VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Computer Networks

Submitted by

AKRAM (1BM21CS013)

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
June-2023 to September-2023

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**Computer Networks**" carried out by **AKRAM (1BM21CS013),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the academic semester June-2023 to September-2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **Computer Networks (22CS4PCCON)** work prescribed for the said degree.

Nandini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

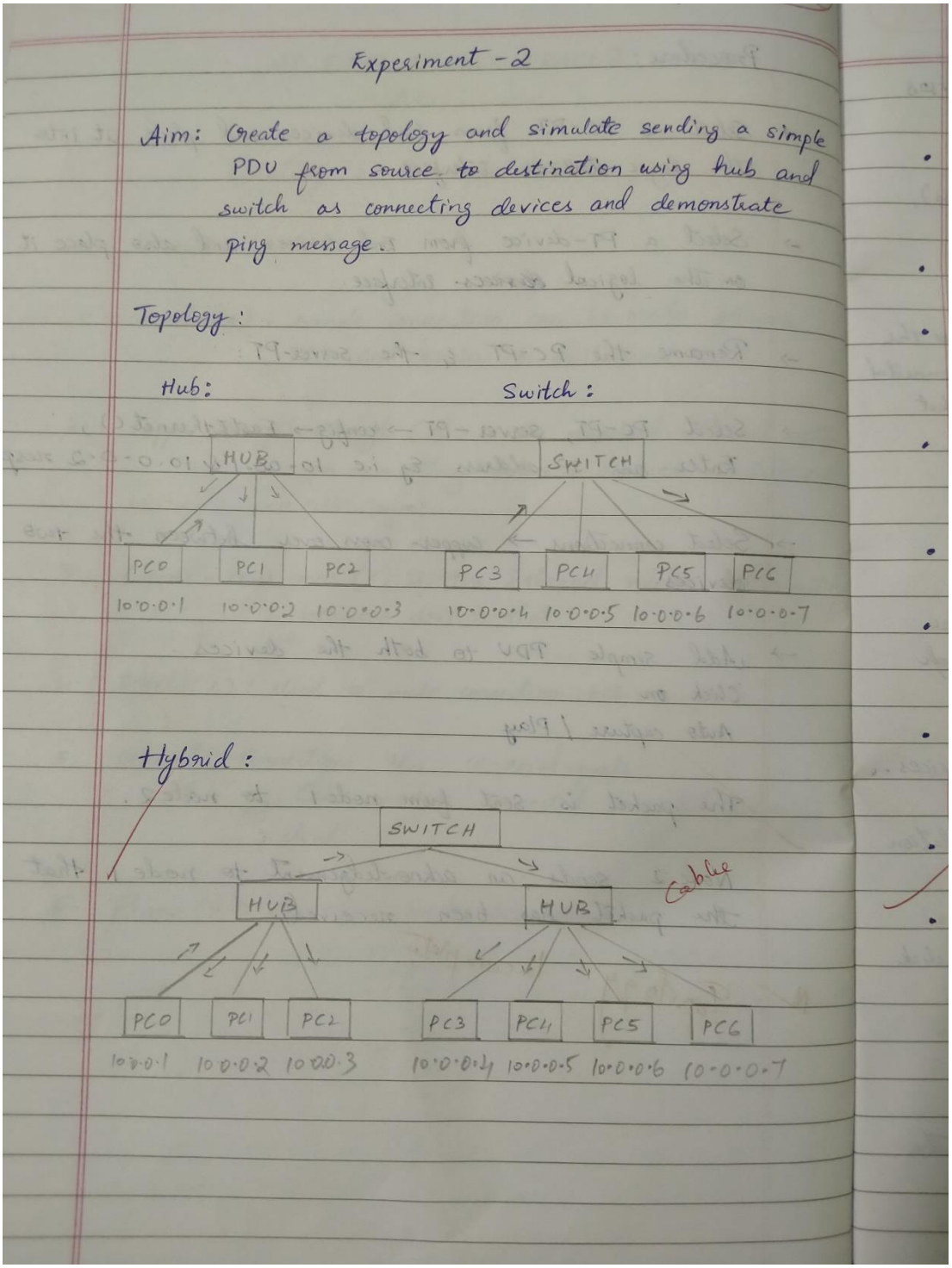
Index

Sl. No.	Date	Experiment Title	Page No.
		CYCLE 1	
1	15/6/23	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrating ping messages.	5
2	22/6/23	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	13
3	13/7/23	Configure default route, static route to the Router.	19
4	13/7/23	Configure DHCP within a LAN and outside LAN.	23
5	20/7/23	Configure Web Server, DNS within a LAN.	26
6	20/7/23	Configure RIP routing Protocol in Routers.	26
7	27/7/23	Configure OSPF routing protocol.	30
8	3/8/23	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	42
9	10/8/23	To construct a VLAN and make a pc communicate among VLAN.	49
10	10/8/23	Demonstrate the TTL/ Life of a Packet.	35
11	10/8/23	To construct a WLAN and make the nodes communicate wirelessly.	53
12	10/8/23	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	46
		CYCLE 2	
13	17/8/23	Write a program for error detecting code using CRC CCITT (16-bits).	57
14	17/8/23	Write a program for congestion control using Leaky bucket algorithm.	61
15	24/8/23	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.	64

16	24/8/23	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.	68
17	24/8/23	Tool Exploration -Wireshark	72

Experiment 1:

Hub and Switch Demo



Procedure : Hub Topology

- Select a hub from the bottom toolbar and place it on the logical interface.
- Place 3 end devices on the logical interface.
- Configure unique IP address and subnet mask on each end devices.
- Connect the end devices to the hub using the copper straight through wire.
- Send a packet from PC0 to PC2
- Start the simulation by clicking auto capture/play on the simulation tab.
- For realtime ping demonstration click on realtime tab and select a end device.
- Enter the command > ping <end device ip>
- The ping response can be viewed on the screen if the destination ip is valid.

Result : > ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL=128
Reply from 10.0.0.5 : bytes = 32 time = 0ms TTL=128
Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL=128

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

Ping statistics for 10.0.0.3 :

Packets : Sent = 4 , Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds :

Minimum = 0ms, Maximum = 0ms, Average = 0ms.

Observation: It is observed that hub broadcasts the packet from the source to all the devices connected to the hub. The end devices reject the packet except to which the packet was sent.

Procedure : Switch

- Select a switch and 4 end devices and place it on the logical interface and connect using copper through wire.
- Configure ip address and subnet mask on each end devices.
- Send a packet b/w two devices and start the simulation to visualize data/packet flow.
- For realtime ping, open realtime option and click on an end device and navigate to command prompt.
- Enter the command > ping <end device ip>
- The ping response can be viewed on the screen if the destination ip is valid.

Result :

>ping 10.0.0.8

Pinging 10.0.0.8 with 32 bytes of data:

Reply from 10.0.0.8 : bytes=32 time=1ms TTL=128
Reply from 10.0.0.8 : bytes=32 time=0ms TTL=128
Reply from 10.0.0.8 : bytes=32 time=0ms TTL=128
Reply from 10.0.0.8 : bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.8:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds.
Minimum = 0ms, Maximum = 1ms, Average = 0ms.

Observation: It is observed that unlike a hub, a switch does not broadcast the packets each and every time instead broadcasts the packets once and sends the packet to the intended user on further communications.

Procedure : Hybrid

- Select a switch and two hubs and place it on the logical interface
- Connect 3 end devices to one hub and 4 end devices to another hub.
- Connect these two hubs to the switch.
- Once ip addresses and subnet masks are configured, send a data packet from PC0 to PC6

- The packet gets broadcasted among the hub networks while it also gets transmitted from the switch to the hub to which PC6 is connected.

- For realtime simulation of ping, click on realtime and select a device and navigate to command prompt

- Enter the command > ping <end device ip>

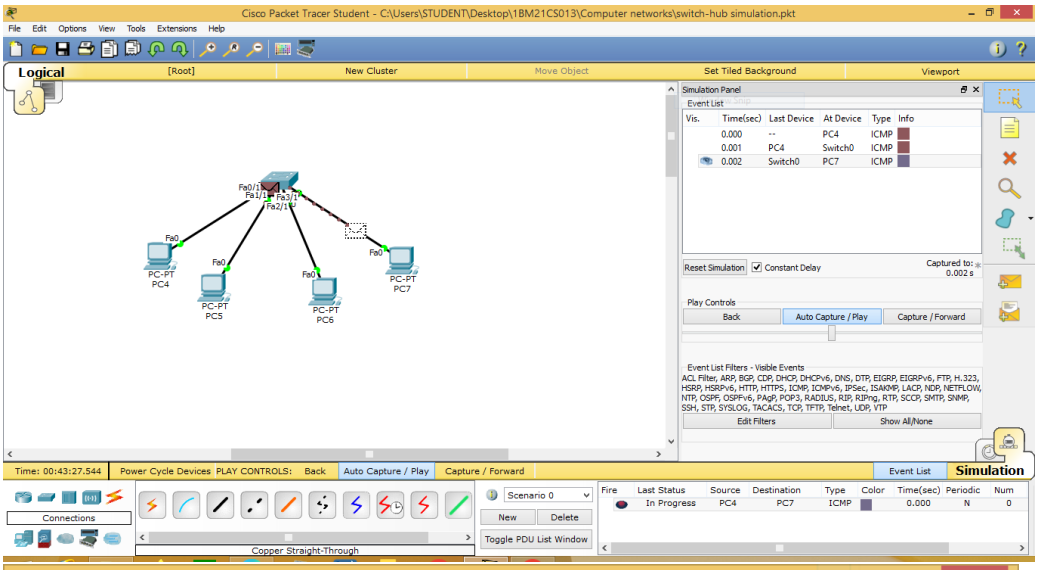
- The ping response can be viewed if the destination ip is valid.

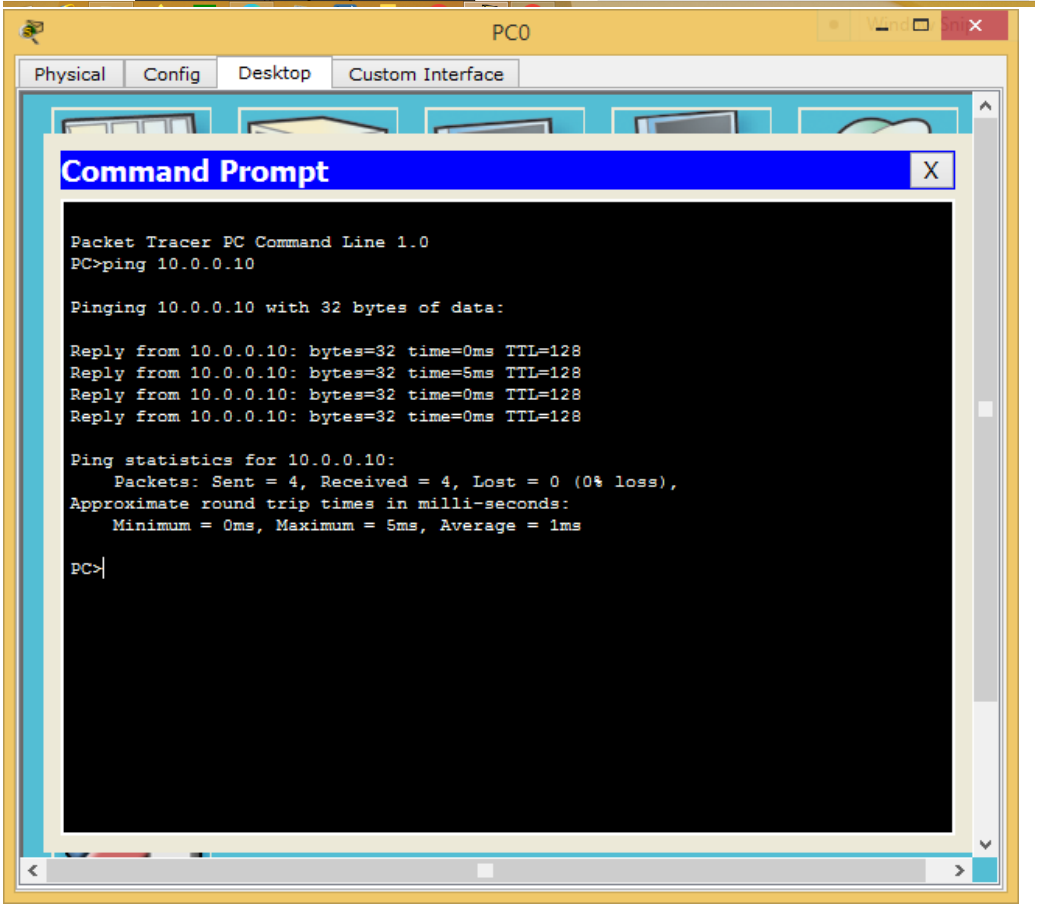
Observation: It is observed that by following a hybrid structure more flexibility can be acheived as few devices can be connected to hub which does not have much security issue while different hubs can be connected to switch which allows more number of end devices to be present in a network while ensuring security. The packets get transmitted within a hub network while the switch transmits the packets to the hub to which the intended end device is connected.

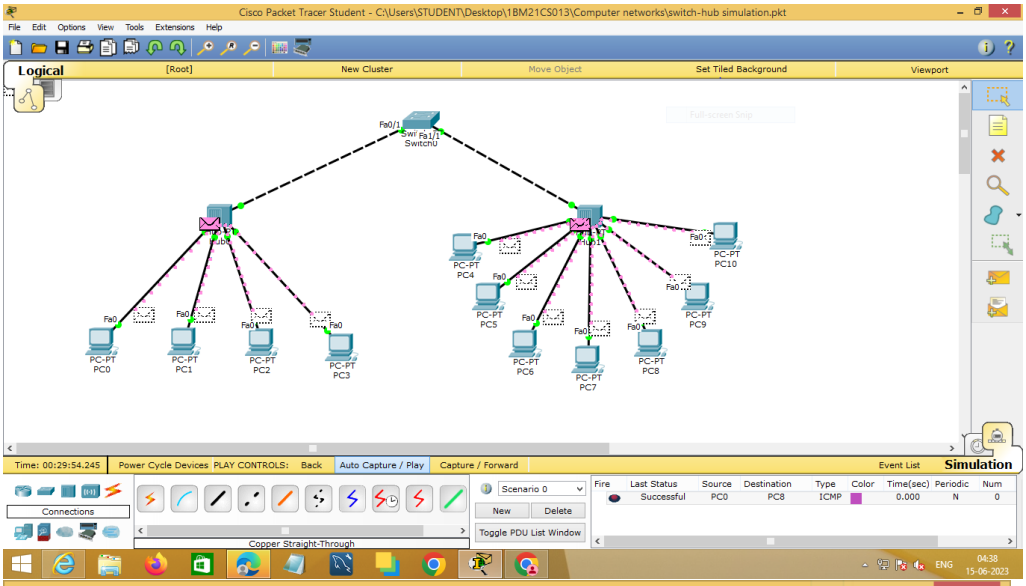
9/10

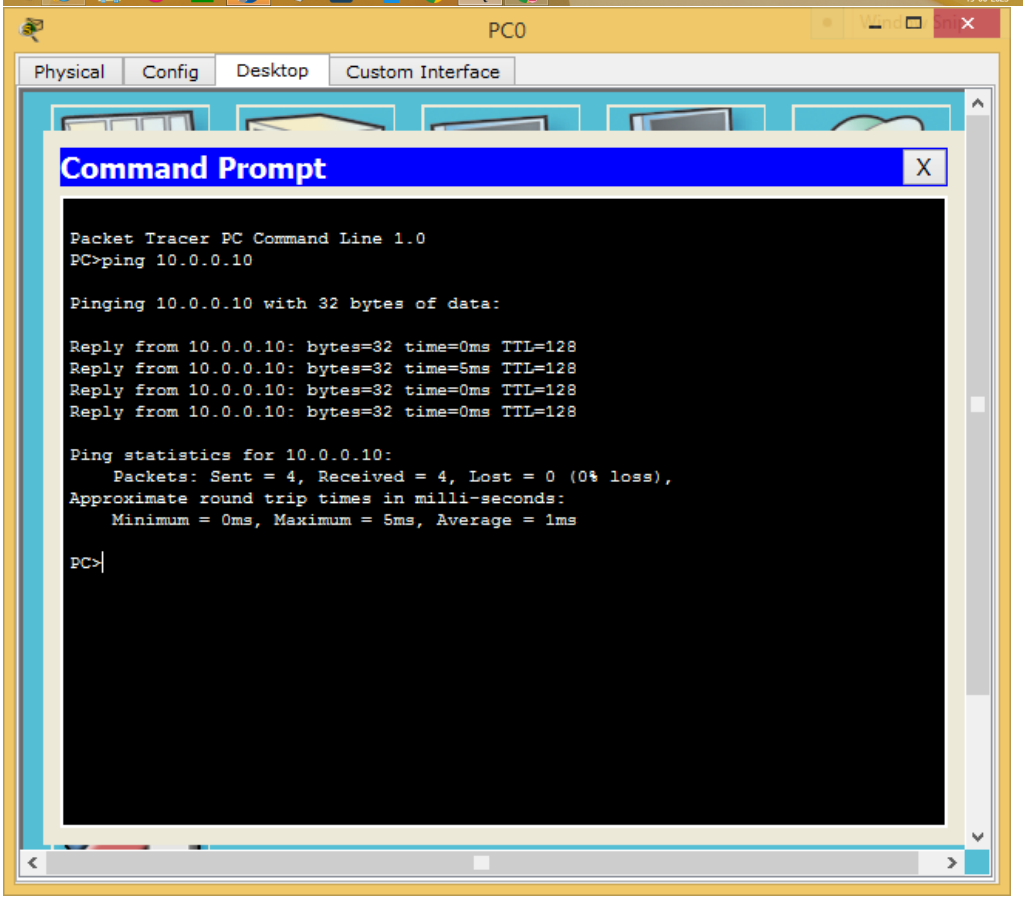
22/6/23

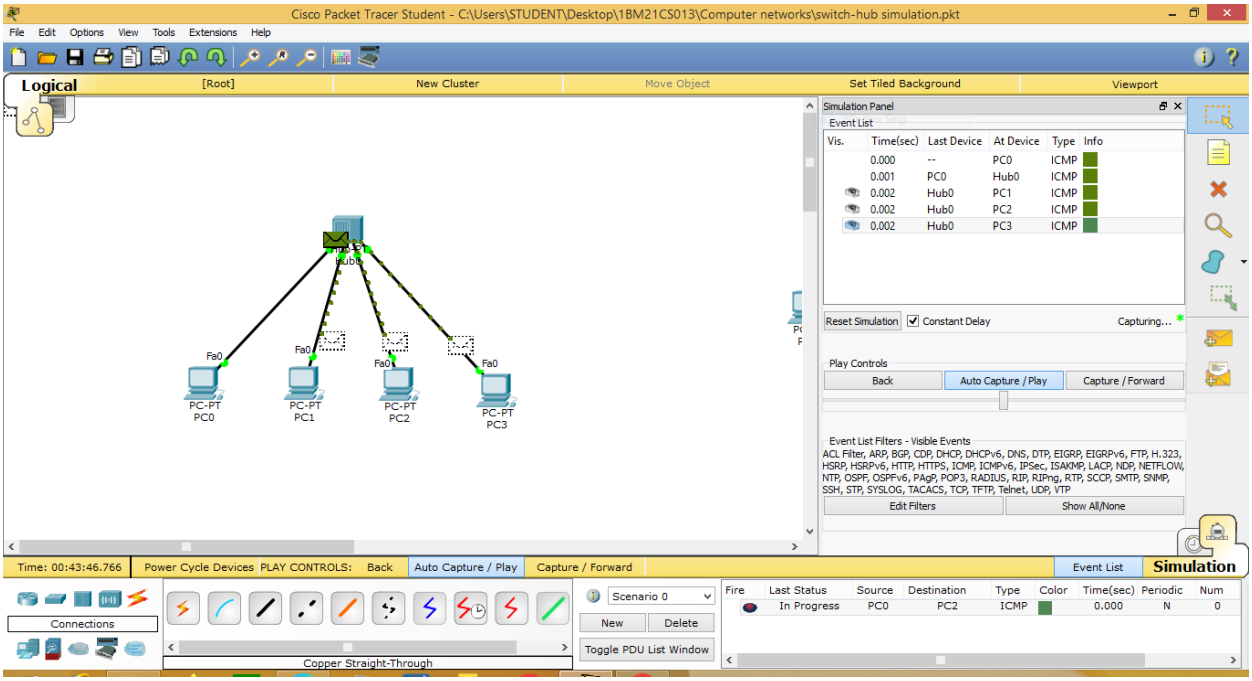
Topology and output screenshots:

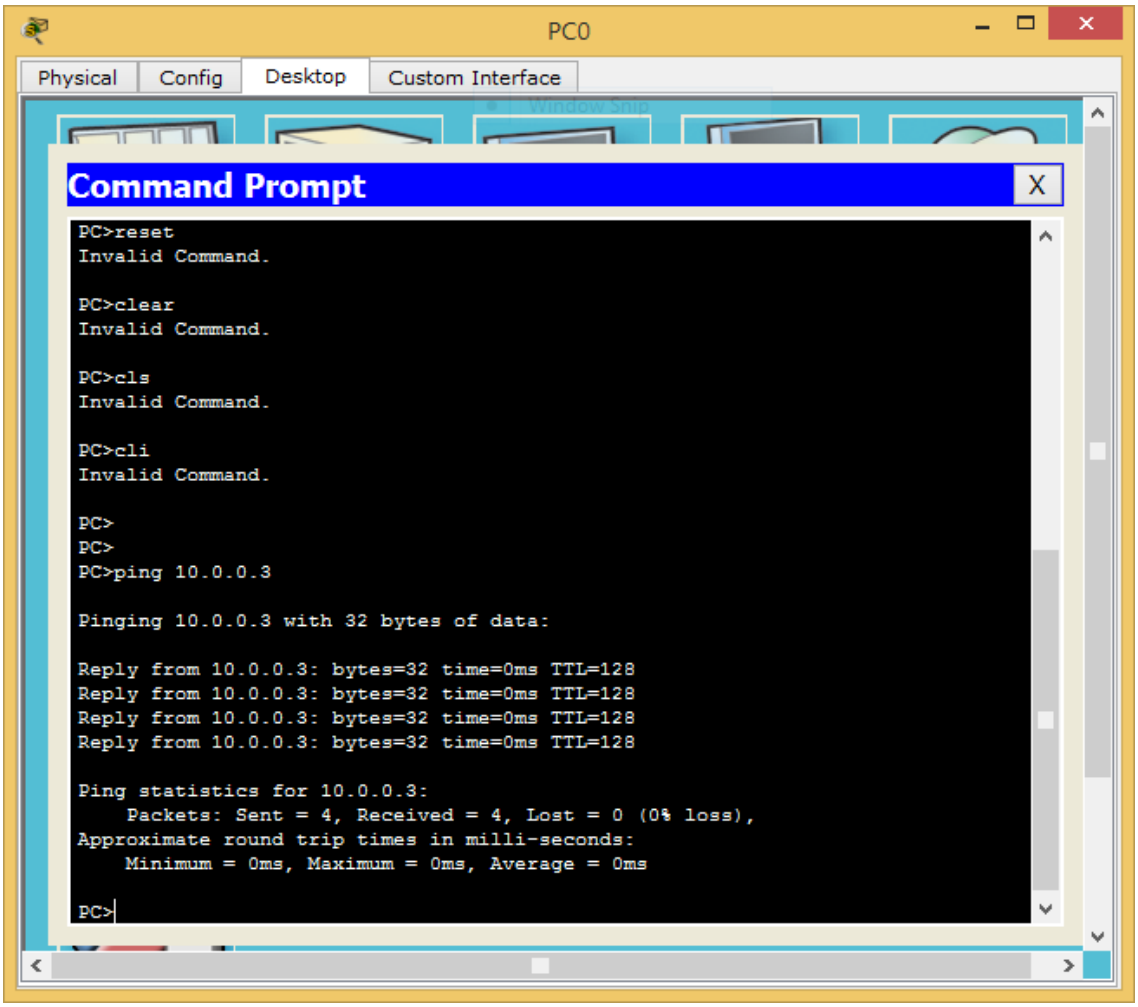





Cisco Packet Tracer Student - C:\Users\STUDENT\Desktop\1BM21CS013\Computer networks\switch-hub simulation.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Switch0
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
Time: 00:29:54.245
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Connections
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Successful
PC0
PC8
ICMP
0.000
N
0

