

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on COMPUTER NETWORKS LAB

Submitted by

CHANDRASEKHAR PATIL (1BM21CS043)

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 JUN-2023 to SEP-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 LAB” carried out by **CHANDRASEKHAR PATIL (1BM21CS043)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 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.

M Lakshmi Neelima

Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

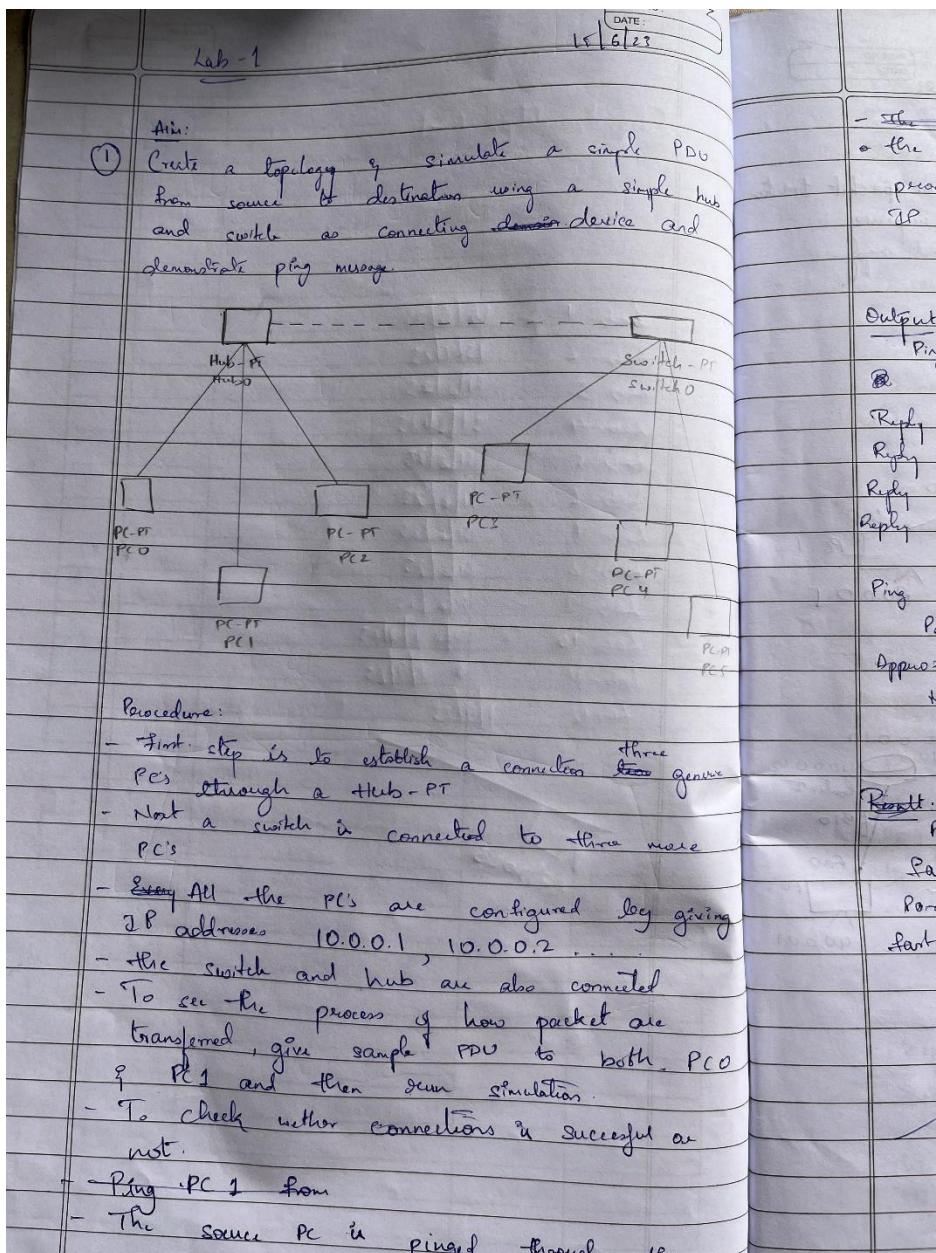
Index

Sl. No.	Date	Experiment Title	Page No.
CYCLE - 1			
1	15/06/2023	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	1-10
2	22/06/2023	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	11-29
3	13/07/2023	Configure default route, static route to the Router.	30-36
4	13/07/2023	Configure DHCP within a LAN and outside LAN.	37-48
5	20/07/2023	Configure Web Server, DNS within a LAN.	49-52
6	20/07/2023	Configure RIP routing Protocol in Routers	53-65
7	27/07/2023	Configure OSPF routing protocol	66-78
8	03/08/2023	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	79-88
9	10/08/2023	To construct a VLAN and make the PC's communicate among a VLAN.	89-98
10	10/08/2023	Demonstrate the TTL/ Life of a Packet.	99-106
11	10/08/2023	To construct a WLAN and make the nodes communicate wirelessly.	107-117
12	10/08/2023	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	118-123
CYCLE - 2			
13	17/08/2023	Write a program for error detecting code using CRC CCITT (16-bits).	124-129
14	17/08/2023	Write a program for congestion control using Leaky bucket algorithm.	130-133
15	24/08/2023	Using TCP/IP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.	134-138
16	24/08/2023	Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.	139-143
17	31/08/2023	Tool Exploration -Wireshark	144-146

LAB 1

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

OBSERVATION:



single PDG
single hub
ice and

- the destination PC
- the destination PC, by going to command prompt, then run ping followed by IP address of PC.

Output:

Pinging 10.0.0.2 with 32 bytes of data:

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

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

Three
four
generations

more

by giving

selected

are

with PC0

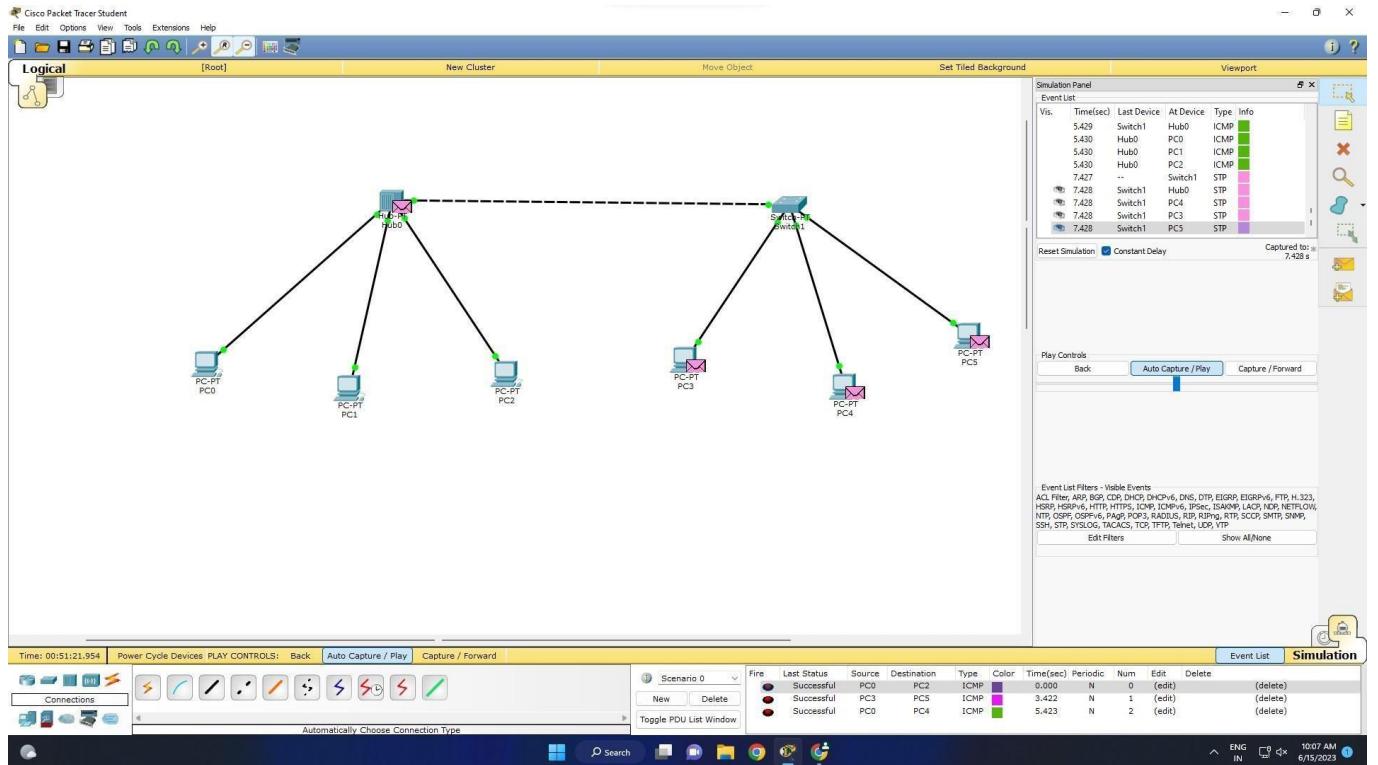
useful or

Result: switch

Port	IP address
Port 1	10.0.0.4
Port Ethernet 0	10.0.0.2
Port 2	10.0.0.1

N
15/6/2023

OUTPUT:



PC0

Physical Config Desktop Custom Interface

Command Prompt

```

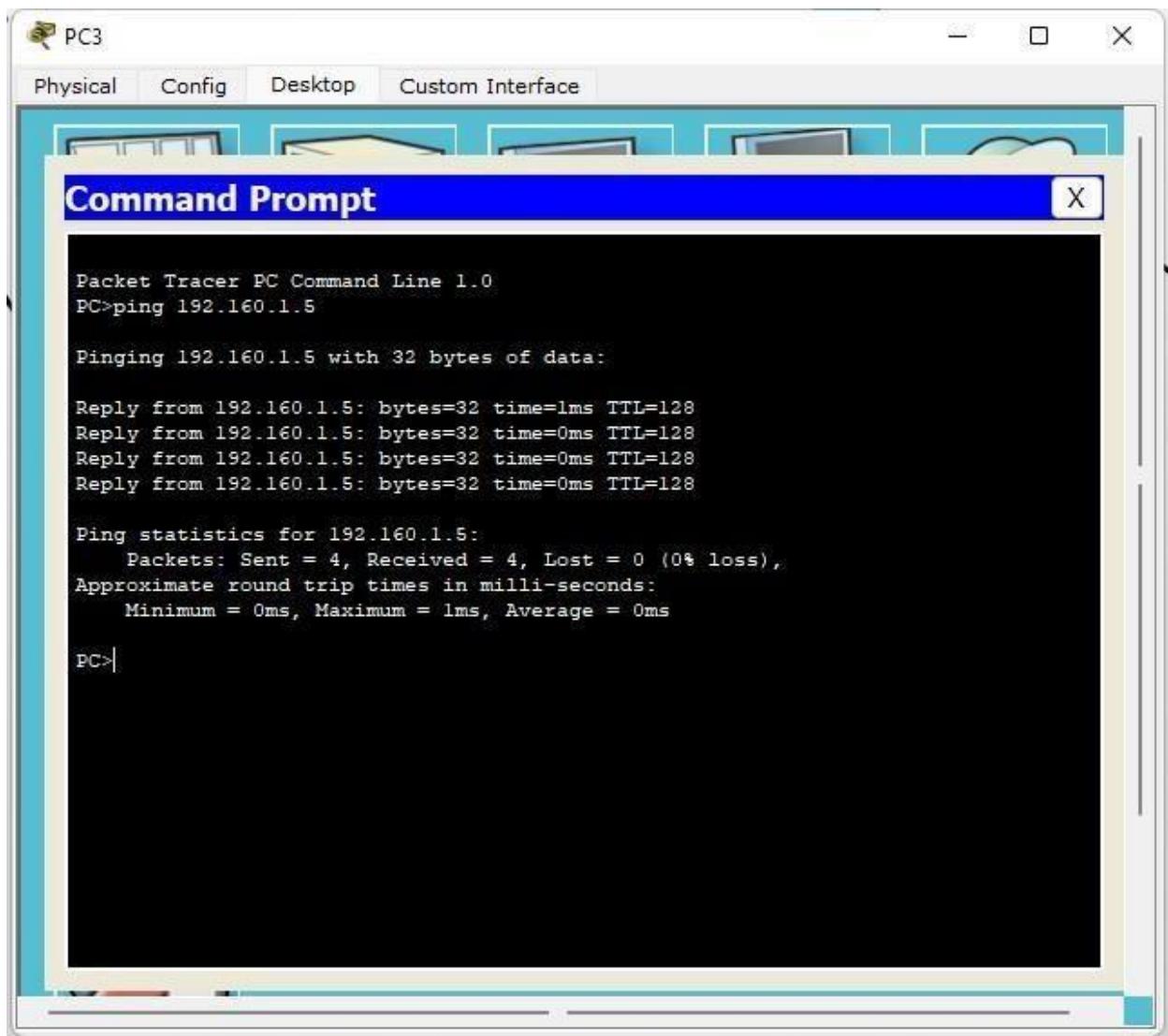
Packet Tracer PC Command Line 1.0
PC>ping 192.160.1.8 with 32 bytes of data:
Reply from 192.160.1.8: bytes=32 time=0ms TTL=128

Ping statistics for 192.160.1.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>ping 192.160.1.8 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.160.1.8:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>192.160.1.2
Invalid Command.

PC>ping 192.160.1.2
Ping statistics for 192.160.1.2:
    Packets: Sent = 32, Received = 32, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>

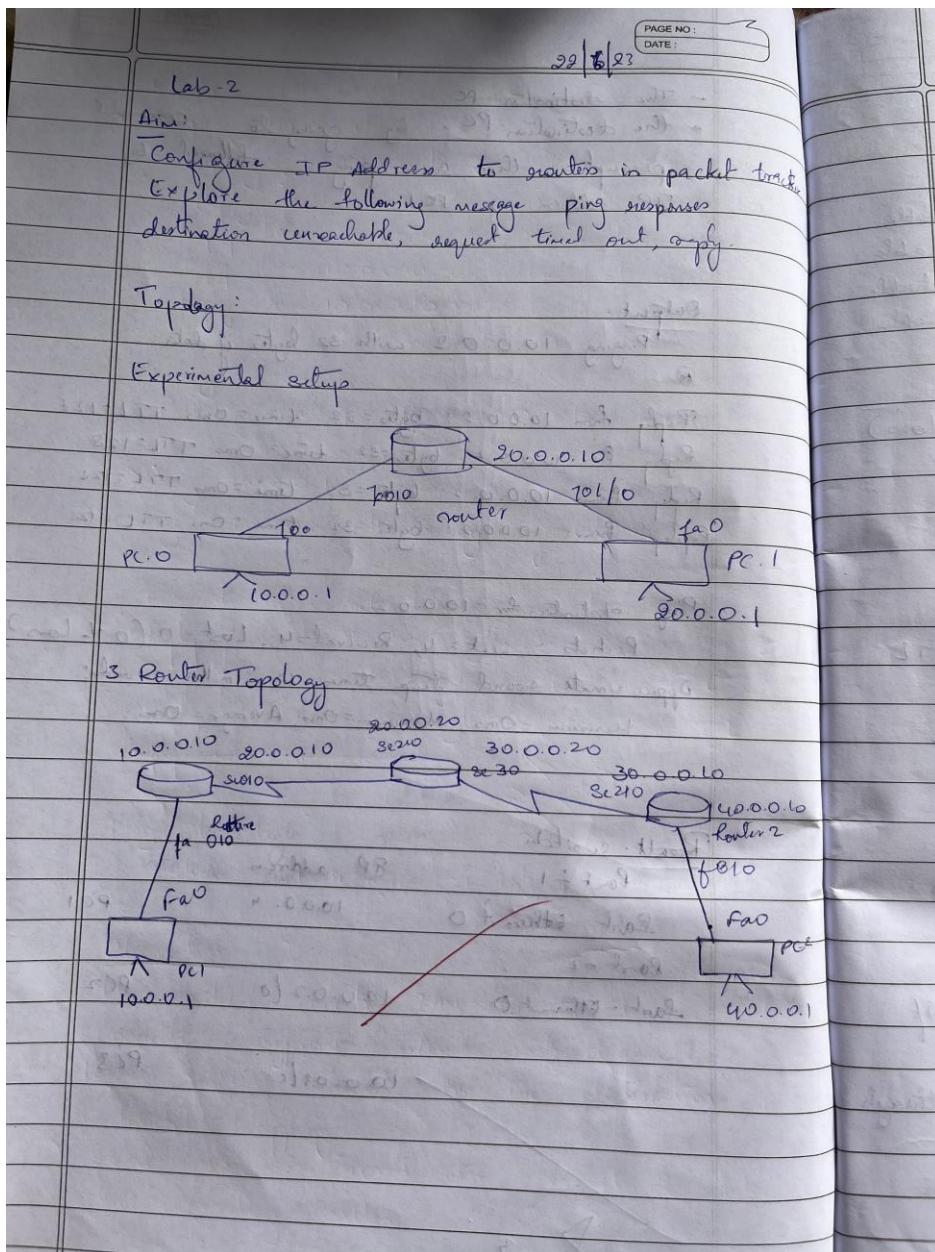
```



LAB 2

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

OBSERVATION:



Result:

(i) > ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Reply from 10.0.0.10: Destination host unreachable

Ping statistics:

Packets sent = 4: Received = 0, loss = 4 (100% loss)

> ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Request timed out

Reply from 40.0.0.10 bytes = 32 time = 2ms TTL

Reply from 40.0.0.10 bytes = 32 time = 2ms TTL

Reply from 40.0.0.10 bytes = 32 time = 2ms TTL

Reply from 40.0.0.10 bytes = 32 time = 2ms TTL

Ping statistics

Packets sent = 4: Received = 1, loss = 1 (25% loss)

1/1
13/12/2023

Observation:

The router connects LAN to the internet. It

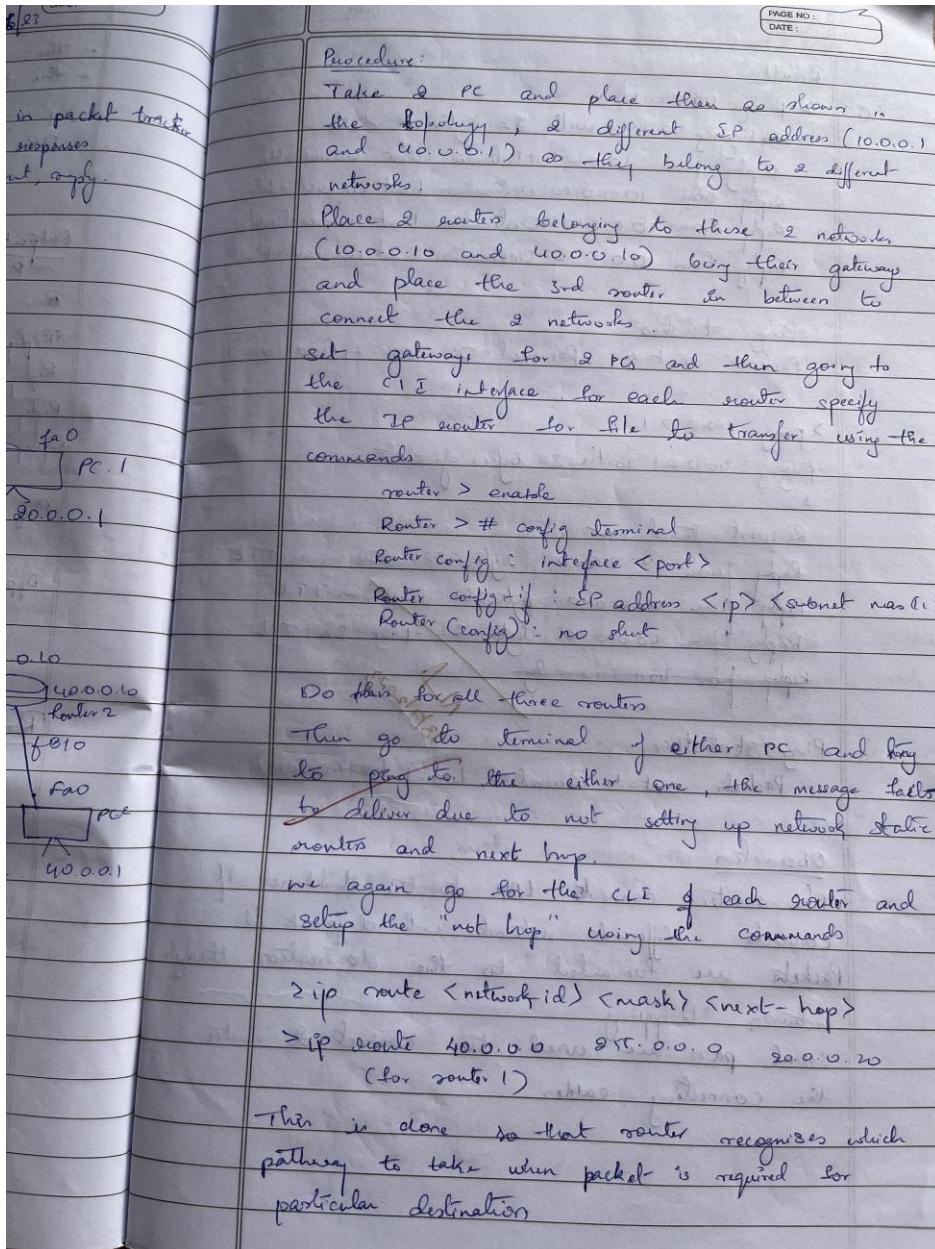
connects different networks with different IDs.

Packets are forwarded to the destination through

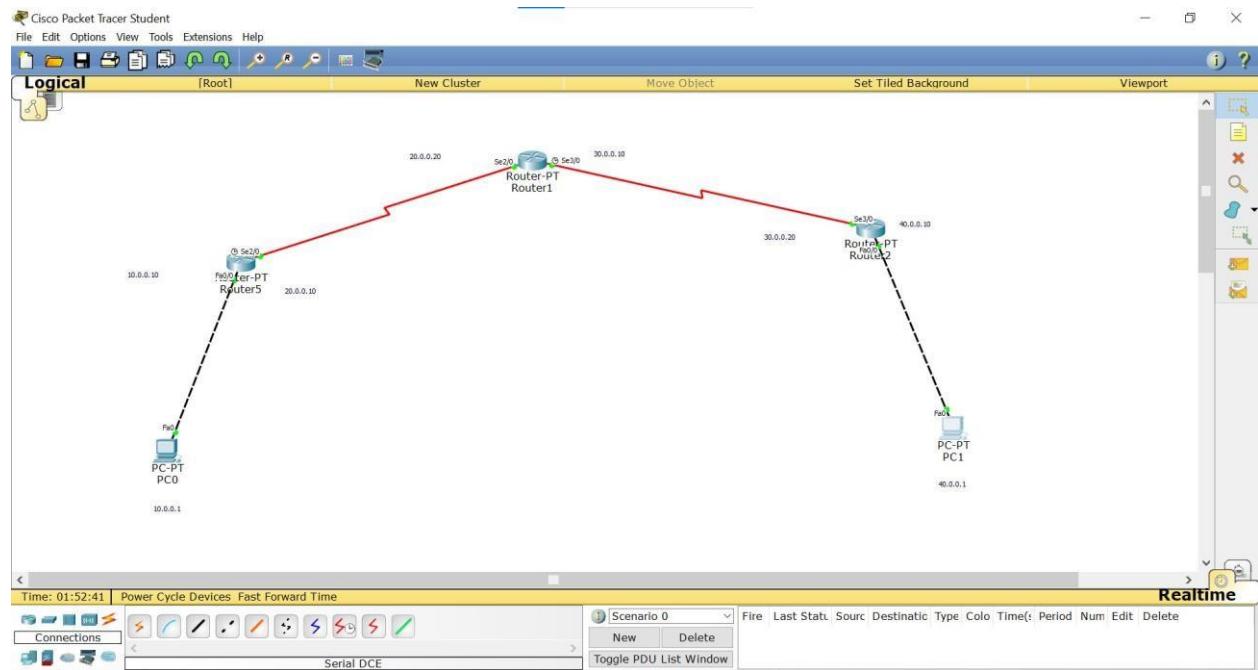
network hopping.

Serial ports are used to connect a router

the connecting cables.



TOPOLOGY:



OUTPUT:

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>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=10ms 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 = 10ms, Average = 3ms

