VISVESVARAYATECHNOLOGICALUNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Vagisha Ajay (1BM22CS346)

in partial fulfillment for the award of the degree of

BACHELOROFENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Academic Year 2024-25 (odd)

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 Network (23CS5PCCON)" carried out by **Vagisha Ajay (1BM22CS346)** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Spoorthi D M Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	09-10-24	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping messages.	5-8
2	09-10-24	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	9-11
3	16-10-24	Configure default route, static route to the Router (Part 1).	12-15
4	23-10-24	Configure default route, static route to the Router (Part 2).	16-19
5	13-11-24	Configure DHCP within a LAN and outside LAN.	20-23
6	20-11-24	Configure RIP routing Protocol in Routers .	24-27
7	20-11-24	Demonstrate the TTL/ Life of a Packet.	28-31
8	27-11-24	Configure OSPF routing protocol.	32-37
9	18-12-24	Configure Web Server, DNS within a LAN.	38-39
10	18-12-24	To construct a simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	40-42
11	18-12-24	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	43-45
12	18-12-24	To construct a VLAN and make the PC's communicate among a VLAN.	46-50
13	18-12-24	To construct a WLAN and make the nodes communicate wirelessly.	51-54
14	18-12-24	Write a program for error detecting code using CRC-CCITT (16-bits).	55-59
15	18-12-24	Write a program for congestion control using Leaky bucket algorithm.	60-62

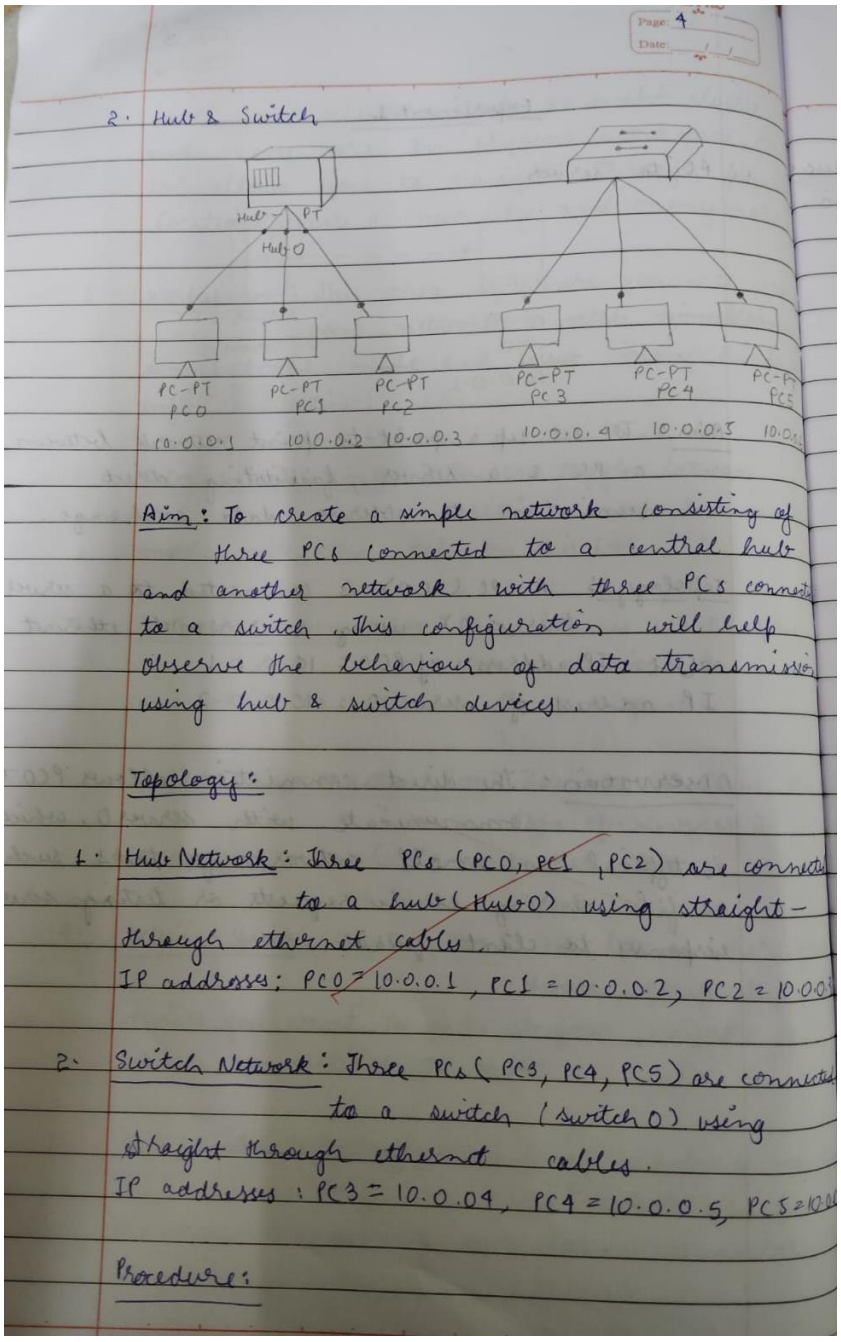
16	18-12-24	Using TCP/IP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.	63-66
17	18-12-24	Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.	67-70

Github Link: https://github.com/VagishaAjay/1BM22CS346_VagishaAjay_CNLab.git

Program 1

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

Topology , Procedure and Observation:



2. Hub & Switch

Aim: To create a simple network consisting of three PCs connected to a central hub and another network with three PCs connected to a switch. This configuration will help observe the behaviour of data transmission using hub & switch devices.

Topology:

1. Hub Network: Three PCs (PC0, PC1, PC2) are connected to a hub (Hub0) using straight-through ethernet cables.
IP addresses: PC0 = 10.0.0.1, PC1 = 10.0.0.2, PC2 = 10.0.0.3

2. Switch Network: Three PCs (PC3, PC4, PC5) are connected to a switch (Switch0) using straight through ethernet cables.
IP addresses: PC3 = 10.0.0.4, PC4 = 10.0.0.5, PC5 = 10.0.0.6

Procedure:

1. Add 1 hub, 1 switch and 6 PCs (PC0, PC1, PC2 for the hub; PC3, PC4, PC5 for the switch) to the cisco packet tracer workspace

2. Use copper straight-through cables to connect PC0, PC1 and PC2 to Hub0. Similarly connect PC3, PC4 & PC5 to switch0 using some type of cables.

3. Assign IP addresses to each PC & obtain subnet mask.

4. Switch to simulation mode to observe data traffic behaviour when packets are sent between the devices.

5. In the hub network, notice how the hub broadcasts packets to all devices, causing potential traffic overload.
In the switch network, observe how the switch forwards packets only to the intended recipient reducing unnecessary traffic.

6. The hub broadcasts data to all connected devices leading to more network congestion, while the switch efficiently sends data only to the correct device, optimizing performance

Observation:

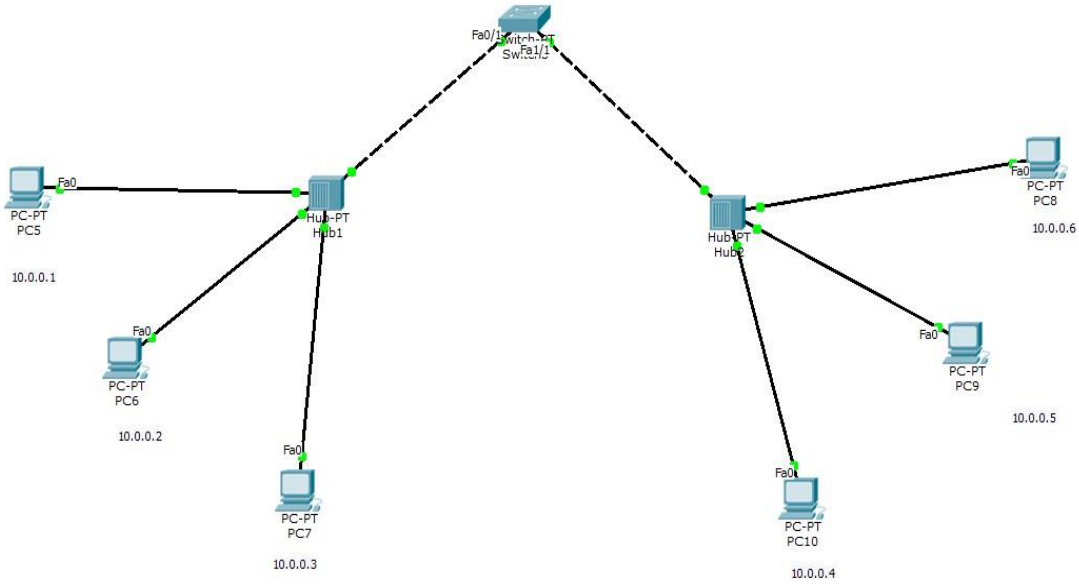
1. The hub broadcasts packets to all devices, which may cause unnecessary traffic.

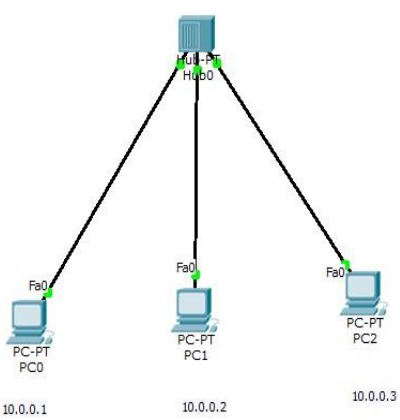
2. The switch forwards packets only to the appropriate device by learning MAC address making it more efficient in reducing traffic.

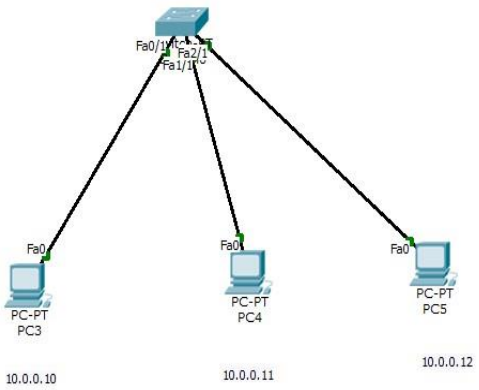
Difference between Hubs & Switches

	Hubs	Switches
1.	Hub broadcast data to all devices.	Switches send it only to the destination.
2.	Hubs create more traffic.	Switches reduces traffic by directing data.
3.	Hubs work at physical layer.	Switches operate at the data link layer.
4.	Hubs are slower due to shared bandwidth.	Switches are faster with dedicated bandwidth.
5.	Hubs are cheaper.	Switches are more expensive but more efficient.

Screen Shots:



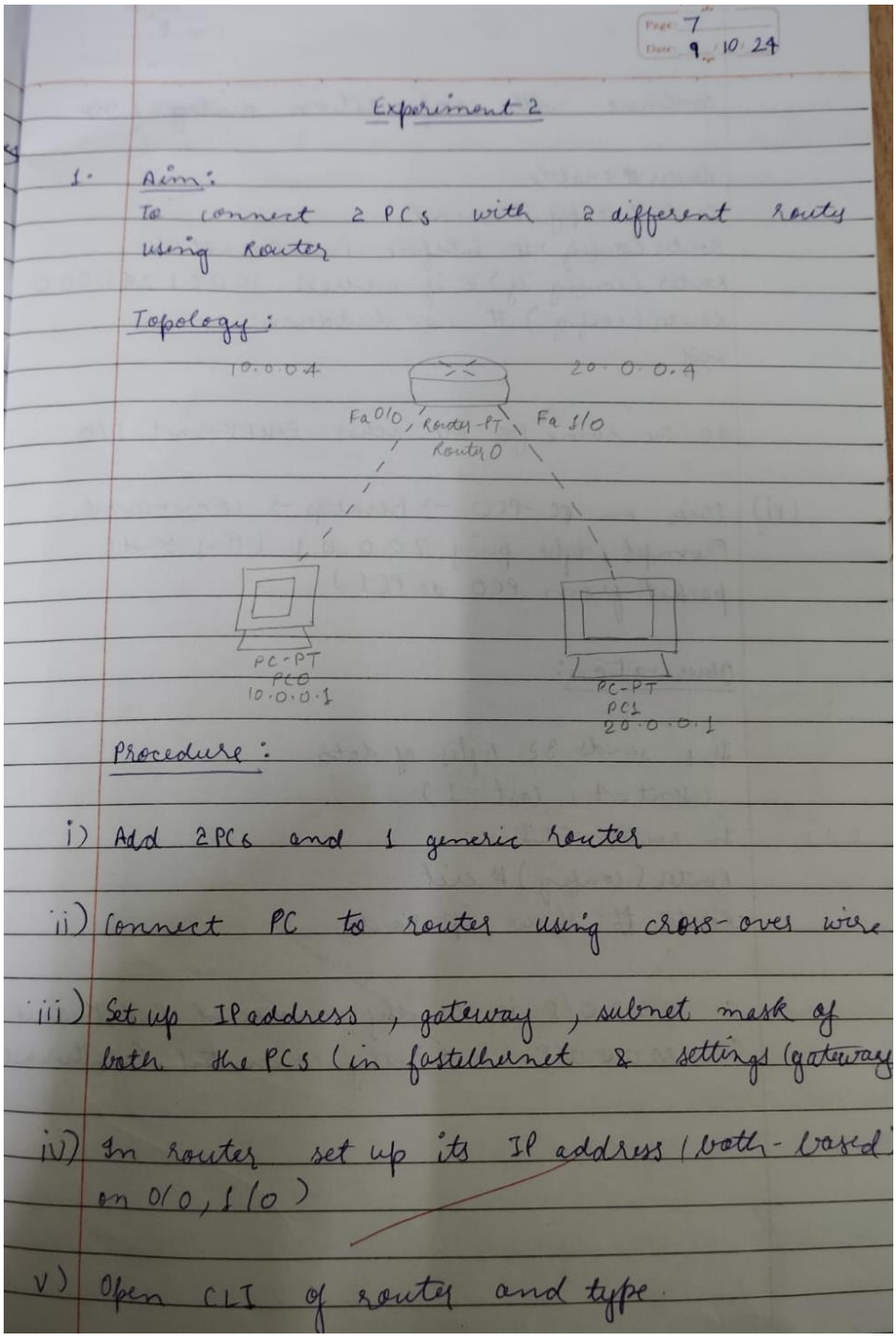




Program 2

Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Topology , Procedure and Observation:



continue with configuration dialog ? no

Router # enable
Router # config terminal
Router (config) # interface FastEthernet 0/0
Router (config-if) # ip address 10.0.0.1 255.0.0.0
Router (config) # no shutdown
exit

do the same for the other FastEthernet 1/0

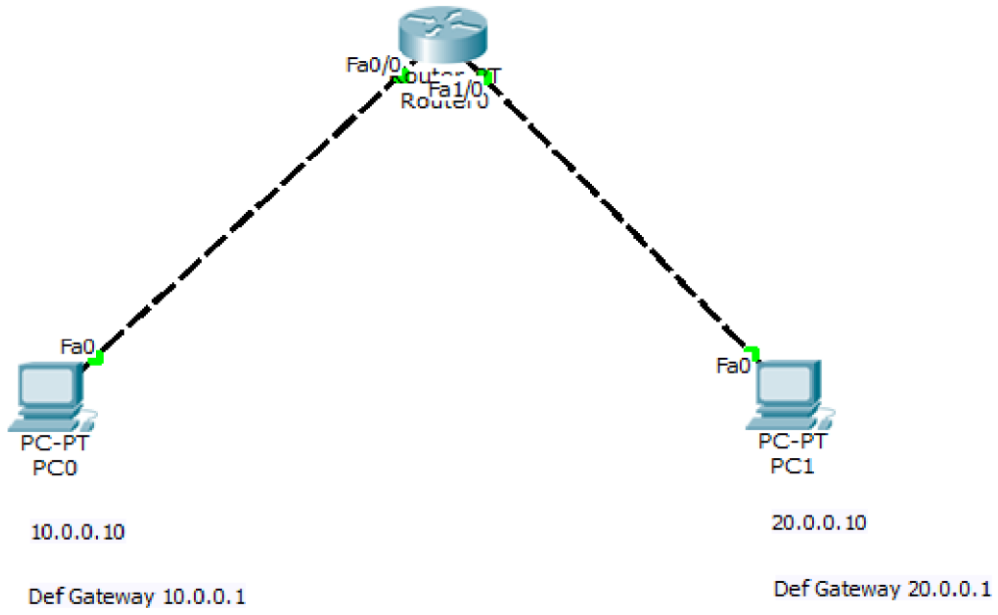
(vi) Click on PC-PC0 → Desktop → command Prompt, type ping 20.0.0.1 (this sends packet from PC0 do PC1)

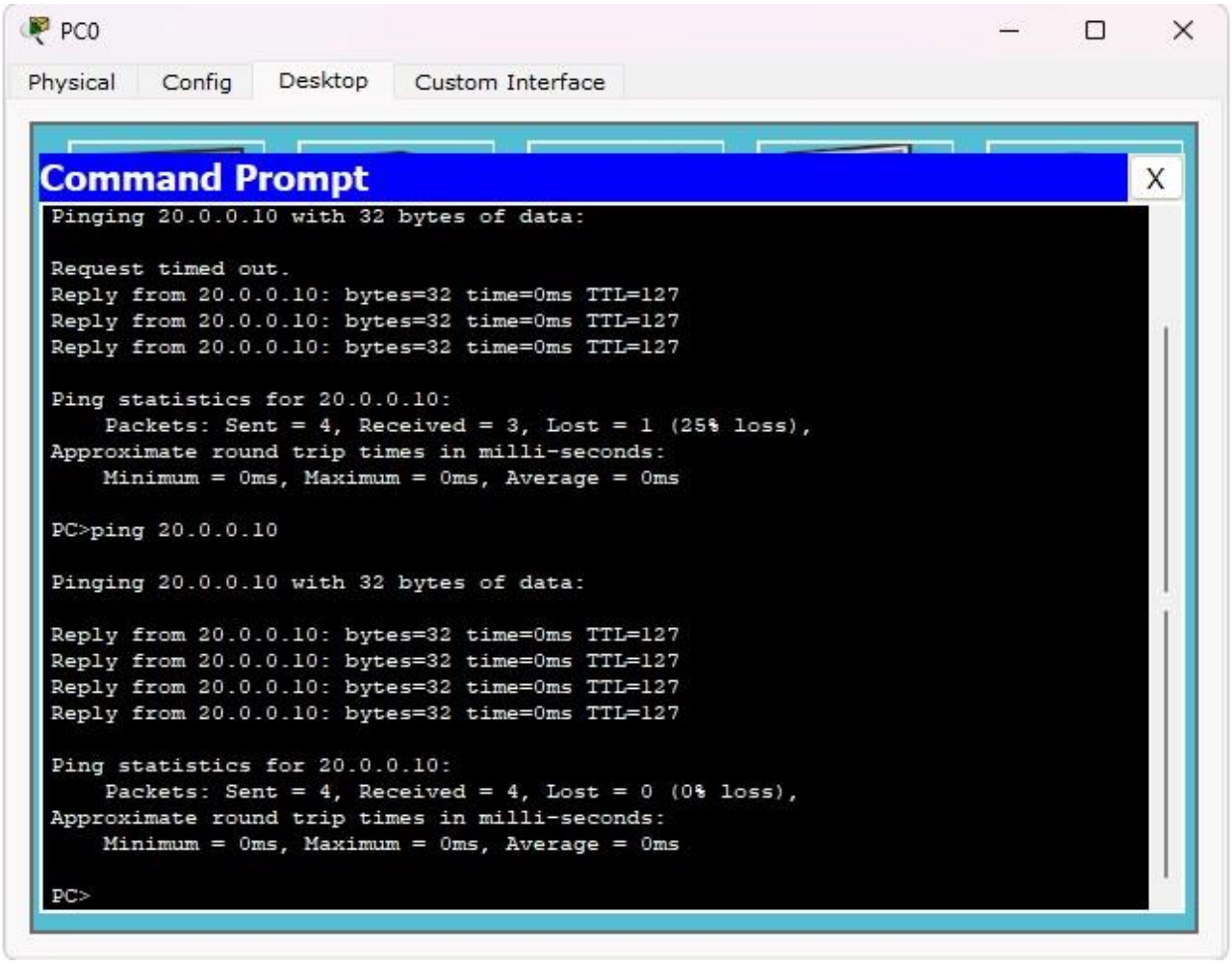
Observation:

This sends 32 bytes of data
(sent = 4, lost = 1)
In router CLI
Router (config) # exit
Router #: show ip route

C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, FastEthernet 1/0

Screen Shots:

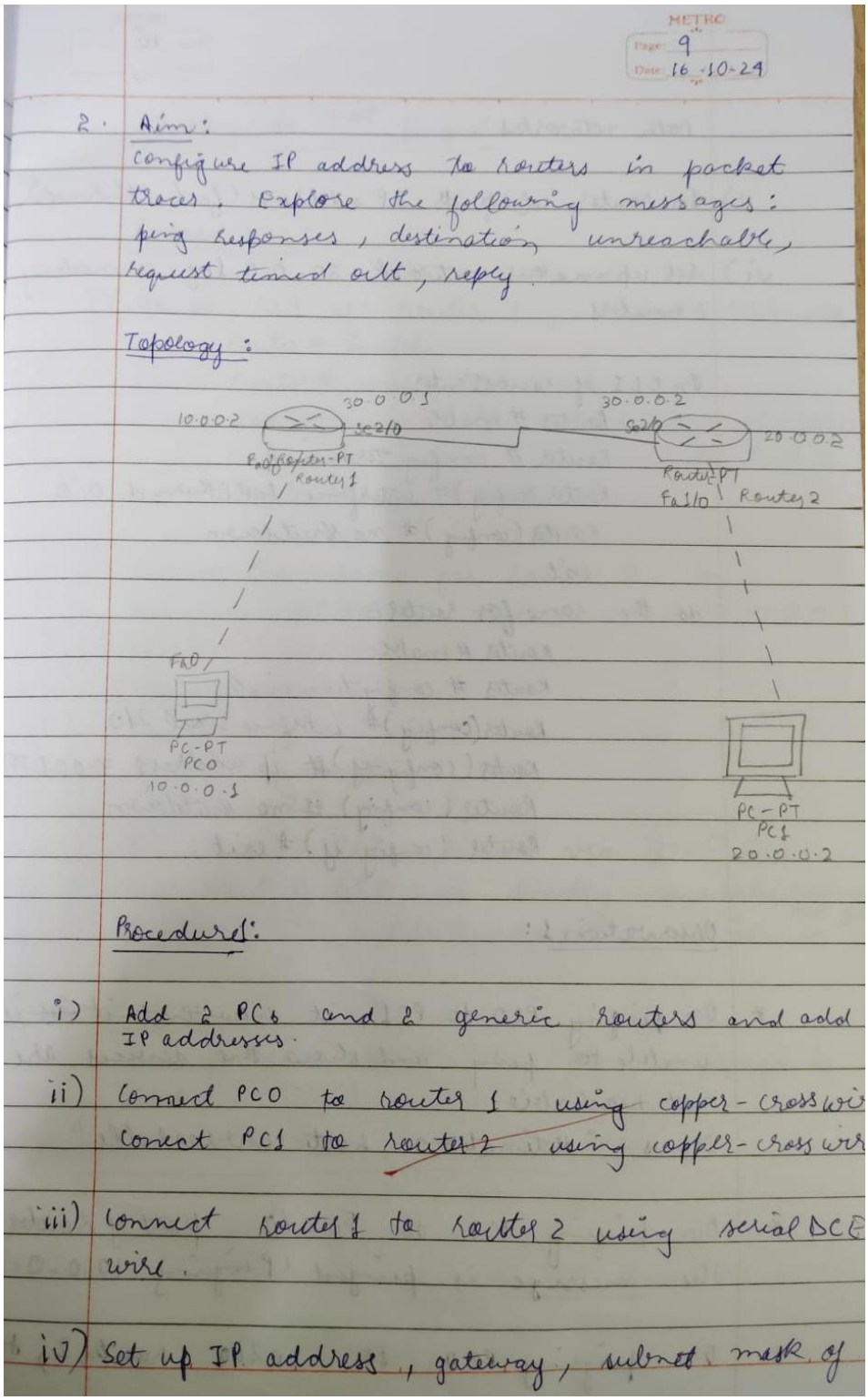




Program 3

Aim: Configure default route, static route to the Router(Part 1).

