 time=20ms TTL=125
Reply from 40.0.0.1: bytes=32 time=3ms 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 = 3ms, Maximum = 20ms, Average = 11ms

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=23ms TTL=125
Reply from 40.0.0.1: bytes=32 time=18ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms 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 = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 23ms, Average = 14ms
```

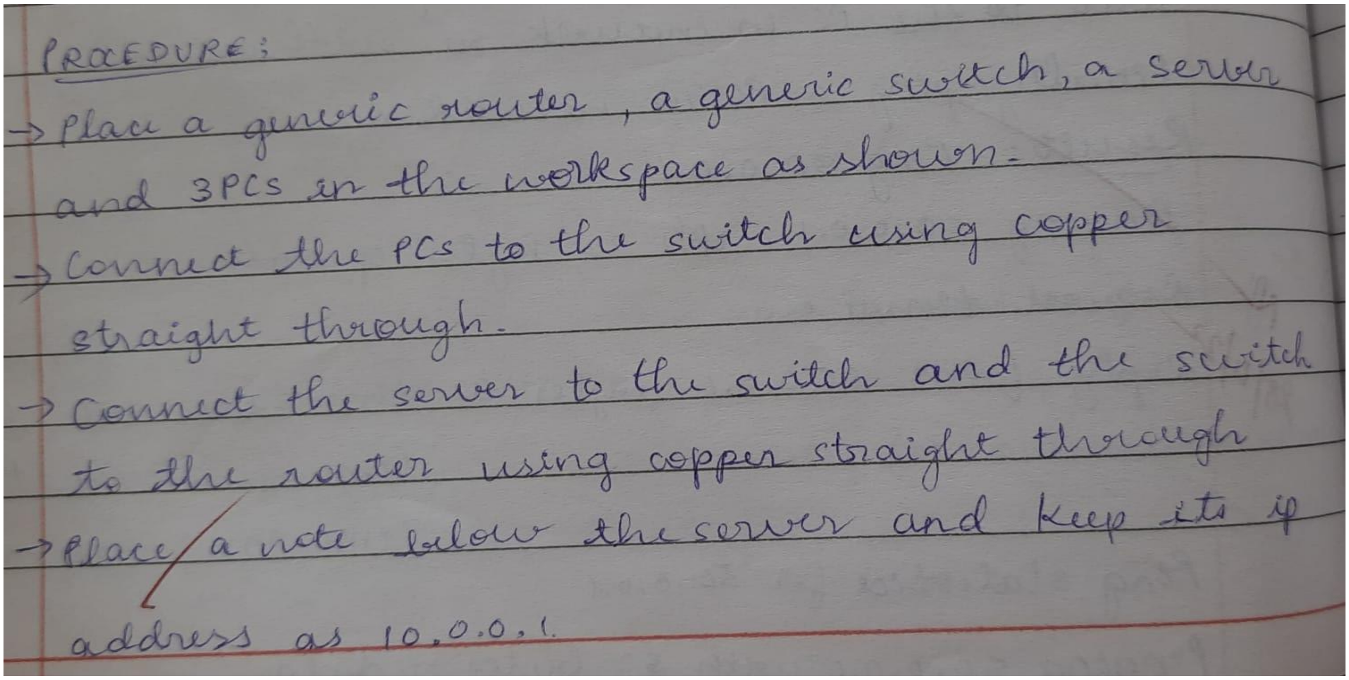
Experiment No - 4

Aim: Configuring DHCP within a LAN in a packet Tracer.

Topology:



Procedure:



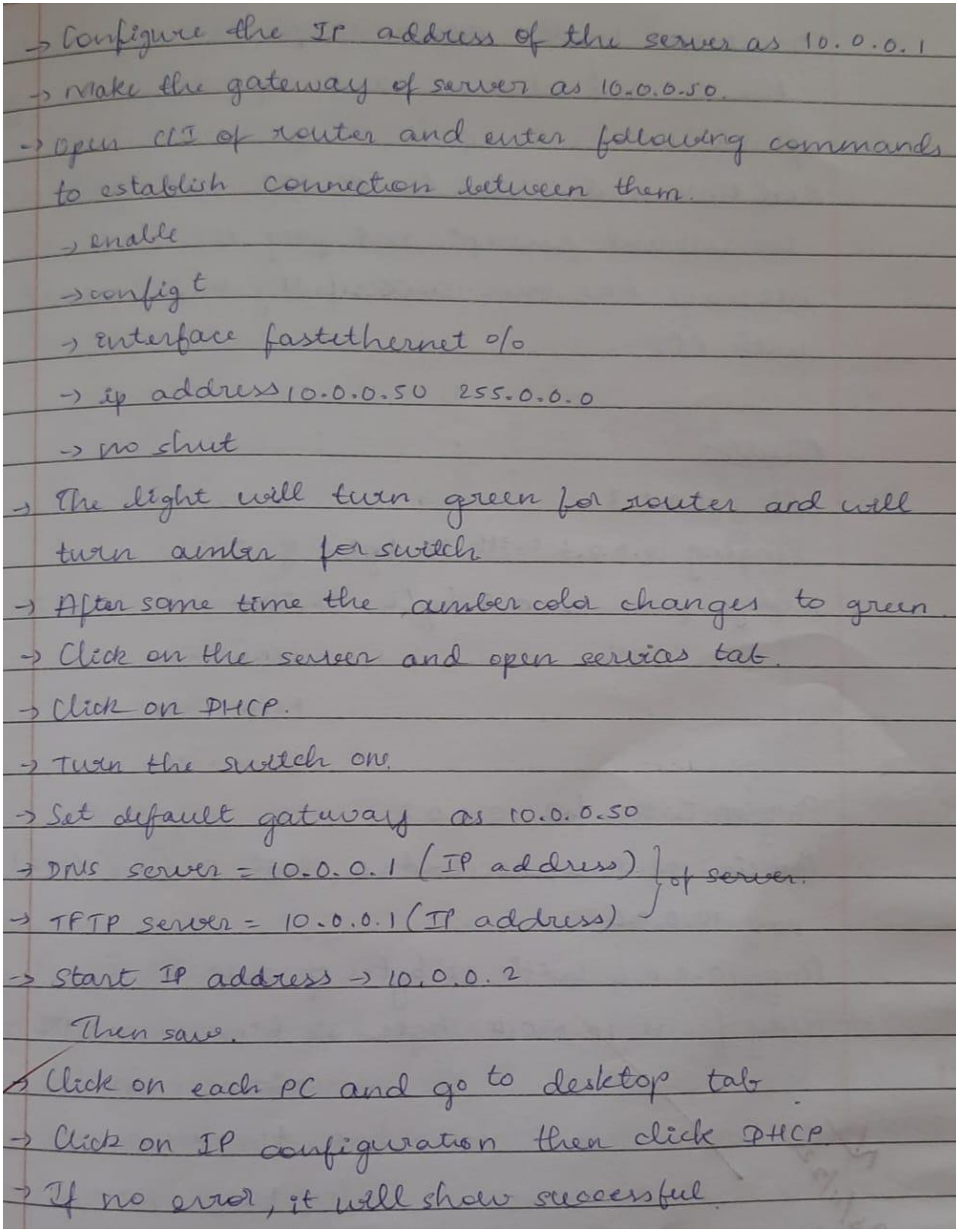
PROCEDURE:

→ Place a generic router, a generic switch, a server and 3PCs in the workspace as shown.

→ Connect the PCs to the switch using copper straight through.

→ Connect the server to the switch and the switch to the router using copper straight through

→ Place a note below the server and keep its ip address as 10.0.0.1.



