VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (22CS4PCCON)

Submitted by

TANMAYI S BALIJA (1BM22CS359)

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 (22CS4PCCON)"** carried out by **TANMAYI S BALIJA (1BM22CS359),** 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.

Prof. Srushti C S Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

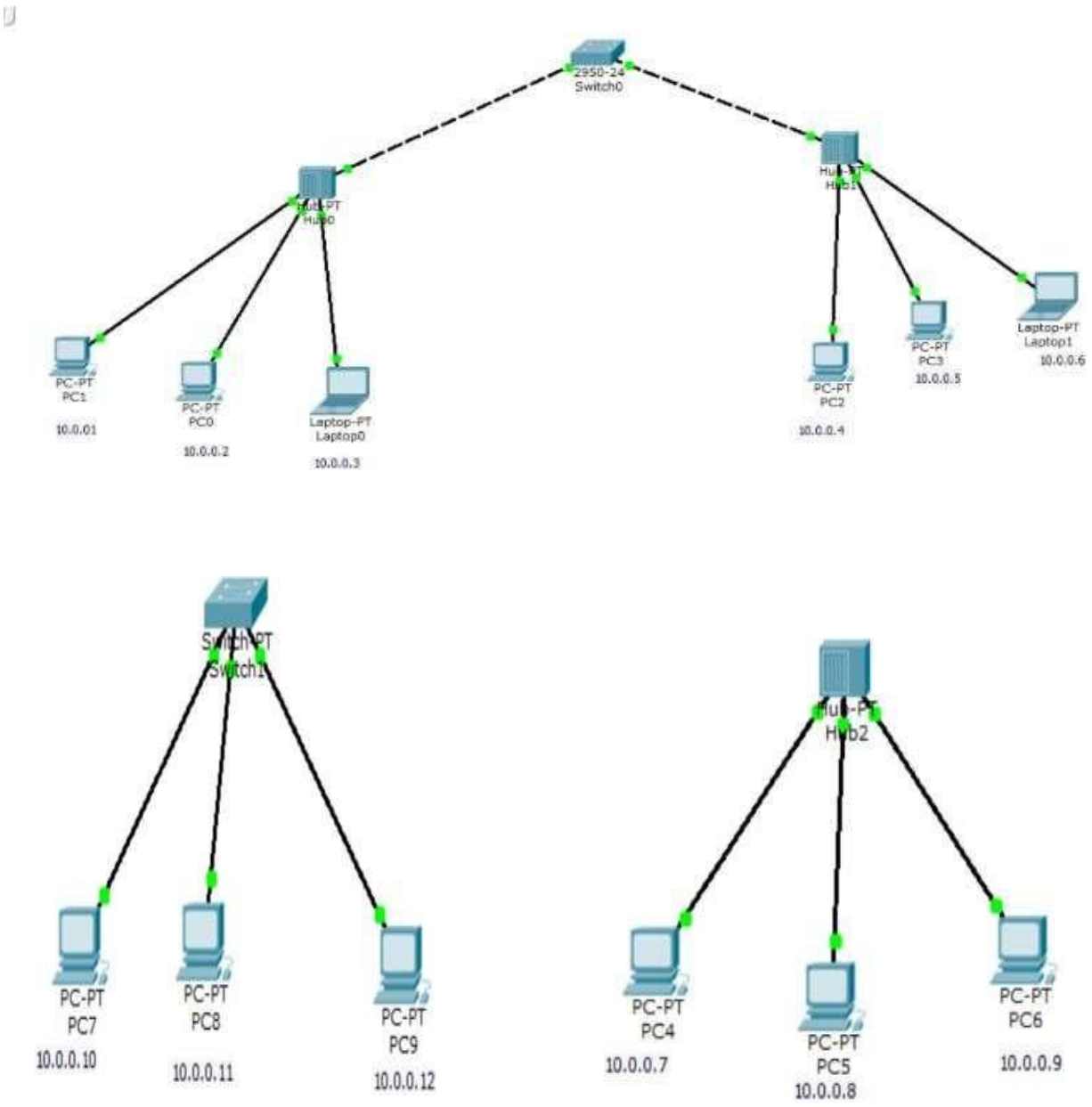
Github Link:https://github.com/tan04mayi/Computer-Networks-Lab

CYCLE - 1

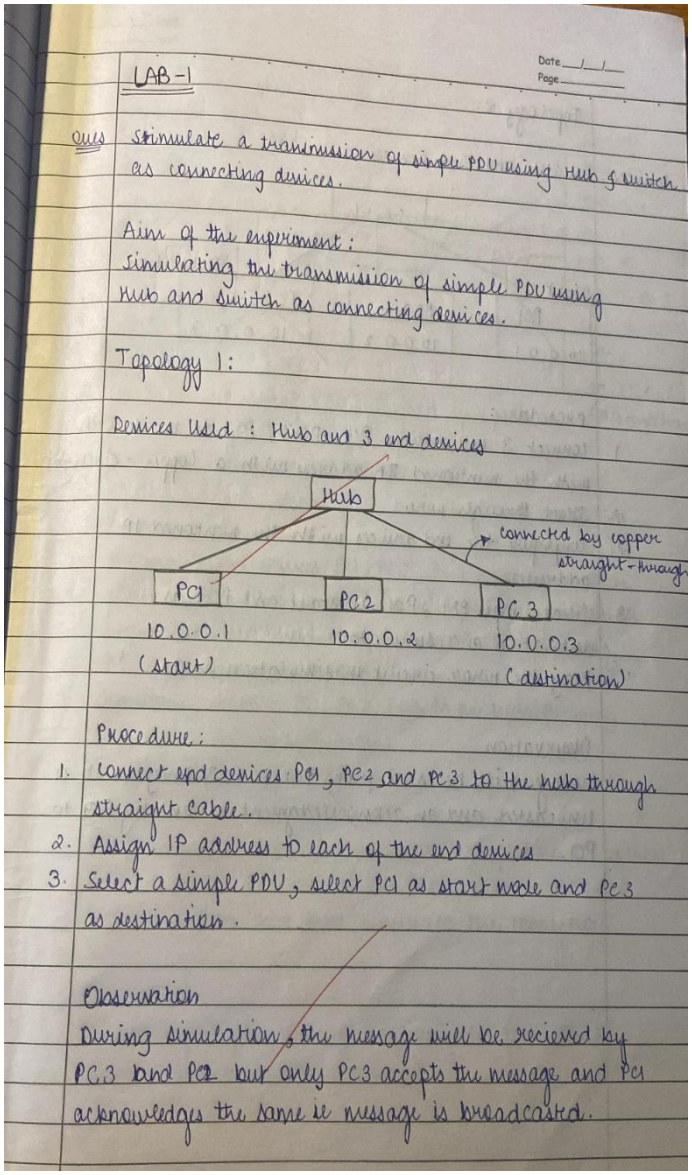
Program 1:

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

Topology:



Procedure and Observations:



LAB-1

Date / /
Page

Aim: Simulate a transmission of simple PDU using Hub & switch as connecting devices.

Aim of the experiment:
Simulating the transmission of simple PDU using Hub and Switch as connecting devices.

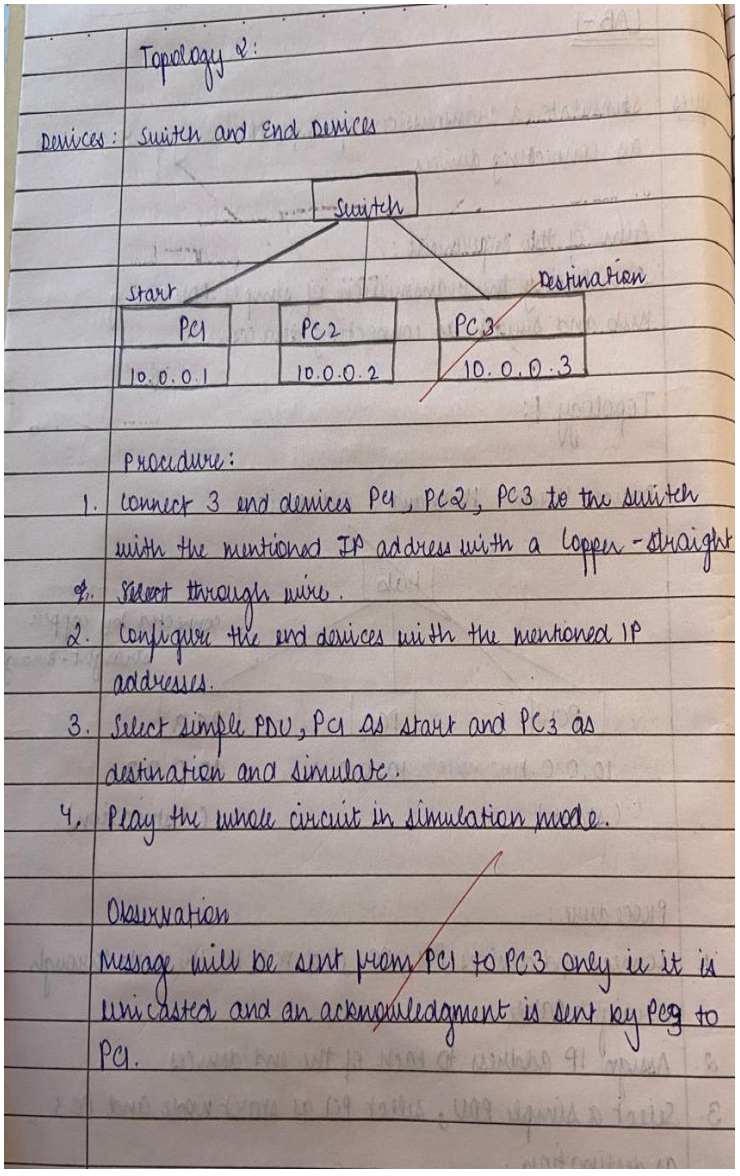
Topology 1:

Devices Used: Hub and 3 end devices

Procedure:

1. Connect end devices PC1, PC2 and PC3 to the hub through straight cable.
2. Assign IP address to each of the end devices
3. Select a simple PDU, select PC1 as start node and PC3 as destination.

Observation
During simulation, the message will be recieved by PC3 and PC2 but only PC3 accepts the message and PC1 acknowledges the same ie message is broadcasted.



Topology 2:

Devices: Switch and End devices

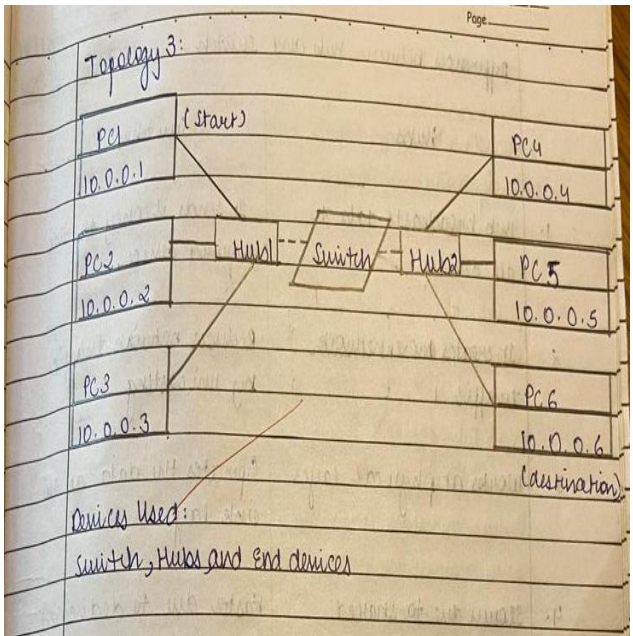
Procedure:

1. Connect 3 end devices PC1, PC2, PC3 to the switch with the mentioned IP address with a copper-straight through wire.
2. Configure the end devices with the mentioned IP addresses.
3. Select simple PDU, PC1 as start and PC3 as destination and simulate.
4. Play the whole circuit in simulation mode.

Observation
Message will be sent from PC1 to PC3 only if it is unicasted and an acknowledgment is sent by PC3 to PC1.

Page

Topology 3:



Devices Used:
Switch, Hubs and end devices

Procedure:
1. Connect 3 end devices PC1, PC2, PC3 with each other
2. Connect the other 3 end devices PC4, PC5, PC6 with each other and then connect them to Hub2 using the copper straight through wire.
3. Similarly connect PC1, PC2, PC3 to the Hub1 using copper straight through wire.
4. Connect Hub1 to Switch and Hub2 to switch using copper cross-over wire.
5. Configure all the end devices with respective IP address
6. Select a simple PDU and assign PC1 as source and PC6 as destination and then simulate the connection.

Observation:
A successful ping message confirms the connectivity between the source and destination.

Date __/__/__
Page

Differences between Hub and Switch.

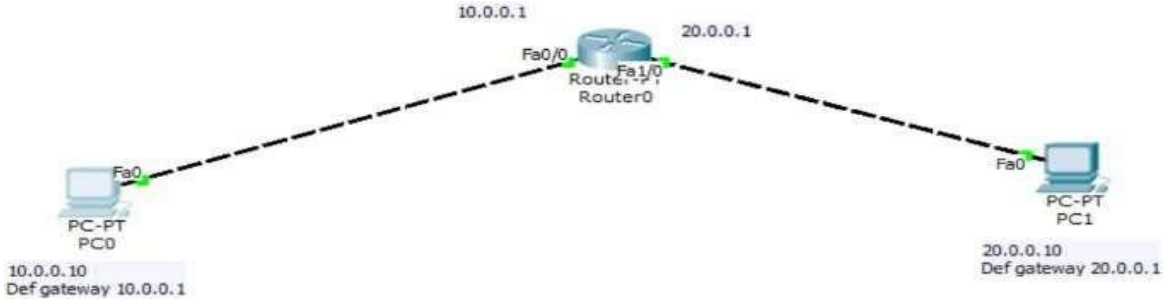
	Hubs	Switches
1.	Hub broadcasts data to all devices	It sends it only to the required device.
2.	It creates more network traffic	Reduces network traffic by unicasting.
3.	Works at physical layer	Operates the data at the link layer.
4.	Slower due to shared bandwidth	Faster due to dedicated bandwidth
5.	Cheaper in cost but less efficient	More expensive compared to hubs and have higher efficiency.

9/10/24

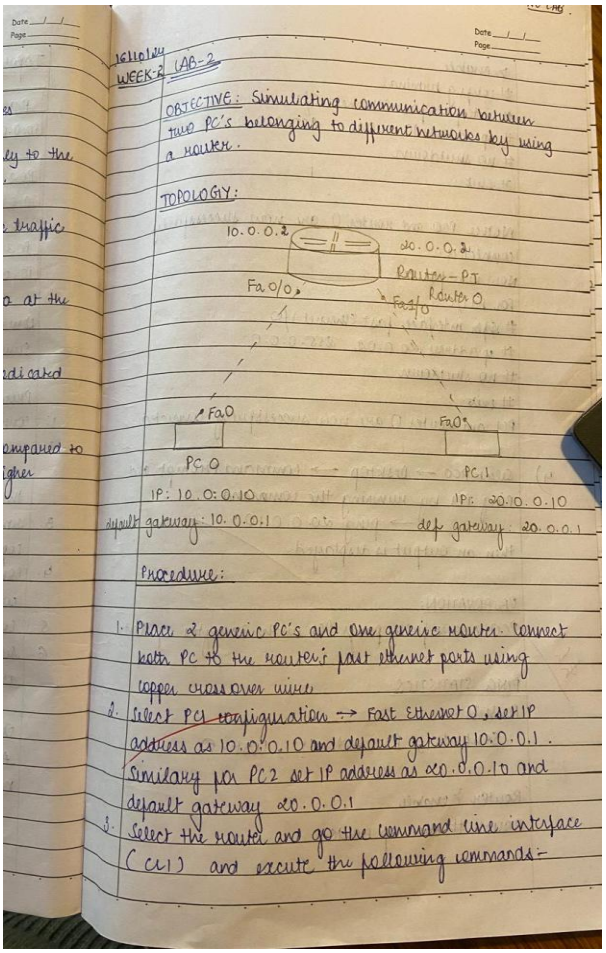
Program 2 :

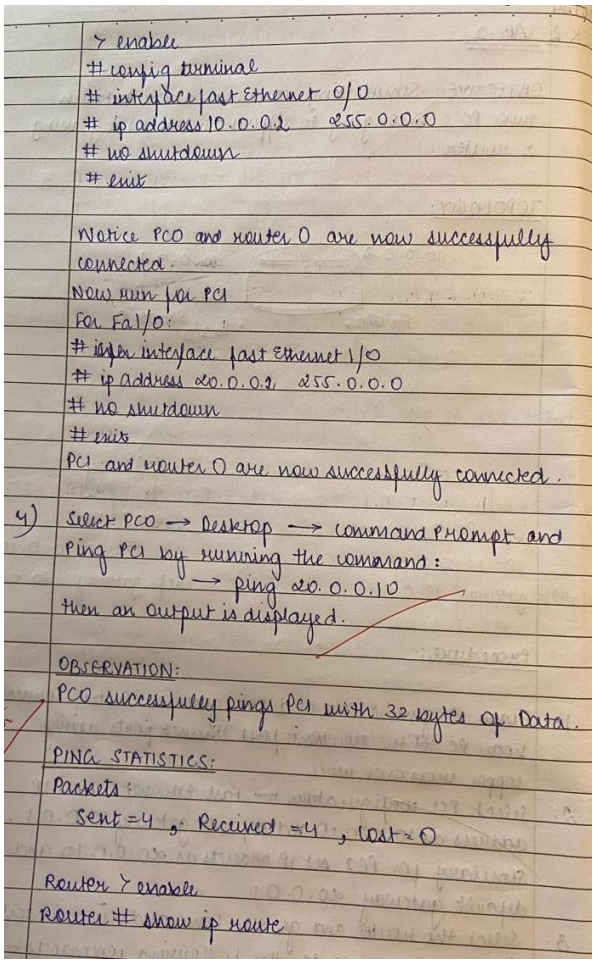
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Topology:

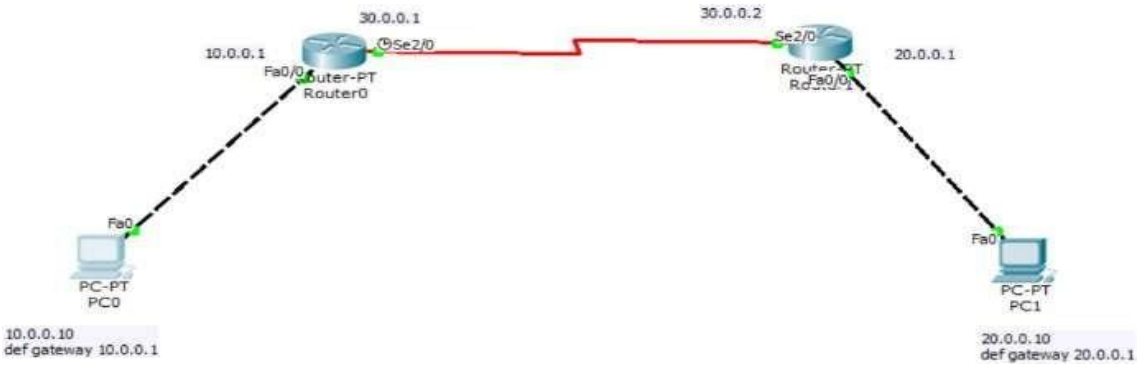


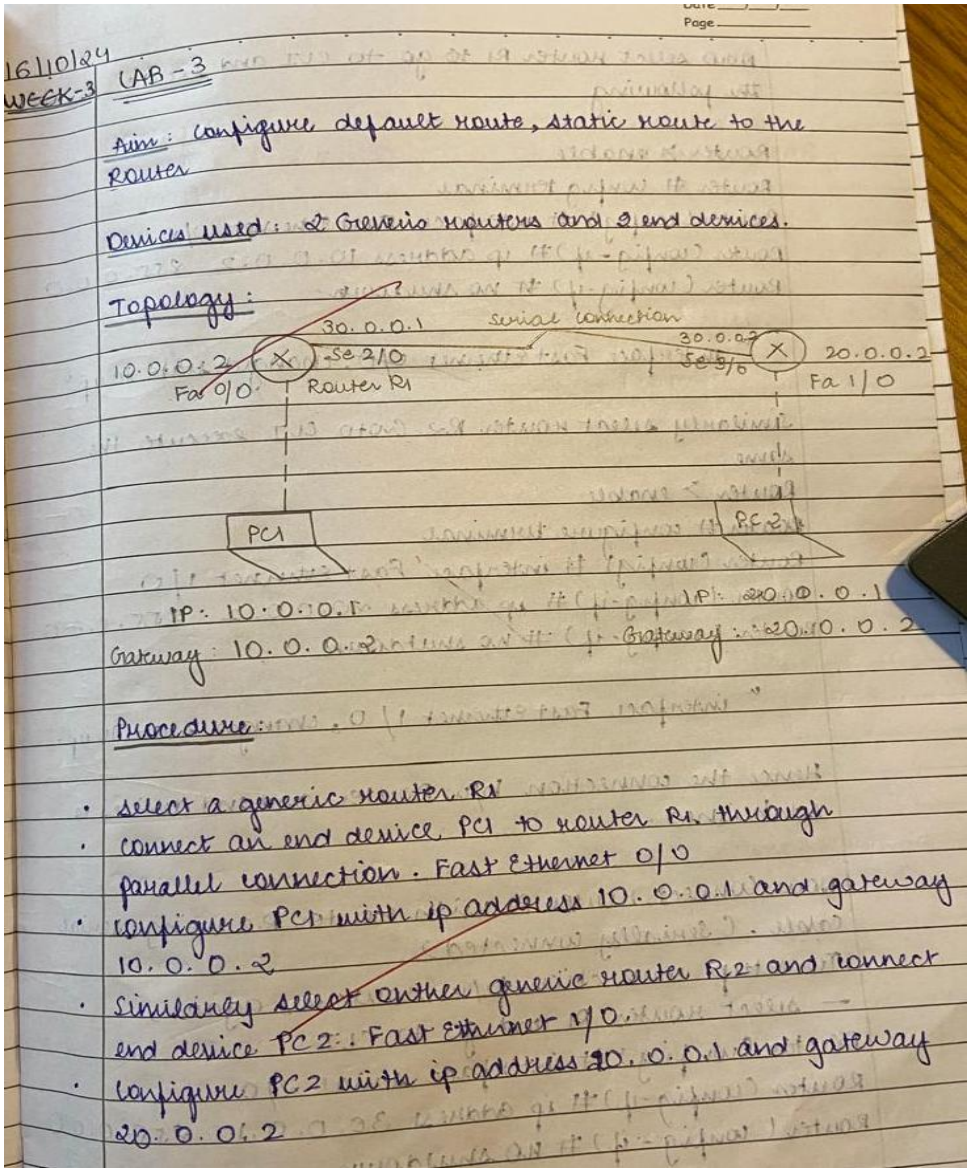
Procedure and Observations:





Part-2:


30.0.0.1
30.0.0.2
10.0.0.1
Se2/0
Fa0/0
Router-PT
Router0
Router-PT
Fa0/0
20.0.0.1
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1
10.0.0.10
def gateway 10.0.0.1
20.0.0.10
def gateway 20.0.0.1


16/10/24
WEEK-3 LAB - 3
Aim: configure default route, static route to the Router
Devices used: 2 Generic routers and 2 end devices.
Topology:
serial connection
30.0.0.1
Se 2/0
10.0.0.2
Fa 0/0
Router R1
30.0.0.2
Se 2/0
20.0.0.2
Fa 1/0
PC1
PC2
IP: 10.0.0.1
Gateway: 10.0.0.2
IP: 20.0.0.1
Gateway: 20.0.0.2
Procedure:
• select a generic router R1
• connect an end device PC1 to router R1 through parallel connection. Fast Ethernet 0/0
• configure PC1 with ip address 10.0.0.1 and gateway 10.0.0.2
• Similarly select another generic router R2 and connect end device PC2: Fast Ethernet 1/0.
• configure PC2 with ip address 20.0.0.1 and gateway 20.0.0.2

Now select router R1 to go to CLI and excute the following

Router > enable
Router # config terminal
Router(config) # interface Fast Ethernet 0/0
Router(config-if)# ip address 10.0.0.2 255.0.0.0
Router(config-if)# no shutdown

" interface Fast Ethernet 0/0, changed state to up"

Similarly select router R2. Goto CLI execute the same.
Router > enable
Router # configure terminal
Router(config)# interface Fast Ethernet 1/0
Router(config-if)# ip address 20.0.0.2 255.0.0.0
Router(config-if)# no shutdown

" interface Fast Ethernet 1/0, changed state to up."

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

Now connect router R1 with router R2 using serial cable. (Serially connected)
To setup connection b/w routers, again,
- select router R1 and go to CLI
Router(config)# interface serial 2/0
Router(config-if)# ip address 30.0.0.1 255.0.0.0
Router(config-if)# no shutdown

Now select router R1 go to CLI and execute the following.
Router > enable
Router # configure terminal
Router

- select router R2 and go to CLI.
Router(config)# interface serial 3/0
Router(config-if)# ip address 30.0.0.2 255.0.0.0
Router(config-if)# no shutdown

" interface serial 2/0 changed state to up
interface serial 3/0 changed state to up "

Observation

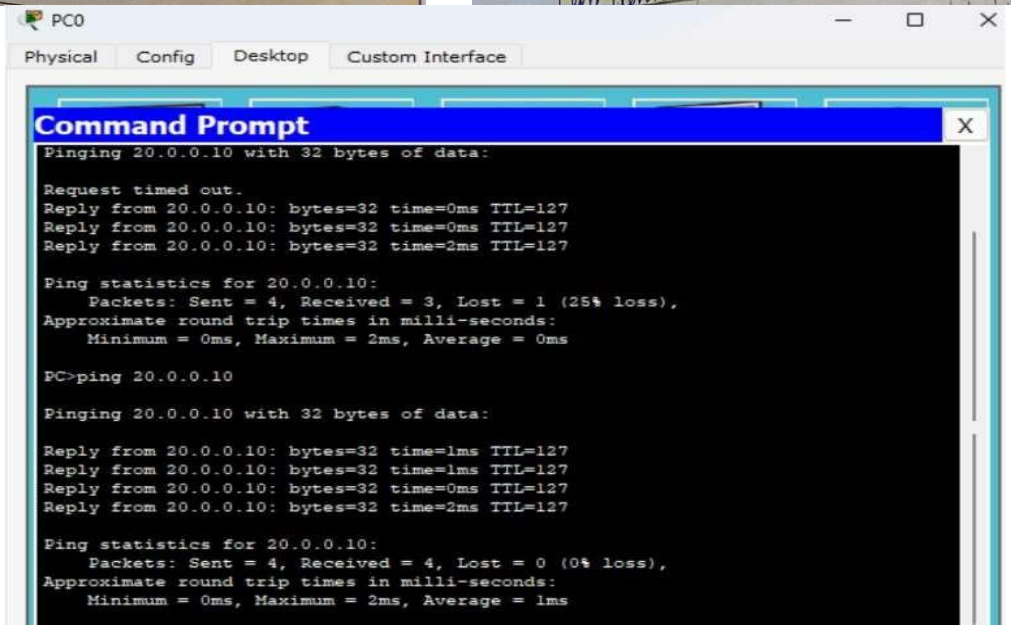
→ After setting up the mentioned topology.
Now try to ping PC2 with PC1.
• Open command prompt for PC1 and type
> ping 20.0.0.1

Output 1:
→ Destination host unreachable
Packets sent: 4 received: 0 lost: 4 loss = 100%.

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

23/10/24
ping 30.0.0.1 → successful
Packets sent: 4 received: 4 lost: 0 lost = 0%.

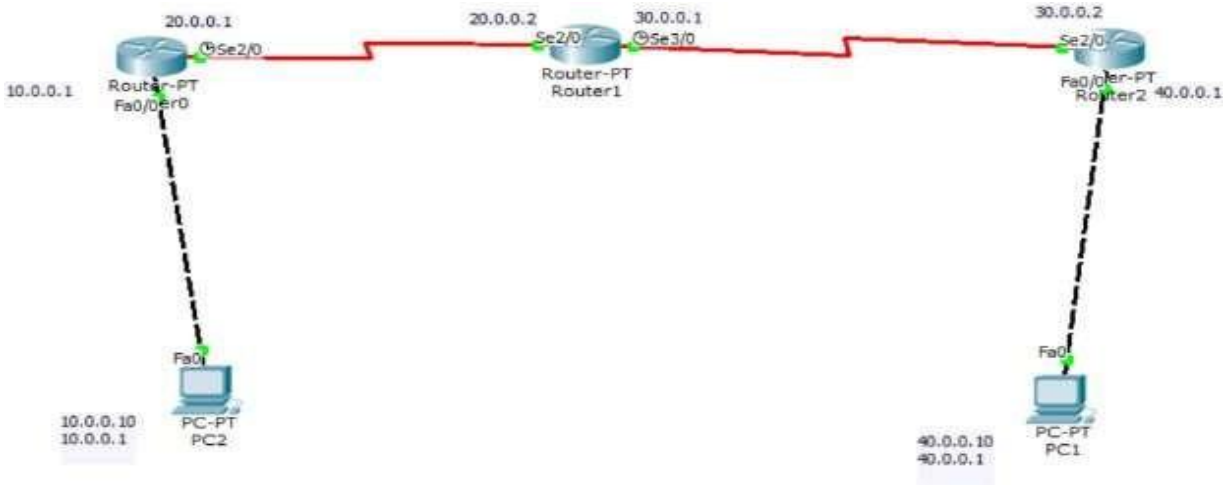
Hence although the routers were connected serially and devices were unable to ping each other.

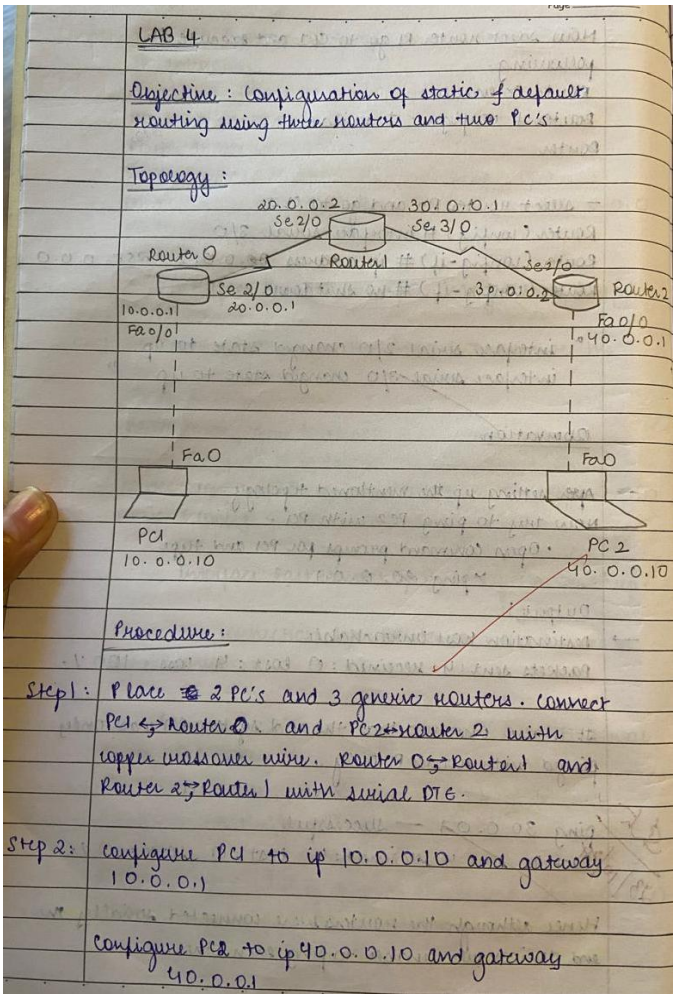


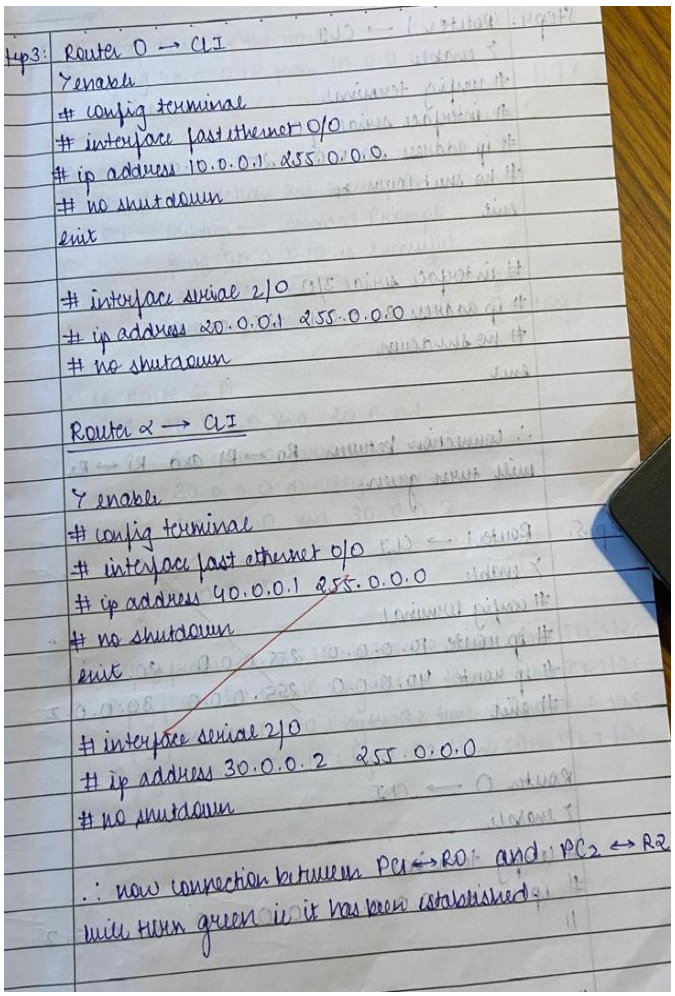
Program 3:

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

Topology, Procedure and Observations :







Step 4: Router 1 → CLI

> enable
config terminal
interface serial 2/0
ip address 20.0.0.2 255.0.0.0
no shutdown
exit

interface serial 3/0
ip address 30.0.0.1 255.0.0.0
no shutdown
exit

∴ connection between $R_0 \leftrightarrow R_1$ and $R_1 \leftrightarrow R_2$ will turn green.

Step 5: Router 1 → CLI

> enable
config terminal
ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2
exit

Router 0 → CLI

> enable
config terminal
ip route 0.0.0.0 0.0.0.0 20.0.0.2
//

Router 2 → CLI

> enable
config terminal
ip route 0.0.0.0 0.0.0.0 30.0.0.1

Step 6: Show ip route in the router, ping 40.0.0.10 from 10.0.0.10 and vice versa.

Observations

- All the connections are successful
- PC1 → Desktop → Command Prompt
  ping 40.0.0.10 is successful
  ping statistics
  Sent = 4, Received = 4, Lost = 0 (0% loss)

- ip route in R1
  S 10.0.0.0 via 20.0.0.1
  C 20.0.0.0 directly
  C 30.0.0.0 directly
  S 40.0.0.0 via 30.0.0.2

- ping 10.0.0.10

Reply from 40.0.0.10 : bytes : 32 time : 0ms 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

13/11/24.