Topology , Procedure and Observation:



2. Aim:

Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Topology:

Procedure:

i) Add 2 PCs and 2 generic routers and add IP addresses.

ii) Connect PC0 to router 1 using copper-cross wire
Connect PC1 to router 2 using copper-cross wire

iii) Connect router 1 to router 2 using serial DCE wire.

iv) Set up IP address, gateway, subnet mask of

both networks.

v) In routers set up its IP address (fast ethernet)

vi) Set up another network 30.0.0.0 by connecting 2 routers.

In CLI of routers enter
Router # enable
Router # config terminal
Router (config) # interface FastEthernet 0/0
Router (config) # no shutdown
exit

do the same for router 2
Router # enable
Router # config terminal
Router (config) # interface serial 2/0
Router (config-if) # ip address 30.0.0.1 255.0.0.0
Router (config) # no shutdown
Router (config-if) # exit.

Observation 1:

* On pinging PC0 to PC1, it shows that it is unable to ping and shows the devices are not reachable.
Shows - "destination host unreachable".

* On pinging PC0 to Fa0/0 port of router the message is pinged (Pinging 10.0.0.2)

* On pinging PC0 to Se2/0 router 2, the

message is not pinged.

Procedure:

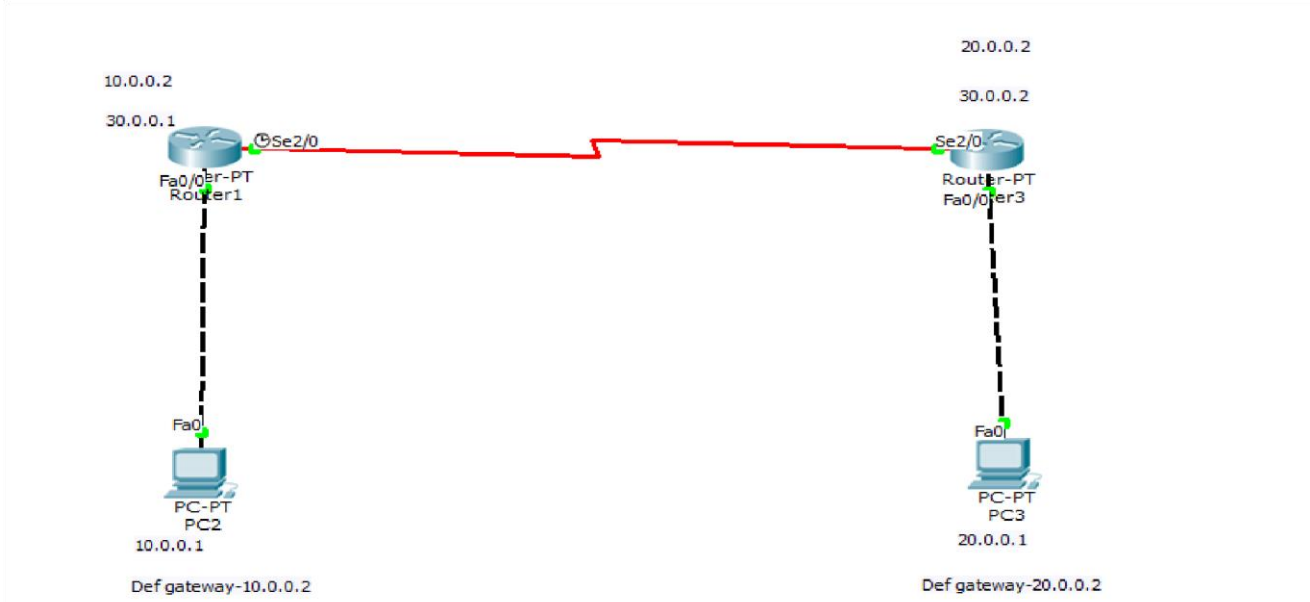
i) Go to CLI of router 1, and in CLI enter
Router# enable
Router# config terminal
Router(config)# ip route 20.0.0.0 255.0.0.0 30.0.0.2
Router(config)# exit

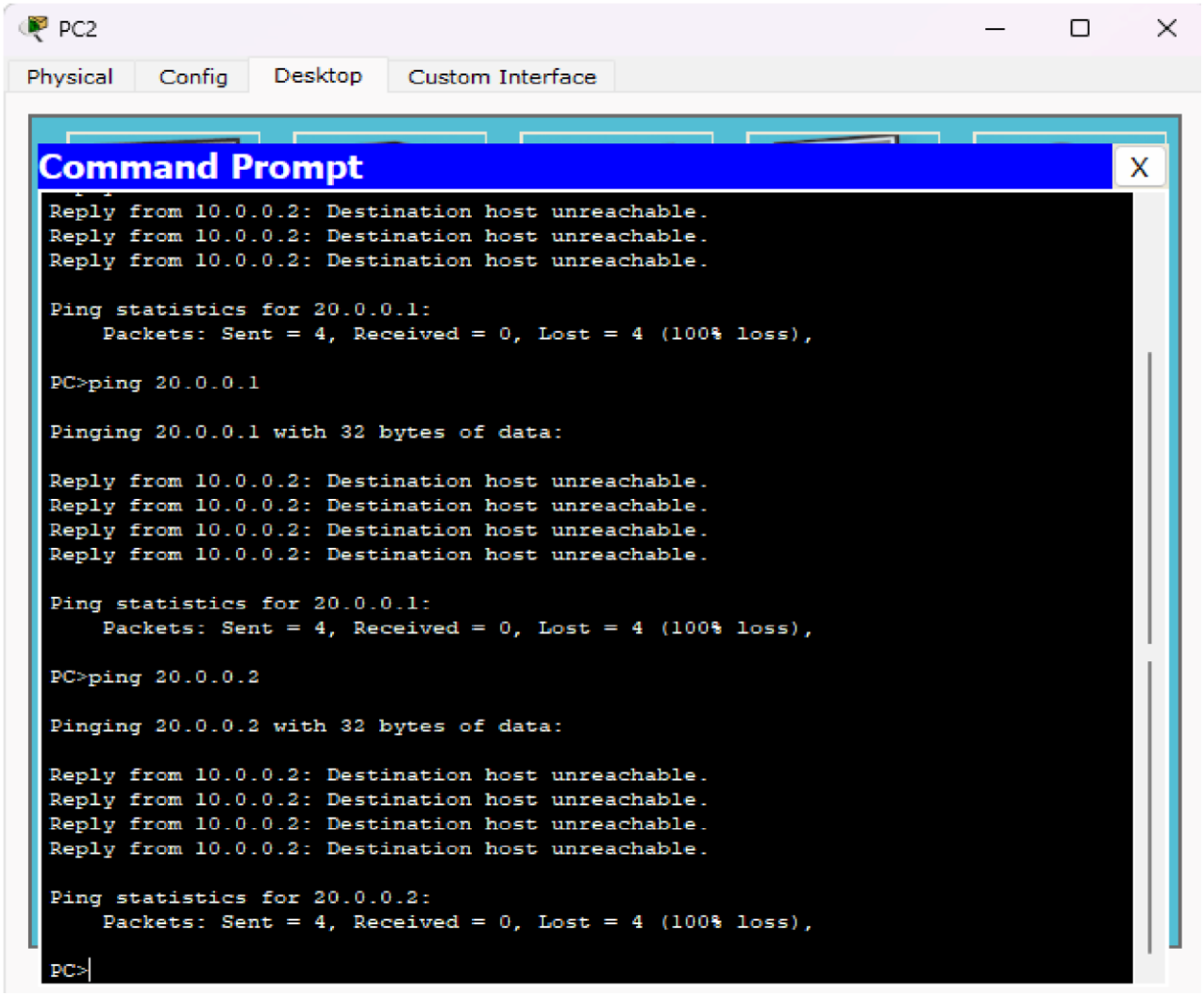
ii) Repeat the same for router 2 by changing 20 to 10 and 30.0.0.2 to 30.0.0.1

Observation 2:
In CLI
show ip route
(for router 2)
S 10.0.0.0/8 [1/0] via 30.0.0.1
C 20.0.0.0/8 is directly connected, Fast Ethernet 0/0
C 30.0.0.0/8 is directly connected, Serial 2/0

Output:
On pinging end devices, the message is send.

Screen Shots:

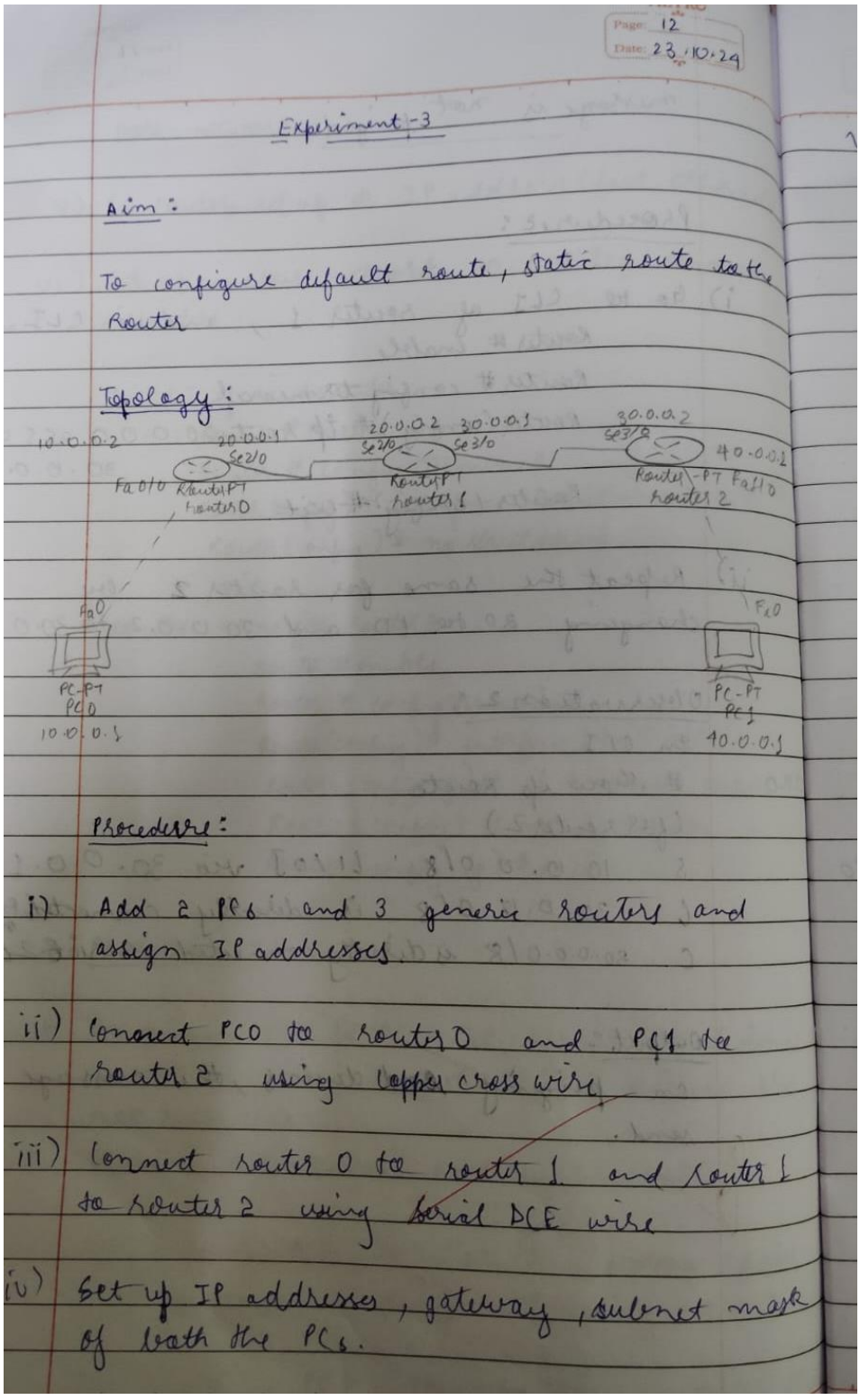




Program 4

Aim: Configure default route, static route to the Router(Part 2).

Topology , Procedure and Observation:



v) In CLI of router 0 enter
Router # enable
Router # config terminal
Router (config) # interface FastEthernet 0/0
Router (config) # ip address 10.0.0.1 255.0.0.0
Router (config) # no shutdown
exit
Router # enable
Router # config terminal
Router (config) # interface serial 2/0
Router (config) # ip address 20.0.0.1 255.0.0.0
Router (config) # no shutdown
Router (config) # exit

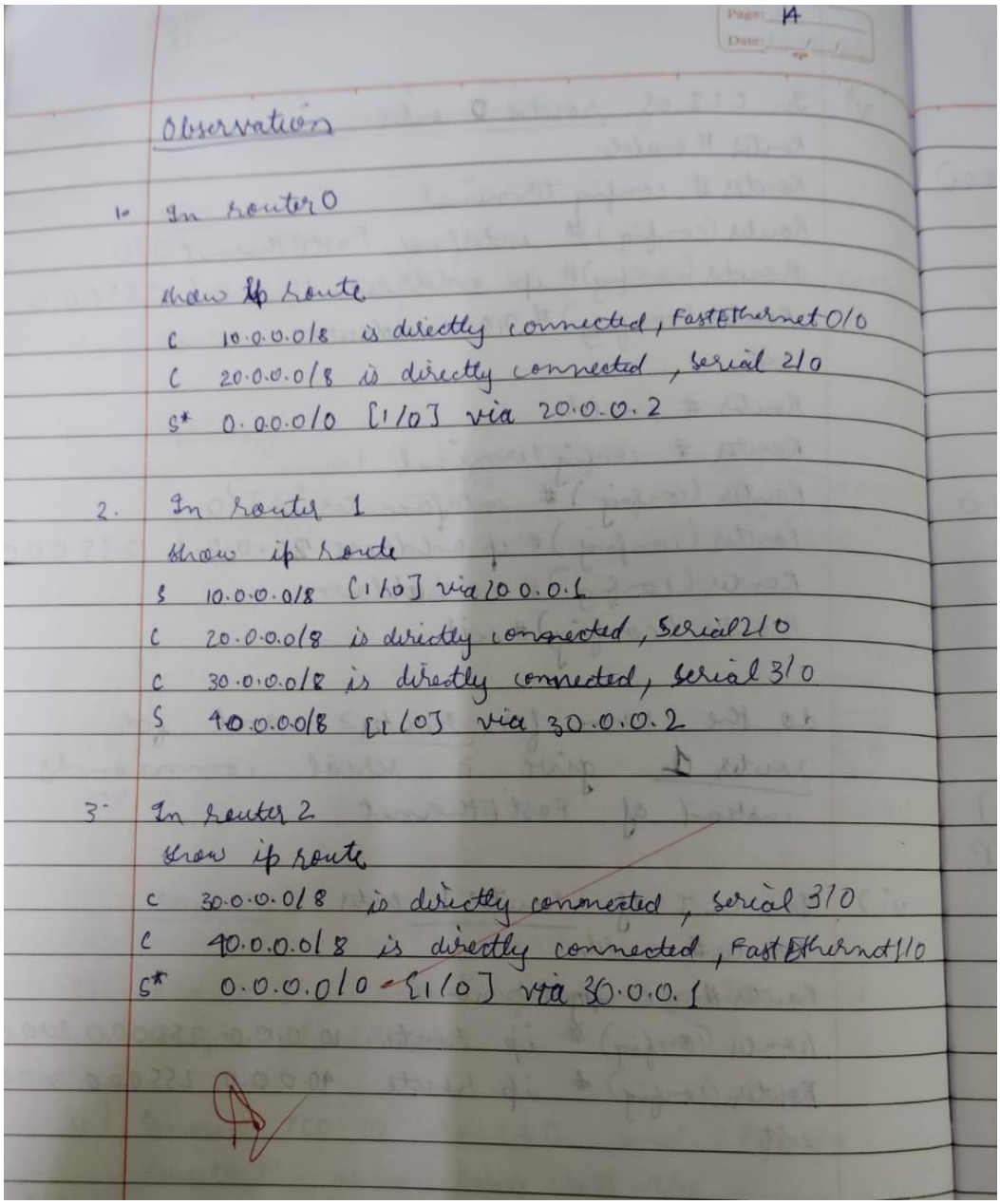
do the same for router 2 and for router 1 give 2 serial commands instead of FastEthernet.

vi) In CLI of router 1 enter
Router # enable
Router # config terminal
Router (config) # ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router (config) # ip route 40.0.0.0 255.0.0.0 30.0.0.2
exit

vii) In CLI of router 0 enter (default routing)
Router (config) # ip route 0.0.0.0 0.0.0.0 20.0.0.2

viii) In CLI of router 2 enter (default routing)
Router (config) # ip route 0.0.0.0 0.0.0.0 30.0.0.1

ix) ping PC0 to PC1, the packets are send.