-> Configure the IP address of the server as 10.0.0.1

-> Make the gateway of server as 10.0.0.50.

-> Open CLI of router and enter following commands to establish connection between them.

-> enable

-> config t

-> interface fastethernet 0/0

-> ip address 10.0.0.50 255.0.0.0

-> no shut

-> The light will turn green for router and will turn amber for switch

-> After some time the amber color changes to green.

-> Click on the server and open services tab.

-> Click on DHCP.

-> Turn the switch on.

-> Set default gatway as 10.0.0.50

-> DNS server = 10.0.0.1 (IP address) } of server.

-> TFTP server = 10.0.0.1 (IP address)

-> Start IP address -> 10.0.0.2

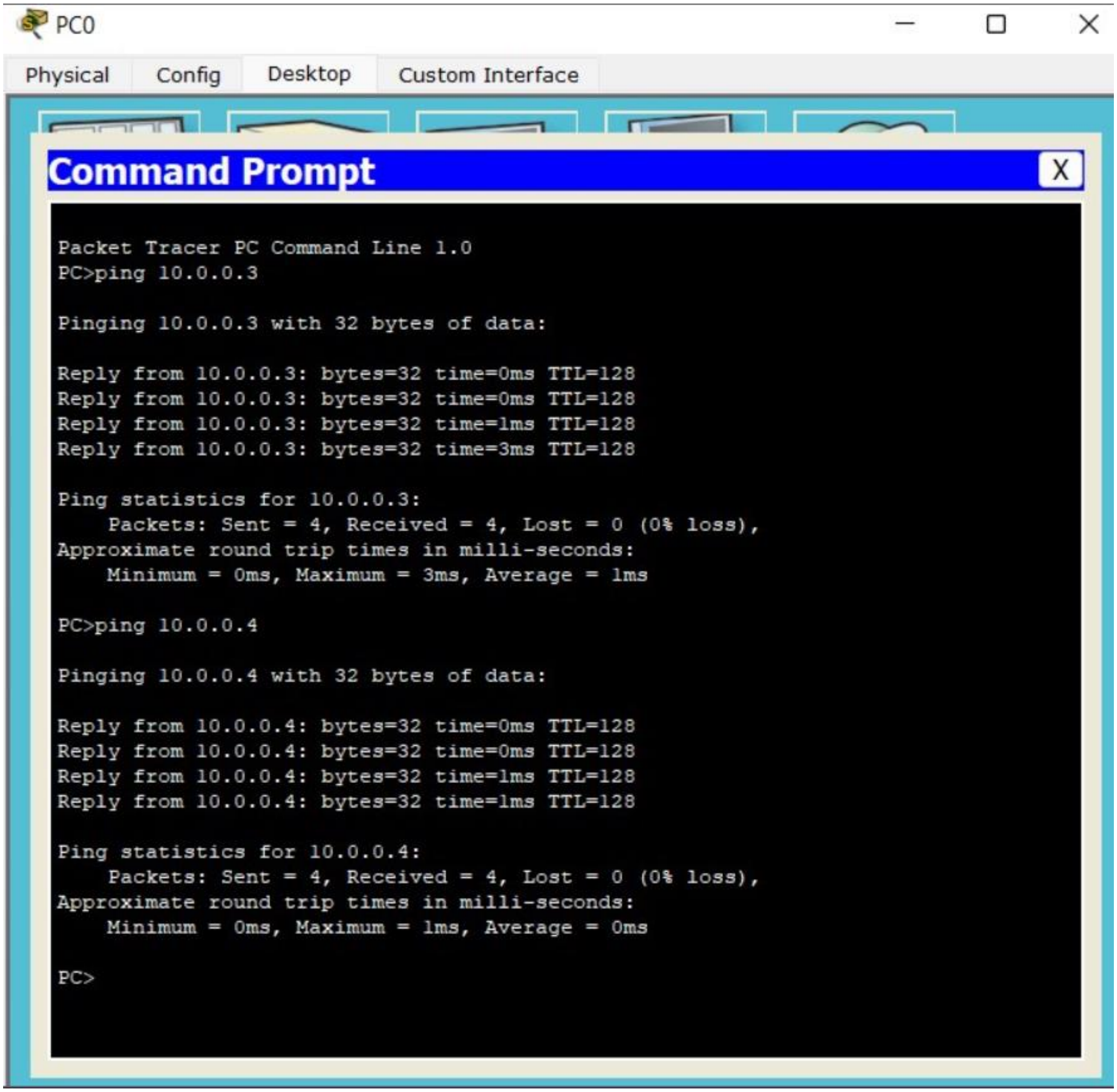
Then save.

-> Click on each PC and go to desktop tab

-> Click on IP configuration then click DHCP.

-> If no error, it will show successful.

Output:

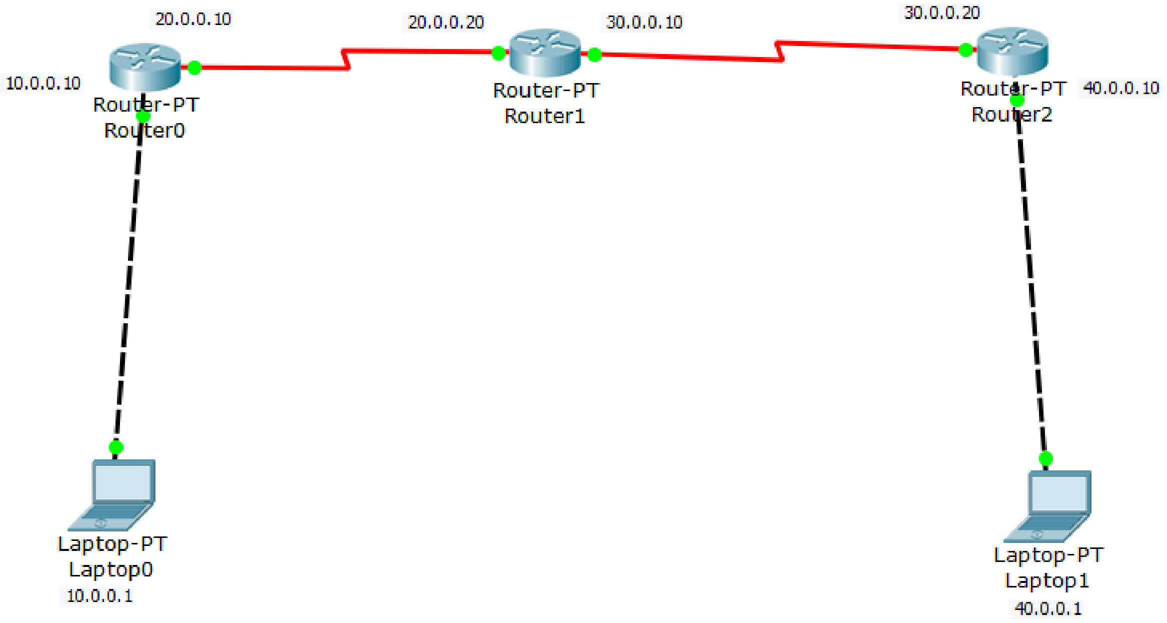