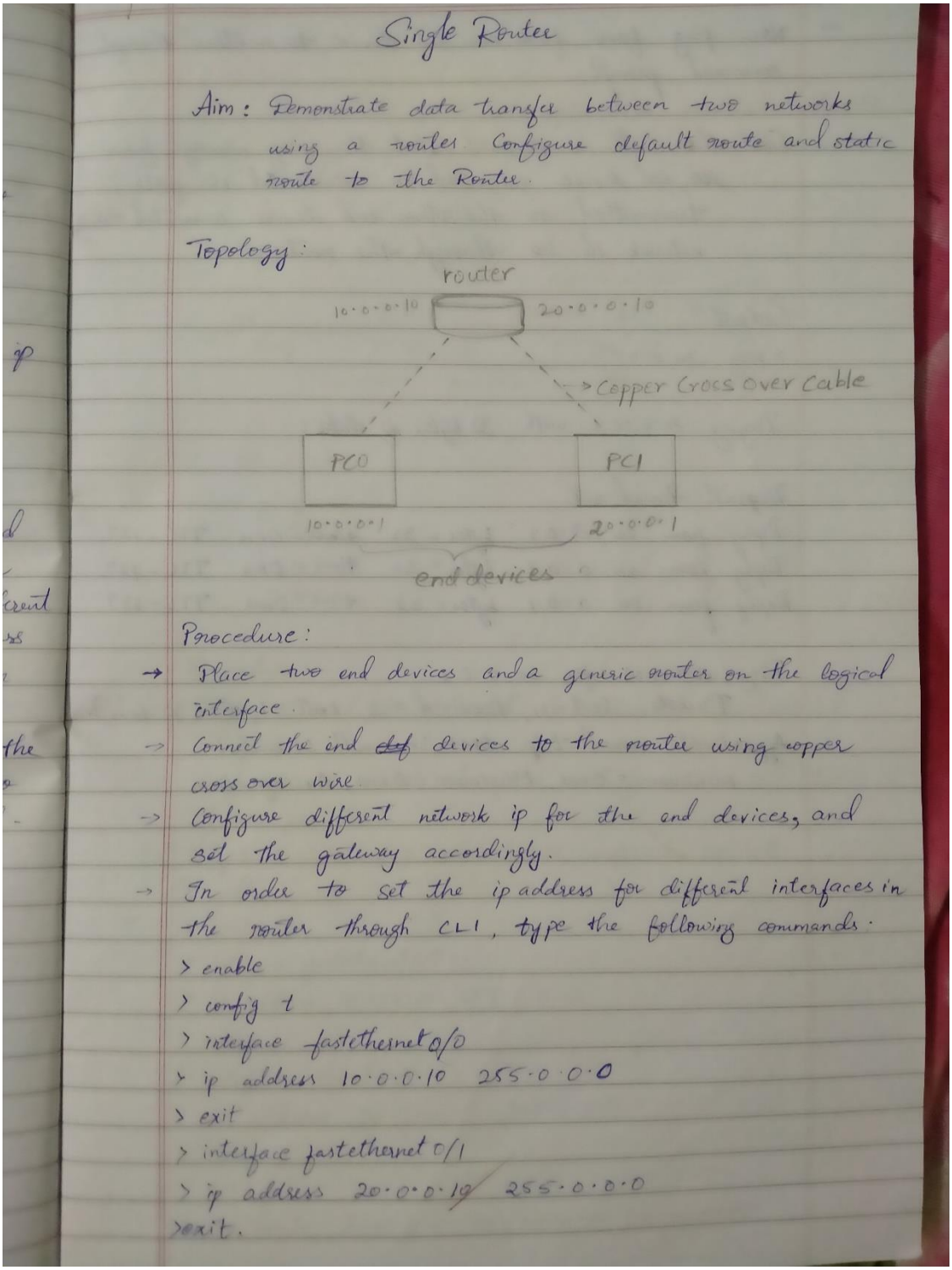

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.10
Pinging 10.0.0.10 with 32 bytes of data:
Reply from 10.0.0.10: bytes=32 time=0ms TTL=128
Reply from 10.0.0.10: bytes=32 time=5ms TTL=128
Reply from 10.0.0.10: bytes=32 time=0ms TTL=128
Reply from 10.0.0.10: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 5ms, Average = 1ms
PC>


Cisco Packet Tracer Student - C:\Users\STUDENT\Desktop\1BM21CS013\Computer networks\switch-hub simulation.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type Info
0.000 -- PC0 ICMP
0.001 PC0 Hub0 ICMP
0.002 Hub0 PC1 ICMP
0.002 Hub0 PC2 ICMP
0.002 Hub0 PC3 ICMP
Reset Simulation
Constant Delay
Capturing...
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
Edit Filters
Show All/None
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
Fa0
Time: 00:43:46.766
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Connections
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num
In Progress PC0 PC2 ICMP 0.000 N 0


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
PC>reset
Invalid Command.
PC>clear
Invalid Command.
PC>cls
Invalid Command.
PC>cli
Invalid Command.
PC>
PC>
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>

Experiment 2:

Single Router and Multi Router



Single Router

Aim: Demonstrate data transfer between two networks using a router Configure default route and static route to the Router.

Topology:

Procedure:

→ Place two end devices and a generic router on the logical interface.

→ Connect the end ~~dev~~ devices to the router using copper cross over wire

→ Configure different network ip for the end devices, and set the gateway accordingly.

→ In order to set the ip address for different interfaces in the router through CLI, type the following commands:

> enable
> config t
> interface fastethernet 0/0
> ip address 10.0.0.10 255.0.0.0
> exit
> interface fastethernet 0/1
> ip address 20.0.0.10 255.0.0.0
> exit.

Date
Page

→ Now ping from from one end device to another through command prompt.

Observation: It is observed that the ping message from ~~each~~ end device connected to network id 10 gets transmitted to the other end device connected to network id 20 through the routes.

Output:

> ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Request timed out

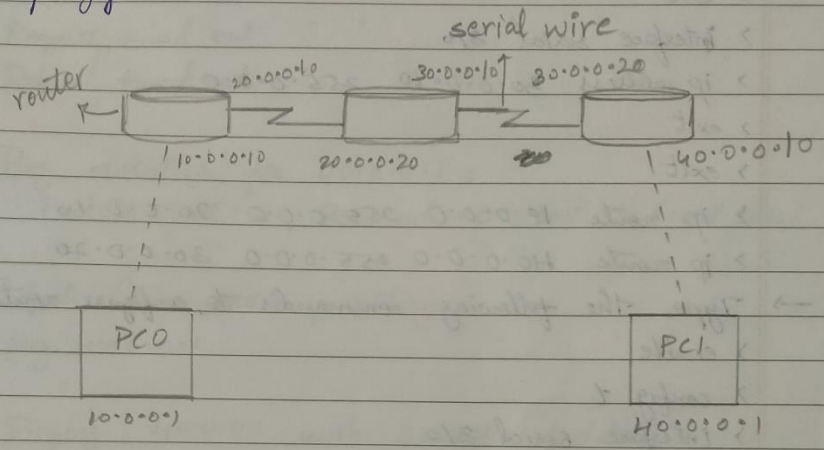
Reply from 20.0.0.1 bytes = 32 time = 0ms TTL = 127
Reply from 20.0.0.1 bytes = 32 time = 0ms TTL = 127
Reply from 20.0.0.1 bytes = 32 time = 0ms TTL = 127

Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1, (25% loss)
Approximate round trip times in milli seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

Multi Router

Aim: Demonstrate data transfer over multiple networks through routers. Configure default route and static route to the routers.

Topology:



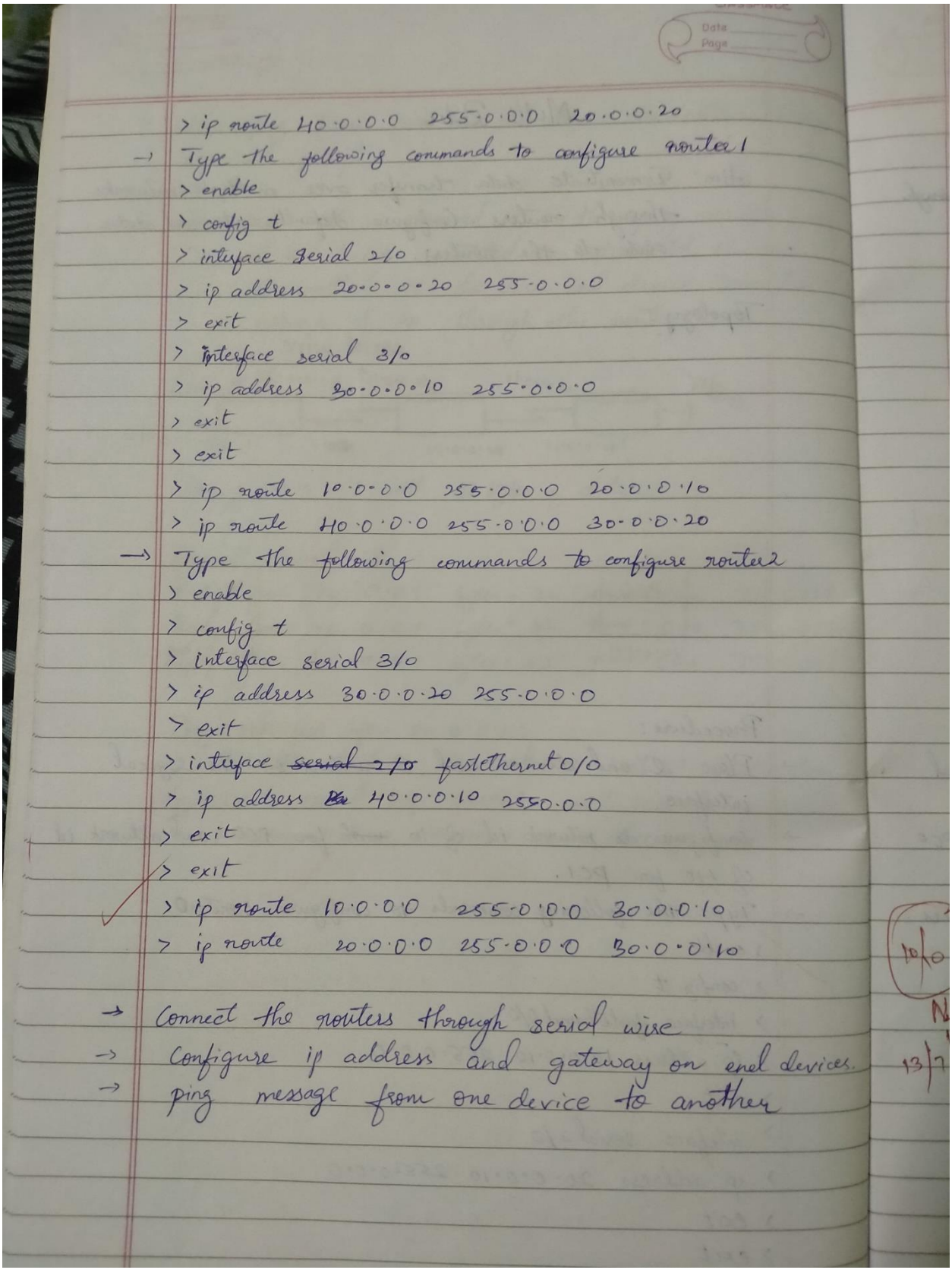
Procedure:

→ Place 2 end devices and 3 routers on the logical interface.

→ Configure a network id of 10 ~~with~~ for PC0 and network id of 40 for PC1.

→ Type the following commands to configure router0:

> enable
> config t
> interface fastethernet 0/0
> ip address 10.0.0.10 255.0.0.0
> exit
> inteface serial 2/0
> ip address 20.0.0.10 255.0.0.0
> exit
> exit
> ip route 30.0.0.0 255.0.0.0 20.0.0.20



> ip route 40.0.0.0 255.0.0.0 20.0.0.20

→ Type the following commands to configure router1

> enable
> config t
> interface Serial 2/0
> ip address 20.0.0.20 255.0.0.0
> exit
> interface serial 3/0
> ip address 30.0.0.10 255.0.0.0
> exit
> exit
> ip route 10.0.0.0 255.0.0.0 20.0.0.10
> ip route 40.0.0.0 255.0.0.0 30.0.0.20

→ Type the following commands to configure router2

> enable
> config t
> interface serial 3/0
> ip address 30.0.0.20 255.0.0.0
> exit
> interface ~~serial 2/0~~ fastethernet 0/0
> ip address ~~Da~~ 40.0.0.10 2550.0.0
> exit
> exit
> ip route 10.0.0.0 255.0.0.0 30.0.0.10
> ip route 20.0.0.0 255.0.0.0 30.0.0.10

→ Connect the routers through serial wire
→ Configure ip address and gateway on end devices.
→ ping message from one device to another

Output: (Before static ip routing)

> ping 40.0.0.1

Pinginging ~~with~~ 40.0.0.1 with 32 bytes of data :

Request timed out
Request timed out
Request timed out
Request timed out

Ping statistics for 40.0.0.1 :
Packets : Sent = 4, Received = 0, Lost = 4 (100% loss)

Output : (After static ip routing)

> ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data :

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

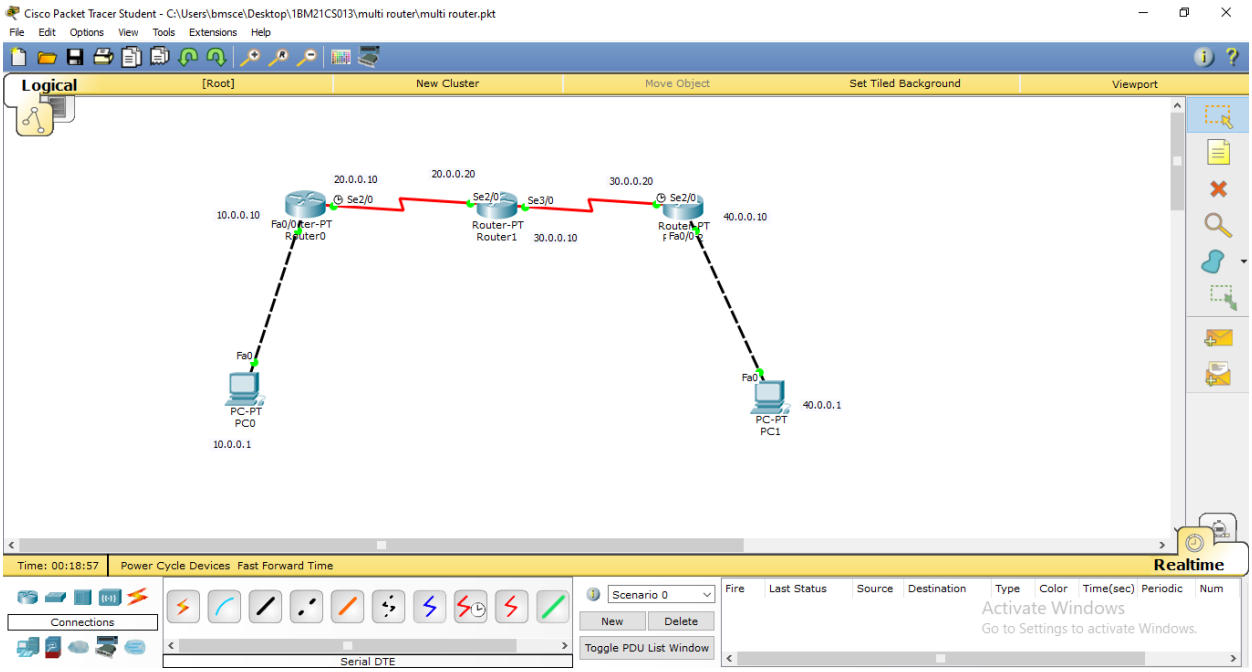
Ping statistics for 40.0.0.1 :
Packets : Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli seconds :
Minimum = 2ms, Maximum = 9ms, Average = 3ms.

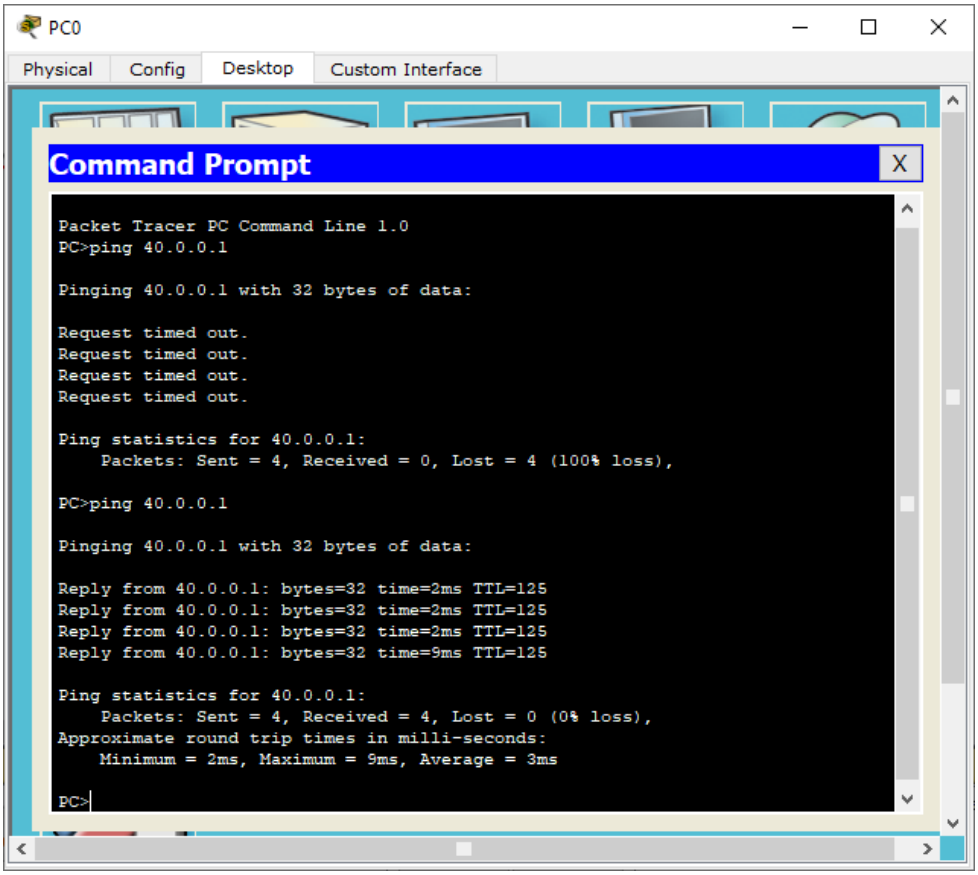
10/10

13/7/22

Observation: The data packets gets transmitted across various networks through the routers with the help of static routing and direct routing.

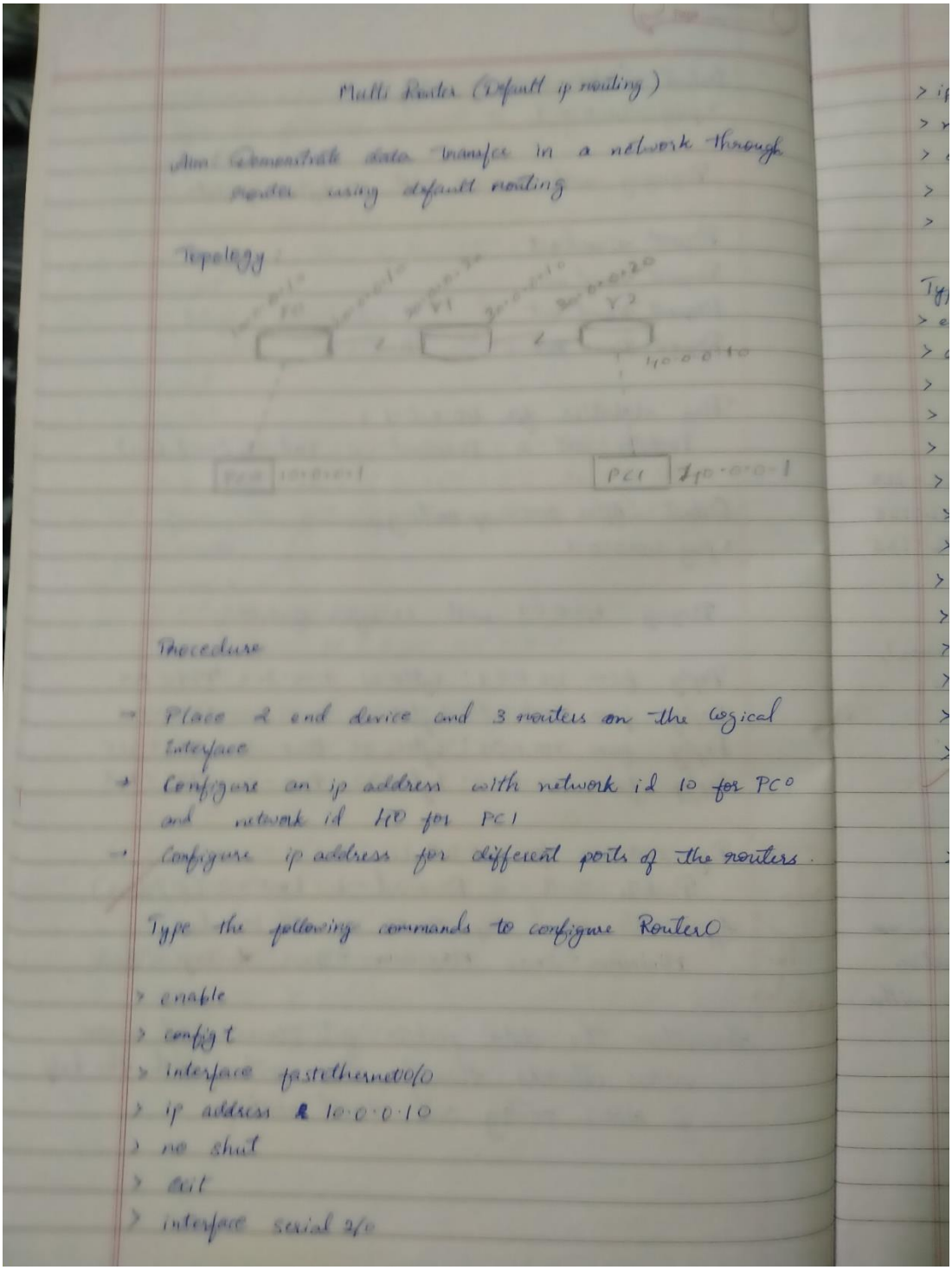
Topology and output screenshots:





Experiment 3:

Default Routing



Multi Router (Default ip routing)

Aim: To demonstrate data transfer in a network through router using default routing

Topology

PC0 10.0.0.1

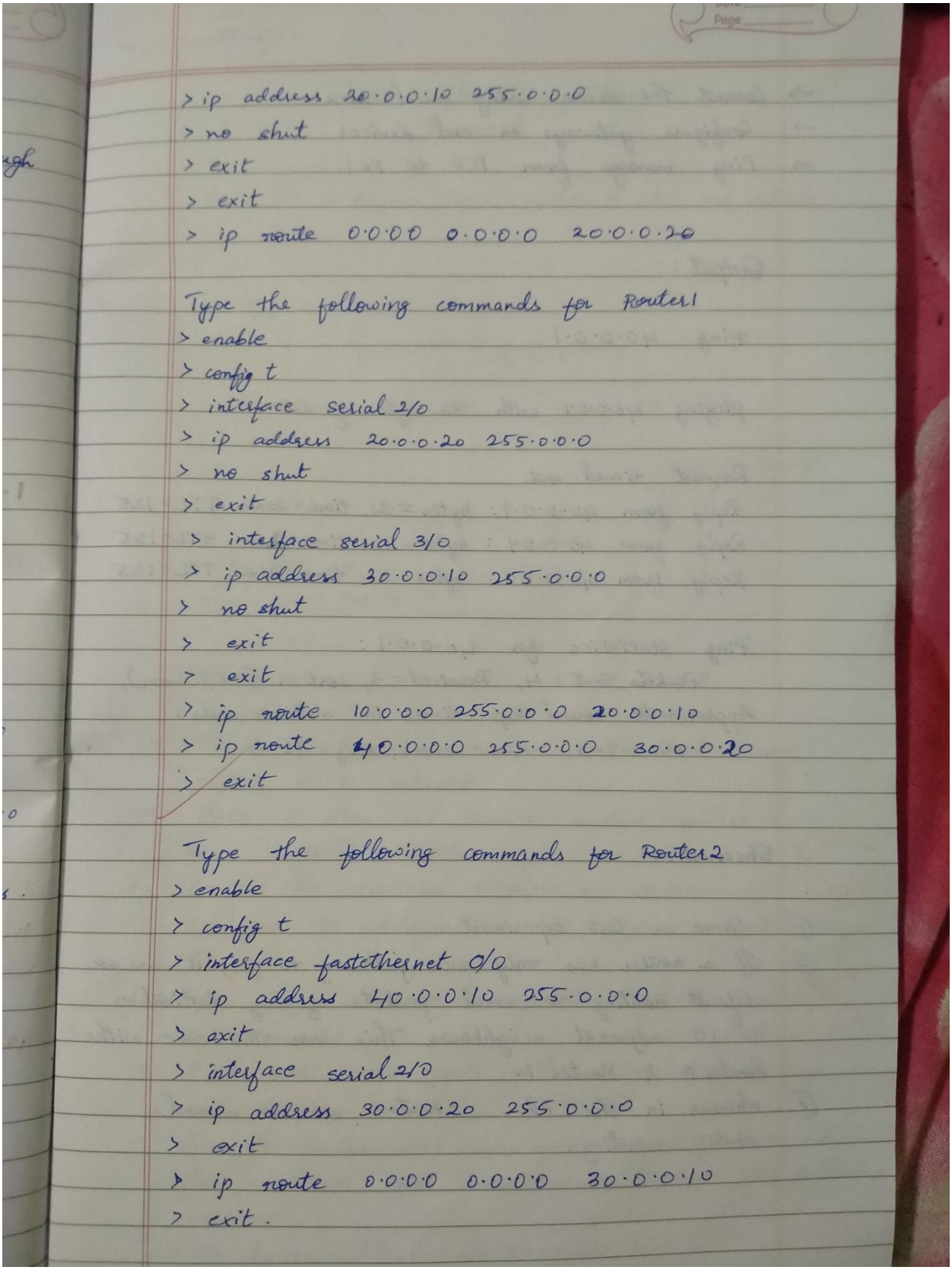
PC1 10.0.0.1

Procedure

- Place 2 end device and 3 routers on the logical Interface
- Configure an ip address with network id 10 for PC0 and network id 10 for PC1
- Configure ip address for different ports of the routers.