Observations

1. In Router 0

show ip route

C 10.0.0.0/8 is directly connected, FastEthernet 0/0

C 20.0.0.0/8 is directly connected, Serial 2/0

S* 0.0.0.0/0 [1/0] via 20.0.0.2

2. In Router 1

show ip route

S 10.0.0.0/8 [1/0] via 20.0.0.1

C 20.0.0.0/8 is directly connected, Serial 2/0

C 30.0.0.0/8 is directly connected, Serial 3/0

S 40.0.0.0/8 [1/0] via 30.0.0.2

3. In Router 2

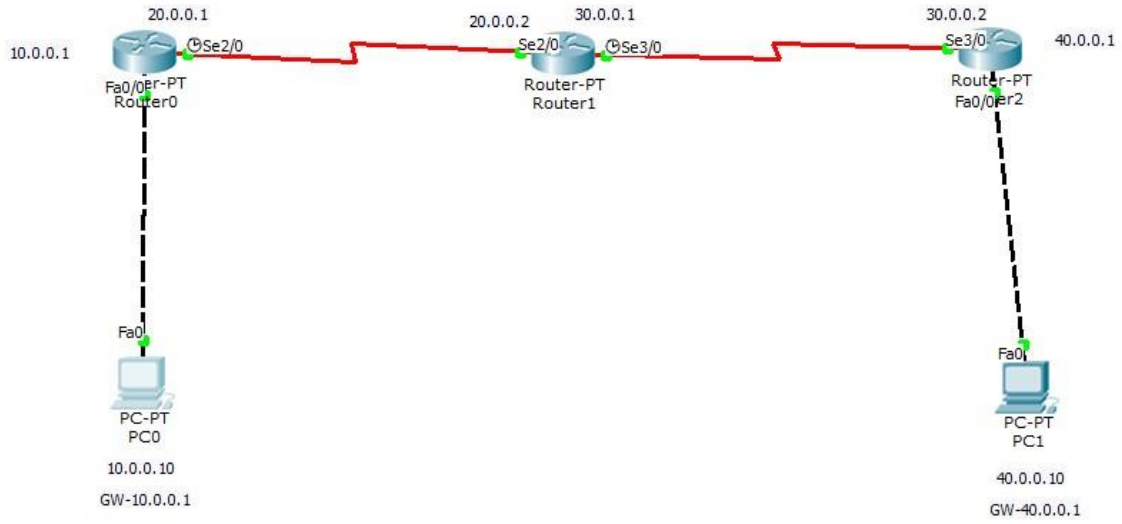
show ip route

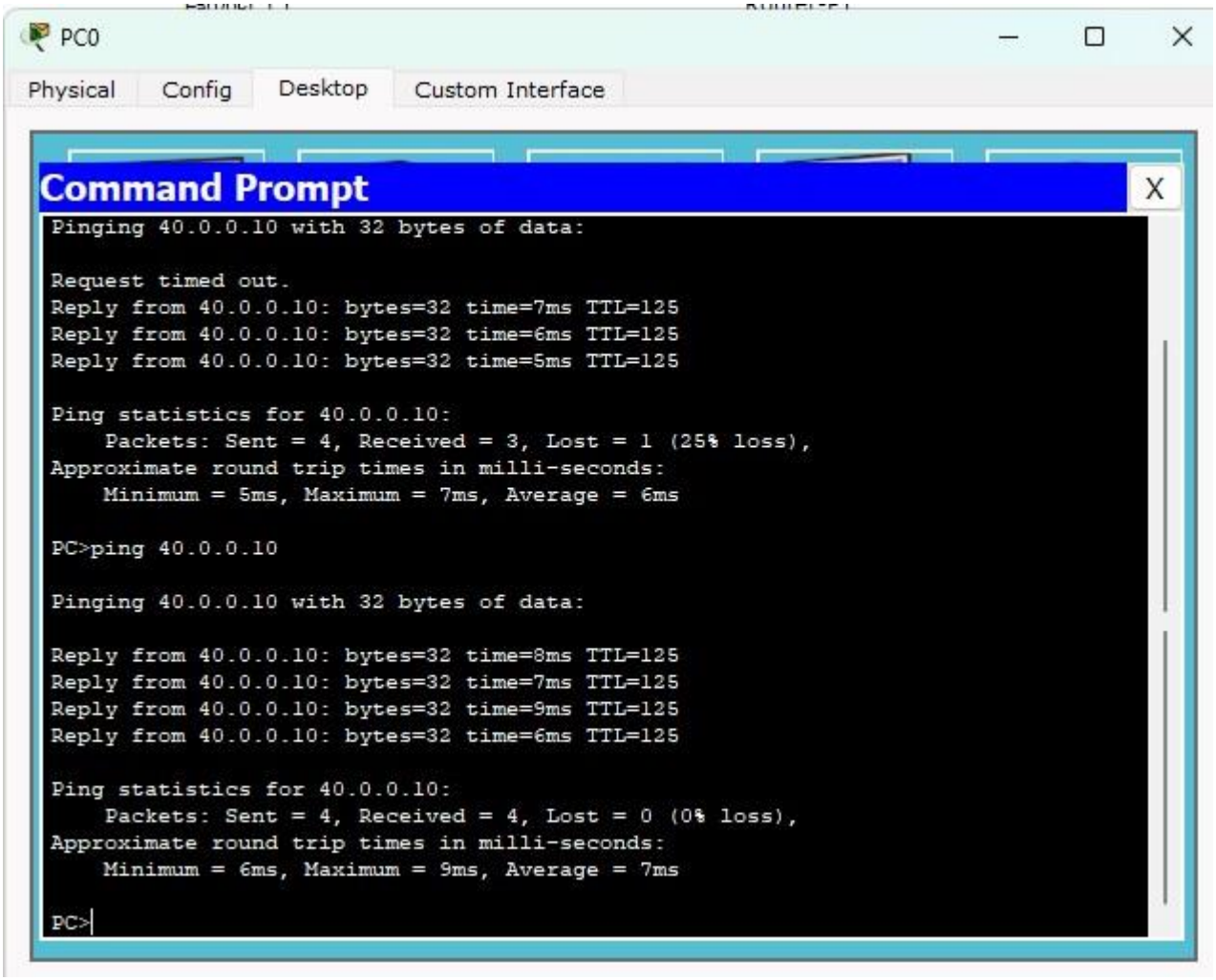
C 30.0.0.0/8 is directly connected, Serial 3/0

C 40.0.0.0/8 is directly connected, FastEthernet 1/0

S* 0.0.0.0/0 [1/0] via 30.0.0.1

Screen Shots:

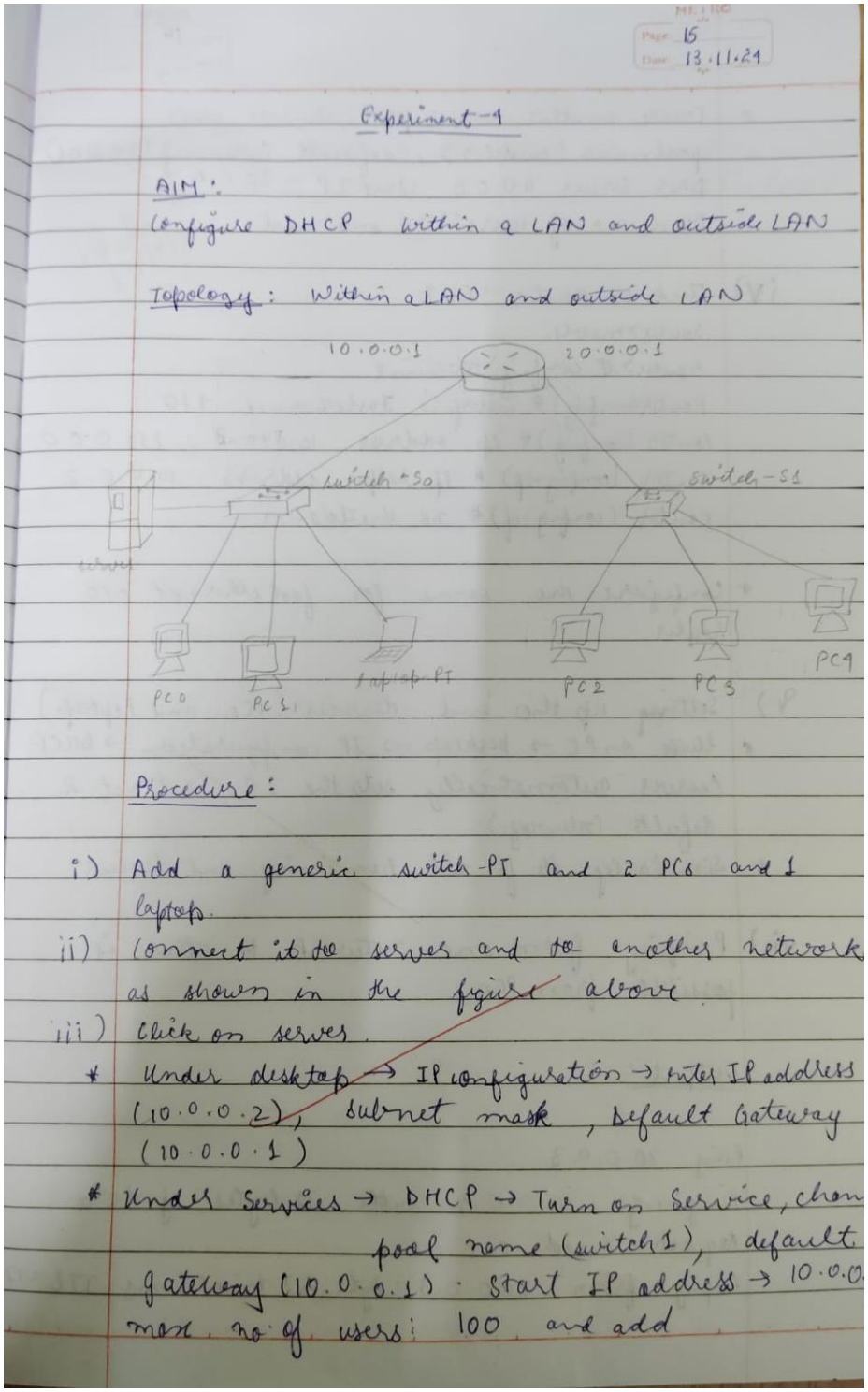




Program 5

Aim: Configure DHCP within a LAN and outside LAN.

Topology , Procedure and Observation:



Page 15
Date 13.11.24

Experiment-1

AIM:
Configure DHCP within a LAN and outside LAN

Topology: Within a LAN and outside LAN

Procedure:

i) Add a generic switch-PT and 2 PCs and 1 laptop.
ii) Connect it to servers and to another network as shown in the figure above.
iii) Click on servers.
* Under desktop → IP configuration → enter IP address (10.0.0.2), subnet mask, default Gateway (10.0.0.1)
* Under Services → DHCP → Turn on Service, chan pool name (switch1), default gateway (10.0.0.1). start IP address → 10.0.0 max. no of users: 100 and add

* Create another server pool
poolname (switch2), default Gateway (20.0.0.
DNS Server 0.0.0.0 start IP – 20.0.0.3
Max no of users: 100 and add.

iv) Click on Router → CLI
Router> enable
Router># config terminal
Router(config)# interface FastEthernet 1/0
Router (config)# ip address 10.0.0.3 255.0.0.0
Router (config-if)# ip helper address 10.0.0.2
Router (config-if)# no shutdown

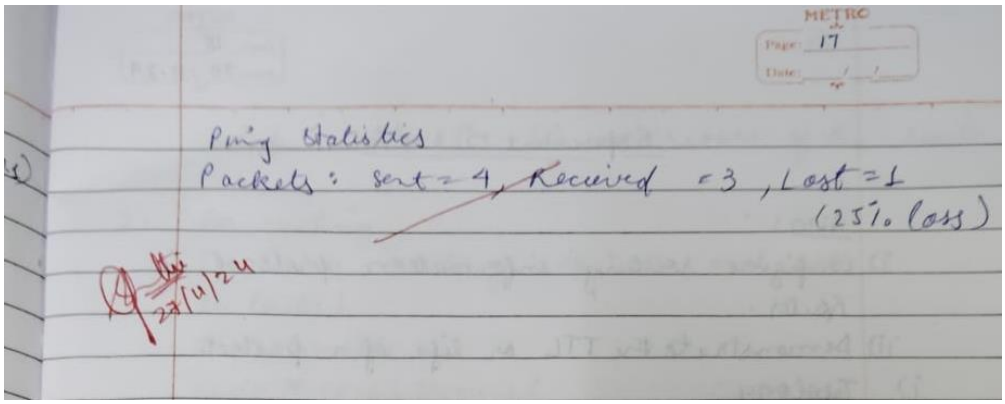
* Configure the same for fast ethernet 0/0 Router.

v) Setting up the end devices (PCs and laptop)
* Click on PC → Desktop → IP configuration → DHCP (server automatically sets the IP, subnet & default Gateway)
* Similarly do for all remaining end devices.

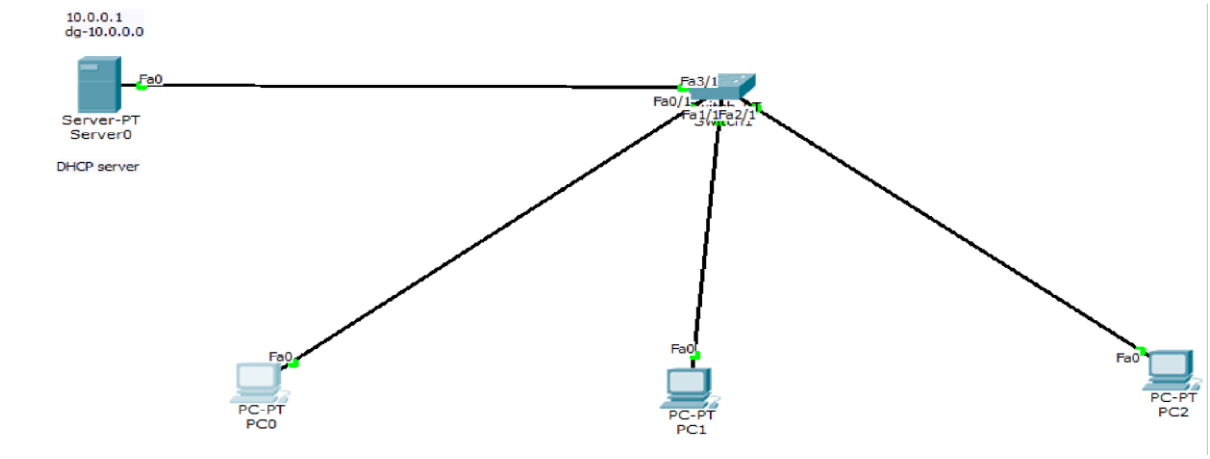
vi) Pinging from one network to other is possible from PC0.

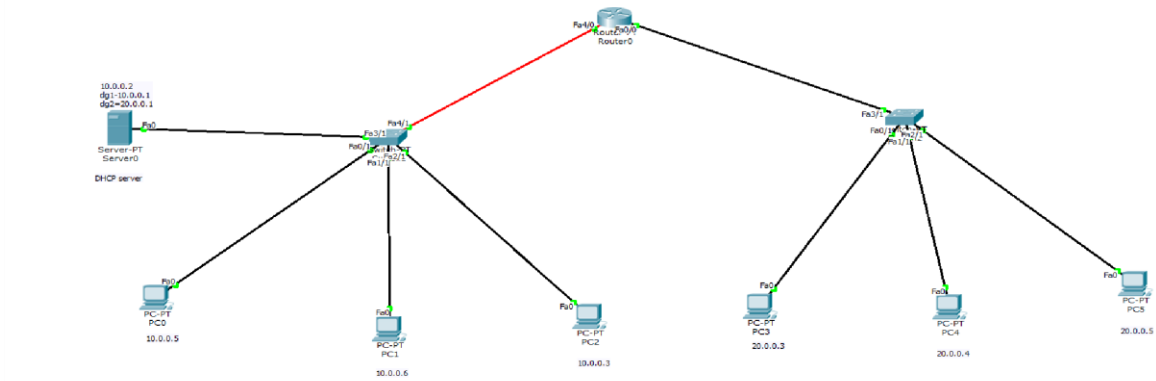
Output:

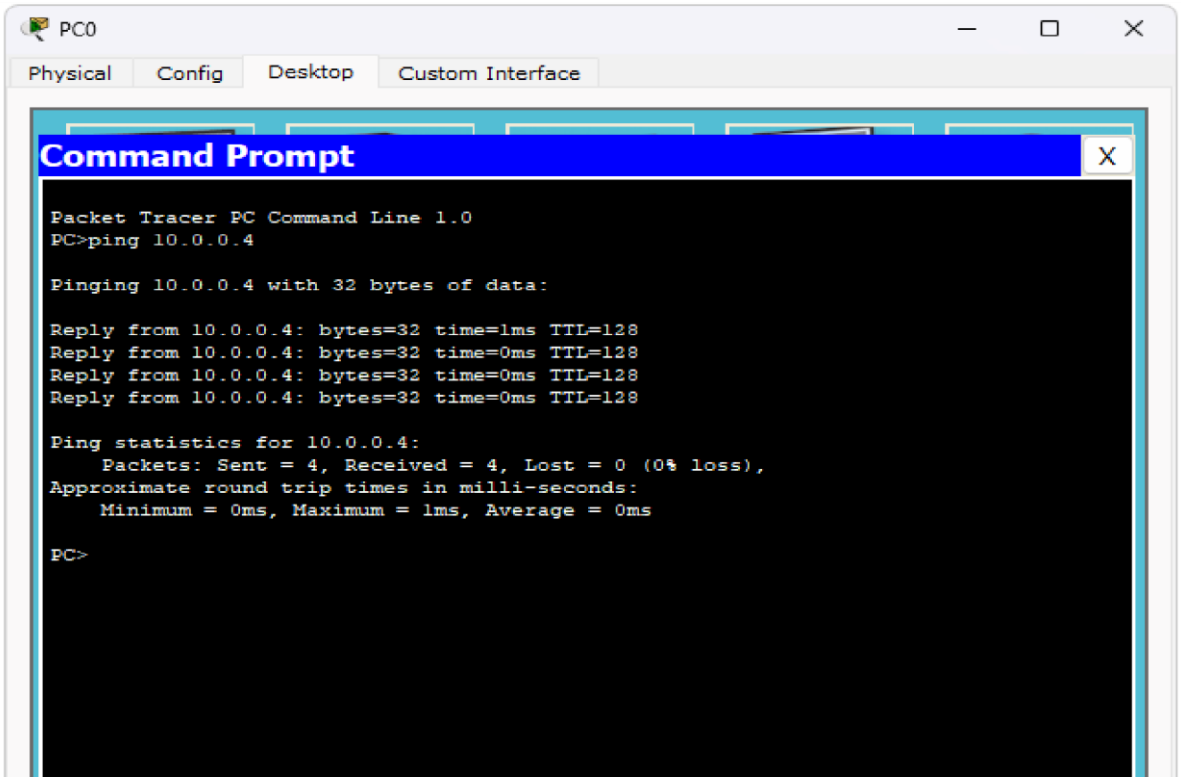
Ping 20.0.0.3
Pinging 20.0.0.3 with 32 bytes of data.
Request timed out
Reply from 200.0.3: bytes: 32 time: 1 ms TTL=1



Screen Shots:



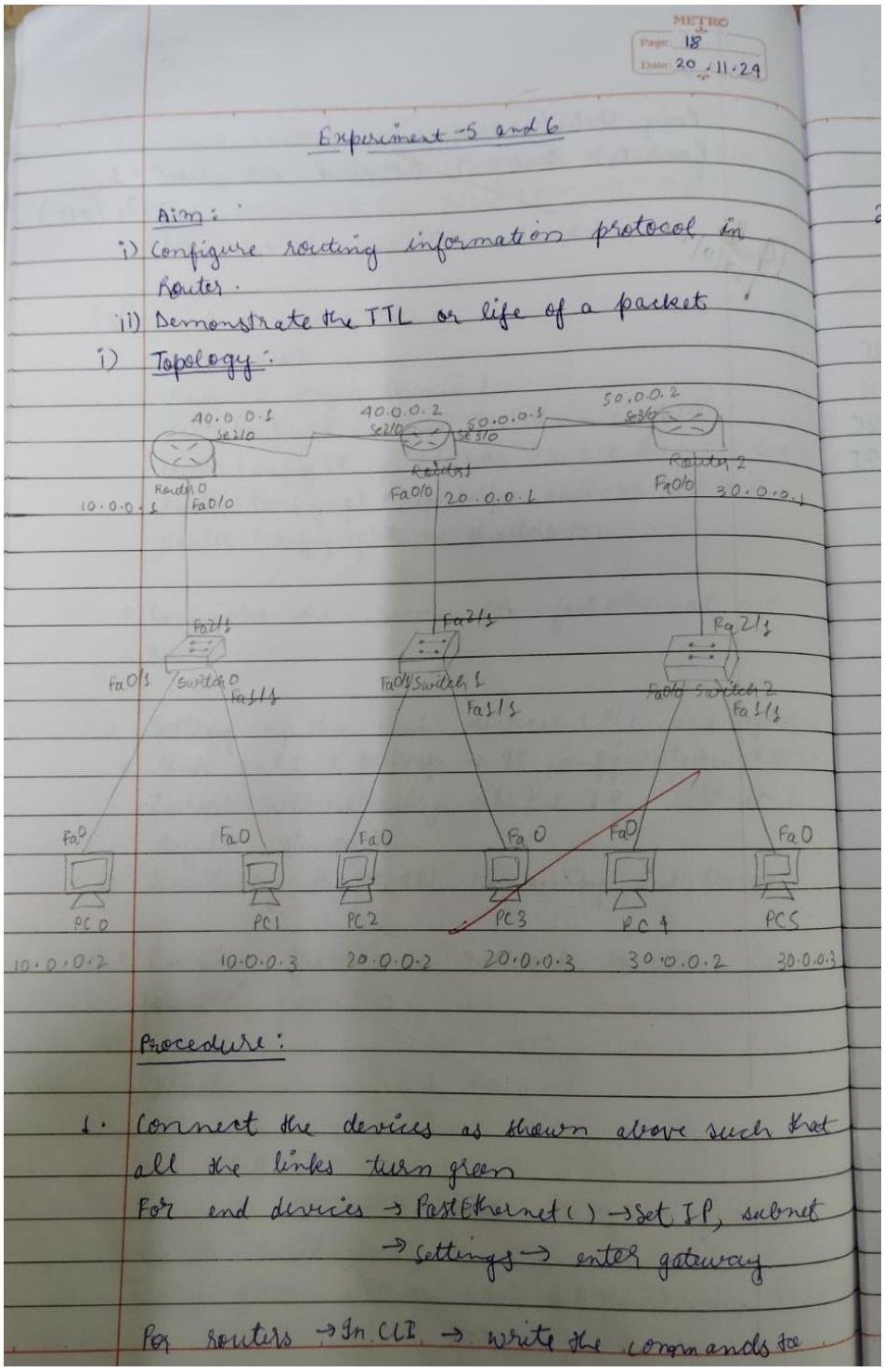



PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms 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>

Program 6

Aim: Configure RIP routing Protocol in Routers .

Topology , Procedure and Observation:



setup all the IP addresses of the routers.

