VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

SUMITH U N (1BM22CS297)

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
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 **SUMITH U N (1BM22CS297),** 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.

Dr. Shashikala	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE, BMSCE	Department of CSE, BMSCE

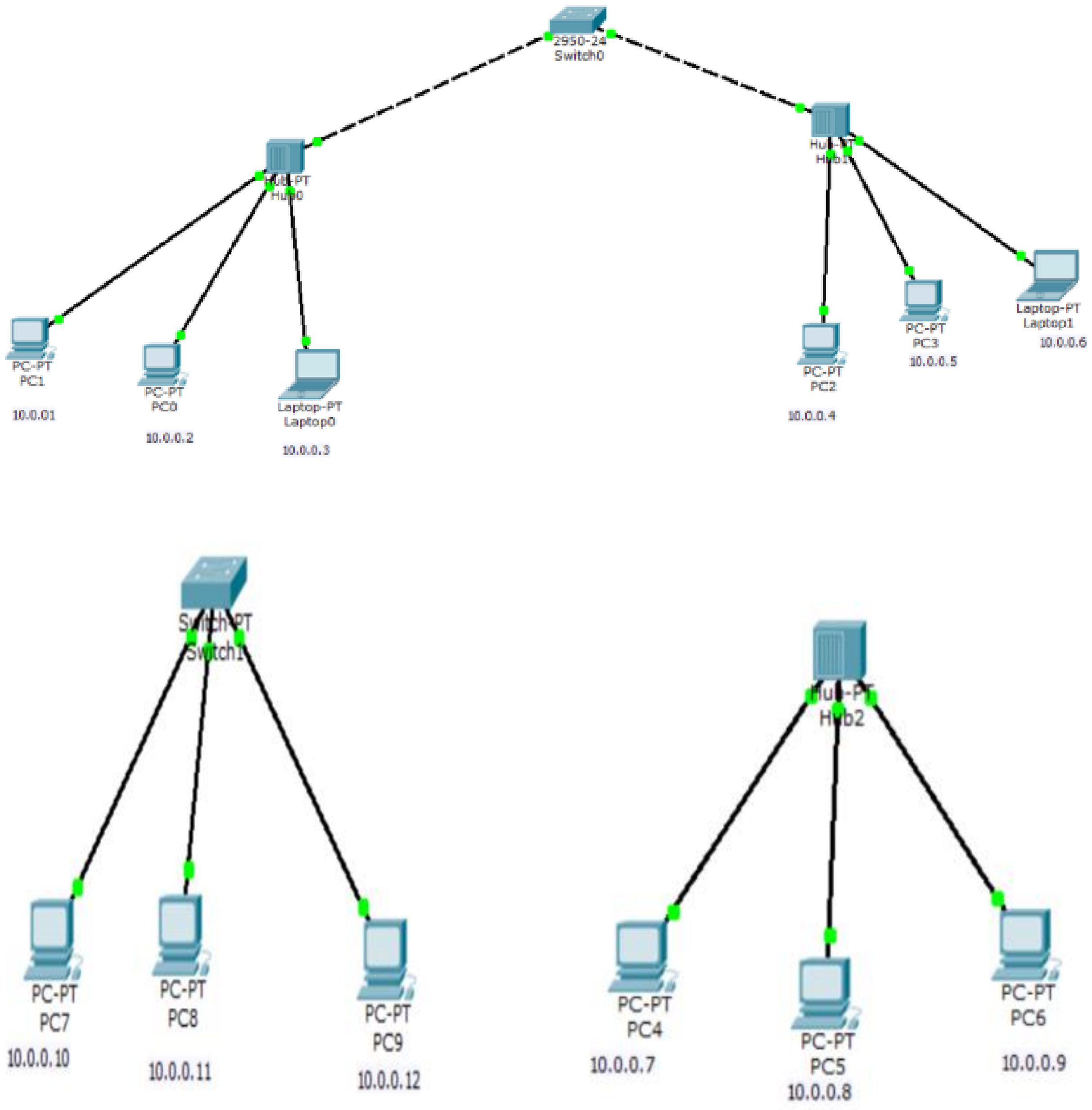
Index

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.	1 - 3
2	16/10/24	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	4 - 6
3	23/10/24	Configure default route, static route to the Router.	7 - 8
4	13/11/24	Configure DHCP within a LAN and outside LAN.	9 - 11
5	20/11/24	Configure RIP routing Protocol in Routers .	12 - 14
6	20/11/24	Demonstrate the TTL/ Life of a Packet.	15 - 16
7	27/11/24	Configure OSPF routing protocol.	17 - 19
8	18/12/24	Configure Web Server, DNS within a LAN.	20 - 21
9	18/12/24	To construct a simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	22 - 24
10	18/12/24	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	25 - 27
11	18/12/24	To construct a VLAN and make the PC's communicate among a VLAN.	28 - 29
12	18/12/24	To construct a WLAN and make the nodes communicate wirelessly.	30 - 31
13	18/12/24	Write a program for error detecting code using CRC-CCITT (16-bits).	32 - 33
14	18/12/24	Write a program for congestion control using Leaky bucket algorithm.	34 - 36
15	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.	37 - 39
16	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.	40 - 41

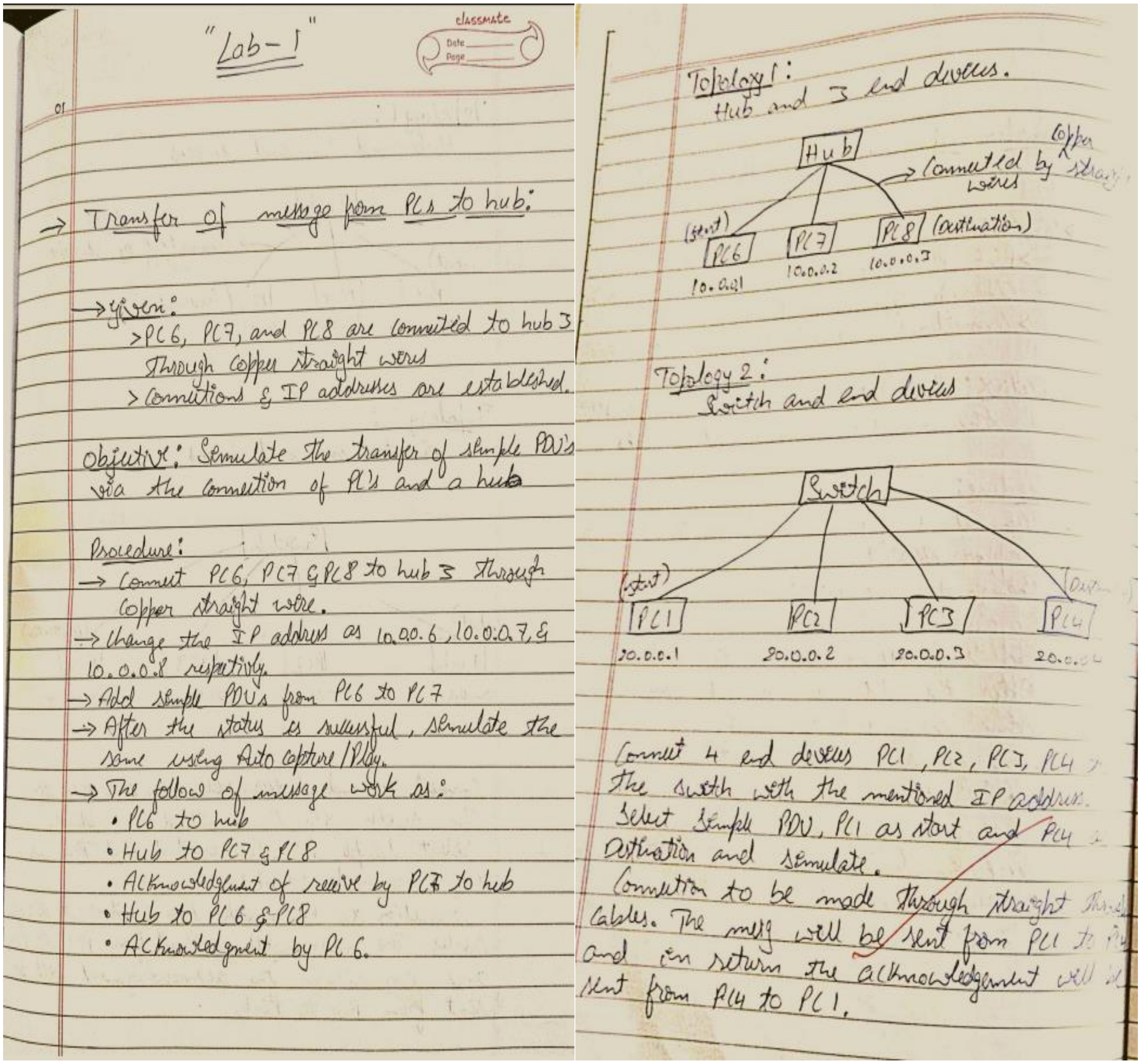
Program 1:

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 Observations:



"Lab-1"

classmate Date____ Page____

01

→ Transfer of message from PCs to hub:

→ given:

>PC6, PC7, and PC8 are connected to hub 3 through copper straight wires

> connections & IP addresses are established.

Objective: Simulate the transfer of simple PDU's via the connection of PC's and a hub

Procedure:

→ Connect PC6, PC7 & PC8 to hub 3 through copper straight wire.

→ Change the IP address as 10.0.0.6, 10.0.0.7, & 10.0.0.8 respectively.

→ Add simple PDU's from PC6 to PC7

→ After the status is successful, simulate the same using Auto capture/Play.

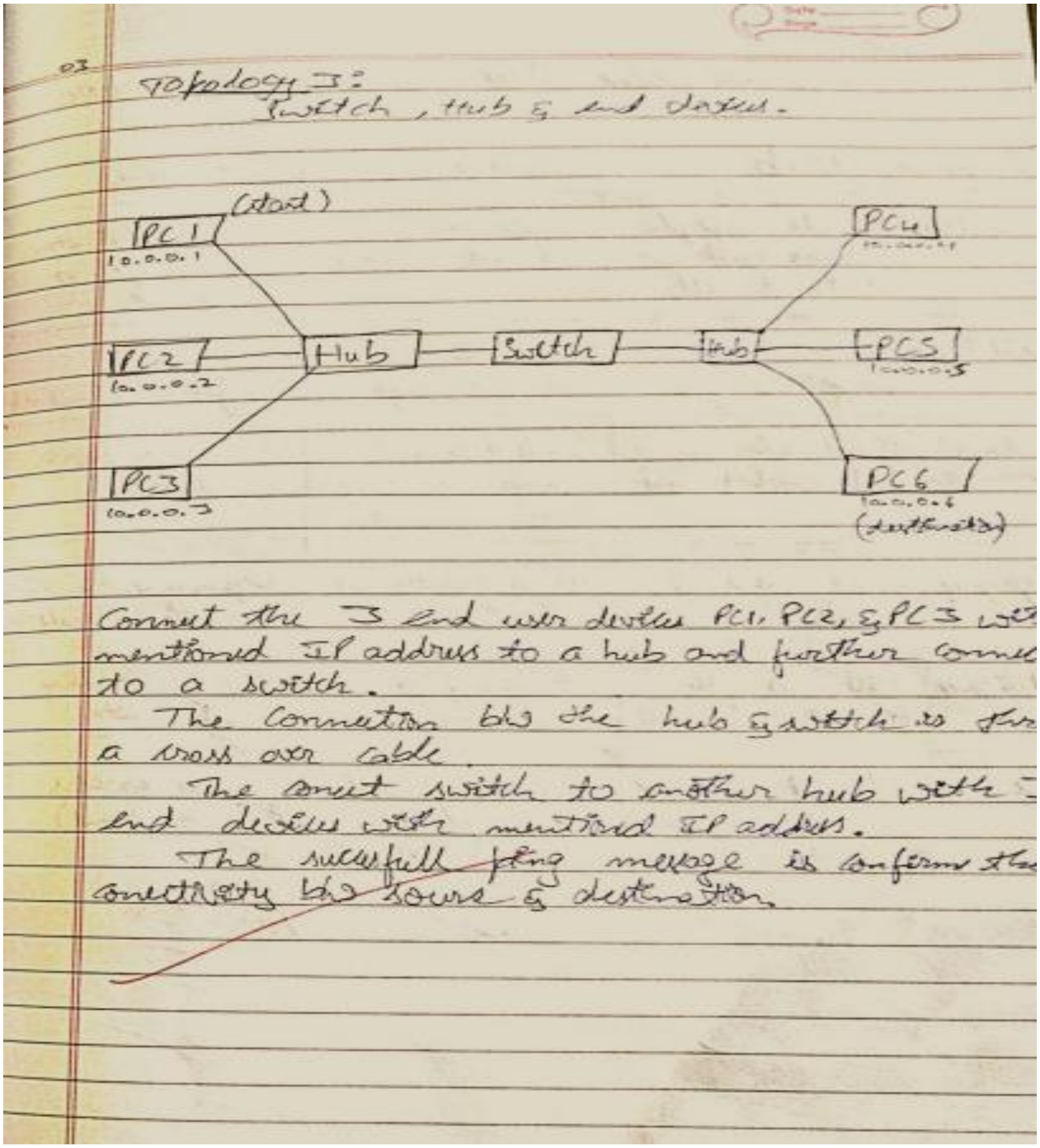
→ The follow of message work as:

- PC6 to hub
- Hub to PC7 & PC8
- Acknowledgment of receive by PC7 to hub
- Hub to PC6 & PC8
- Acknowledgment by PC6.

Topology 1:
Hub and 3 end devices.

Topology 2:
Switch and end devices

Connect 4 end devices PC1, PC2, PC3, PC4 to the switch with the mentioned IP address. Select Simple PDU, PC1 as start and PC4 as destination and simulate. Connection to be made through straight through cables. The msg will be sent from PC1 to PC4 and in return the acknowledgement will be sent from PC4 to PC1.



03 Topology 3:
Switch, hub & end devices.

(start)
PC1
10.0.0.1
PC2
10.0.0.2
PC3
10.0.0.3
Hub
Switch
Hub
PC4
10.0.0.4
PC5
10.0.0.5
PC6
10.0.0.6
(destination)

Connect the 3 end user devices PC1, PC2, & PC3 wit mentioned IP address to a hub and further connec to a switch.

The Connection b/w the hub & switch is thr a cross over cable.

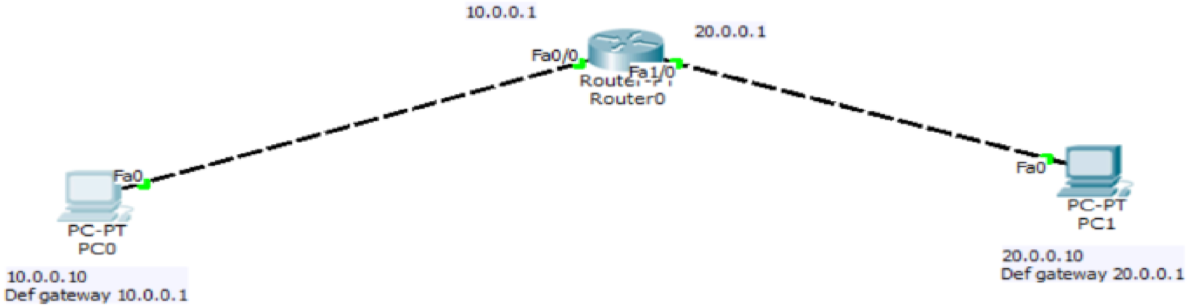
The conect switch to another hub with 3 end devices with mentioned IP address.

The succesfull ping message is conform the conectivity b/w the sours & destination.

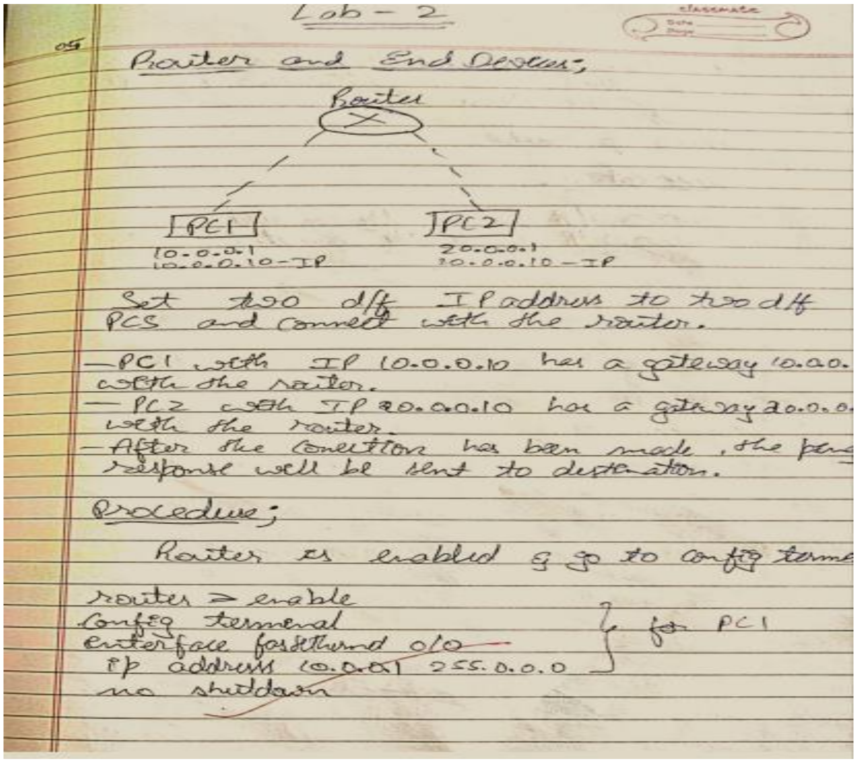
Program 2 :

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

Topology:



Procedure and Observations:



Lab - 2

05

Router and End Devices;

Router

PC1

10.0.0.1
10.0.0.10 - IP

PC2

20.0.0.1
20.0.0.10 - IP

Set two d/f IP address to two d/f PCS and connect with the router.

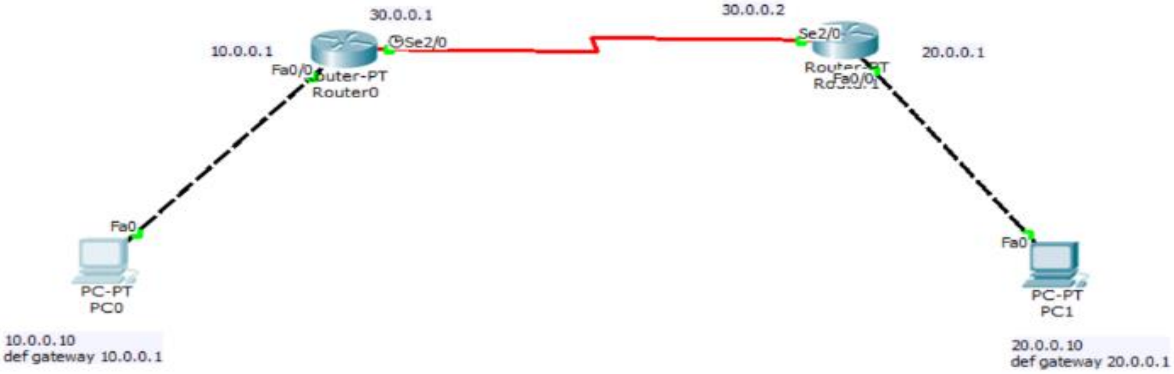
- PC1 with IP 10.0.0.10 has a gateway 10.0.0.
with the router.
- PC2 with IP 20.0.0.10 has a gateway 20.0.0.0
with the router.
- After the connection has been made, the ping
response will be sent to destination.