Type the following commands to configure Router0

> enable
> config t
> interface fastethernet0/0
> ip address 10.0.0.10
> no shut
> exit
> interface serial 2/0



> ip address 20.0.0.10 255.0.0.0
> no shut
> exit
> exit
> ip route 0.0.0.0 0.0.0.0 20.0.0.20

Type the following commands for Router1
> enable
> config t
> interface serial 2/0
> ip address 20.0.0.20 255.0.0.0
> no shut
> exit
> interface serial 3/0
> ip address 30.0.0.10 255.0.0.0
> no shut
> exit
> exit
> ip route 10.0.0.0 255.0.0.0 20.0.0.10
> ip route 40.0.0.0 255.0.0.0 30.0.0.20
> exit

Type the following commands for Router2
> enable
> config t
> interface fastethernet 0/0
> ip address 40.0.0.10 255.0.0.0
> exit
> interface serial 2/0
> ip address 30.0.0.20 255.0.0.0
> exit
> ip route 0.0.0.0 0.0.0.0 30.0.0.10
> exit.

→ Connect the devices using serial wire
→ Configure gateways on end devices
→ Ping message from PC0 to PC1

Output:

Ping 40.0.0.1

pinging 40.0.0.1 with 32 bytes of data

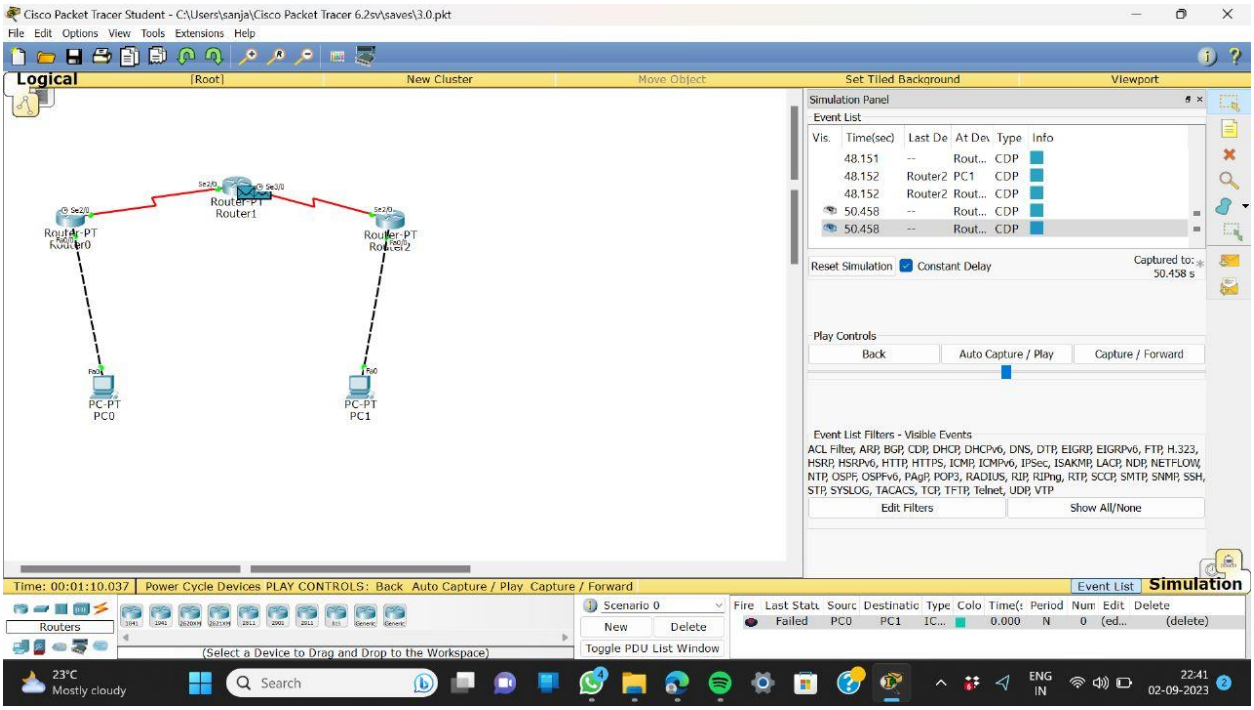
Request timed out
Reply from 40.0.0.1: bytes = 32 time = 20ms TTL = 125
Reply from 40.0.0.1: bytes = 32 time = 9ms TTL = 125
Reply from 40.0.0.1: bytes = 32 time = 21ms TTL = 125

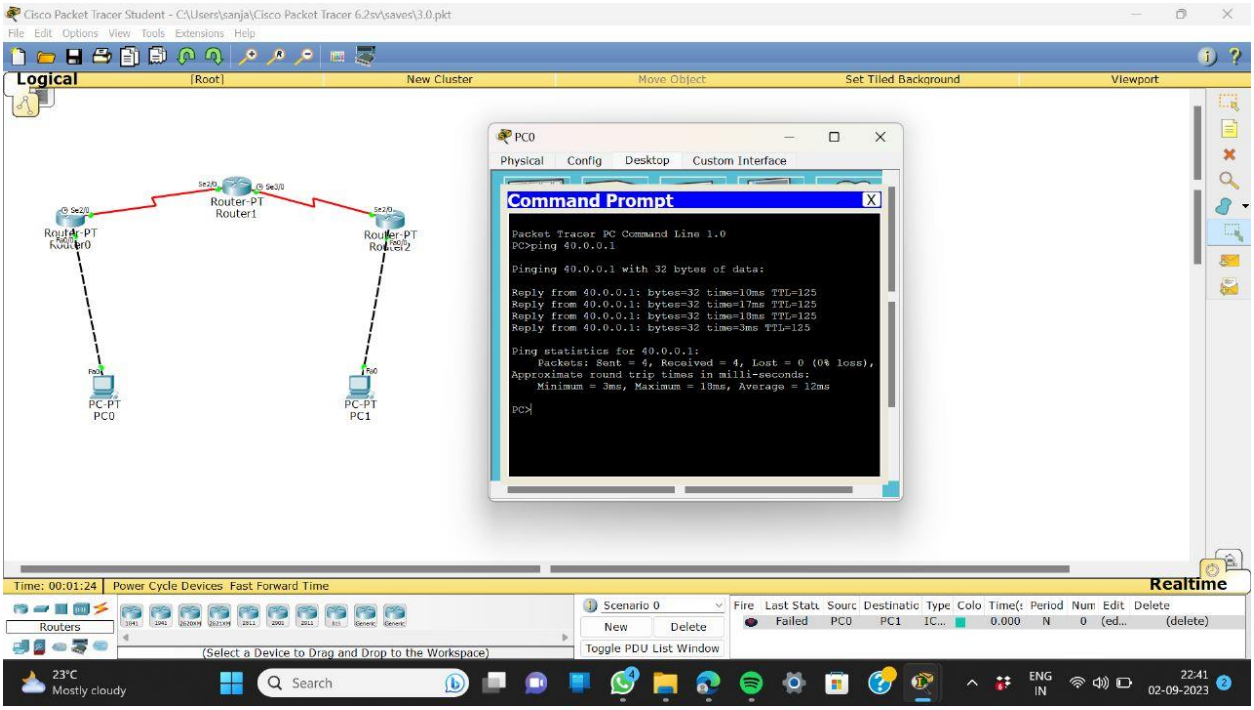
Ping statistics for 40.0.0.1:
Packets Sent = 4, Received = 3, Lost = 1 (25% Loss),
Approximate round trip times in milli seconds:
Min = 9ms, Max = 21 ms, Avg = 16 ms

Observation:

① Same as last experiment
② If a router has only one pathway to go, it can use default routing to send packets of any destination to its adjacent neighbours. This was the case with Router 0 & Router 1.
③ whereas in the other 2 routers, we do usual static routing.

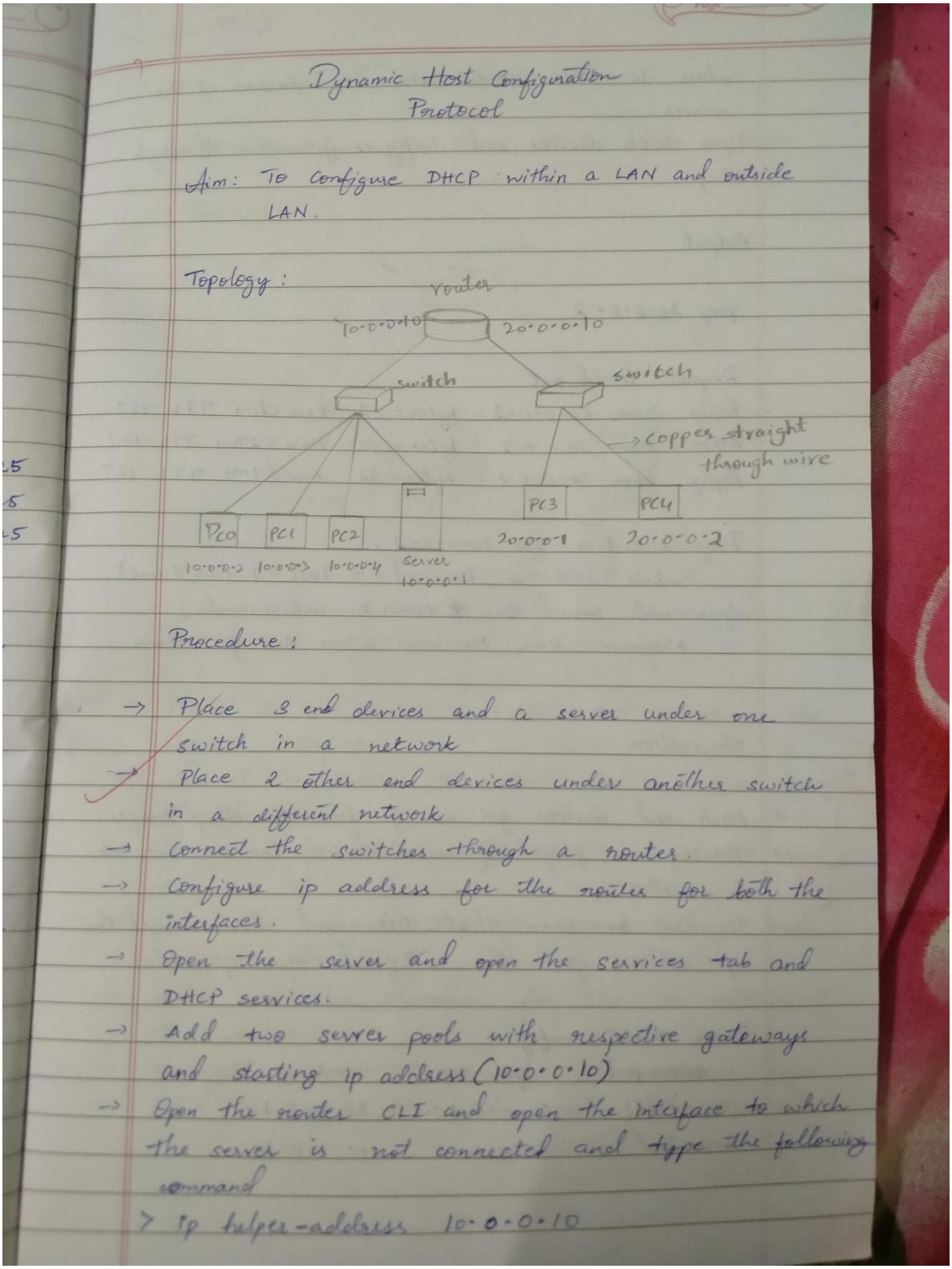
Topology and output screenshots:





Experiment 4:

Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol

Aim: To configure DHCP within a LAN and outside LAN.

Topology:

Procedure:

→ Place 3 end devices and a server under one switch in a network
→ Place 2 other end devices under another switch in a different network
→ Connect the switches through a router.
→ Configure ip address for the router for both the interfaces.
→ Open the server and open the services tab and DHCP services.
→ Add two server pools with respective gateways and starting ip address (10.0.0.10)
→ Open the router CLI and open the interface to which the server is not connected and type the following command

> ip helper-address 10.0.0.10

where 10.0.0.10 is the static ip address of the server.

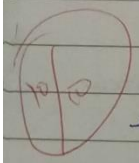
→ Open each devices and configure ip address through dhcp.

Output

ping 20.0.0.2

Request timed out
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127
Reply from 20.0.0.2: bytes=32 time<27ms TTL=127
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milliseconds:
Minimum = 0ms, Maximum = 27ms, Average = 9ms.



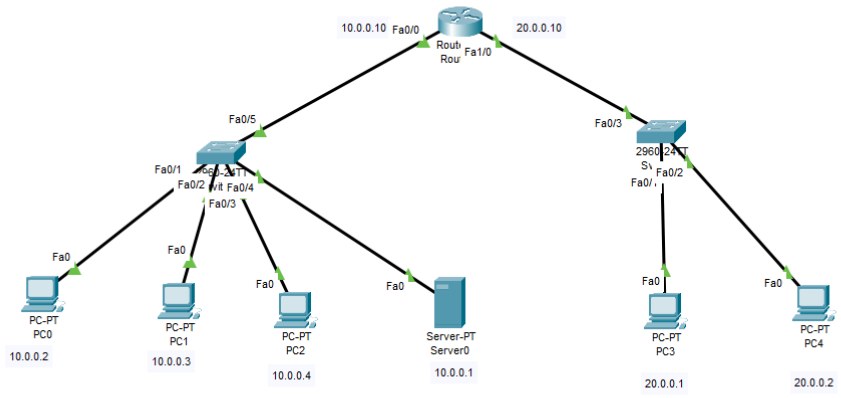
Observation

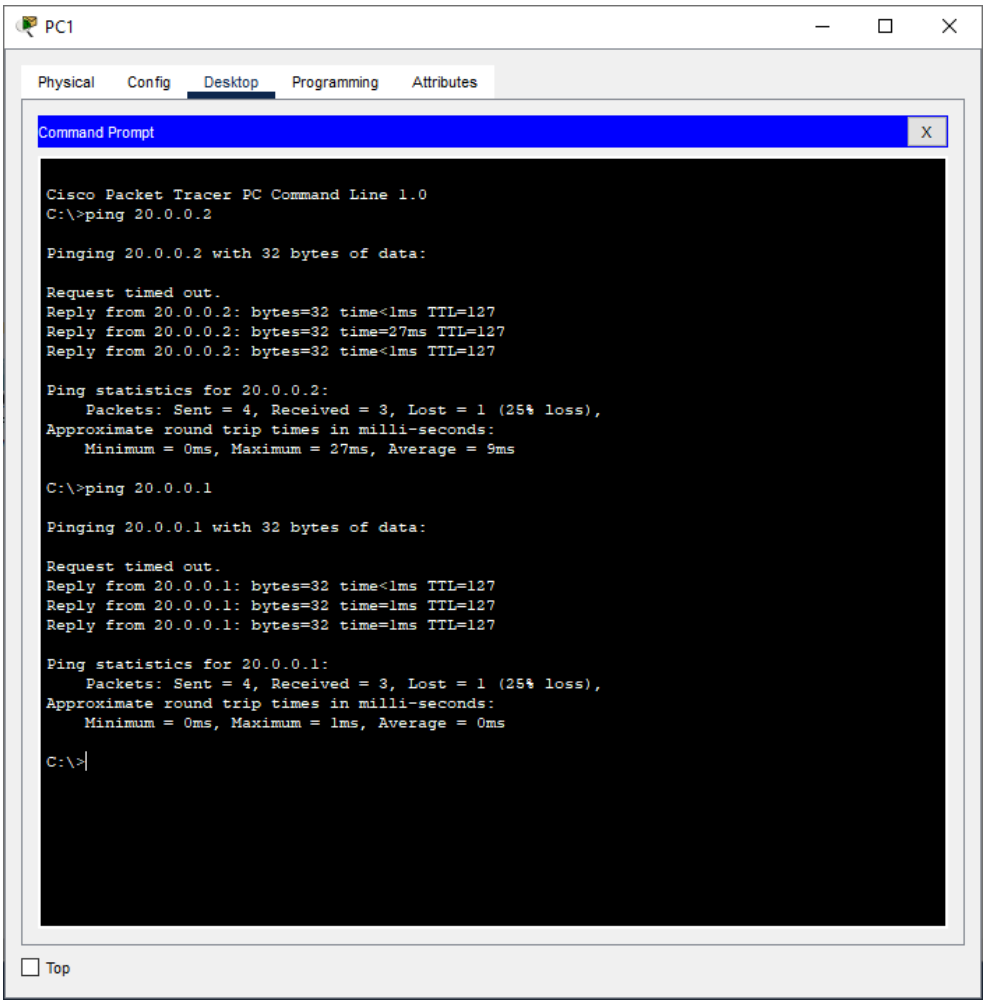
→ Each end devices get an ip from the dhcp server
→ Message is pinged from a device in one network to another.
→ It can be seen that the end devices get the ip address from the configured start ip address.

Server DHCP config:

pool	gateway	start ip
serverPool1	20.0.0.10	20.0.0.1
serverPool	10.0.0.10	10.0.0.2

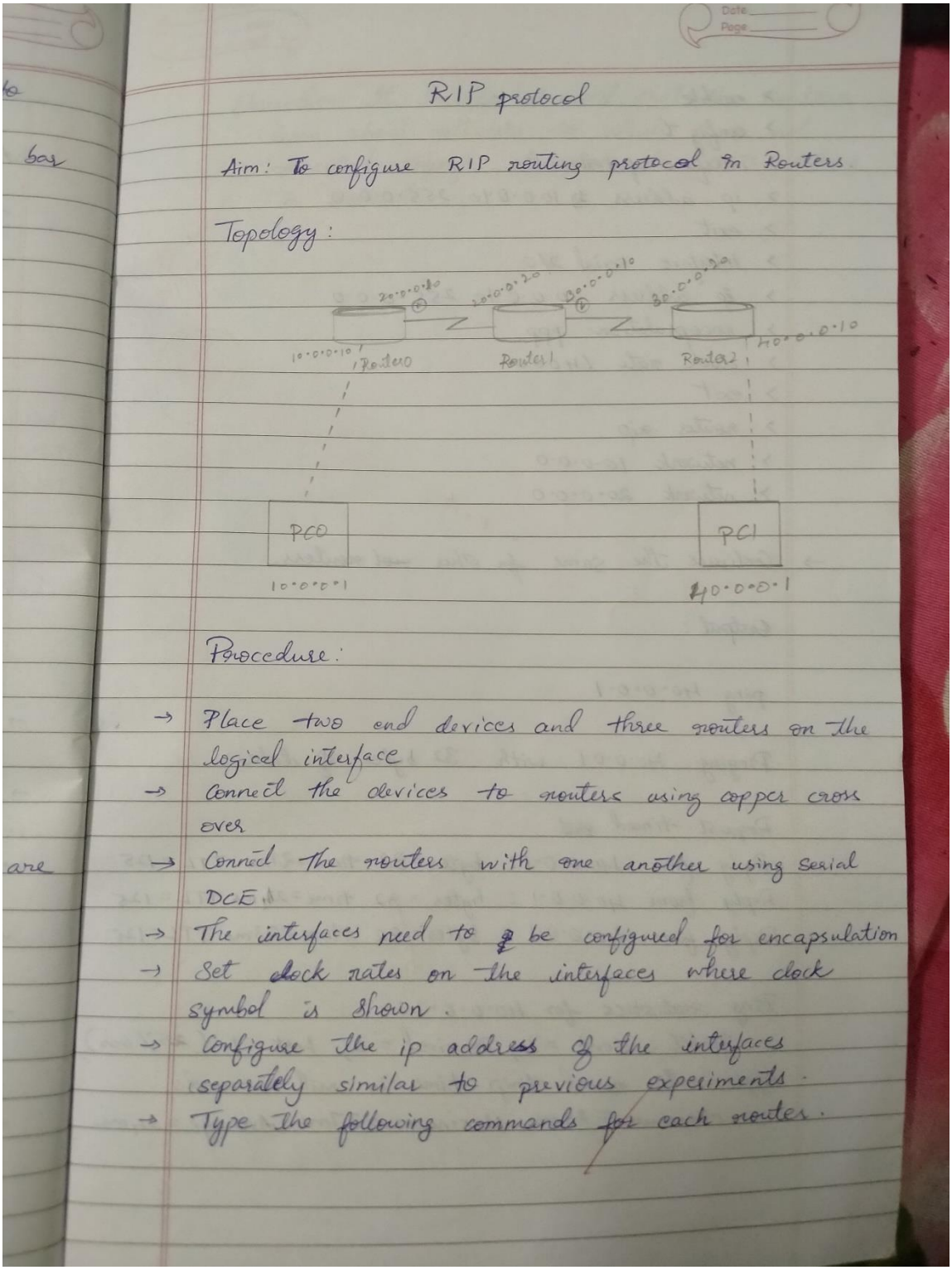
Topology and output screenshots:





Experiment 5:

RIP routing protocol



RIP protocol

Aim: To configure RIP routing protocol in Routers

Topology:

Procedure:

→ Place two end devices and three routers on the logical interface
→ Connect the devices to routers using copper cross over
→ Connect the routers with one another using serial DCE.
→ The interfaces need to be configured for encapsulation
→ Set clock rates on the interfaces where clock symbol is shown.
→ Configure the ip address of the interfaces separately similar to previous experiments.
→ Type the following commands for each routes.

> enable
> config t
> interface fastethernet
> ip address 10.0.0.10 255.0.0.0
> exit
> interface serial 2/0
> ip address 20.0.0.10 255.0.0.0
> encapsulation PPP
> clock rate 64000
> exit
> router rip
> network 10.0.0.0
> network 20.0.0.0

exit

→ Continue the same for other ~~not~~ routers.