2. For routing
For each routers go to CLI and enter
Ex: Router 1
router > enable
router # config terminal
router (config) # router rip
Router (config-router) # network 40.0.0.0
router (config-router) # network 50.0.0.0
router (config-router) # network 20.0.0.0

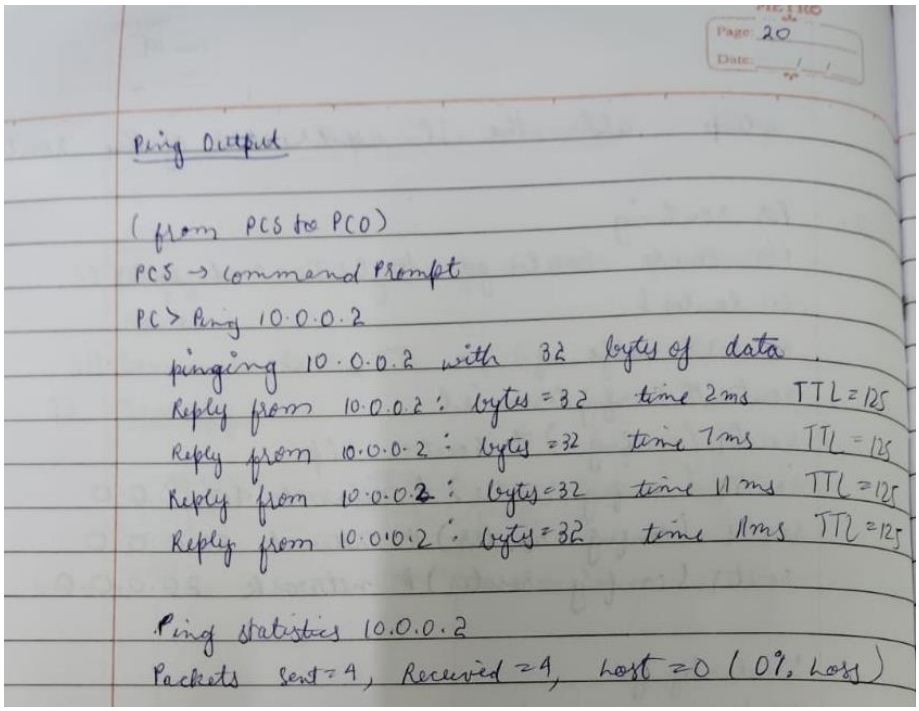
Similarly for
Router 0 → connect to network 10.0.0.0 and 40.0.0.0
Router 2 → connect to network 50.0.0.0 and 30.0.0.0

3. Once this setup is complete, we can the message from one device to any other end device.

Output:

In Router 0
router # show ip route
C 10.0.0.0/8 is directly connected, fastEthernet 0/0
R 20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:15, Serial 2/0
R 30.0.0.0/8 [120/1] via 40.0.0.2, 00:00:15, Serial 2/0
C 40.0.0.0/8 is directly connected, Serial 2/0
R 50.0.0.0/8 [120/1] via 10.0.0.2, 00:00:15, Serial 2/0

Similarly the output is shown for router 1 and router 2.



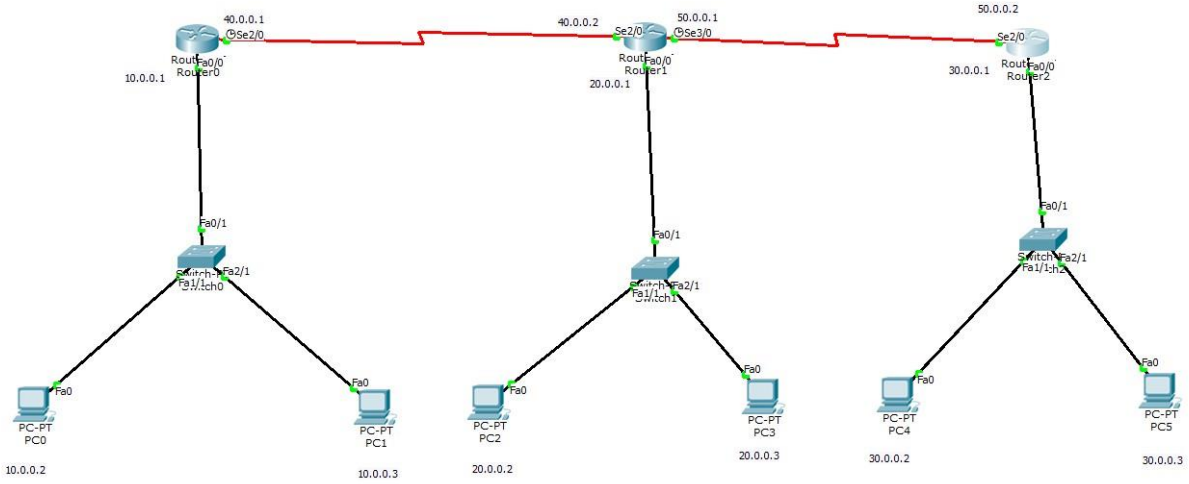
Page: 20
Date: / /

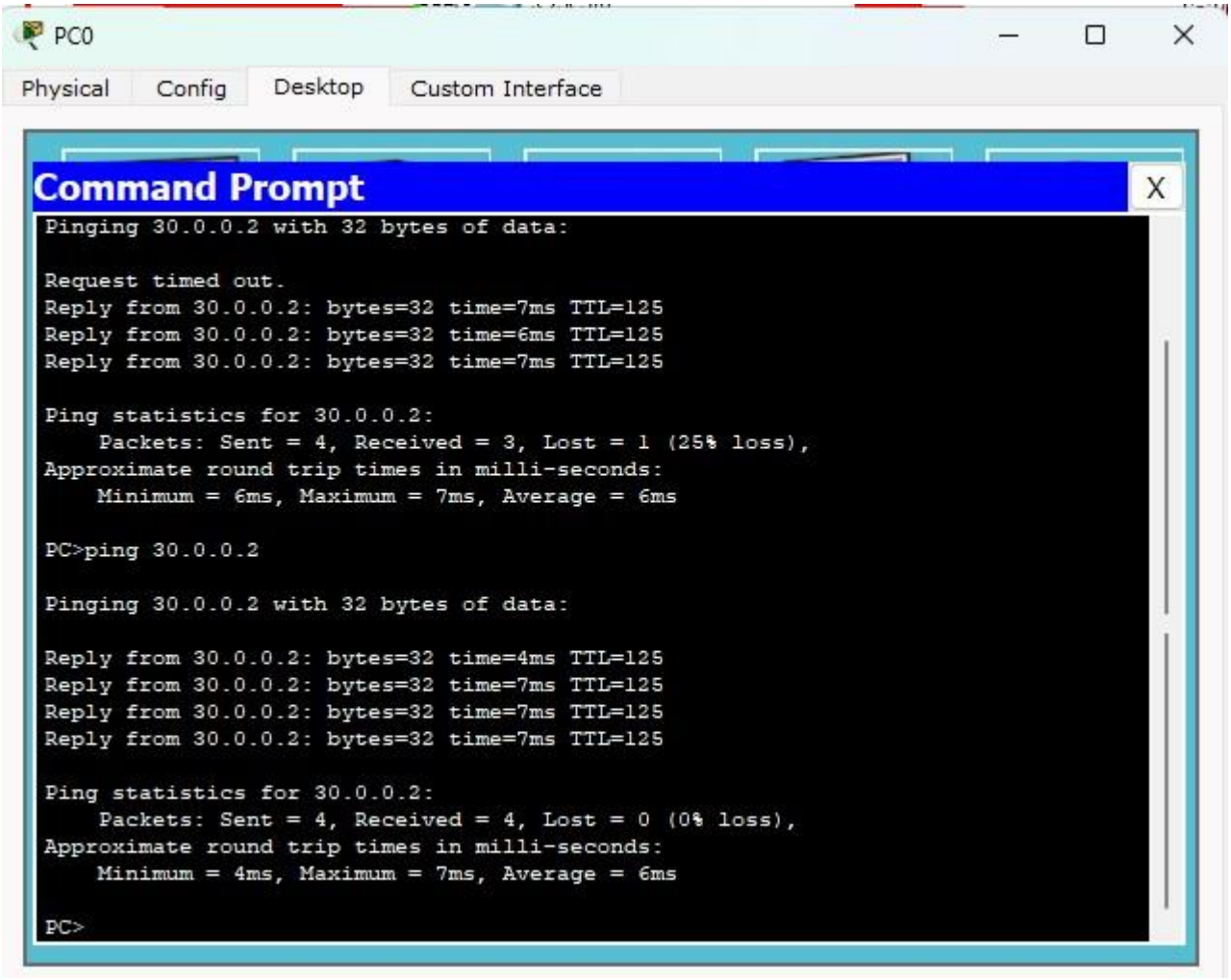
Ping Output

(from PC5 to PC0)
PC5 → command Prompt
PC> Ping 10.0.0.2
pinging 10.0.0.2 with 32 bytes of data.
Reply from 10.0.0.2: bytes=32 time 2ms TTL=125
Reply from 10.0.0.2: bytes=32 time 7ms TTL=125
Reply from 10.0.0.2: bytes=32 time 11ms TTL=125
Reply from 10.0.0.2: bytes=32 time 11ms TTL=125

Ping statistics 10.0.0.2
Packets sent=4, Received=4, Lost=0 (0% Loss)

Screen Shots:

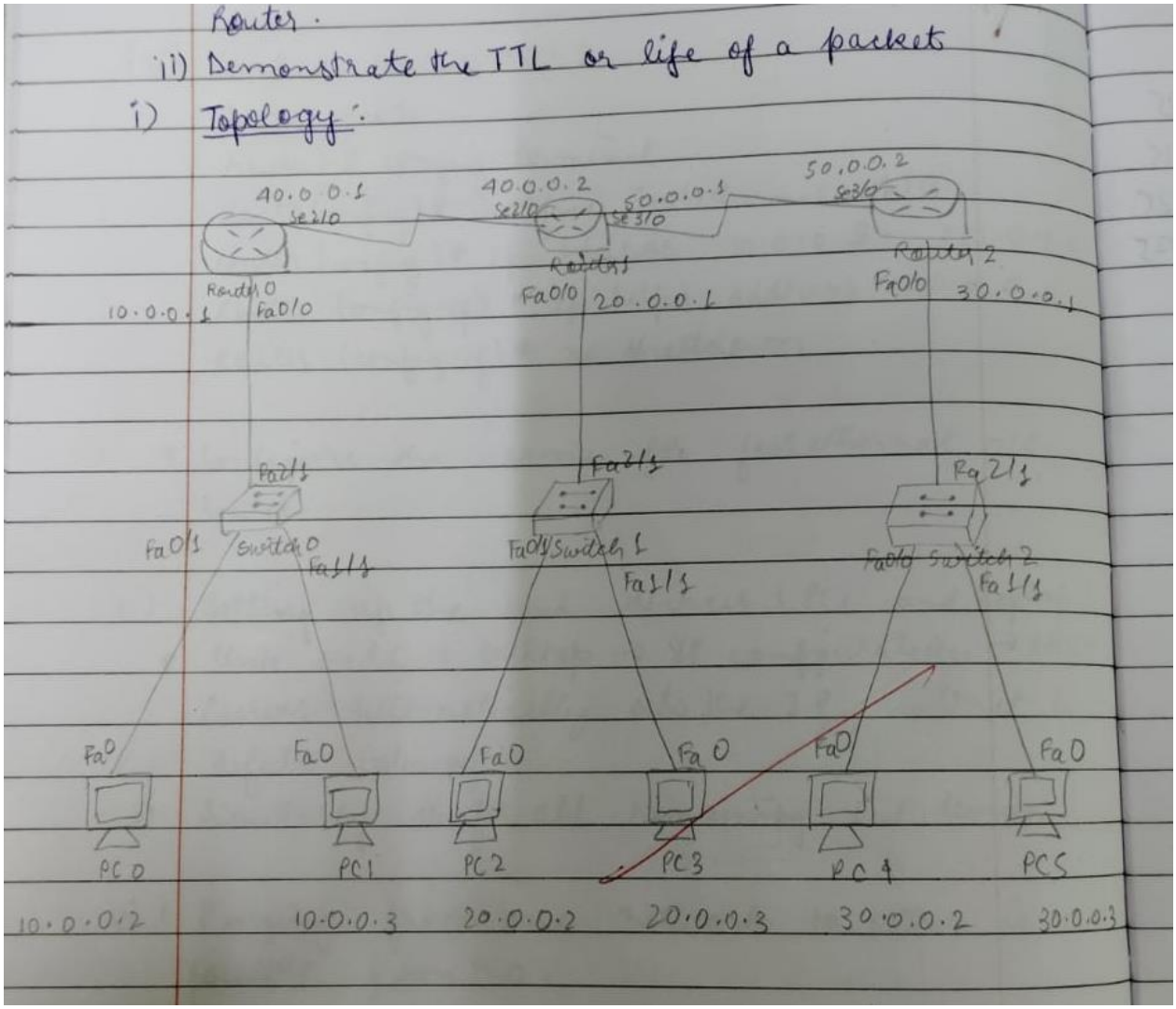




Program 7

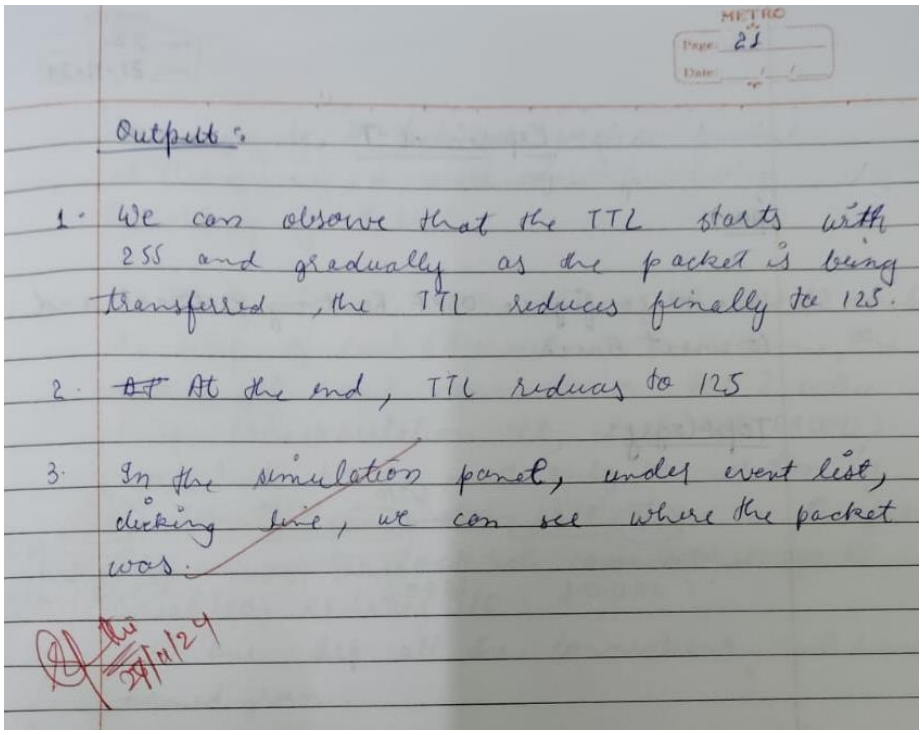
Aim: Demonstrate the TTL/ Life of a Packet .

Topology , Procedure and Observation:



Procedure:

1. Click on simulation mode
2. Click on Packet option on the source to destination end device.
3. Enter Auto capture/Play
4. We can see the packet movement from the source to destination and then the acknowledgement sent back to the source
5. At each point, we can click on the packet and view the OSI model, Inbound PDV details, outbound PDV details.

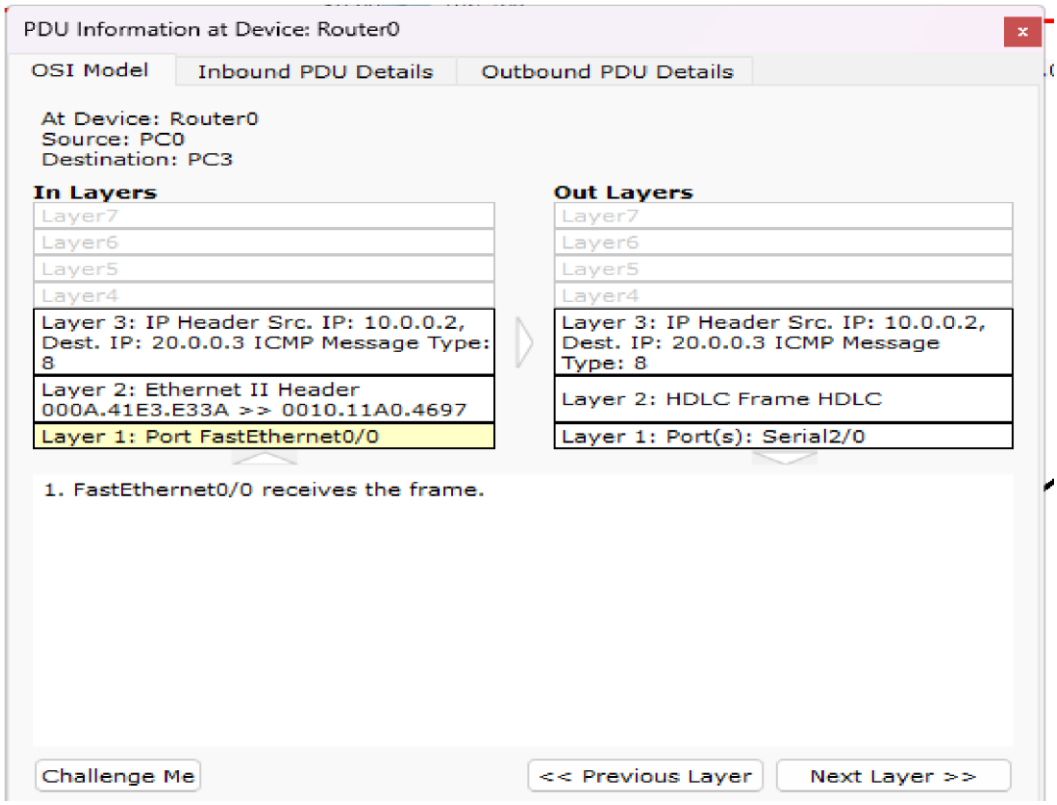


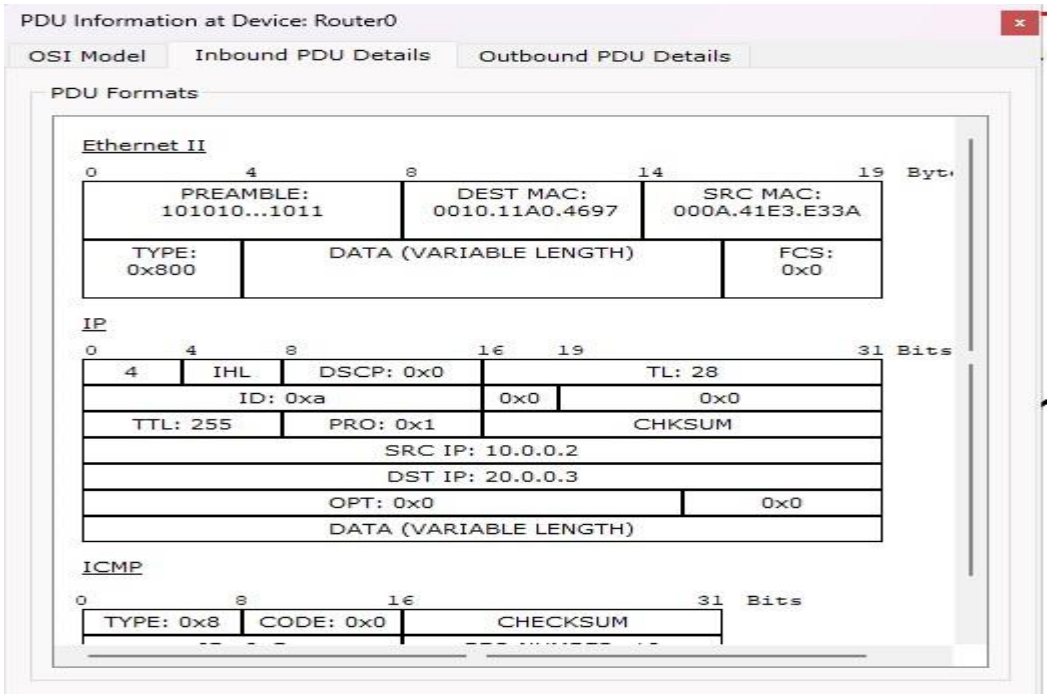
Output:

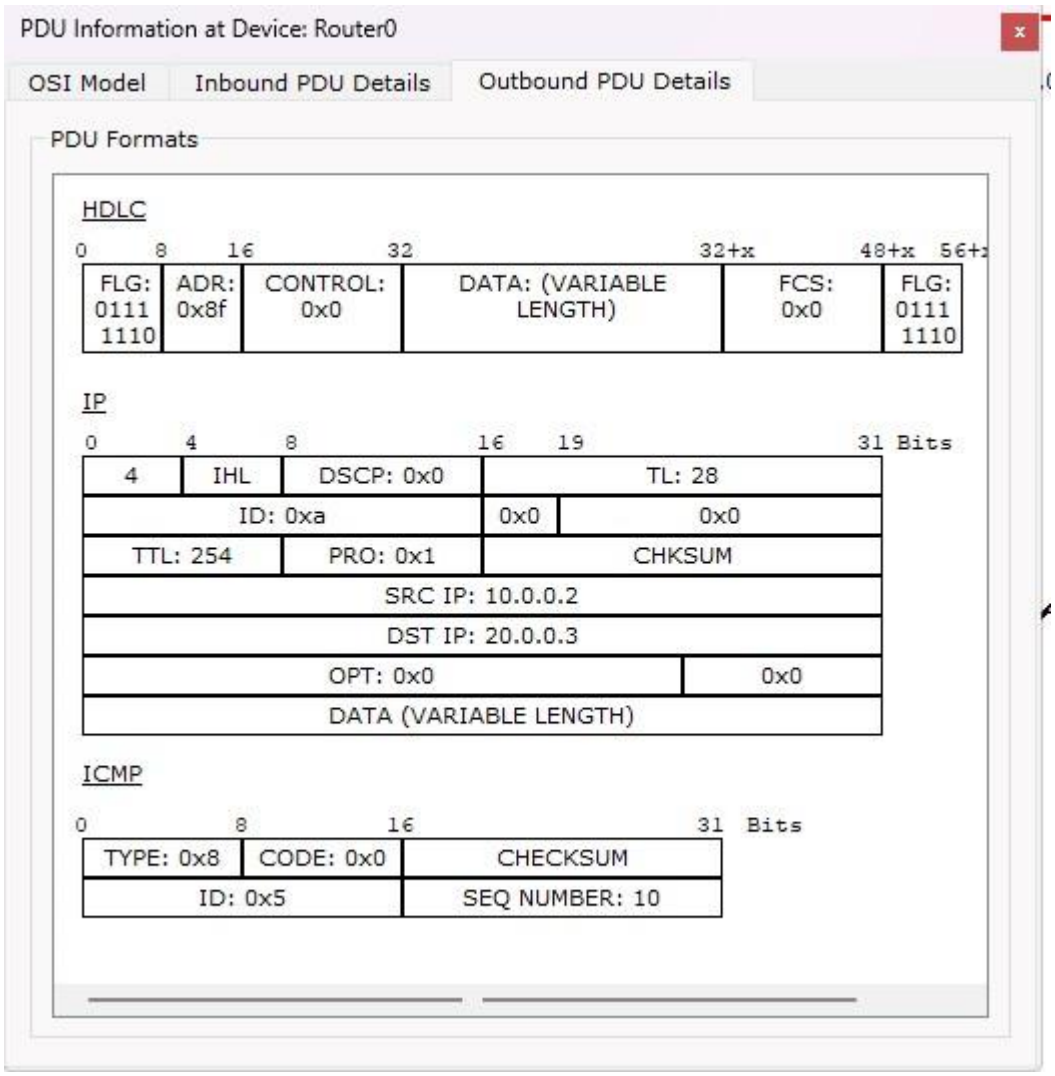
1. We can observe that the TTL starts with 255 and gradually as the packet is being transferred, the TTL reduces finally to 125.
2. At the end, TTL reduces to 125
3. In the simulation panel, under event list, clicking one, we can see where the packet was.