PC>

```

Cisco Packet Tracer Student

File Edit Options View Tools Extensions Help

Logical [Root] New Cluster Move Object Set Tiled Background Viewport

Event List

Vis.	Time(sec)	Last Device	At Device	Type	Info
465.354	--	Router0	PC1	CDP	
525.353	--	Router0	CDP		
525.354	--	Router0	PC0	CDP	
525.354	--	Router0	PC1	CDP	
585.355	--	Router0	CDP		
585.356	--	Router0	PC0	CDP	
585.356	--	Router0	PC1	CDP	

Reset Simulation Constant Delay Captured till: 385.356 s

Play Controls Back Auto Capture / Play Capture / Forward

Event List Filters - Visible Events

Scenario 0

Fire Last Status Source Destination Type Color Time(sec) Periodic Num

Autocomp Window

Connections

Toggle PDU List Window

Time: 00:27:16.137 Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward

25°C Mostly sunny 5:10 AM ENG 6/22/2023

LAB 3

Configure default route, static route to the Router.

OBSERVATION:

PAGE NO: _____
DATE: 13/7/23

Lab 3

Aim: Configure default route, static route to PC2
Router.

Topology:

1 (loss)

2ms TTL

TTL

loss)

met. If

2 ids

tion through

router

Procedure:

- Connect 3 routers and 2 PCs using copper cross-over cable for PC to router and a serial DCE cable to connect router to router.
- Set the IP address of both PCs & respective gateway.
- For all 3 routers set the respective 2 IP address in CLI mode by using the commands.

Step 1: Enable

Step 2: Config T

Step 3: interface Fast Ethernet 0/0

Step 4: IP address 10.0.0.10 255.0.0.0

Step 5: no shutdown

Step 6: exit

Step 7: Interface Sc 2/0.

Step 8: IP address 20.0.0.10 255.0.0.0

Step 9: no shutdown

Step 10: Exit

Step 11: Exit

- Repeat the commands for other 2 routers with the respective IP address.
- for Router 1, set the IP address routes of other IP address, statically by using following steps

Step 1: Config T

Step 2: IP route 10.0.0.0 255.0.0.0 20.0.0.10

Step 3: IP route 40.0.0.0 255.0.0.0 30.0.0.20

Step 4: Exit

Step 5: Exit

Step 6: Show IP route

- for Router 0 & Router 2 we set default IP router which means it can access any IP address with any subnet mask.

- Set the default IP route by following these commands

Step 1: Config T

Step 2: IP route 0.0.0.0 0.0.0.0 20.0.0.20

Step 3: IP route 0.0.0.0 0.0.0.0 30.0.0.10

- Step 3 is given for Router 0 & step 3 command for Router 1

← Go to PO's command prompt & type ping message to send packet across

Ping output:

Packet-tracer PC command line 1.0

PC > ping 40.0.0.1

Pinging 40.0.0.1

Request timed out

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

Reply from 40.0.0.1 : bytes=32 time=16ms TTL=125

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

other
tips

Ping statistics for 40.0.0.1

packets: sent=4, received=3, lost=1 (25% loss)

Approximate round trip times in milliseconds:

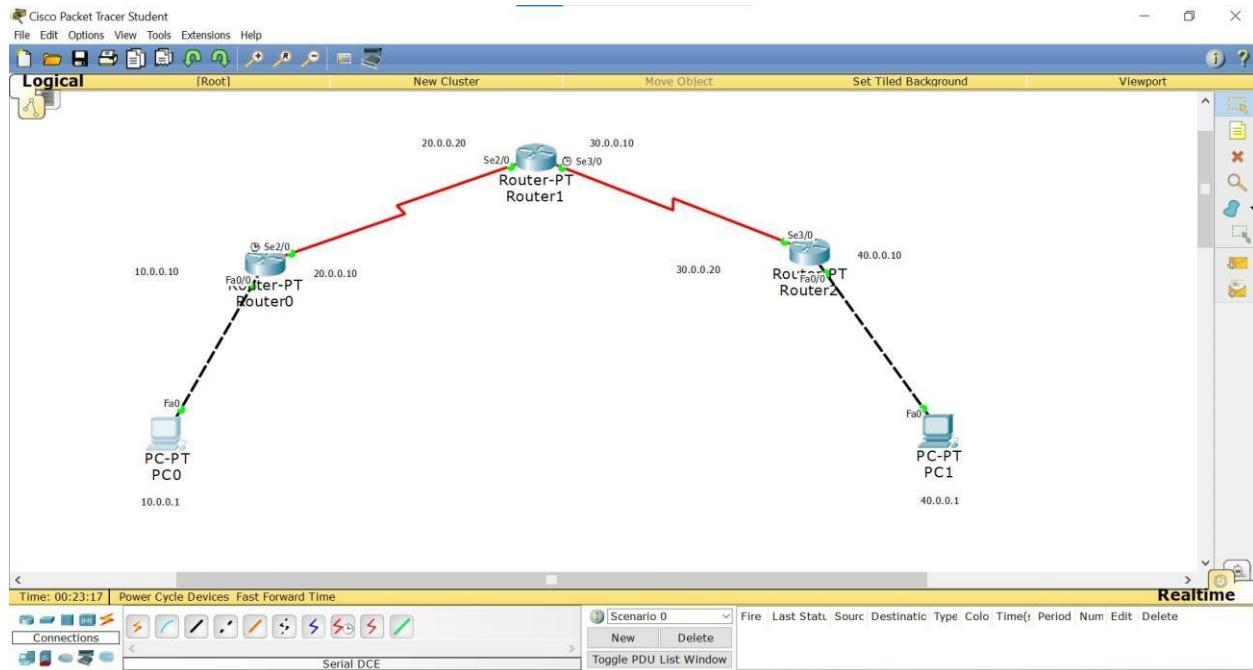
minimum=2ms, maximum=16ms, Average=6ms.

Observation:

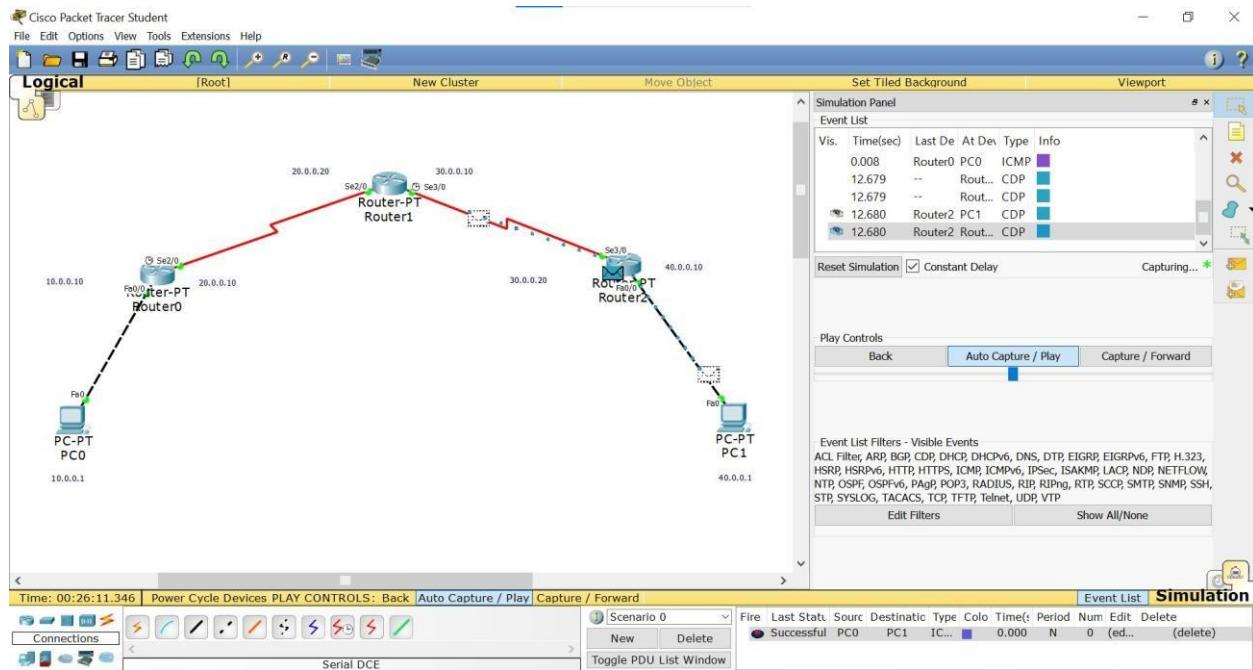
- A default route is the route which takes effect when no other route is available for an IP address destination.
- If a packet is received the device first checks the destination address if the IP destination address is not local the device checks its routing table if the remote destination subnet is not listed then the packet is forwarded to the next hop toward the destination using the default route.
- The process repeats until the packet is delivered.

PL
31/8/2023

TOPOLOGY:



OUTPUT:



PC0

Physical Config Desktop Custom Interface

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=16ms 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 = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 16ms, Average = 6ms

PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time=21ms TTL=125
Reply from 40.0.0.1: bytes=32 time=9ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms 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 = 21ms, Average = 9ms

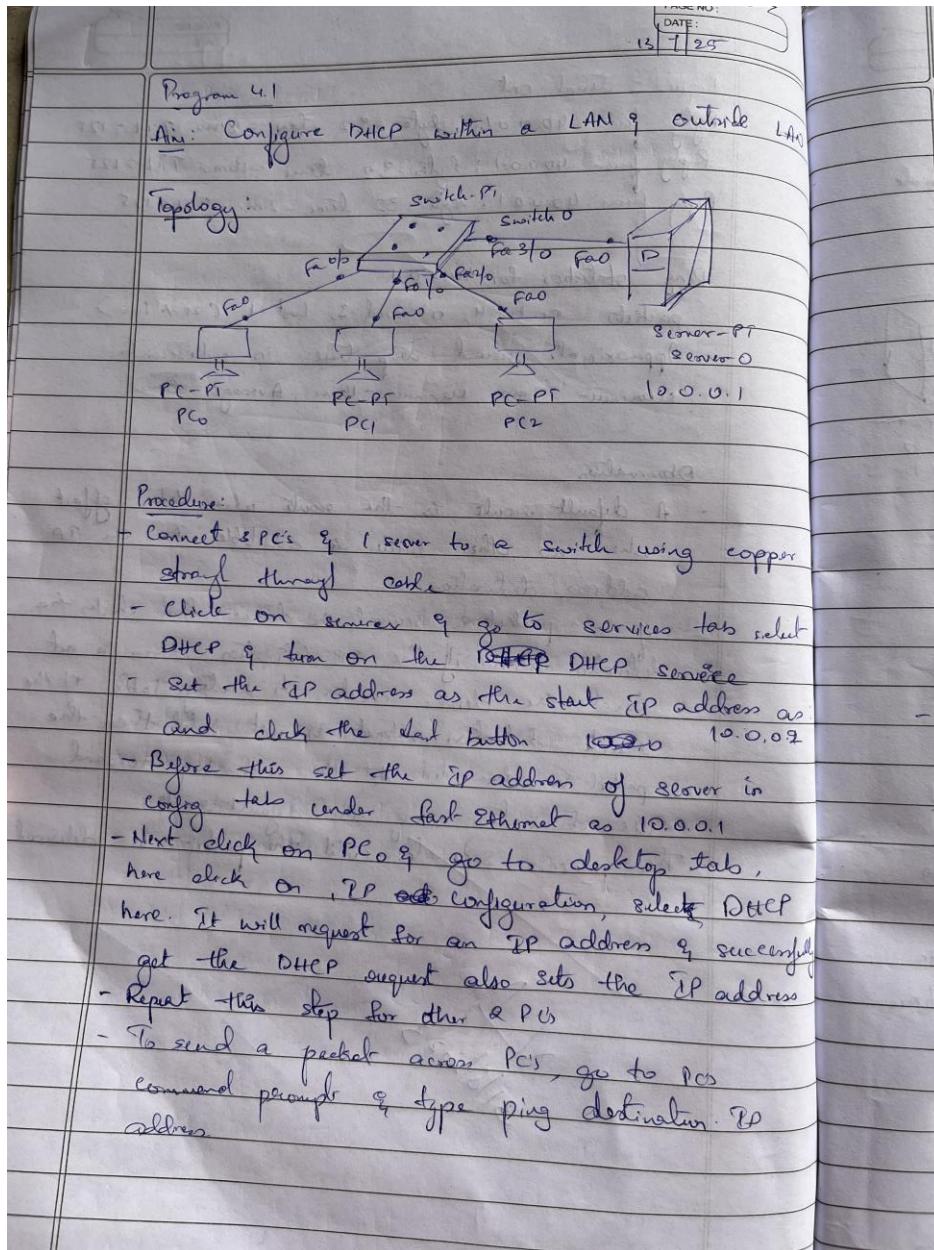
PC>
```

LAB 4

Configure DHCP within a LAN and outside LAN.