```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=3ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 1ms

PC>ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128

Ping statistics for 10.0.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>
```

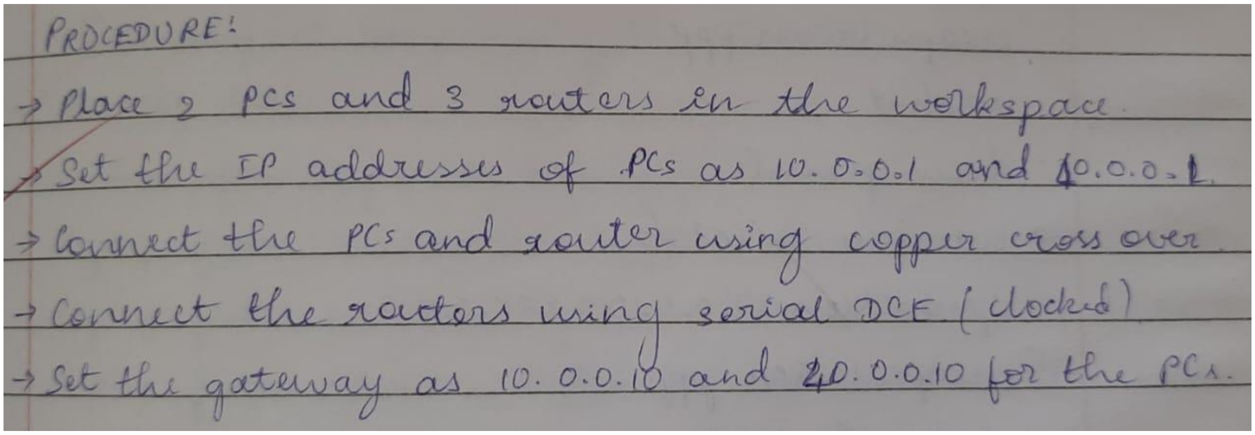
Experiment No – 5

Aim: Configuring RIP Routing Protocol in Routers.

Topology:



Procedure:



PROCEDURE:

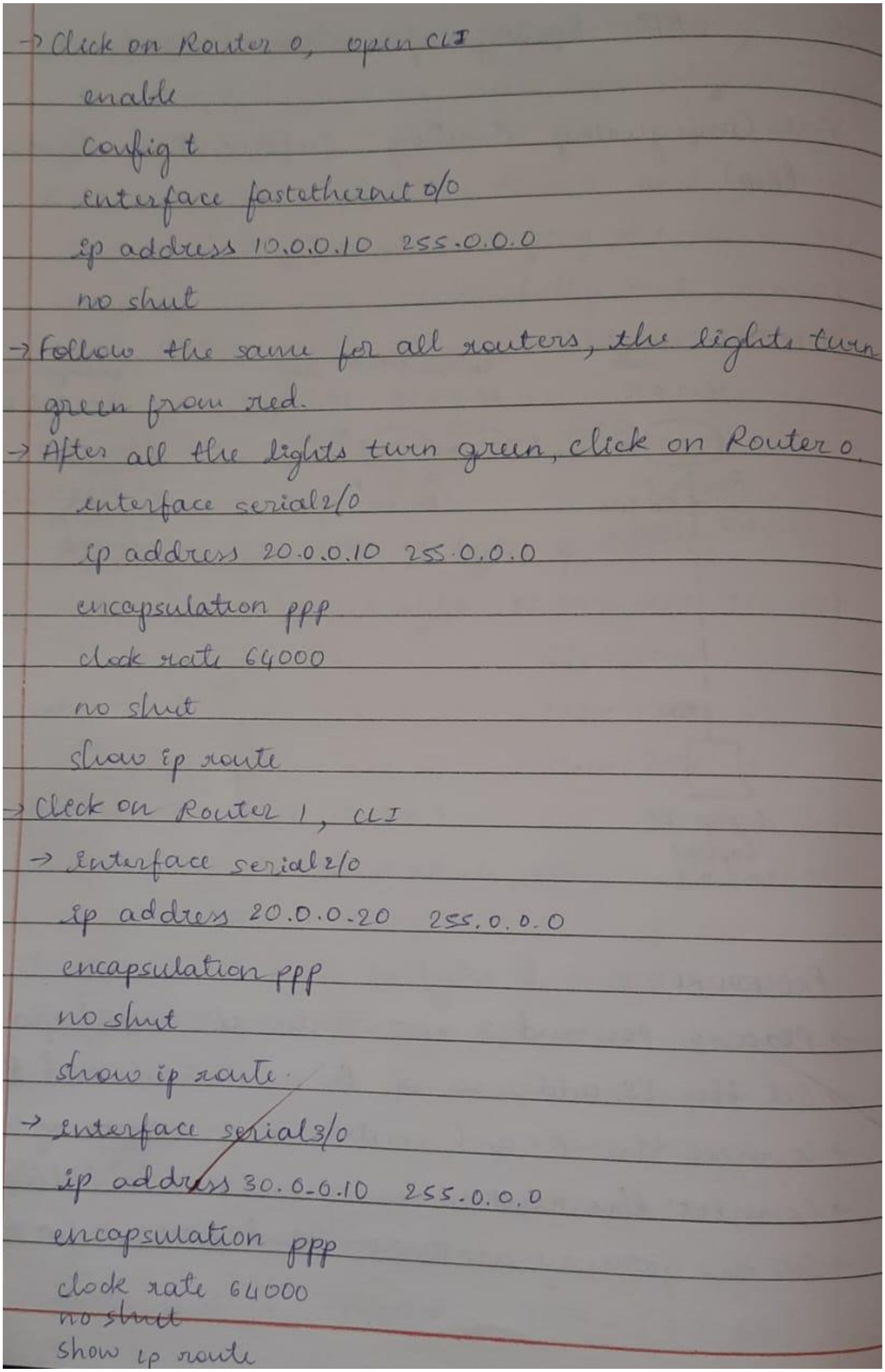
→ Place 2 PCs and 3 routers in the workspace.

→ Set the IP addresses of PCs as 10.0.0.1 and 40.0.0.1.

→ Connect the PCs and router using copper cross over.

→ Connect the routers using serial DCE (clocked)

→ Set the gateway as 10.0.0.10 and 40.0.0.10 for the PCs.



-> Click on Router 0, open CLI

```
enable
config t
enterface fastethernet 0/0
ip address 10.0.0.10 255.0.0.0
no shut
```

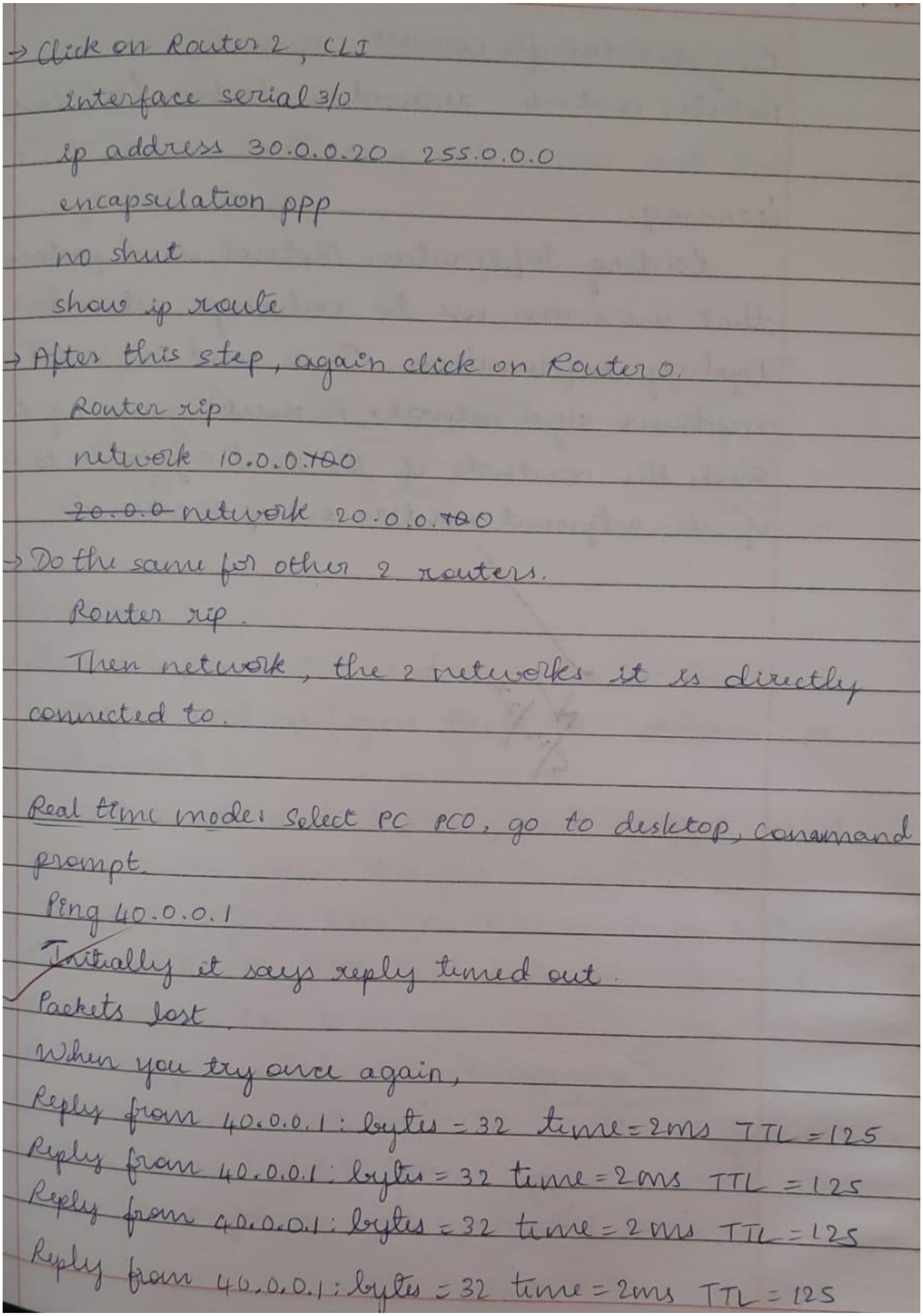
-> Follow the same for all routers, the lights turn green from red.

-> After all the lights turn green, click on Router 0.