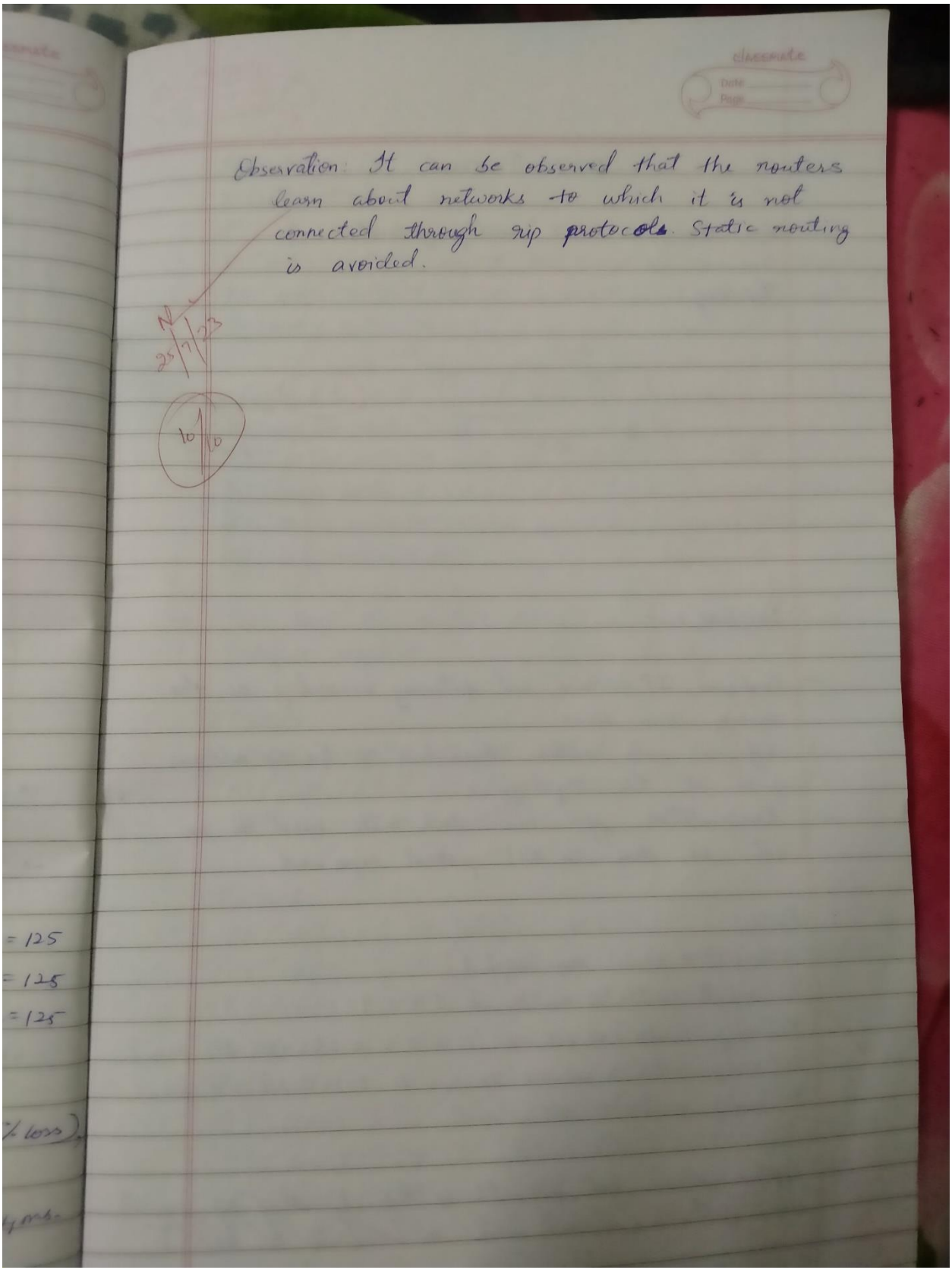
Output:

ping 40.0.0.1

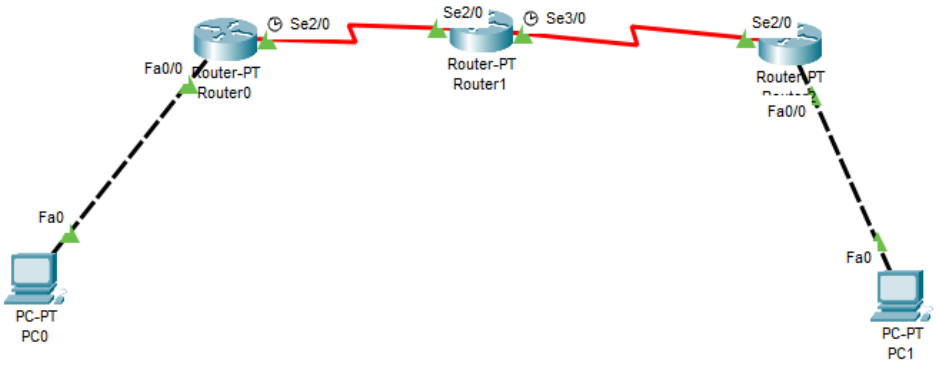
Pinging 40.0.0.1 with 32 bytes of data:

Request timed out
Reply from 40.0.0.1: bytes=32 time=27ms TTL=125
Reply from 40.0.0.1: bytes=32 time=24ms TTL=125
Reply from 40.0.0.1: bytes=32 time=23ms TTL=125

Ping statistics for 40.0.0.1:
Packet: Sent = 4, Received = 3, Lost = 1 (25% loss)
Approximate round trip time in milli-seconds:
Minimum = 23ms, Maximum = 27ms, Average = 24ms.


Observation: It can be observed that the routers learn about networks to which it is not connected through rip protocols. Static routing is avoided.
25/7/23
10/10
= 125
= 125
= 125

Topology and output screenshots:



PC0

Physical Config Desktop Programming Attributes

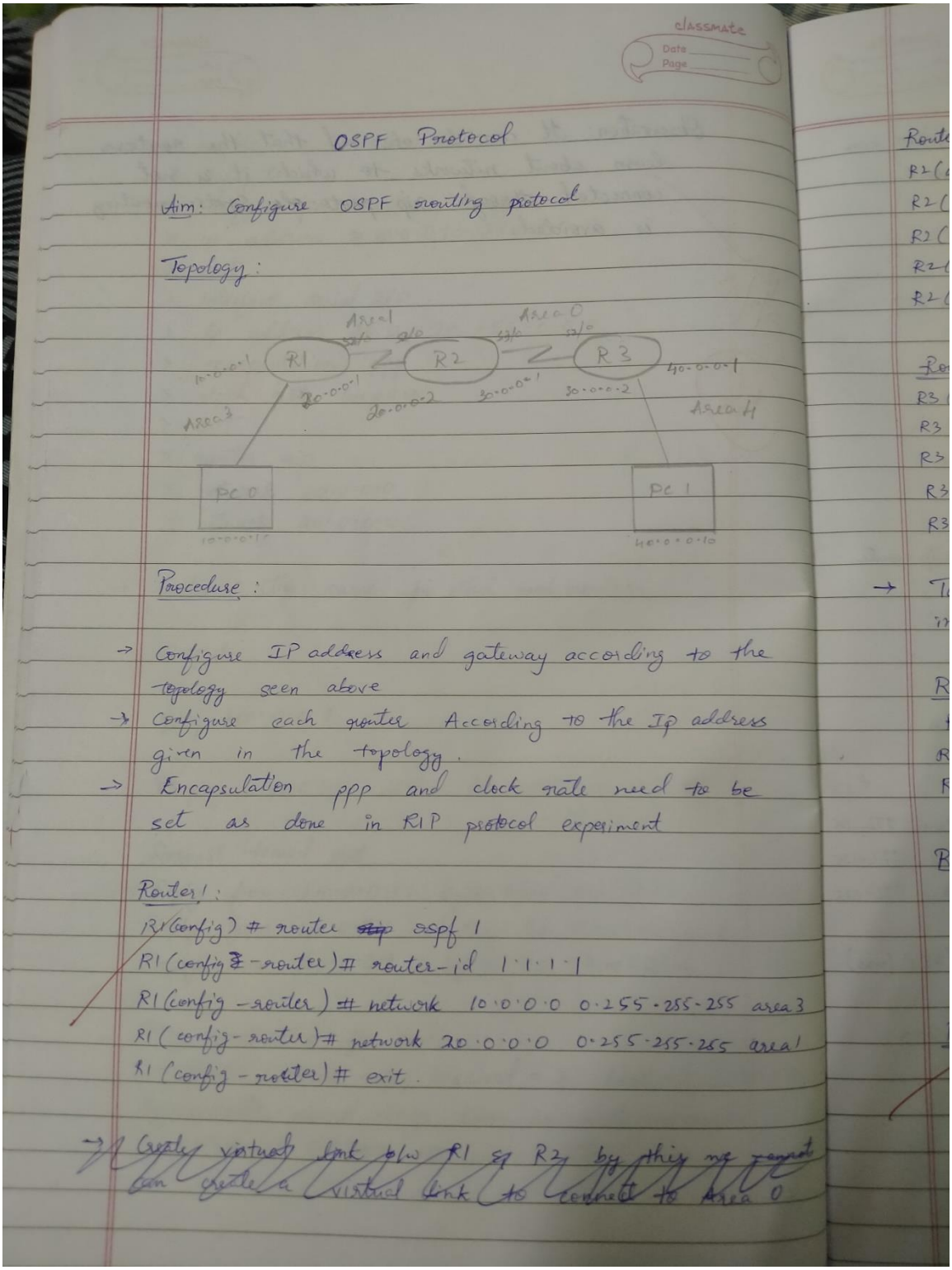
Command Prompt X

```
Reply from 30.0.0.10: bytes=32 time=24ms TTL=254
Reply from 30.0.0.10: bytes=32 time=20ms TTL=254
Reply from 30.0.0.10: bytes=32 time=14ms TTL=254
Reply from 30.0.0.10: bytes=32 time=19ms TTL=254

Ping statistics for 30.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 24ms, Average = 19ms

C:\>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=24ms TTL=253
Reply from 40.0.0.10: bytes=32 time=29ms TTL=253
Reply from 40.0.0.10: bytes=32 time=25ms TTL=253
Reply from 40.0.0.10: bytes=32 time=2ms TTL=253

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 29ms, Average = 20ms

C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.1: bytes=32 time=29ms TTL=125
Reply from 40.0.0.1: bytes=32 time=27ms TTL=125
Reply from 40.0.0.1: bytes=32 time=20ms TTL=125

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 20ms, Maximum = 29ms, Average = 25ms

C:\>
```

Top

Experiment 6:

OSPF Routing protocol



OSPF Protocol

Aim: Configure OSPF routing protocol

Topology:

Procedure:

→ Configure IP address and gateway according to the topology seen above
→ Configure each router According to the Ip address given in the topology.
→ Encapsulation ppp and clock rate need to be set as done in RIP protocol experiment

Router1:

R1(config)# router ospf 1
R1(config-router)# router-id 1.1.1.1
R1(config-router)# network 10.0.0.0 0.255.255.255 area 3
R1(config-router)# network 20.0.0.0 0.255.255.255 area 1
R1(config-router)# exit.

→ Create virtual link b/w R1 & R2 by this we can create a virtual link to connect to Area 0

Router 2:

R2(config)# router ospf 1
R2(config-router)# router-id 2.2.2.2
R2(config-router)# network 20.0.0.0 0.255.255.255 area 1
R2(config-router)# network 30.0.0.0 0.255.255.255 area 0
R2(config-router)# exit.

Router 3:

R3(config)# router ospf 1
R3(config-router)# router-id 3.3.3.3
R3(config-router)# network 30.0.0.0 0.255.255.255 area 0
R3(config-router)# network 40.0.0.0 0.255.255.255 area 2
R3(config-router)# exit

→ To keep the routers Active we have to configure interface loopback

Router 1:

R1(config-if)# interface loopback 0
R1(config-if)# ip address 172.16.1.252 255.255.255.0
R1(config-if)# no shutdown

Router 2:

R2(config-if)# interface loopback 0
R2(config-if)# ip address 172.16.1.253 255.255.255.0
R2(config-if)# no shutdown.

Router 3:

R3(config-if)# interface loopback 0
R3(config-if)# ip address 172.16.1.254 255.255.255.0
R3(config-if)# no shutdown

→ Create a virtual link b/w R1, R2 by this we can create a virtual link to connect to area 0.

Router R1:

R1 (config) # router ospf 1
R1 (config-router) # area 1 virtual-link 2.2.2.2

Router R2:

~~R2 (conf)~~
R2 (config) # router ospf 1
R2 (config-router) # area 1 virtual-link 1.1.1.1
R2 (config-router) # exit

Finally, After creating virtual ~~##~~ link, show ip route for all routers.

Result:

PC> ping 40.0.0.10

pinging 40.0.0.10 with 32 bytes of data

Request timed out
~~R~~ Reply from 40.0.0.10: bytes = 32 time = 10ms TTL=125
Reply from 40.0.0.10: bytes = 32 time = 2ms TTL=125
Reply from 40.0.0.10: bytes = 32 time = 9ms TTL=125

ping statistics for 40.0.0.10:
~~Pa~~ Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round-trip ~~t~~ in milli seconds
Minimum = 2ms, Maximum = 10ms, Average = 7ms.

→ Create a virtual link b/w R1, R2 by this we can create a virtual link to connect to area 0.

Router R1:

R1(config)# router ospf 1
R1(config-router)# area 1 virtual-link 2.2.2.2

Router R2:

~~R2(conf)~~
R2(config)# router ospf 1
R2(config-router)# area 1 virtual-link 1.1.1.1
R2(config-router)# exit

Finally, After creating virtual ~~##~~ link, show ip route for all routers.

Result:

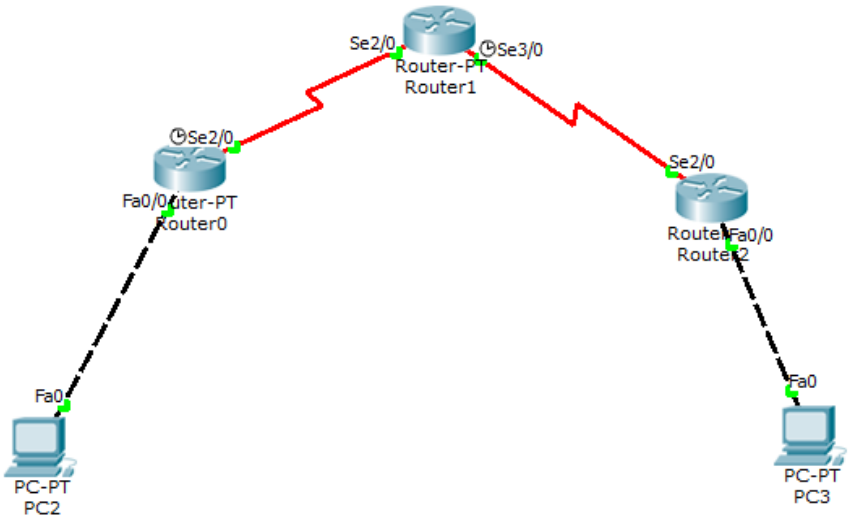
PC> ping 40.0.0.10

pinging 40.0.0.10 with 32 bytes of data

Request timed out
~~Re~~ Reply from 40.0.0.10: bytes = 32 time = 10ms TTL=125
Reply from 40.0.0.10: bytes = 32 time = 2ms TTL=125
Reply from 40.0.0.10: bytes = 32 time = 9ms TTL=125

ping statistics for 40.0.0.10:
~~Pa~~ Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round ~~trip~~ in milli seconds
Minimum = 2ms, Maximum = 10ms, Average = 7ms.

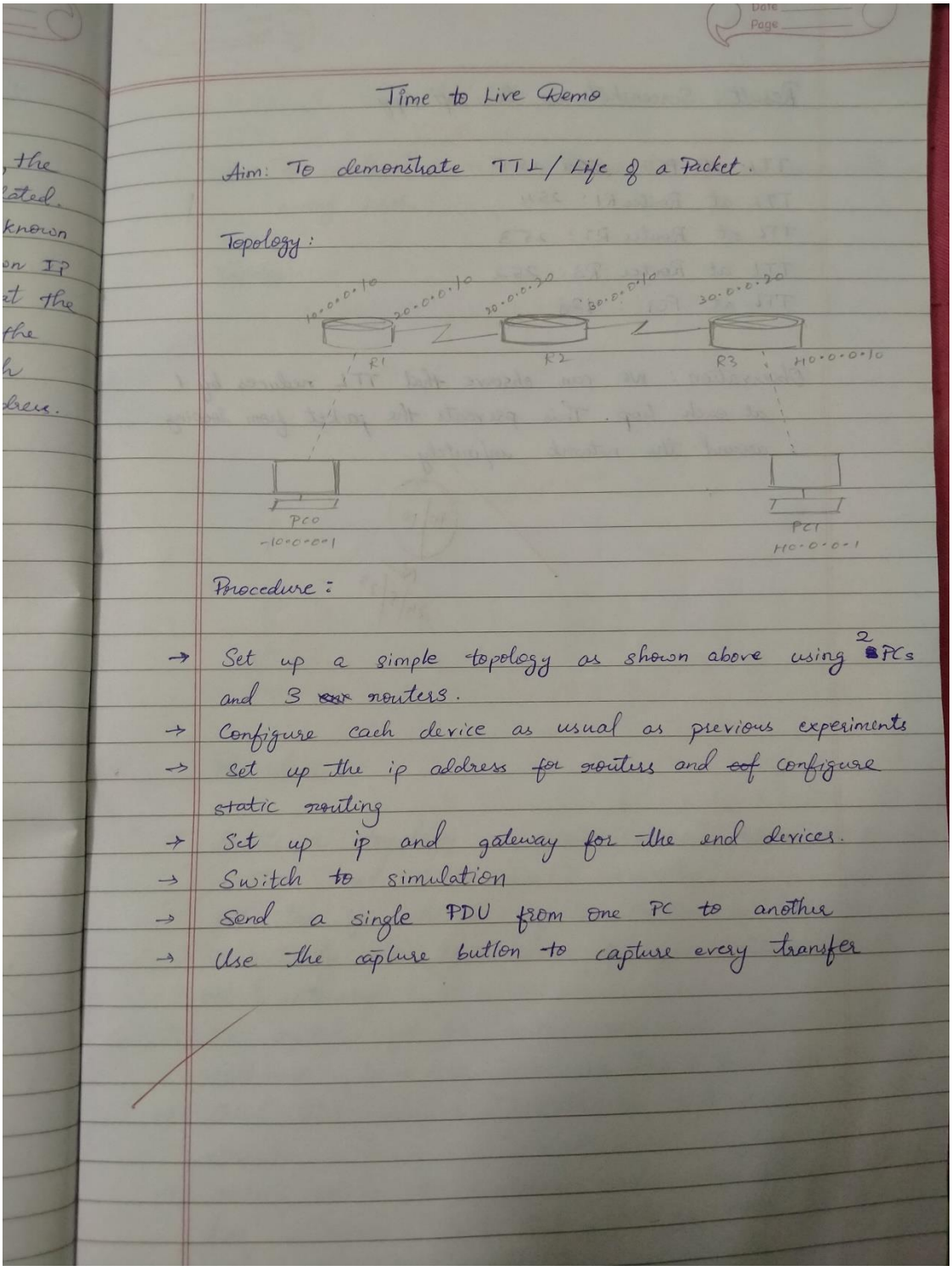
Topology and output screenshots:




```
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=9ms TTL=125
Reply from 40.0.0.10: bytes=32 time=2ms TTL=125
Reply from 40.0.0.10: bytes=32 time=2ms TTL=125
Reply from 40.0.0.10: bytes=32 time=2ms 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 = 2ms, Maximum = 9ms, Average = 3ms
```

Experiment 7:

Demonstrate TTL of a packet



Time to Live Demo

Aim: To demonstrate TTL / Life of a Packet.

Topology:

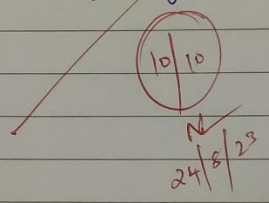
Procedure:

→ Set up a simple topology as shown above using 2 PCs and 3 routers.
→ Configure each device as usual as previous experiments
→ Set up the ip address for routers and configure static routing
→ Set up ip and gateway for the end devices.
→ Switch to simulation
→ Send a single PDU from one PC to another
→ Use the capture button to capture every transfer

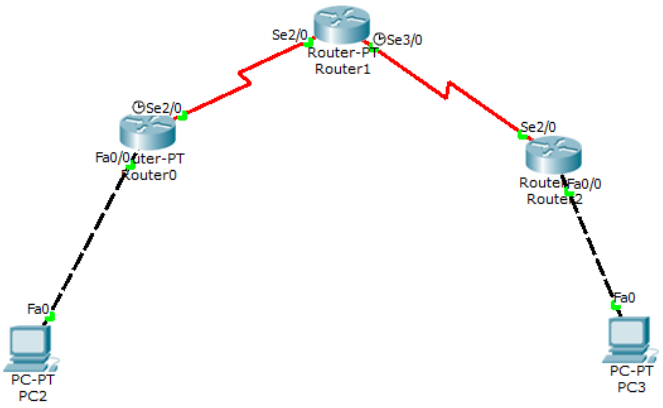
Result: Screenshots in the soft copy.

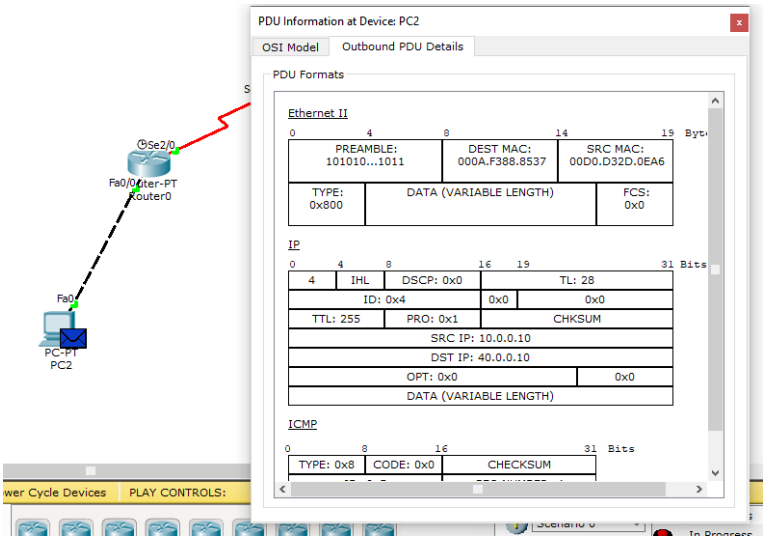
TTL at PC0: 255
TTL at Router R1: 254
TTL at Router R2: 253
TTL at Router R3: 252
TTL at PC1 : 128

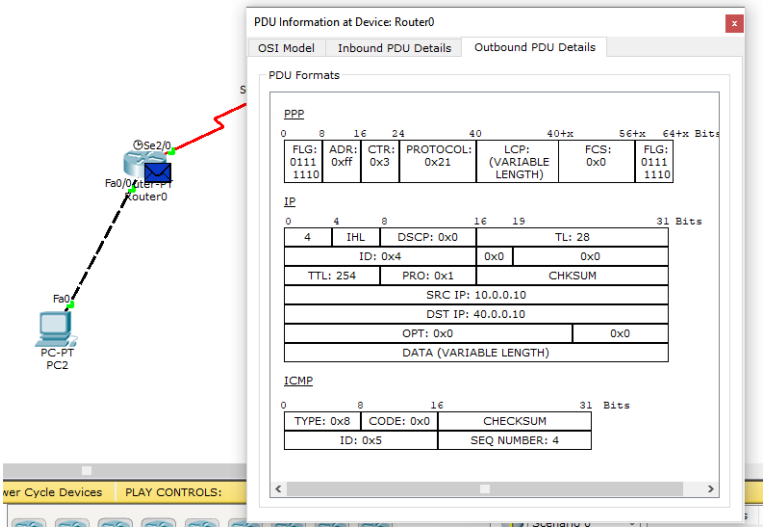
Observation: We can observe that TTL reduces by 1 at each hop. This prevents the packet from looping around the network infinitely.