29/11/24

Screen Shots:



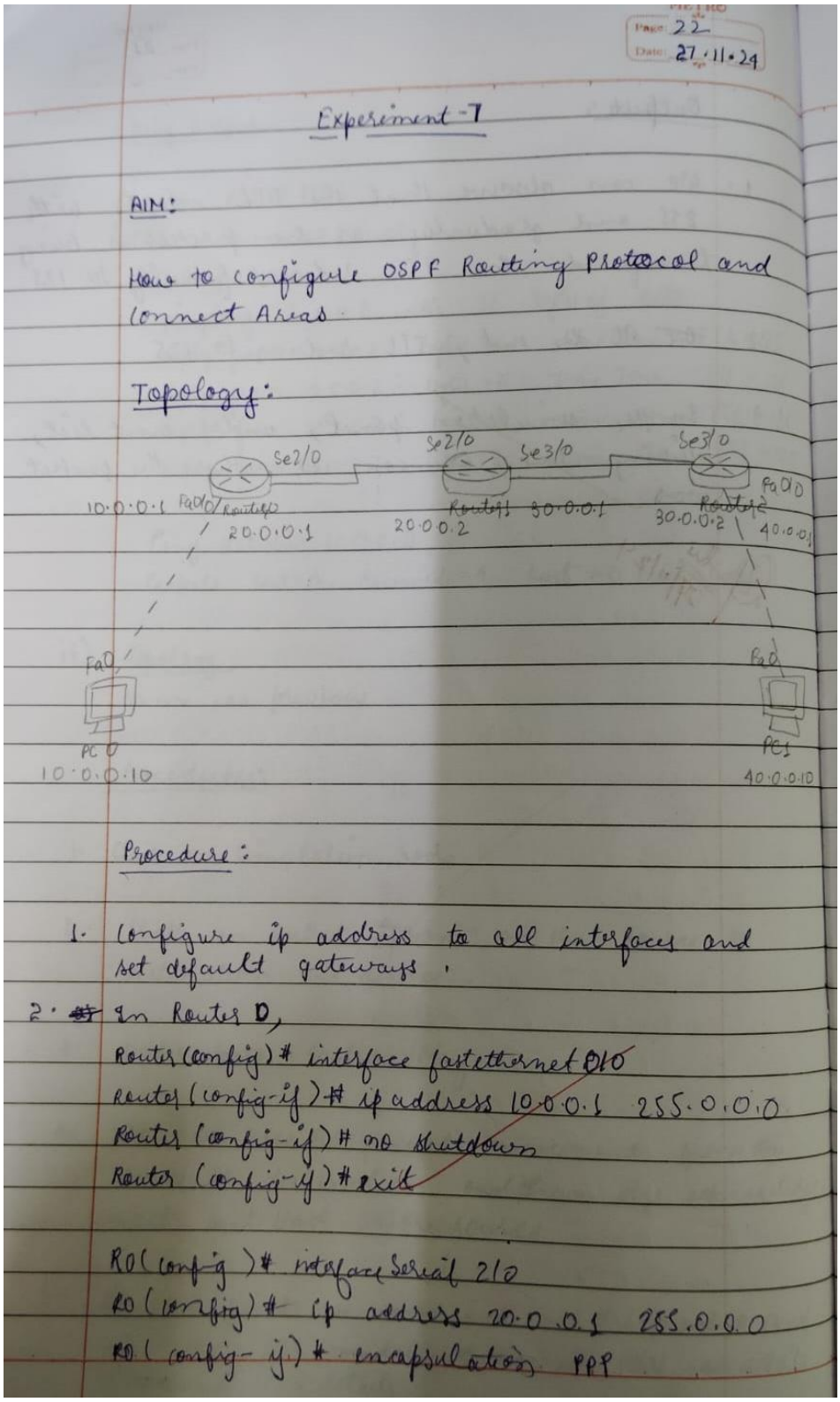



PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
HDLC
0 8 16 32 32+x 48+x 56+x
FLG: 0111 1110
ADR: 0x8f
CONTROL: 0x0
DATA: (VARIABLE LENGTH)
FCS: 0x0
FLG: 0111 1110
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
0x0
TTL: 254
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 10

Program 8

Aim: Configure OSPF routing protocol .

Topology , Procedure and Observation:



Page: 22
Date: 27.11.24

Experiment-7

AIM:

How to configure OSPF Routing protocol and connect Areas

Topology:

Procedure:

1. Configure ip address to all interfaces and set default gateways.
2. In Router 0,

Router(config)# interface fastethernet 0/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit

R0(config)# interface Serial 2/0
R0(config)# ip address 20.0.0.1 255.0.0.0
R0(config-if)# encapsulation PPP

R0 (config-if) # clock rate 64000
R0 (config-if) # no shutdown
R0 (config-if) # exit.

Similarly we set up the IPs of R1 and R2 while the setup of fast Ethernet remains same, the setting up of serial connections has 2 extra lines (encapsulation PPP, clock rate 64000). Clock rate 64000 must only be written if the serially connected port have a DCE symbol. Here, we write clock rate command for R0 Serial 2/0, R1 Serial 3/0.
After this step all the connections must have turned green.

3. To enable IP routing by configuring OSPF routing protocol in all routers.

Router R0 → CLI
R0 (config) # Router OSPF 1
R0 (config-router) # Router-id 1.1.1.1
R0 (config-router) # Network 10.0.0.0 0.255.255.255 area 3

R0 (config-router) # network 20.0.0.0 0.255.255.255 area 1
R0 (config-router) # exit.

Router R1 → CLI
R1 (config) # router ospf 1
R1 (config-router) # router-id 2.2.2.2
R1 (config-router) # network 200.0.0 0.255.255.255 area 1
R1 (config-router) # network 30.0.0.0 0.255.255.255 area 0
R1 (config-router) # exit.

Router R2 → CLI
R2(config)# router ospf 1
R2(config-router)# router-id 3.3.3.3
R2(config-router)# network 30.0.0.0 0.255.255.255 area 0
R2(config-router)# network 40.0.0.0 0.255.255.255 area 2
R2(config-router)# exit

1. Once the setting up of networking area is done, we configure loopback address to routers.

R0(config-if)# interface loopback 0
R0(config-if)# ip add 172.16.1.252 255.255.0.0
R0(config-if)# no shutdown

R1(config-if)# interface loopback 0
R1(config-if)# ip add 172.16.1.253 255.255.0.0
R1(config-if)# no shutdown

R2(config-if)# interface loopback 0
R2(config-if)# ip add 172.16.1.254 255.255.0.0
R2(config-if)# no shutdown

5. On checking routing table of R3 using show ip route we can see that R3 doesn't know about area 3
Gateway of last resort is not set

O IA 20.0.0.0/8 [110/128] via 30.0.0.1, Serial 1/0
C 40.0.0.0/8 is directly connected, fast Ethernet 0/0
C 30.0.0.0/8 is directly connected, Serial 2/0

Since R3 doesn't know about area 3 we have

to create a virtual link between R0 and R1.

6. Creating virtual link between R1, R0

On Router R0

R0(config)# router ospf 1

R0(config-router)# area 1 virtual-link 2.2.2.2

R0(config-router)# exit

On Router R1

R1(config)# router ospf 1

R1(config-router)# area 1 virtual-link 1.1.1.1

R1(config-router)# exit.

7. Now, check routing table of R3.

Once all these steps are completed, the message can now be pinged from 1 end-device to other.

Observation

In R2

Router # show ip route

O IA 20.0.0.0/8 [110/128] via 30.0.0.1, 00:57:25 Serial2/0

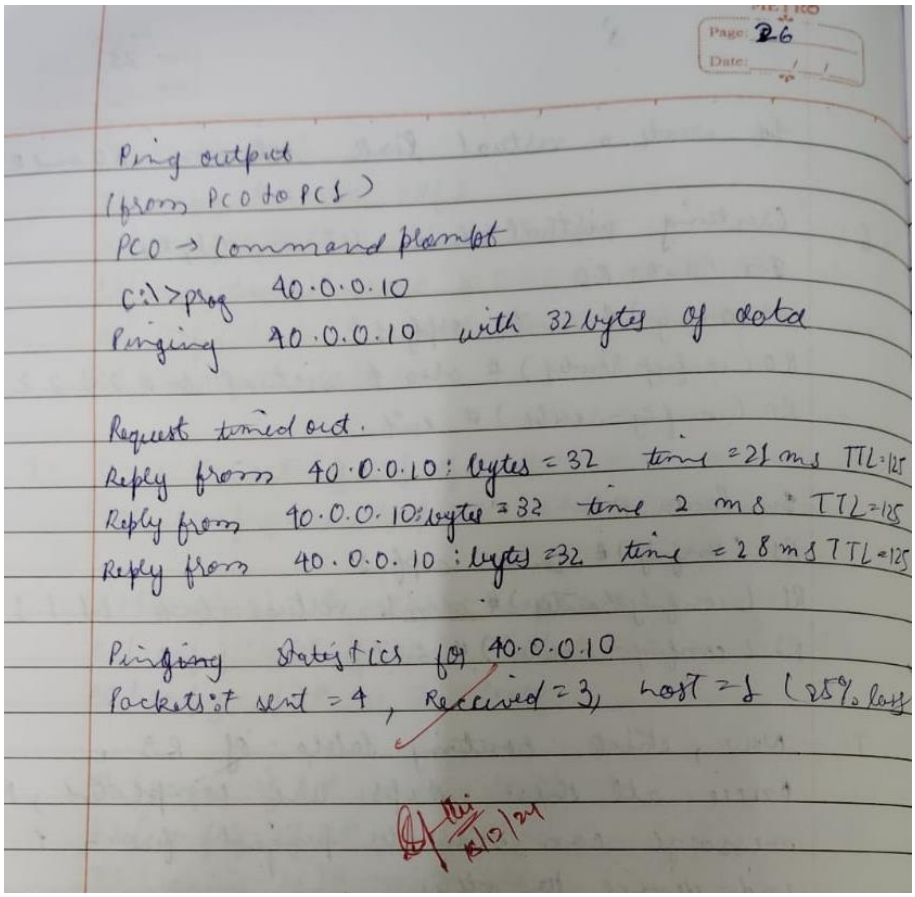
C 40.0.0.0/8 is directly connected, FastEthernet 0/0

O IA 10.0.0.0/8 [110/129] via 30.0.0.1 00:57:25, Serial2/0

C 30.0.0.0/8 is directly connected, Serial 2/0

C 172.16.0.0/16 is directly connected, Loopback.

Similarly the output is shown for Router 0 and 1


Page: 26
Date:

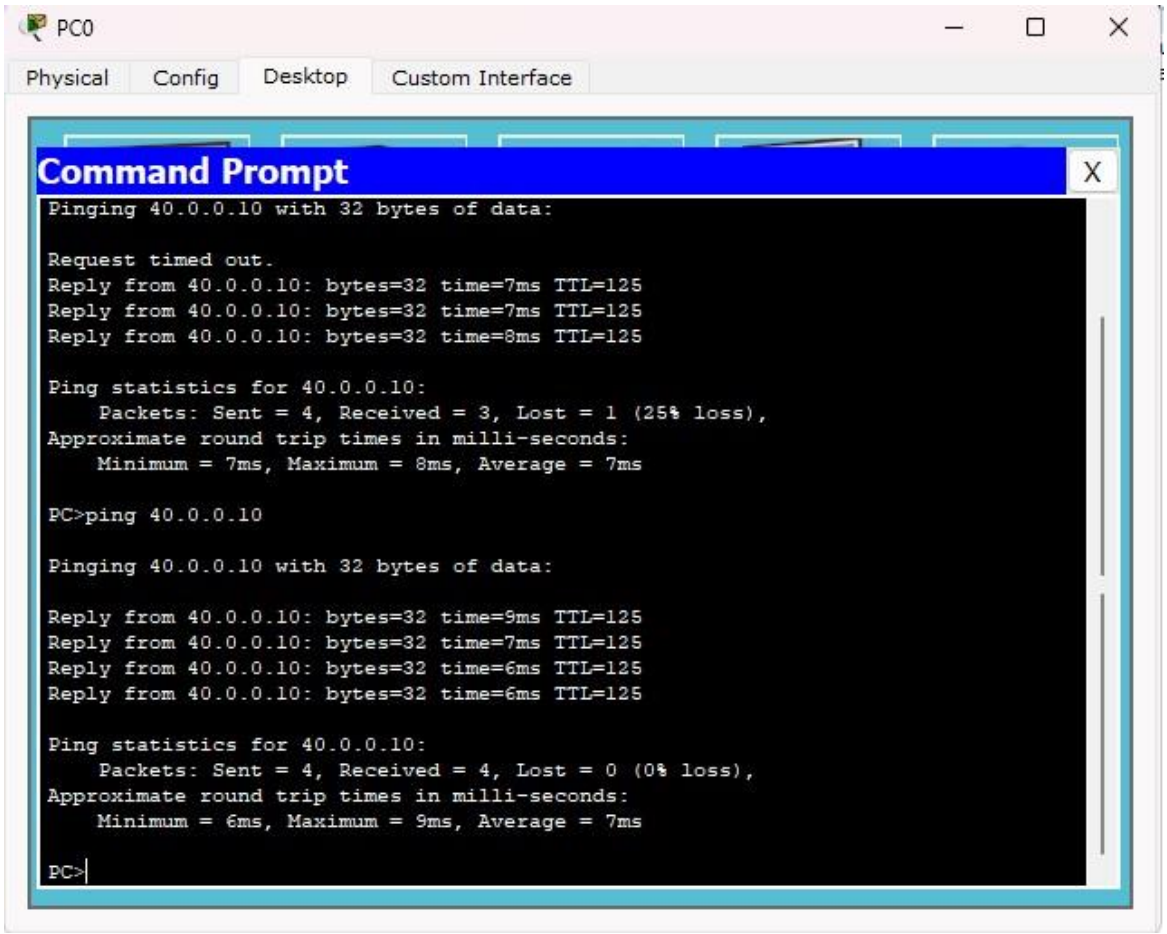
Ping output
(from PC0 to PC5)
PC0 → Command prompt
C:\>ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data

Request timed out.
Reply from 40.0.0.10: bytes = 32 time = 21 ms TTL = 125
Reply from 40.0.0.10: bytes = 32 time 2 ms TTL = 125
Reply from 40.0.0.10 : bytes = 32 time = 28 ms TTL = 125

Pinging Statistics for 40.0.0.10
Packets: sent = 4, Received = 3, Lost = 1 (25% loss)

8/10/24

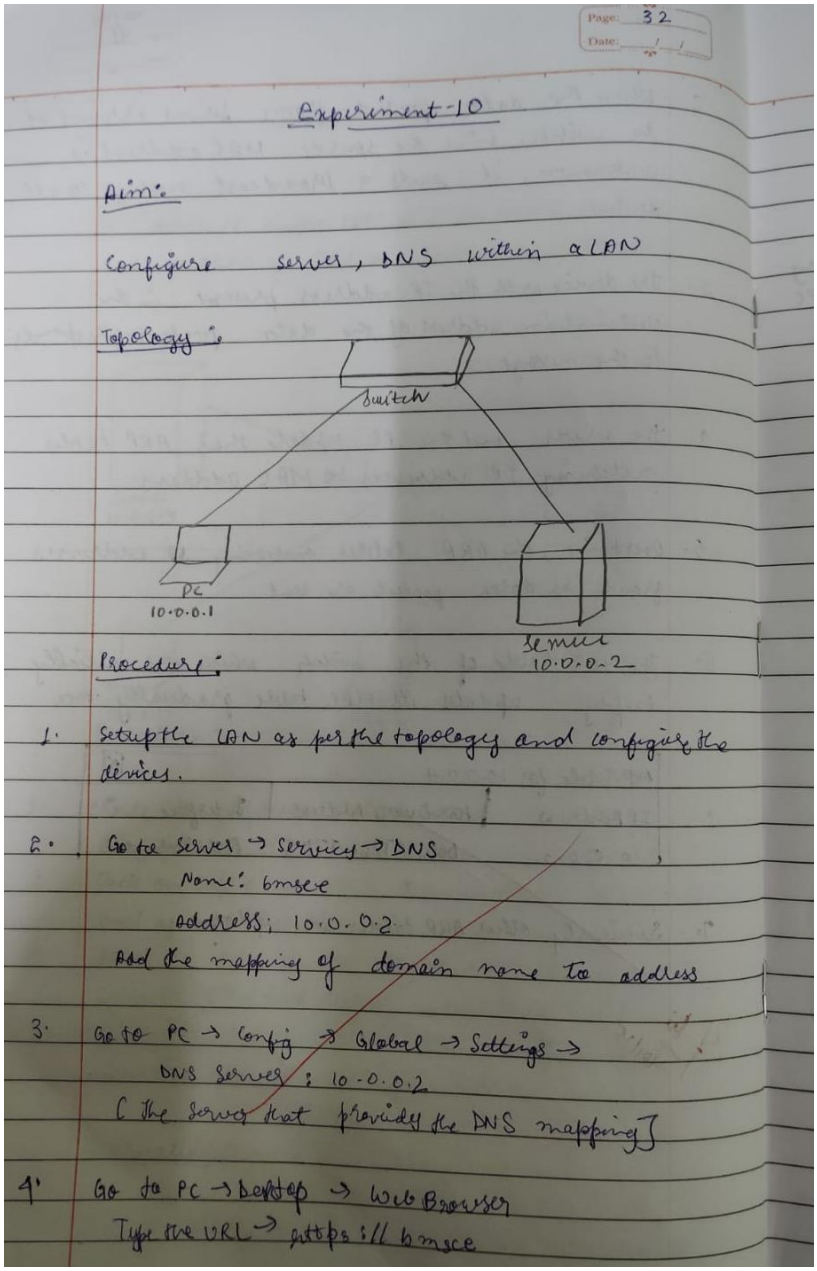
Screen Shots:



Program 9

Aim: Configure Web Server, DNS within a LAN.