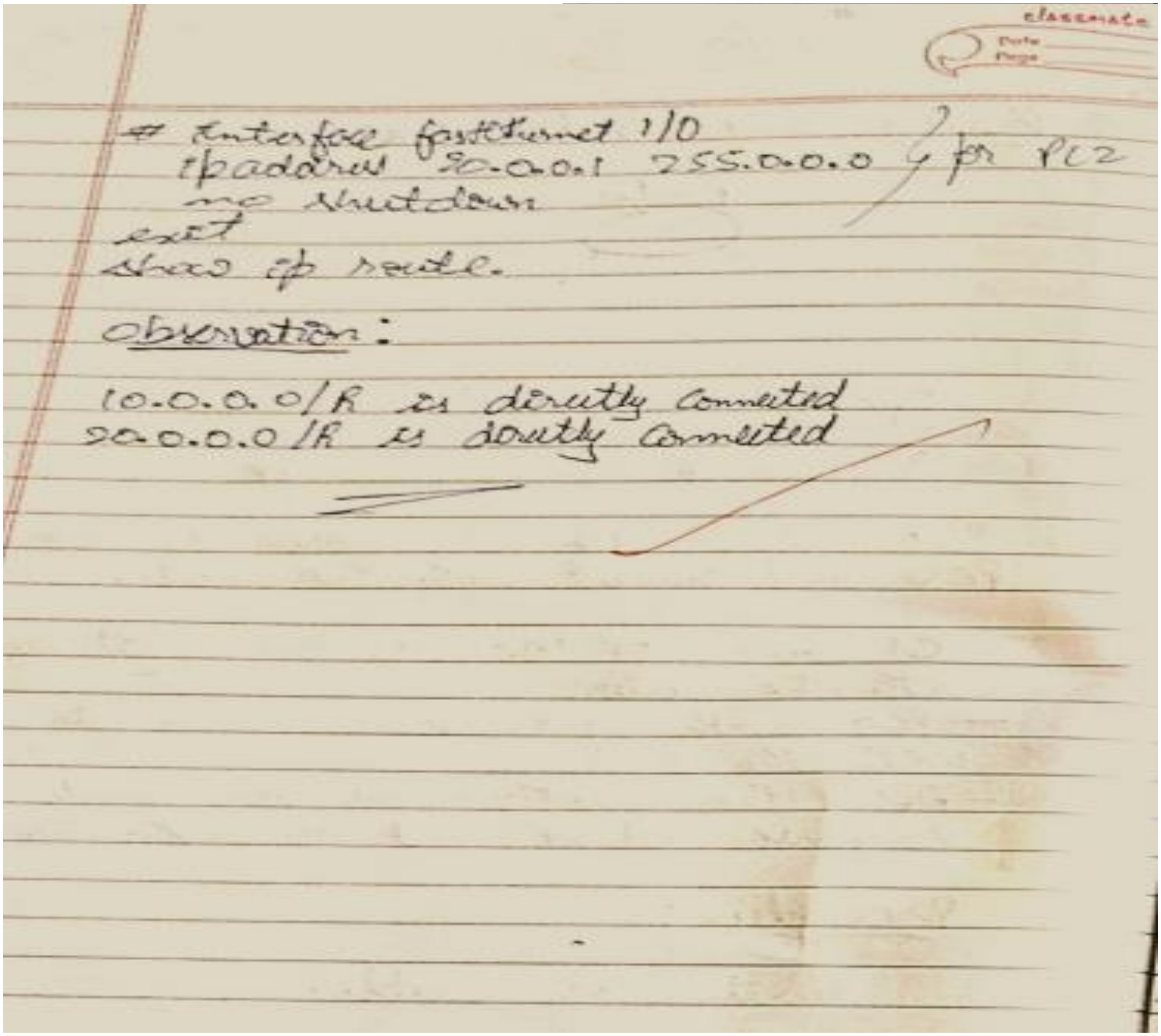
Procedure;

Router is enabled & go to config termi

```
router > enable
config terminal
enterface fasEthernet 0/0
 ip address 10.0.0.1 255.0.0.0
no shutdown
```

} for PC1

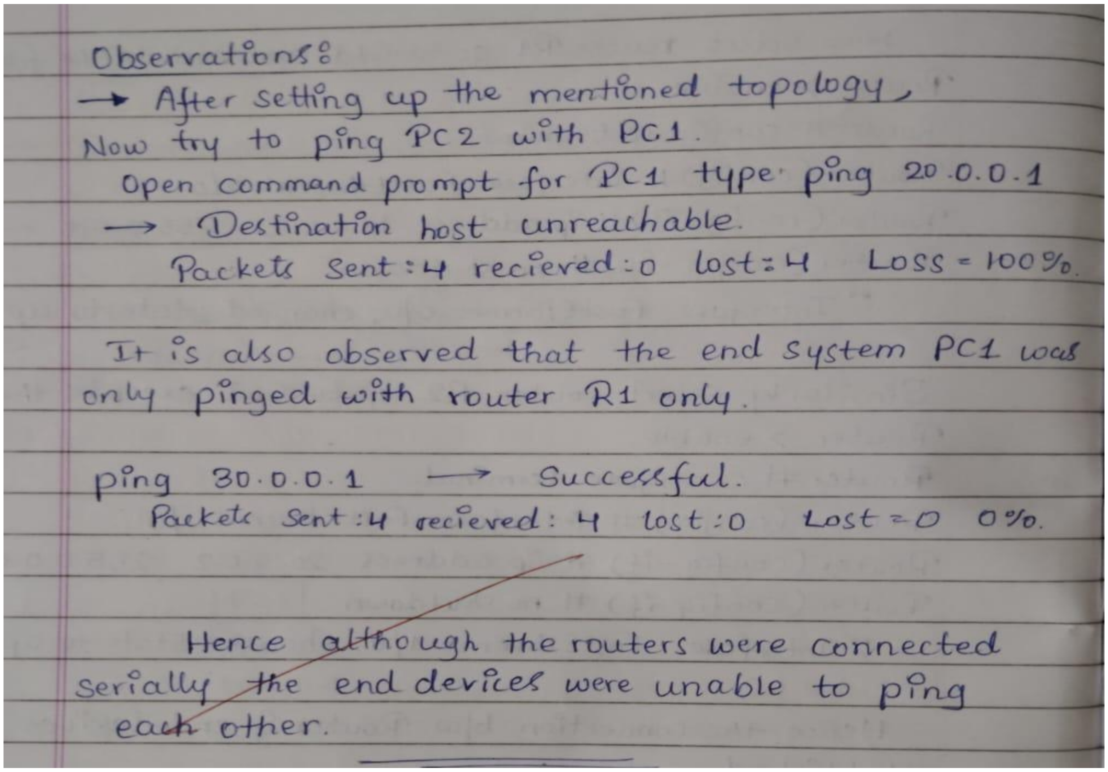




```
# Interface fastEthernet 1/0
  ipaddress 20.0.0.1 255.0.0.0 } for PC2
  no shutdown
exit
show ip route.
```

observation:

10.0.0.0/8 is directly connected
20.0.0.0/8 is directly connected



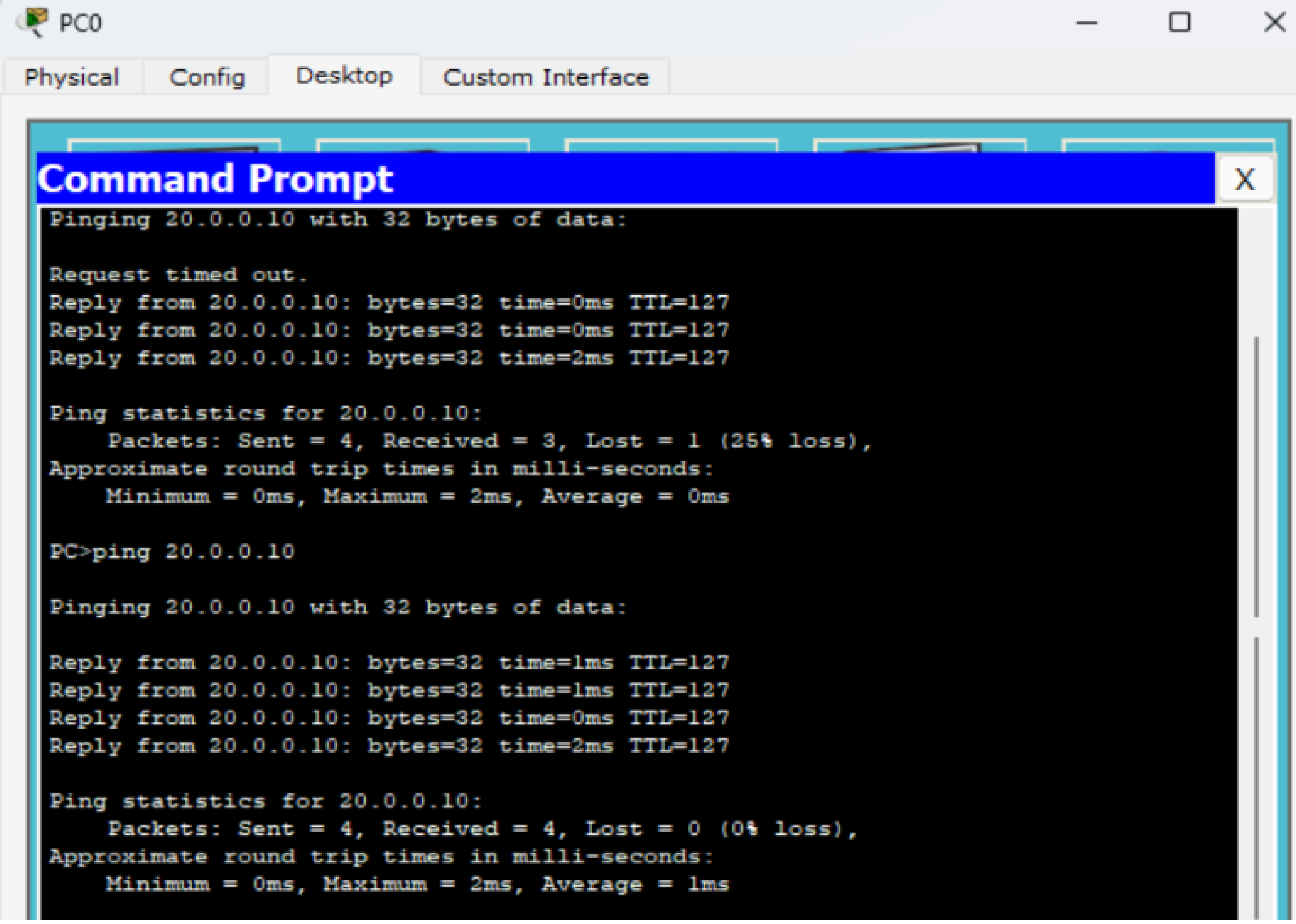
Observations:

→ After setting up the mentioned topology,
Now try to ping PC2 with PC1.
Open command prompt for PC1 type ping 20.0.0.1
→ Destination host unreachable.
Packets Sent : 4 recieved : 0 lost : 4 Loss = 100%.

It is also observed that the end system PC1 was only pinged with router R1 only.

ping 30.0.0.1 → Successful.
Packets Sent : 4 recieved : 4 lost : 0 Lost = 0 0%.

Hence although the routers were connected serially the end devices were unable to ping each other.

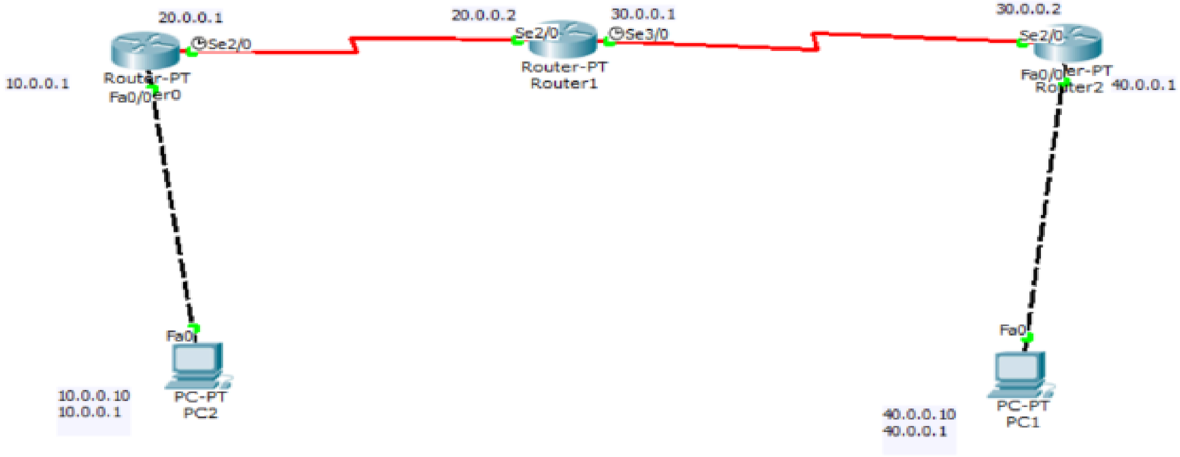


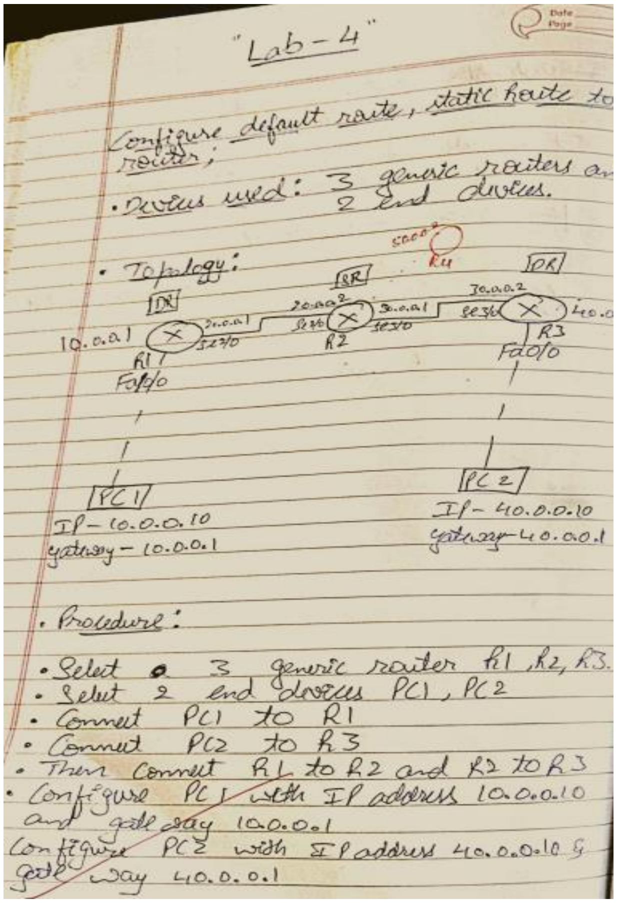
```
Pinging 20.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=2ms TTL=127

Ping statistics for 20.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, 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=1ms TTL=127
Reply from 20.0.0.10: bytes=32 time=1ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=2ms TTL=127

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 = 2ms, Average = 1ms
```

Program 3:

Aim: Configure default route, static route to the Router**.**

Topology, Procedure and Observations :





"Lab-4"

Configure default route, static route to router;

- Devices used: 3 generic routers an
  2 end devices.

- Topology:

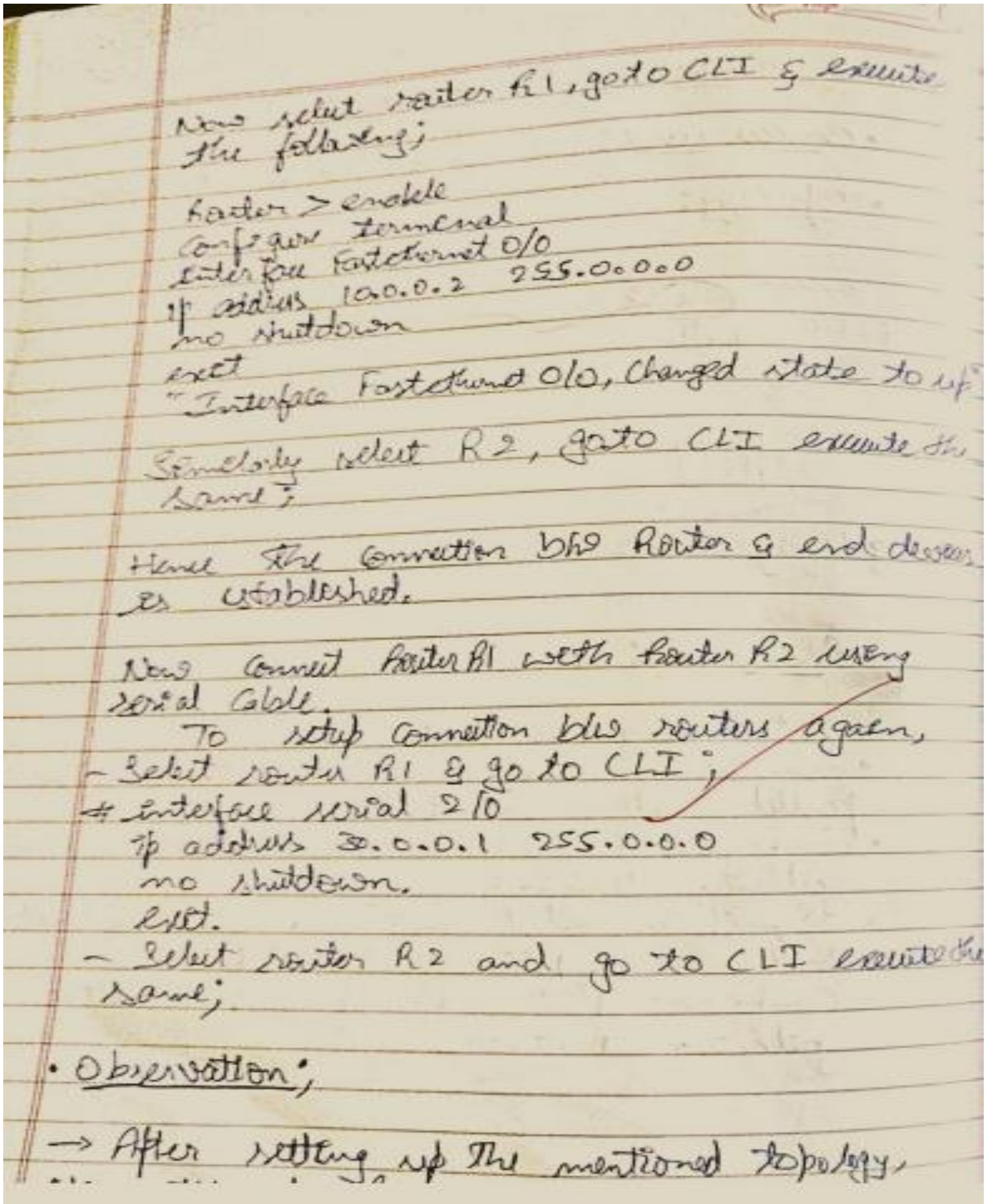
5000
R4
DR
BR
DR
10.0.0.1
20.0.0.1
Se2/0
R1
Fa0/0
20.0.0.2
Se3/0
R2
30.0.0.1
Se3/0
30.0.0.2
Se3/0
R3
40.0
Fa0/0
PC1
PC2

IP - 10.0.0.10
gateway - 10.0.0.1

IP - 40.0.0.10
gateway - 40.0.0.1

- Procedure:

- Select 3 generic router R1, R2, R3.
- Select 2 end devices PC1, PC2
- Connect PC1 to R1
- Connect PC2 to R3
- Then Connect R1 to R2 and R2 to R3
- Configure PC1 with IP address 10.0.0.10 and gateway 10.0.0.1
Configure PC2 with IP address 40.0.0.10 & gateway 40.0.0.1



Now select router R1, goto CLI & execute the following;

```
Router > enable
configure terminal
interface Fastethernet 0/0
ip address 10.0.0.2  255.0.0.0
no shutdown
exit
```

"Interface Fastethernet 0/0, Changed state to up"

Similarly select R2, goto CLI execute the same;

Hence the connection b/w Router & end devices is established.

Now connect Router R1 with Router R2 using serial Cable.