OBSERVATION:

TOPOLOGY:



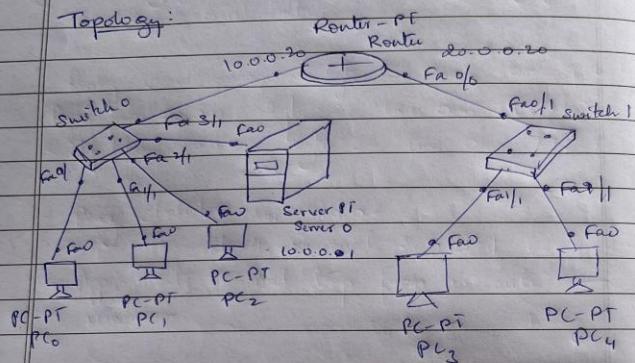
work	LAN	<p>ping output:</p> <p>Packet tracer PC command line 1.0</p> <p>PC > Ping 10.0.0.3</p> <p>Pinging 10.0.0.3 with 32 bytes of data.</p> <p>Reply from 10.0.0.3 bytes=32 time=0ms TTL=128</p> <p>Reply from 10.0.0.3 bytes=32 time=0ms TTL=128</p> <p>Reply from 10.0.0.3 bytes=32 time=1ms TTL=128</p> <p>Reply from 10.0.0.3 bytes=32 time=0ms TTL=128</p>
PT	-0	
	1	
oppor		<p>Ping statistics from 10.0.0.3</p> <p>Packets: sent=4, received=4, lost=0 (0% loss)</p> <p>Approximate round trip times in milliseconds:</p> <p>Minimum = 0ms Maximum = 1ms, Average = 0ms</p>
select		
as:	0.02	
in		
HCP		
carefully		
ress		
		<p><u>Observations:</u></p> <ul style="list-style-type: none"> - DHCP is used to dynamically assign an IP address to any device or node. - It is a client server protocol in which servers manage a pool of unique IP addresses & also keep about client configuration parameters. - DHCP-enabled clients sends a request to DHCP server when they want to connect to a network. - The DHCP server responds to the client request by providing IP configuration information from address pools, previously specified by a network administrator. <p style="text-align: center;">N 3/8</p>

Program 4.2

13-7-23

Aim: Configure DHCP within a LAN & outside
LAN

Topology:



Procedure:

- * Add a Router or switch or PCs to 4.1 program network & connect the router to both switches.
- * Set the server IP address of server and with the help of server set the first 3 PCs IP address through DHCP.
- * Now set the router IP address with the following commands statically.

Step 1: No

Step 2: Enable

Step 3: Config T

Step 4: Interface fast Ethernet 4/0

Step 5: IP address 10.0.0.20 255.0.0.0

Step 6: No Shot

Step 7: Exit

Step 8: In
Step 9: I
Step 10: H
Step 11: E
Step 12: I
Step 13: S

* Go to
* Again
Connect
Step 11
Step 15
Step 16
Step 17
Step 18
* Now
do
S1
* Now
to
IP
+ Now
pac
pin
ci

PAGE NO: _____
 DATE: _____

Step 8: Interface Fast Ethernet 0/0
 Step 9: IP address 20.0.0.20 255.0.0.0
 Step 10: No shut
 Step 11: Exit
 Step 12: Exit
 Step 13: Show IP address route.

9 outside

11 switch 1

Fast 11

FAD

PC 4

to 4.1

to

and

11

* Go to server & set the gateway as 10.0.0.20
 * Again go to scalar CLI & follow these commands
 Step 14: Config T
 Step 15: interface Fast Ethernet 0/0
 Step 16: IP helper-address 10.0.0.1
 Step 17: No shut
 Step 18: Exit
 * Now go to server services & add one more ^{pool} as server pool 1, start IP address as 20.0.0.2 & default gateway as 20.0.0.20. Then click add
 * Now set the other 2 PCs IP address by going to Desktop → IP config & select DHCP which will automatically generate its IP address.
 * Now the network is complete & can send packets from any PC to other by typing ping destination IP address in their respective command prompt.

A/D
 31/8/2023

Ping output:

Packet tracer PC command line 1.0

PC > Ping 200.0.2

pinging 200.0.2 with 32 bytes of data
Request timed out

Reply from 200.0.2 bytes=32 time=0ms TTL=128

Reply from 200.0.2 bytes=32 time=0ms TTL=128

Reply from 200.0.2 bytes=32 time=0ms TTL=128

Ping statistics for 200.0.2

Packets sent = 4, Received = 3 Lost = 1 (25.0% loss)

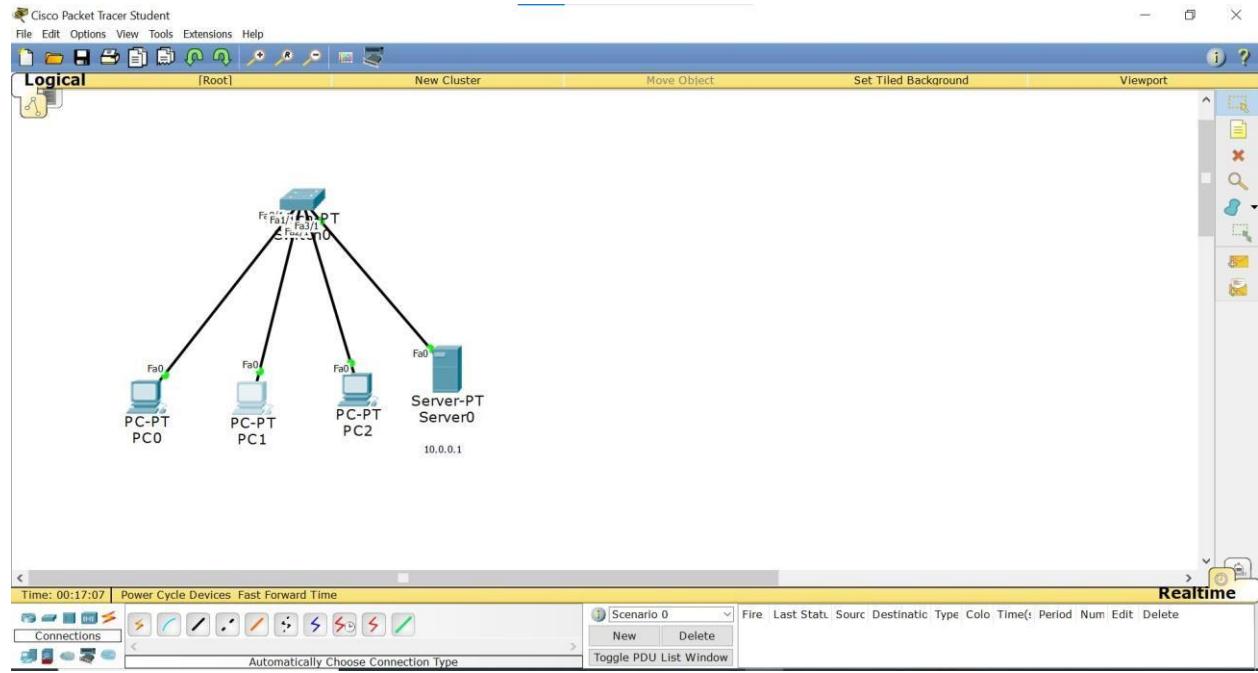
Approximate round trip times in milliseconds

minimum = 0ms maximum = 0ms Average = 0ms

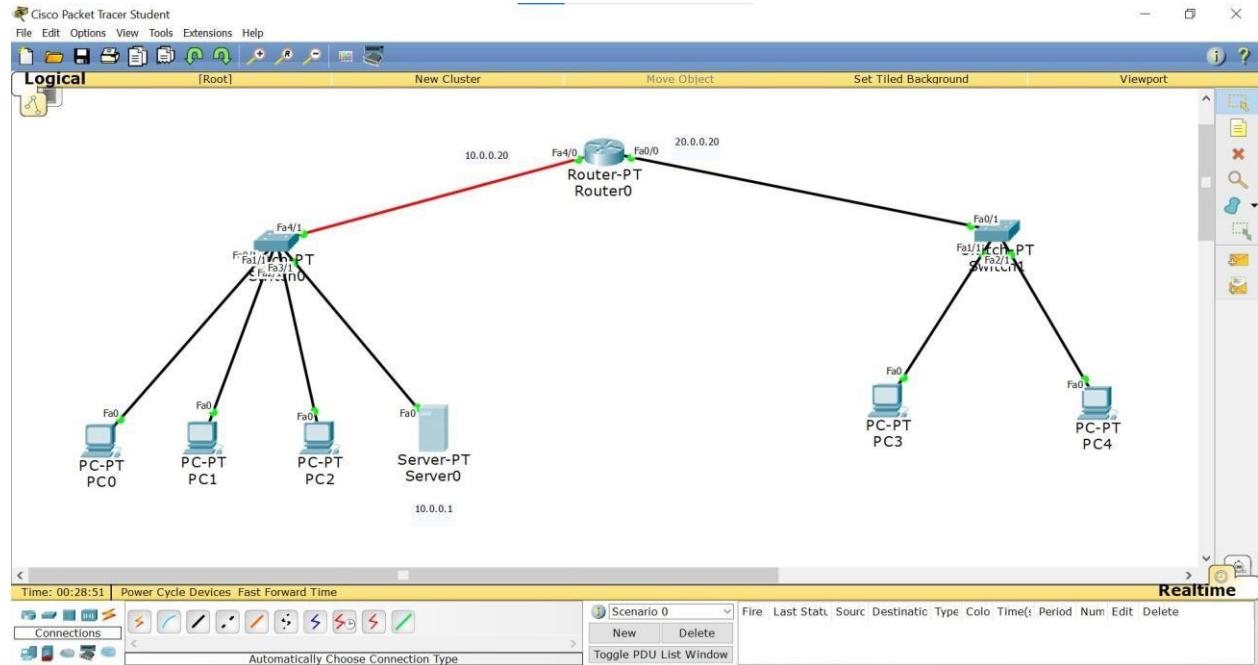
Observations:

- * DHCP is used to assign IP address dynamically to different devices.
- * To assign continuous IP address we create a server pool where we assign the starting IP address & a default gateway number.
- * For PCs under different switches we create a different server pool again & start.
- This task is of delivering the packets to correct destination IP address & also send back the packet to original device.