Command Prompt

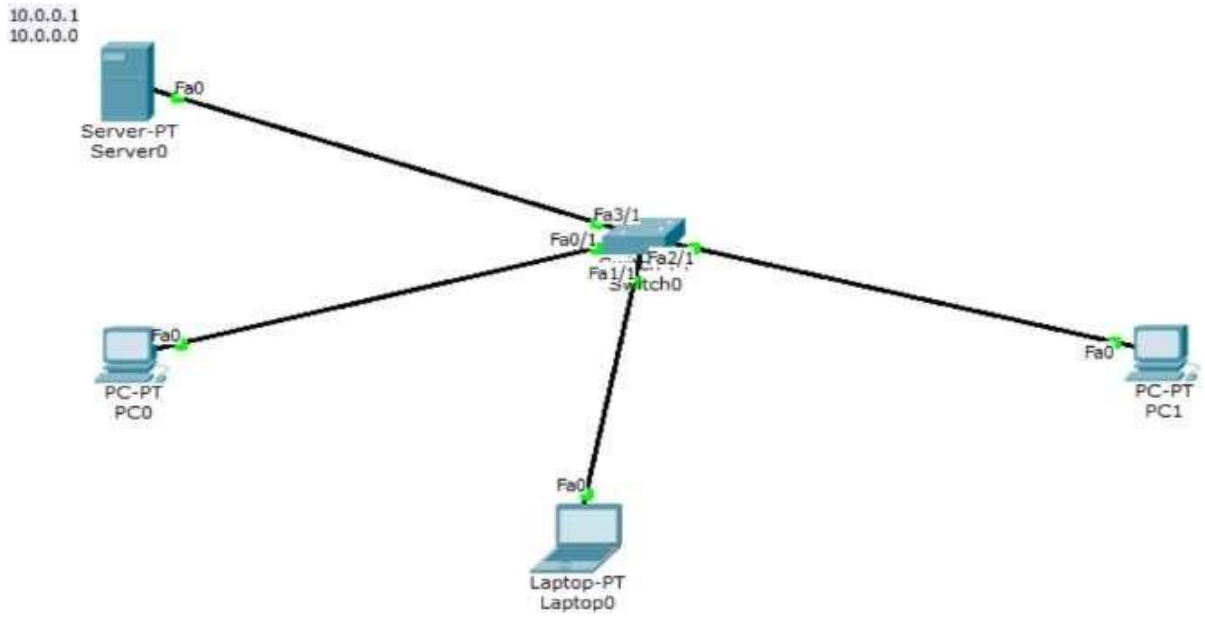
```
Pinging 40.0.0.10 with 32 bytes of data:

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:

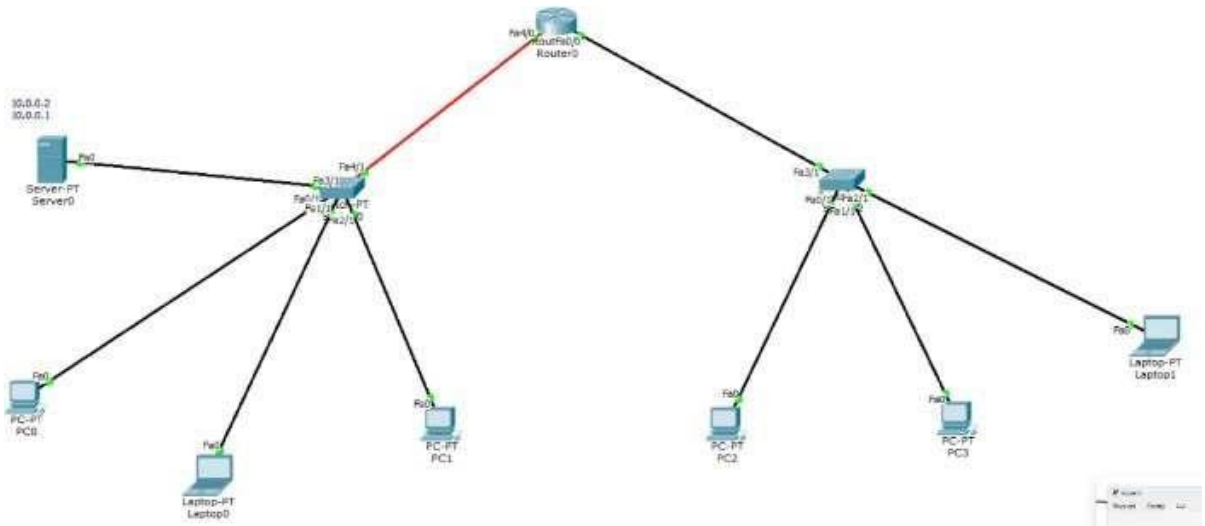
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

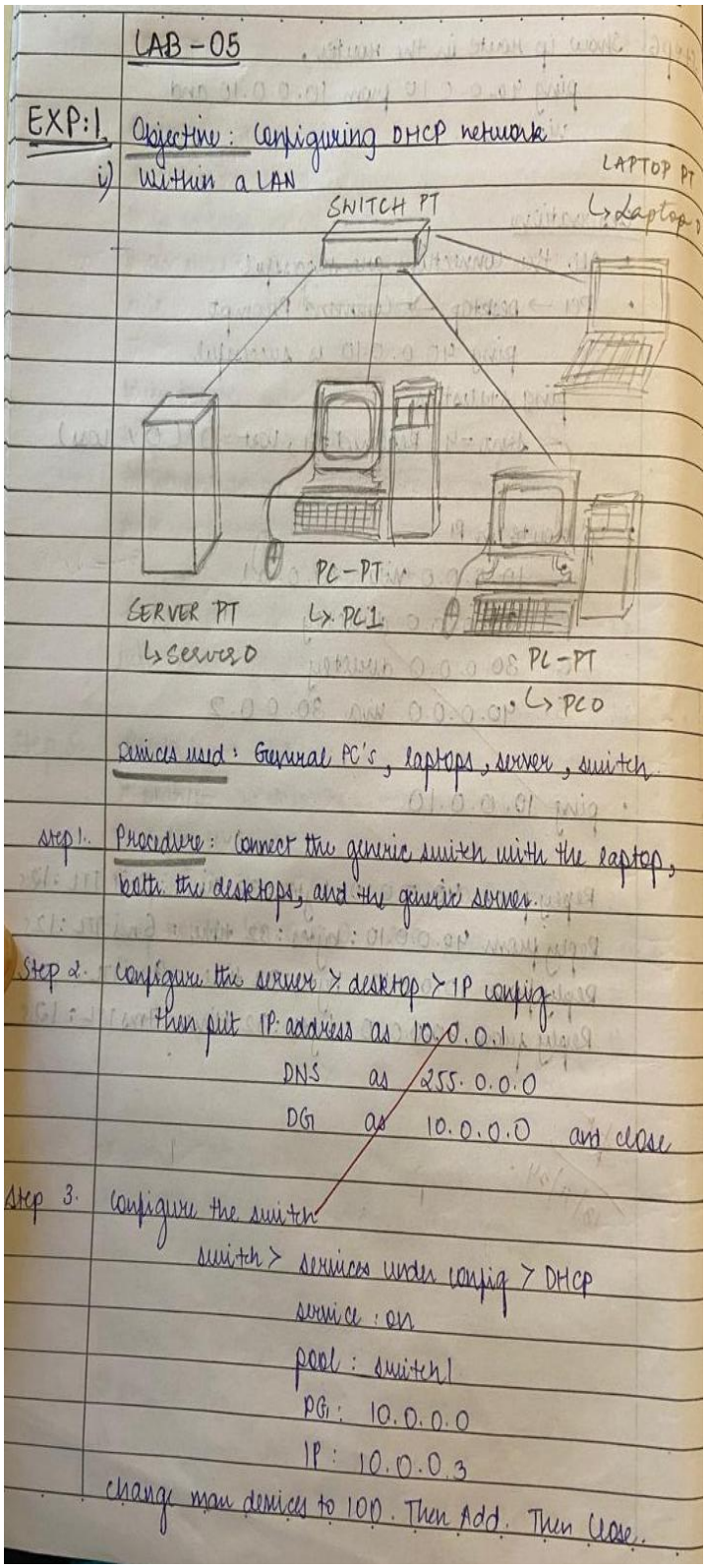
Within LAN



Outside LAN



Procedure and Observation:



LAB - 05

EXP:1 Objective: Configuring DHCP network

i) Within a LAN

SWITCH PT

LAPTOP PT

↳ Laptop 0

PC-PT

SERVER PT ↳ PC1

↳ Server 0

PC-PT

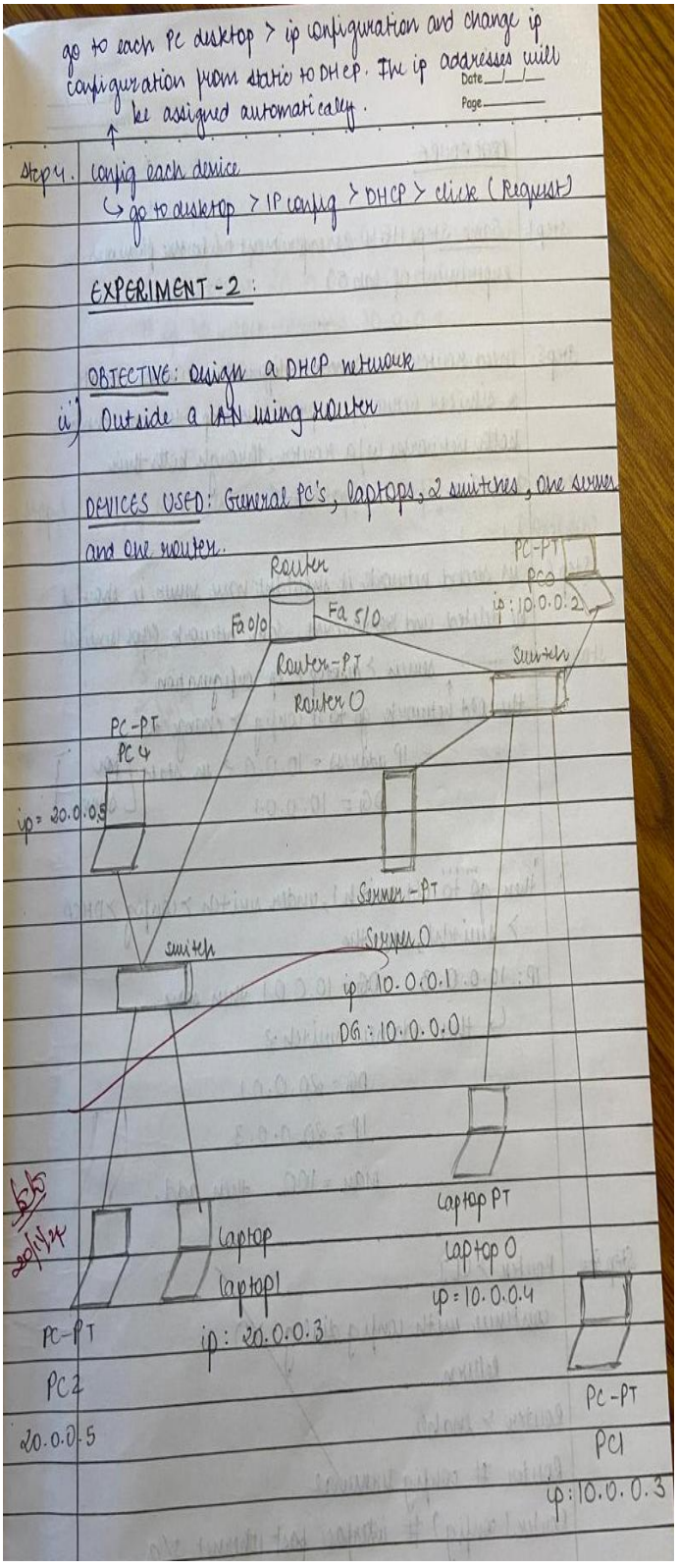
↳ PC 0

Devices used: General PC's, laptops, server, switch

Step 1. Procedure: Connect the generic switch with the laptop, both the desktops, and the generic server.

Step 2. Configure the server > desktop > IP config.
Then put IP address as 10.0.0.1
DNS as 255.0.0.0
DG as 10.0.0.0 and close

Step 3. Configure the switch
switch > services under config > DHCP
service : on
pool : switch1
PG: 10.0.0.0
IP: 10.0.0.3
change max devices to 100. Then Add. Then close.



go to each PC desktop > ip configuration and change ip configuration from static to DHCP. The ip addresses will be assigned automatically.

Step 4. Config each device
↳ go to desktop > IP config > DHCP > click (Request)

EXPERIMENT - 2 :

OBJECTIVE: Design a DHCP network

ii) Outside a LAN using router

DEVICES USED: General PC's, laptops, 2 switches, one server and one router.

Router
Fa 0/0
Fa 5/0
Router-PT
Router 0
PC-PT
PC0
ip: 10.0.0.2
Switch
PC-PT
PC4
ip = 20.0.0.5
Server-PT
Server 0
ip 10.0.0.1
DG: 10.0.0.0
switch
laptop
laptop1
ip: 20.0.0.3
Laptop PT
Laptop 0
ip = 10.0.0.4
PC-PT
PC2
20.0.0.5
PC-PT
PC1
ip: 10.0.0.3

PROCEDURE:

step1: Same steps 1 to 4 as experiment 1 (ie the previous experiment of lab 5)

step5: need router to connect 2 different networks. Create a similar network as present in step 1-4 and connect both networkes w/a router, through both their switches.

step6: in second network it shouldnt have server ie should be deleted and ~~re~~ removed. Same network (for servers)

Start → server > desktop > ip configuration

the old network go to IP config > change to
IP address = 10.0.0.2 in static [for server]
DG = 10.0.0.1

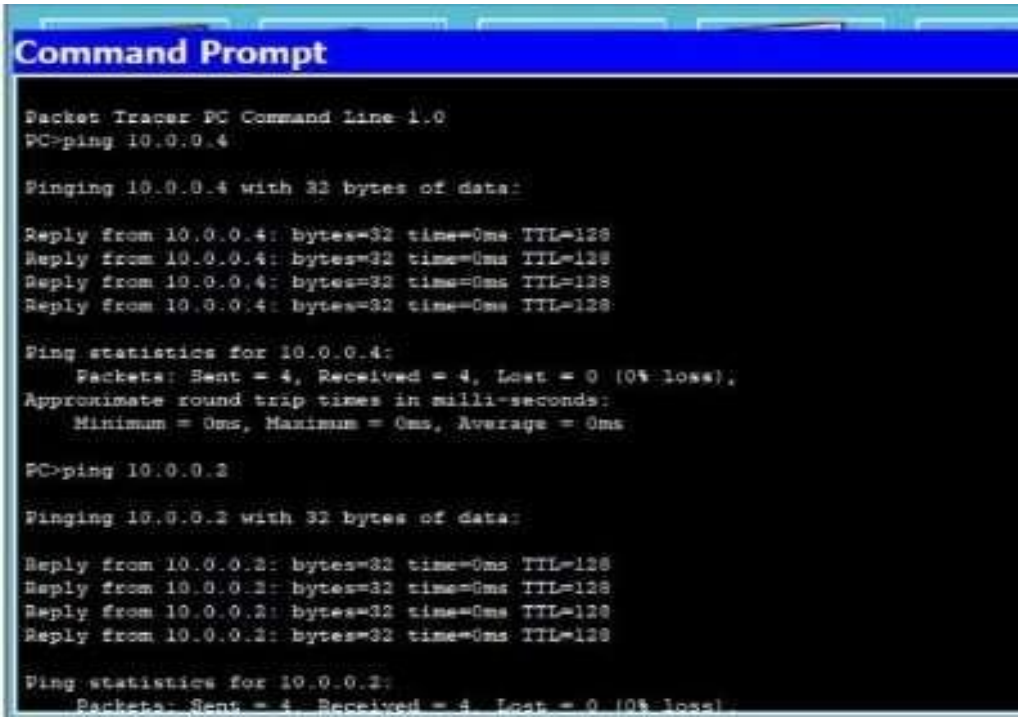
then go to old switch1, under switch > config > DHCP > switch1, edit the
IP: 10.0.0.3 DG = 10.0.0.1 then save.
↳ then add new switch 2
DG = 20.0.0.1
IP = 20.0.0.3
Max = 100 then add.

step7. Router > CLI
continue with config dialog: NO
Return
Router > enable
Router # config terminal
Router (config)# interface fast ethernet 5/0
ip address 10.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shutdown
exit

Router > enable
config terminal
interface fast ethernet 6/0
ip address 20.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shutdown
exit

step8: Config each device > desktop > IP config > DHCP > click (Request)

Within the LAN:

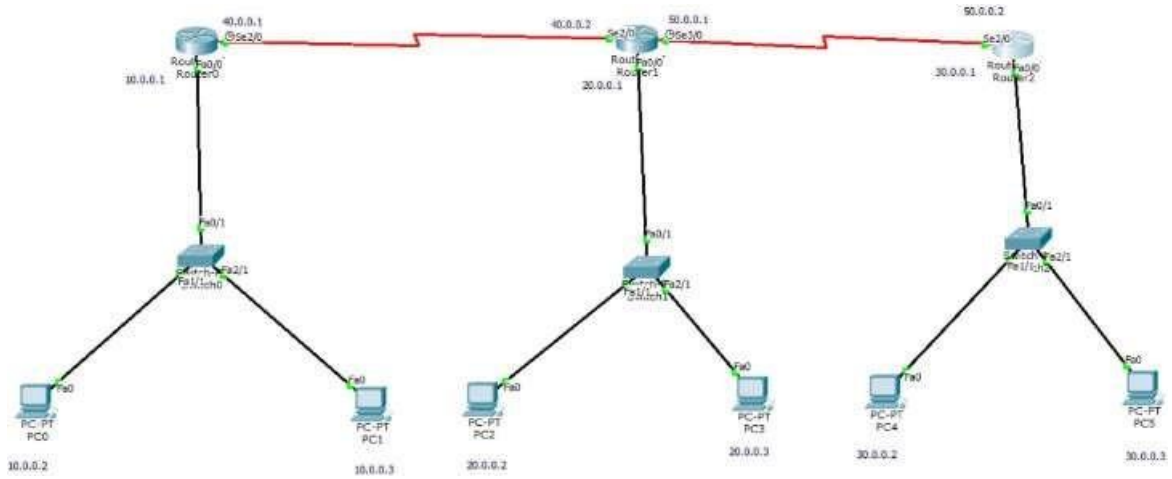


Outside LAN:

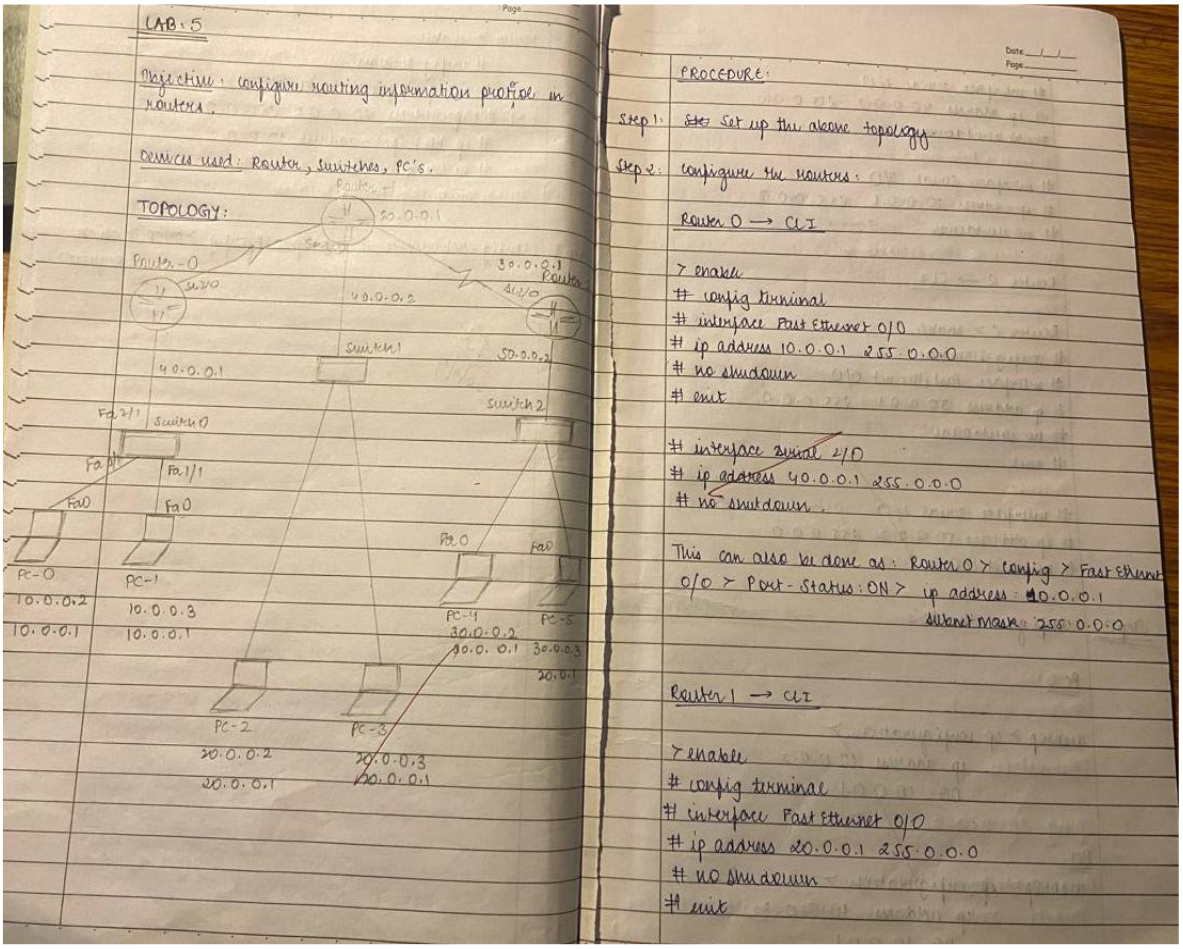
Program 5:

Aim: Configure RIP routing Protocol in Routers.

Topology:



Procedure and Observation:



interface serial 2/0
ip address 40.0.0.2 255.0.0.0
no shutdown

interface serial 3/0
ip address 50.0.0.1 255.0.0.0
no shutdown

Router 2 → CLI

Router 2 > enable
config terminal
interface FastEthernet 0/0
ip address 30.0.0.1 255.0.0.0
no shutdown
exit

interface serial 2/0
ip address: 50.0.0.2 255.0.0.0
no shutdown

Step 3: configure the PC's.

PC 0

desktop > ip configuration >
ip address: 10.0.0.2
DG: 10.0.0.1

PC 1
desktop > ip configuration >
ip address: 10.0.0.3
DG: 10.0.0.1

Date / /
Page

PC 2
desktop > ip configuration >
ip address: 20.0.0.2
DG : 20.0.0.1

PC 3
desktop > ip configuration >
ip address: 20.0.0.3
DG : 20.0.0.1

PC 4

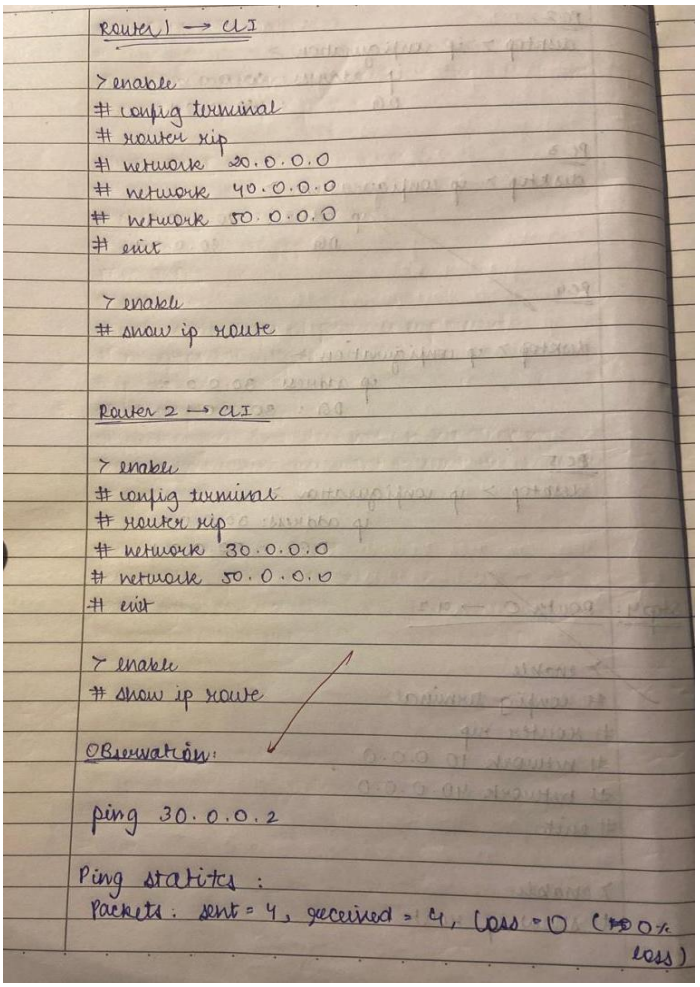
desktop > ip configuration >
ip address: 30.0.0.2
DG : 30.0.0.1

PC 5
desktop > ip configuration >
ip address: 30.0.0.3
DG : 30.0.0.1

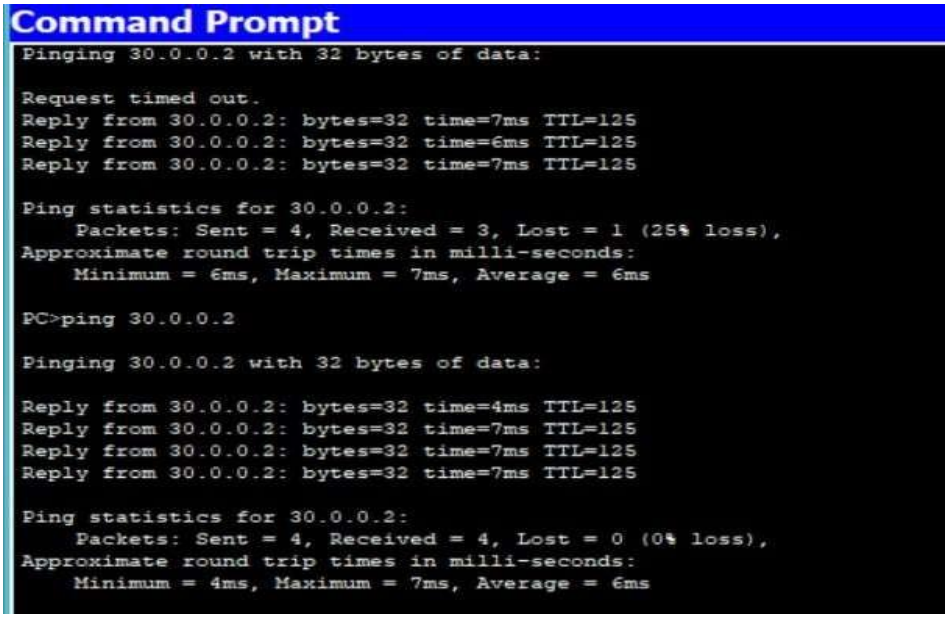
Step 4: Router 0 → CLI

> enable
config terminal
router rip
network 10.0.0.0
network 40.0.0.0
exit

> enable
show ip route



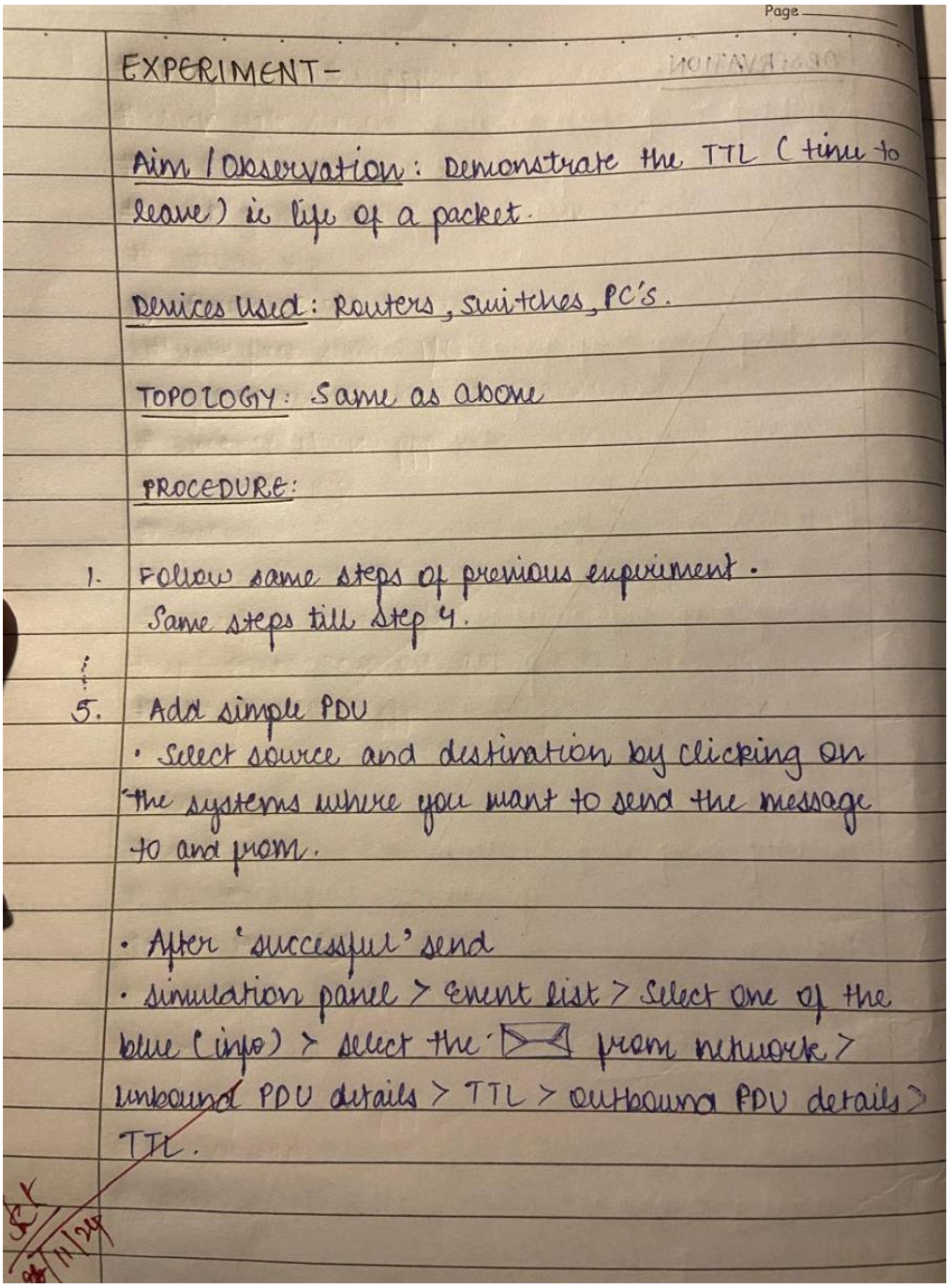
Observation Screenshot:



Program 6:

Aim: Demonstrate the TTL/ Life of a Packet.

Procedure and Observation:



EXPERIMENT-

Aim / Observation: Demonstrate the TTL (time to leave) in life of a packet.

Devices Used: Routers, switches, PC's.

TOPOLOGY: Same as above

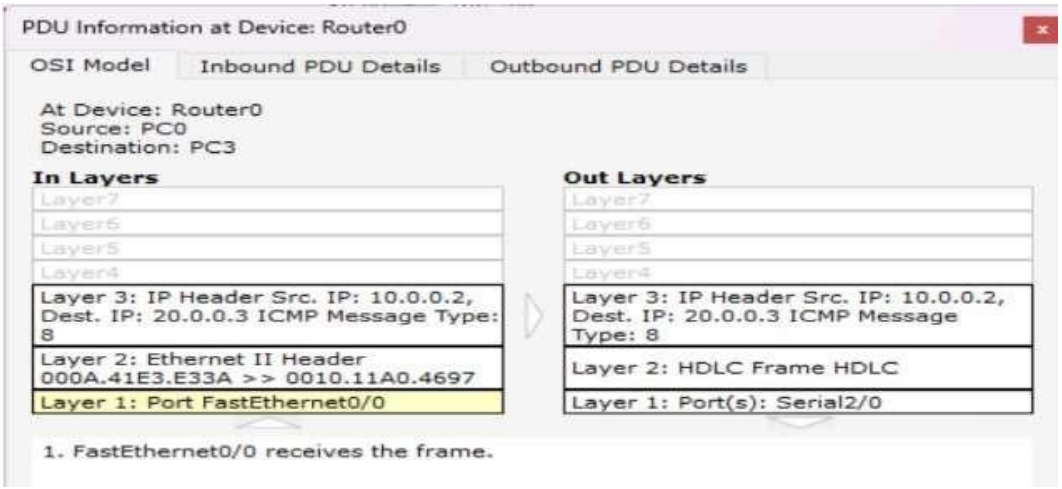
PROCEDURE:

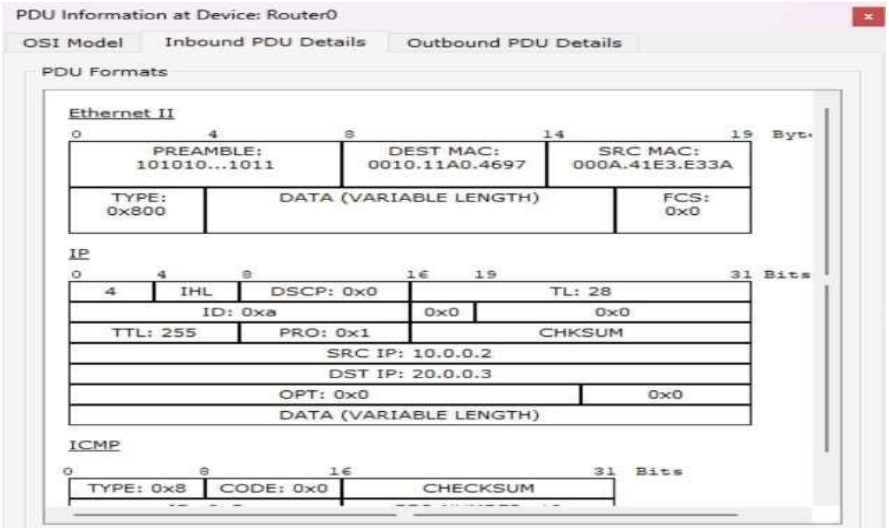
1. Follow same steps of previous experiment.
   Same steps till step 4.

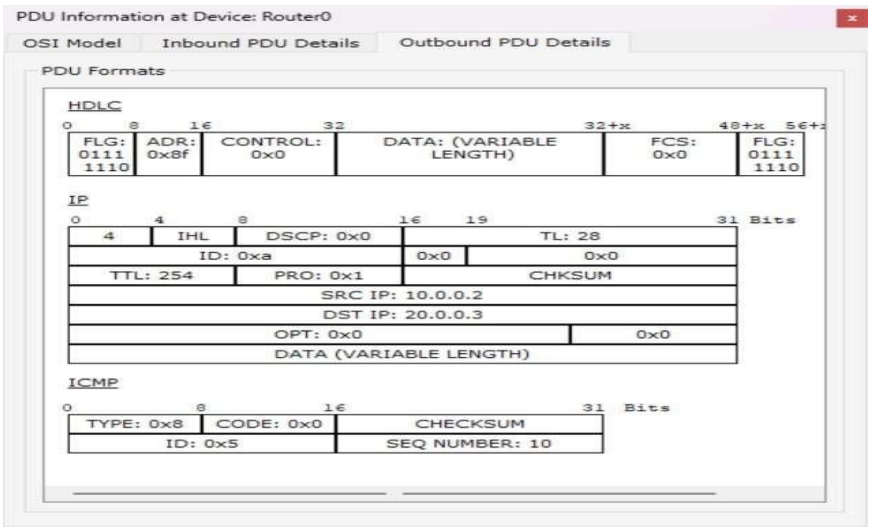
5. Add simple PDU
   • Select source and destination by clicking on the systems where you want to send the message to and from.

• After 'successful' send
• simulation panel > event list > Select one of the blue (info) > select the ✉ from network > inbound PDU details > TTL > outbound PDU details > TTL.

Observation Screenshot:



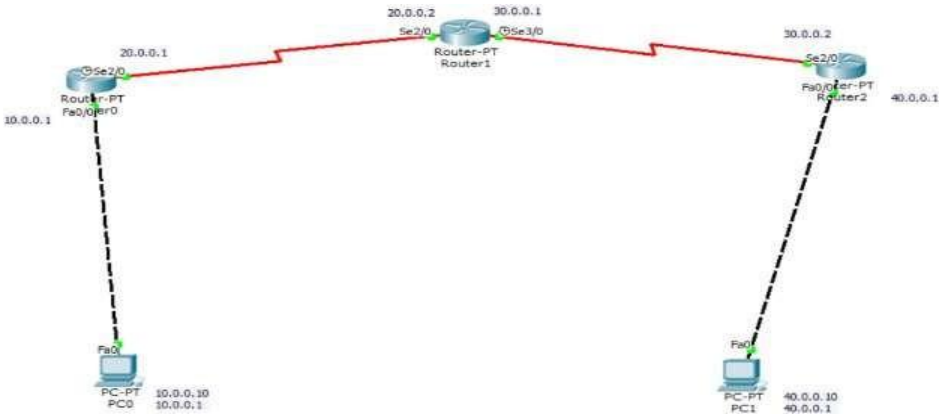




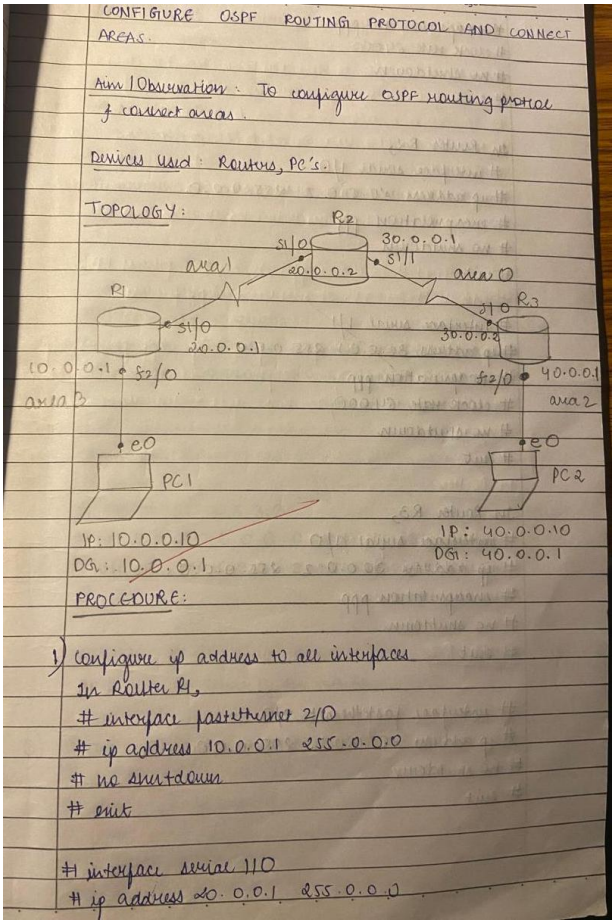
Program 7:

Aim: Configure OSPF routing protocol**.**

Topology:



Procedure and Observation:


CONFIGURE OSPF ROUTING PROTOCOL AND CONNECT AREAS.

Aim / Observation : To configure OSPF routing protocol & connect areas.

Devices used : Routers, PC's.

TOPOLOGY:

IP: 10.0.0.10
DG: 10.0.0.1

IP: 40.0.0.10
DG: 40.0.0.1

PROCEDURE:

1) Configure ip address to all interfaces
In Router R1,
interface fastethernet 2/0
ip address 10.0.0.1 255.0.0.0
no shutdown
exit

#1 interface serial 1/0
ip address 20.0.0.1 255.0.0.0

encapsulation ppp
clock rate 64000
no shutdown
exit

In Router R2,
interface serial 1/0
ip address 20.0.0.2 255.0.0.0
encapsulation ppp
no shutdown
exit

interface serial 1/1
ip address 30.0.0.1 255.0.0.0
encapsulation ppp
clock rate 64000
no shutdown
exit

In Router R3,
interface serial 1/0
ip address 30.0.0.2 255.0.0.0
encapsulation ppp
no shutdown
exit

interface fastethernet 2/0
ip address 40.0.0.1 255.0.0.0
no shutdown
exit

2) Now, enable ip routing by configuring ospf routing ~~protocol~~ protocol in all routers.

In Router R1,
router ospf 1
router-id 1.1.1.1
network 10.0.0.0 0.255.255.255 area 3
network 30.0.0.0 0.255.255.255 area 1
exit

In Router R2,
router ospf 1
router-id 2.2.2.2
network 20.0.0.0 0.255.255.255 area 1
network 30.0.0.0 0.255.255.255 area 0
exit

In Router R3,
router ospf 1
router-id 3.3.3.3
network 30.0.0.0 0.255.255.255 area 0
network 40.0.0.0 0.255.255.255 area 2
exit

We have to configure router id when we configure ospf. It is used to identify the router.

3) Now checking routing table of R1,

show ip route

Gateway of last resort is not set

C 10.0.0.0/8 is directly connected, FastEthernet2/0
C 20.0.0.0/8 is directly connected, serial 1/0
O IA 40.0.0.0/8 via 20.0.0.2, 00:04:23, serial
O IA 30.0.0.0/8 via 20.0.0.2, 00:07:29, serial

Here, R2 knows Area 0. Network 20.0.0.0 connected R2 from R1, so R1 knows networks through this network.

R3(config)# router ospf 1, here, 1 is process ID, it can be 1-65535. It initializes ospf process.

There must be one interface up to keep ospf process up so it better to configure loopback address to routers. It is a virtual interface never goes down once we configured.

R1 Router

interface loopback 0
ip add 172.16.1.253 255.255.0.0
no shutdown

Router R2

interface loopback 0
ip add 172.16.1.253 255.255.0.0
no shutdown

Router R3

interface loopback 0
ip add 172.16.1.254 255.255.0.0
no shutdown

4) Check routing table of R3

R3# show ip route

Gateway of last resort is not set

O IA 20.0.0.0/8 via 30.0.0.1, 00:18:58, serial1/0
C 40.0.0.0/8 is directly connected, FastEthernet 2/0
C 30.0.0.0/8 is directly connected, serial 1/0

Here, R3 doesn't know about the area 3 so we have to create virtual link between R1 and R2.

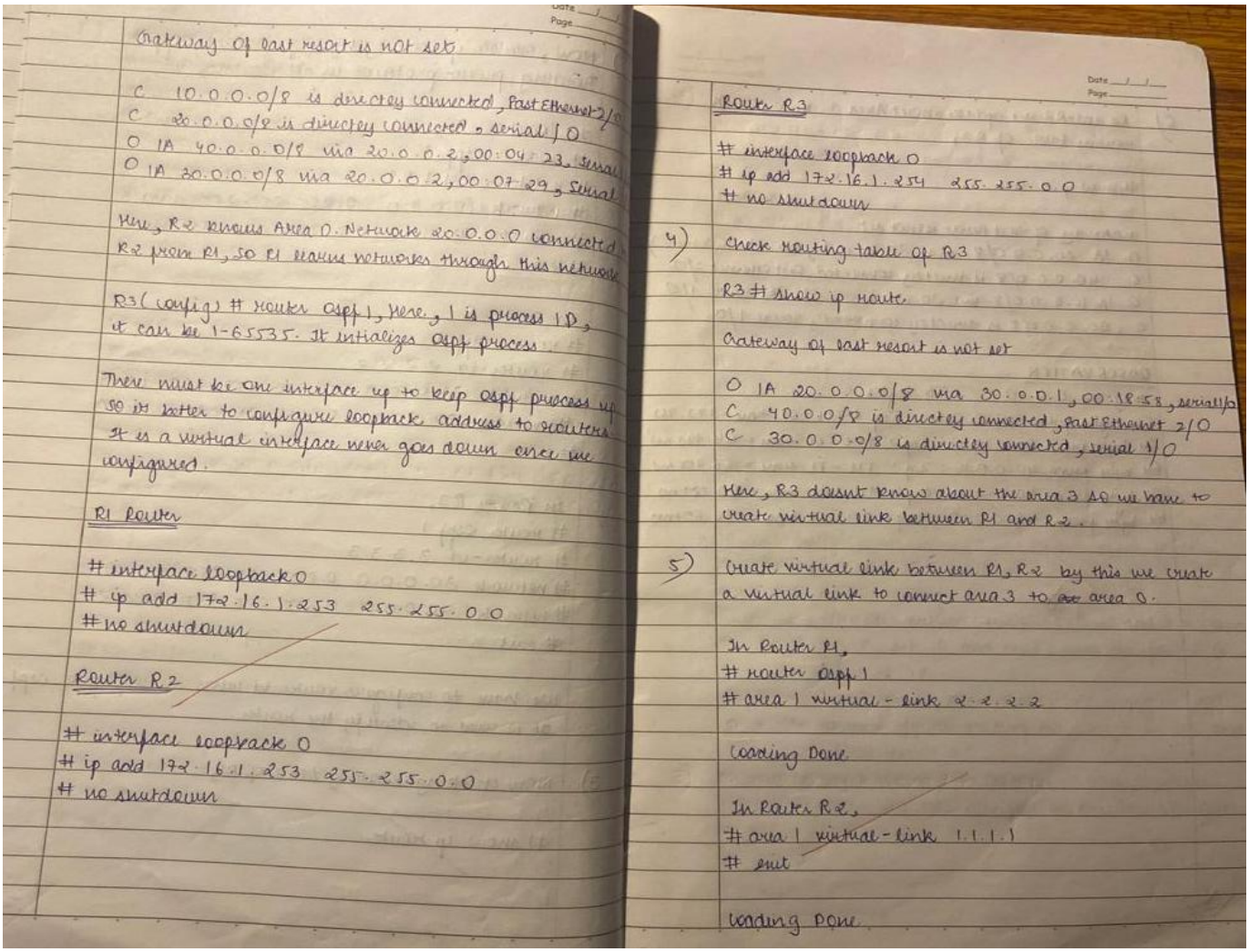
5) Create virtual link between R1, R2 by this we create a virtual link to connect area 3 to ~~as~~ area 0.

In Router R1,
router ospf 1
area 1 virtual-link 2.2.2.2

Loading Done

In Router R2,
area 1 virtual-link 1.1.1.1
exit

Loading Done



Gateway of last resort is NOT set

C 10.0.0.0/8 is directly connected, FastEthernet 2/0
C 20.0.0.0/8 is directly connected, Serial 1/0
O IA 40.0.0.0/8 via 20.0.0.2, 00:04:23, Serial
O IA 30.0.0.0/8 via 20.0.0.2, 00:07:29, Serial

Here, R2 knows Area 0. Network 20.0.0.0 connected R2 from R1, so R1 knows networks through this network

R3(config)# router ospf 1, Here, 1 is process ID, it can be 1-65535. It initializes ospf process.

There must be one interface up to keep ospf process up so its better to configure loopback address to routers. It is a virtual interface never goes down once we configured.

R1 Router

interface loopback 0
ip add 172.16.1.253 255.255.0.0
no shutdown

Router R2

interface loopback 0
ip add 172.16.1.253 255.255.0.0
no shutdown

Router R3

interface loopback 0
ip add 172.16.1.254 255.255.0.0
no shutdown

4) Check routing table of R3

R3 # show ip route

Gateway of last resort is not set

O IA 20.0.0.0/8 via 30.0.0.1, 00:18:58, serial1/0
C 40.0.0.0/8 is directly connected, FastEthernet 2/0
C 30.0.0.0/8 is directly connected, serial 1/0

Here, R3 doesn't know about the area 3 so we have to create virtual link between R1 and R2.

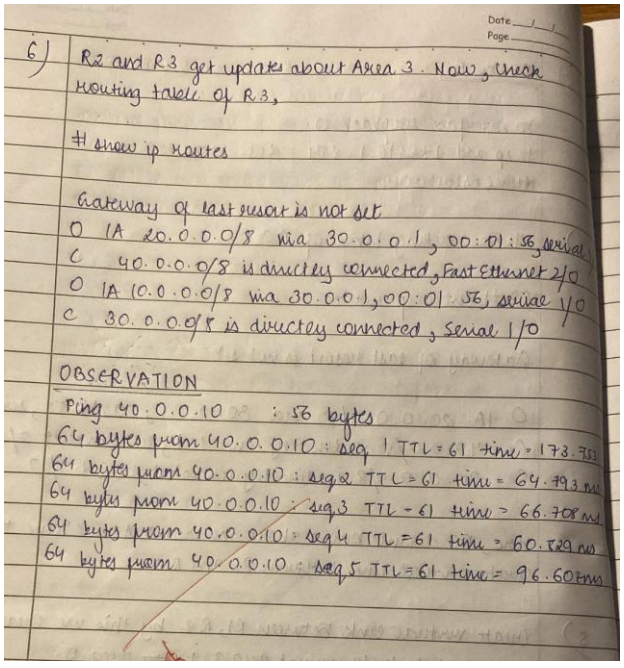
5) Create virtual link between R1, R2 by this we create a virtual link to connect area 3 to ~~as~~ area 0.

In Router R1,
router ospf 1
area 1 virtual-link 2.2.2.2

Loading Done

In Router R2,
area 1 virtual-link 1.1.1.1
end

Loading Done



6) R2 and R3 get updates about Area 3. Now, Check routing table of R3,

show ip routes

Gateway of last resort is not set
O IA 20.0.0.0/8 via 30.0.0.1, 00:01:56, serial
C 40.0.0.0/8 is directly connected, FastEthernet 2/0
O IA 10.0.0.0/8 via 30.0.0.1, 00:01:56, serial 1/0
C 30.0.0.0/8 is directly connected, Serial 1/0

OBSERVATION

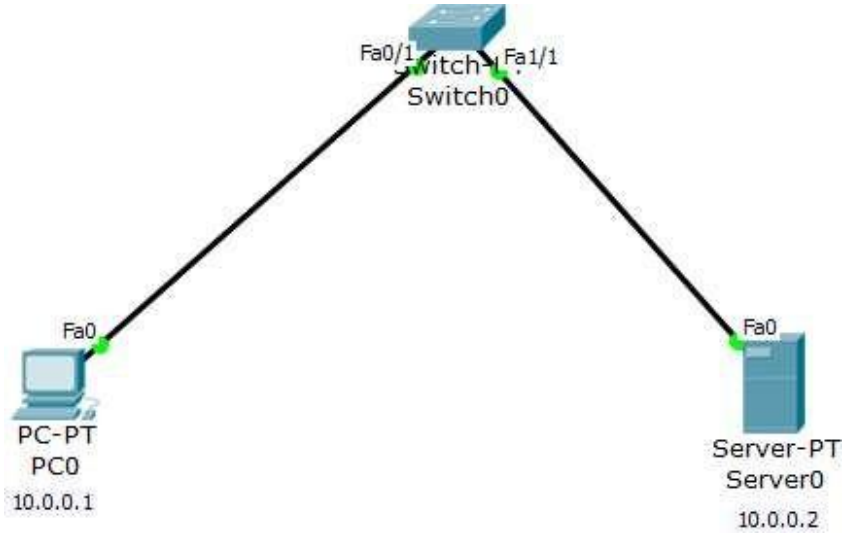
Ping 40.0.0.10 : 56 bytes
64 bytes from 40.0.0.10 : seq 1 TTL=61 time=173.753
64 bytes from 40.0.0.10 : seq 2 TTL=61 time=64.793 ms
64 bytes from 40.0.0.10 : seq 3 TTL=61 time=66.708 ms
64 bytes from 40.0.0.10 : seq 4 TTL=61 time=60.729 ms
64 bytes from 40.0.0.10 : seq 5 TTL=61 time=96.607ms