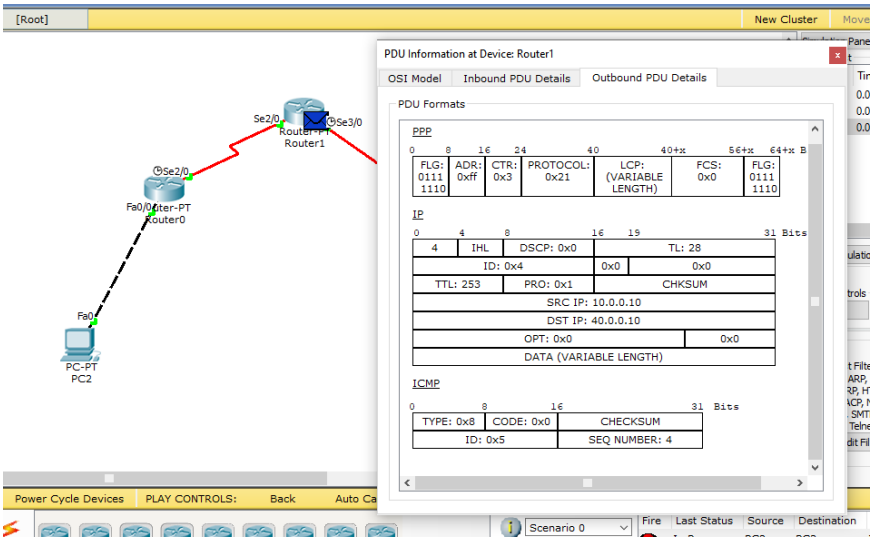


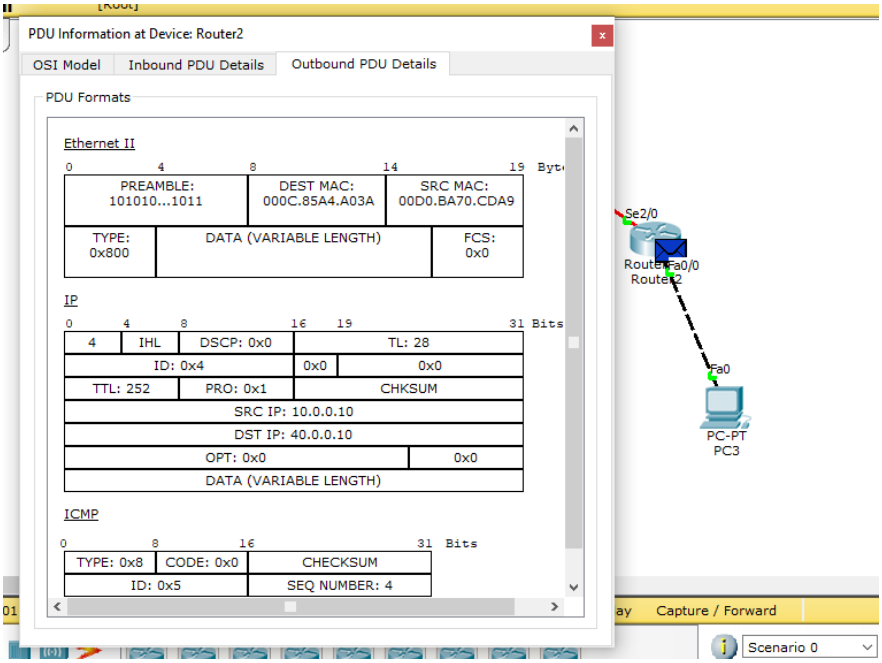
Topology and output screenshots:





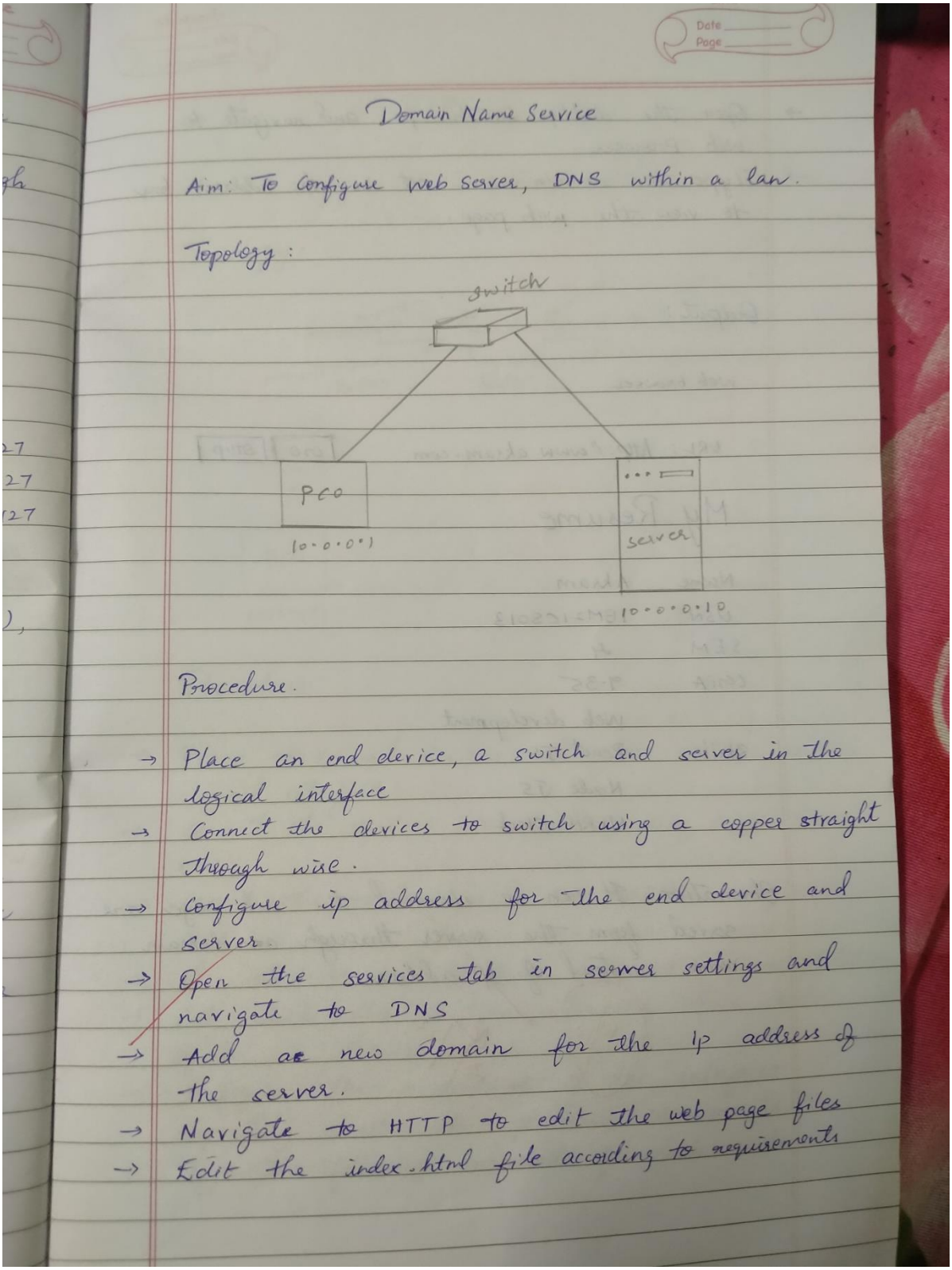



[Root]
New Cluster
PDU Information at Device: Router1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
PPP
FLG: 0111 1110
ADR: 0xff
CTR: 0x3
PROTOCOL: 0x21
LCP: (VARIABLE LENGTH)
FCS: 0x0
FLG: 0111 1110
IP
4
IHL
DSCP: 0x0
TL: 28
ID: 0x4
0x0
0x0
TTL: 253
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.10
DST IP: 40.0.0.10
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 4
Se2/0
Se3/0
Router1
Se2/0
Fa0/0
Router0
Fa0
PC-PT
PC2
Power Cycle Devices
PLAY CONTROLS:
Back
Scenario 0
Fire
Last Status
Source
Destination


PDU Information at Device: Router2
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 000C.85A4.A03A
SRC MAC: 00D0.BA70.CDA9
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
4
IHL
DSCP: 0x0
TL: 28
ID: 0x4
0x0
0x0
TTL: 252
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.10
DST IP: 40.0.0.10
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 4
Se2/0
Router2
Fa0/0
Fa0
PC-PT
PC3
Capture / Forward
Scenario 0

Experiment 8:

Configure Web Server, DNS within a LAN



Domain Name Service

Aim: To Configure web server, DNS within a lan.

Topology:

Procedure.

→ Place an end device, a switch and server in the logical interface
→ Connect the devices to switch using a copper straight through wire.
→ Configure ip address for the end device and server.
→ Open the services tab in server settings and navigate to DNS
→ Add a new domain for the ip address of the server.
→ Navigate to HTTP to edit the web page files
→ Edit the index.html file according to requirements

→ Open the desktop in the pc and navigate to Web Browser

→ Type the domain name set in the address bar to view the web page.

Output:

Web browser

URL: http://www.akram.com [GO] [STOP]

My Resume

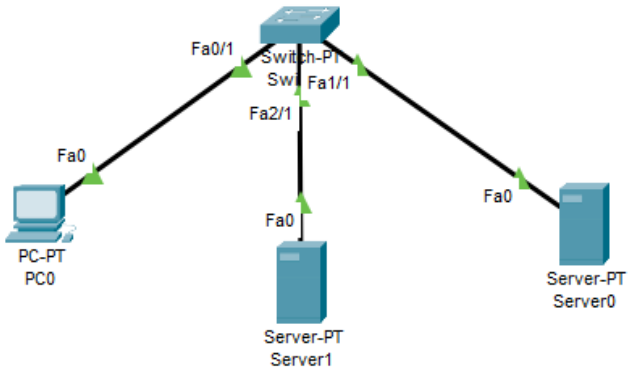
Name	Akram
USN	1BM21CS013
SEM	4
CGPA	9.35
Skills	Web development
	React JS
	Node JS
	MERN Stack

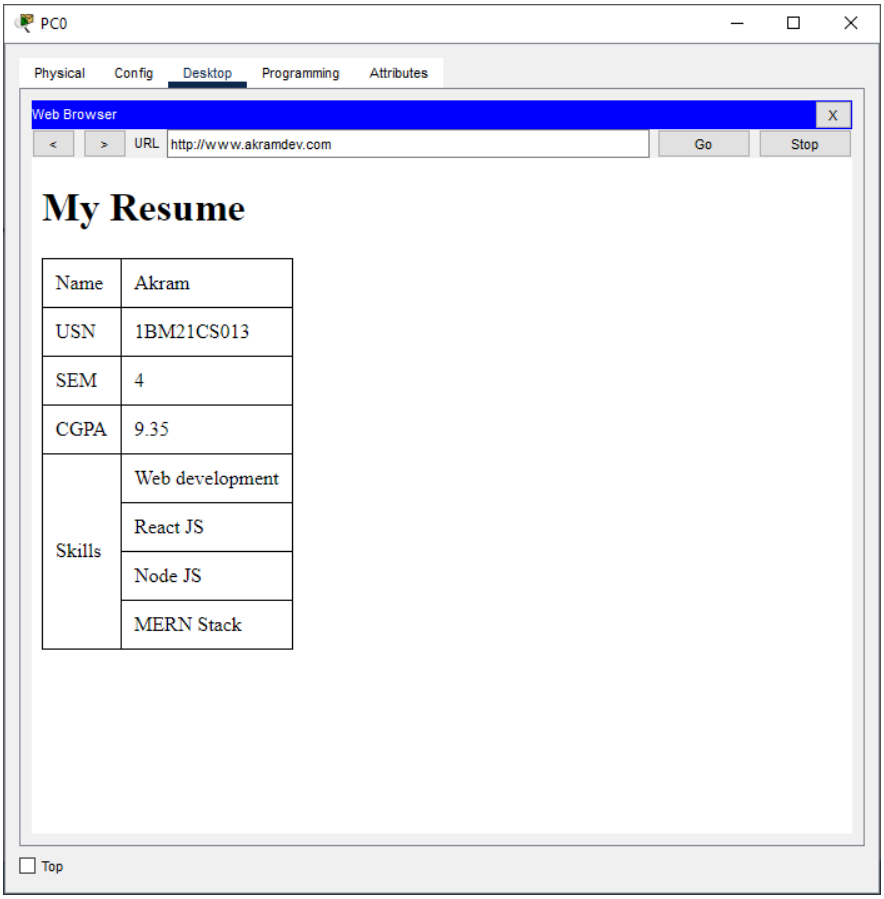
Observation: It can be observed the web pages are saved from the server through a domain name instead of ip address.

10/10

25/7/23

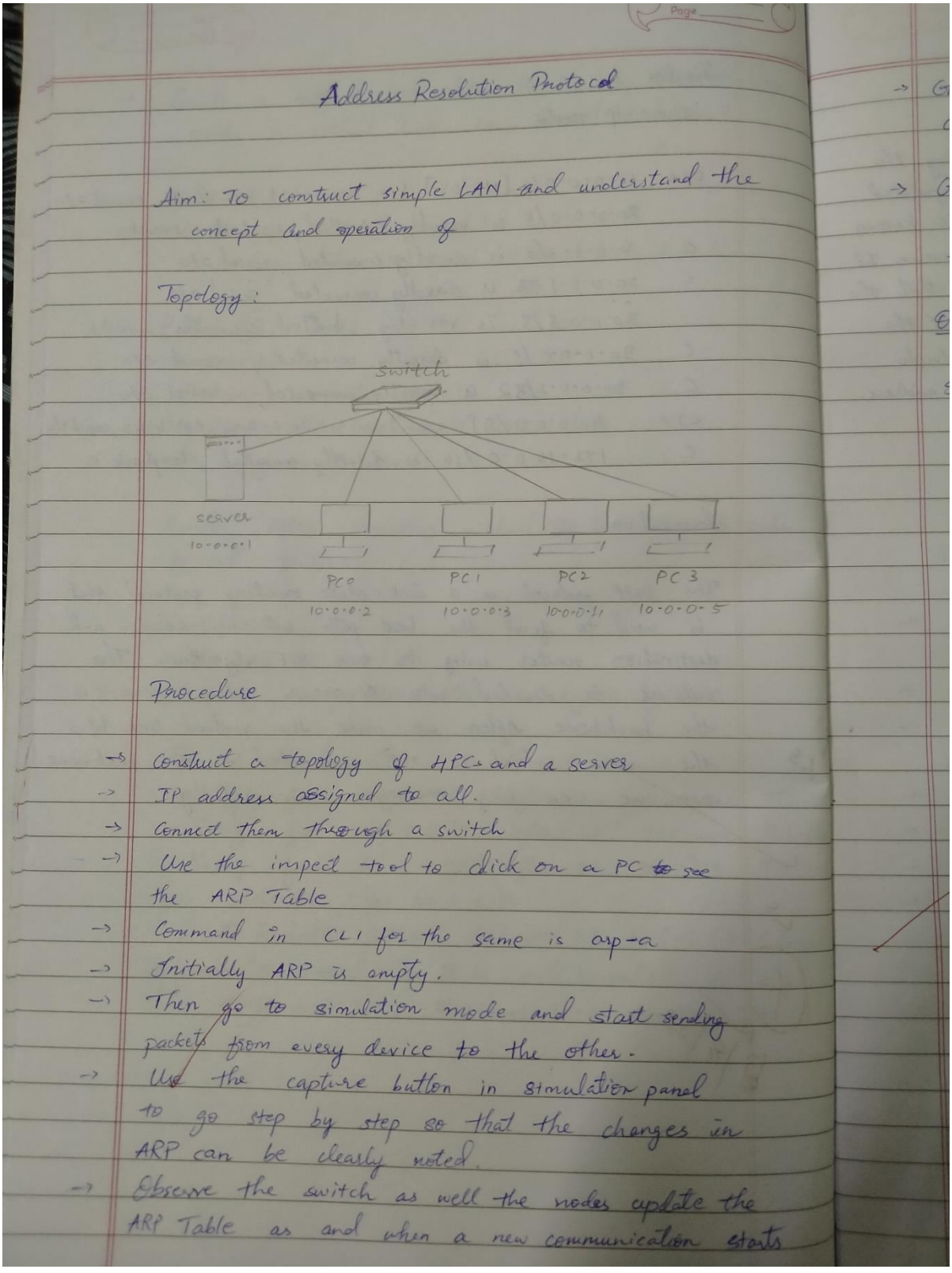
Topology and output screenshots:





Experiment 9:

Address Resolution Protocol



Address Resolution Protocol

Aim: To construct simple LAN and understand the concept and operation of

Topology:

Procedure

-> Construct a topology of HPC's and a server
-> IP address assigned to all.
-> Connect them through a switch
-> Use the inspect tool to click on a PC to see the ARP Table
-> Command in CLI for the same is arp-a
-> Initially ARP is empty.
-> Then go to simulation mode and start sending packets from every device to the other.
-> Use the capture button in simulation panel to go step by step so that the changes in ARP can be clearly noted.
-> Observe the switch as well the nodes update the ARP Table as and when a new communication starts

→ Go to the CLI of any device and give commands to check that device's ARP table.

> arp -a

→ Go to the switch CLI and type the command

> show mac address-table

to see the address table of the entire LAN.

Output:

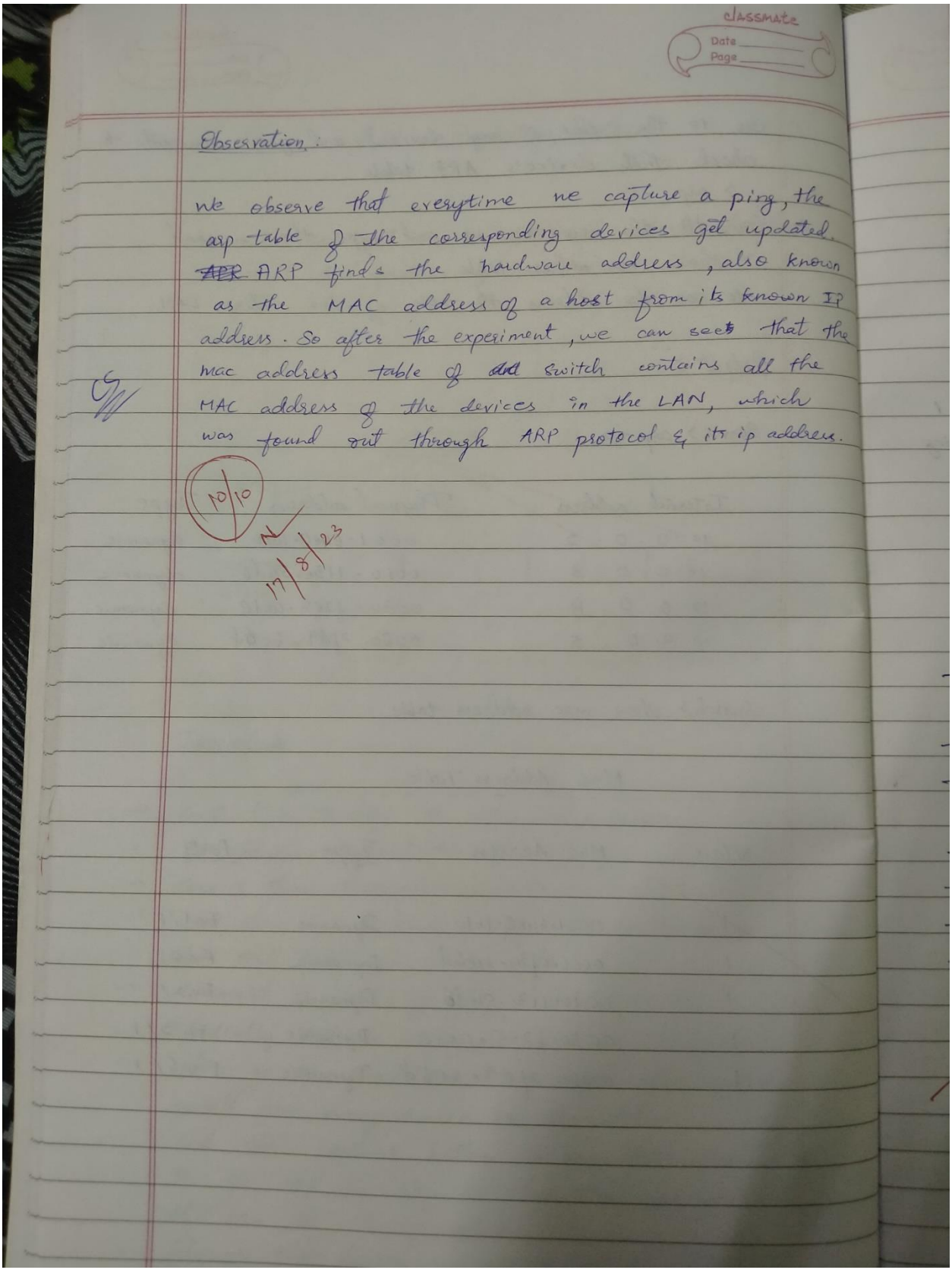
Server> arp -a

Internet address	Physical address	Type
10.0.0.2	0001.64e0.1610	dynamic
10.0.0.3	0010.113e.54d6	dynamic
10.0.0.4	0030.f2e5.ae1e	dynamic
10.0.0.5	0060.2fd9.6c6d	dynamic

Switch> show mac address-table

Mac Address Table

Vlan	Mac Address	Type	Ports
1	0001.64e0.1610	Dynamic	Fa1/1
1	000c.cf3a.4e6d	Dynamic	Fa0/1
1	0010.113e.54d6	Dynamic	Fa2/1
1	0030.f2e5.ae1e	Dynamic	Fa3/1
1	0060.2fd9.6c6d	Dynamic	Fa6/1



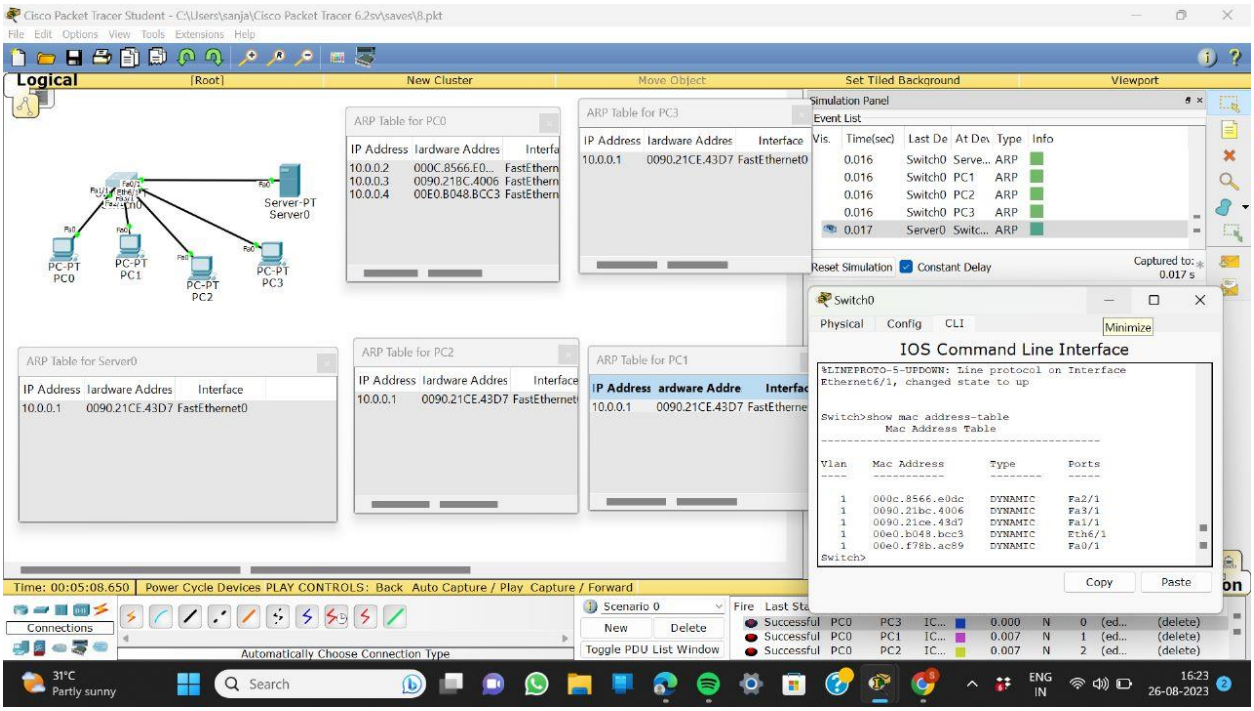
Observation:

We observe that everytime we capture a ping, the arp table of the corresponding devices get updated. ~~ARR~~ ARP finds the hardware address, also known as the MAC address of a host from its known IP address. So after the experiment, we can see that the mac address table of ~~dd~~ switch contains all the MAC address of the devices in the LAN, which was found out through ARP protocol & its ip address.

10/10

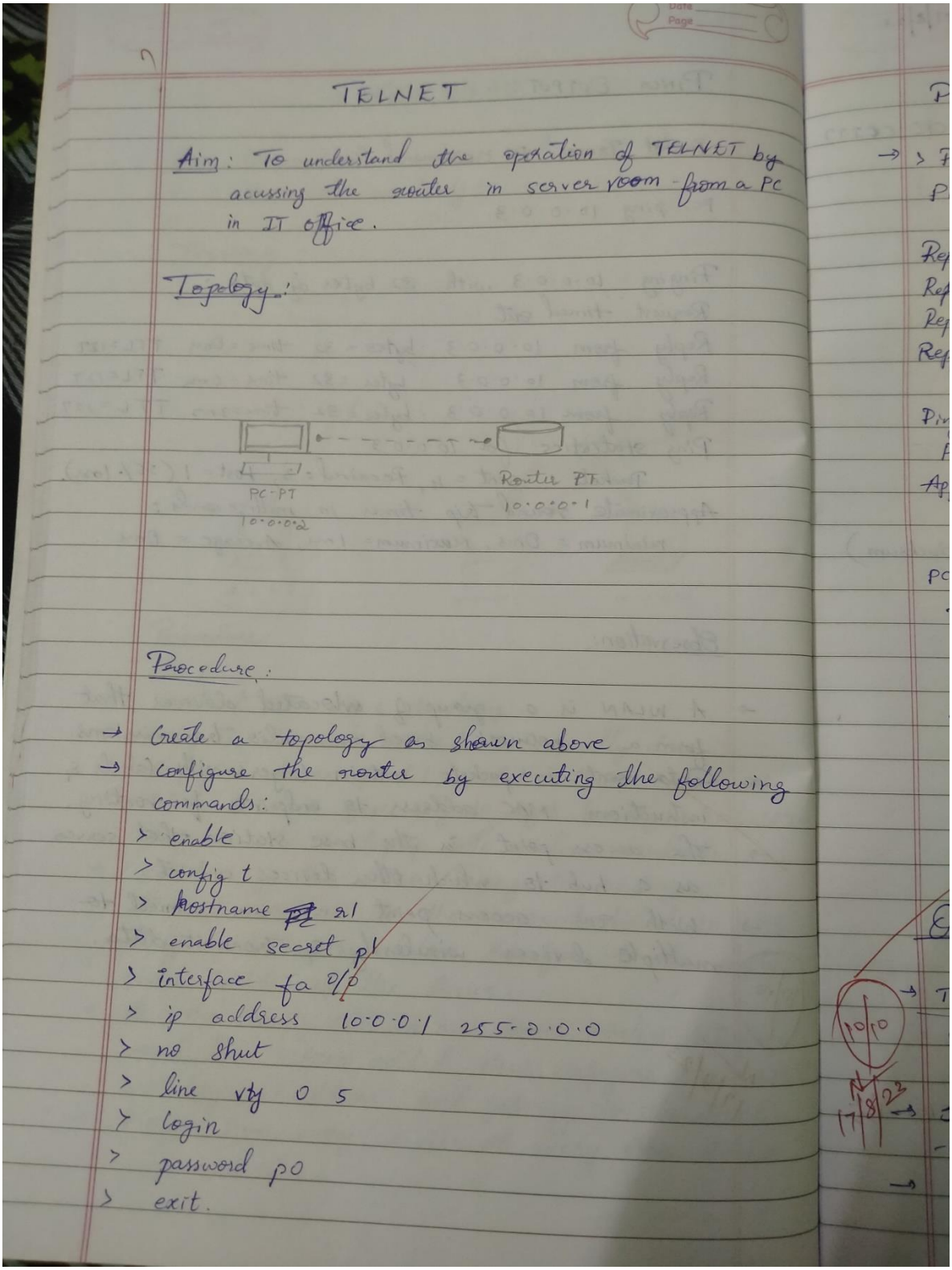
17/8/23

Topology and output screenshots:



Experiment 10:

TELNET



TELNET

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

Topology:

Procedure:

→ Create a topology as shown above
→ Configure the router by executing the following commands:

> enable
> config t
> 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.