To setup connection b/w routers again,

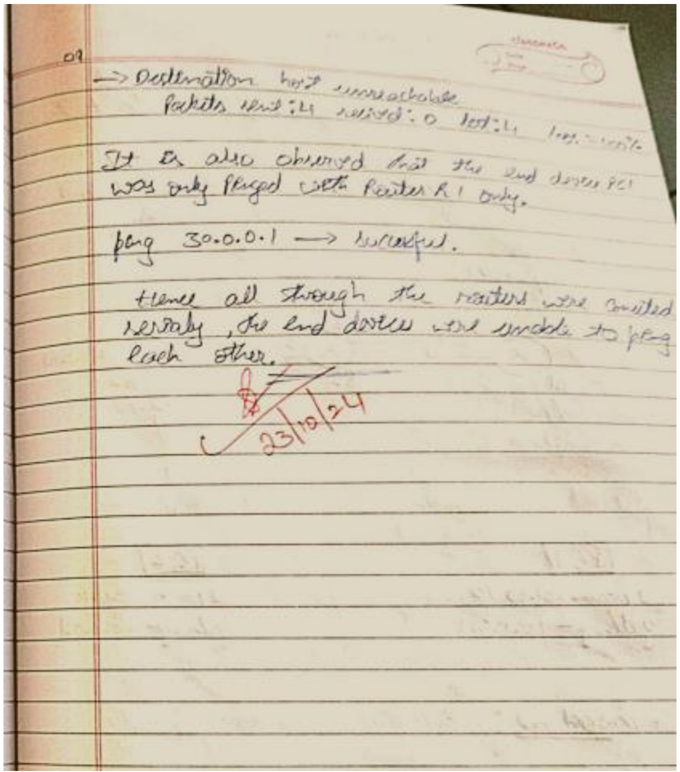
- Select router R1 & go to CLI;

```
# interface serial 2/0
  ip address 20.0.0.1  255.0.0.0
  no shutdown.
  exit.
```

- Select router R2 and go to CLI execute the same;

• Observation;

→ After setting up the mentioned topology,



09

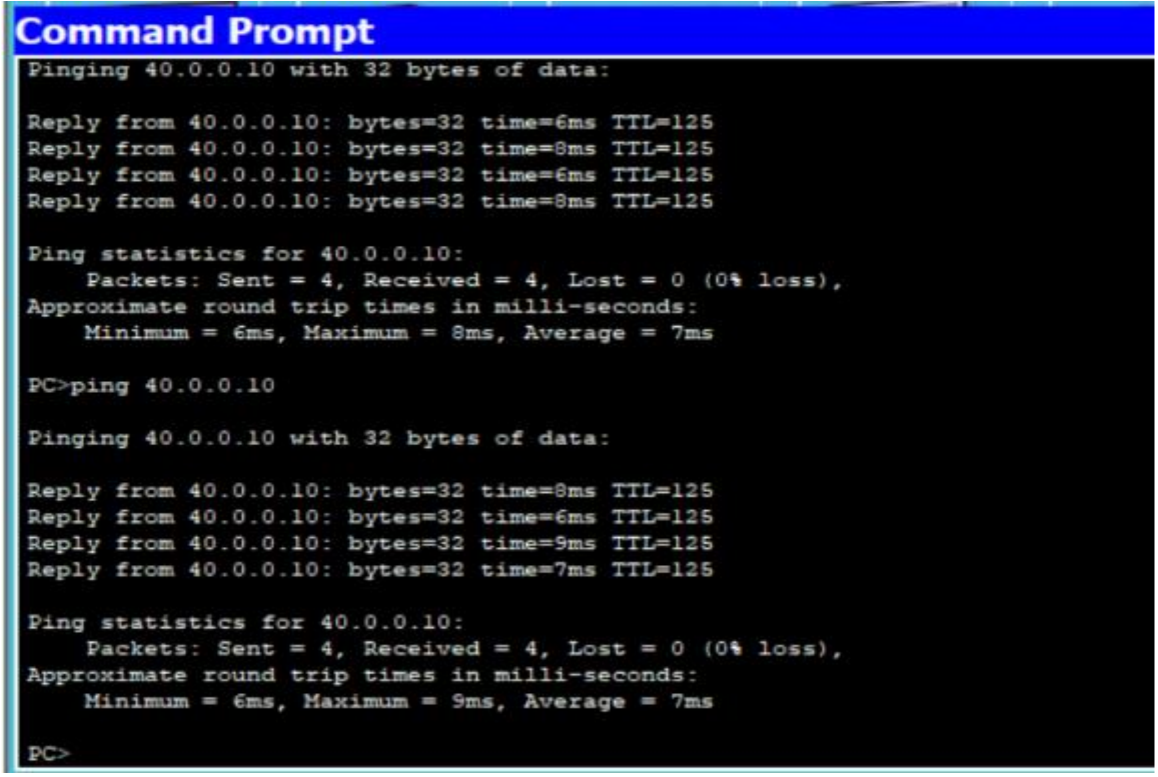
→ Destination host unreachable
Packets sent: 4 received: 0 lost: 4 loss: 100%

It is also observed that the end device PC1 was only pinged with Router R1 only.

ping 30.0.0.1 → successful.

Hence all though the routers were connected serially, the end devices were unable to ping each other.

23/10/24



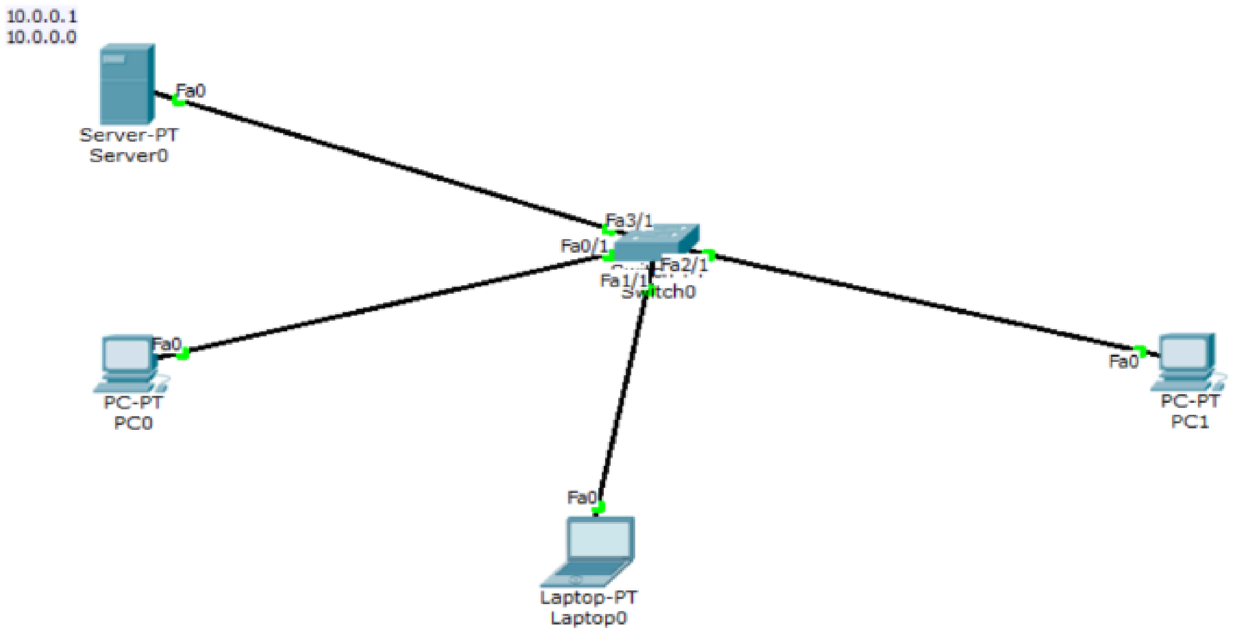
```
Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 8ms, Average = 7ms

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=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=9ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 9ms, Average = 7ms

PC>
```

Program 4:

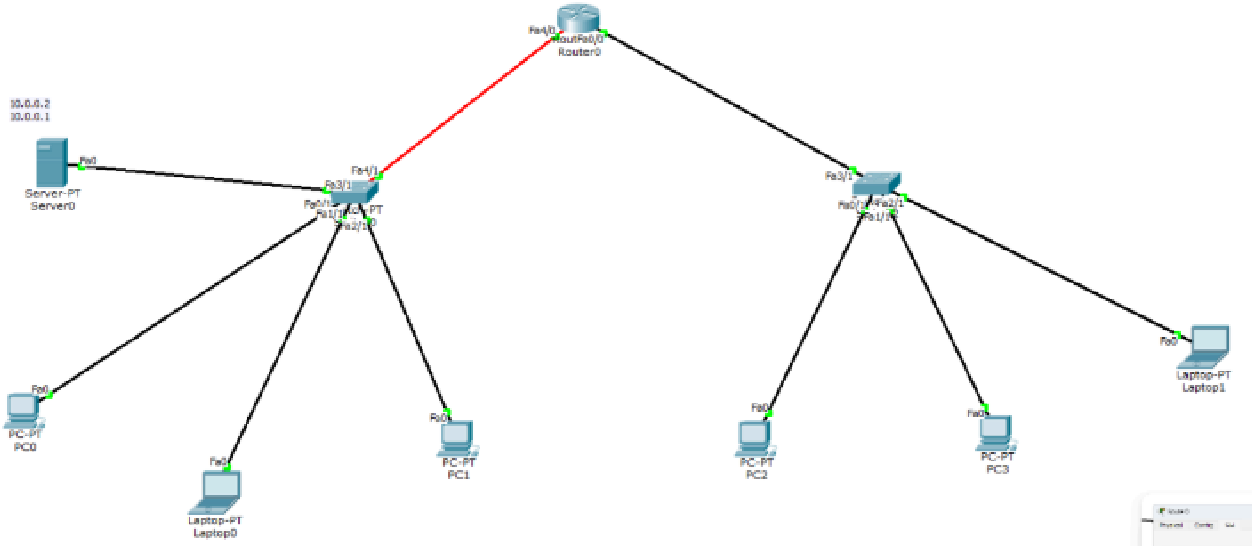
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

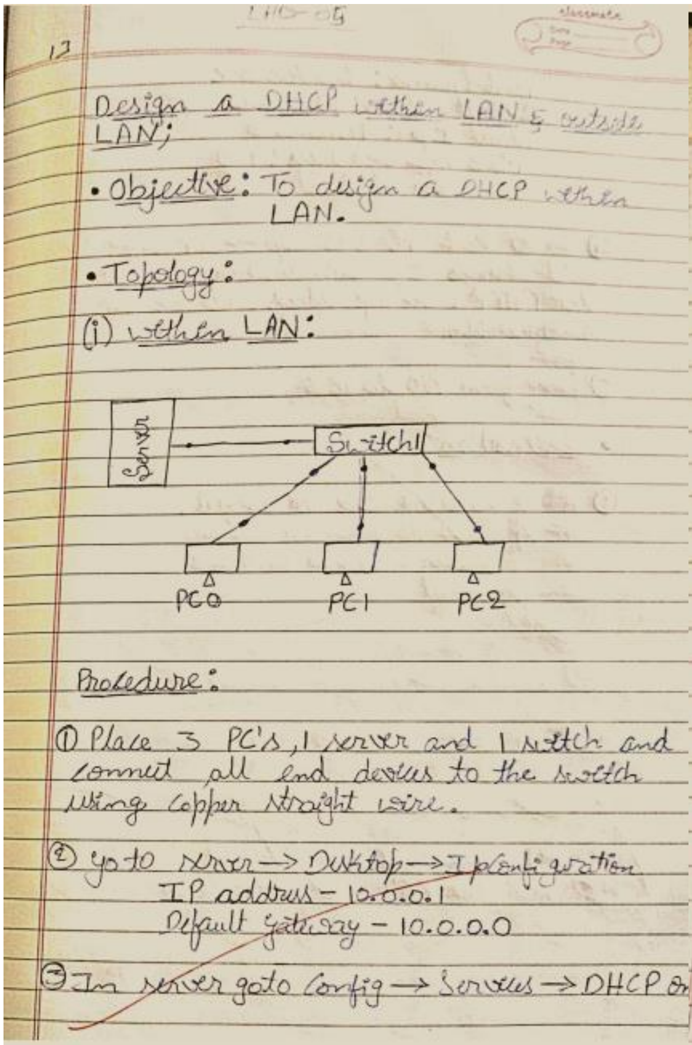
Within LAN



Outside LAN



Procedure and Observation:



Design a DHCP within LAN & outside LAN;

• Objective: To design a DHCP within LAN.

• Topology:

(i) within LAN:

Server

Switch1

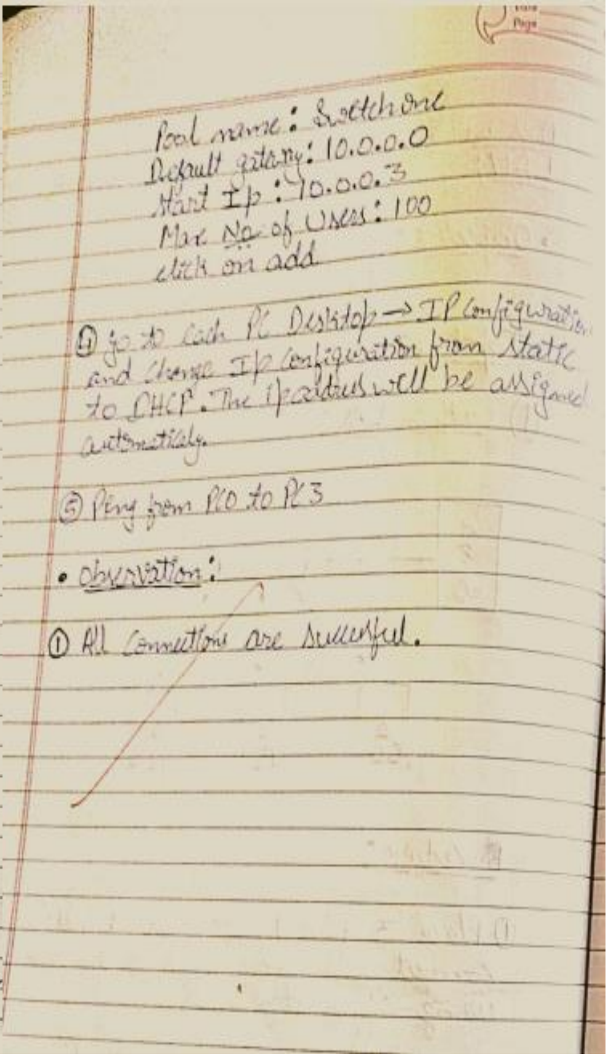
PC0 PC1 PC2

Procedure:

① Place 3 PC's, 1 server and 1 switch and connect all end devices to the switch using copper straight wire.

② go to server → Desktop → IP configuration
IP address - 10.0.0.1
Default gateway - 10.0.0.0

③ In server goto Config → Services → DHCP on



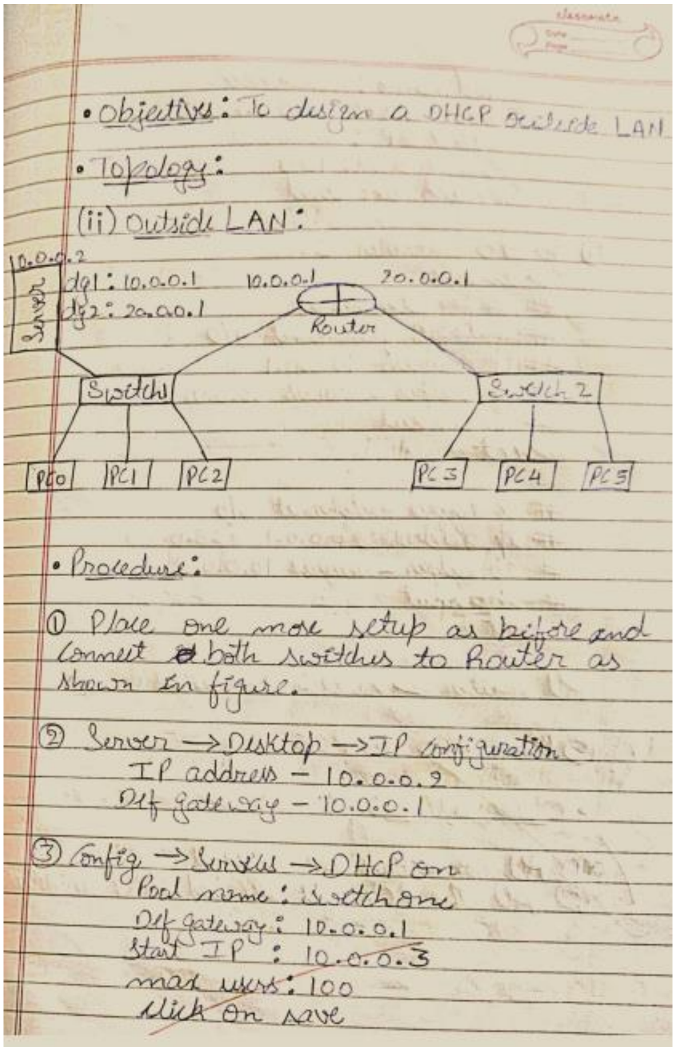
Pool name: Switch one
Default gateway: 10.0.0.0
Start Ip: 10.0.0.3
Max No of Users: 100
click on add

④ go to each PC Desktop → IP Configuration and change IP configuration from static to DHCP. The IP address will be assigned automatically.

⑤ Ping from PC0 to PC3

• Observation:

① All connections are successful.



- Objectives: To design a DHCP outside LAN

- Topology:

(ii) Outside LAN:

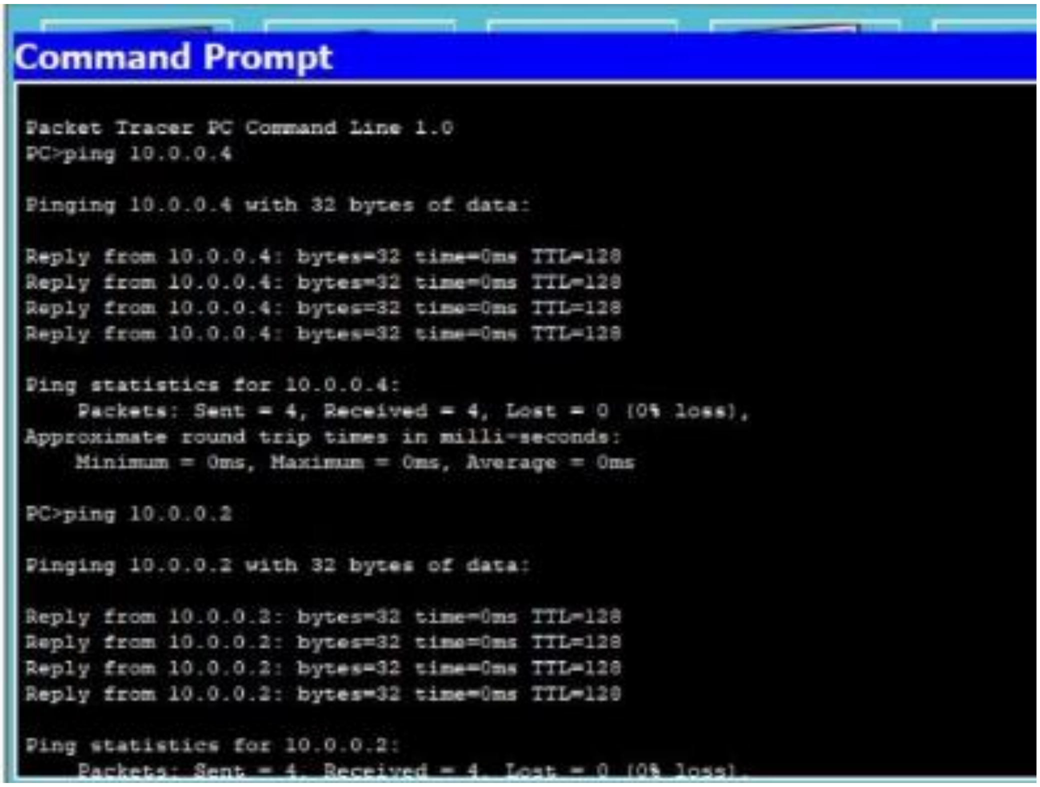
Server
10.0.0.2
dg1: 10.0.0.1
dg2: 20.0.0.1
10.0.0.1
20.0.0.1
Router
Switch
Switch 2
PC0
PC1
PC2
PC3
PC4
PC5

- Procedure:

① Place one more setup as before and connect both switches to Router as shown in figure.