PROGRAM 4.1:

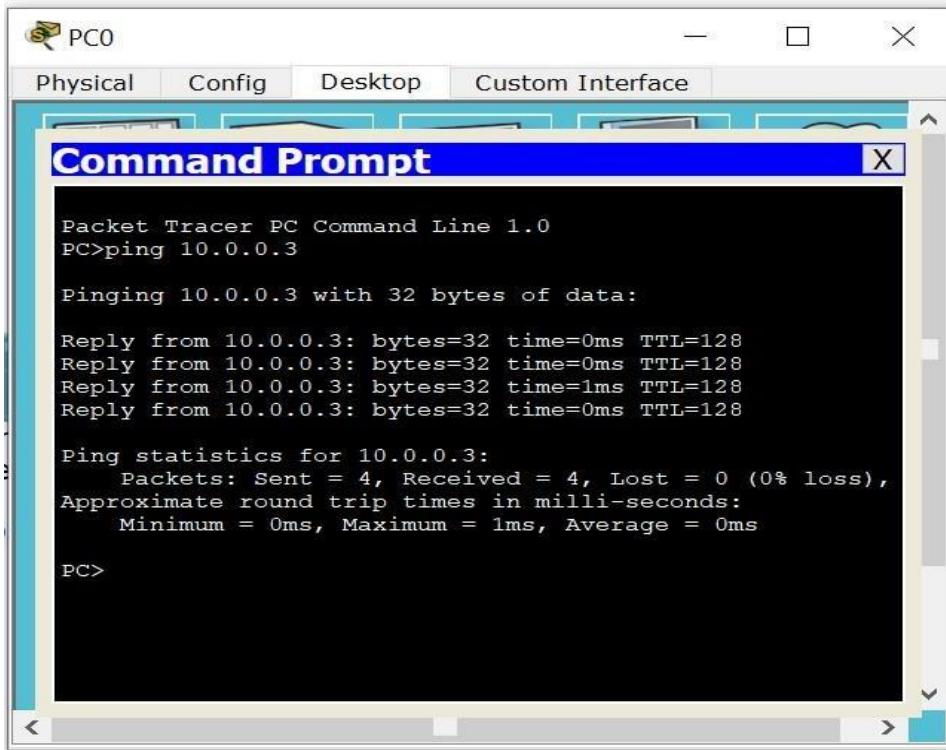


PROGRAM 4.2:

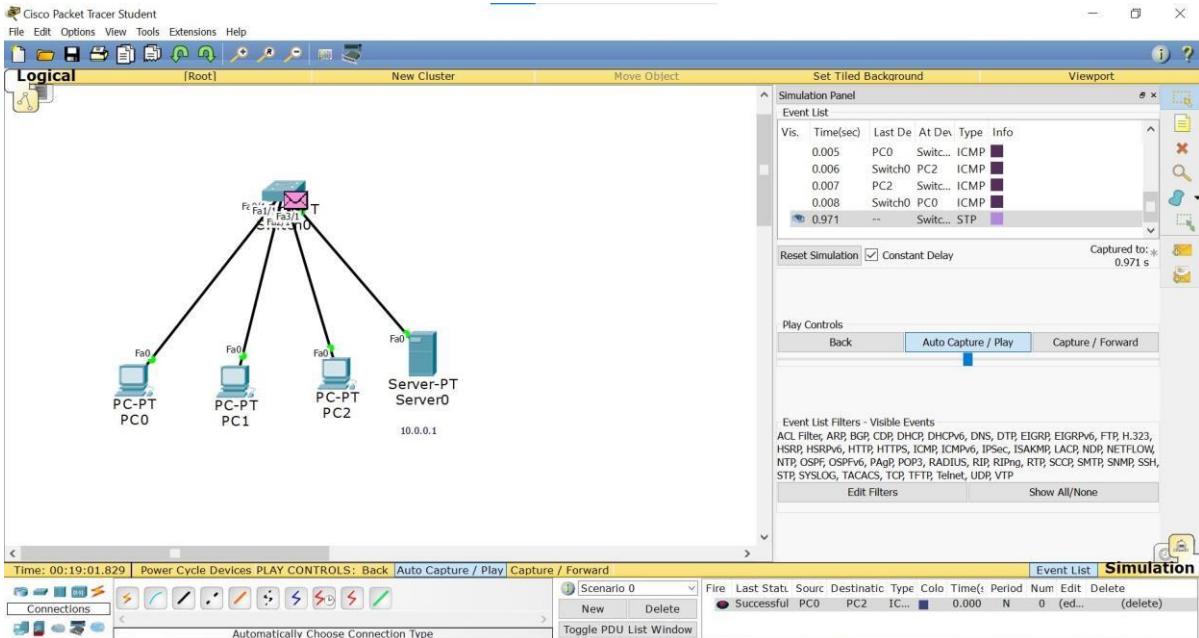


OUTPUT:

PROGRAM 4.1:



Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=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 = 1ms, Average = 0ms
PC>



PROGRAM 4.2:

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127

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

PC>ping 20.0.0.3

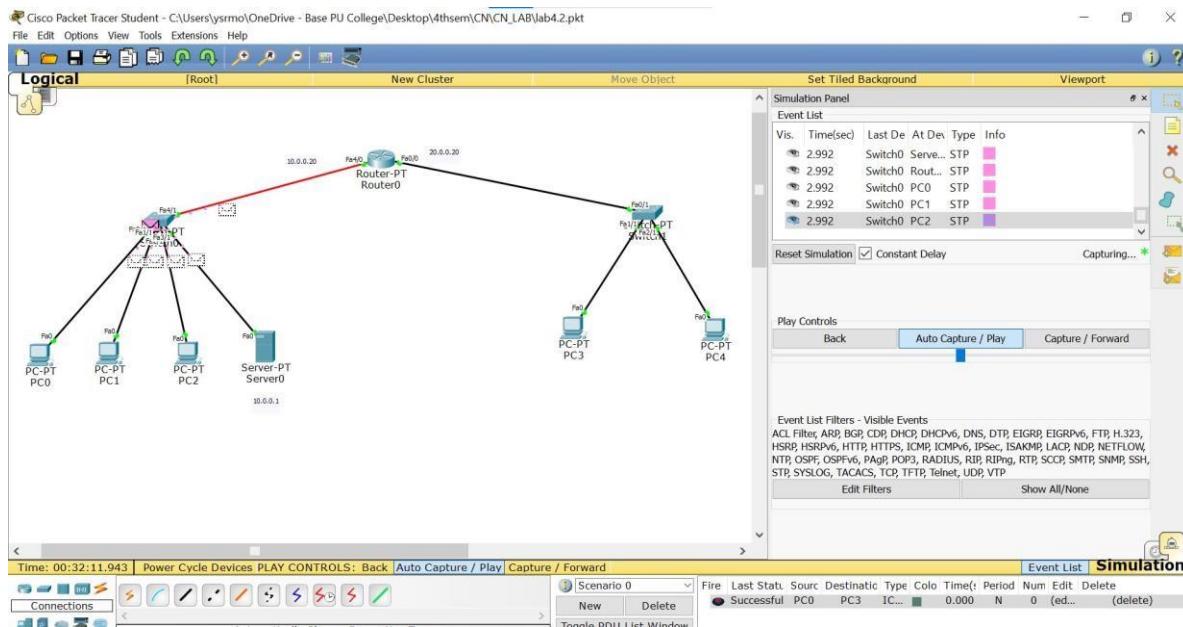
Pinging 20.0.0.3 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>

```



WEEK5

Configure Web Server, DNS within a LAN.

OBSERVATION:

Lab 5.(i)

Aim: Configure DNS server with LAN

Topology:

Procedure:

- 1) Create a topology as shown above
- 2) Configure the PC with IP address & gateway and subnet mask.
- 3) Add DNS at 10.0.0.2 to the PCo & the IP address subnet mask as 10.0.0.1 and gateway 10.0.0.2
- 4) Click on cluster & click on services → HTTP & HTTP2. click 'ON' then select the PMS click on 'ON' and add Name: test.com Address: 10.0.0.2
- 5) Now go to config → fast Ethernet - IP address 10.0.0.2 subnet mask 255.0.0.0
- 6) Now go to settings on desktop → web browser & type the 'test.com' in URL & click go.
- 7) Now go to settings in the config gateway 10.0.0.2 DNS server 10.0.0.2
- 8) Now go to PC click on desktop → web browser & type the 'test.com' in URL & click on go
- 9) Now will see the 'index.html' that has written in the (HTTP) button.

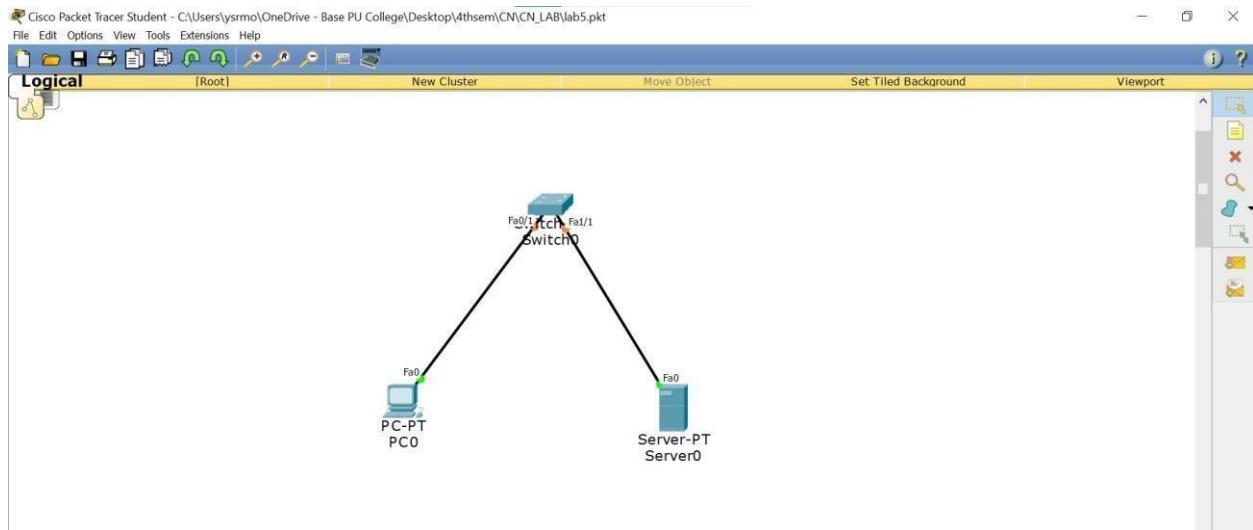
Observation:

DNS is the Domain Name system. DNS is linked to the Internet & focused on a system using Internet Protocol (IP). DNS servers are required for the working of DNS. The IP address is calculated with the aid of a lookup table.

→ PC communicates with the server using the IP address as well as using the domain.

N
21/8/27

TOPOLOGY:



LAB 6

Configure RIP routing Protocol in Routers.

OBSERVATION:

Lab - 5 (ii)

RIP - Routing information Protocol:

Aim: Configuring RIP - Routing Protocol in router.

Topology:

Procedure:

- 1) Three routers & 2 PC's are connected as shown in topology.
- 2) Configure the PC's with proper IP address & gateway address.
- 3) Similarly, configure the routers with the proper IP address in CLI mode.

- N. enable
- config T
- Interface fast Ethernet 0/0
- IP address 10.0.0.1 255.0.0.0
- Encapsulation PPP
- clock rate 64000.
no shut.

Note: The encapsulation PPP should be given to all the routers & clock rate 64000.
Command should be only given to the clock symbol sides of the router (i.e. open sides)

→ for making the router to know about the other devices in the previous 2 experiments we used p. state & the other with dynamic address, but here we use a routing protocol algorithm that itself makes the router to know other devices.

→ router ip

→ network 20.0.0.0 } router 2

→ network 30.0.0.0

→ router ip

→ network 30.0.0.0 } router 3

→ network 40.0.0.0

→ router ip

→ network 10.0.0.0 } router 1

→ network 20.0.0.0

Ping output:

PC > ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Reply from 40.0.0.1: bytes=32 time: 0ms TTL: 128

Ping statistics from 40.0.0.1

Packets sent = 4 Received = 4 Lost = 0 (0% loss)

Approximate round trip times in ms

minimum = 0ms, maximum = 0ms, Average = 0ms

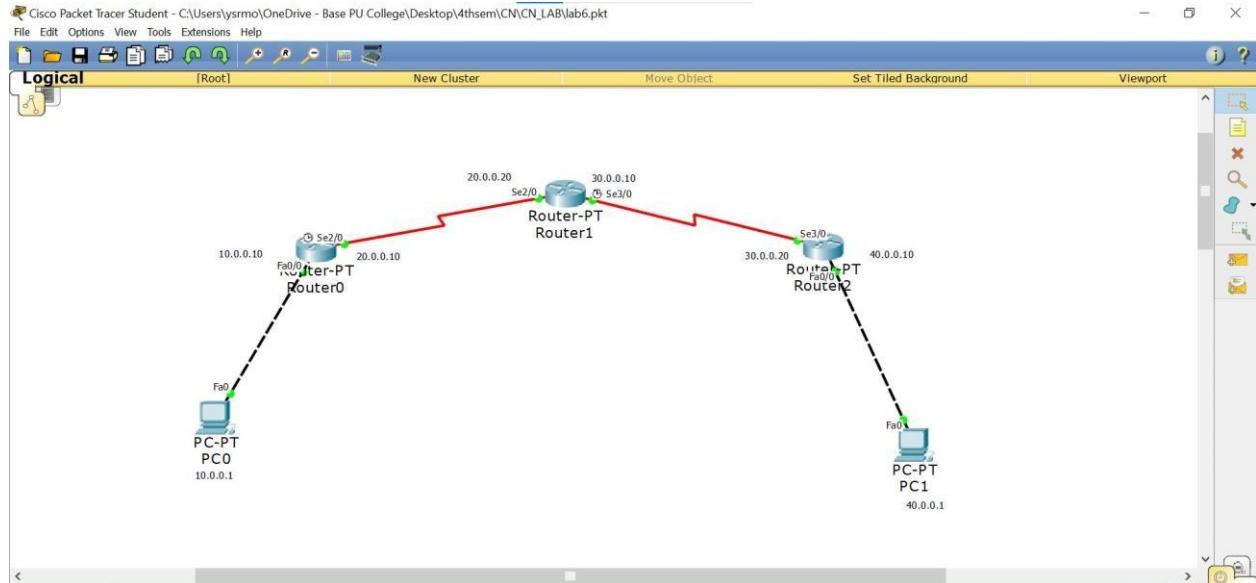
Observation:

RIP is the Routing Information Protocol as a distance vector protocol that uses hop count as its primary metric. RIP defines how routers should share information when moving traffic among an interconnected group of local area network.

→ The RIP protocol here, used to connect the routers to one other & PCs using RIP procedure protocol & message is pinged successfully.

ND
3/8/23

TOPOLOGY:



OUTPUT:

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

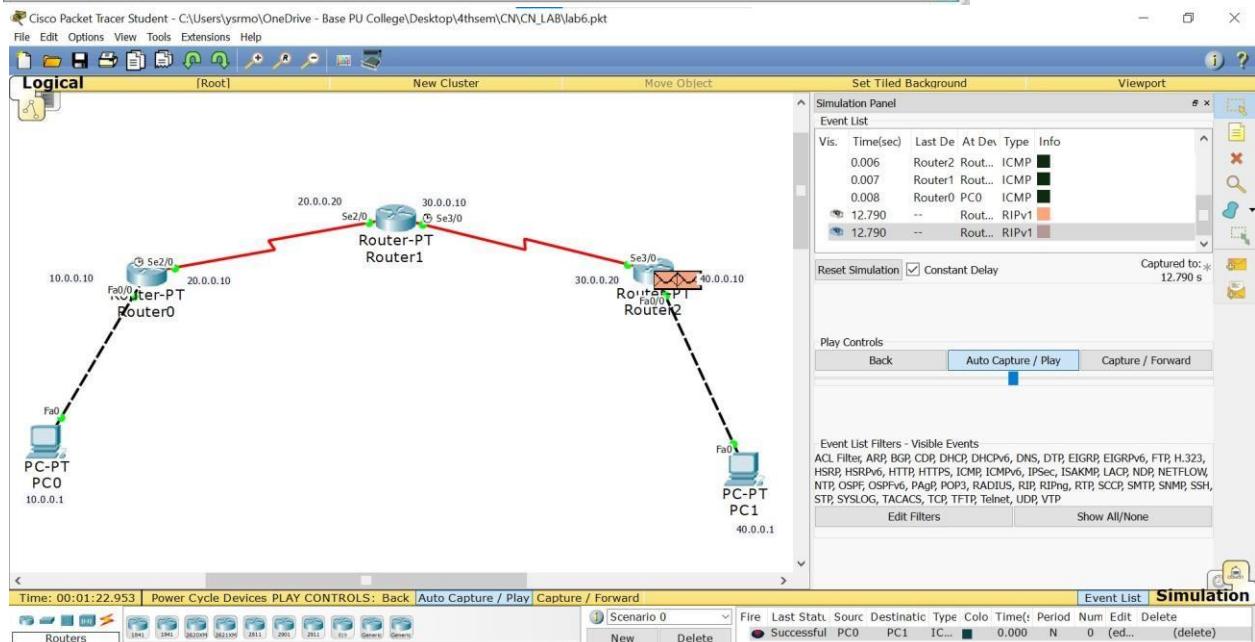
Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.1: bytes=32 time=8ms TTL=125
Reply from 40.0.0.1: bytes=32 time=5ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 5ms, Maximum = 10ms, Average = 7ms

PC>

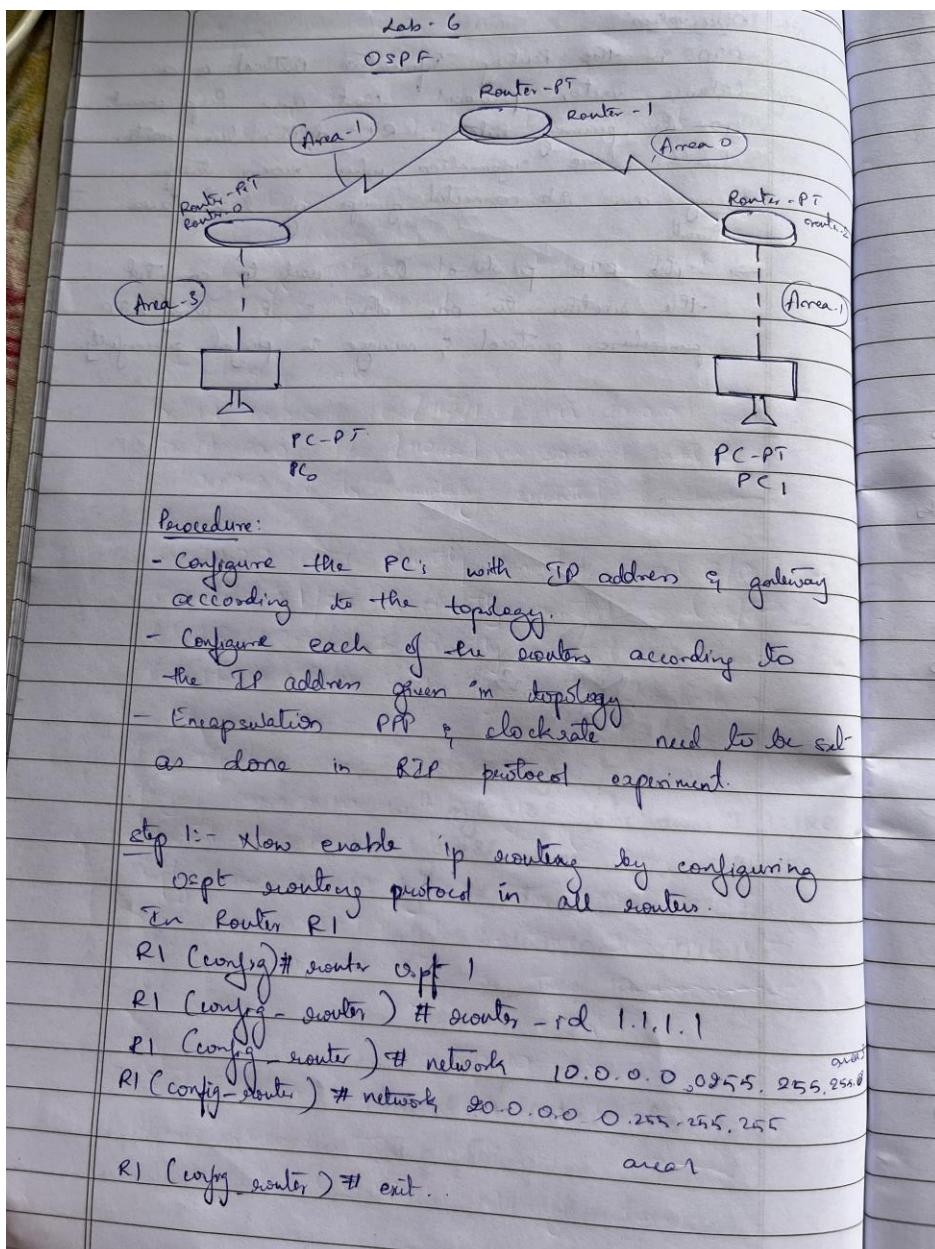
```