Ping message to Router

→ > 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

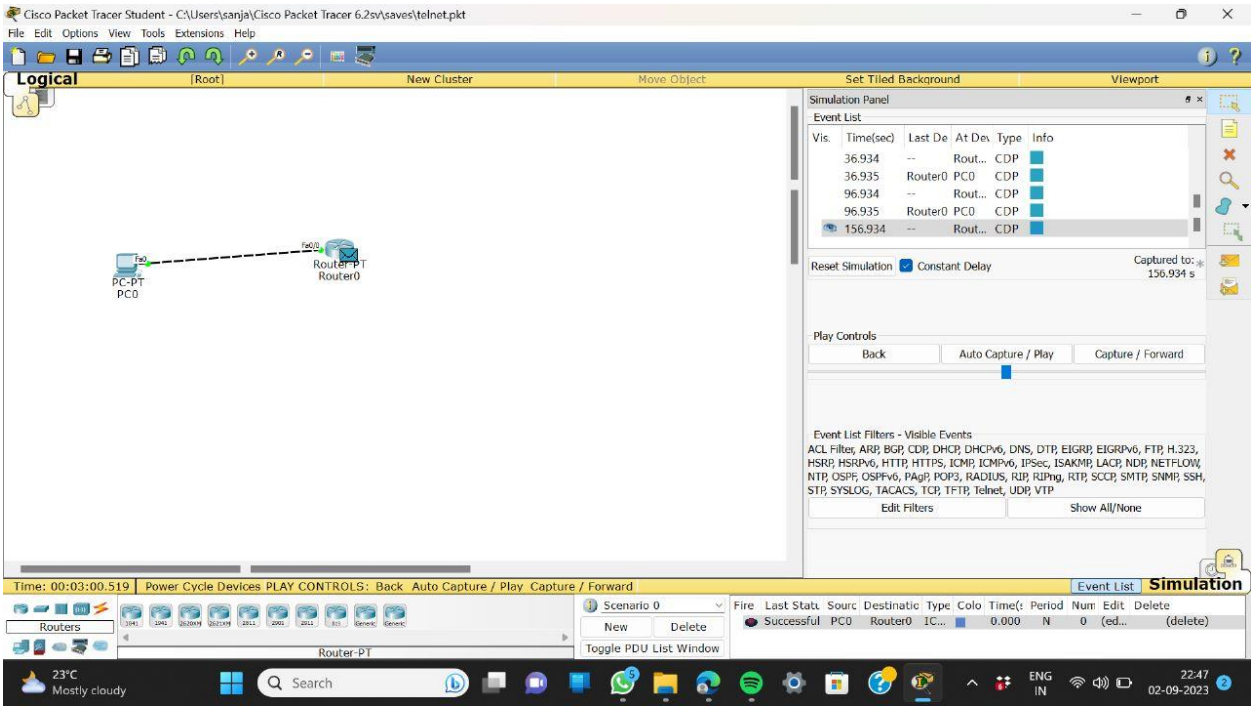
Ping statistics
Packets sent = 4, Received = 4, lost = 0 (0% loss)
Approximate round trip times in milliseconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms.

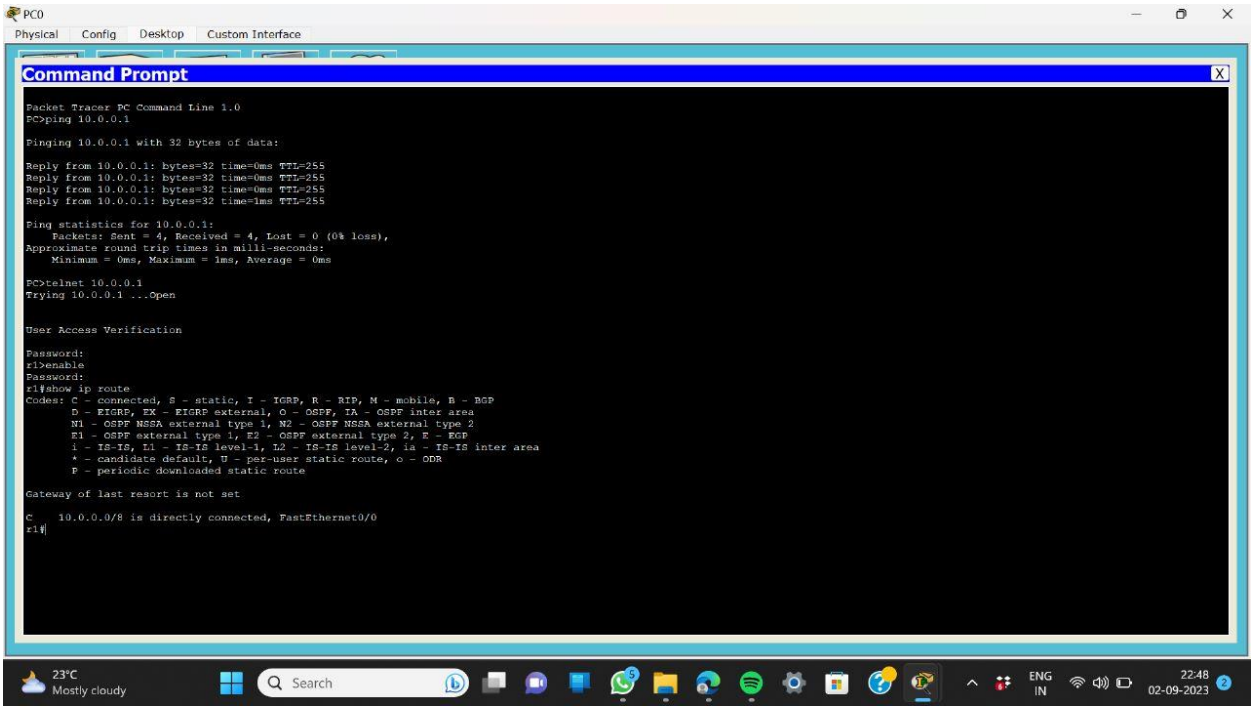
PC> Telnet 10.0.0.1
Typing 10.0.0.1 ...open
User Access verification
Password: p0
R1> enable
Password: p1
~~R1#configt~~
R1# show ip route
C 10.0.0.0/8 is directly connected, FastEthernet0/0

Observation:

→ TELNET stands for Teletype Network. It is a type of protocol that enables one compiler to connect to the local compiler.
→ It is used as a standard TCP/IP protocol for virtual terminal service provided by ISO.
→ TELNET operates on a client/server principle.

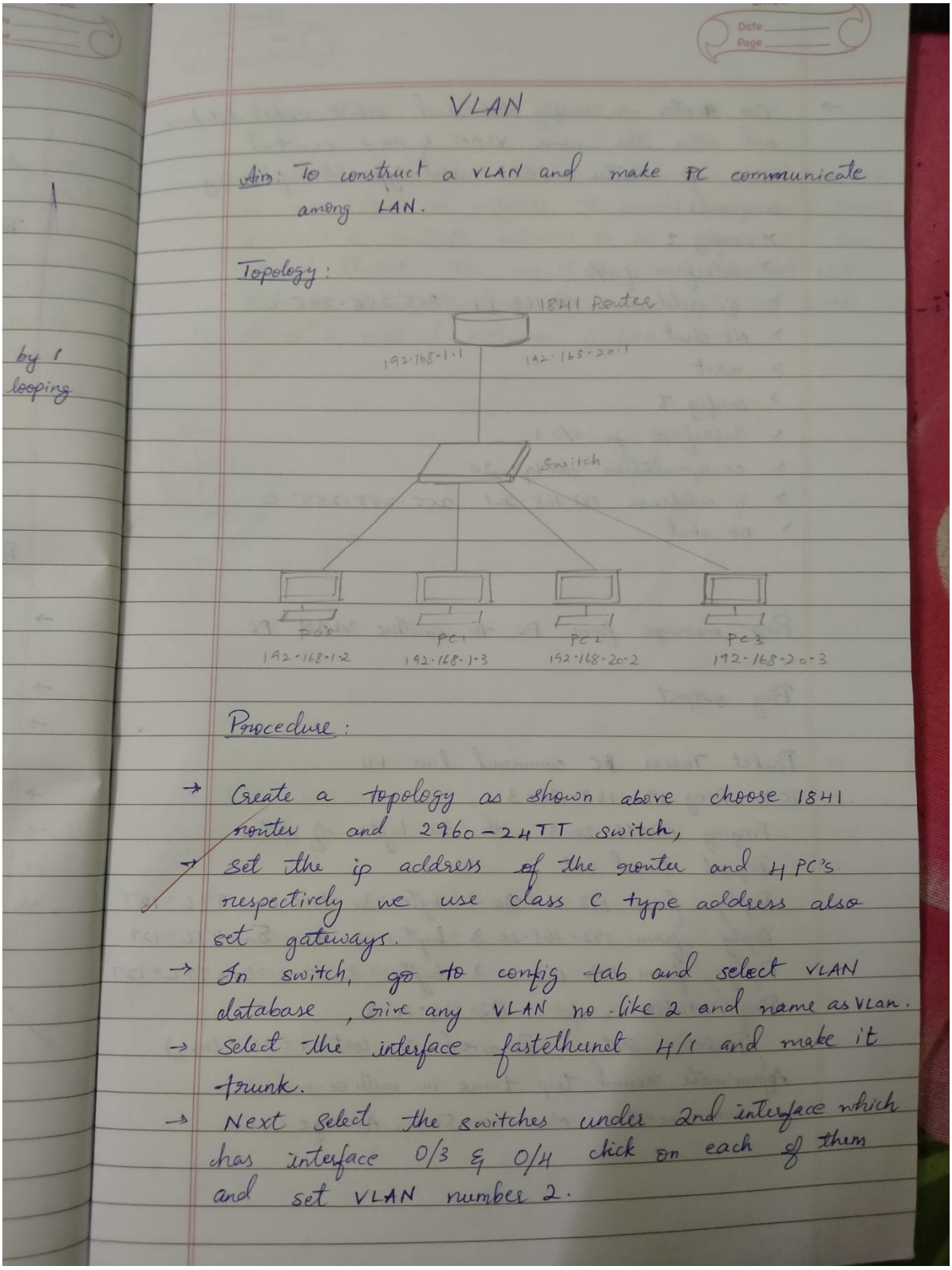
Topology and output screenshots:





Experiment 11:

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



VLAN

Aim: To construct a VLAN and make PC communicate among LAN.

Topology:

Procedure:

→ Create a topology as shown above choose 1841 router and 2960-24TT switch,
→ Set the ip address of the router and 4 PC's respectively we use class C type address also set gateways.
→ In switch, go to config tab and select VLAN database, Give any VLAN no like 2 and name as VLAN.
→ Select the interface fastethernet 4/1 and make it trunk.
→ Next select the switches under 2nd interface which has interface 0/3 & 0/4 click on each of them and set VLAN number 2.

→ Go router → config tab and select VLAN database and enter the name VLAN & no 2 created.

→ Go to router → CLI and type the following commands:

```
> config t
> interface fa0/0
> ip address 192.168.1.1 255.255.255.0
> No shut
> exit
> config t
> interface fa 0/0.1
> encapsulation dot1q 20
> ip address 192.168.20.1 255.255.255.0
> no shut
```

Ping message from PC to another VLAN PC

Ping output

```
Packet Tracer PC command line 1.0
PC> Ping 192.168.20.3
  Pinging 192.168.20.3 with 32 bytes of data
  Request timed out
  Reply from 192.168.20.3: bytes=32 time=0ms TTL=127
  Reply from 192.168.20.3: bytes=32 time=5ms TTL=127
  Reply from 192.168.20.3: bytes=32 time=0ms TTL=127
  Ping statistics for 192.168.20.3
    Packets: Sent=4, Received=3, Lost=1 (25% loss)
Approximate round trip times in milliseconds:
    Minimum=0ms, Maximum=5ms, Average=1ms
```

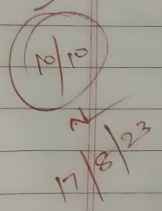
AN database
ted.
following

TTL = 127
TTL = 127
TTL = 127

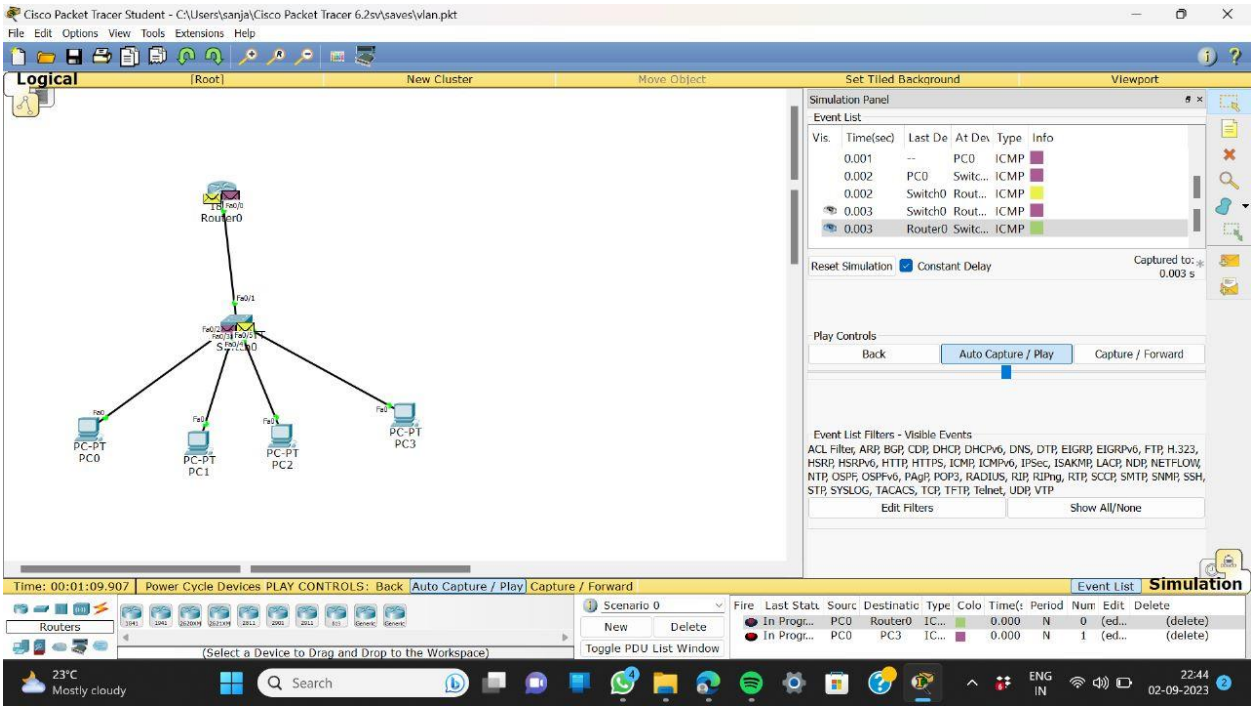
(oss)

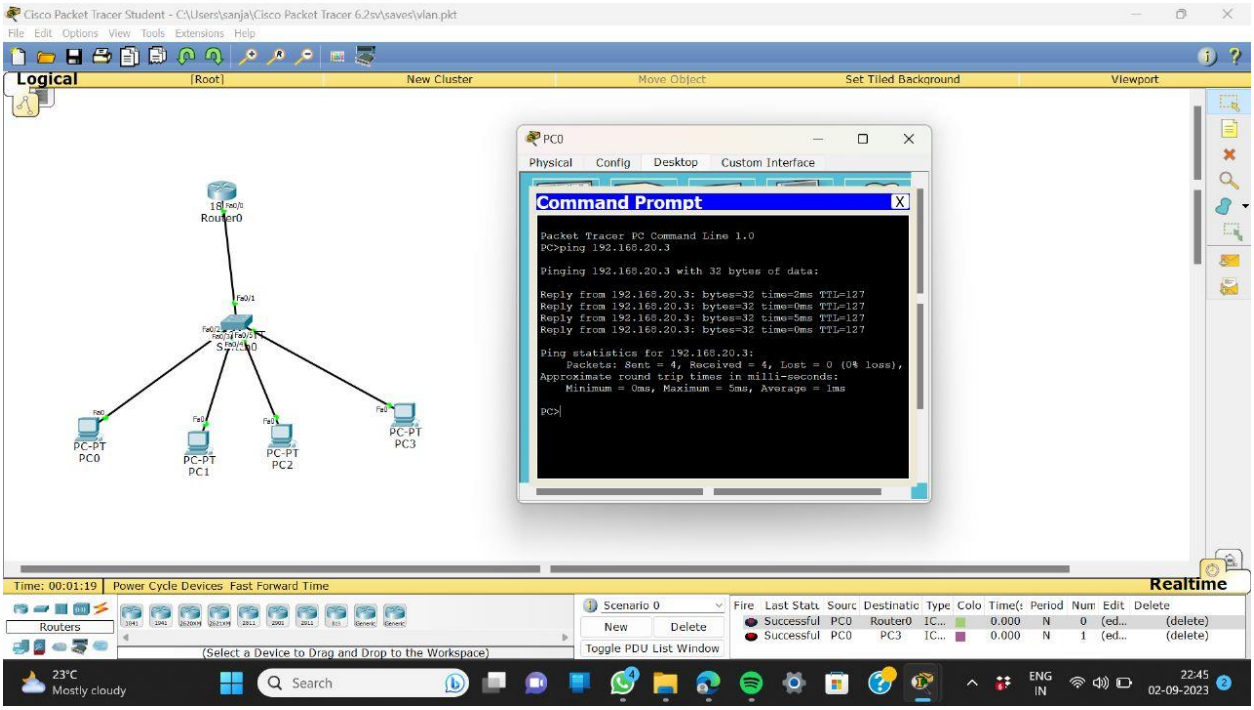
Observation:

VLAN is a custom network which is created from one or more local area networks. It enables a group of devices available in multiple networks to be combined into one logical network. The result becomes a virtual LAN that is administered exactly like a physical VLAN. It is a virtual extension of LAN.



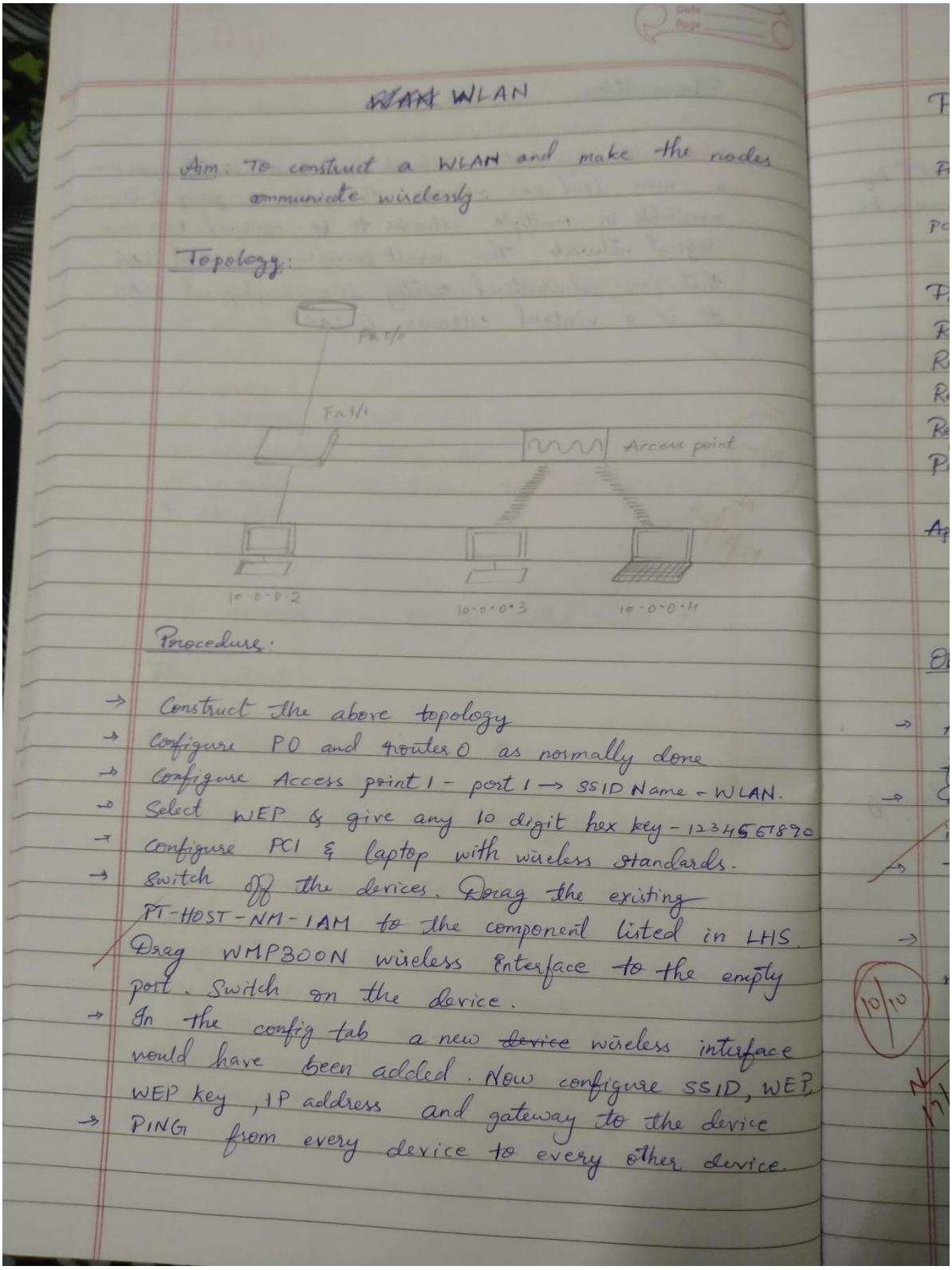
Topology and output screenshots:





Experiment 12:

To construct a WLAN and make the nodes communicate wirelessly



WLAN

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

Topology:

Procedure:

→ Construct the above topology
→ Configure PO and Router 0 as normally done
→ Configure Access point 1 - port 1 → SSID Name - WLAN.
→ Select WEP & give any 10 digit hex key - 1234567890
→ Configure PC1 & laptop with wireless standards.
→ Switch off the devices. Drag the existing PT-HOST-NM-1AM to the component listed in LHS. Drag WMP300N wireless Enterface to the empty port. Switch on the device.
→ In the config tab a new ~~device~~ wireless interface would have been added. Now configure SSID, WEP, WEP key, IP address and gateway to the device
→ PING from every device to every other device.

10/10

PING OUTPUT:

Packet Tracer PC Command line

PC> ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:
Request timed out
Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL = 127
Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL = 127
Reply from 10.0.0.3 : bytes = 32 time = 2ms TTL = 127
Ping statistics for 10.0.0.3
Packets : sent = 4, Received = 3, Lost = 1 (25% loss).
Approximate round trip times in milliseconds:
minimum = 0ms, Maximum = 1ms, Average = 0ms.