② Server → Desktop → IP configuration
IP address – 10.0.0.2
Def gateway – 10.0.0.1

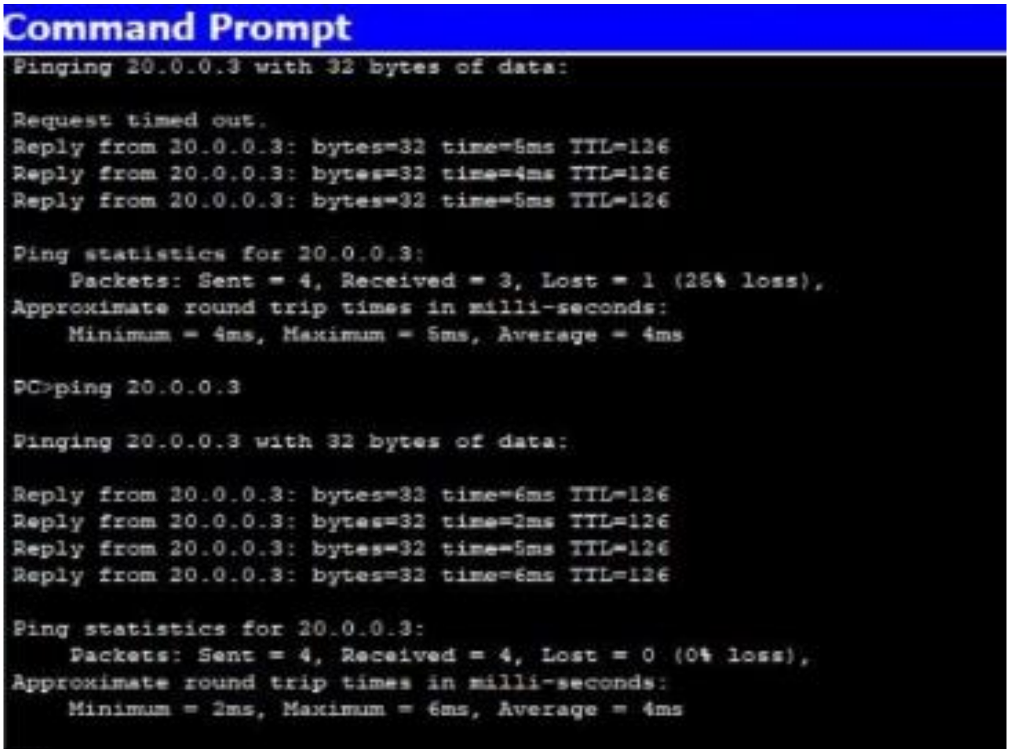
③ Config → Services → DHCP on
Pool name: switchOne
Def gateway: 10.0.0.1
Start IP: 10.0.0.3
max users: 100
click on save



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=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
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 = 0ms, Average = 0ms

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=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Within LAN



Command Prompt

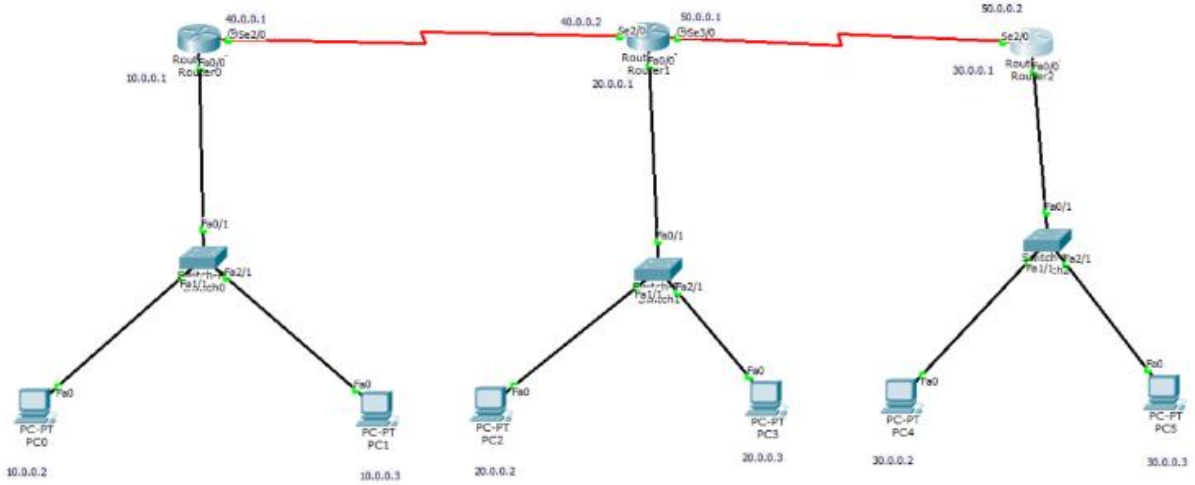
```
Pinging 20.0.0.3 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.3: bytes=32 time=5ms TTL=126
Reply from 20.0.0.3: bytes=32 time=4ms TTL=126
Reply from 20.0.0.3: bytes=32 time=5ms TTL=126

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 5ms, Average = 4ms

PC>ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data:

Reply from 20.0.0.3: bytes=32 time=6ms TTL=126
Reply from 20.0.0.3: bytes=32 time=2ms TTL=126
Reply from 20.0.0.3: bytes=32 time=5ms TTL=126
Reply from 20.0.0.3: bytes=32 time=6ms TTL=126

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 6ms, Average = 4ms
```

Outside LAN

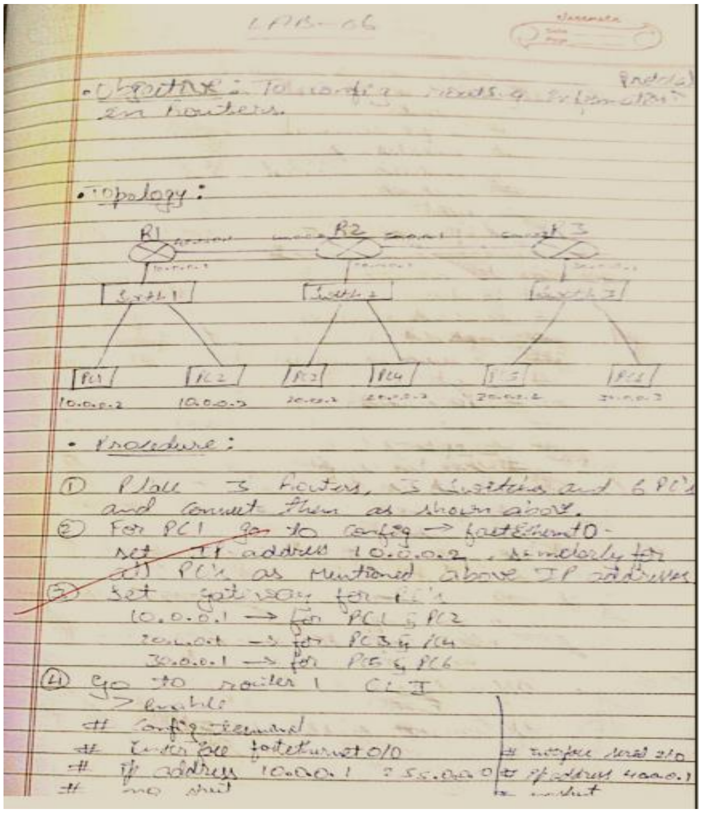
Program 5:

Aim: Configure RIP routing Protocol in Routers.

Topology:



Procedure and Observation:



LAB-06

• Objective: To config routing information (protocol) in routers.

• Topology:

R1
R2
R3
Switch1
Switch 2
Switch 3
PC1
PC2
PC3
PC4
PC5
PC6
10.0.0.2
10.0.0.3
20.0.0.2
20.0.0.3
30.0.0.2
30.0.0.3

• Procedure:

① Place 3 Routers, 3 Switches and 6 PC's and connect them as shown above.

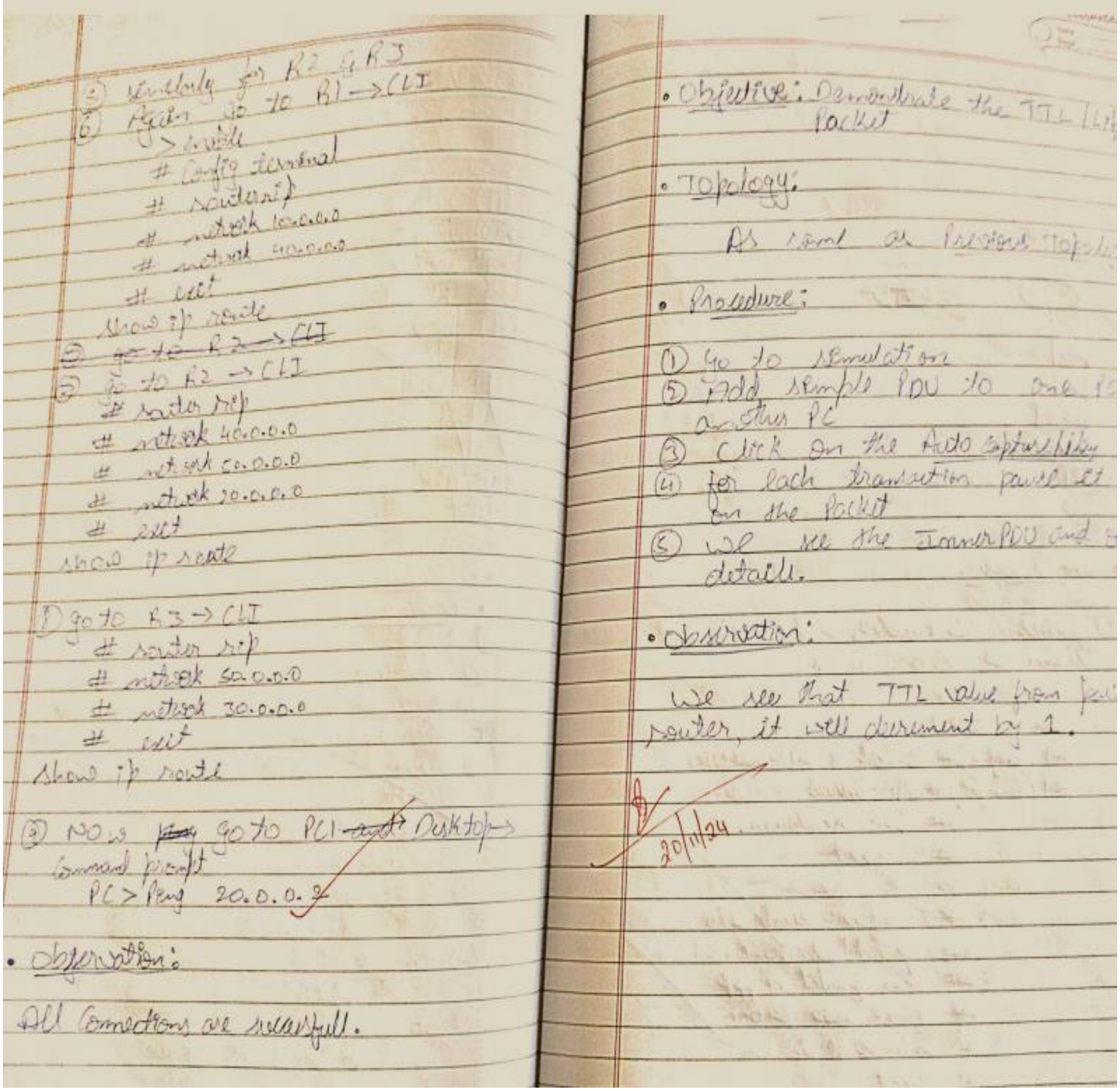
② For PC1 go to config → fastEthernet0 - set IP address 10.0.0.2, similarly for all PC's as mentioned above IP address.

③ Set gateway for PC's
10.0.0.1 → for PC1 & PC2
20.0.0.1 → for PC3 & PC4
30.0.0.1 → for PC5 & PC6

④ Go to router 1 CLI

```
>enable
# config terminal
# interface fastethernet 0/0
# ip address 10.0.0.1 255.0.0.0
# no shut
```

```
# interface serial 2/0
# ip address 40.0.0.1
# no shut
```



(5) similarly for R2 & R3

(6) Again go to R1 -> CLI

```
> enable
# config terminal
# router rip
# network 10.0.0.0
# network 40.0.0.0
# exit
show ip route
```

(5) ~~go to R2 -> CLI~~

(5) go to R2 -> CLI

```
# router rip
# network 40.0.0.0
# network 50.0.0.0
# network 20.0.0.0
# exit
show ip route
```

(7) go to R3 -> CLI

```
# router rip
# network 50.0.0.0
# network 30.0.0.0
# exit
show ip route
```

(8) Now ~~ping~~ go to PC1 ~~and~~ Desktop -> command prompt

```
PC > Ping 20.0.0.2
```

- Observation:

All Connections are successfull.

- Objective: Demonstrate the TTL (Lif
  Packet

- Topology:

As same as Previous Topolo

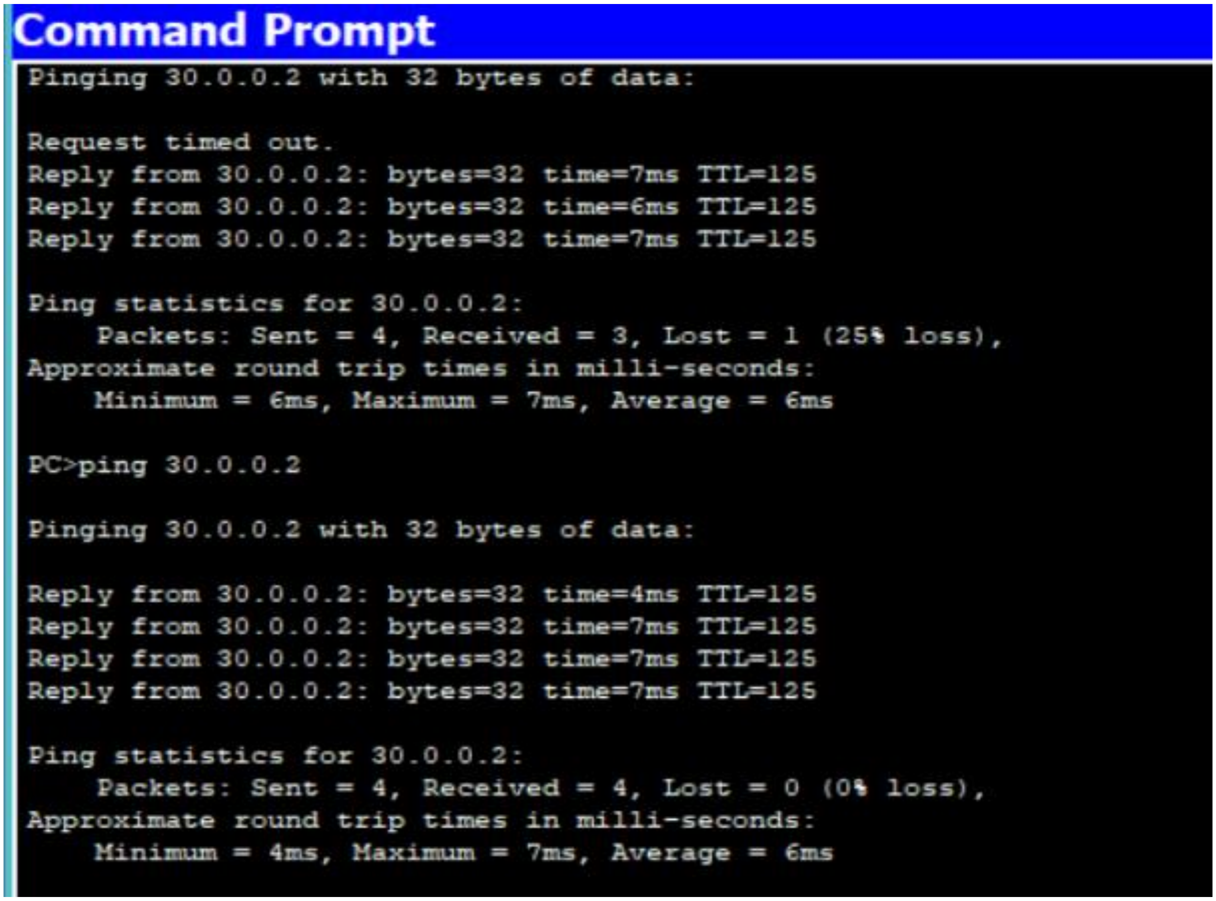
- Procedure:

(1) Go to simulation
(2) Add simple PDU to one P
another PC
(3) Click on the Auto capture/play
(4) for each transaction pause it
on the Packet
(5) We see the Inner PDU and
details.

- Observation:

We see that TTL value from fe
router, it will decrement by 1.

20/11/24.

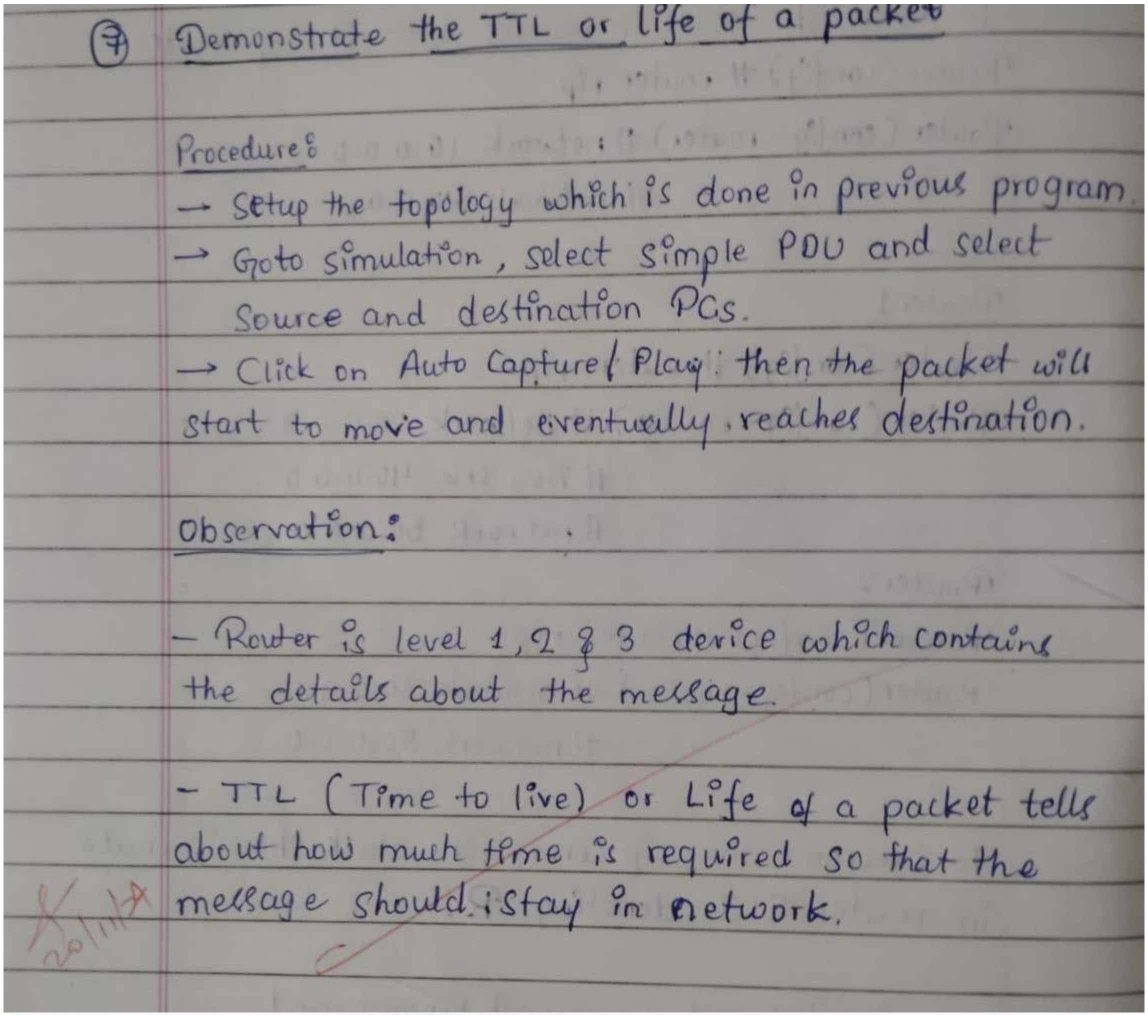


```
Pinging 30.0.0.2 with 32 bytes of data:

Request timed out.
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=6ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125

Ping statistics for 30.0.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 7ms, Average = 6ms

PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:

Reply from 30.0.0.2: bytes=32 time=4ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125

Ping statistics for 30.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 7ms, Average = 6ms
```

Program 6:

Aim: Demonstrate the TTL/ Life of a Packet.

Procedure and Observation:



(7) Demonstrate the TTL or life of a packet

Procedure:

→ Setup the topology which is done in previous program.

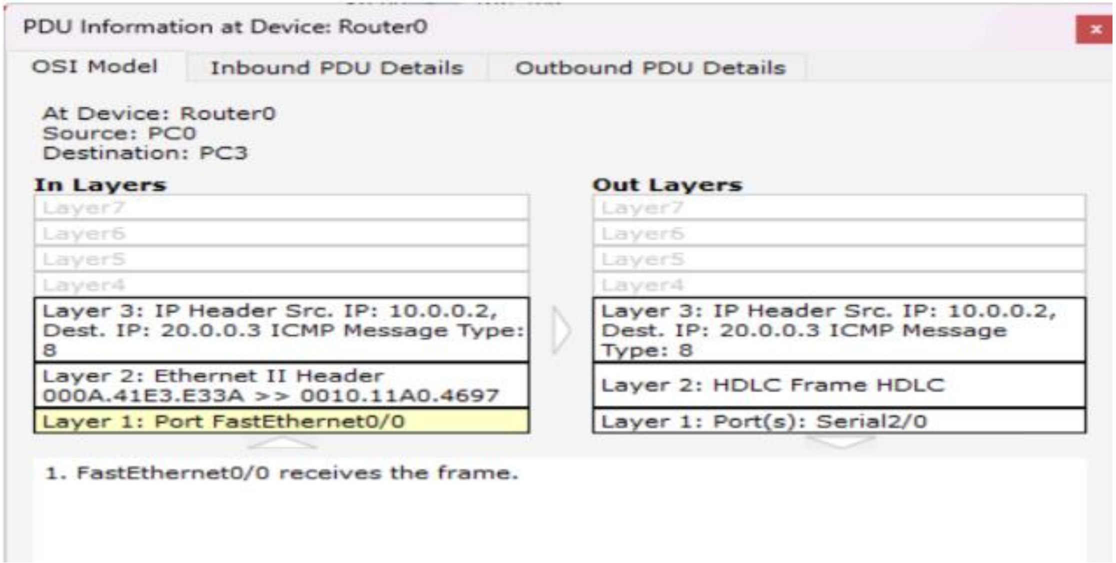
→ Goto simulation, select simple PDU and select Source and destination PCs.

→ Click on Auto Capture/Play: then the packet will start to move and eventually reaches destination.

Observation:

– Router is level 1, 2 & 3 device which contains the details about the message.

– TTL (Time to live) or Life of a packet tells about how much time is required so that the message should stay in network.