```
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
```

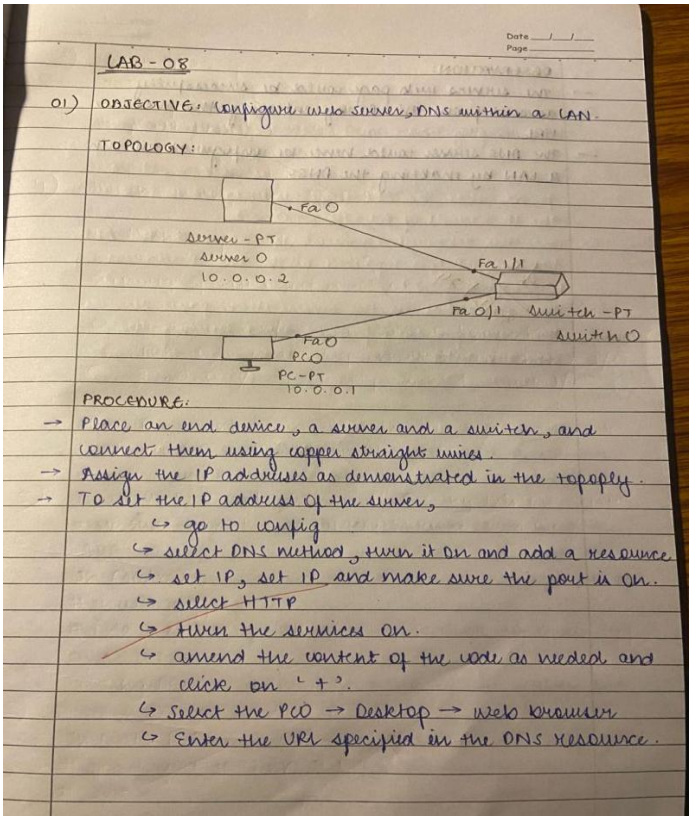
Program 8:

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

Topology:



Procedure and Observations:

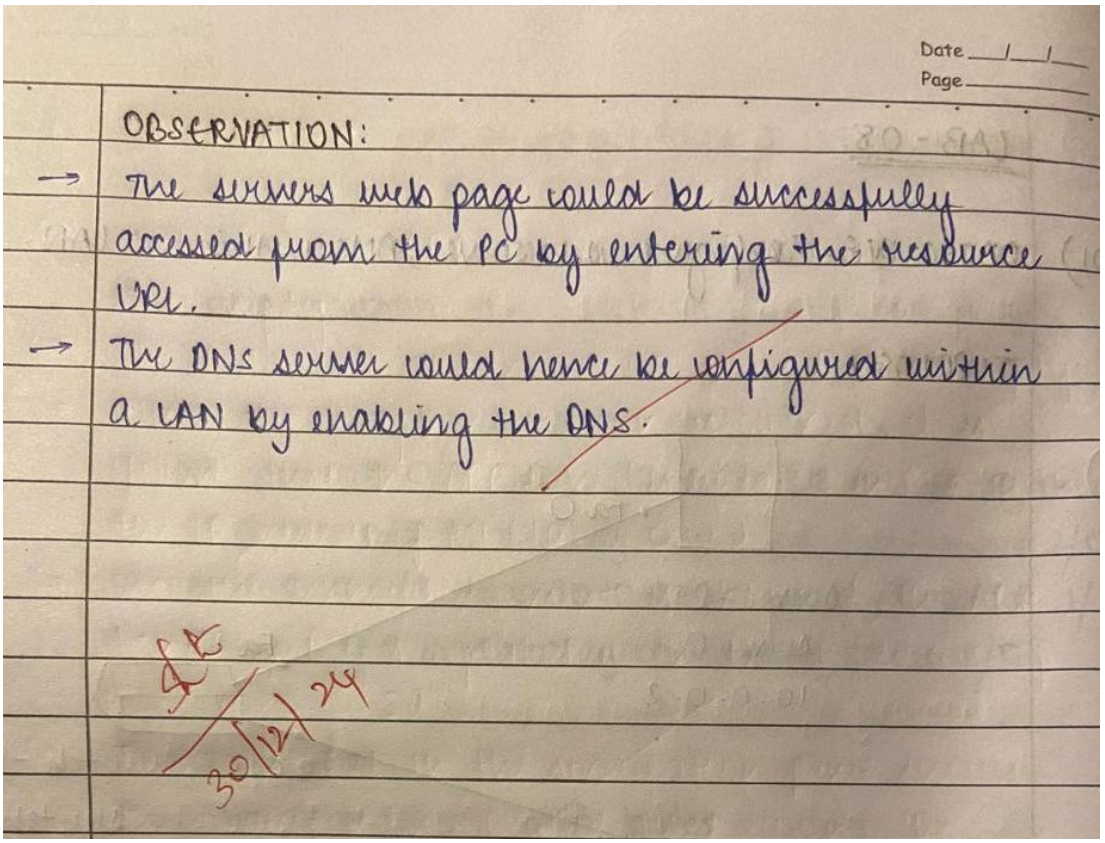

LAB - 08

01) OBJECTIVE: Configure web server, DNS within a LAN.

TOPOLOGY:

PROCEDURE:

→ Place an end device, a server and a switch, and connect them using copper straight wires.
→ Assign the IP addresses as demonstrated in the topology.
→ To set the IP address of the server,
  ↳ go to config
  ↳ select DNS method, turn it on and add a resource.
  ↳ set IP, set IP and make sure the port is on.
  ↳ select HTTP
  ↳ turn the services on.
  ↳ amend the content of the code as needed and click on '+'.
  ↳ Select the PC0 → Desktop → Web browser
  ↳ Enter the URL specified in the DNS resource.



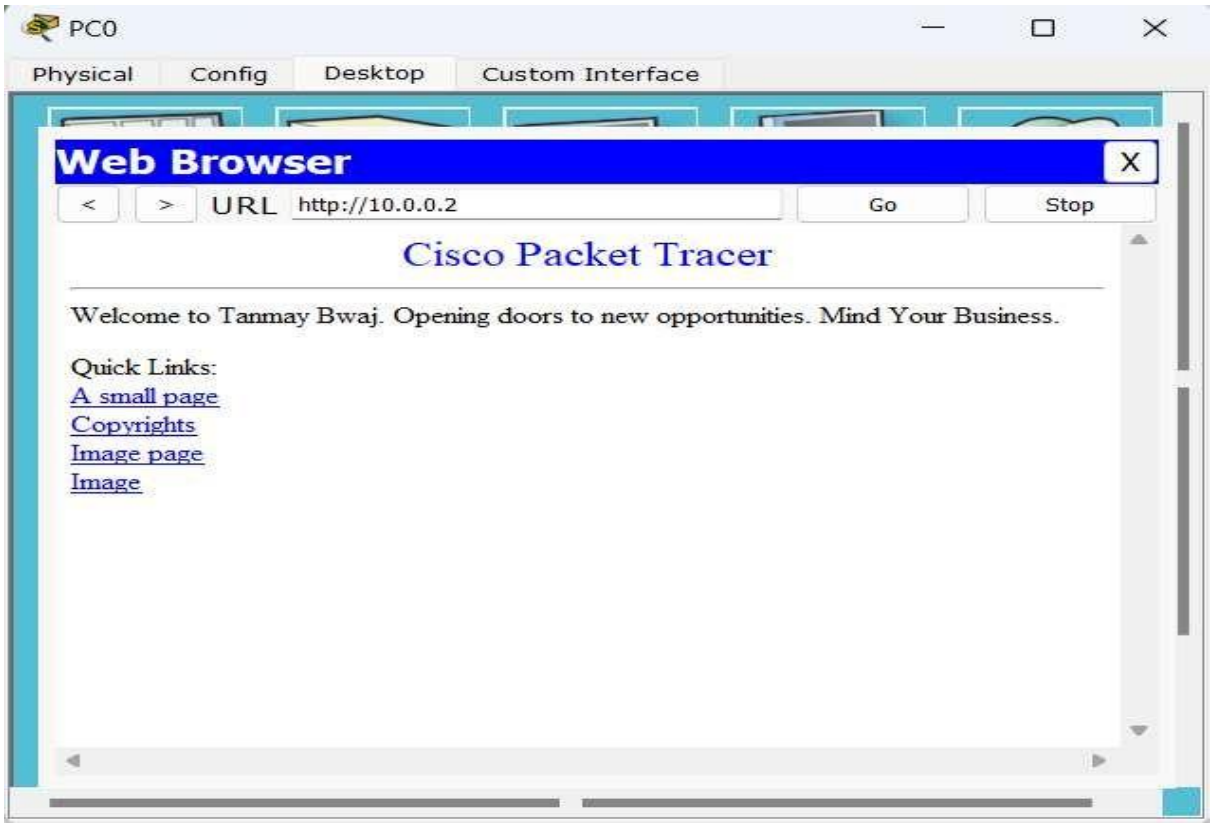
OBSERVATION:

→ The servers web page could be successfully accessed from the PC by entering the resource URL.

→ The DNS server could hence be configured within a LAN by enabling the DNS.

30/12/24

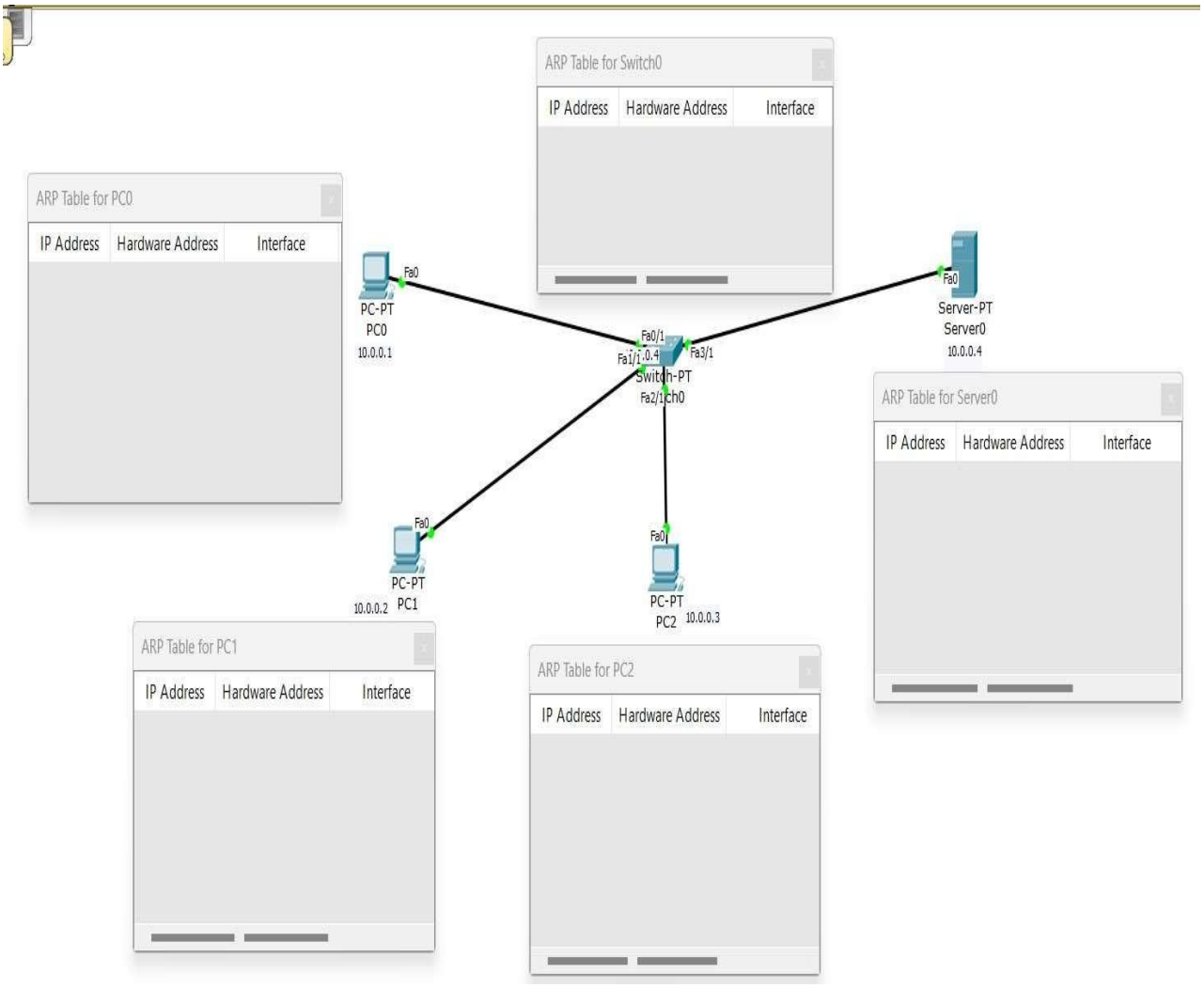
Observation Screenshot:



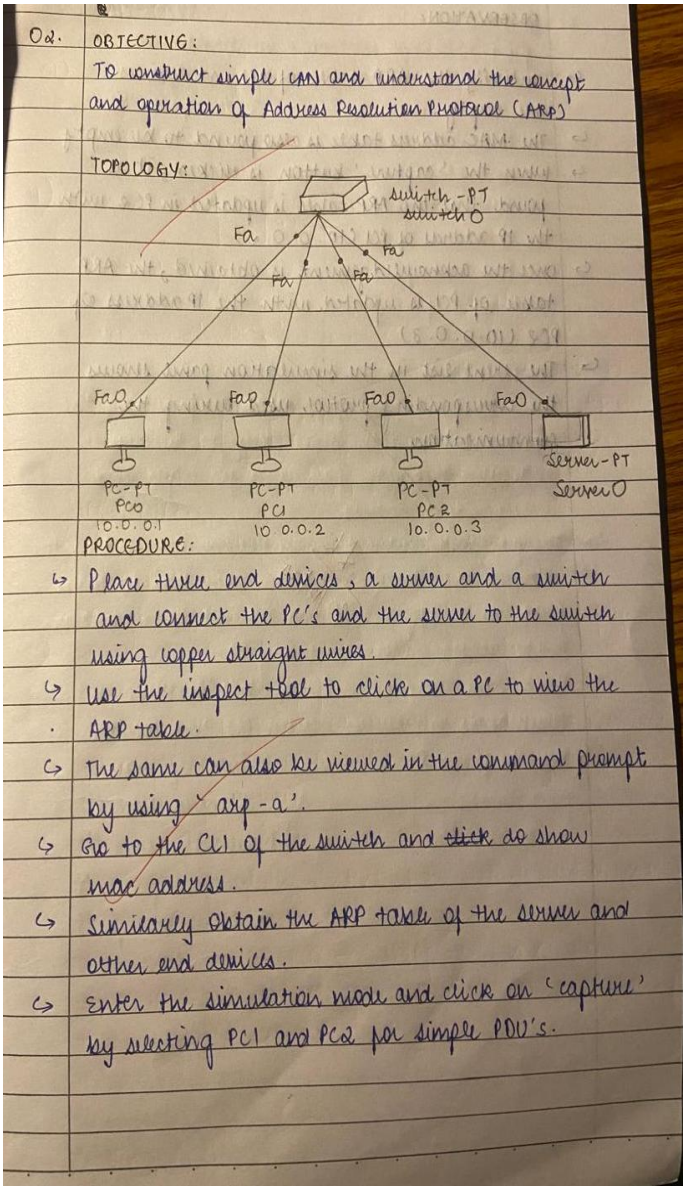
Program 9:

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

Topology:



Procedure and Observations:



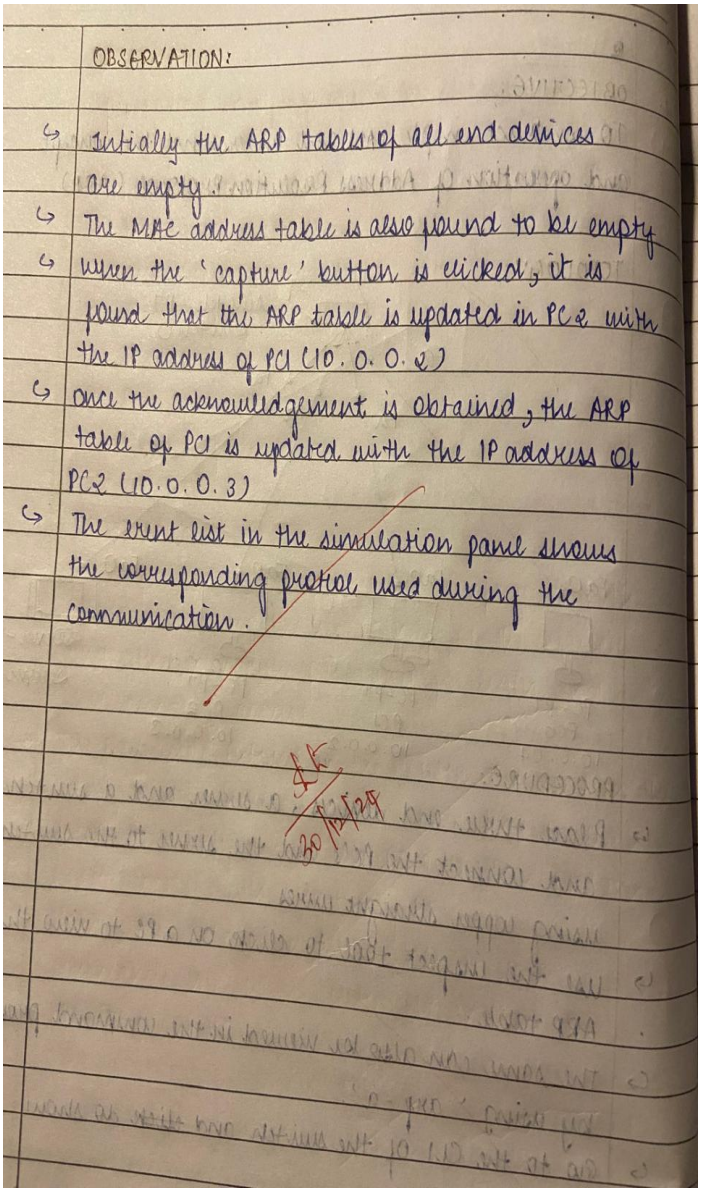
02. OBJECTIVE:

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

TOPOLOGY:

PROCEDURE:

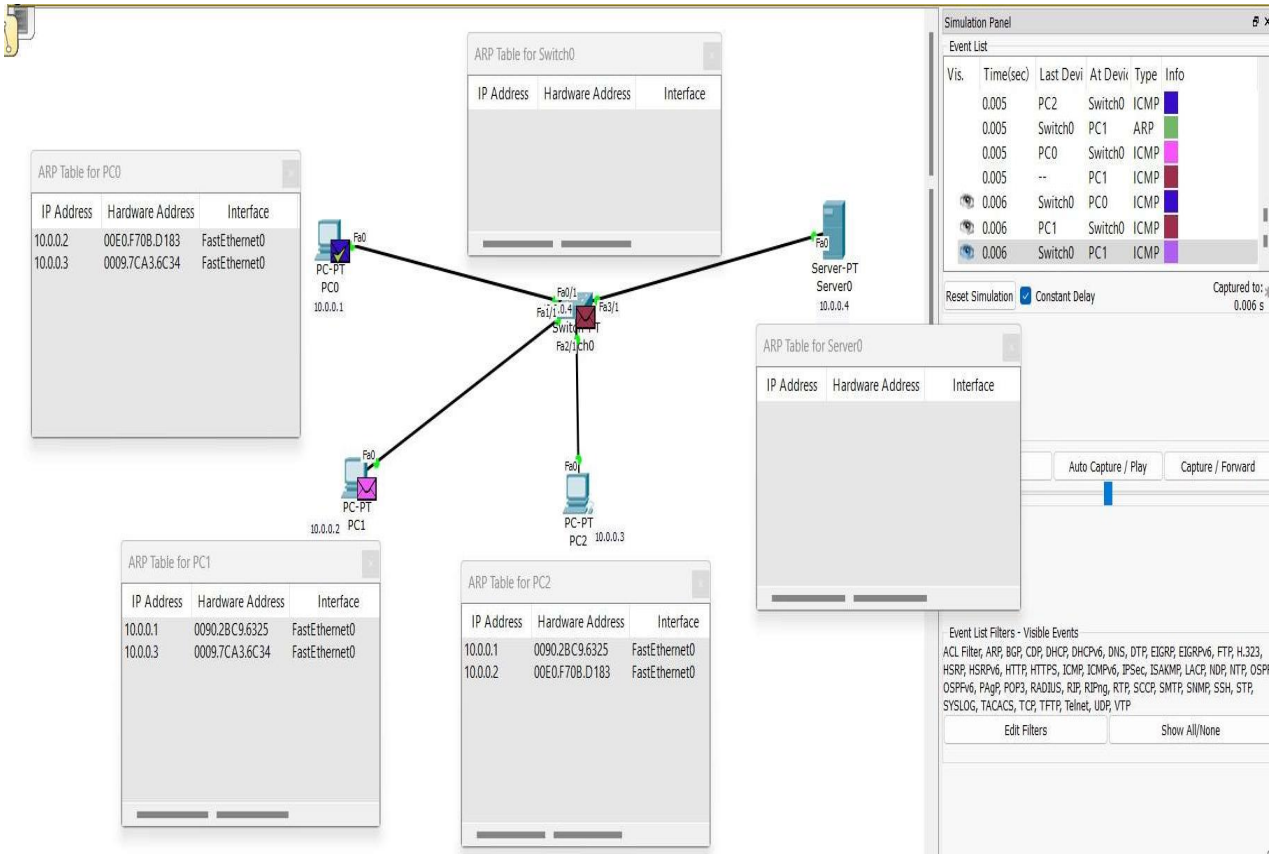
- Place three end devices, a server and a switch and connect the PC's and the server to the switch using copper straight wires.
- Use the inspect tool to click on a PC to view the ARP table.
- The same can also be viewed in the command prompt by using 'arp -a'.
- Go to the CLI of the switch and ~~click~~ do show mac address.
- Similarly obtain the ARP table of the server and other end devices.
- Enter the simulation mode and click on 'capture' by selecting PC1 and PC2 for simple PDU's.



OBSERVATION:

- Initially the ARP tables of all end devices are empty.
- The MAC address table is also found to be empty
- When the 'capture' button is clicked, it is found that the ARP table is updated in PC2 with the IP address of PC1 (10.0.0.2)
- Once the acknowledgement is obtained, the ARP table of PC1 is updated with the IP address of PC2 (10.0.0.3)
- The event list in the simulation panel shows the corresponding protocol used during the communication.

Observation Screenshot:



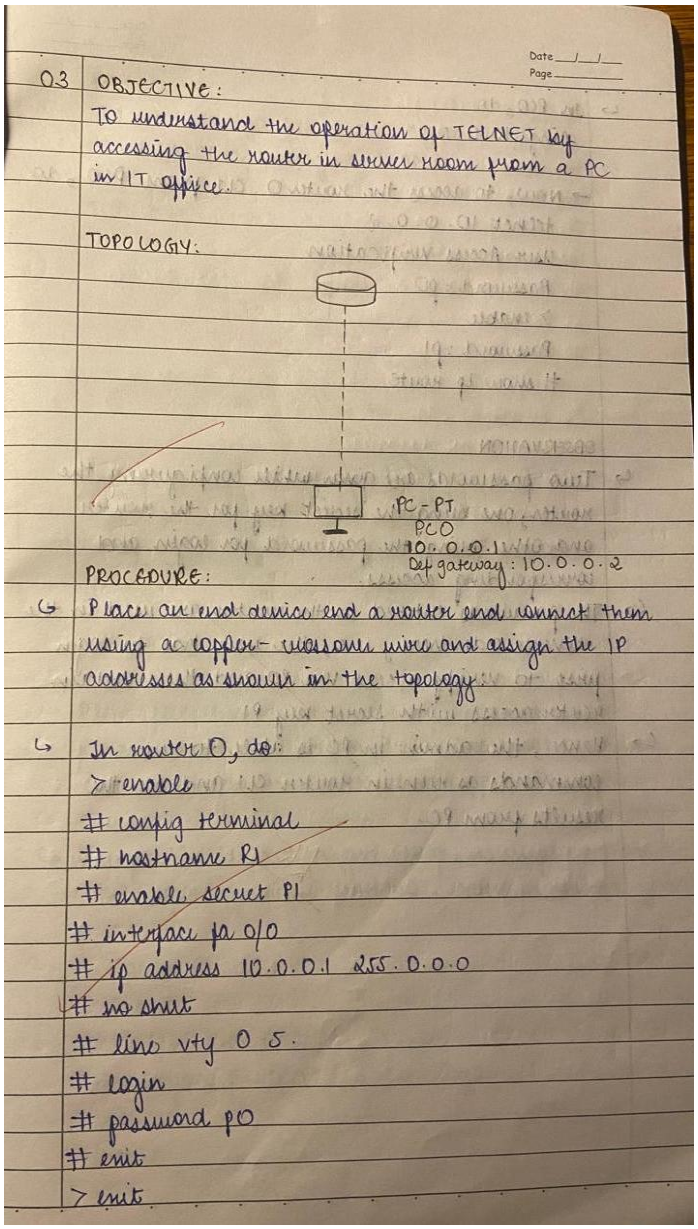