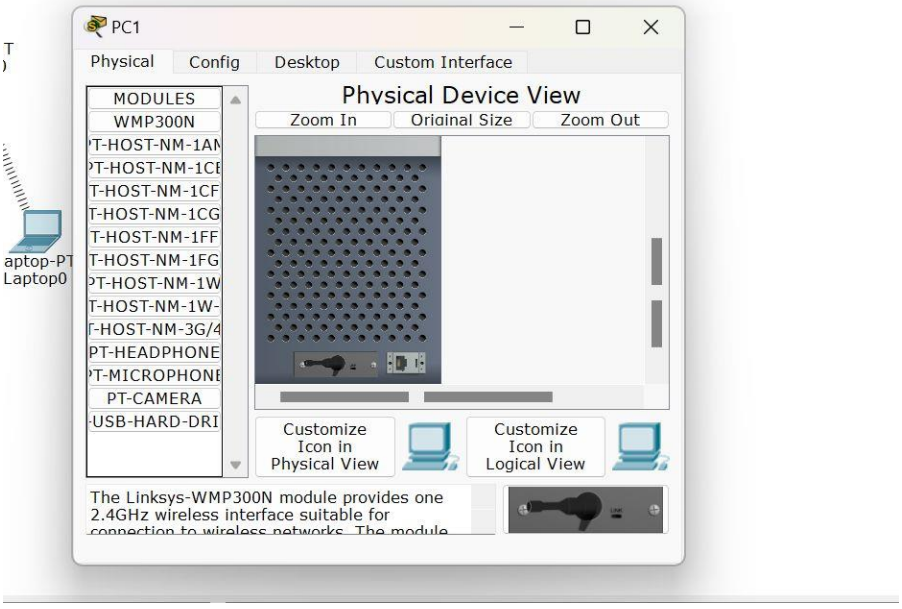
Observation:

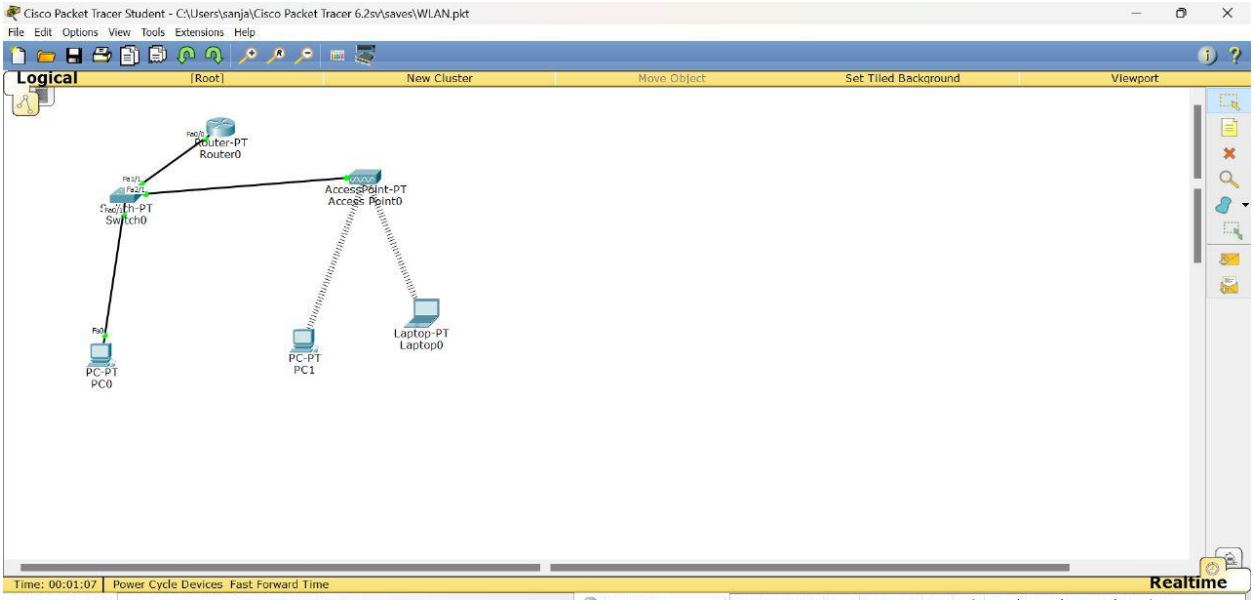
→ A WLAN is a group of colocated devices that form a network based on radio transmissions.
→ Data sent in packets contain layers with labels & instructions, MAC addresses to endpoints for routing.
→ The access point is the base station that serves as a hub to which other devices connect.
→ With one access point we can connect to multiple devices wirelessly & transmit data.

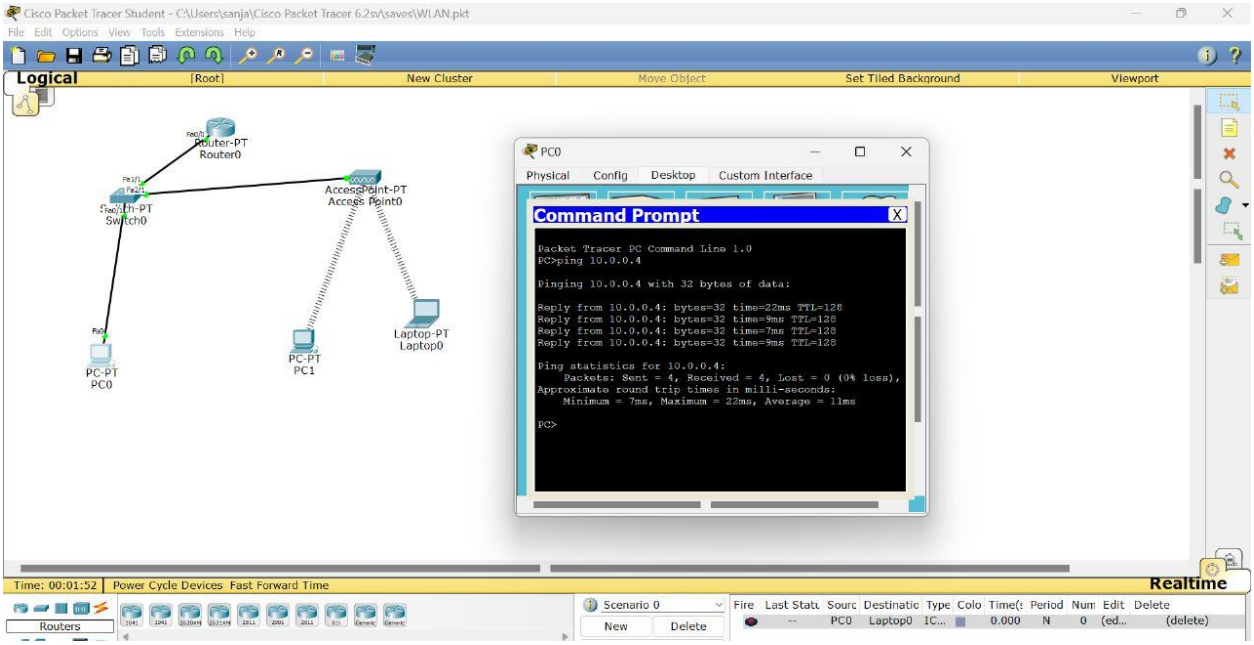
10/10

17/8/23

Topology and output screenshots:



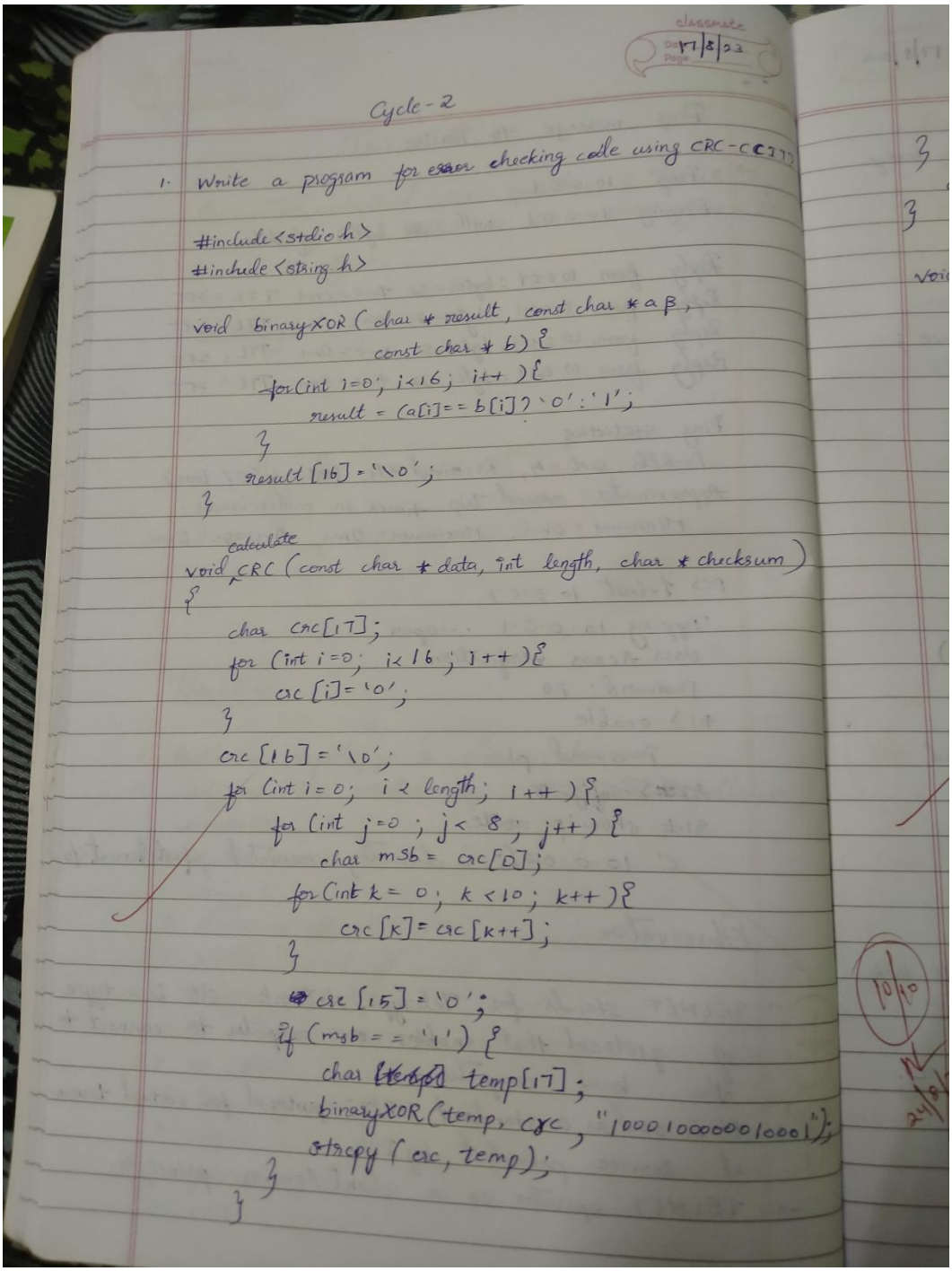

Cisco Packet Tracer Student - C:\Users\sanja\Cisco Packet Tracer 6.2sv\saves\WLAN.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Router-PT
Router0
Switch-PT
Switch0
Access Point-PT
Access Point0
PC-PT
PC0
PC-PT
PC1
Laptop-PT
Laptop0
Time: 00:01:07
Power Cycle Devices
Fast Forward Time
Realtime


Cisco Packet Tracer Student - C:\Users\sanja\Cisco Packet Tracer 6.2sv\saves\WLAN.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Router-PT
Router0
Switch-PT
Switch0
Access Point-PT
Access Point0
PC-PT
PC0
PC-PT
PC1
Laptop-PT
Laptop0
PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=22ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms 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 = 7ms, Maximum = 22ms, Average = 11ms
PC>
Time: 00:01:52
Power Cycle Devices
Fast Forward Time
Realtime
Routers
Scenario 0
New
Delete
Fire
Last Statu
Sourc
Destinatio
Type
Colo
Time(s
Period
Num
Edit
Delete
PC0
Laptop0
0.000
N
0
(ed...
(delete)

Cycle 2

Experiment 13:

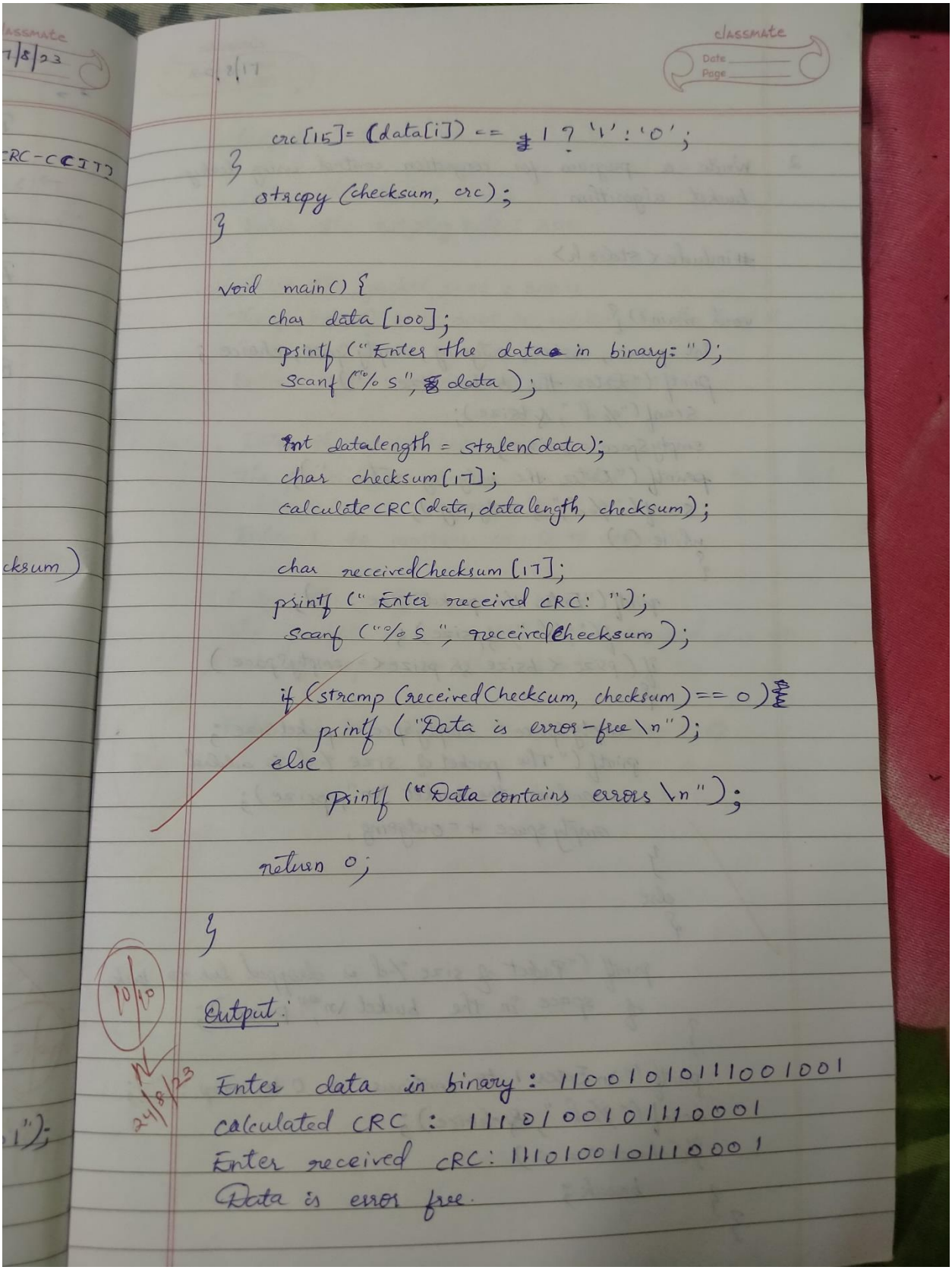
Write a program for error detecting code using CRC-CCITT (16-bits).



Cycle-2

1. Write a program for error checking code using CRC-CCITT

```
#include<stdio.h>
#include<string.h>

void binaryXOR(char *result, const char *a,
               const char *b){
    for(int i=0; i<16; i++){
        result = (a[i]==b[i])?'0':'1';
    }
    result[16]='\0';
}

void calculateCRC(const char *data, int length, char *checksum)
{
    char crc[17];
    for(int i=0; i<16; i++){
        crc[i]='0';
    }
    crc[16]='\0';
    for(int i=0; i<length; i++){
        for(int j=0; j<8; j++){
            char msb = crc[0];
            for(int k=0; k<16; k++){
                crc[k]=crc[k++];
            }
            crc[15]='0';
            if(msb=='1'){
                char temp[17];
                binaryXOR(temp, crc, "1000100000010001");
                strcpy(crc, temp);
            }
        }
    }
```



```
    crc[15] = (data[i]) == '1' ? '1' : '0';
}
strcpy(checksum, crc);
}

void main() {
    char data[100];
    printf("Enter the data in binary: ");
    scanf("%s", data);

    int datalength = strlen(data);
    char checksum[17];
    calculateCRC(data, datalength, checksum);

    char receivedChecksum[17];
    printf("Enter received CRC: ");
    scanf("%s", receivedChecksum);

    if (strcmp(receivedChecksum, checksum) == 0)
        printf("Data is error-free\n");
    else
        printf("Data contains errors\n");

    return 0;

}
```

Output:

```
Enter data in binary : 1100101011001001
Calculated CRC : 1110100101110001
Enter received CRC: 1110100101110001
Data is error free.
```

Screenshots:

crc.c > calculateCRC(const char *, int, char *)

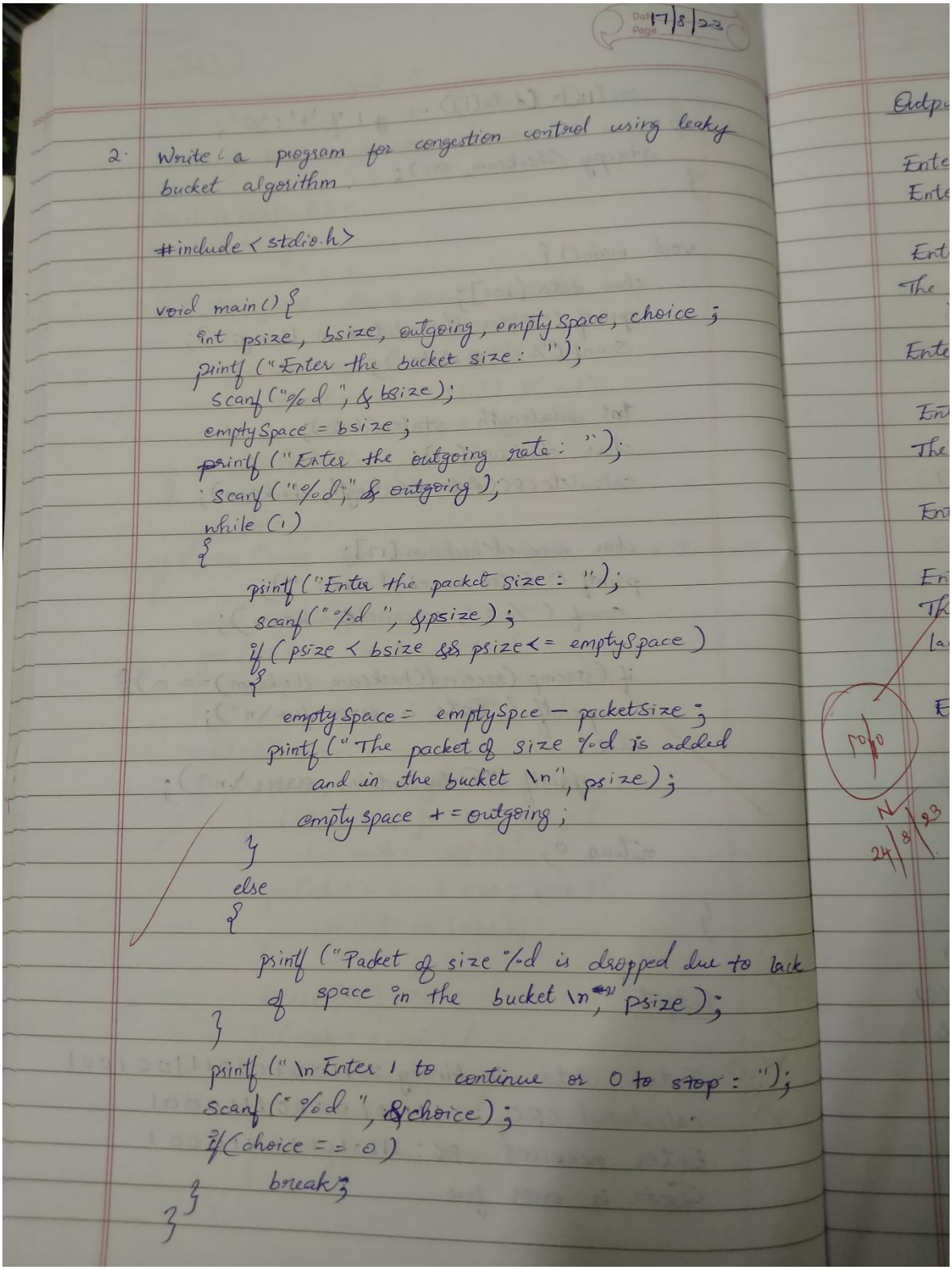
```c
#include <stdio.h>
#include <string.h>

// Function to perform bitwise XOR on binary strings
void binaryXOR(char *result, const char *a, const char *b)
{
  for (int i = 0; i < 16; i++)
    result[i] = (a[i] == b[i]) ? '0' : '1';
  result[16] = '\0';
}

// Function to calculate CRC-CCITT checksum
void calculateCRC(const char *data, int length, char *checksum)
{
  char crc[17];
  for (int i = 0; i < 16; i++)
    crc[i] = '0';
  crc[16] = '\0';

  for (int i = 0; i < length; i++)
  {
    for (int j = 0; j < 8; j++)
    {
      char msb = crc[0];
      for (int k = 0; k < 16; k++)
        crc[k] = crc[k + 1];
      crc[15] = '0';

      if (msb == '1')
      {
        char temp[17];
        binaryXOR(temp, crc, "10001000000100001"); // CRC_POLY in binary
        strcpy(crc, temp);
      }
    }
    crc[15] = (data[i] == '1') ? '1' : '0';
  }

  strcpy(checksum, crc);
}

int main()
{
  char data[100];
  printf("Enter data in binary: ");
  scanf("%s", data);
```

```
41
42  int main()
43  {
44      char data[100];
45      printf("Enter data in binary: ");
46      scanf("%s", data);
47
48      int dataLength = strlen(data);
49      char checksum[17];
50      calculateCRC(data, dataLength, checksum);
51
52      printf("Calculated CRC: %s\n", checksum);
53
54      // Verify the received data
55      char receivedChecksum[17];
56      printf("Enter received CRC: ");
57      scanf("%s", receivedChecksum);
58
59      if (strcmp(receivedChecksum, checksum) == 0)
60          printf("Data is error-free.\n");
61      else
62          printf("Data contains errors.\n");
63      return 0;
64  }
```

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\CRC 0> ./crc
Enter data in binary: 11001010111001001
Calculated CRC: 1110100101110001
Enter received CRC: 1110100101110001
Data is error-free.
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\CRC 0>
```

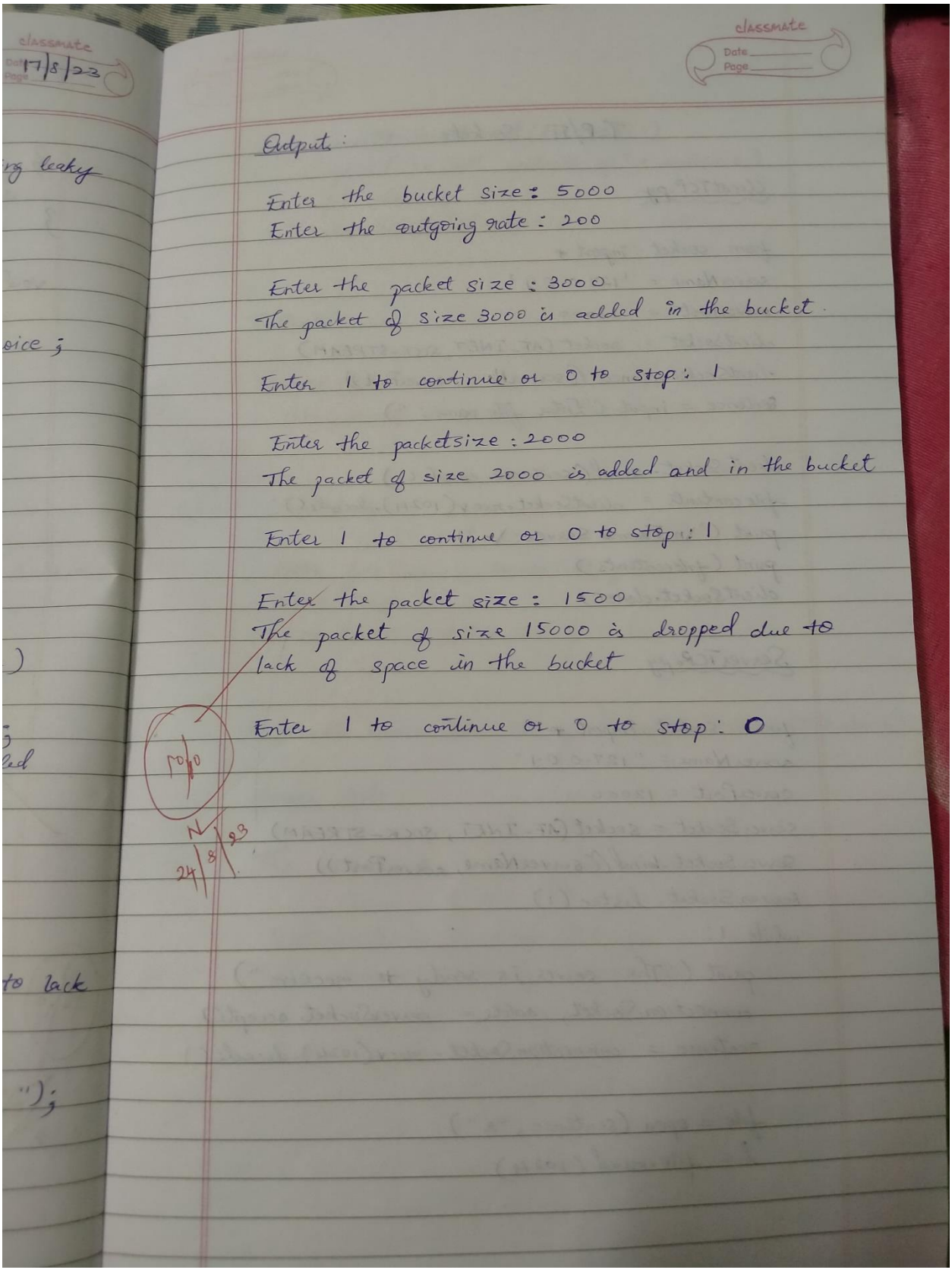
Experiment 14:

Write a program for congestion control using Leaky bucket algorithm.



2. Write a program for congestion control using leaky bucket algorithm