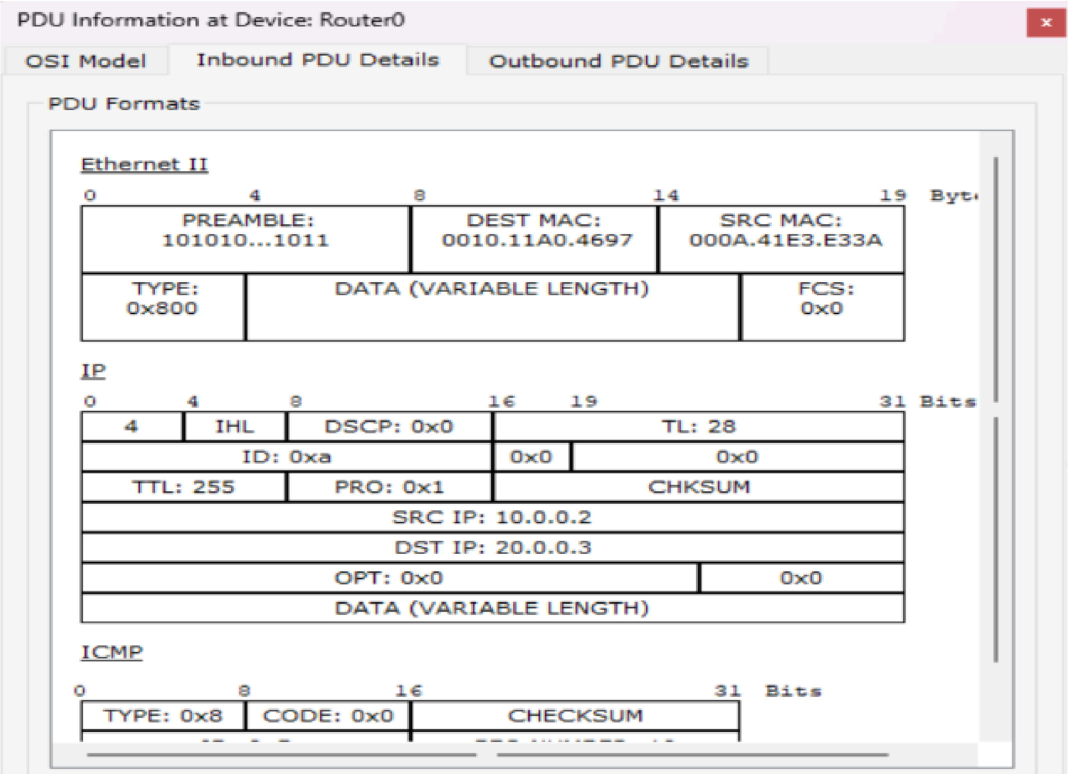


PDU Information at Device: Router0

At Device: Router0
Source: PC0
Destination: PC3

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer 3: IP Header Src. IP: 10.0.0.2, Dest. IP: 20.0.0.3 ICMP Message Type: 8	Layer 3: IP Header Src. IP: 10.0.0.2, Dest. IP: 20.0.0.3 ICMP Message Type: 8
Layer 2: Ethernet II Header 000A.41E3.E33A >> 0010.11A0.4697	Layer 2: HDLC Frame HDLC
Layer 1: Port FastEthernet0/0	Layer 1: Port(s): Serial2/0

1. FastEthernet0/0 receives the frame.



PDU Information at Device: Router0

PDU Formats

Ethernet II

0 4 8 14 19 Byt

PREAMBLE: 101010...1011	DEST MAC: 0010.11A0.4697	SRC MAC: 000A.41E3.E33A
TYPE: 0x800	DATA (VARIABLE LENGTH)	FCS: 0x0

IP

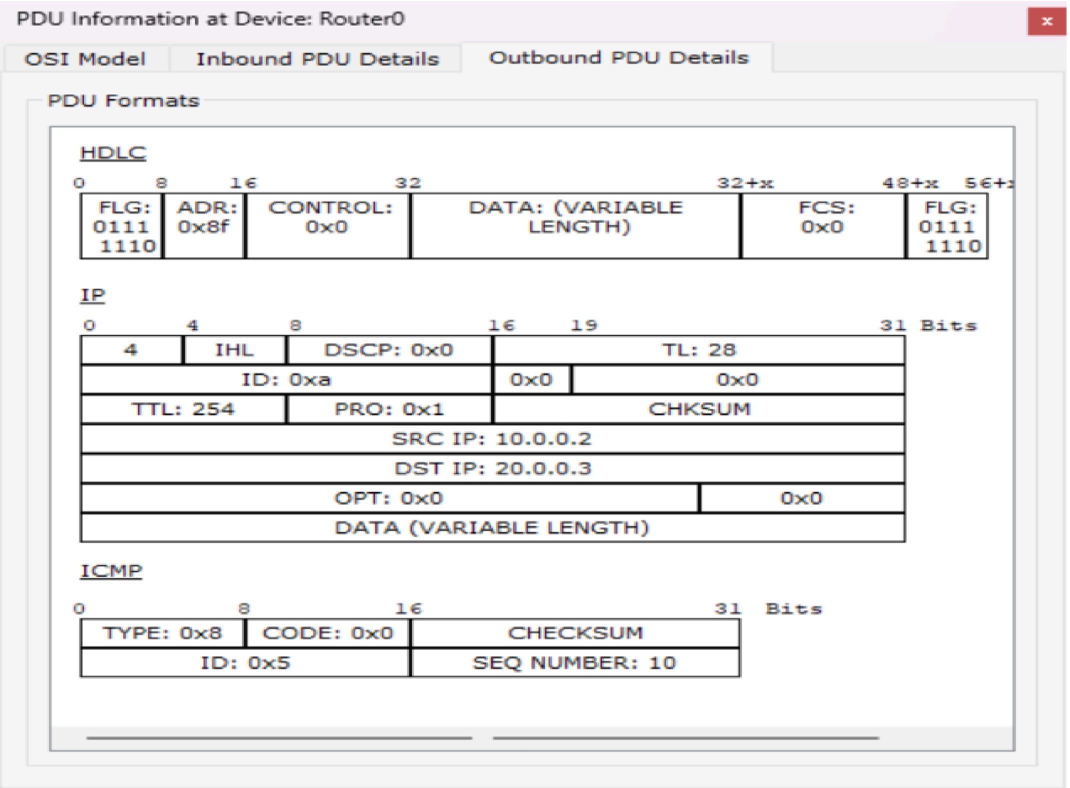
0 4 8 16 19 31 Bits

4	IHL	DSCP: 0x0	TL: 28
ID: 0xa		0x0	0x0
TTL: 255		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



PDU Information at Device: Router0

PDU Formats

HDLC

0 8 16 32 32+x 48+x 56+:

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

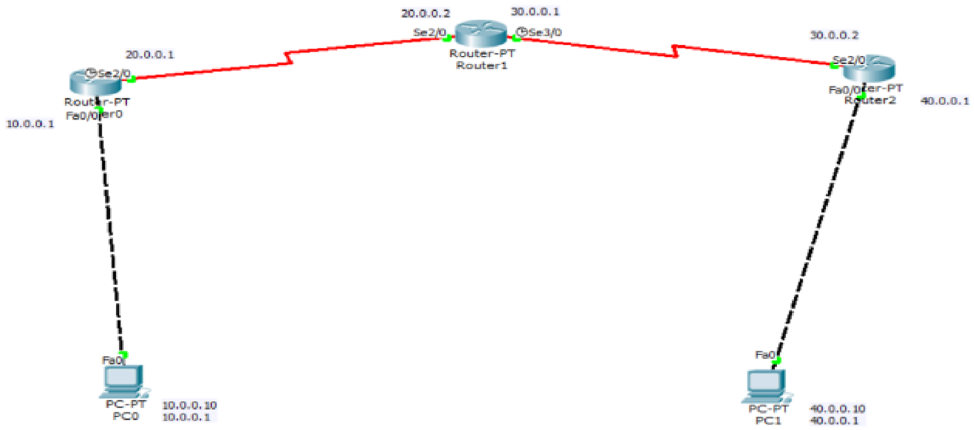
0 8 16 31 Bits

TYPE: 0x8	CODE: 0x0	CHECKSUM
ID: 0x5		SEQ NUMBER: 10

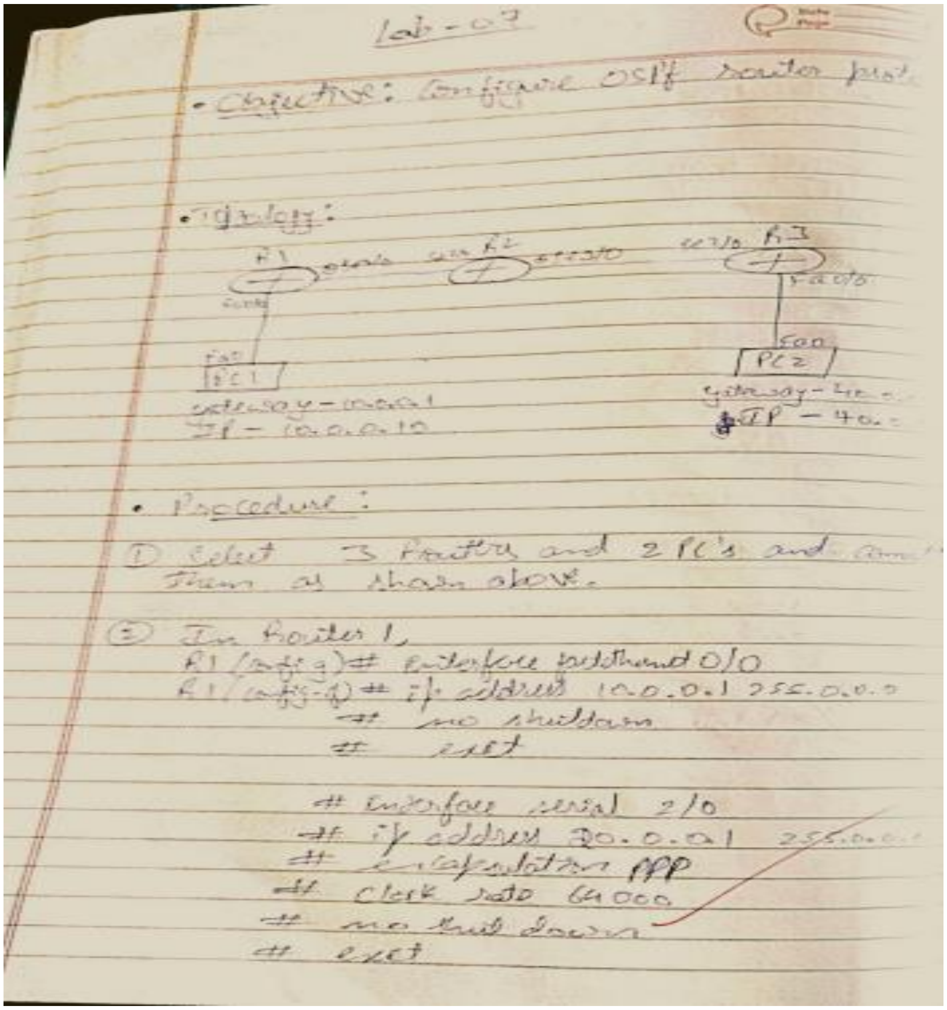
Program 7:

Aim: Configure OSPF routing protocol.

Topology:



Procedure and Observation:



Lab - 07

- Objective: Configure OSPF router prot

- Topology:

R1 Se2/0 Se2/0 R2 Se3/0 Se2/0 R3

Fa0/0 Fa0/0

Fa0 Fa0

PC1 PC2

gateway - 10.0.0.1 gateway - 40.0.0.1

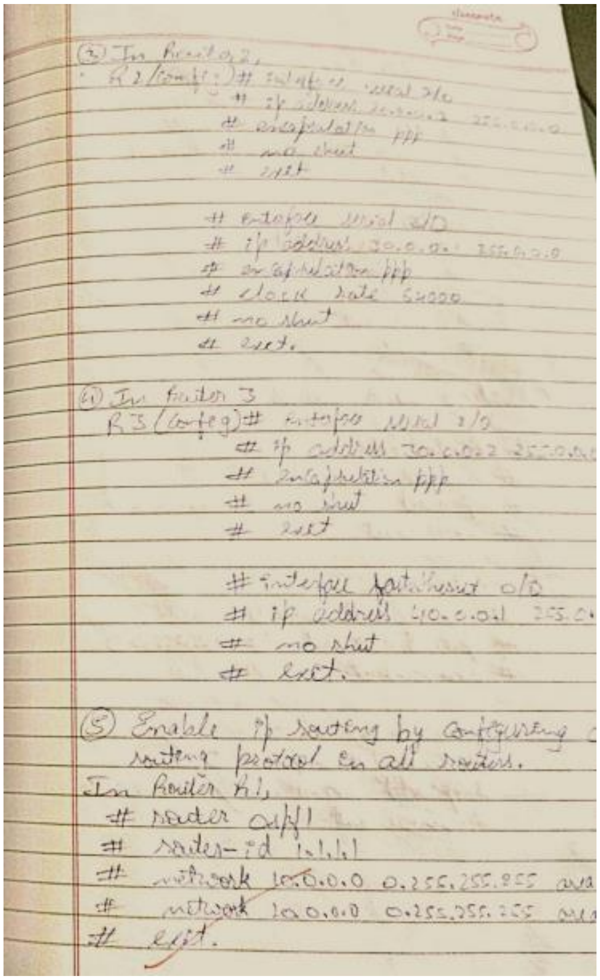
IP - 10.0.0.10 IP - 40.0.0.10

- Procedure :

① Select 3 Routers and 2 PC's and conn them as shown above.

② In Router 1,

```
R1 (config)# interface fastethernet 0/0
R1 (config-if)# ip address 10.0.0.1 255.0.0.0
        # no shutdown
        # exit

        # interface serial 2/0
        # ip address 20.0.0.1 255.0.0.0
        # encapsulation PPP
        # clock rate 64000
        # no shut down
        # exit
```



③ In Router2,

```
R2(config)# interface serial 2/0
           # ip address 20.0.0.2 255.0.0.0
           # encapsulation ppp
           # no shut
           # exit

           # interface serial 3/0
           # ip address 30.0.0.1 255.0.0.0
           # encapsulation ppp
           # clock rate 64000
           # no shut
           # exit.
```

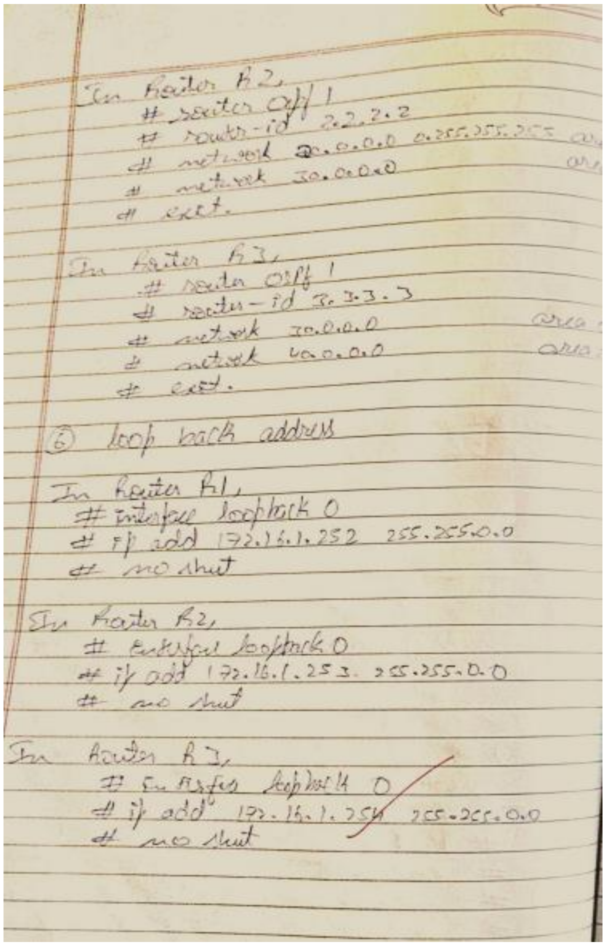
④ In Router 3

```
R3(config)# interface serial 2/0
           # ip address 30.0.0.2 255.0.0.
           # encapsulation ppp
           # no shut
           # exit

           # interface fastethernet 0/0
           # ip address 40.0.0.1 255.0.
           # no shut
           # exit.
```

⑤ Enable ip routing by configuring o
routing protocol on all routers.

In Router R1,

```
# router ospf 1
# router-id 1.1.1.1
# network 10.0.0.0 0.255.255.255 area
# network 20.0.0.0 0.255.255.255 are
# exit.
```



In Router R2,

```
# router ospf 1
# router-id 2.2.2.2
# network 20.0.0.0 0.255.255.255 ar
# network 30.0.0.0                 are
# exit.
```

In Router R3,

```
# router ospf 1
# router-id 3.3.3.3
# network 30.0.0.0                 area
# network 40.0.0.0                 area
# exit.
```

⑥ loop back address

In Router R1,

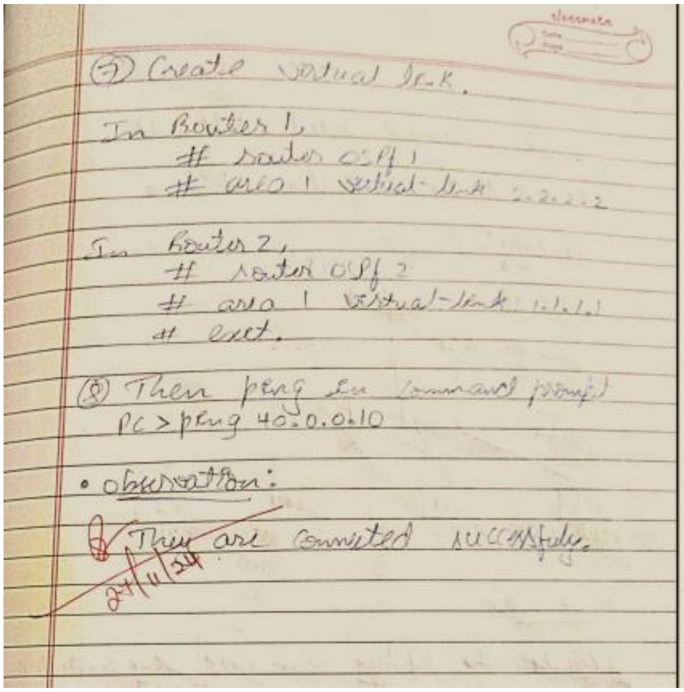
```
# interface loopback 0
# ip add 172.16.1.252 255.255.0.0
# no shut
```

In Router R2,

```
# interface loopback 0
# ip add 172.16.1.253 255.255.0.0
# no shut
```

In Router R3,

```
# interface loopback 0
# ip add 172.16.1.254 255.255.0.0
# no shut
```



(7) Create virtual link.

In Router 1,
# router ospf 1
# area 1 virtual-link 2.2.2.2

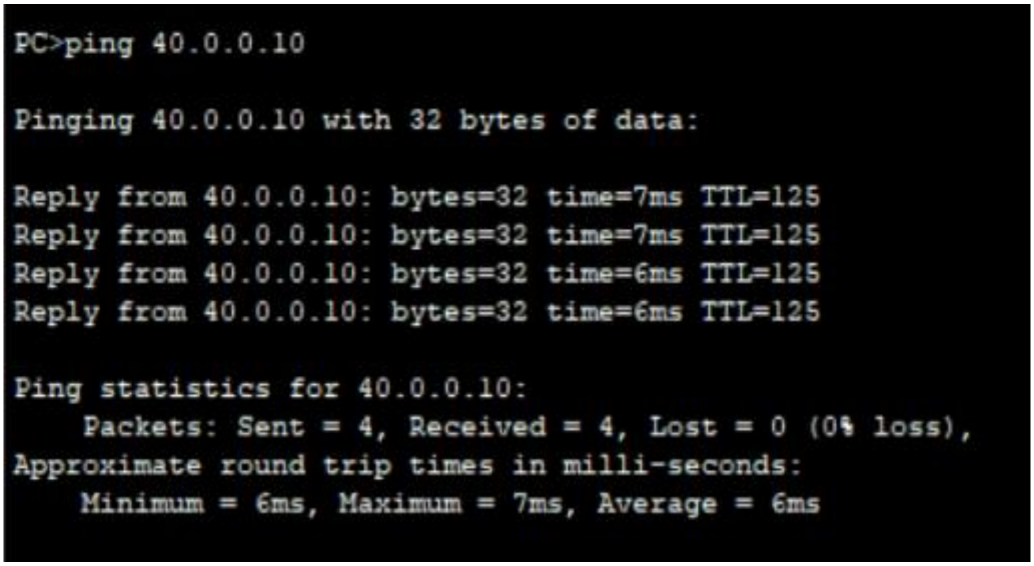
In Router 2,
# router ospf 2
# area 1 virtual-link 1.1.1.1
# exit.