```
Switch>show mac address-table
          Mac Address Table
-------------------------------------------

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:



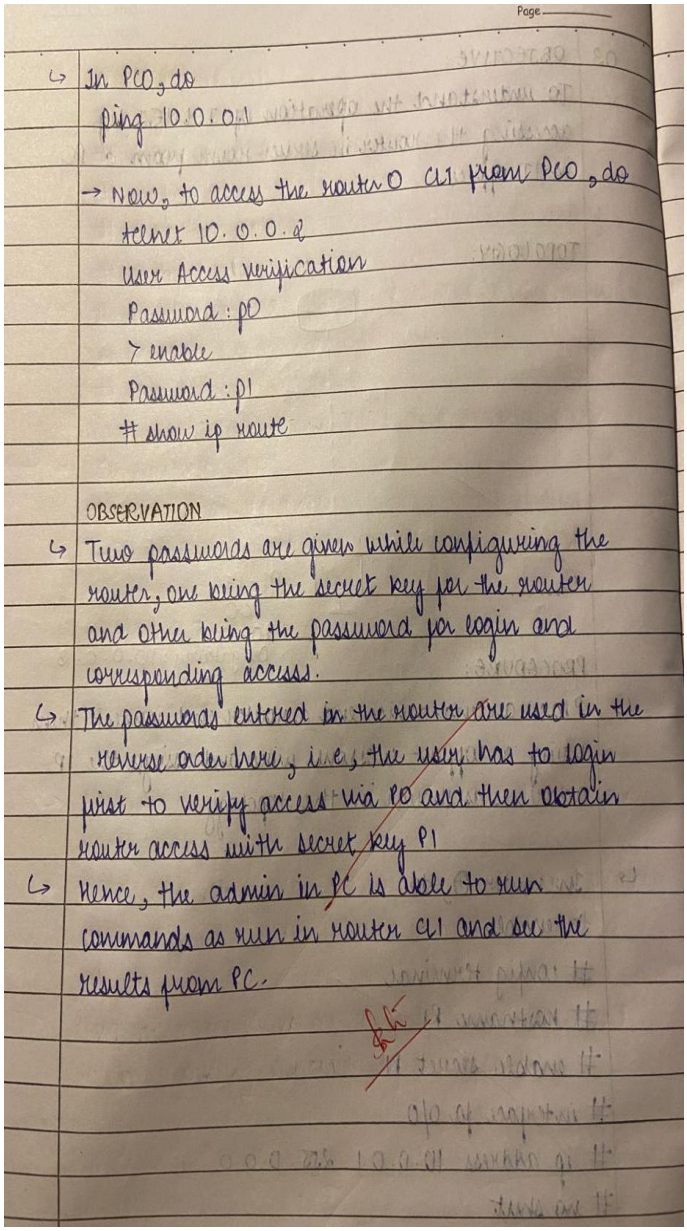
Date / /
Page

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

TOPOLOGY:

PROCEDURE:
↳ Place an end device and a router and connect them using a copper-crossover wire and assign the IP addresses as shown in the topology.

↳ In router 0, do:
> enable
config terminal
hostname R1
enable secret P1
interface fa 0/0
ip address 10.0.0.1 255.0.0.0
no shut
line vty 0 5.
login
password p0
exit
> exit

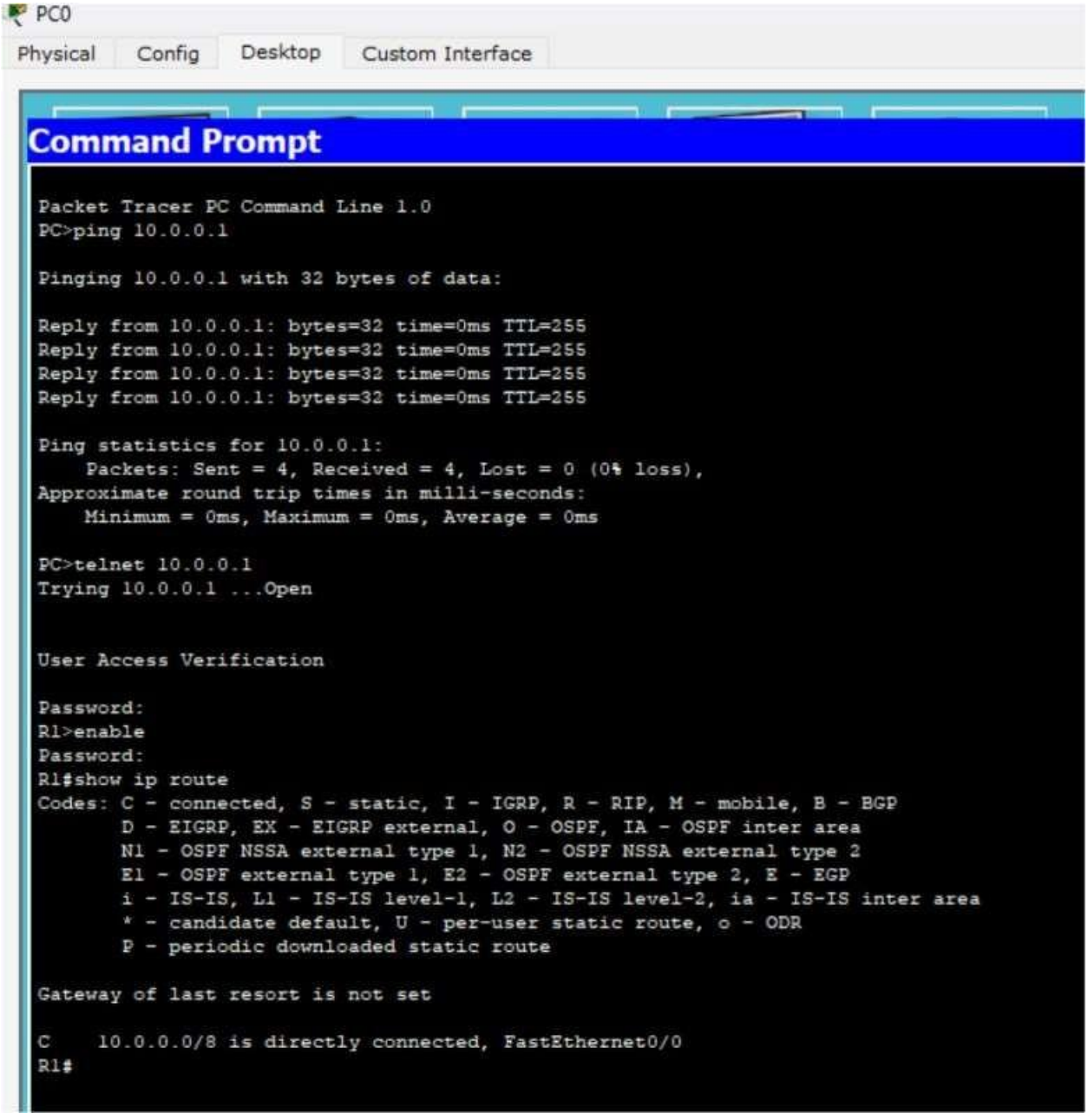


↳ In PC0, do
ping 10.0.0.1

→ Now, to access the router 0 CLI from PC0, do
telnet 10.0.0.2
User Access verification
Password: p0
> enable
Password: p1
show ip route

OBSERVATION
↳ Two passwords are given while configuring the router, one being the secret key for the router and other being the password for login and corresponding accsss.
↳ The passwords entered in the router are used in the reverse order here, i.e, the user has to login first to verify access via p0 and then obtain router access with secret key p1
↳ Hence, the admin in PC is able to run commands as run in router CLI and see the results from PC.

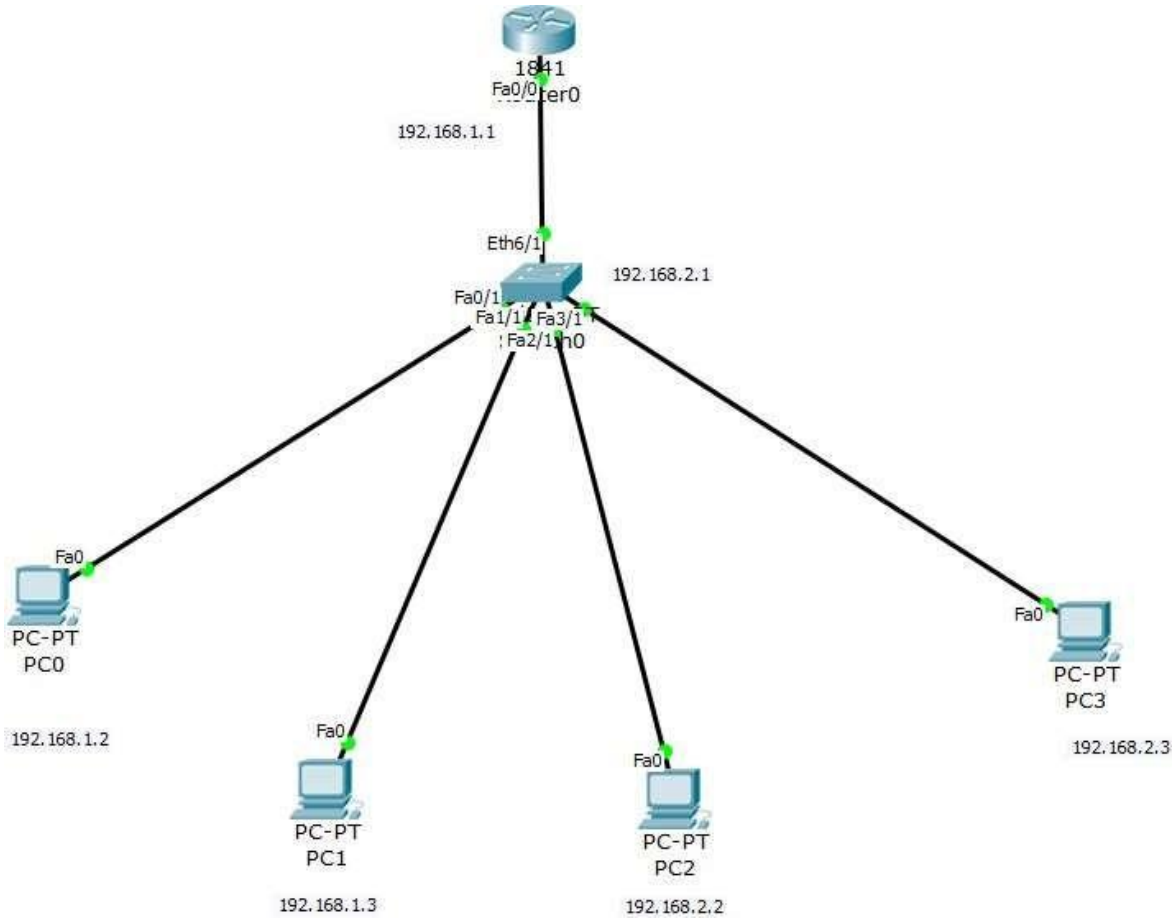
Observation Screenshot:



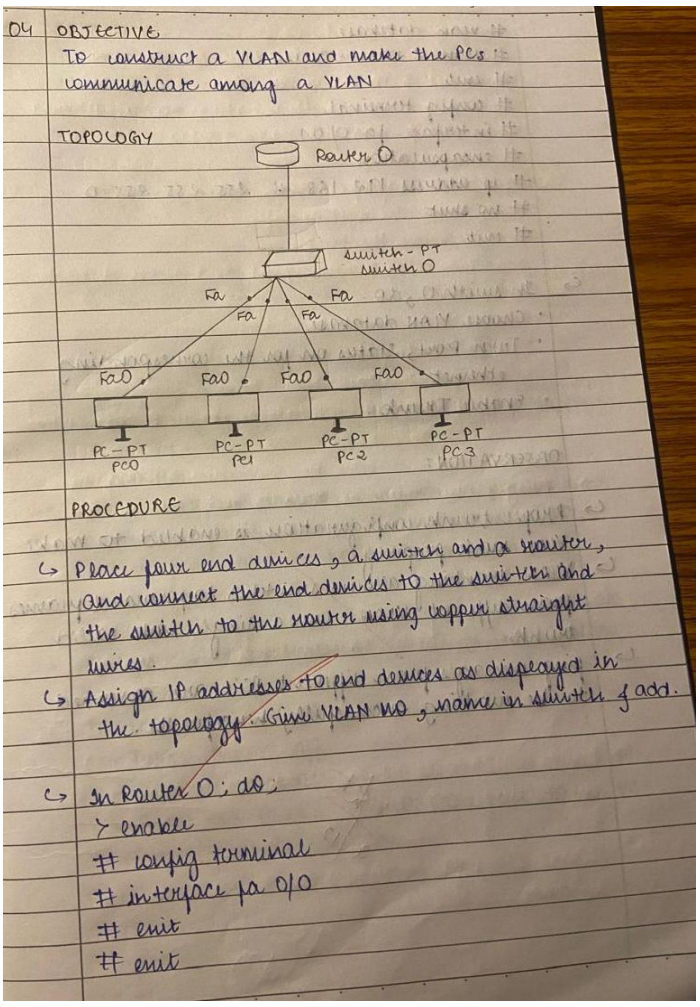
Program 11:

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

Topology:



Procedure and Observations:



04 OBJECTIVE

To construct a VLAN and make the PCs communicate among a VLAN

TOPOLOGY

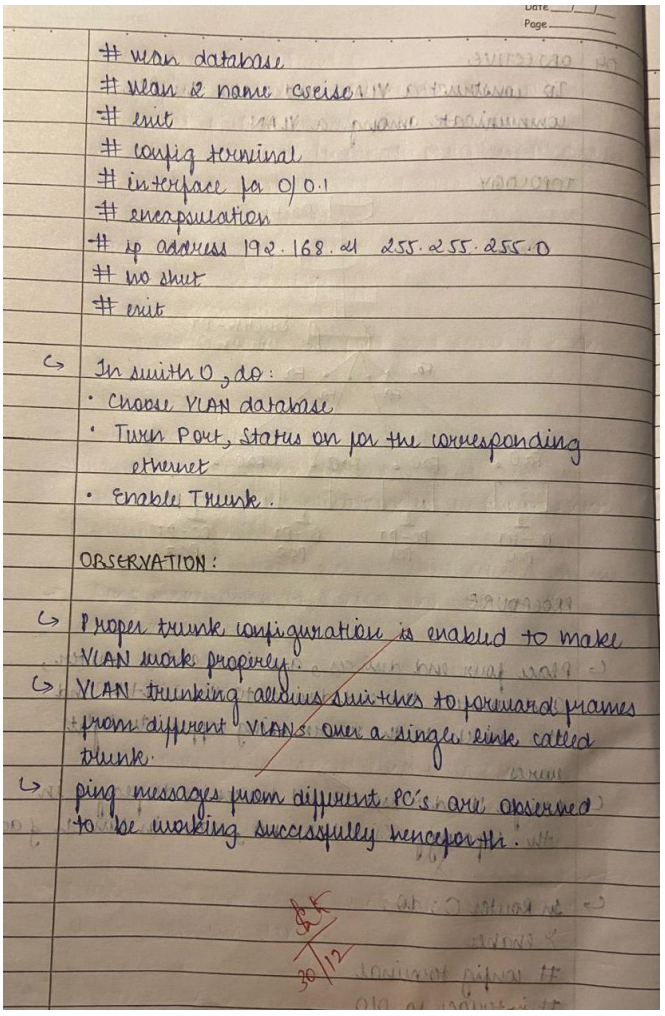
PROCEDURE

↳ Place four end devices, a switch and a router, and connect the end devices to the switch and the switch to the router using copper straight wires.