Topology , Procedure and Observation:



Page: 32
Date: / /

Experiment-10

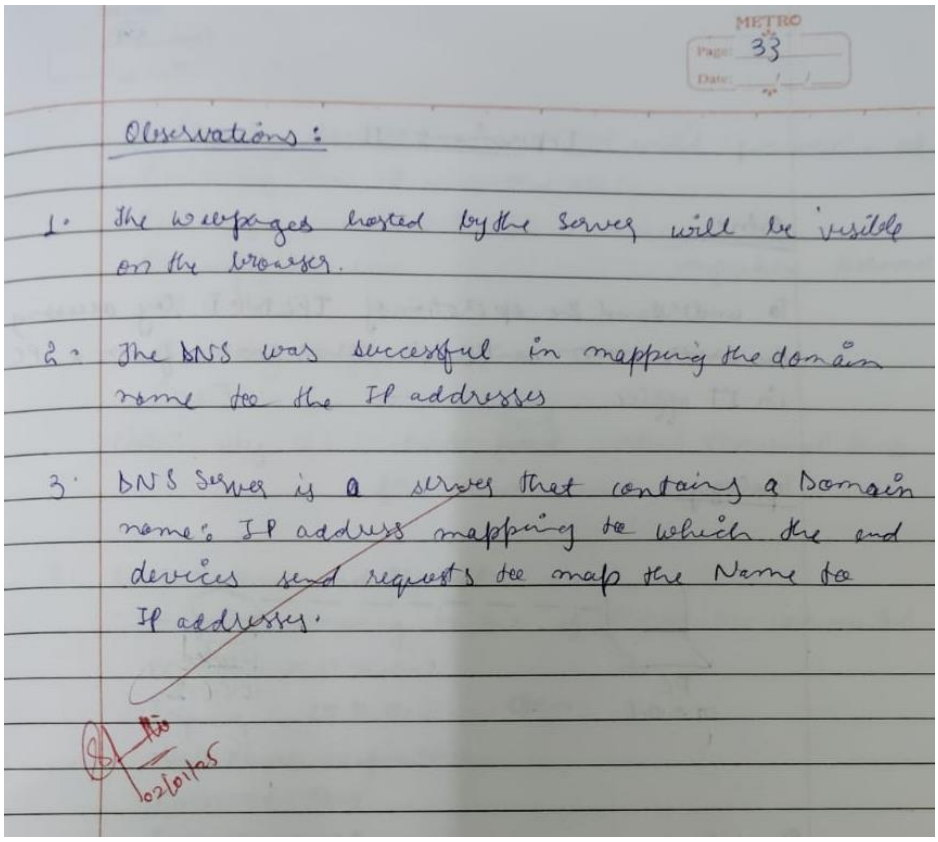
Aim:

Configure server, DNS within a LAN

Topology:

Procedure:

1. Setup the LAN as per the topology and configure the devices.
2. Go to Server → Service → DNS
   Name: bmsce
   Address: 10.0.0.2
   Add the mapping of domain name to address
3. Go to PC → Config → Global → Settings →
   DNS Server : 10.0.0.2
   (The Server that provides the DNS mapping)
4. Go to PC → Desktop → Web Browser
   Type the URL → https://bmsce



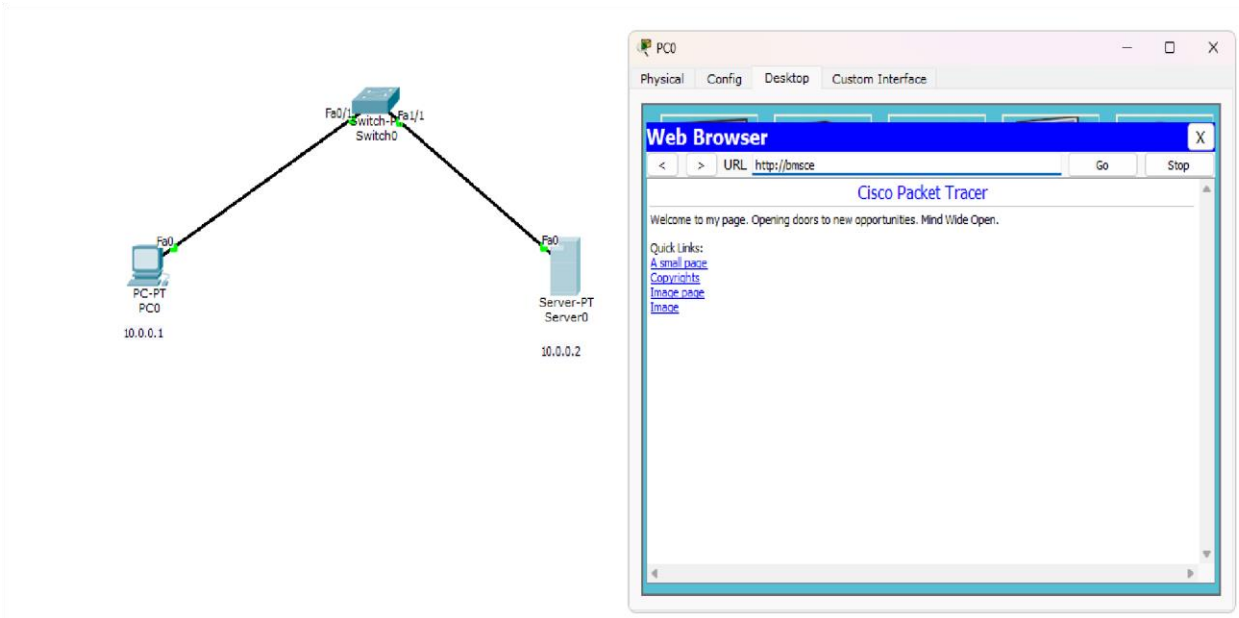
Observations :

1. The webpages hosted by the servers will be visible on the browser.

2. The DNS was successful in mapping the domain name for the IP addresses

3. DNS Server is a server that contains a Domain name & IP address mapping for which the end devices send requests for map the Name for IP addresses.

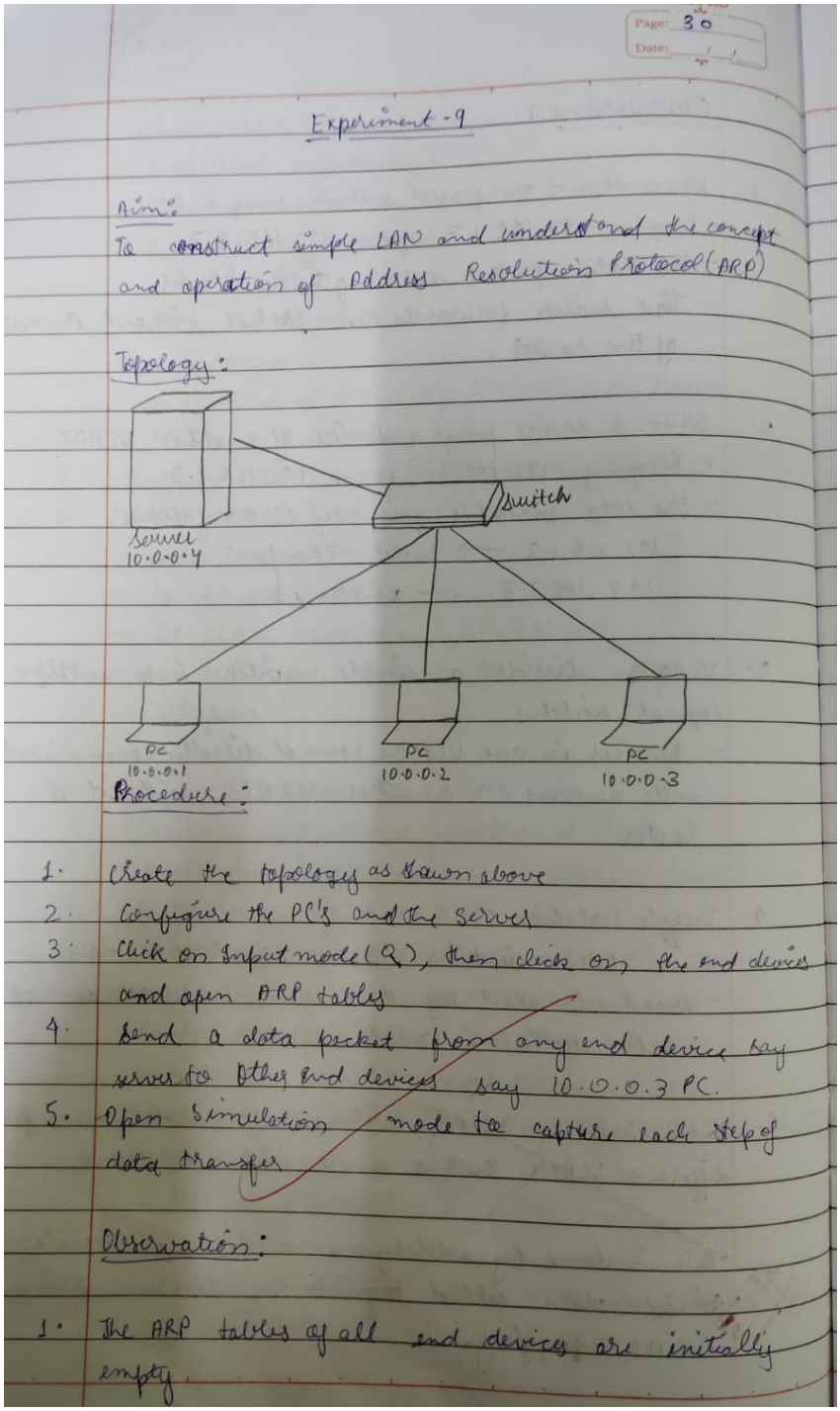
Screen Shots:



Program 10

Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

Topology , Procedure and Observation:



Page: 30
Date: / /

Experiment - 9

Aim:
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

Topology:

Procedure:

1. Create the topology as shown above
2. Configure the PC's and the server
3. Click on Inspect mode (Q), then click on the end devices and open ARP tables
4. Send a data packet from any end device say server to other end devices say 10.0.0.3 PC.
5. Open simulation mode to capture each step of data transfer

Observation:

1. The ARP tables of all end devices are initially empty.

2. When the data packet from server arriving at the switch, since the source MAC address is unknown, it sends a broadcast message to all devices

3. The device with the IP address present in the destination address of the data packet responds to the message.

4. The server and the PC update their ARP tables matching IP addresses to MAC address

5. Overtime, the ARP tables ~~matching~~ IP addresses grows as data packets are sent.

6. The MAC table of the switch which was initially empty updates its MAC table gradually once

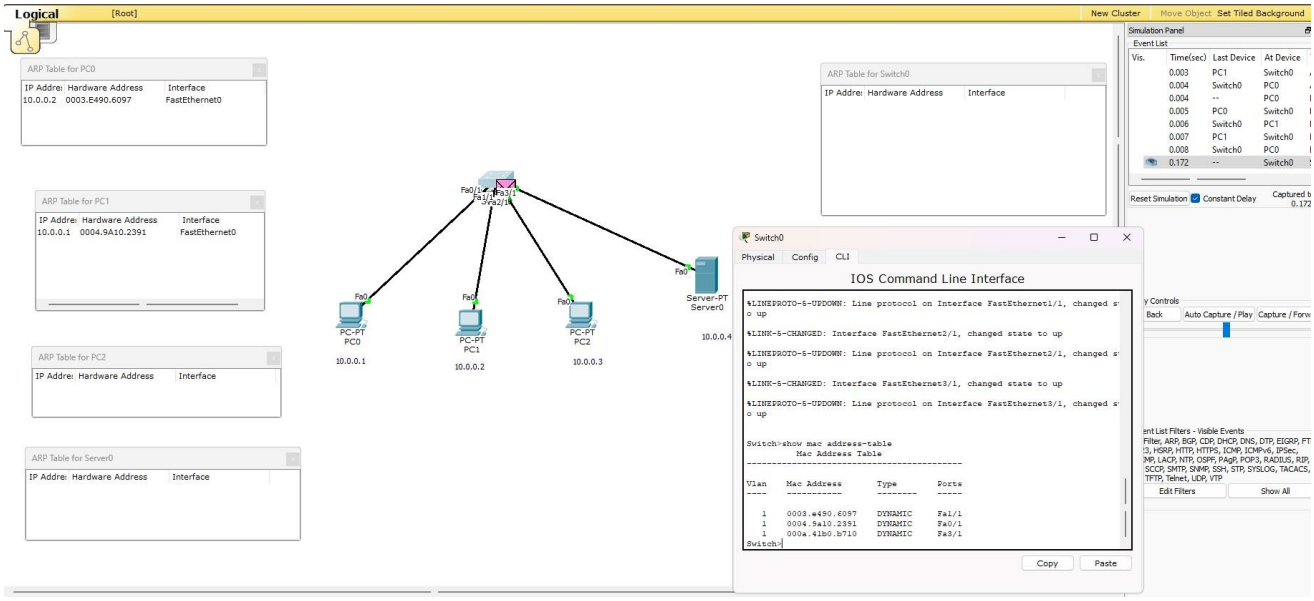
A

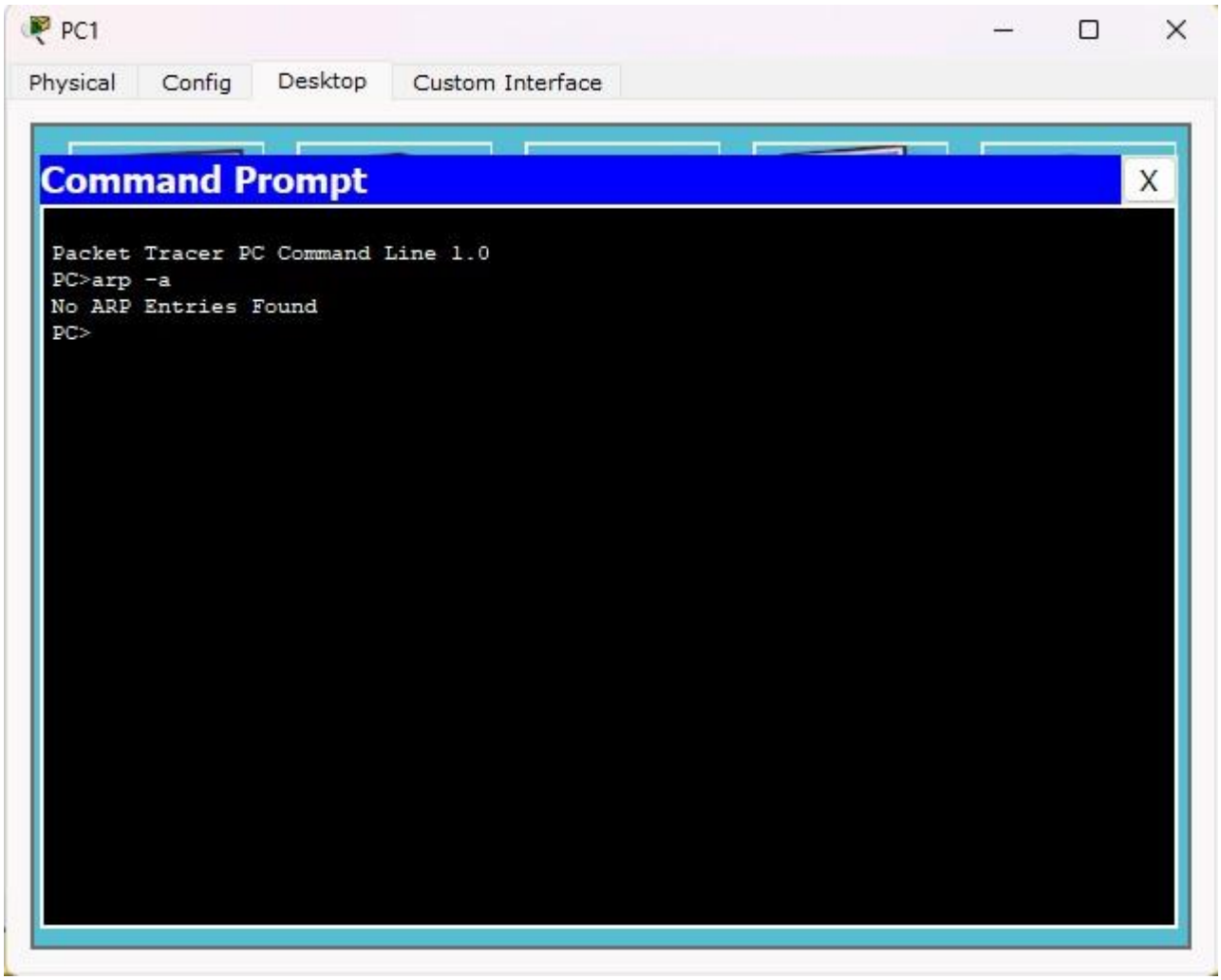
ARP Table for 10.0.0.4		
IP Address	Hardware Address	Interface
10.0.0.3	0001.C726.4785	FastEthernet0

7. Similarly other ARP tables are updated

Ok 10/25

Screen Shots:

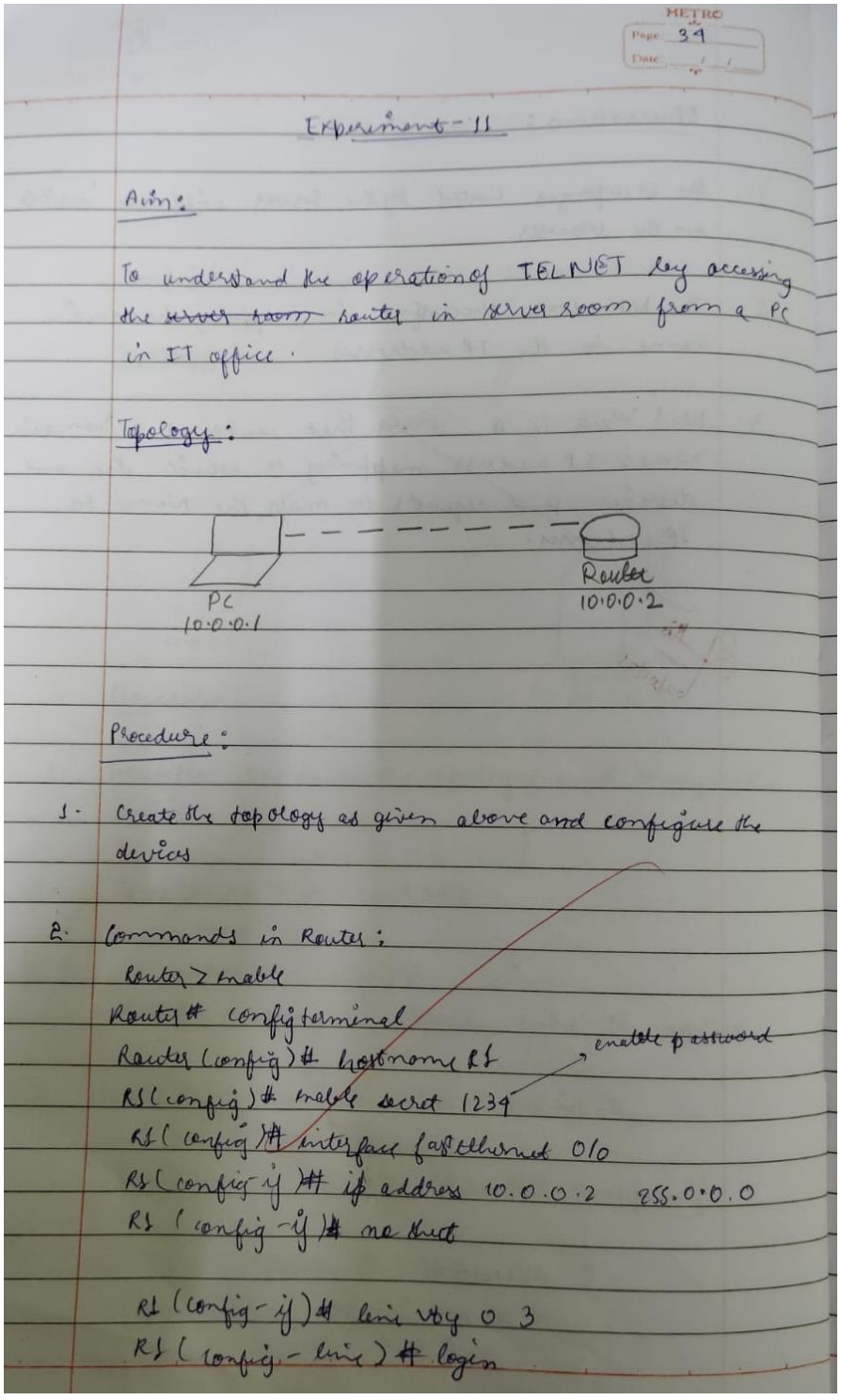




Program 11

Aim: To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.

Topology , Procedure and Observation:



Experiment - 11

Aim:

To understand the operation of TELNET by accessing the ~~server room~~ router in server room from a PC in IT office.

Topology:

Procedure:

1. Create the topology as given above and configure the devices

2. Commands in Router:

Router > enable
Router # config terminal
Router (config) # hostname R1
R1 (config) # enable secret 1234 → enable password
R1 (config) # interface fastethernet 0/0
R1 (config-if) # ip address 10.0.0.2 255.0.0.0
R1 (config-if) # no shut

R1 (config-if) # line vty 0 3
R1 (config-line) # login

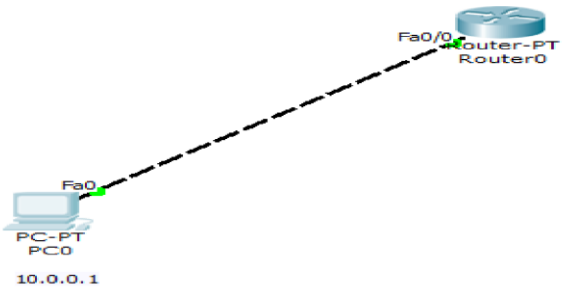
% Login disabled on line 194 until 'password' is set
R1(config-line)# password 4321
R1(config-line)# exit
R1(config)# exit
R1# wr
Building configuration...
[OK]

Note: vty 0 3 : First ~~four~~ five virtual terminal lines for Telnet access

3. In PC's command Prompt
- First try pinging to see if devices are connected
PC> telnet 10.0.0.2
Trying 10.0.0.2 ... Open
User Access Verification
Password: 4321
Password: 4321
R1> enable
Password: 1234
R1# show ip route
C 10.0.0.0/8 is directly connected, FastEthernet 0/0
R1#

Observations:
1. The admin in PC is able to run commands as run in router CLI and see the results from PC
2. Telnet allows user to establish a remote services with another device like router, over a TCP/IP network
3. Using Telnet, ~~no~~ we can access and control the remote devices CLI as if you were physically connected to it.