```
enterface serial2/0
ip address 20.0.0.10 255.0.0.0
encapsulation ppp
clock rate 64000
no shut
show ip route
```

-> Click on Router 1, CLI

```
-> interface serial2/0
ip address 20.0.0.20 255.0.0.0
encapsulation ppp
no shut
show ip route.
-> enterface serial3/0
ip address 30.0.0.10 255.0.0.0
encapsulation ppp
clock rate 64000
no shut
show ip route
```



→ Click on Router 2, CLI

```
interface serial 3/0
ip address 30.0.0.20 255.0.0.0
encapsulation ppp
no shut
show ip route
```

→ After this step, again click on Router 0.

```
Router rip
network 10.0.0.~~10~~0
~~20.0.0~~ network 20.0.0.~~10~~0
```

→ Do the same for other 2 routers.

Router rip.

Then network, the 2 networks it is directly connected to.

Real time mode: Select PC PC0, go to desktop, command prompt.

```
Ping 40.0.0.1
```

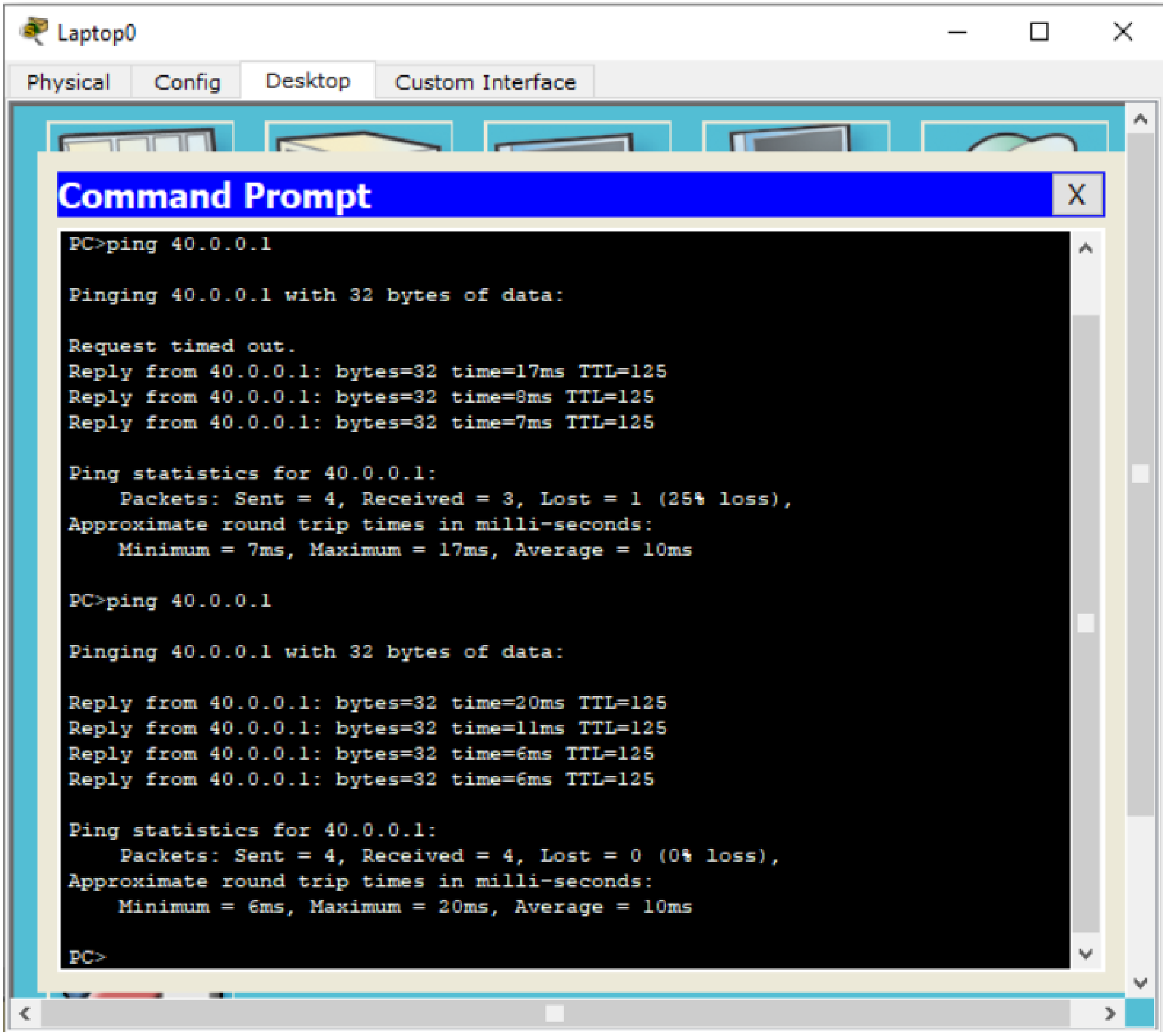
Initially it says reply timed out.

Packets lost.

When you try once again,

```
Reply from 40.0.0.1: bytes = 32 time=2ms TTL =125
Reply from 40.0.0.1: bytes = 32 time = 2ms TTL =125
Reply from 40.0.0.1: bytes = 32 time = 2ms TTL = 125
Reply from 40.0.0.1: bytes = 32 time = 2ms TTL = 125
```

Output:

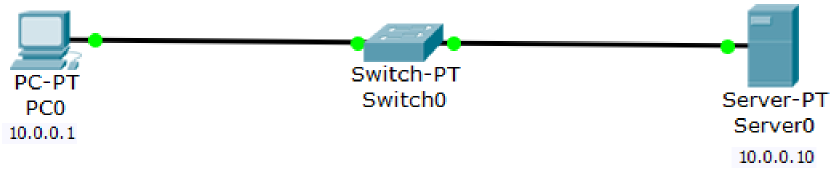


```
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.1: bytes=32 time=17ms TTL=125
Reply from 40.0.0.1: bytes=32 time=8ms TTL=125
Reply from 40.0.0.1: bytes=32 time=7ms 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 = 7ms, Maximum = 17ms, Average = 10ms

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=20ms TTL=125
Reply from 40.0.0.1: bytes=32 time=11ms TTL=125
Reply from 40.0.0.1: bytes=32 time=6ms TTL=125
Reply from 40.0.0.1: bytes=32 time=6ms 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 = 6ms, Maximum = 20ms, Average = 10ms