(8) Then ping in command prompt
PC > ping 40.0.0.10

• observation:

They are connected successfuly.

27/11/24

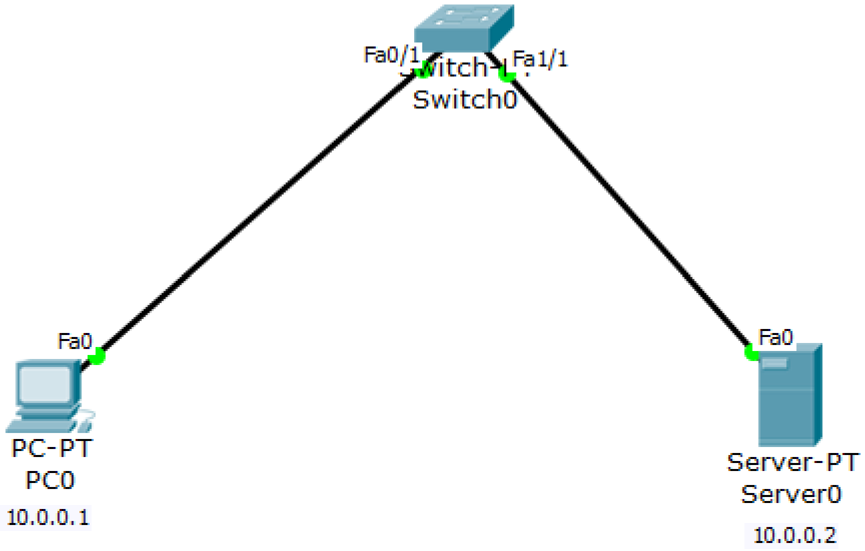


```
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 7ms, Average = 6ms
```

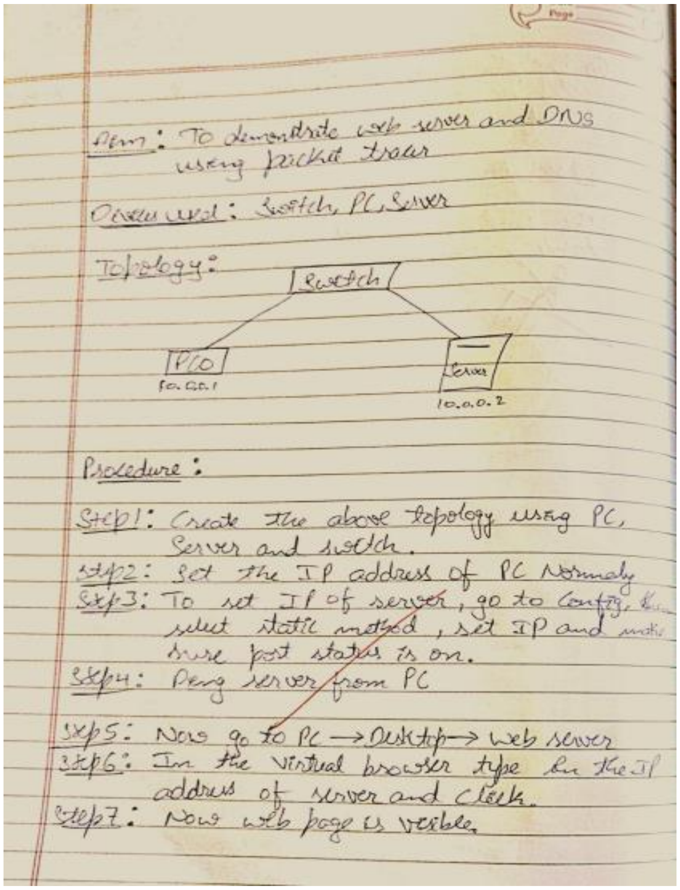
Program 8:

Aim: Configure Web Server, DNS within a LAN**.**

Topology:



Procedure and Observations:



Aim: To demonstrate web server and DNS using packet tracer

Devices used: Switch, PC, Server

Topology:

Switch
PC0
10.0.0.1
Server
10.0.0.2

Procedure:

Step1: Create the above topology using PC, Server and switch.

Step2: Set the IP address of PC Normally

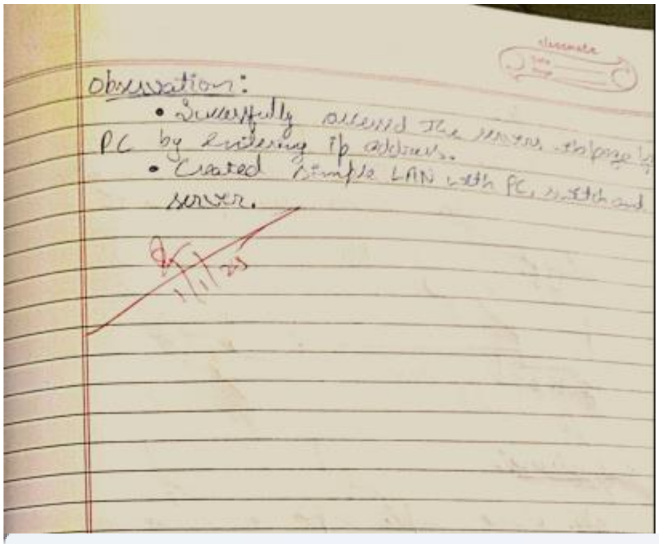
Step3: To set IP of server, go to Config, then select static method, set IP and make sure port status is on.

Step4: Ping server from PC

Step5: Now go to PC → Desktop → web server

Step6: In the virtual browser type in the IP address of server and click.

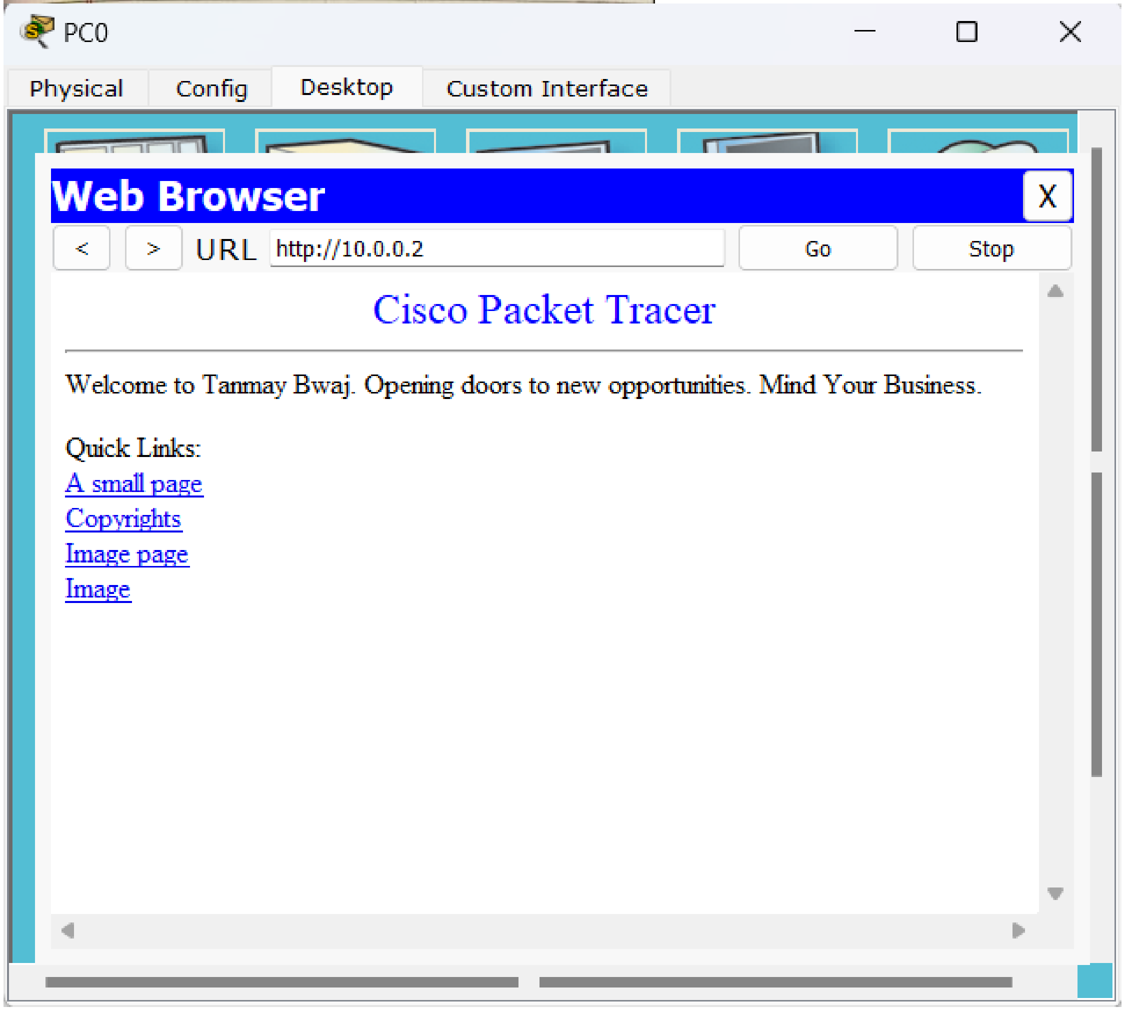
Step7: Now web page is visible.



Observation:

- Successfully accessed the server webpage by PC by entering ip address.
- Created simple LAN with PC, switch and server.

1/11/23



http://10.0.0.2

Cisco Packet Tracer

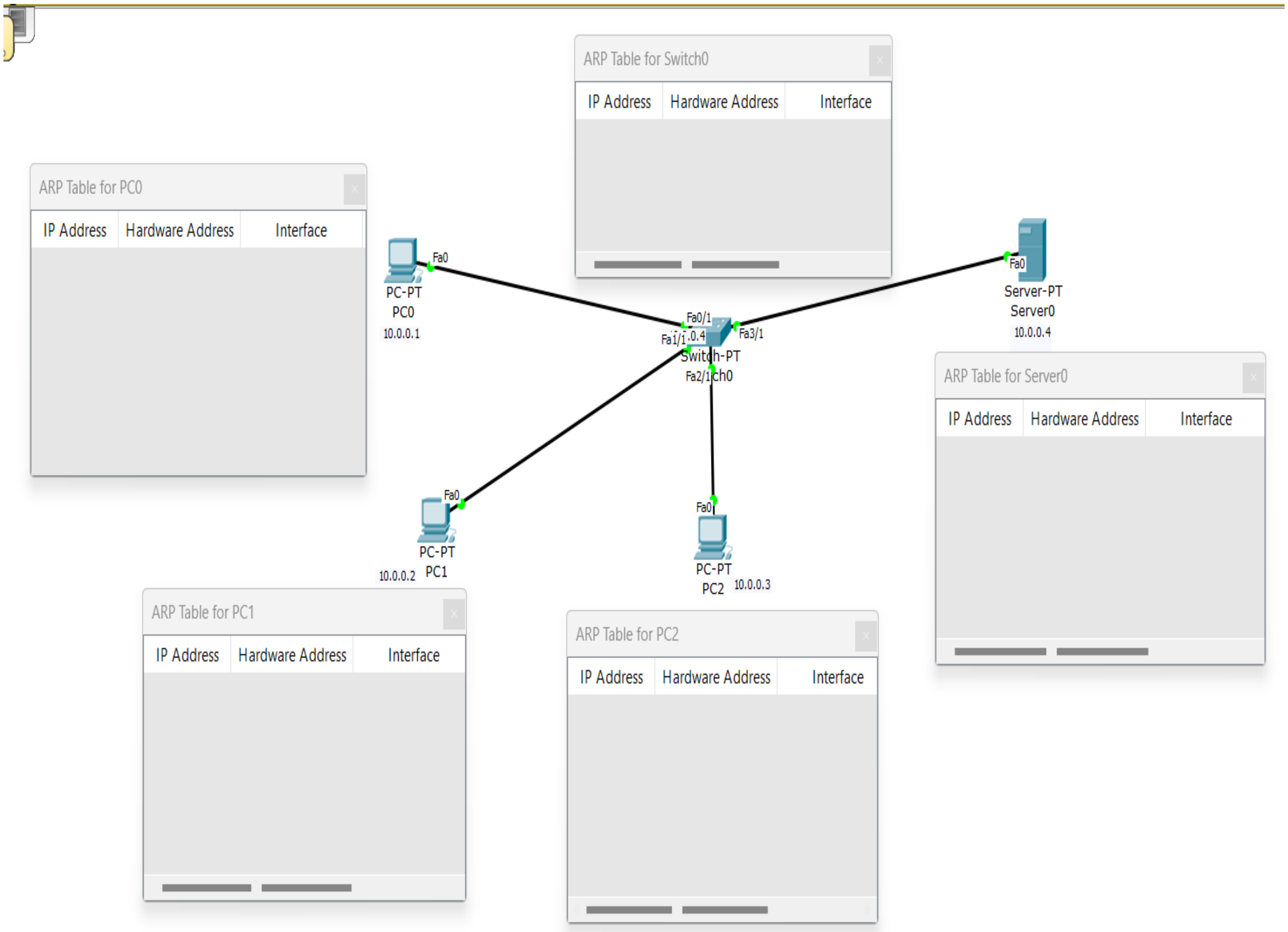
Welcome to Tanmay Bwaj. Opening doors to new opportunities. Mind Your Business.

Quick Links:
A small page
Copyrights
Image page
Image

rogram 9:

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

Topology:



ARP Table for Switch0

IP Address	Hardware Address	Interface

ARP Table for PC0

IP Address	Hardware Address	Interface

ARP Table for Server0

IP Address	Hardware Address	Interface

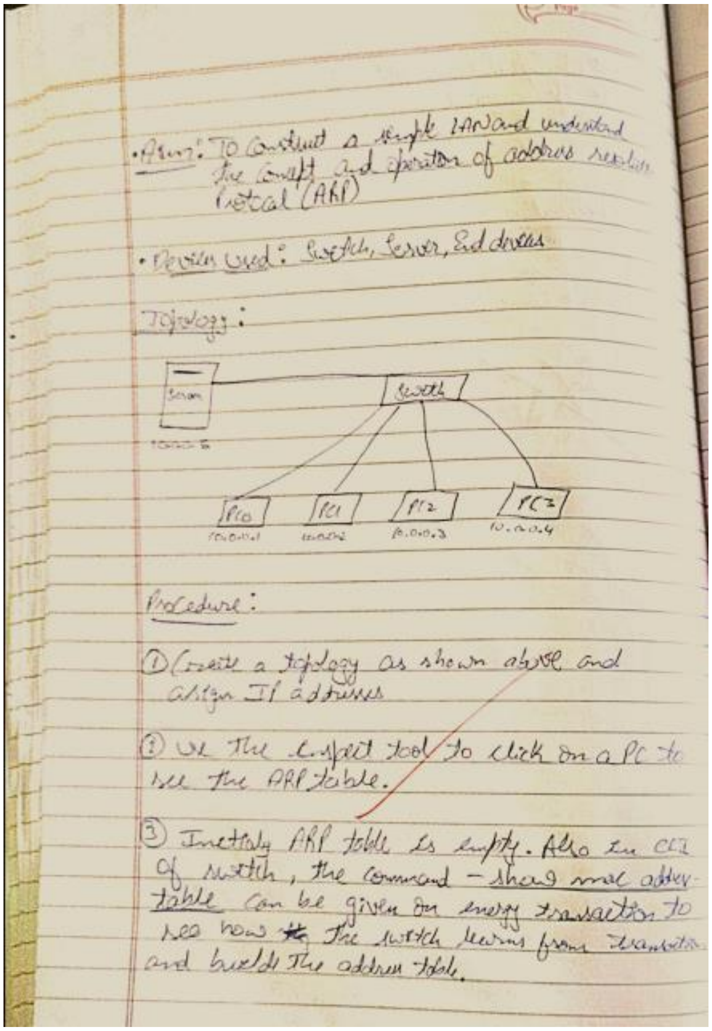
ARP Table for PC1

IP Address	Hardware Address	Interface

ARP Table for PC2

IP Address	Hardware Address	Interface

Procedure and Observations:



• Aim: To construct a simple LAN and understand the concept and operation of address resolution protocol (ARP)

• Devices Used: Switch, Server, End devices

Topology:

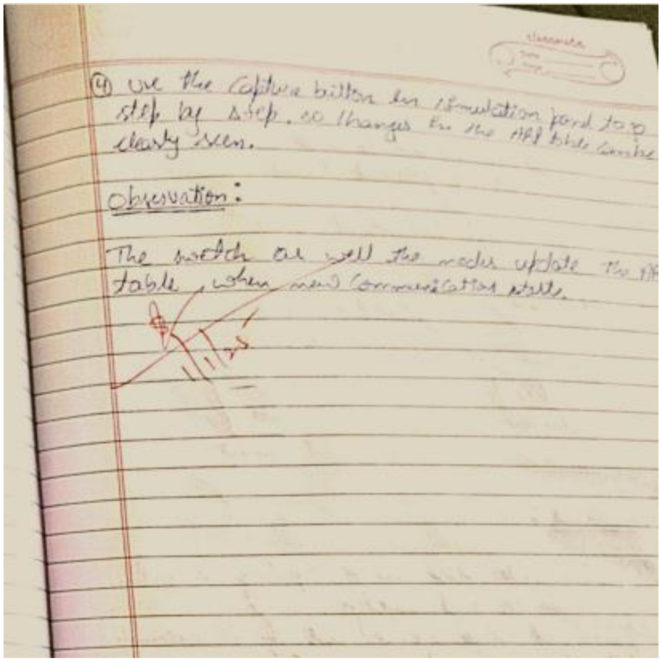
Server
10.0.0.5
Switch
PC0
10.0.0.1
PC1
10.0.0.2
PC2
10.0.0.3
PC3
10.0.0.4

Procedure:

① Create a topology as shown above and assign IP addresses

② Use the inspect tool to click on a PC to see the ARP table.

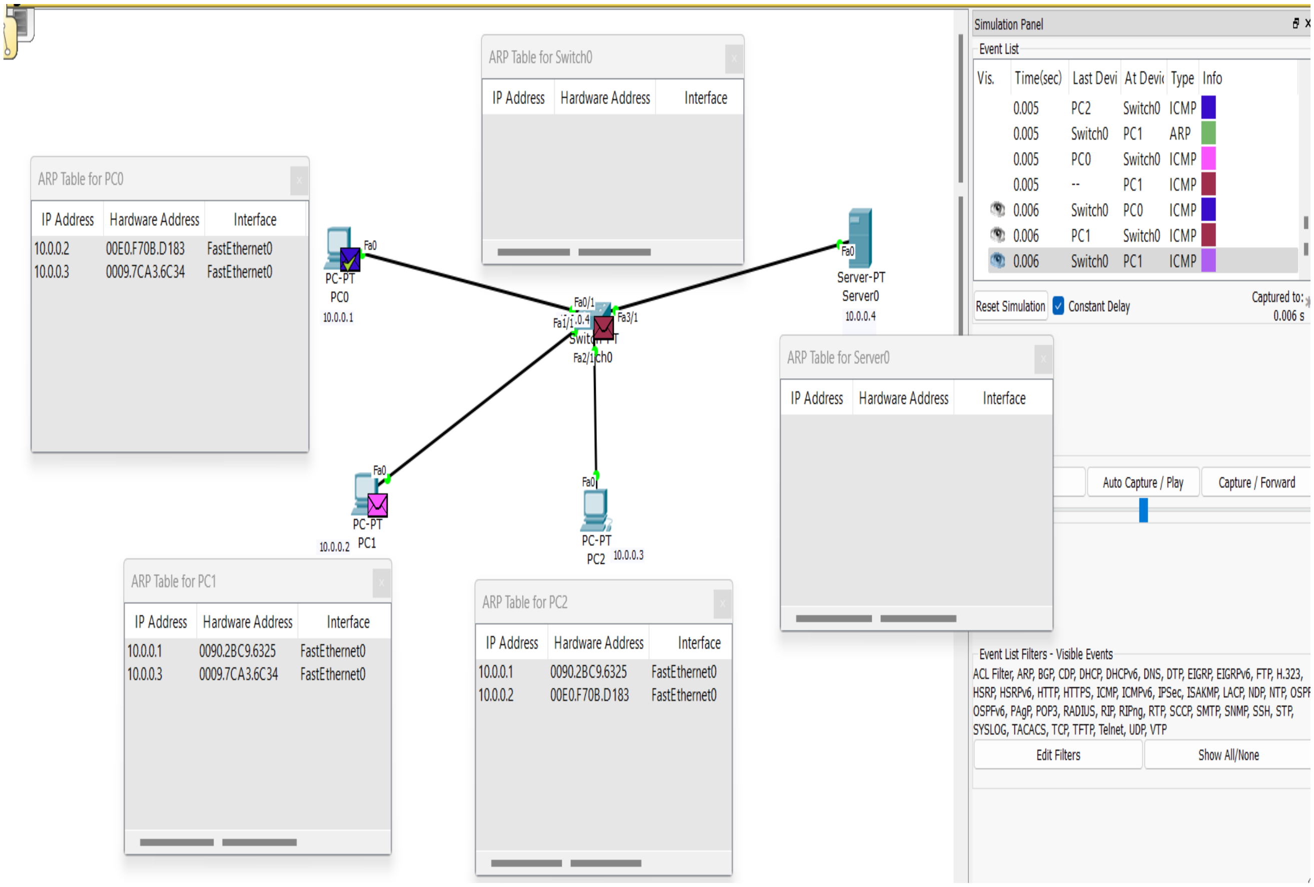
③ Initially ARP table is empty. Also in CLI of switch, the command - show mac address-table can be given on every transaction to see how the switch learns from transaction and builds the address table.