```
#include <stdio.h>

void main(){
    int psize, bsize, outgoing, emptySpace, choice;
    printf("Enter the bucket size: ");
    scanf("%d", &bsize);
    emptySpace = bsize;
    printf("Enter the outgoing rate: ");
    scanf("%d", &outgoing);
    while(1)
    {
        printf("Enter the packet size: ");
        scanf("%d", &psize);
        if(psize < bsize && psize <= emptySpace)
        {
            emptySpace = emptySpce - packetSize;
            printf("The packet of size %d is added
                and in the bucket \n", psize);
            emptyspace += outgoing;
        }
        else
        {
            printf("Packet of size %d is dropped due to lack
                of space in the bucket \n", psize);
        }
        printf("\n Enter 1 to continue or 0 to stop: ");
        scanf("%d", &choice);
        if(choice == 0)
            break;
    }
}
```



Outputs:

Enter the bucket size: 5000
Enter the outgoing rate: 200

Enter the packet size: 3000
The packet of size 3000 is added in the bucket.

Enter 1 to continue or 0 to stop: 1

Enter the packetsize: 2000
The packet of size 2000 is added and in the bucket

Enter 1 to continue or 0 to stop: 1

Enter the packet size: 1500
The packet of size 15000 is dropped due to lack of space in the bucket

Enter 1 to continue or 0 to stop: 0

10/10

24/8/23

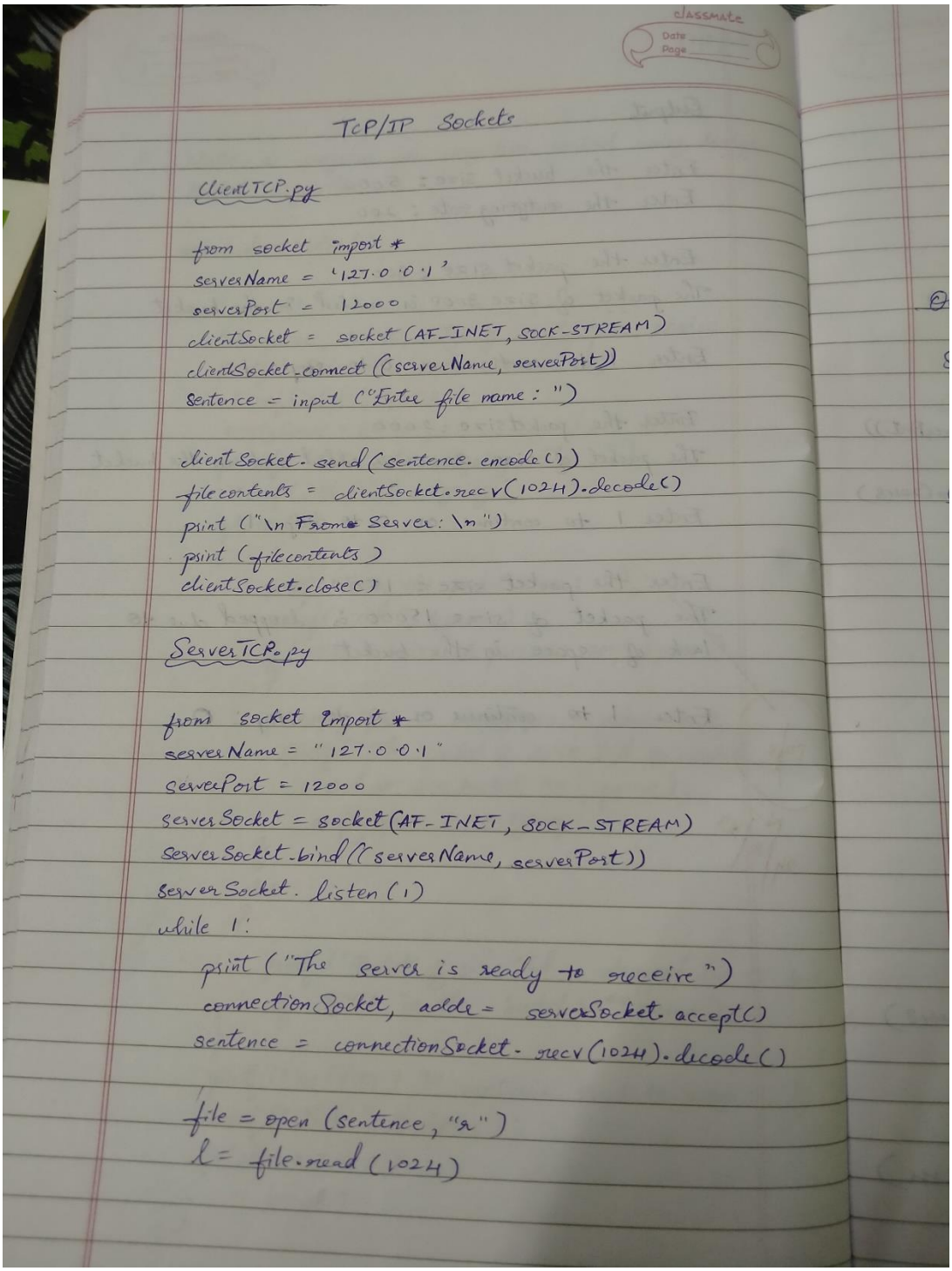
Screenshots:

```c
#include <stdio.h>

void main()
{
  int psize, bsize, outgoing, emptyspace, choice;
  printf("Enter the Bucket size = ");
  scanf("%d", &bsize);
  emptyspace = bsize;
  printf("Enter the outgoing rate = ");
  scanf("%d", &outgoing);
  while (1)
  {
    printf("\nEnter the packet size = ");
    scanf("%d", &psize);
    if (psize <= bsize && psize <= emptyspace)
    {
      emptyspace = emptyspace - psize;
      printf("The Packet of size %d is added and in the bucket \n", psize);
    }
    else
      printf("The Packet of size %d is dropped due to lack of space in the bucket\n");
    if ((emptyspace + outgoing) < bsize)
      emptyspace += outgoing;
    else if (bsize - emptyspace > 0)
      emptyspace = bsize;
    printf("\nEnter 1 to Continue or 0 to Stop: ");
    scanf("%d", &choice);
    if (choice == 0)
      break;
  }
}
```

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\Leaky bucket> gcc leakybucket.c -o leakybucket
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\Leaky bucket> ./leakybucket
Enter the Bucket size = 5000
Enter the outgoing rate = 200

Enter the packet size = 3000
The Packet of size 3000 is added and in the bucket

Enter 1 to Continue or 0 to Stop: 1

Enter the packet size = 2000
The Packet of size 2000 is added and in the bucket

Enter 1 to Continue or 0 to Stop: 1

Enter the packet size = 1500
The Packet of size 1500 is dropped due to lack of space in the bucket

Enter 1 to Continue or 0 to Stop: 0
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\Leaky bucket>
```

Experiment 15:

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.



TCP/IP Sockets

ClientTCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Enter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("\n From Server: \n")
print(filecontents)
clientSocket.close()
```

ServerTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

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

```
connectionSocket.send(l.encode())
print("\n Sent contents of" + sentence)
file.close()
connectionSocket.close().
```

Output:

Server Side:

The server is ready to receive

Client Side:

Enter file name: serverTCP.py
From Server:
from socket import *
:
:
(code under serverTCP.py is printed as written above)

Server Side:

The server is ready to receive

Sent contents of serverTCP.py
The server is ready to receive.

Screenshots:

```python
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))

filename = input("Enter the filename: ")
clientSocket.send(filename.encode())
fileContents = clientSocket.recv(1024).decode()
print("\n File contents :\n")
print(fileContents)

clientSocket.close()
```

```python
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
  print("Server is listening")
  connectionSocket, addr = serverSocket.accept()
  filename = connectionSocket.recv(1024).decode()

  file = open(filename,"r")
  fileContents = file.read(1024)

  connectionSocket.send(fileContents.encode())
  print("\nSent file contents of " + filename)
  file.close()
  connectionSocket.close()

```

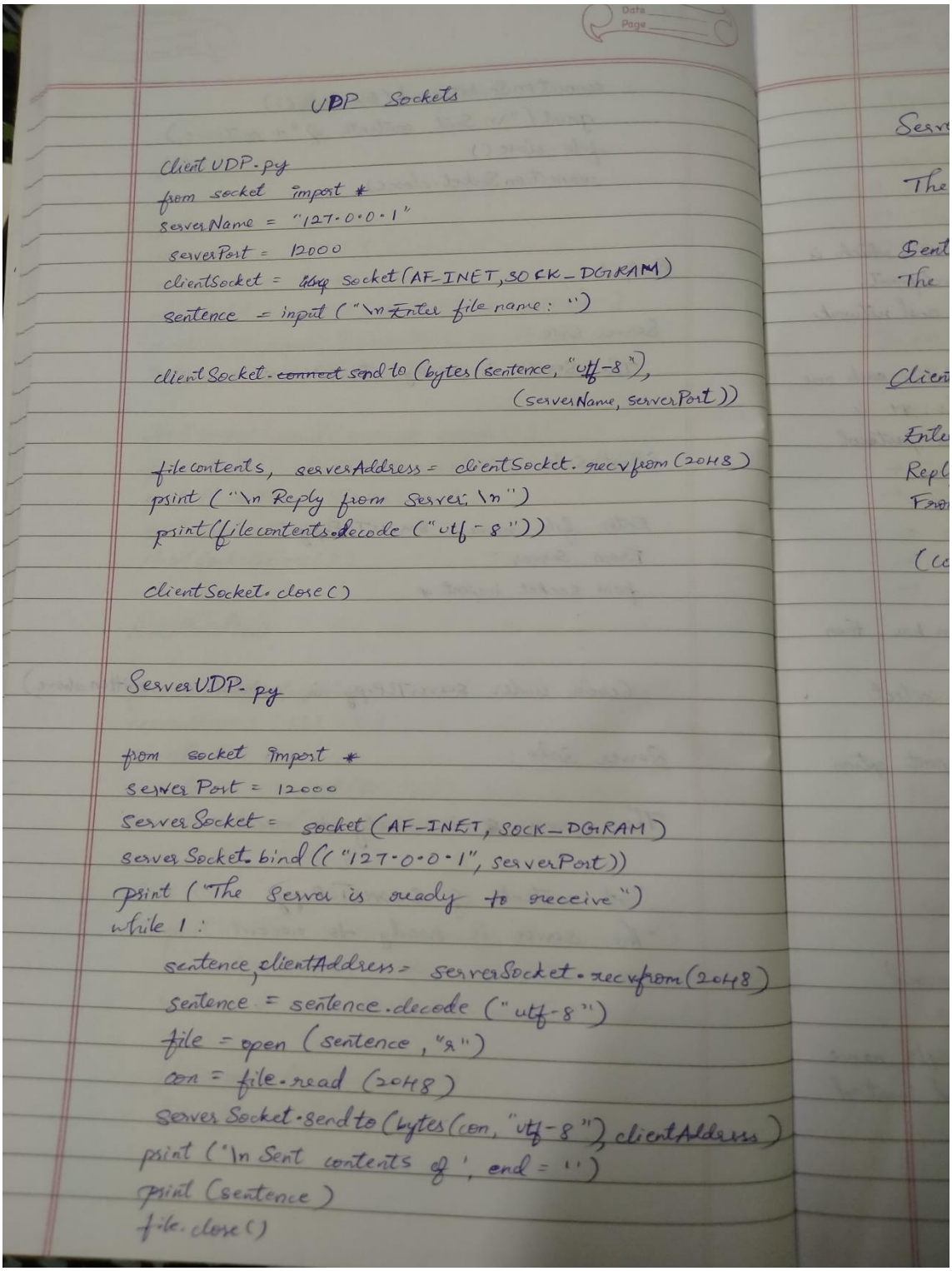
Output:

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\TCP> python server.py
Server is listening

Sent file contents of server.py
Server is listening
```

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\TCP> python client.py
Enter the filename: server.py

 File contents :

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
  print("Server is listening")
  connectionSocket, addr = serverSocket.accept()
  filename = connectionSocket.recv(1024).decode()

  file = open(filename,"r")
  fileContents = file.read(1024)

  connectionSocket.send(fileContents.encode())
  print("\nSent file contents of " + filename)
  file.close()
  connectionSocket.close()


PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\TCP>
```

Experiment 16:

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.



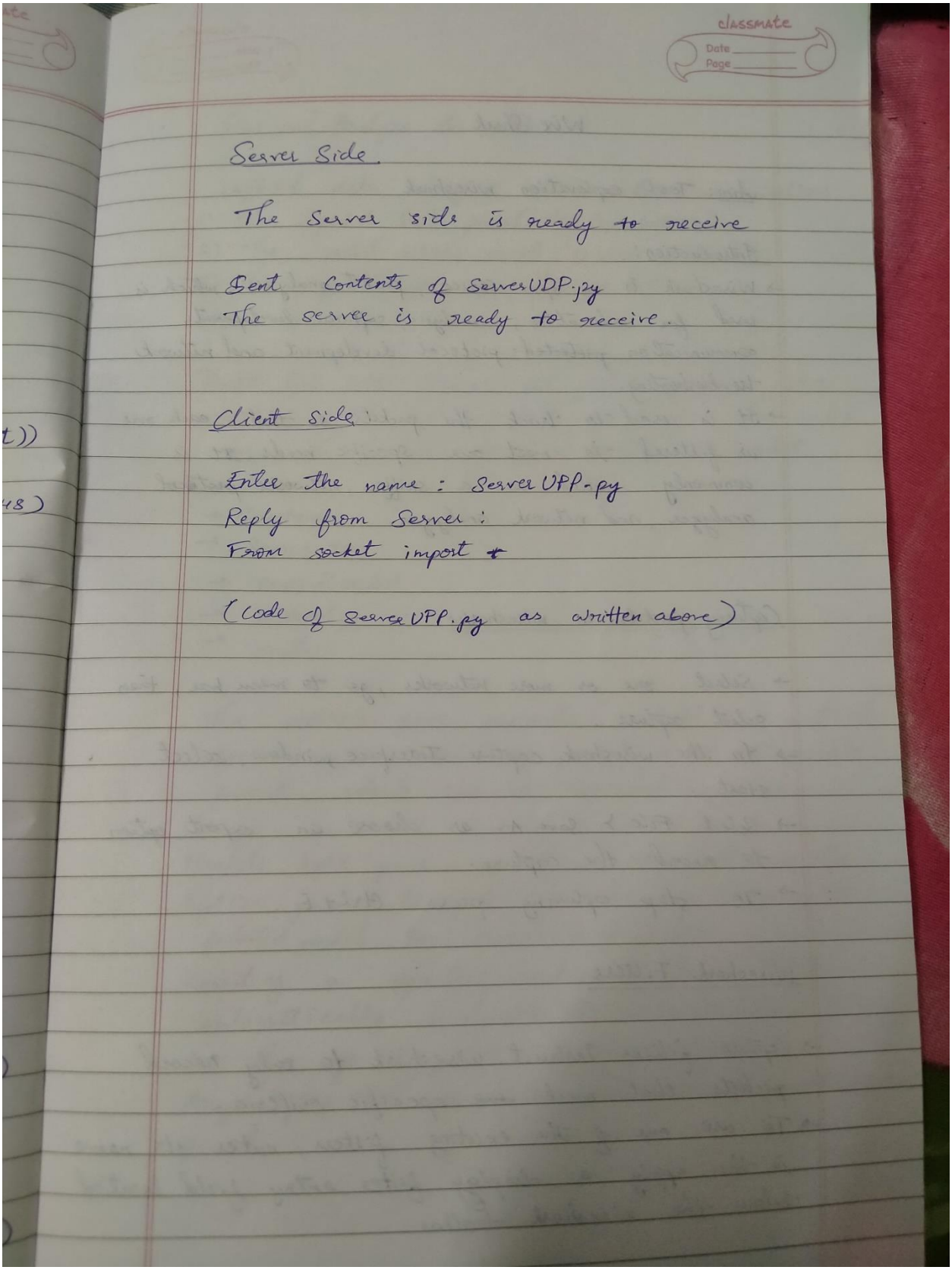
UDP Sockets

Client UDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_DGRAM)
sentence = input("\nEnter file name: ")

clientSocket.sendto(bytes(sentence,"utf-8"),
                    (serverName, serverPort))

fileContents, serverAddress = clientSocket.recvfrom(2048)
print("\n Reply from Server:\n")
print(fileContents.decode("utf-8"))

clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(("127.0.0.1",serverPort))
print("The Server is ready to receive")
while 1:
    sentence,clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence,"r")
    con = file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
print('\n Sent contents of ', end = '')
print(sentence)
file.close()
```



Server Side

The Server side is ready to receive

Sent Contents of ServerUDP.py
The server is ready to receive.

Client Side:

Enter the name : ServerUPP.py
Reply from Server :
From socket import *

(Code of ServerUPP.py as written above)

Screenshots:

```python
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_DGRAM)

filename = input("Enter filename: ")

clientSocket.sendto(bytes(filename,"utf-8"),(serverName,serverPort))

fileContents, serverAddress = clientSocket.recvfrom(2048)
print("\nFile contents: \n")
print(fileContents.decode("utf-8"))

clientSocket.close()
```

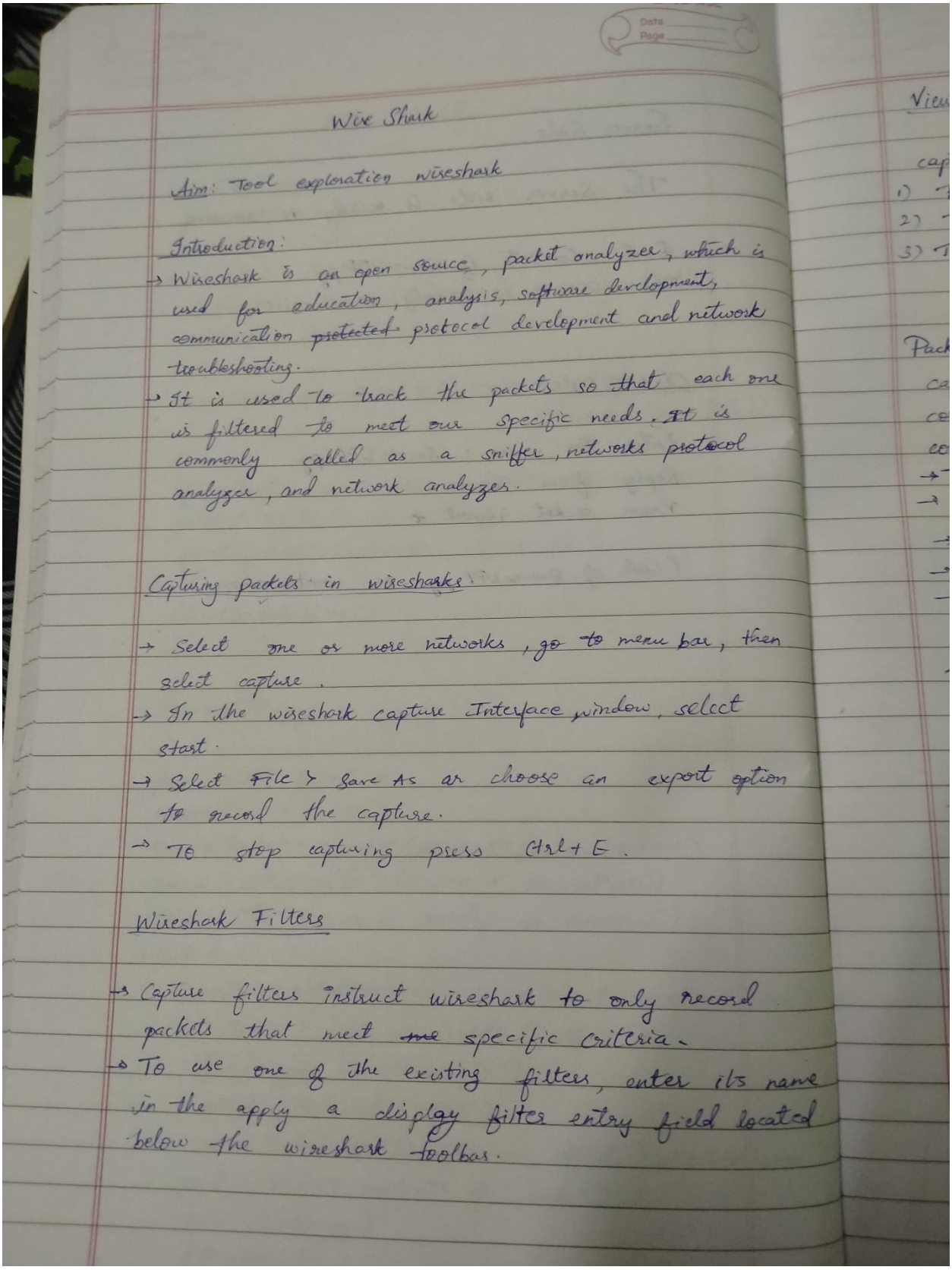
```python
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind((serverName, serverPort))

while 1:
    print("Server is listening")
    filename, clientAddress = serverSocket.recvfrom(2048)
    filename = filename.decode("utf-8")

    file = open(filename, "r")
    fileContents = file.read(2048)
    serverSocket.sendto(bytes(fileContents, "utf-8"), clientAddress)
    print("\nSent file contents of file " + filename)
    file.close()

```

Output:

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\UDP> python client.py
Enter filename: server.py

File contents:

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind((serverName, serverPort))

while 1:
    print("Server is listening")
    filename, clientAddress = serverSocket.recvfrom(2048)
    filename = filename.decode("utf-8")

    file = open(filename, "r")
    fileContents = file.read(2048)
    serverSocket.sendto(bytes(fileContents, "utf-8"), clientAddress)
    print("\nSent file contents of file " + filename)
    file.close()

PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\UDP>
```

```
PS D:\BMSCE\Academics\Semester IV\Computer networks\Lab\UDP> python server.py
Server is listening

Sent file contents of file server.py
Server is listening
```

Tool Exploration - Wireshark



Wire Shark

Aim: Tool exploration wireshark

Introduction:

→ Wireshark is an open source, packet analyzer, which is used for education, analysis, software development, communication protocol development and network troubleshooting.

→ It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, networks protocol analyzer, and network analyzer.

Capturing packets in wiresharks

→ Select one or more networks, go to menu bar, then select capture.

→ In the wireshark capture Interface window, select start.

→ Select File > Save As or choose an export option to record the capture.

→ To stop capturing press Ctrl + E.

Wireshark Filters

→ Capture filters instruct wireshark to only record packets that meet specific criteria.

→ To use one of the existing filters, enter its name in the apply a display filter entry field located below the wireshark toolbar.

View and Analyze the packets

captured data interface contains three main sections;
1) The packet list panel (the top section)
2) The packet details panel (middle section)
3) The packet bytes pane (bottom section)

Packet list pane shows all packets found in active capture file. Each packet has its own row and a corresponding number assigned to it. Each packet contain
- → Timestamp
- → Source IP
- → Destination IP
- → Protocol
- → Length

The details pane, presents protocol and protocol fields of the selected packet in a collapsible format, which can be expanded on click.

Packets byte pane is present at the bottom of the bytes pane, which displays the raw data of the selected packet in hexadecimal bytes.
Selecting a specific position of this data automatically highlights its corresponding section in the packet details and vice versa.
Any bytes that cannot be printed are represented by a period.