PC>
```

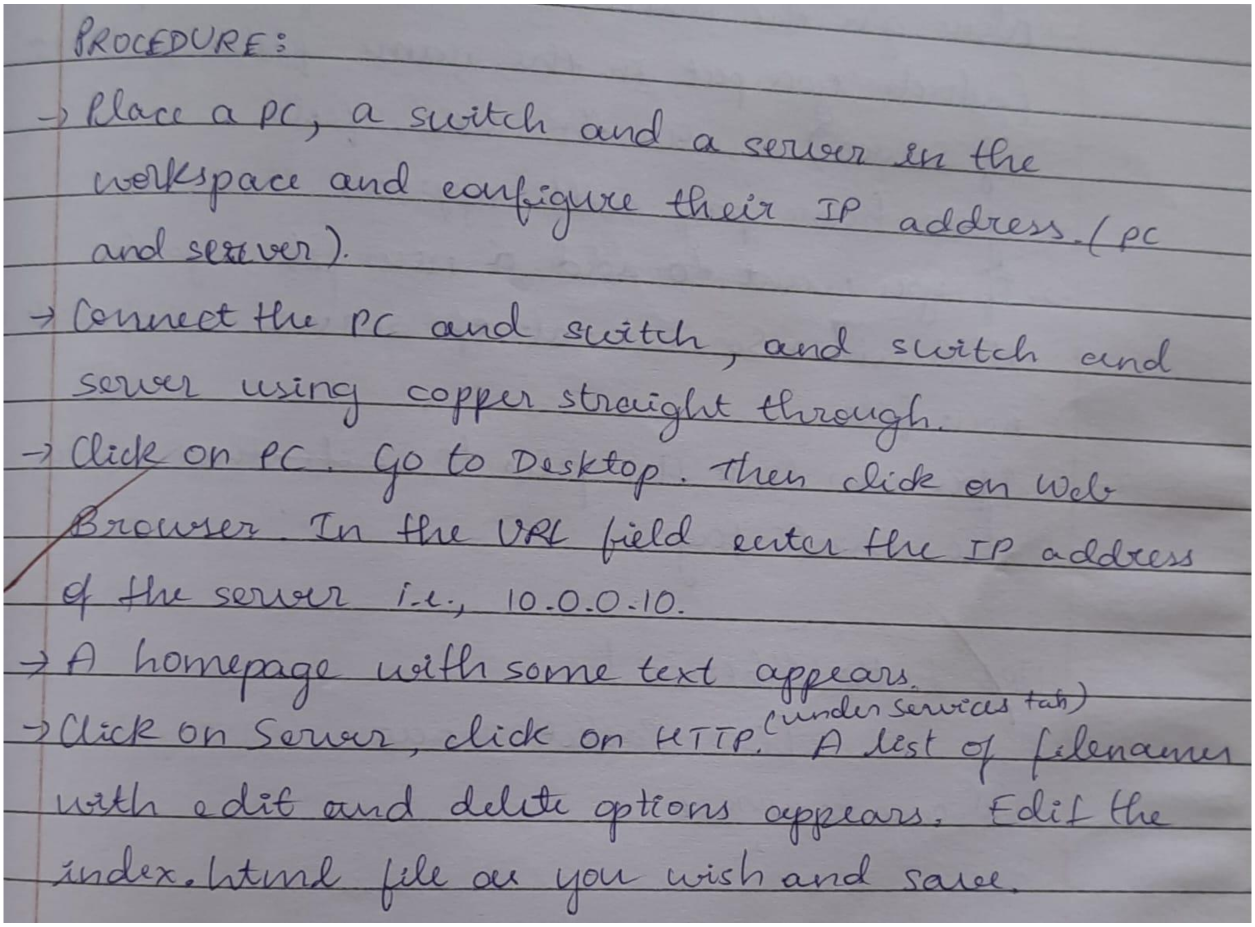
Experiment No – 6

Aim: Demonstration of WEB server and DNS using Packet Tracer.

Topology:



Procedure:



PROCEDURE:

→ Place a PC, a switch and a server in the workspace and configure their IP address. (PC and server).

→ Connect the PC and switch, and switch and server using copper straight through.

→ Click on PC. Go to Desktop. Then click on Web Browser. In the URL field enter the IP address of the server i.e., 10.0.0.10.

→ A homepage with some text appears.

→ Click on Server, click on HTTP (under Services tab). A list of filenames with edit and delete options appears. Edit the index.html file as you wish and save.

-> Go back to PC -> desktop -> Web browser.

-> Give server IP address (10.0.0.10) in the URL field.

-> The homepage with updated changes should appear.

-> Go back to server. Under Services tab, click on DNS. Click on ON.

-> Set a name for your home page.

-> Give the address as 10.0.0.10 , Then add.

-> Now go back to PC -> desktop -> Web browser.

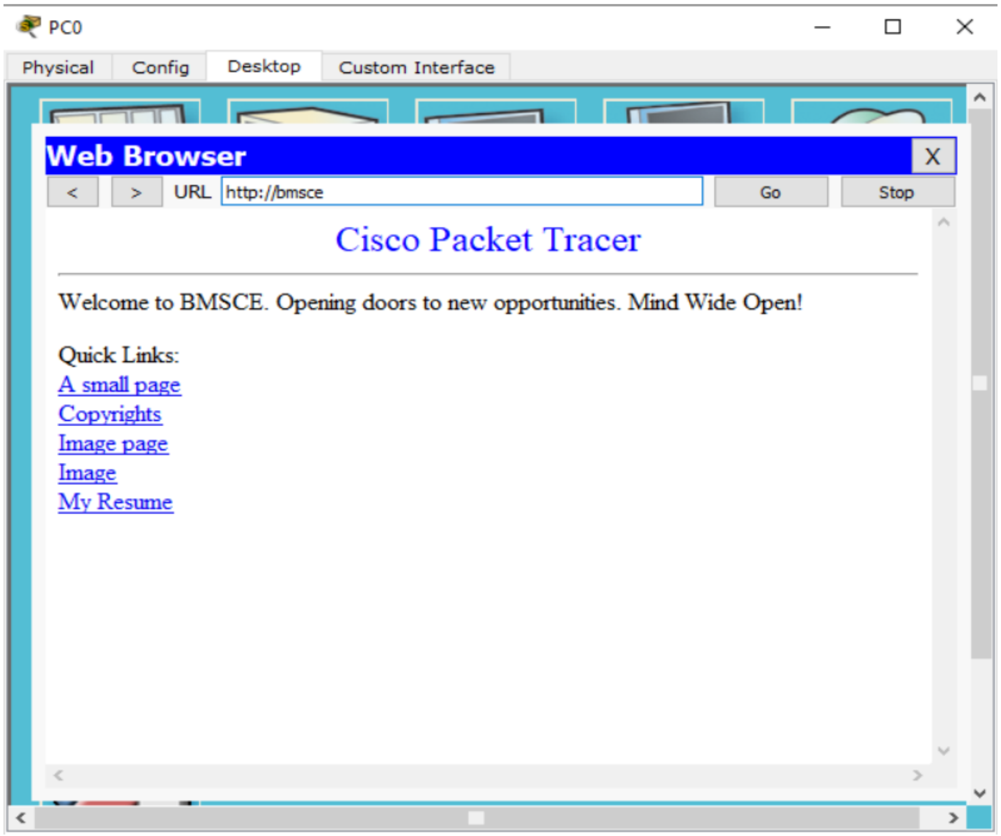
-> Now give the name to chose for the page (which you put in the name field of DNS of Server), under the URL field.

-> Your home page will appear.

-> If you want to add a new file, go to HTTP under Services tab of server and click on new file.

-> You can link the files to each other using the ahref tag.

Output:

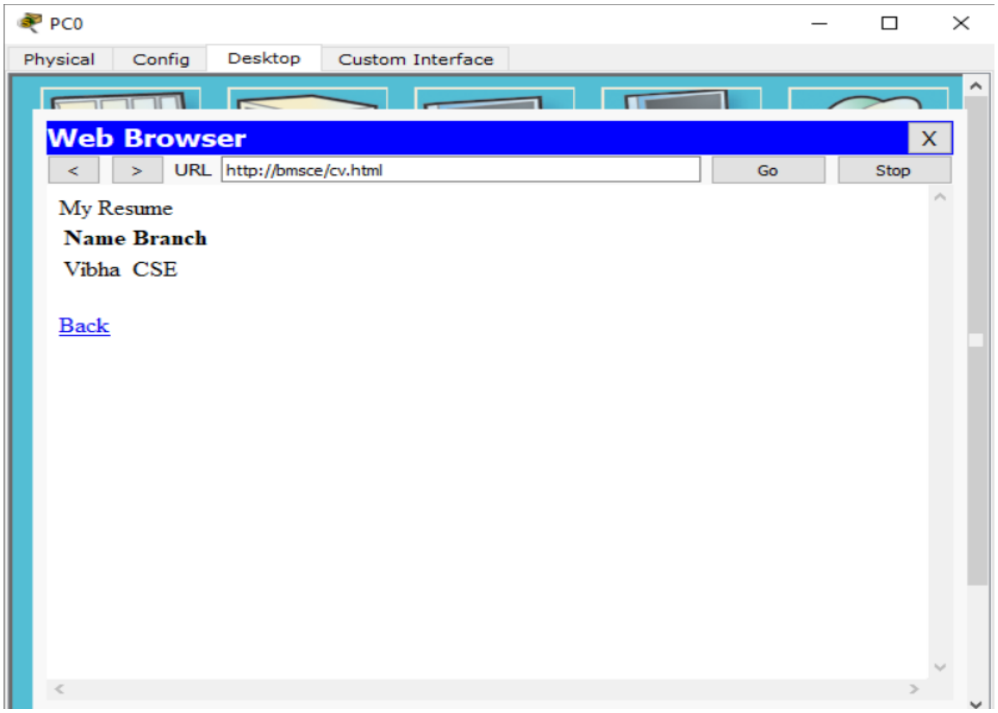


http://bmsce

Cisco Packet Tracer

Welcome to BMSCE. Opening doors to new opportunities. Mind Wide Open!

Quick Links:
A small page
Copyrights
Image page
Image
My Resume



http://bmsce/cv.html

My Resume

Name	**Branch**
Vibha	CSE

Back

Cycle - 2

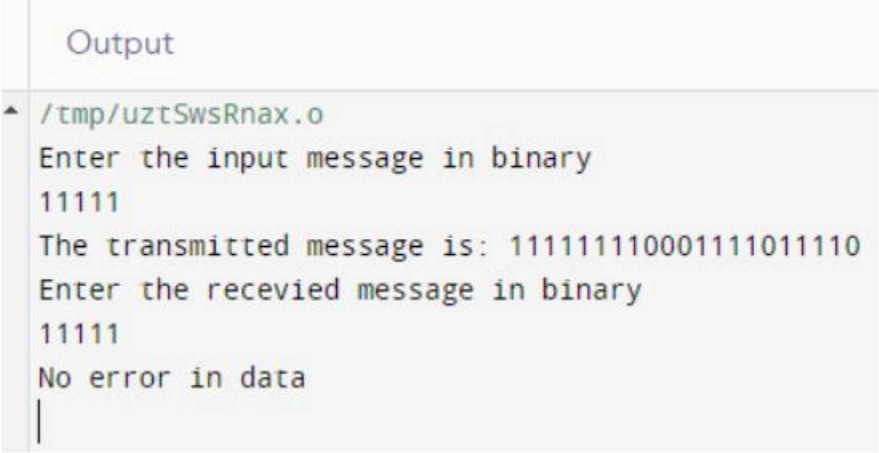
Experiment No – 1

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

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include <iostream>
#include <string.h>
using namespace std;

int crc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op, ip);
    if (mode) {
        for (int i = 1; i < strlen(poly); i++)
            strcat(op, "0");
    }
    /* Perform XOR on the msg with the selected polynomial */
    for (int i = 0; i < strlen(ip); i++) {
        if (op[i] == '1') {
            for (int j = 0; j < strlen(poly); j++) {
                if (op[i + j] == poly[j])
                    op[i + j] = '0';
                else
                    op[i + j] = '1';
            }
        }
    }
    /* check for errors. return 0 if error detected */
    for (int i = 0; i < strlen(op); i++)
        if (op[i] == '1')
            return 0;
    return 1;
}

int main()
{
    char ip[50], op[50], recv[50];
    char poly[] = "10001000000100001";

    cout << "Enter the input message in binary"<< endl;
    cin >> ip;
    crc(ip, op, poly, 1);
    cout << "The transmitted message is: " << ip << op + strlen(ip) << endl;
    cout << "Enter the recevied message in binary" << endl;
    cin >> recv;
    if (crc(recv, op, poly, 0))
        cout << "No error in data" << endl;
    else
        cout << "Error in data transmission has occurred" << endl;

    return 0;
}
```