Screen Shots:



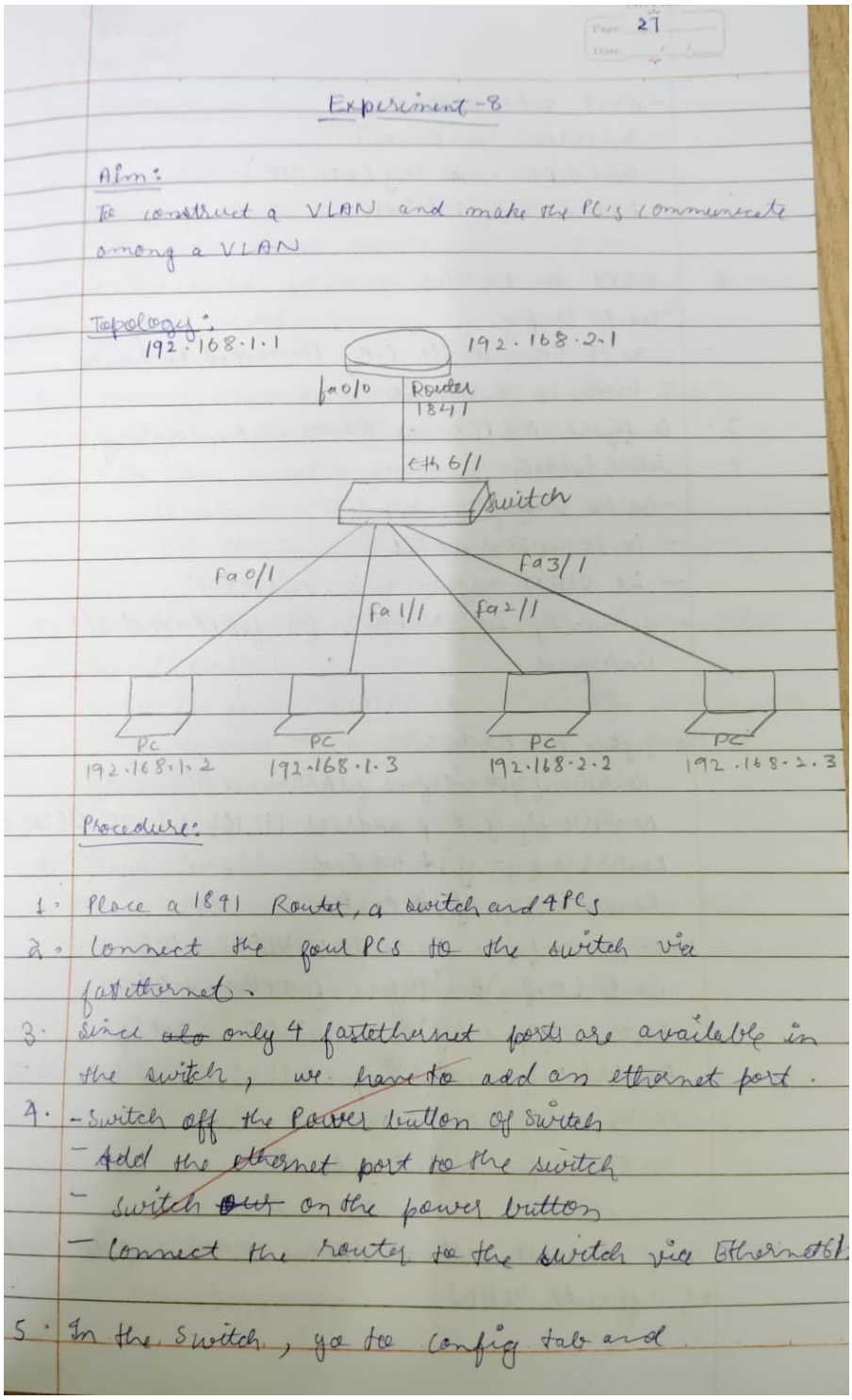
Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open


User Access Verification

Password:
R1>enable
Password:
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
R1#
```

Program 12

Aim: To construct a VLAN and make the PC's communicate among a VLAN .

Topology , Procedure and Observation:



Experiment-8

Aim:

To construct a VLAN and make the PC's communicate among a VLAN

Topology:

Procedure:

1. Place a 1841 Router, a switch and 4 PCs
2. Connect the four PCs to the switch via fastethernet.
3. Since ~~also~~ only 4 fastethernet ports are available in the switch, we have to add an ethernet port.
4. - Switch off the Power button of switch
   - Add the ethernet port to the switch
   - Switch ~~out~~ on the power button
   - Connect the Router to the switch via Ethernet 6/1
5. In the Switch, go to config tab and

- Select VLAN Database
- Give VLAN number say 2
- Give VLAN name say (cseise)
- Add it to the Database

6. Select the Switch:
- Go to config
- Go to Ethernet 0/1, i.e., connected to Router
- Make it the trunk

7. Configure the PCs as shown in the topology

8. Select Switch
- Go to config
- Go to FastEthernet 2/1
- Set VLAN number as 2, i.e., 'cseise'
- similarly set VLAN 2 for fastEthernet 3/1 interfaces

9. Configure the Router:

Router(config)# interface fastEthernet 0/0
Router(config-if)# ip address 192.168.1.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# exit

Now, we configure the Router's VLAN interface.

Router(config)# interface fastethernet 0/0.1
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ip address 192.168.2.1 255.255.255.0
Router(config-subif)# no shut
Router(config-subif)# exit

10. Ping devices within same VLAN and to devices of different VLAN

Observations:

1. When devices are pinged within same VLAN:
   - Pinging 192.168.1.3 from 192.168.1.2
   - The data packet doesn't go to the router
   - The switch forwards the packet without the need of the router.

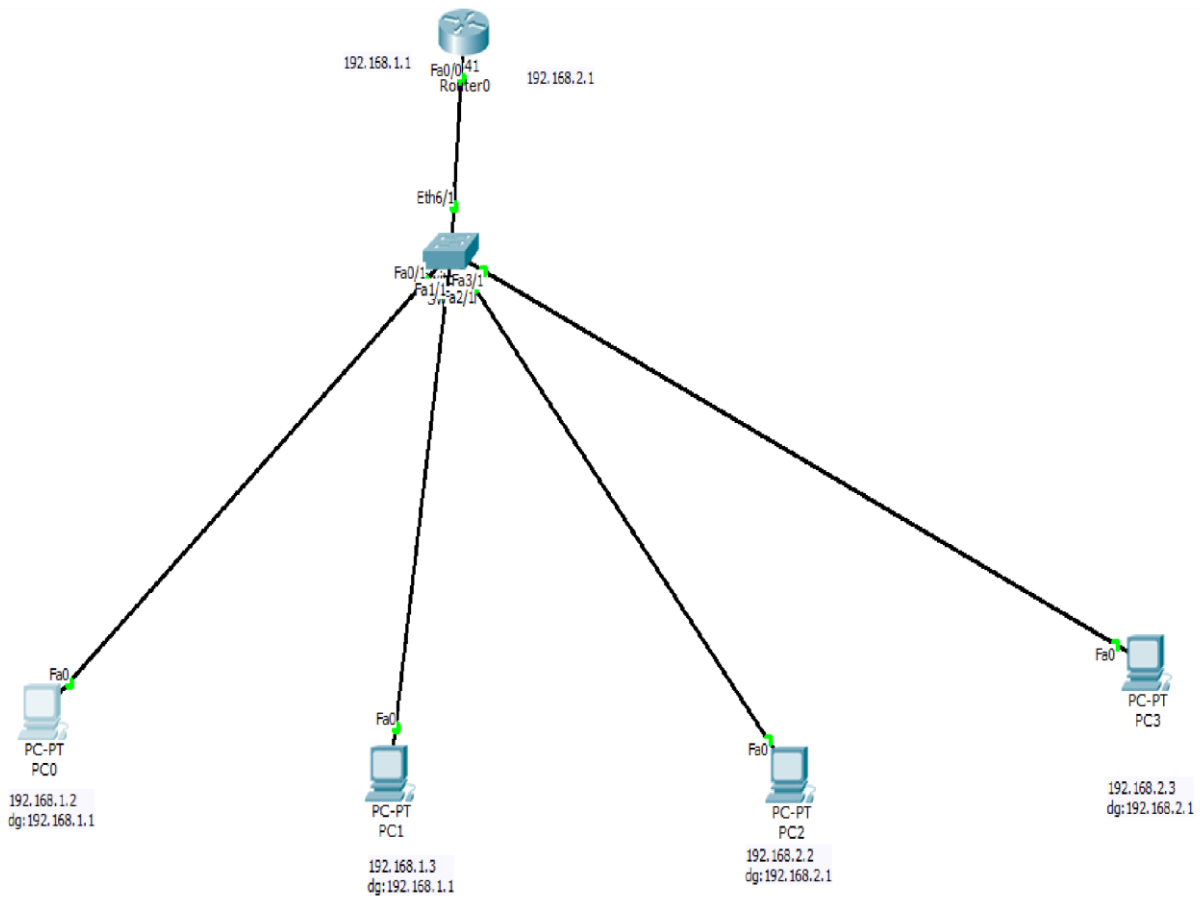
2. When a device pings a device of another VLAN
   - Pinging 192.168.2.3 from 192.168.1.2
   - The data packet's journey is as follows:
     192.168.1.2 → Switch → Router
     192.168.2.3 ← switch ←

3. VLANs divide a single switch into multiple logical switches.
   - Devices in one VLAN cannot directly communicate with devices in another VLAN without a router

4. Traffic Isolation:
   - Each VLAN maintains its own broadcast domain
   - Broadcasts sent by devices in on VLAN do not reach devices in another VLAN

5. VLAN trunking allows switches to forward frames from different VLANs over a single link called trunk
   - This is done by adding an additional header information called tag to the ethernet frame
   - VLAN tagging.

Screen Shots:



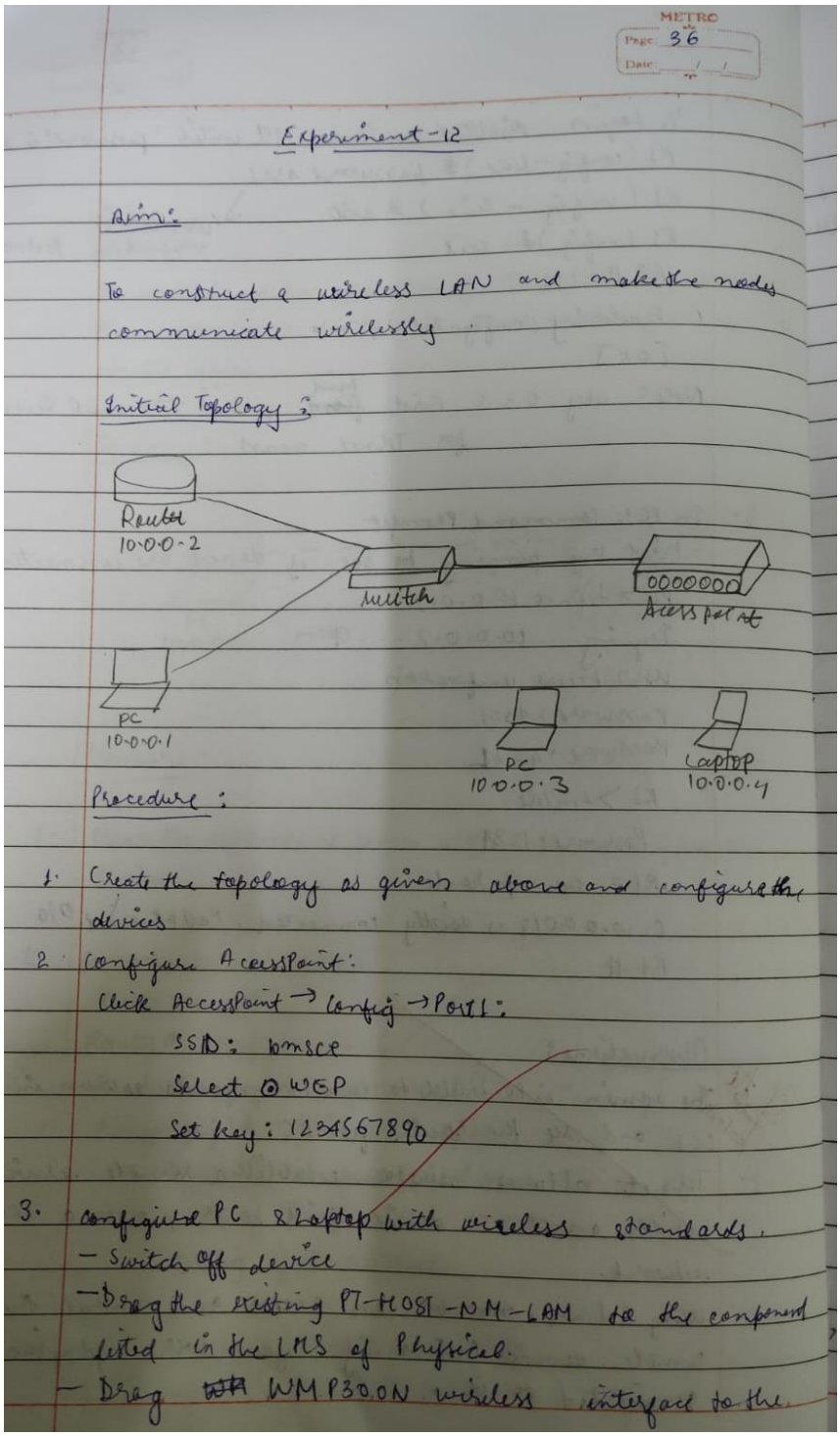
Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=4ms TTL=127

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 4ms, Average = 1ms

PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=2ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms

PC>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.3: bytes=32 time=3ms TTL=127
Reply from 192.168.2.3: bytes=32 time=2ms TTL=127
Reply from 192.168.2.3: bytes=32 time=1ms TTL=127

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 2ms

PC>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Reply from 192.168.2.3: bytes=32 time=2ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms

PC>
```

Program 13

Aim: To construct a WLAN and make the nodes communicate wirelessly.

Topology , Procedure and Observation:



Experiment - 12

Aim:

To construct a wireless LAN and make the nodes communicate wirelessly.

Initial Topology:

Router
10.0.0.2

Switch

Access Point

PC
10.0.0.1

PC
10.0.0.3

Laptop
10.0.0.4

Procedure:

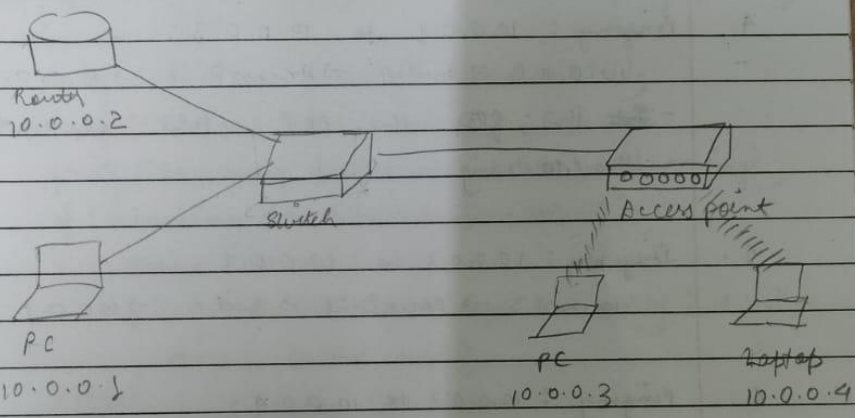
1. Create the topology as given above and configure the devices
2. Configure AccessPoint:
   Click AccessPoint → Config → Port1:
   SSID: bmsce
   Select ⊙ WEP
   Set key: 1234567890
3. Configure PC & Laptop with wireless standards.
   - Switch off device
   - Drag the existing PT-HOST-NM-1AM to the component listed in the LHS of Physical.
   - Drag WMP300N wireless interface to the

empty port
- Switch on the device

4. In the config tab, a new wireless interface was added.

5. Configure the device by entering SSID, WEP, WEP key, IP address and Gateway.

Topology after wireless configuration:



6. Ping from every device to every other device to check for connection

Observations:

1. We were able to ping from every device to every other device.

2. Access Point
- Creates bridge between wired and wireless devices
- SSID Broadcasting: announces the wireless network name (SSID) to allow device to connect using WEP, WPA or WPA2

3. WMP300N wireless interface:
- wireless network adapter that enables device to communicate with access point using wireless signals.

4. Pinging: 10.0.0.1 to 10.0.0.3:
10.0.0.1 → Switch → AccessPoint → 10.0.0.3
- This is after the ARP tables are updated after broadcasting.

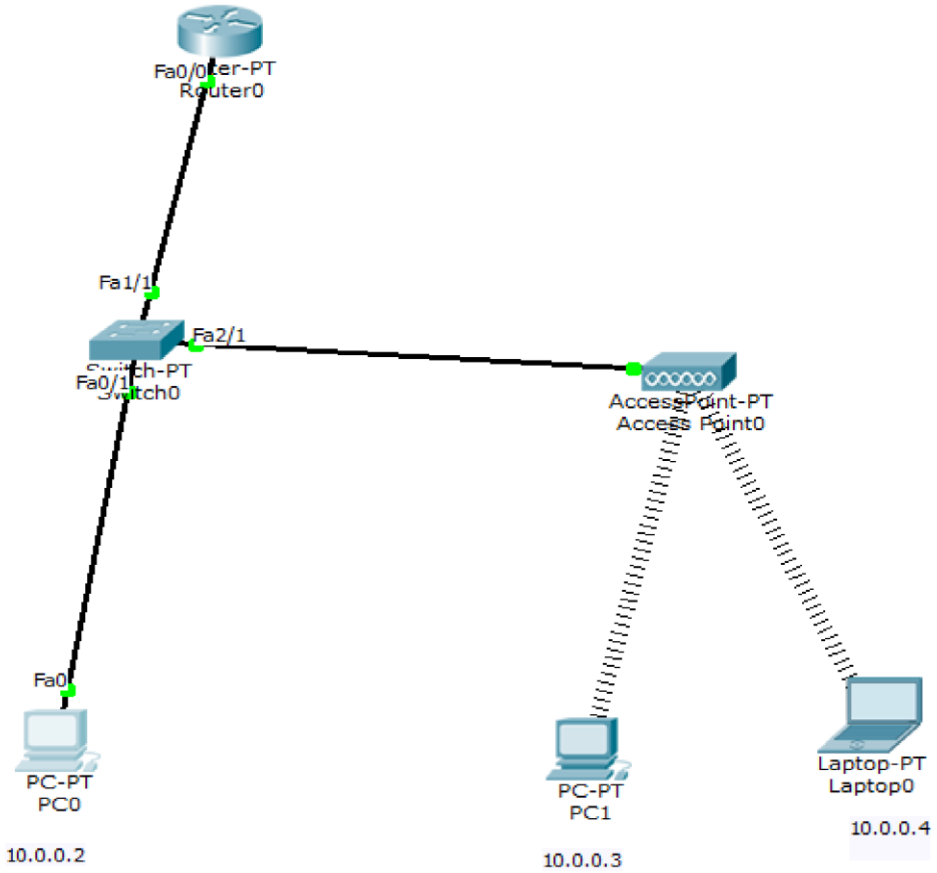
5. Pinging: 10.0.0.3 to 10.0.0.1
10.0.0.3 → AccessPoint → Switch → 10.0.0.1

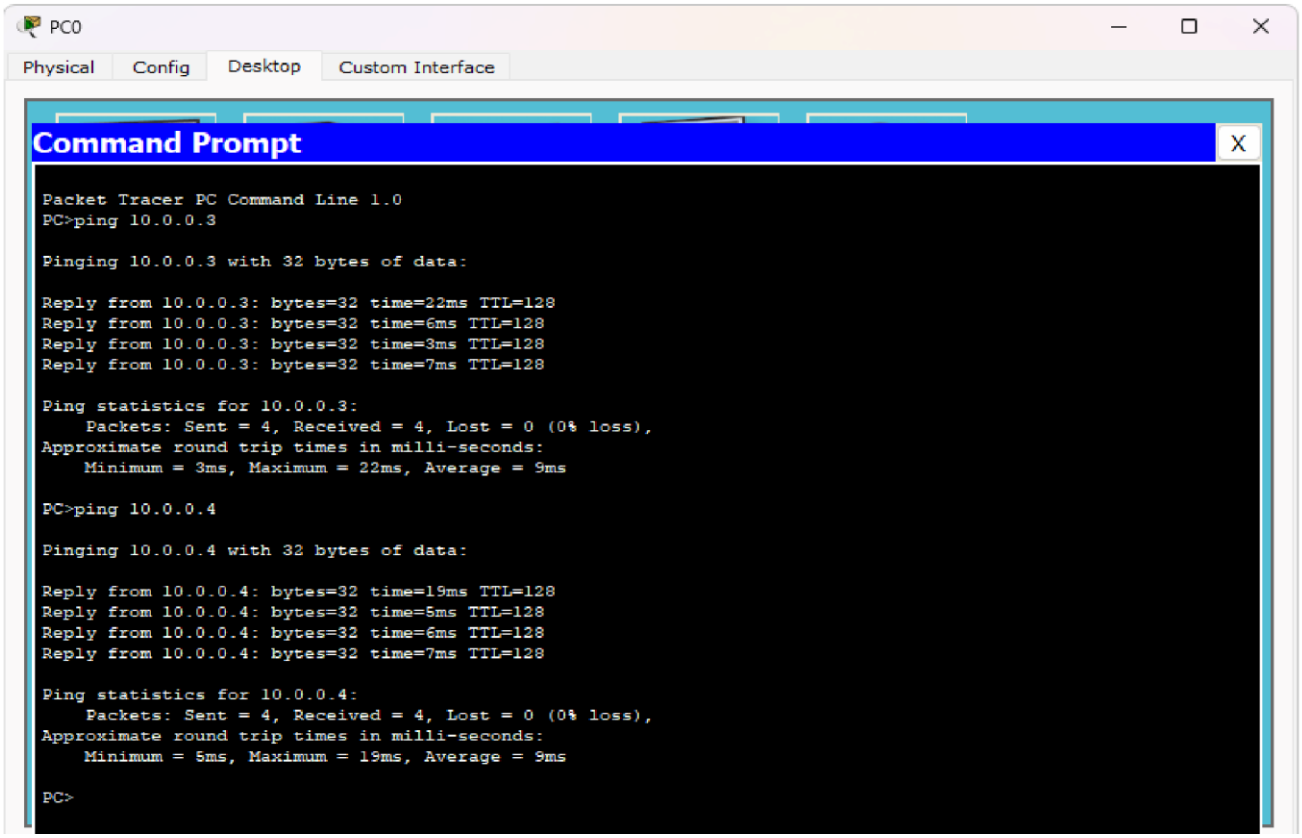
6. Pinging: 10.0.0.3 to 10.0.0.4:
10.0.0.3 → AccessPoint → 10.0.0.4