↳ Assign IP addresses to end devices as displayed in the topology. Give VLAN no, name in switch if add.

↳ In Router 0; do;

> enable
config terminal
interface fa 0/0
exit
exit



vlan database
vlan 2 name cseiso
exit
config terminal
interface fa 0/0.1
encapsulation
ip address 192.168.21 255.255.255.0
no shut
exit

↳ In switch 0, do:

- Choose VLAN database
- Turn Port, Status on for the corresponding ethernet
- Enable Trunk.

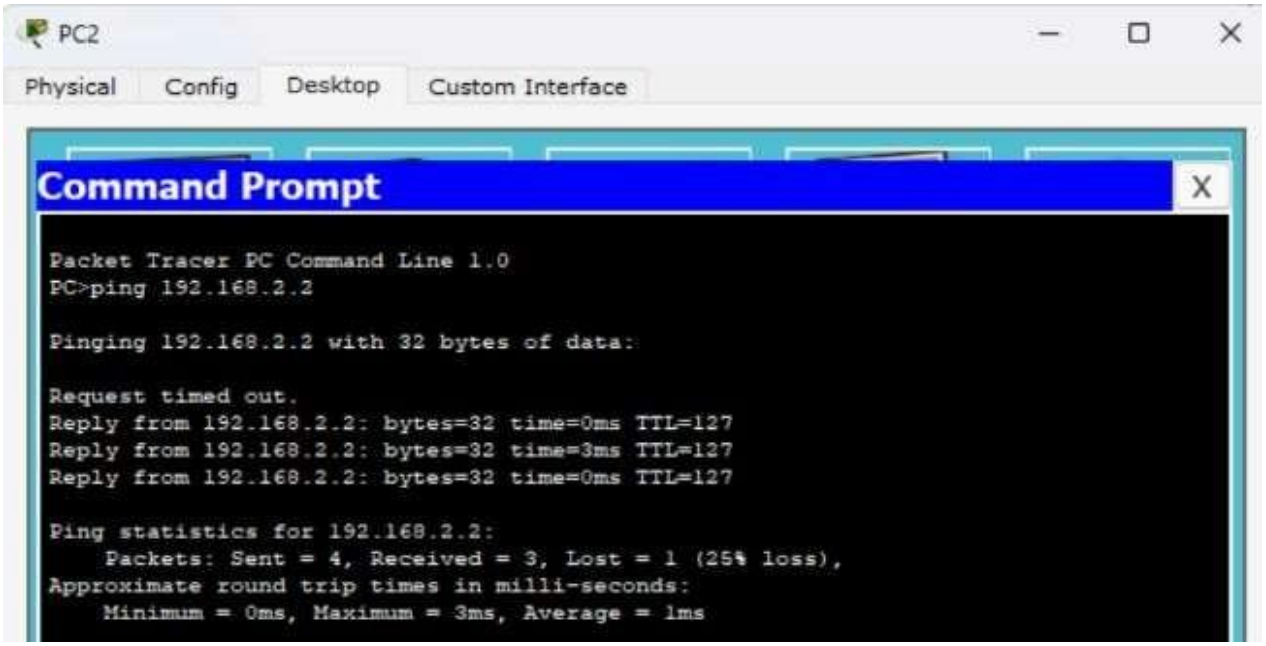
OBSERVATION:

↳ Proper trunk configuration is enabled to make VLAN work properly.

↳ VLAN trunking allows switches to forward frames from different VLANs over a single link called trunk.

↳ ping messages from different PC's are observed to be working successfully henceforth.

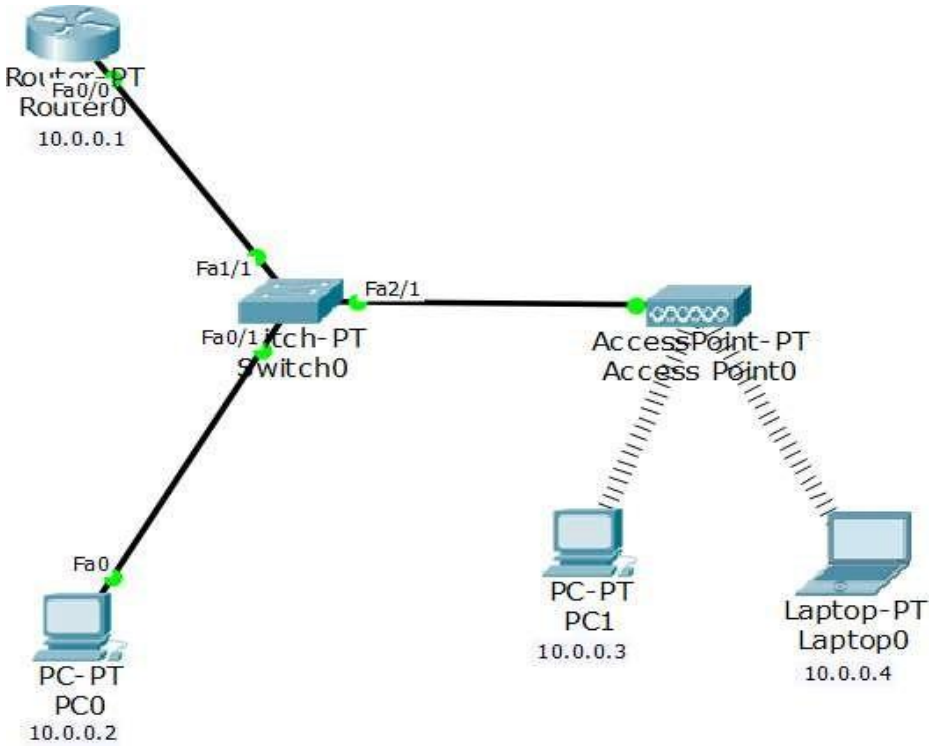
30/12



Program 12:

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

Topology:

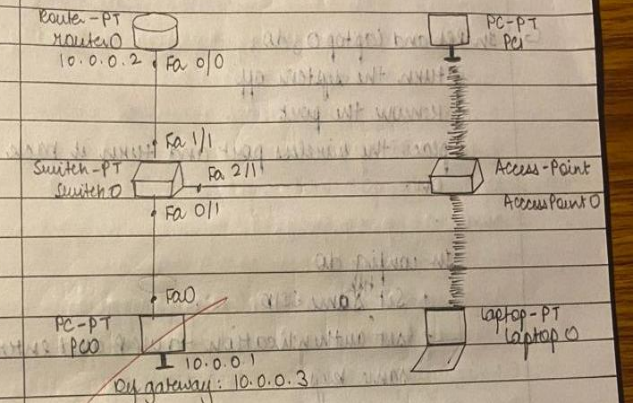


Procedure and Observations:

05 OBJECTIVE

To construct a WLAN and make the nodes communicate wirelessly.

TOPOLOGY



PROCEDURE

↳ Place three end devices, a switch, a router and an access-point. Connect the end device PC0, access-point and the Router0 to switch 0 using copper-straight wire

↳ Assign the IP addresses as shown in the topology.

↳ In PC0, do:
- Turn the PC off
- Remove the port
- Place the Linksys-WMP300N port to the PC and turn it back on.

↳ Configure Access Point 0:
- Port status should be set to 'ON'.
- Set SSID name as 'BMSCE CSECN'.
- Set channel authentication to 'WEP' and set key as '1234567890'.

↳ In PC1 and Laptop 0, do:
- Turn the system off
- Remove the port
- Place the wireless port and turn it back on.

In config do
- Set the same SSID
- Set authentication to WEP and enter same keys.

↳ Ping from different devices and observe the transmissions.

OBSERVATIONS:

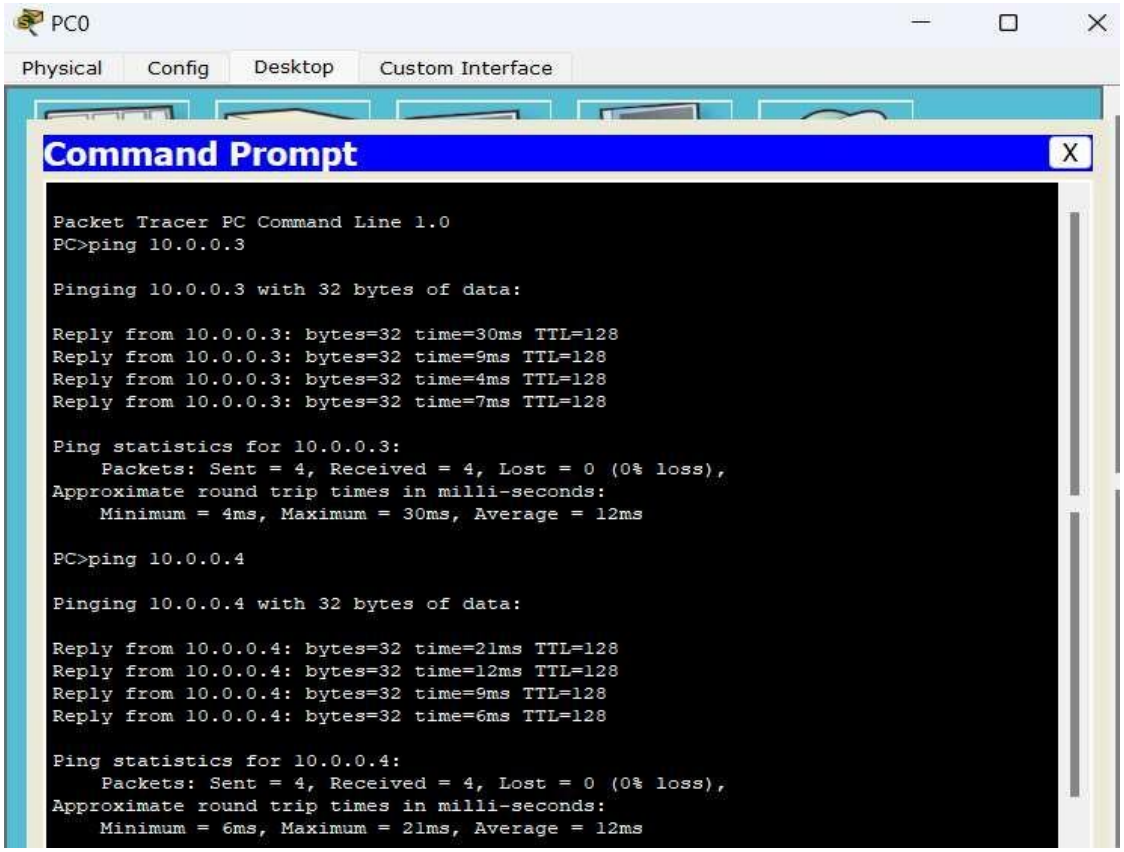
↳ After the setup of PC1 and Laptop 0, wireless connections with dashed lines were observed in connection with Access Point 0, indicating successful wireless connections.

↳ Device could connect to WLAN since they were in the network range.

→ Signal strength decreases with increase in distance.

30/12/24

Output Screenshot:



CYCLE - 2

Program 13:

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

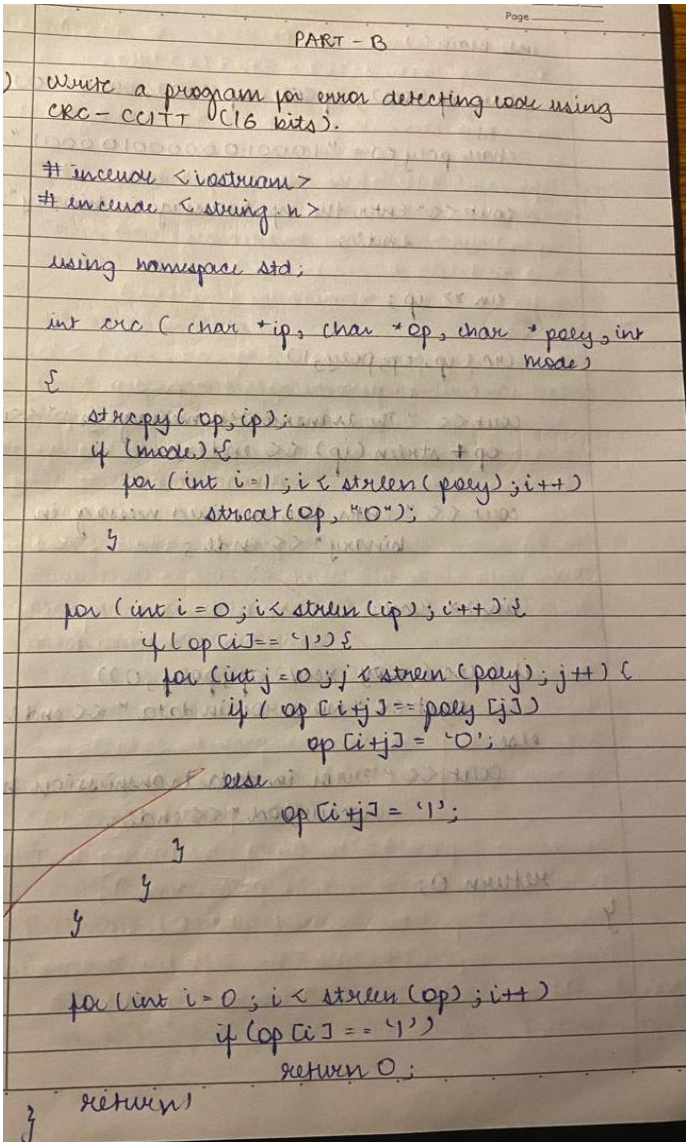
```
#include <iostream>
#include <string.h>
using namespace std;

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;
```

}

Output:

Enter the input message in binary: 110010
The transmitted message is: 11001000011011000010001
Enter the received message in binary: 101011
Error in data transmission has occurred

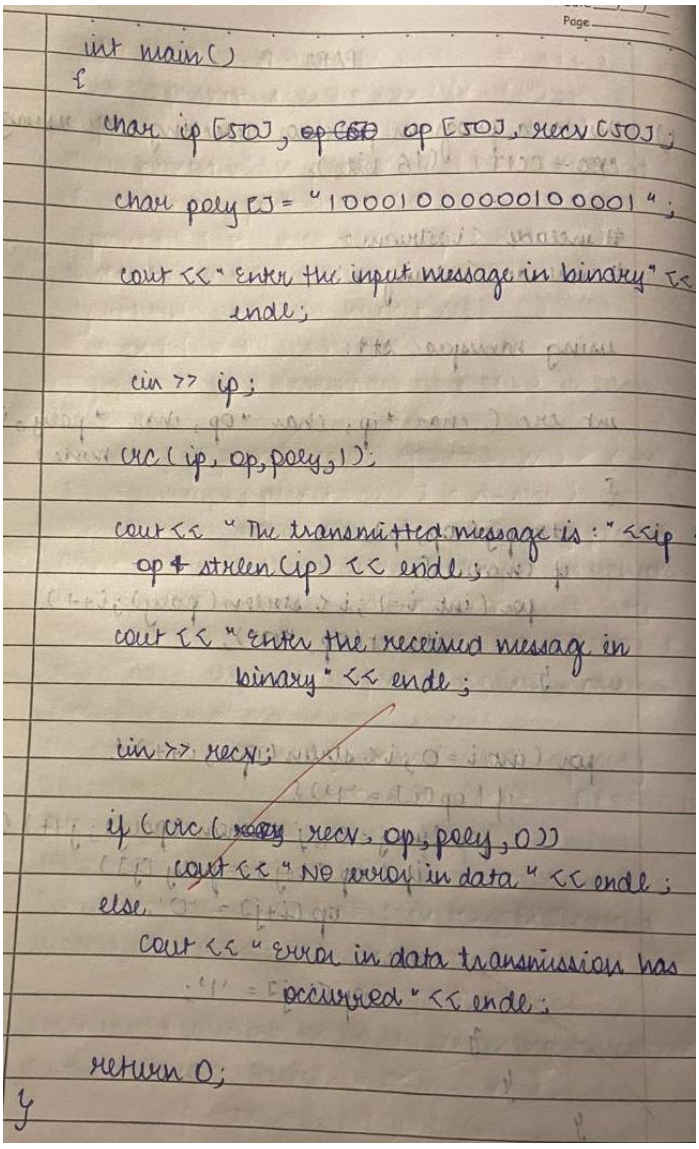
Observations:



PART - B

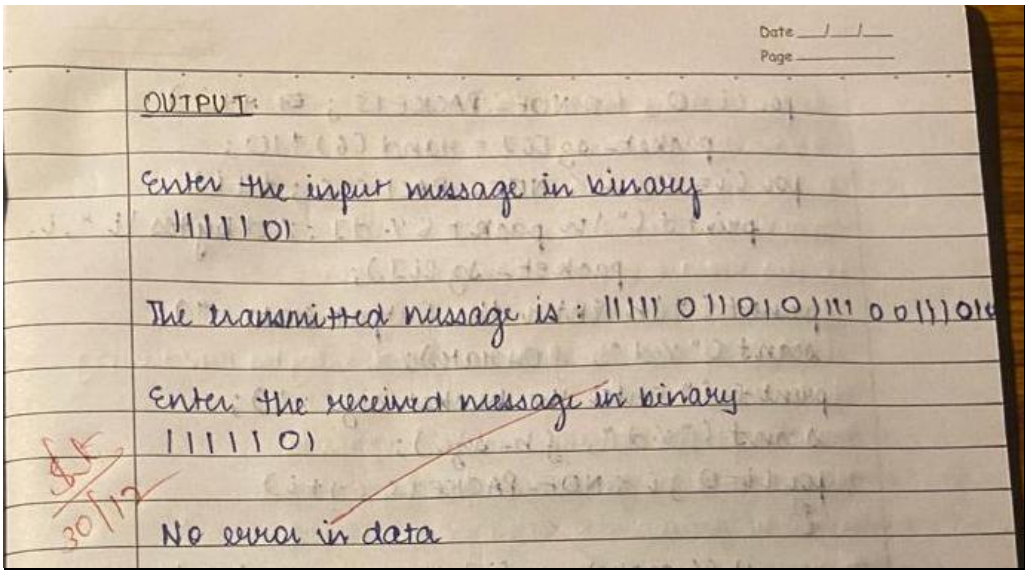
Write a program for error detecting code using CRC - CCITT (16 bits).

```
#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");
   }

   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';
         }
      }
   }

   for (int i = 0 ; i < strlen (op) ; i++)
      if (op [i] == '1')
         return 0;
   return 1;
}
```



```
int main()
{
   char ip [50], op [50], recv [50];

   char poly [] = "10001000000100001";

   cout << "enter the input message in binary" << endl;

   cin >> ip;

   crc ( ip, op, poly, 1);

   cout << "The transmitted message is :" << ip
      op + strlen (ip) << endl;

   cout << "enter the 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;
}
```

Output Screenshot:



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 <cstdlib> // for rand() and srand()
#include <unistd.h> // for sleep()
#define NOF_PACKETS 10 // Number of packets

using namespace std;

// Function to generate a random packet size
int rand(int a) {
   int rn = (random() % 10) % a;
   return rn == 0 ? 1 : rn; // Ensure a minimum value of 1
}

int main() {
   int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm = 0, op;

   // Generate random packet sizes
   for (i = 0; i < NOF_PACKETS; ++i)
      packet_sz[i] = rand(6) * 10;

   // Display generated packet sizes
   for (i = 0; i < NOF_PACKETS; ++i)
      printf("\npacket[%d]: %d bytes", i, packet_sz[i]);

   // Input bucket size and output rate
   cout << "\nEnter the Output rate: ";
   cin >> o_rate;
   cout << "Enter the Bucket Size: ";
   cin >> b_size;
```

```
    // Process each packet
    for (i = 0; i < NOF_PACKETS; ++i) {
        if ((packet_sz[i] + p_sz_rm) > b_size) { // Check for bucket overflow
            if (packet_sz[i] > b_size) { // Packet size exceeds bucket size
                printf("\n\nIncoming packet size (%d bytes) is greater than bucket capacity (%d bytes) -
PACKET REJECTED", packet_sz[i], b_size);
            } else {
                printf("\n\nBucket capacity exceeded - PACKETS REJECTED!");
            }
        } else {
            // Add packet to the bucket
            p_sz_rm += packet_sz[i];
            printf("\n\nIncoming Packet size: %d bytes", packet_sz[i]);
            printf("\nBytes remaining to transmit: %d bytes", p_sz_rm);

            // Transmission simulation
            int 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) { // Check if remaining size can be transmitted
                        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("\tBytes remaining to transmit: %d", p_sz_rm);
                } else {
                    printf("\nTime left for transmission: %d units", p_time - clk);
                    printf("\nNo packets to transmit!!");
                    break;
                }
            }
        }
    }
    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

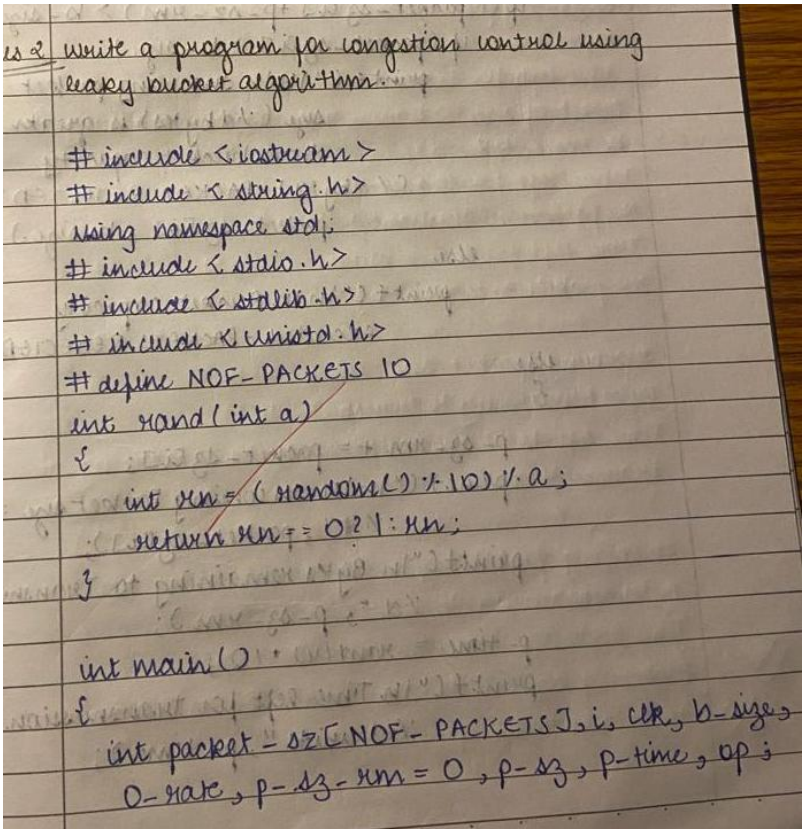
Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 30 units
Packet of size 30 Transmitted -- Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

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!!

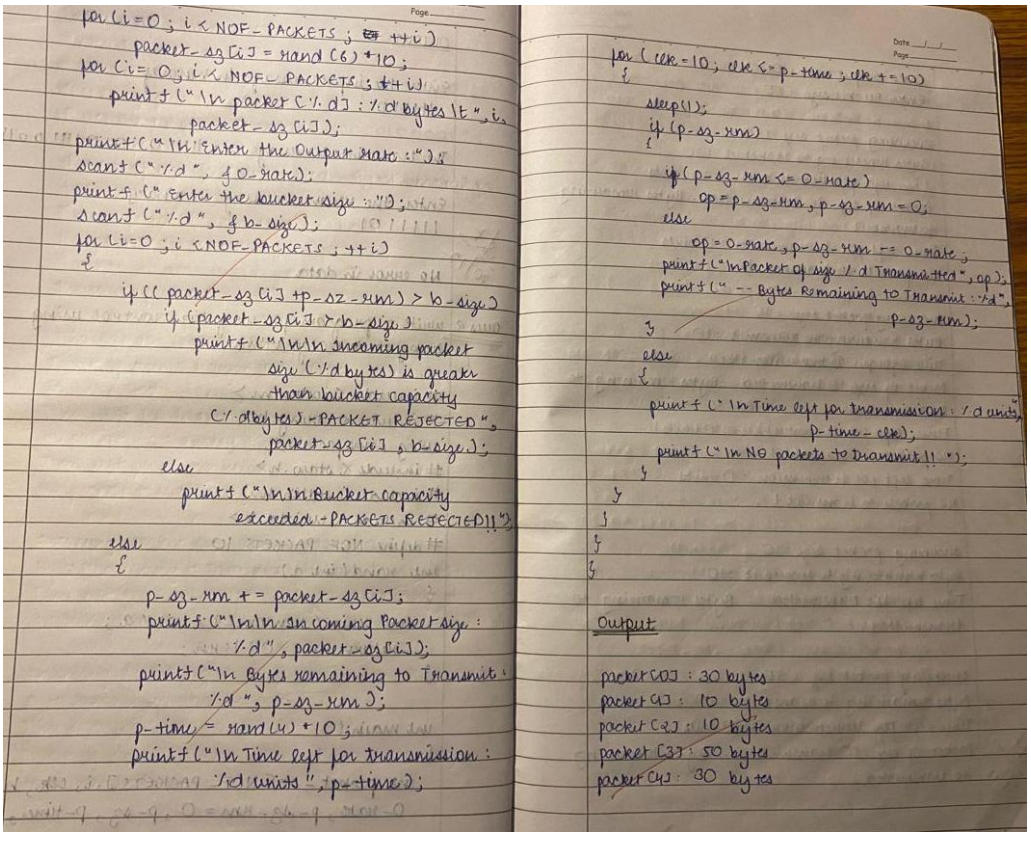
Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
Packet of size 10 Transmitted -- Bytes Remaining to Transmit: 0
Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 20 units
Packet of size 20 Transmitted -- Bytes Remaining 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!!

Observation:



```
Q2 write a program for congestion control using
leaky bucket algorithm.

#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("\n packet[%d] : %d bytes \t", i,
          packet_sz[i]);
printf("\n Enter 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)
      printf("\n\n Incoming packet
              size (%d bytes) is greater
              than bucket capacity
        (%d bytes) - PACKET REJECTED",
              packet_sz[i], b_size);
    else
      printf("\n\n Bucket capacity
              exceeded - PACKETS REJECTED!!");
  else
  {
    p_sz_rm += packet_sz[i];
    printf("\n\n Incoming Packet size :
            %d", packet_sz[i]);
    printf("\n Bytes remaining to Transmit :
            %d", p_sz_rm);
    p_time = rand(4) * 10;
    printf("\n Time 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)
      op = p_sz_rm, p_sz_rm = 0;
    else
      op = o_rate, p_sz_rm -= o_rate;
    printf("\n Packet of size %d Transmitted", op);
    printf("\n -- Bytes Remaining to Transmit : %d",
            p_sz_rm);
  }
  else
  {
    printf("\n Time left for transmission : %d units",
            p_time - clk);
    printf("\n NO packets to Transmit !! ");
  }
}
}
}
```

Output

```
packet[0] : 30 bytes
packet[1] : 10 bytes
packet[2] : 10 bytes
packet[3] : 50 bytes
packet[4] : 30 bytes
```

Enter output rate : 100
Enter 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
Time left for transmission : 0 units
NO packets to transmit !

Incoming packet size : 10
Bytes remaining to transmit : 10
Time left for transmission : 30
Packet of size 10 transmitted ... Bytes remaining to transmitted
Time left for transmission : 10 units
NO packet to transmit !
Time left for transmission : 0 units
NO packets to transmit !

Incoming packet size : 10
Bytes remaining to transmit : 10
Time left for transmitted .... Bytes remaining to transmitted.

Incoming packet size : 50
Bytes remaining to transmit : 50
Time left for transmission : 10 units
Packet of size 50 transmitted .. Bytes remaining to transmitted.

30/12

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:

Client Side:

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>

int main() {
   int soc, n;
   char buffer[1024], fname[50];
   struct sockaddr_in addr;

   soc = socket(PF_INET, SOCK_STREAM, 0);
   if (soc == -1) {
      perror("Socket creation failed");
      return 1;
   }

   addr.sin_family = AF_INET;
   addr.sin_port = htons(7891);
   addr.sin_addr.s_addr = inet_addr("127.0.0.1");

   printf("\nConnecting to the server...");
   while (connect(soc, (struct sockaddr *)&addr, sizeof(addr)) != 0) {
      printf(".");
      fflush(stdout);
      sleep(1);
   }

   printf("\nClient is connected to Server");
   printf("\nEnter file name: ");
```

```
    scanf("%s", fname);

    if (send(soc, fname, sizeof(fname), 0) == -1) {
        perror("Failed to send filename");
        return 1;
    }

    printf("\nReceived response:\n");

    while ((n = recv(soc, buffer, sizeof(buffer) - 1, 0)) > 0) {
        buffer[n] = '\0';
        printf("%s", buffer);
    }

    if (n == -1) {
        perror("Failed to receive data");
    }

    close(soc);

    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:

Server Side:

```
#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];
```

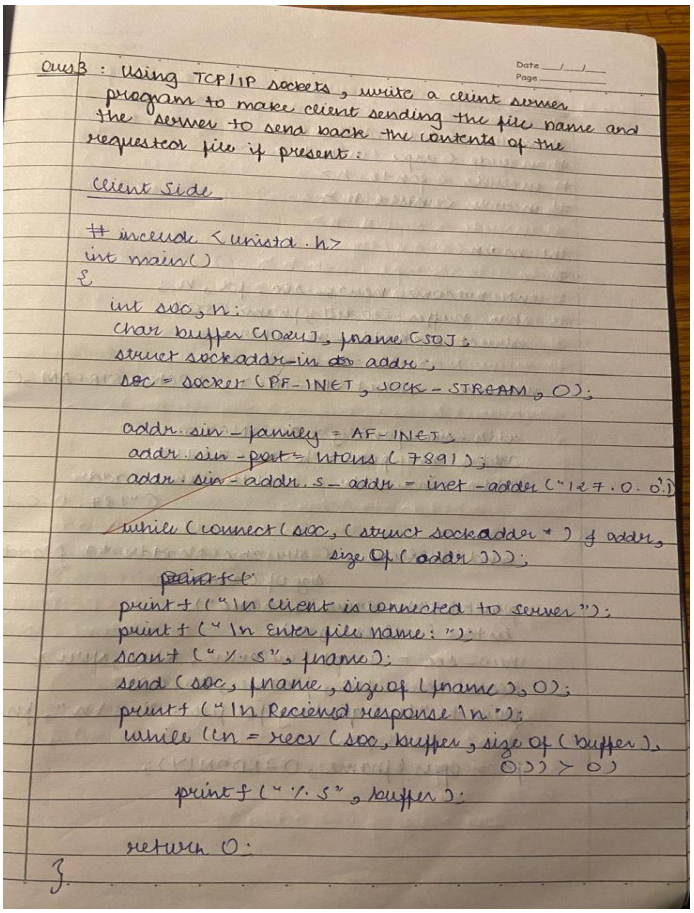
```
    struct sockaddr_in addr;
    welcome = socket(PF_INET, SOCK_STREAM,
    0); addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr =
    inet_addr("127.0.0.1");
    bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
    printf("\nServer is Online");
    /* listen 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.

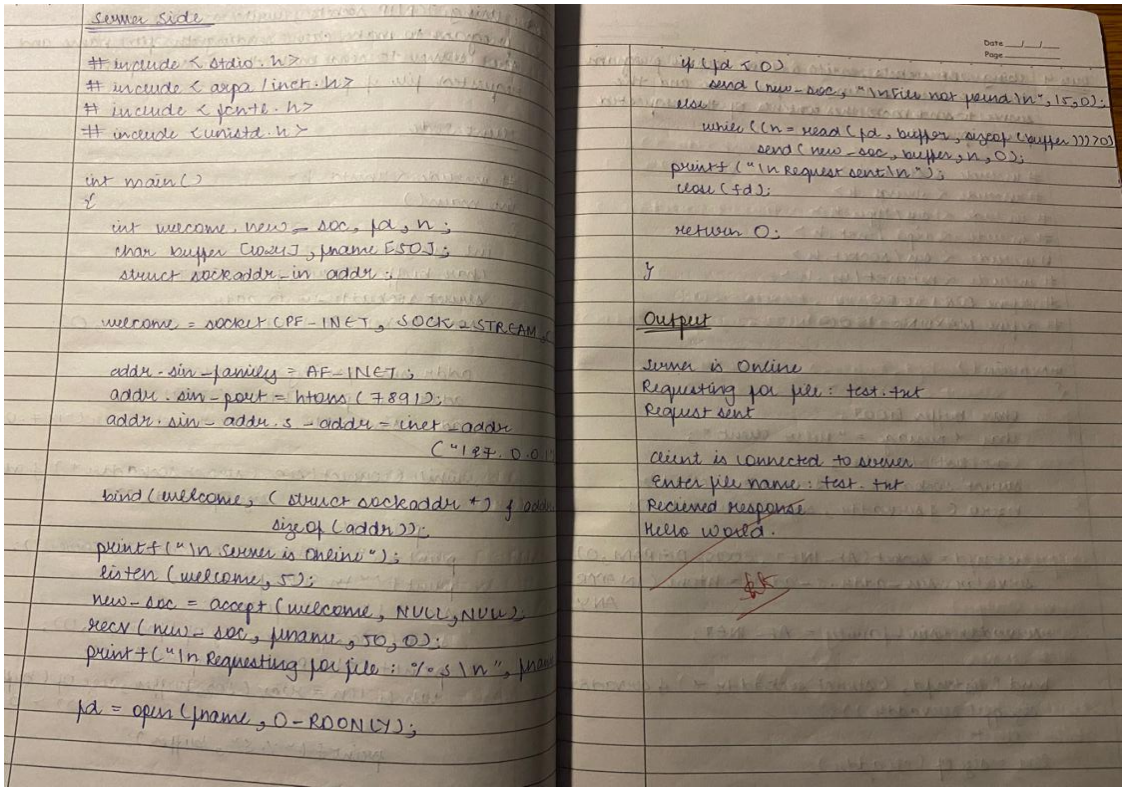
Observation:



Q.no 3 : 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.

Client Side

```
#include <unistd.h>
int main()
{
    int soc, n;
    char buffer[30], fname[50];
    struct sockaddr_in addr;
    soc = 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");

    while(connect(soc, (struct sockaddr*) &addr,
                  sizeof(addr)));
    printf("\n Client is connected to server");
    printf("\n Enter file name: ");
    scanf("%s", fname);
    send(soc, fname, sizeof(fname), 0);
    printf("\n Recieved response \n");
    while((n = recv(soc, buffer, sizeof(buffer),
                    0)) > 0)
        printf("%s", buffer);

    return 0;
}
```



Server Side

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    int welcome, new_soc, fd, n;
    char buffer[30], fname[50];
    struct sockaddr_in addr;

    welcome = socket(PF_INET, SOCK_STREAM,

    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("\n Server is online");
    listen(welcome, 5);
    new_soc = accept(welcome, NULL, NULL);
    recv(new_soc, fname, 50, 0);
    printf("\n Requesting for file: %s\n", fname);

    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("\n Request 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
Recieved 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 Side:

```
#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));
}
```

Cilent Side:

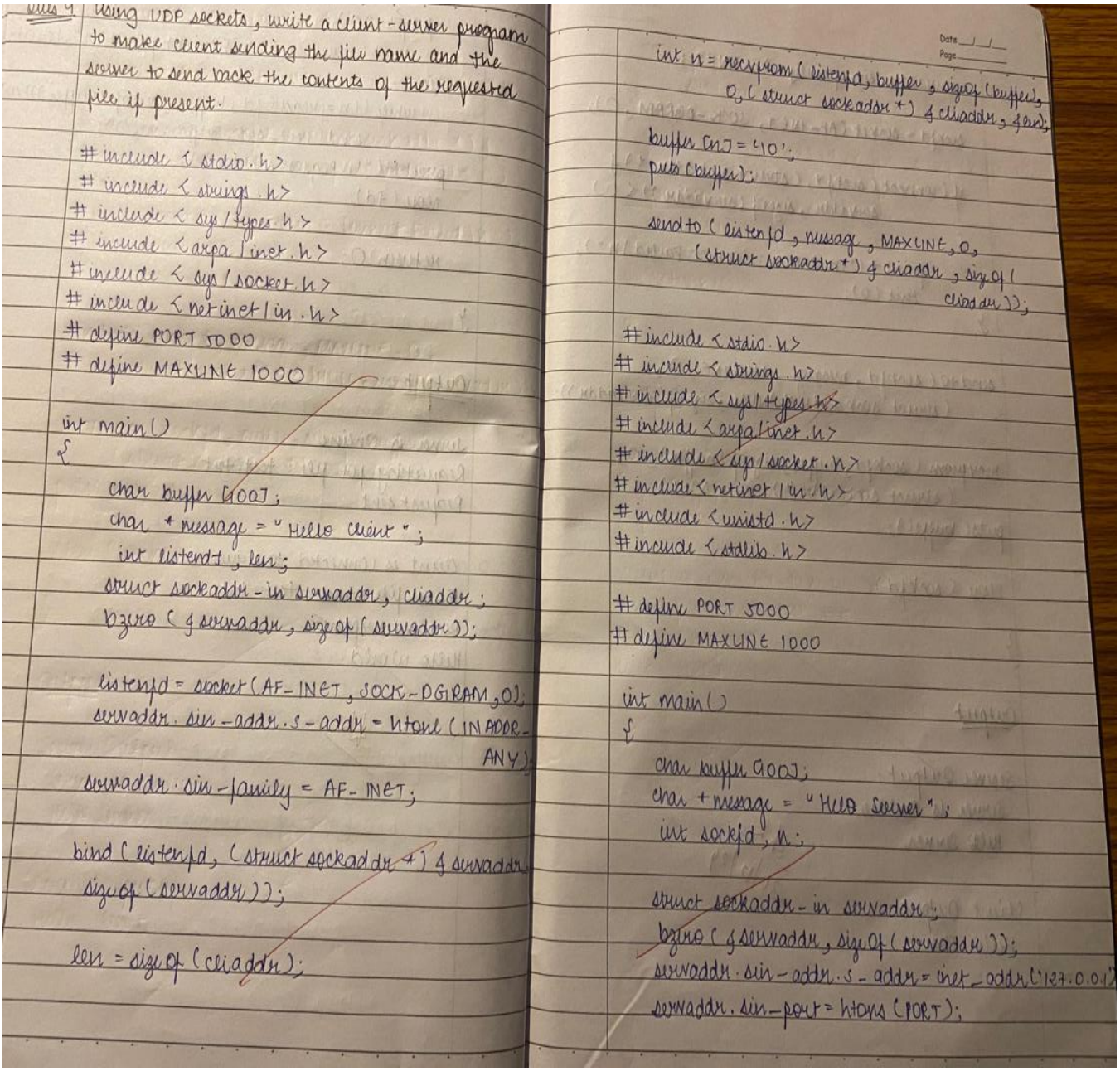
```
program #include
<stdio.h>  #include
<strings.h> #include
<sys/types.h> #include
<arpa/inet.h> #include
<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Server";
    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

Observation:



```
servaddr.sin_family = AF_INET;

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

if(connect(sockfd, (struct sockaddr *)&
        servaddr, sizeof(servaddr)) < 0)
{
    printf("\n Error : Connect Failed \n");
    exit(0);
}

sendto(sockfd, message, MAXLINE, 0,
    (struct sockaddr *)NULL, sizeof(servaddr));

recvfrom(sockfd, buffer, sizeof(buffer), 0,
    (struct sockaddr *)NULL, NULL);
puts(buffer);

close(sockfd);

}
```

Output

Server Output:
Server is Online
Hello Server

SF
30/12/24

Client Output:
Hello Client