Output:



```
/tmp/uztSwsRnax.o
Enter the input message in binary
11111
The transmitted message is: 111111110001111011110
Enter the recevied message in binary
11111
No error in data
```

Experiment No – 2

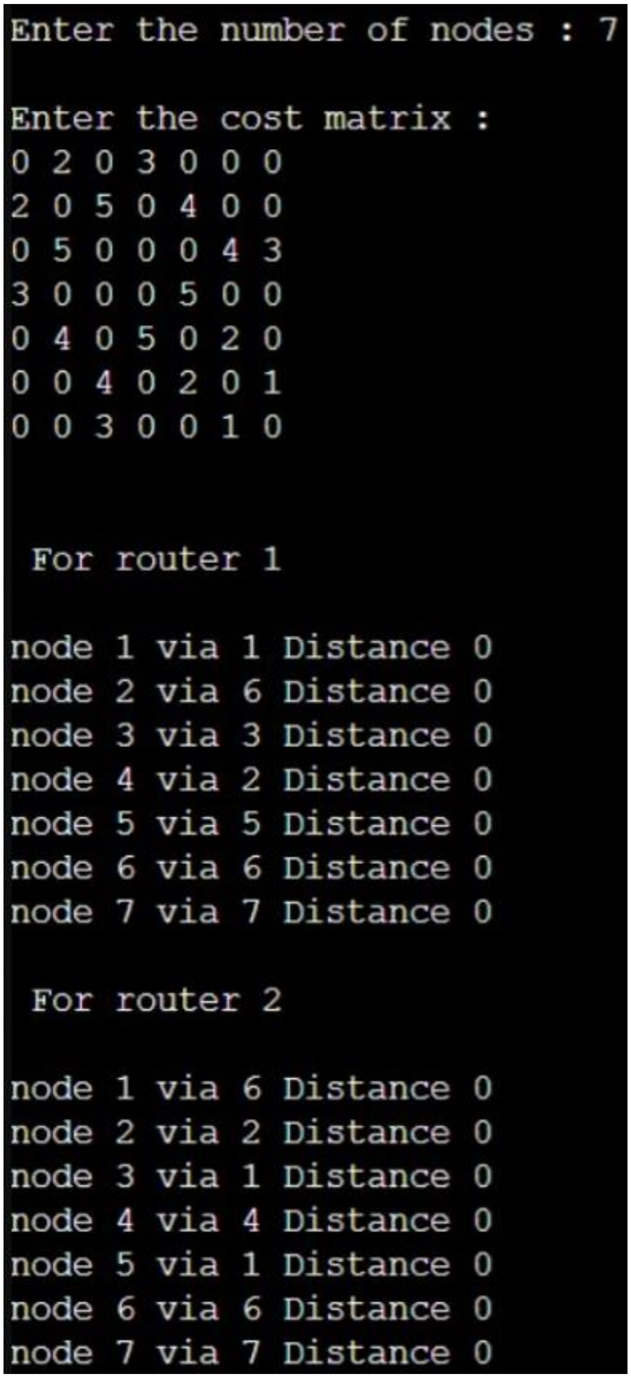
Aim: Write a program for distance vector algorithm to find suitable path for transmission.