7. Every device is now connected to every other device in the WLAN

02/01/25

Screen Shots:





PART-B

Program 14

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

Code :

```
def xor(a, b):
    # XOR operation between two binary strings
    result = []
    for i in range(1, len(b)):
        result.append('0' if a[i] == b[i] else '1') return
    ''.join(result)

def mod2div(dividend, divisor): #
    Performs Modulo-2 division
    pick = len(divisor)
    tmp = dividend[:pick]

    while pick < len(dividend): if
        tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]
        else:
            tmp = xor('0' * pick, tmp) + dividend[pick] pick
        += 1

    # For the last set of bits if
    tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    return tmp

def encode_data(data, key): #
    Encode data with CRC
    l_key = len(key)
    padded_data = data + '0' * (l_key - 1)
    remainder = mod2div(padded_data, key)
    codeword = data + remainder
    return codeword, remainder

def check_data(received_data, key): #
    Check received data for errors
    remainder = mod2div(received_data, key)
    return '0' * (len(key) - 1) == remainder

# Main program
```

```
    if __name__ == "__main__":
        print("Error Detection using CRC-CCITT (8-bits)")

# Transmitter
        data = input("Enter data to be transmitted: ").strip()
        key = input("Enter the Generating polynomial: ").strip()

        print("\n- - - - - - - - - - - - - - - - - - - - - - - -     ")
        padded_data = data + '0' * (len(key) - 1) print("Data
        padded with n-1 zeros:", padded_data)

        encoded_data, crc = encode_data(data, key)
        print("CRC or Check value is:", crc)
        print("Final data to be sent:", encoded_data)
        print("- - - - - - - - - - - - - - - - - - - - - - - -     ")

        # Receiver
        received_data = input("\nEnter the received data: ").strip()
        print("\n- - - - - - - - - - - - - - - - -     ")
        print("Data received:", received_data)

        if check_data(received_data, key): print("No
            error detected")
        else:
            print("Error detected")
        print("- - - - - - - - - - - - - - - - -     ")
```

Output

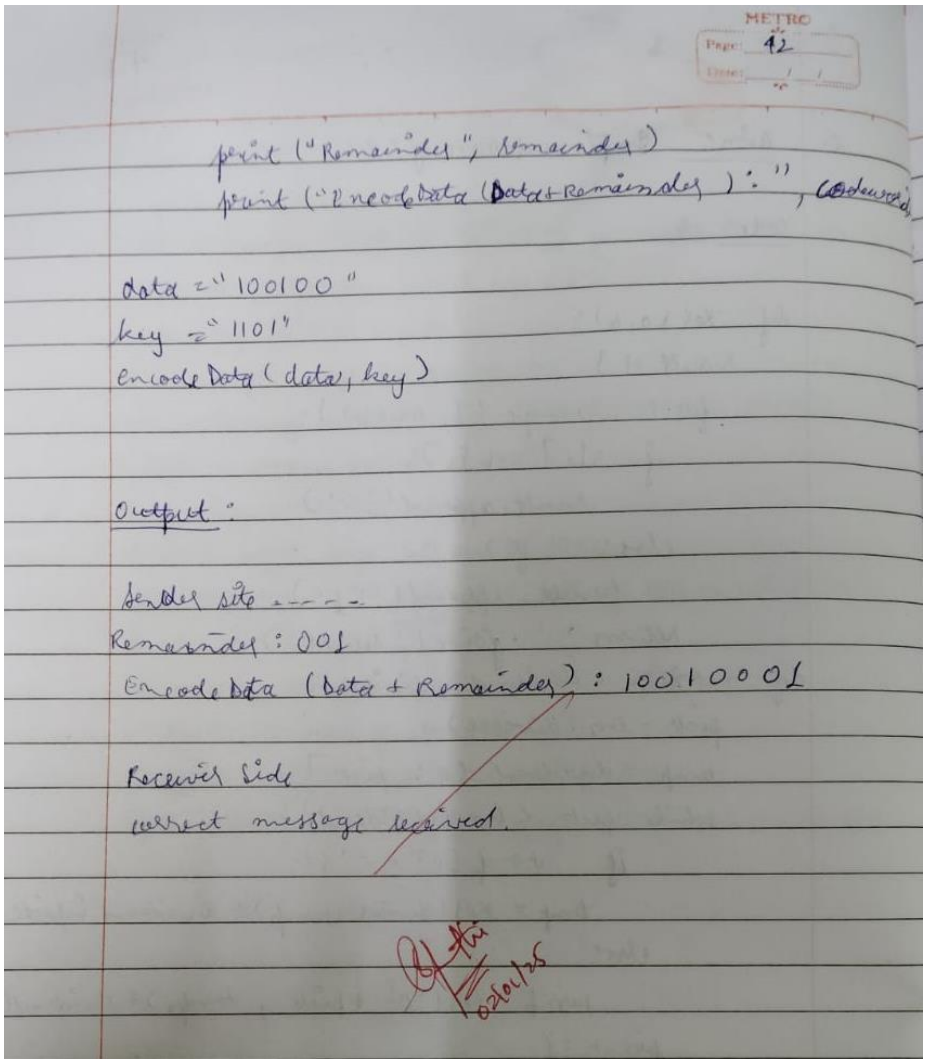
```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

---------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
---------------------------------------

Enter the received data: 10011000100011

-----------------------------
Data received: 10011000100011
Error detected
```

```
Error Detection using CRC-CCITT (8-bits)
Enter data to be transmitted: 1001100
cell output actions rating polynomial: 100001011

---------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 10100010
Final data to be sent: 100110010100010
---------------------------------------

Enter the received data: 100110010100010

-----------------------------
Data received: 100110010100010
No error detected
-----------------------------
```

2. Aim: Implementation of CRC

Code:

```
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 modDiv(dividend, divisor):
    pick = len(divisor)
    temp = dividend[0: pick]
    while pick < len(dividend):
        if temp[0] == '1':
            temp = xor(divisor, temp) + dividend[pick]
        else:
            temp = xor('0' * pick, temp) + dividend[pick]
        pick += 1
    if temp[0] == '1':
        temp = xor(divisor, temp)
    else:
        temp = xor('0' * pick, temp)
    checkword = temp
    return checkword
def encodeData(data, key):
    l_key = len(key)
    append_data = data + '0' * (l_key - 1)
    remainder = modDiv(append_data, key)
    codeword = data + remainder
```



print ("Remainder", remainder)
print ("Encode Data (Data+Remainder) :", codeword)

data = "100100"
key = "1101"
encodeData (data, key)

Output :

Sender side ----
Remainder : 001
Encode Data (Data + Remainder) : 1001001

Receiver Side
correct message received.

02/01/25

Program 15

Write a program for congestion control using Leaky bucket algorithm.

Code :

```
# Getting user inputs
storage = int(input("Enter initial packets in the bucket: "))
no_of_queries = int(input("Enter total no. of times bucket content is checked: ")) bucket_size
= int(input("Enter total no. of packets that can be accommodated in the bucket: "))
input_pkt_size = int(input("Enter no. of packets that enters the bucket at a time: "))
output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time: "))

for i in range(no_of_queries):  # space left
    size_left = bucket_size - storage
    if input_pkt_size <= size_left: #
        update storage
        storage += input_pkt_size
    else:
        print("Packet loss =", input_pkt_size)

    print(f"Buffer size = {storage} out of bucket size = {bucket_size}")

    # as packets are sent out into the network, the size of the storage decreases storage
    -= output_pkt_size
```

Output

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

Cycle 2

1. Aim : Implementation of Leaky Bucket Algorithm.

Code :

```
#include <stdio.h>
int main(){
  int incoming, outgoing, bucket_size, n, store=0;
  printf("Enter bucket size, outgoing rate and no of input");
  scanf("%d %d %d", &bucket_size, &outgoing, &n);
  while(n!=0){
    printf("Enter the incoming packet size :");
    scanf("%d", &incoming);
    printf("Incoming packet size %d \n", incoming);
    if(incoming <= (bucket_size - store)){
      store += incoming;
      printf("Bucket buffer size %d out of %d \n",
        store, bucket_size);
    }else{
      printf("Dropped %d no. of packets \n", incoming -
        (bucket_size - store));
      printf("Bucket buffer size %d out of %d \n", store,
        bucket_size);
      store = bucket_size;
    }
    store = store - outgoing;
    printf("After outgoing %d packets left out of %d in
      buffer \n", store, bucket_size);
    n--;
  }
}
```

Output:

Enter bucket size, outgoing rate & no of inputs : 100 20 3
Enter the incoming packet size : 30
Incoming packet size 30
Bucket buffer size 30 out of 100
After outgoing 10 bytes left out of 100 in buffer
Enter incoming packet size : 50
Incoming packet size 50
Bucket buffer size 60 out of 100
After outgoing 40 bytes left out of 100 in buffer
Enter the incoming packet size : 80
Incoming packet size 80
Dropped 20 no of packets
Bucket buffer size 40 out of 100
After outgoing 80 bytes left out of 100 in buffer.

02/01/26

Program 16

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

Code:

Client.py

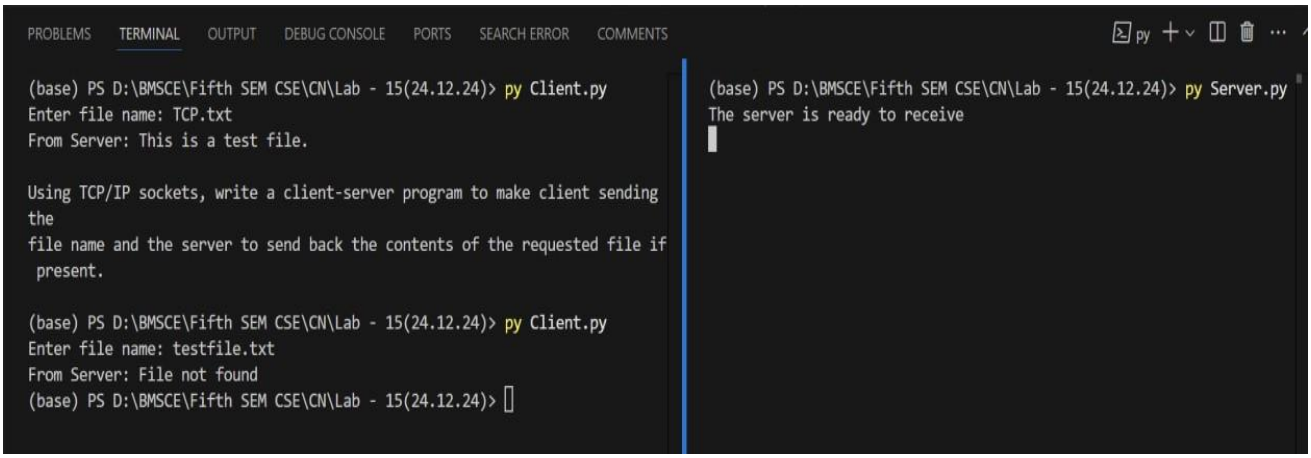
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))  # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

Server.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number to listen on

# Create TCP socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))  # Bind socket to the address and port
serverSocket.listen(1)  # Listen for 1 connection
print("The server is ready to receive")

while True:
    # Accept a connection
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Try opening the file try:
```

```
file = open(sentence, "r")  # Open file in read mode
            fileContents = file.read(1024)  # Read file content (up to 1024 bytes)
            file.close()
        except FileNotFoundError:
            # Send error message if file not found
            connectionSocket.send("File not found".encode())

        # Close the connection
        connectionSocket.close()
```

Output



3. Aim : Implementation of TCP/IP

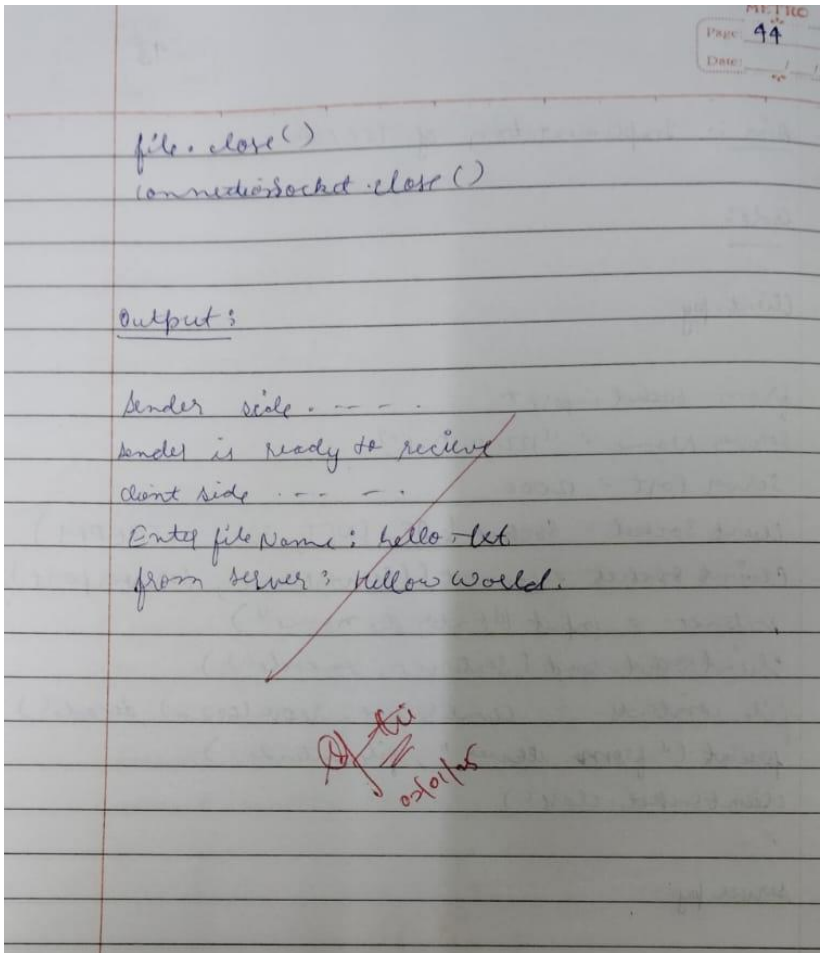
Code:

Client.py

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
ClientSocket = Socket (AF_INET, SOCK_STREAM)
ClientSocket = (connect ((ServerName, Serverport))
sentence = input ("Enter the name")
ClientSocket.send (sentence.encode())
file contents = ClientSocket.recv (6024).decode()
print ("from server", filecontents)
ClientSocket.close()
```

Server.py

```
from Socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
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).dec
    file = Open (sentence, "r")
    l = file.read (1024)
    connectionSocket.send (l.encode())
```


Page: 44
Date:
file.close()
connectionSocket.close()
Output:
Sender side. - - - .
Sender is ready to recieve
client side - - - - .
Enter file Name: hello.txt
from server: Hellow World.

Program 17

Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.

Code:

ClientUDP.py

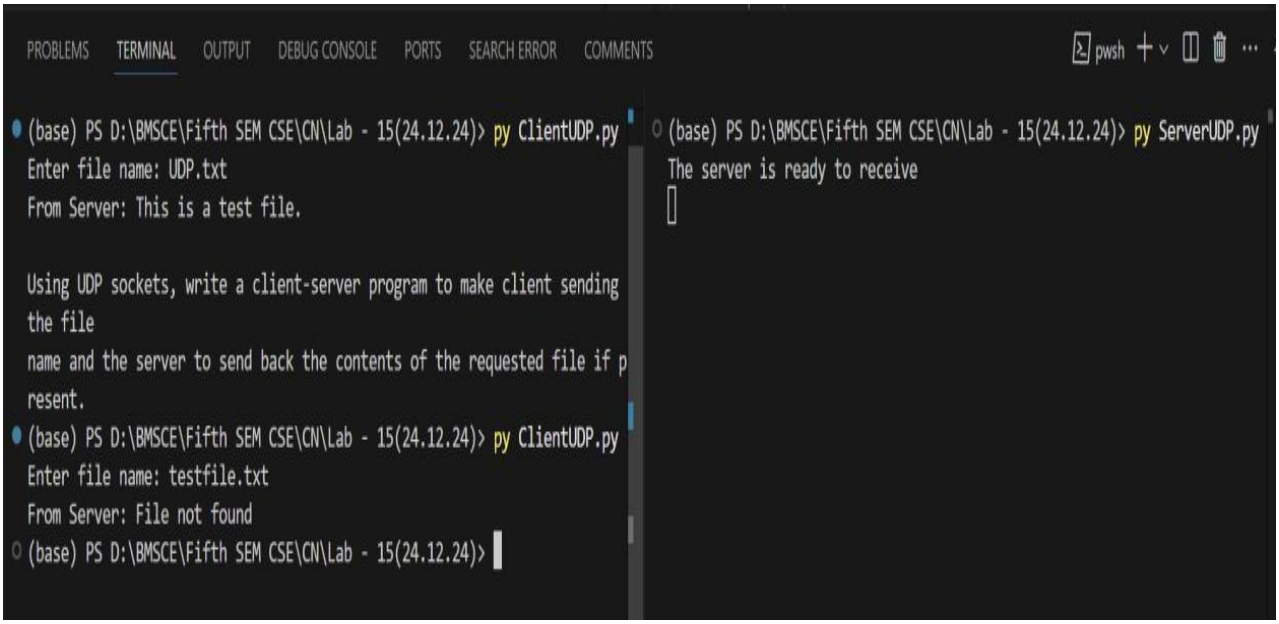
```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("Enter file name: ")
clientSocket.sendto(sentence.encode(), (serverName, serverPort))

filecontents, serverAddress = clientSocket.recvfrom(2048)
print('From Server:', filecontents.decode())

clientSocket.close()
```

ServerUDP.py

```
from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

print("The server is ready to receive")

while True:
    sentence, clientAddress = serverSocket.recvfrom(2048)
```

```
try:
    with open(sentence.decode(), "r") as file:
        l = file.read(2048)
        serverSocket.sendto(l.encode(), clientAddress)
        print(f"Sent back to client: {l}")
except FileNotFoundError:
    serverSocket.sendto("File not found.".encode(), clientAddress)
```

Output



4. Aim: Implement UDP

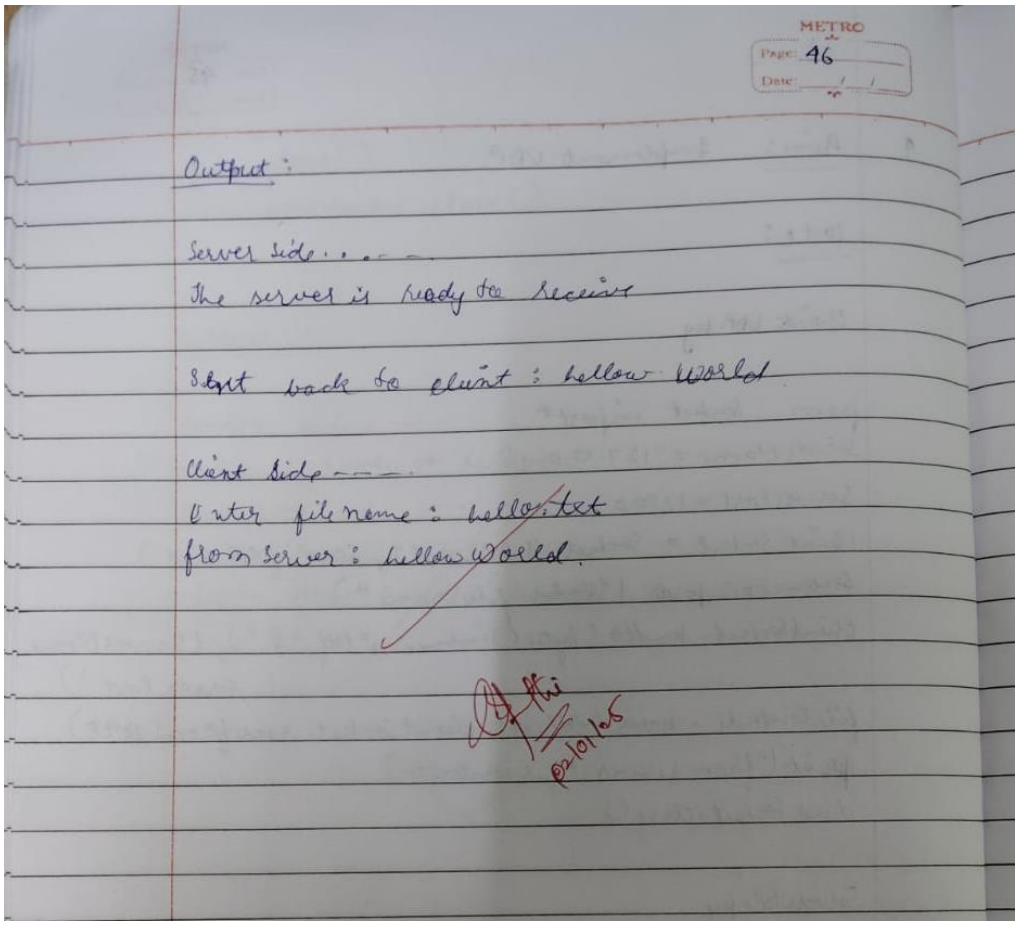
Code:

Client UDP.py

```
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()
```

ServerUDP.py

```
from socket import*
serverPort = 12000
serverSocket = socket (AF_INET, SOCK_DGRAM)
serverSocket.bind (("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    file = open (sentence, "r")
    l = file.read (2048)
    server.socket.sendto (bytes (l, "utf-8"), clientAddress)
    print ('sent back to client", l)
file.close()
```


METRO
Page: 46
Date: / /
Output:
Server side. . . .
The server is ready to receive
Sent back to client : hellow World
Client Side ----
Enter file name : hello.txt
from Server : hellow World.
21/10/25