(4) Use the Capture button in Simulation panel to step by step. so changes in the ARP table can be clearly seen.

Observation:

The switch as well the nodes update the AR table, when new communication starts.



ARP Table for PC0

IP Address	Hardware Address	Interface
10.0.0.2	00E0.F70B.D183	FastEthernet0
10.0.0.3	0009.7CA3.6C34	FastEthernet0

ARP Table for Switch0

IP Address	Hardware Address	Interface

ARP Table for Server0

IP Address	Hardware Address	Interface

ARP Table for PC1

IP Address	Hardware Address	Interface
10.0.0.1	0090.2BC9.6325	FastEthernet0
10.0.0.3	0009.7CA3.6C34	FastEthernet0

ARP Table for PC2

IP Address	Hardware Address	Interface
10.0.0.1	0090.2BC9.6325	FastEthernet0
10.0.0.2	00E0.F70B.D183	FastEthernet0

Simulation Panel

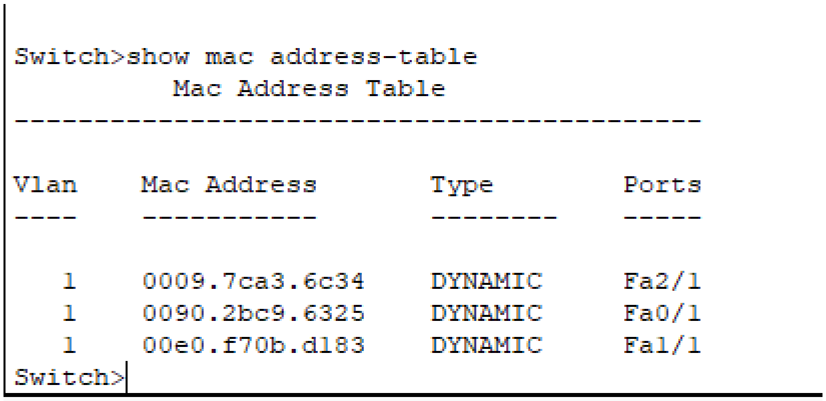
Event List

Vis.	Time(sec)	Last Devi	At Devic	Type	Info
	0.005	PC2	Switch0	ICMP	🟦
	0.005	Switch0	PC1	ARP	🟩
	0.005	PC0	Switch0	ICMP	🟪
	0.005	--	PC1	ICMP	🟥
	0.006	Switch0	PC0	ICMP	🟦
	0.006	PC1	Switch0	ICMP	🟥
	0.006	Switch0	PC1	ICMP	🟪

Reset Simulation ☑ Constant Delay Captured to: 0.006 s

Event List Filters - Visible Events

ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP

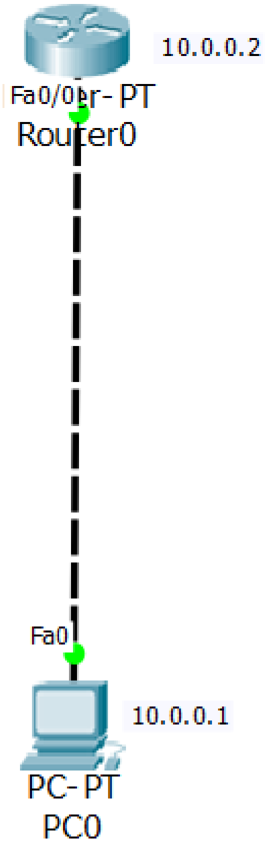


```
Switch>show mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    0009.7ca3.6c34    DYNAMIC     Fa2/1
   1    0090.2bc9.6325    DYNAMIC     Fa0/1
   1    00e0.f70b.d183    DYNAMIC     Fa1/1
Switch>
```

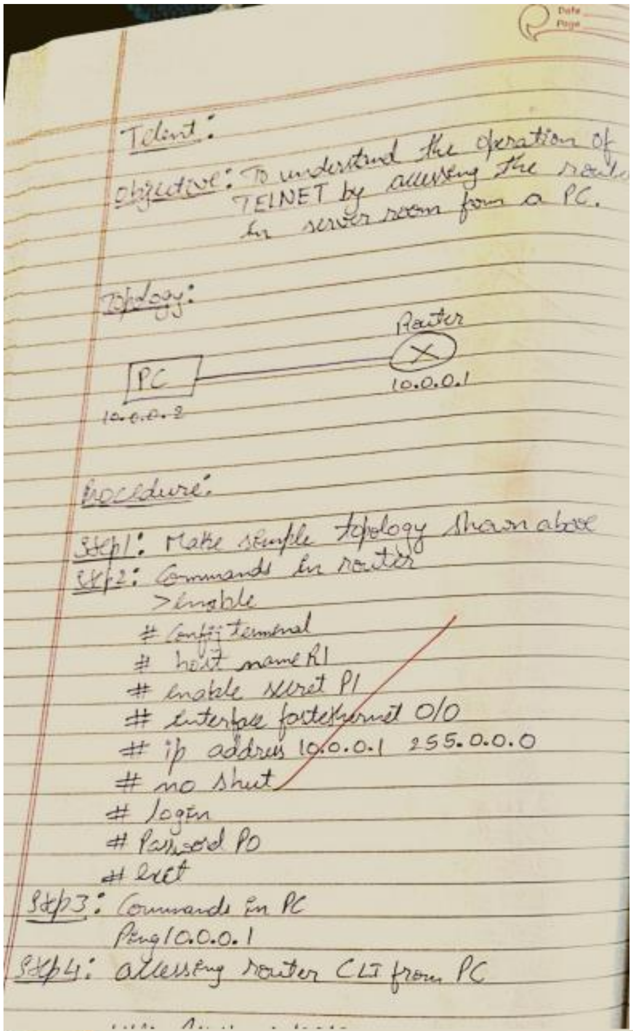
Program 10:

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 Observations:



Telnet:

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

Topology:

Router

PC

10.0.0.1

10.0.0.2

Procedure:

Step1: Make simple topology shown above

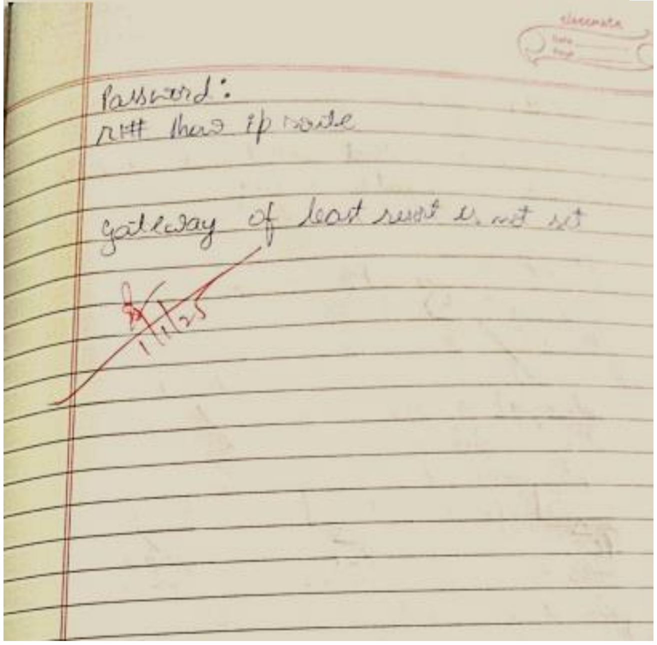
Step2: Commands in router

```
>enable
# config terminal
# host name R1
# enable secret P1
# interface fastethernet 0/0
# ip address 10.0.0.1 255.0.0.0
# no shut
# login
# Password P0
# exit
```

Step3: Commands in PC

```
Ping 10.0.0.1
```

Step4: accessing router CLI from PC

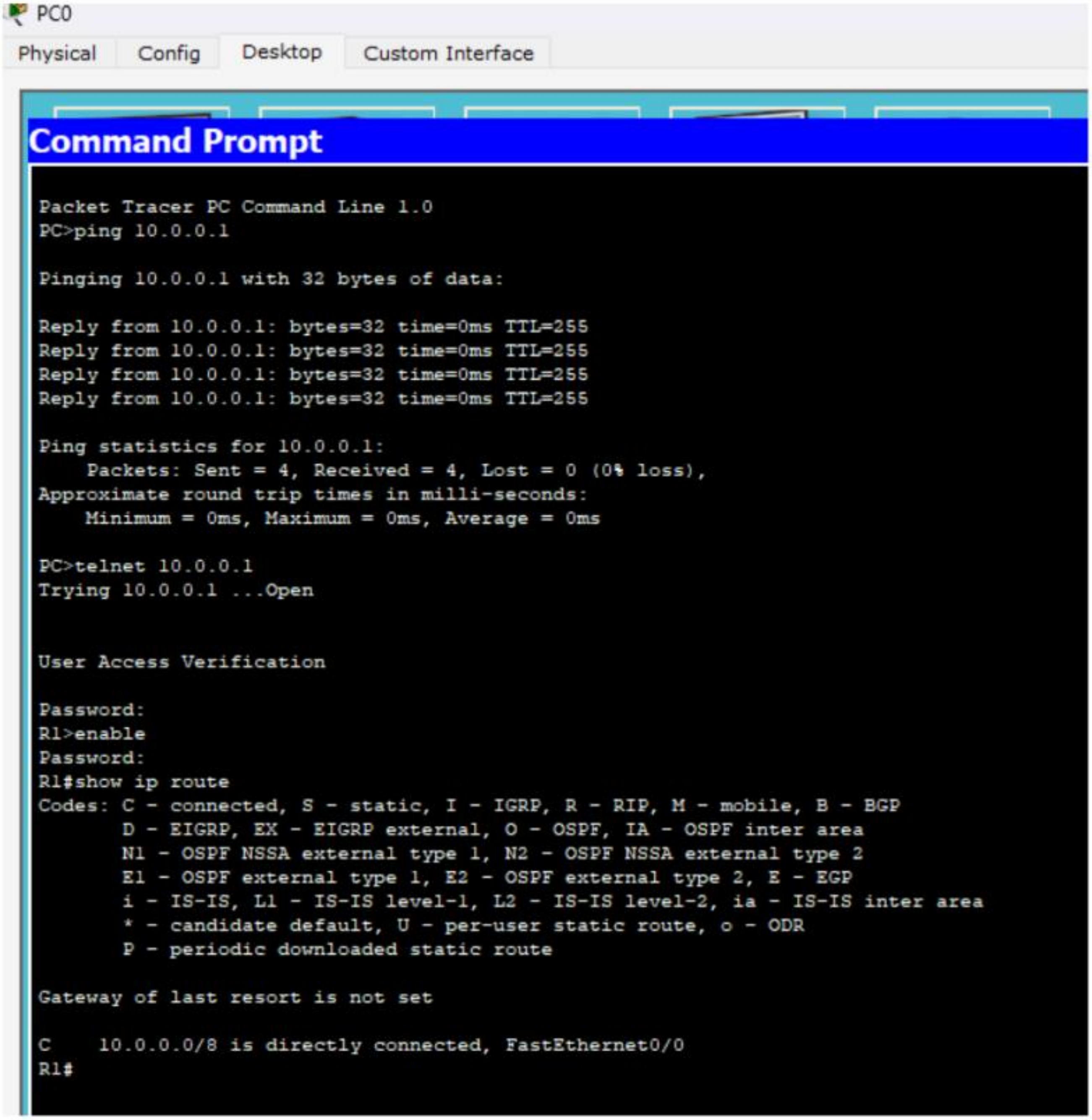


Password:

R1# show ip route

gateway of last resort is not set

1/11/25

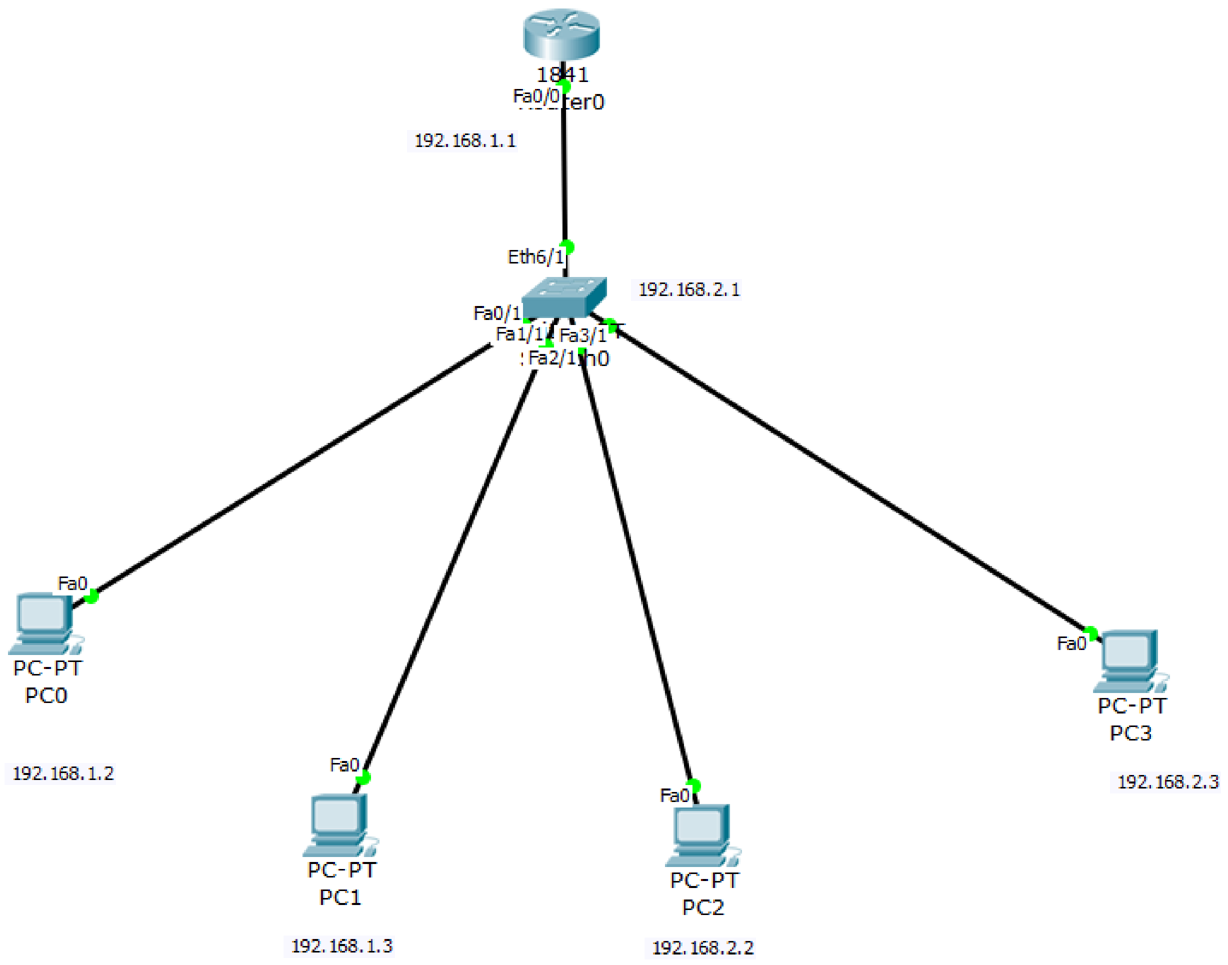


```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>telnet 10.0.0.1
Trying 10.0.0.1 ...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#
```

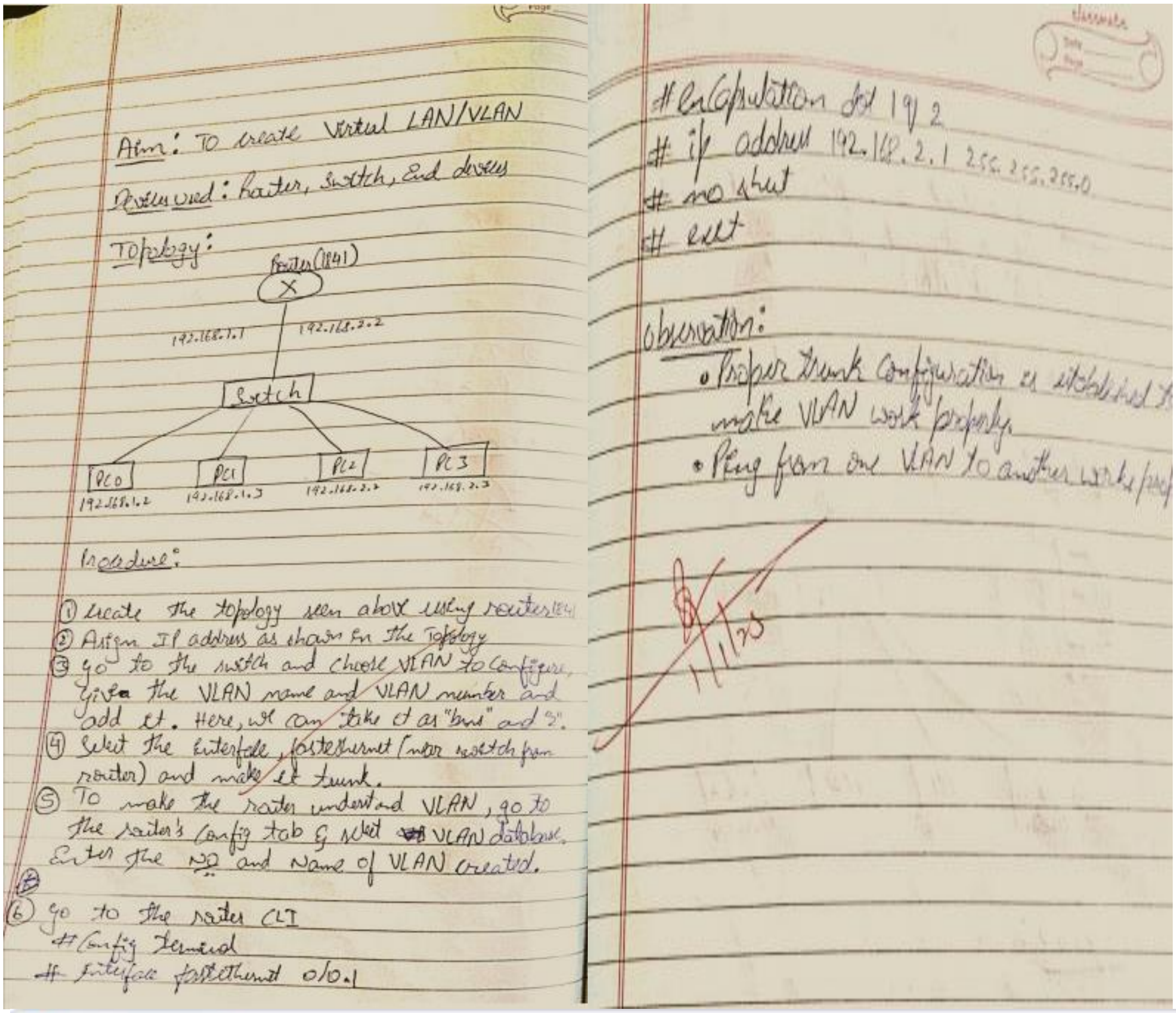
Program 11:

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

Topology:



Procedure and Observations:



Aim: To create Virtual LAN/VLAN

Devices used: Router, switch, End devices

Topology:

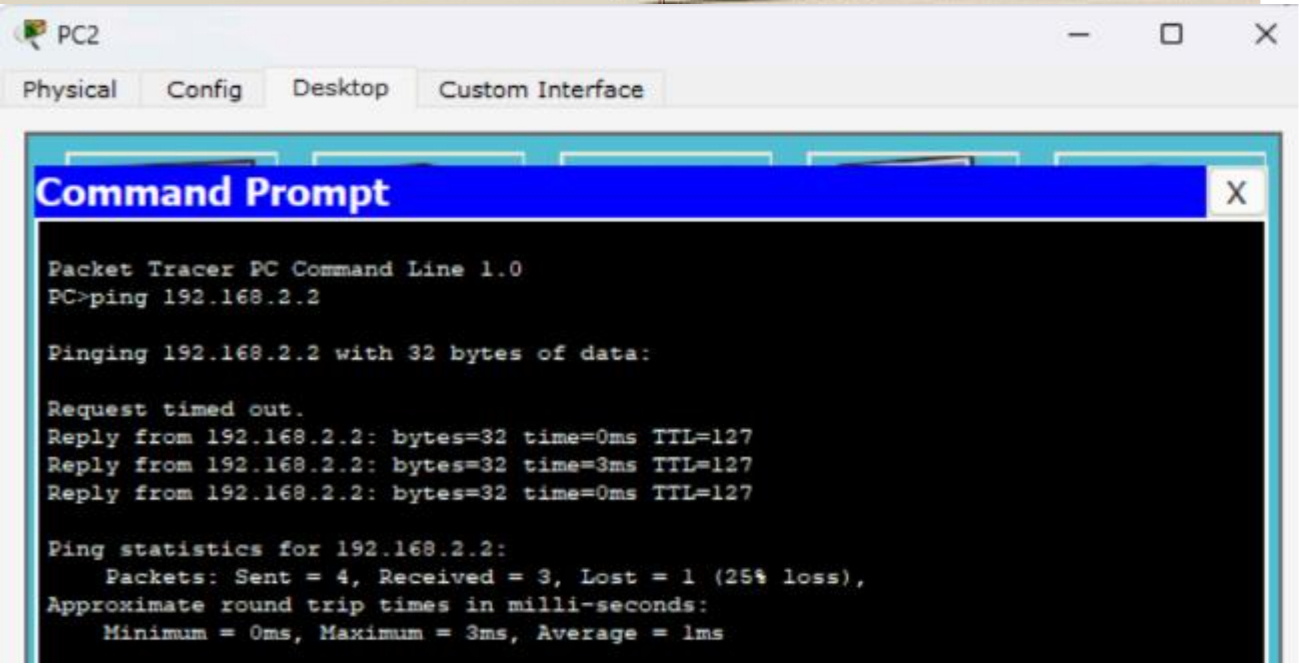
Router(1841)
192.168.1.1
192.168.2.2
Switch
PC0
192.168.1.2
PC1
192.168.1.3
PC2
192.168.2.2
PC3
192.168.2.3

Procedure:

① Create the topology seen above using router(184
② Assign IP address as shown in the Topology
③ go to the switch and choose VLAN to configure, give the VLAN name and VLAN number and add it. Here, we can take it as "brus" and 3".
④ Select the interface, fastethernet (near switch from router) and make it trunk.
⑤ To make the router understand VLAN, go to the router's config tab & select VLAN database. Enter the No and Name of VLAN created.
⑥ go to the router CLI
# Config terminal
# interface fastethernet 0/0.1
# encapsulation dot 1Q 2
# ip address 192.168.2.1 255.255.255.0
# no shut
# exit

Observation:
- Proper trunk configuration is established to make VLAN work properly.
- Ping from one VLAN to another works prop

1/11/25

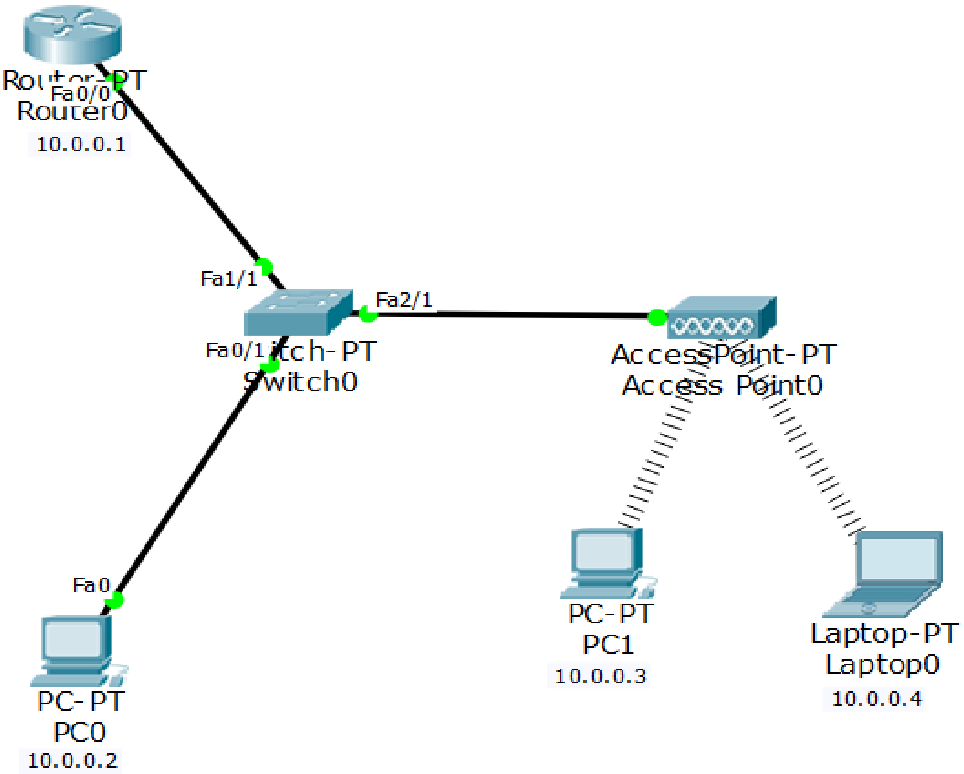


```
Packet Tracer PC Command Line 1.0
PC>ping 192.168.2.2

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=3ms 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 = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

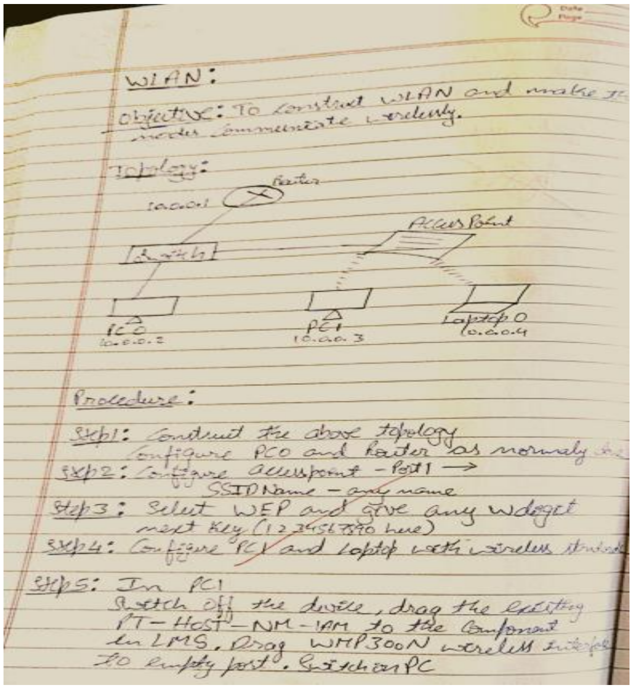
Program 12:

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

Topology:



Procedure and Observations:



WLAN:

Objective: To construct WLAN and make the nodes communicate wirelessly.

Topology:

Router
10.0.0.1
Switch
Access Point
PC0
10.0.0.2
PC1
10.0.0.3
Laptop0
10.0.0.4

Procedure:

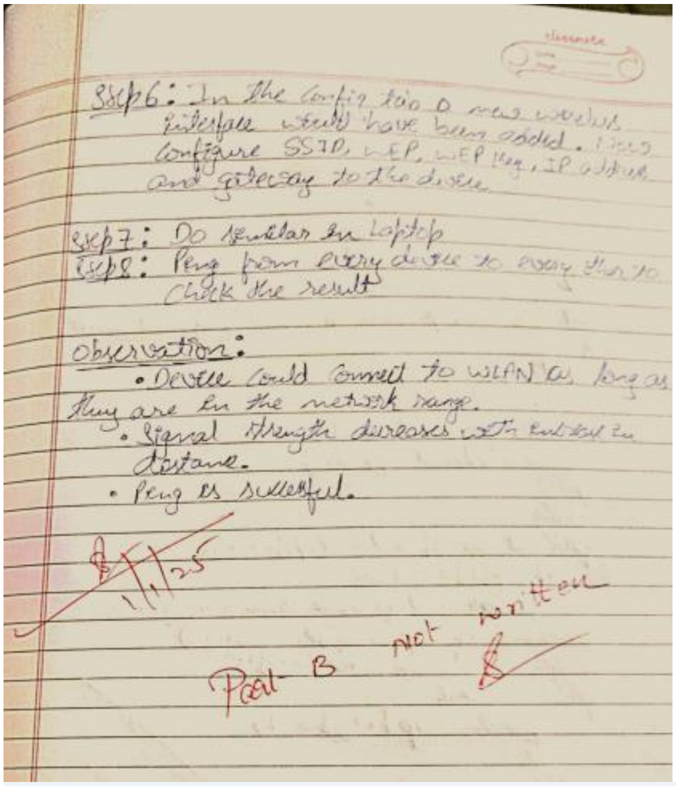
Step1: Construct the above topology
Configure PC0 and Router as normally

Step2: Configure accesspoint - Port1 →
SSID Name - any name

Step3: Select WEP and give any 10 digit
next key (1234567890 here)

Step4: Configure PC1 and Laptop with wireless standard

Step5: In PC1
Switch off the device, drag the existing
PT-HOST-NM-1AM to the Component
in LMS. Drag WMP300N wireless interface
to empty port. Switch on PC



Step6: In the config tab a new wireless interface would have been added. Now configure SSID, WEP, WEP key, IP address and gateway to the device.

Step7: Do similar in laptop

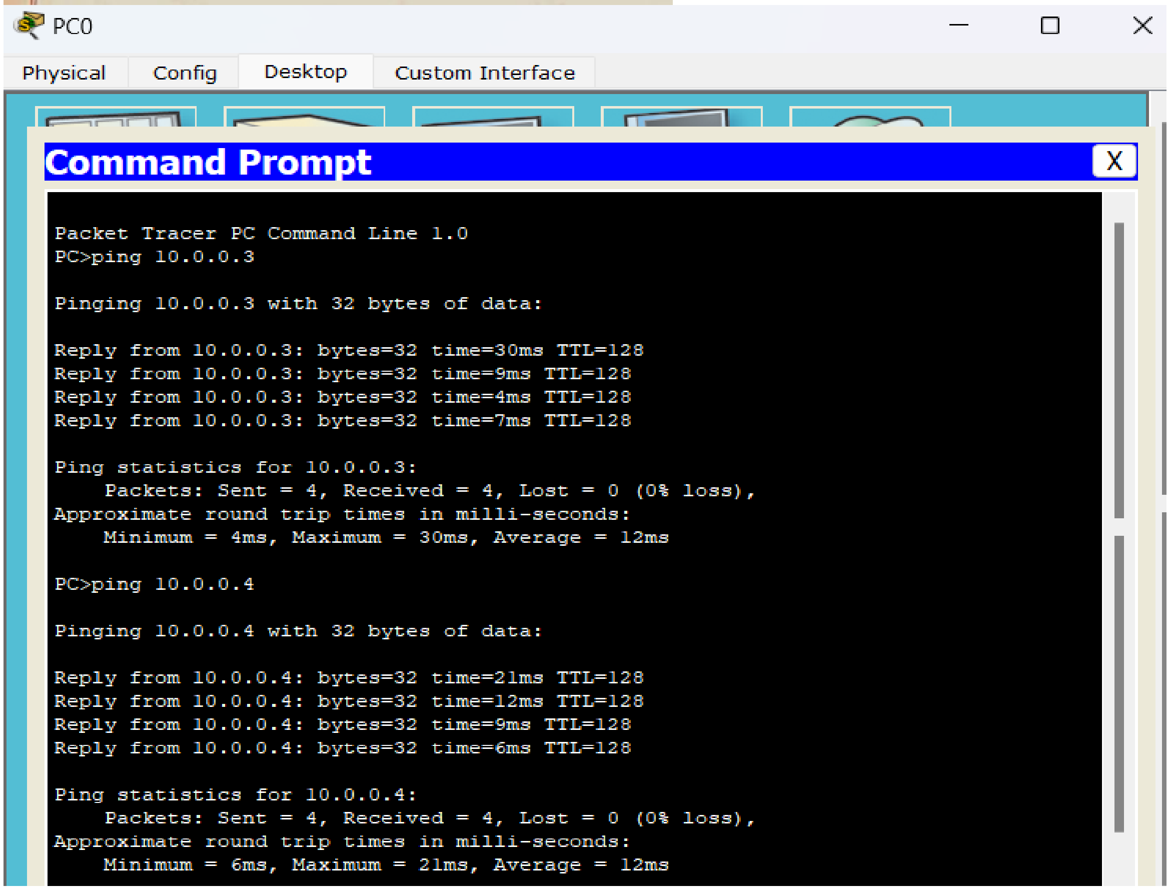
Step8: Ping from every device to every other to check the result

Observation:

- Device could connect to WLAN as long as they are in the network range.
- Signal strength decreases with increase in distance.
- Ping is successful.

1/1/25

Part B not written



```
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=30ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=4ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms 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 = 4ms, Maximum = 30ms, Average = 12ms

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=21ms TTL=128
Reply from 10.0.0.4: bytes=32 time=12ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms 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 = 6ms, Maximum = 21ms, Average = 12ms
```

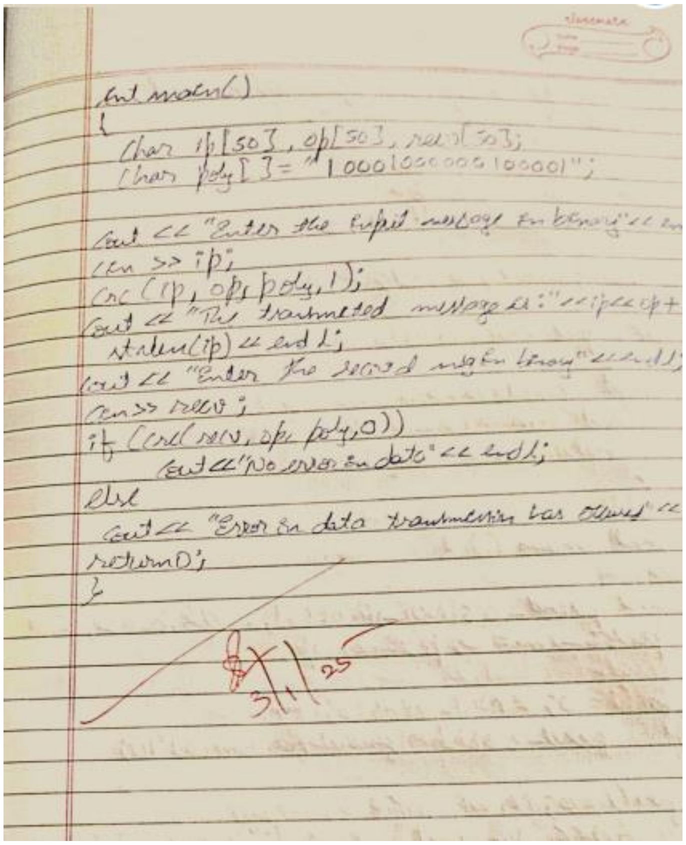
CYCLE - 2

Program 13:

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

```
#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];
    /* x 16 + x12 + x5 + 1 */
     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 received 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;
}
```

Observations:



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

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

3/1/25

Program 14:

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

Algorithm:

```
1. Start
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted.
      (Reject packets whosesize is greater than the bucket size.)
6. Stop
```

Code:

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

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 10
int rand(int a){
      int rn = (random() % 10) % a;
      return rn == 0 ? 1 : rn;
}
int main() {
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
    for(i = 0; i<NOF_PACKETS; ++i)
          packet_sz[i] = rand(6) * 10;
    for(i = 0; i<NOF_PACKETS; ++i)
         printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
    printf("\nEnter the Output rate:");
    scanf("%d", &o_rate);
    printf("Enter the Bucket Size:");
    scanf("%d", &b_size);
    for(i = 0; i<NOF_PACKETS; ++i){
         if( (packet_sz[i] + p_sz_rm) > b_size)
              if(packet_sz[i] > b_size)/*compare the packet size with bucket size*/
                   printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity
                                   (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
              else
                    printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
        else {
             p_sz_rm += packet_sz[i];
             printf("\n\nIncoming Packet size: %d", packet_sz[i]);
             printf("\nBytes remaining to Transmit: %d", p_sz_rm);
             p_time = rand(4) * 10;
```

```
           printf("\nTime left for transmission: %d units", p_time);
          for(clk = 10; clk <= p_time; clk += 10) {
              sleep(1);
               if(p_sz_rm) {
                    if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                     printf("\nPacket of size %d Transmitted", op);
                     printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
               }
               else {
                 printf("\nTime left for transmission: %d units", p_time-clk);
                 printf("\nNo packets to transmit!!");
}}}}
return 0;
}
```

OUTPUT:

```
packet[0]:30 bytes
packet[1]:10 bytes
packet[2]:10 bytes
packet[3]:50 bytes
packet[4]:30 bytes
packet[5]:50 bytes
packet[6]:10 bytes
packet[7]:20 bytes
packet[8]:30 bytes
packet[9]:10 bytes
Enter the Output rate:100
Enter the Bucket Size:50
Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 20 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 30 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!
Incoming Packet size: 10
Bytes remaining to Transmit: 10
```

```
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 50
Bytes remaining to Transmit: 50
Time left for transmission: 10 units
Packet of size 50 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 30 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 50
Bytes remaining to Transmit: 50
Time left for transmission: 20 units
Packet of size 50 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 20 units
Packet of size 20 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 20 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!
Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!
```

Program 15:

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.

Algorithm:

Client Side

```
1. Start.
2. Create a socket using the socket() system call.
3. Connect the socket to the server's address using the connect() system call.
4. Send the filename of the required file using the send() system call.
5. Read the contents of the file sent by the server using the recv() system call.
6. Stop.
```

Code:

```
#include <unistd.h>
int main()
{
    int soc, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    /* socket creates an endpoint for communication and returns a file descriptor */
    soc = socket(PF_INET, SOCK_STREAM, 0);
    /*
    * sockaddr_in is used for ip manipulation
    * we define the port and IP for the connection.
    */
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    /* keep trying to establish connection with server */
    while(connect(soc, (struct sockaddr *) &addr, sizeof(addr))) ;
        printf("\nClient is connected to Server");
    printf("\nEnter file name: ");
    scanf("%s", fname);
     /* send the filename to the server */
    send(soc, fname, sizeof(fname), 0);
     printf("\nRecieved response\n");0
     /* keep printing any data received from the server */
     while ((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
         printf("%s", buffer);
     return 0;
}
```

Algorithm:

Server Side

1. Start.
2. Create a socket using socket() system call.
3. Bind the socket to an address using bind() system call.
4. Listen to the connection using listen() system call.
5. accept connection using accept()
6. Receive filename and transfer contents of file with client.
7. Stop.

Code:

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
     int welcome, new_soc, fd, n;
     char buffer[1024], fname[50];
     struct sockaddr_in addr;
     welcome = socket(PF_INET, SOCK_STREAM, 0);
     addr.sin_family = AF_INET;
     addr.sin_port = htons(7891);
     addr.sin_addr.s_addr = inet_addr("127.0.0.1");
     bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
     printf("\nServer is Online");
     /* listen for connections from the socket */
     listen(welcome, 5);
     /* accept a connection, we get a file descriptor */
    new_soc = accept(welcome, NULL, NULL);
    /* receive the filename */
    recv(new_soc, fname, 50, 0);
    printf("\nRequesting for file: %s\n", fname);
    /* open the file and send its contents */
    fd = open(fname, O_RDONLY);
    if (fd < 0)
         send(new_soc, "\nFile not found\n", 15, 0);
    else
        while ((n = read(fd, buffer, sizeof(buffer))) > 0)
   send(new_soc, buffer, n, 0);
   printf("\nRequest sent\n");
   close(fd);
   return 0;
}
```

OUTPUT:

```
Server is Online.
Requesting for file : test.txt
Request sent.

Client is connected to server
Enter file name : test.txt
Received Response
Hello World.
```

Program 16:

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

Code:

```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Client";
    int listenfd, len;
    struct sockaddr_in servaddr, cliaddr;
    bzero(&servaddr, sizeof(servaddr));
    // Create a UDP Socket
    listenfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    // bind server address to socket descriptor
    bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    //receive the datagram
    len = sizeof(cliaddr);
    int n = recvfrom(listenfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len);
    //receive message from server
    buffer[n] = '\0';
    puts(buffer);
    // send the response
    sendto(listenfd, message, MAXLINE, 0,(struct sockaddr*)&cliaddr, sizeof(cliaddr));
}


// udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
```

```
#include<unistd.h>
#include<stdlib.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Server";
    int sockfd, n;
    struct sockaddr_in servaddr;
    // clear servaddr
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    // create datagram socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    // connect to server
   if(connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
       printf("\n Error : Connect Failed \n");
       exit(0);
    }
    // request to send datagram
    // no need to specify server address in sendto
    // connect stores the peers IP and port
    sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)NULL, sizeof(servaddr));
    // waiting for response
    recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)NULL, NULL);
    puts(buffer);
    // close the descriptor
    close(sockfd);
}
```

Output:

```
//Server output
Server is Online.
Hello Server

//Client Output
Hello Client
```