Program:

```
#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);
   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];
         rt[i].from[j]=j;
```

```
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {
                    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");
}
```

Output:



```
Enter the number of nodes : 7

Enter the cost matrix :
0 2 0 3 0 0 0
2 0 5 0 4 0 0
0 5 0 0 0 4 3
3 0 0 0 5 0 0
0 4 0 5 0 2 0
0 0 4 0 2 0 1
0 0 3 0 0 1 0



 For router 1

node 1 via 1 Distance 0
node 2 via 6 Distance 0
node 3 via 3 Distance 0
node 4 via 2 Distance 0
node 5 via 5 Distance 0
node 6 via 6 Distance 0
node 7 via 7 Distance 0

 For router 2

node 1 via 6 Distance 0
node 2 via 2 Distance 0
node 3 via 1 Distance 0
node 4 via 4 Distance 0
node 5 via 1 Distance 0
node 6 via 6 Distance 0
node 7 via 7 Distance 0
```

Experiment No – 3

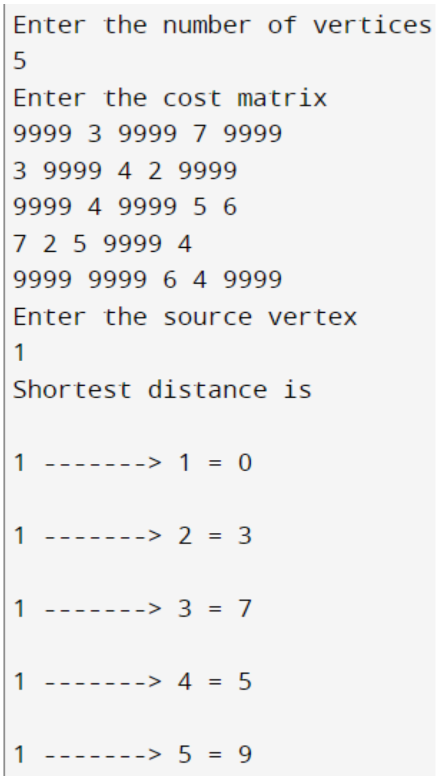
Aim: Implement Dijkstra's algorithm to compute the shortest path for a given topology.

Program:

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

int main()
{
    printf("\nenter the number of vertices\n");
    scanf("%d",&n);
    printf("\nenter the cost matrix \n");
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            scanf("%d",&c[i][j]);
        }
    }
    printf("\nenter the source vertex\n");
    scanf("%d",&src);
    dijkistra();
    return 1;
}
```

```
void dijkistra()
{
    int dist[10],vis[10],j,count,min,u;
    for(j=1;j<=n;j++)
    {
        dist[j]=c[src][j];
    }
    for(j=1;j<=n;j++)
    {
        vis[j]=0;
    }
    dist[src]=0;
    vis[src]=1;
    count=1;
    while(count!=n)
    {
        min=9999;
        for(j=1;j<=n;j++)
        {
            if(dist[j]<min && vis[j]!=1)
            {
                min=dist[j];
                u=j;
            }
        }
        vis[u]=1;
        count++;
        for(j=1;j<=n;j++)
        {
            if(min+c[u][j]<dist[j] && vis[j]!=1)
            {
                dist[j]=min+c[u][j];
            }
        }
    }
    printf("\n shortest distance is \n");
    for(j=1;j<=n;j++)
    {
        printf("\n%d -------> %d = %d \n ",src,j,dist[j]);
    }
}
```

Output:



```
Enter the number of vertices
5
Enter the cost matrix
9999 3 9999 7 9999
3 9999 4 2 9999
9999 4 9999 5 6
7 2 5 9999 4
9999 9999 6 4 9999
Enter the source vertex
1
Shortest distance is

1 -------> 1 = 0

1 -------> 2 = 3

1 -------> 3 = 7

1 -------> 4 = 5

1 -------> 5 = 9
```

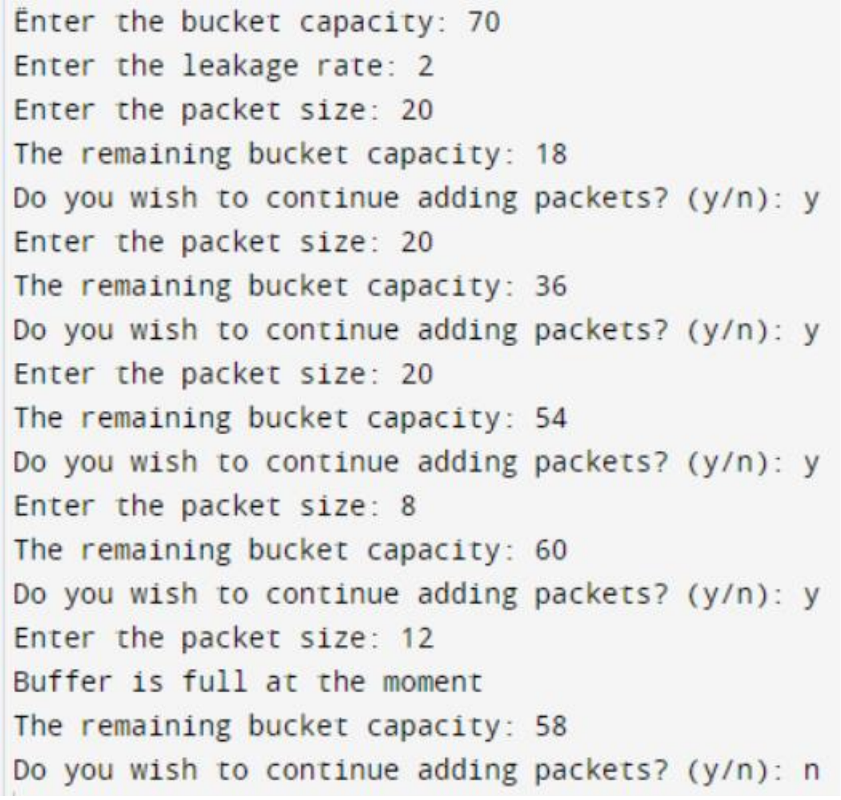
Experiment No – 4

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

Program:

```
#include <iostream>
using namespace std;

int main()
{
    int bsize=0,capacity=0,psize=0,rate=0;
    char ans='y';

    cout << "Ënter the bucket capacity: ";
    cin>>capacity;

    cout<<"\nEnter the leakage rate: ";
    cin>>rate;

    while(ans=='y' || ans=='Y'){
        cout<<"\n\nEnter the packet size: ";
        cin>>psize;

        if((bsize+psize)>capacity)
        cout<<"\nBuffer is full at the moment";

        else if((bsize+psize)<=capacity)
        bsize+=psize;
        bsize-=rate;
        cout<<"\nThe remaining bucket capacity: "<<bsize;


        cout<<"\nDo you wish to continue adding packets? (y/n): ";
        cin>>ans;
    }
    return 0;
}
```

Output:



```
Enter the bucket capacity: 70
Enter the leakage rate: 2
Enter the packet size: 20
The remaining bucket capacity: 18
Do you wish to continue adding packets? (y/n): y
Enter the packet size: 20
The remaining bucket capacity: 36
Do you wish to continue adding packets? (y/n): y
Enter the packet size: 20
The remaining bucket capacity: 54
Do you wish to continue adding packets? (y/n): y
Enter the packet size: 8
The remaining bucket capacity: 60
Do you wish to continue adding packets? (y/n): y
Enter the packet size: 12
Buffer is full at the moment
The remaining bucket capacity: 58
Do you wish to continue adding packets? (y/n): n
```

Experiment No – 5

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.

Program:

Server:

```
from socket import *
serverName='DESKTOP-9CJQB77'
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()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    file.close()
    connectionSocket.close()
```

Client:

```
from socket import *
serverName='DESKTOP-9CJQB77'
serverPort=12530
clientSocket=socket(AF_INET,SOCK_STREAM)
```

clientSocket.connect((serverName,serverPort))

sentence=input("Enter file name")

clientSocket.send(sentence.encode())

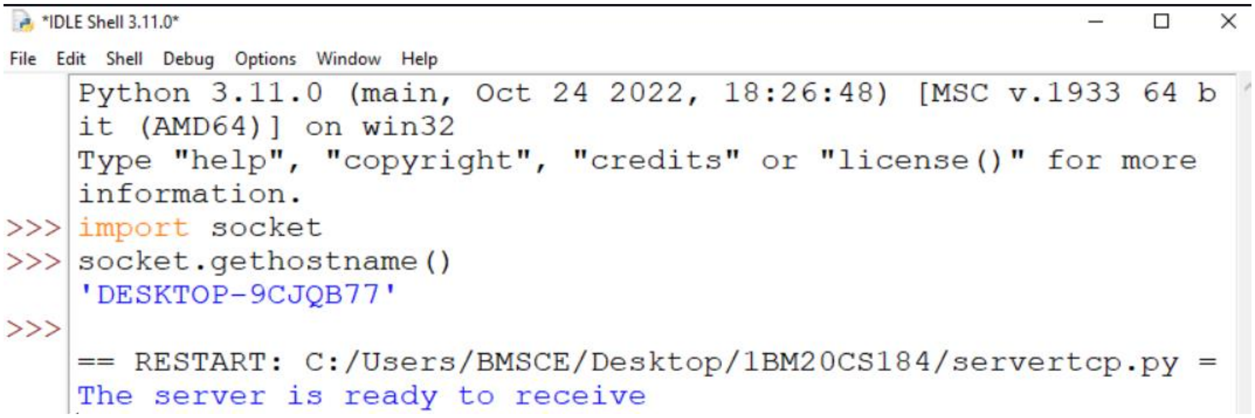
filecontents=clientSocket.recv(1024).decode()

print('From Server:',filecontents)

clientSocket.close()

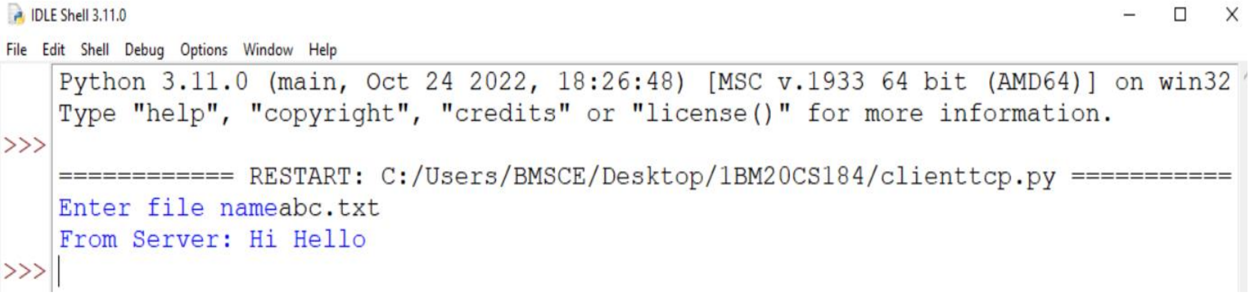
Output:

Server:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>> import socket
>>> socket.gethostname()
'DESKTOP-9CJQB77'
>>>
== RESTART: C:/Users/BMSCE/Desktop/1BM20CS184/servertcp.py =
The server is ready to receive
```

Client:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/BMSCE/Desktop/1BM20CS184/clienttcp.py ===========
Enter file nameabc.txt
From Server: Hi Hello
>>>
```

Experiment No – 6

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.

Program:

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")
 l=file.read(2048)
 serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
 print ('\nSent contents of ', end = ' ')
 print (sentence)
 # for i in sentence:
 # print (str(i), end = '')
 file.close()
```

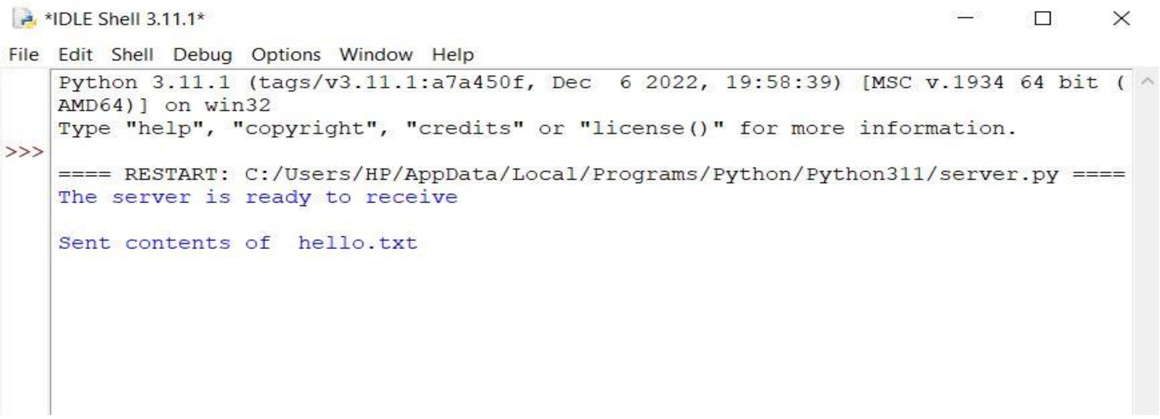
Client:

```
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()
```

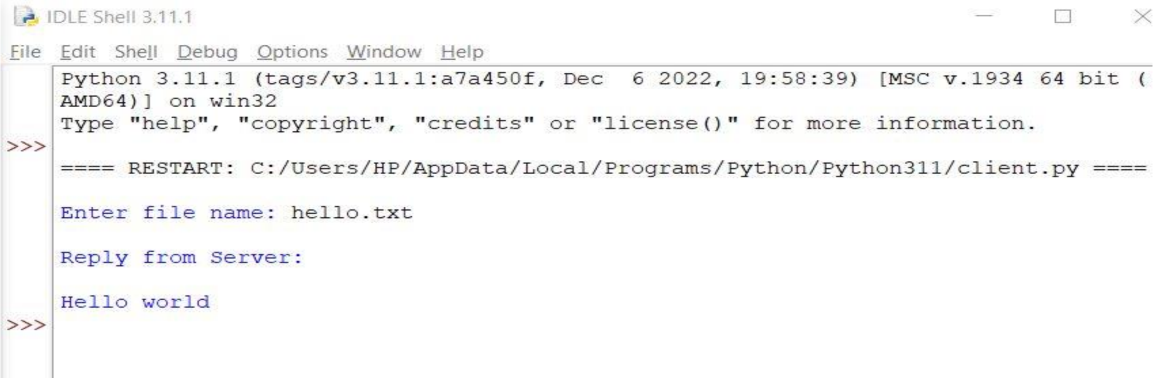
Output:

Server:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/HP/AppData/Local/Programs/Python/Python311/server.py ====
The server is ready to receive

Sent contents of  hello.txt
```

Client:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/HP/AppData/Local/Programs/Python/Python311/client.py ====

Enter file name: hello.txt

Reply from Server:

Hello world
>>>
```