LAB 7

Configure OSPF routing protocol.

OBSERVATION:



In router R2
 R2 (config) # router ospf 1
 R2 (config-router) # network 10.0.0.0 0.255.255.255 area 0
 R2 (config-router) # network 30.0.0.0 0.255.255.255 area 0
 R2 (config-router) # exit

In Router R3
 R3 (config) # router ospf 1
 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

Step 4: Loopback in serial interface
 In router R1
 R1 (config-if) # interface loopback 0
 R1 (config-if) # ip address 172.16.1.252 255.255.255.0
 R1 (config-if) # no shutdown

In router R2 in serial interface:
 R2 (config-if) # interface loopback 0
 R2 (config-if) # ip address 172.16.1.253 255.255.255.0
 R2 (config-if) # no shutdown

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

N
 3/8/2023

Step 3: Virtual backbone link
In router R1

R1 (config) # router ospf 1
R1 (config-router) # area 1 virtual link 192.2.2.2

In router R2

R2 (config) # router ospf 1
R2 (config-router) # area 1 virtual link 192.1.1.1
R2 (config-router) # exit

→ show ip route

0. IA 10.0.0.1/8 [110/129] via 30.0.0.1 serial 3/0
0. IA 30.0.0.0/8 [110/128] via 30.0.0.1 serial 3/0
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks

- C. 30.0.0.0/8 is directly connected, serial 3/0
- C. 30.0.0.1/32 is directly connected serial 3/0
- C. 40.0.0.0/8 is directly connected fast ethernet 0/0
- C. 192.16.0.0/16 is directly connected, loopback 0.

Ping output:

pinging 40.0.0.10 with 32 bytes of data
Request timed out

Reply from 40.0.0.10 bytes=32 time=2ms TTL=125

Reply from 40.0.0.10 bytes=32 time=9ms TTL=125

Reply from 40.0.0.10 bytes=32 time=10ms TTL=125

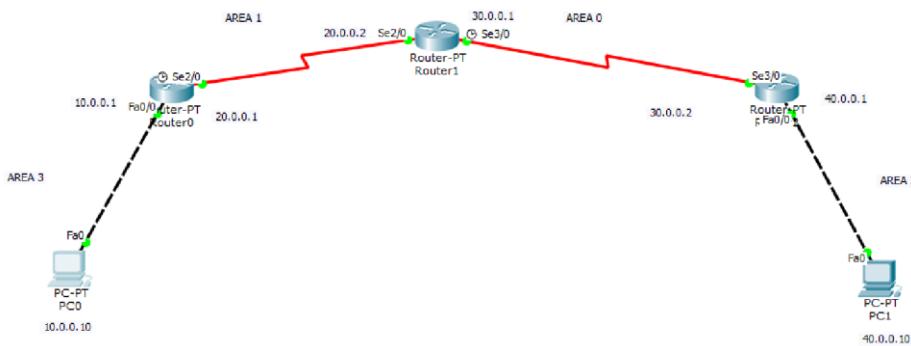
Pinging statistics for 40.0.0.10

packets: sent=4, received=3, lost=1 (25% loss)

Approx round trip in ms:

min=2ms, Max=10ms, Average=7ms

TOPOLOGY:



OUTPUT:

PC0

Physical Config Desktop Custom Interface

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

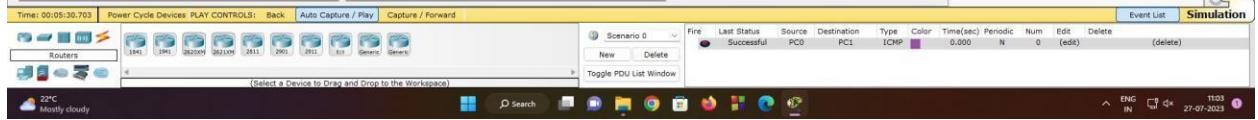
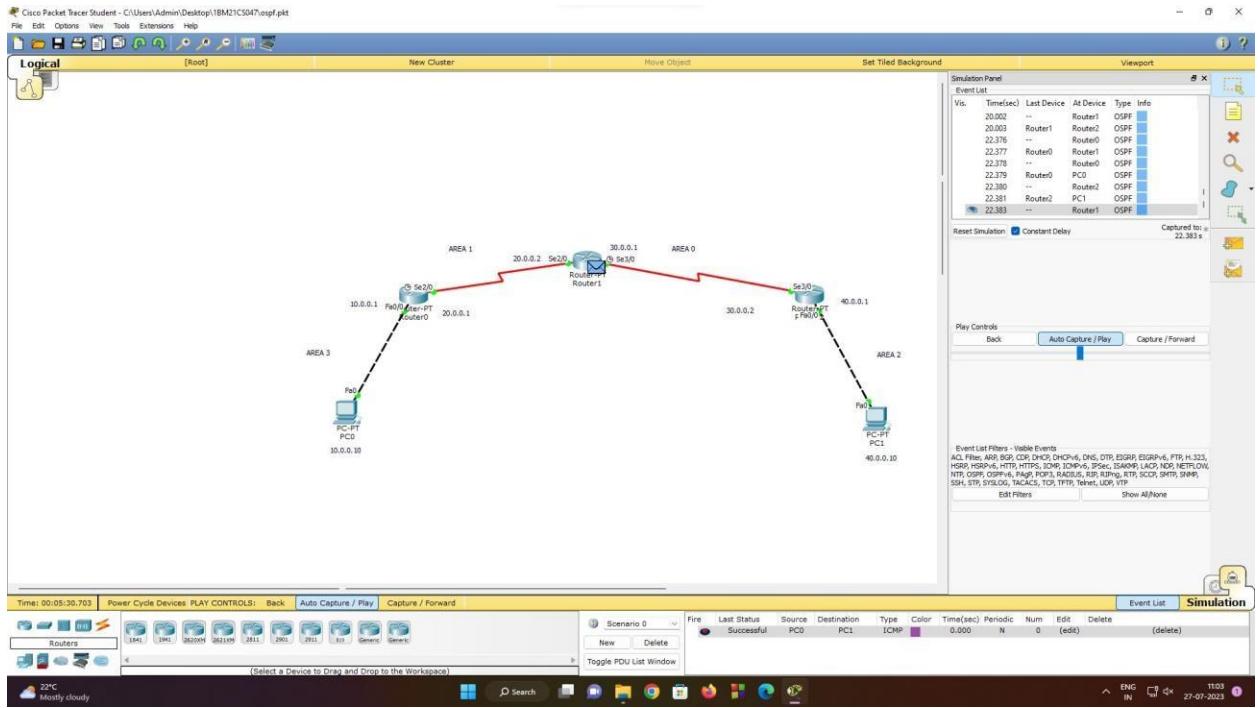
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=4ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=12ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 4ms, Maximum = 12ms, Average = 7ms

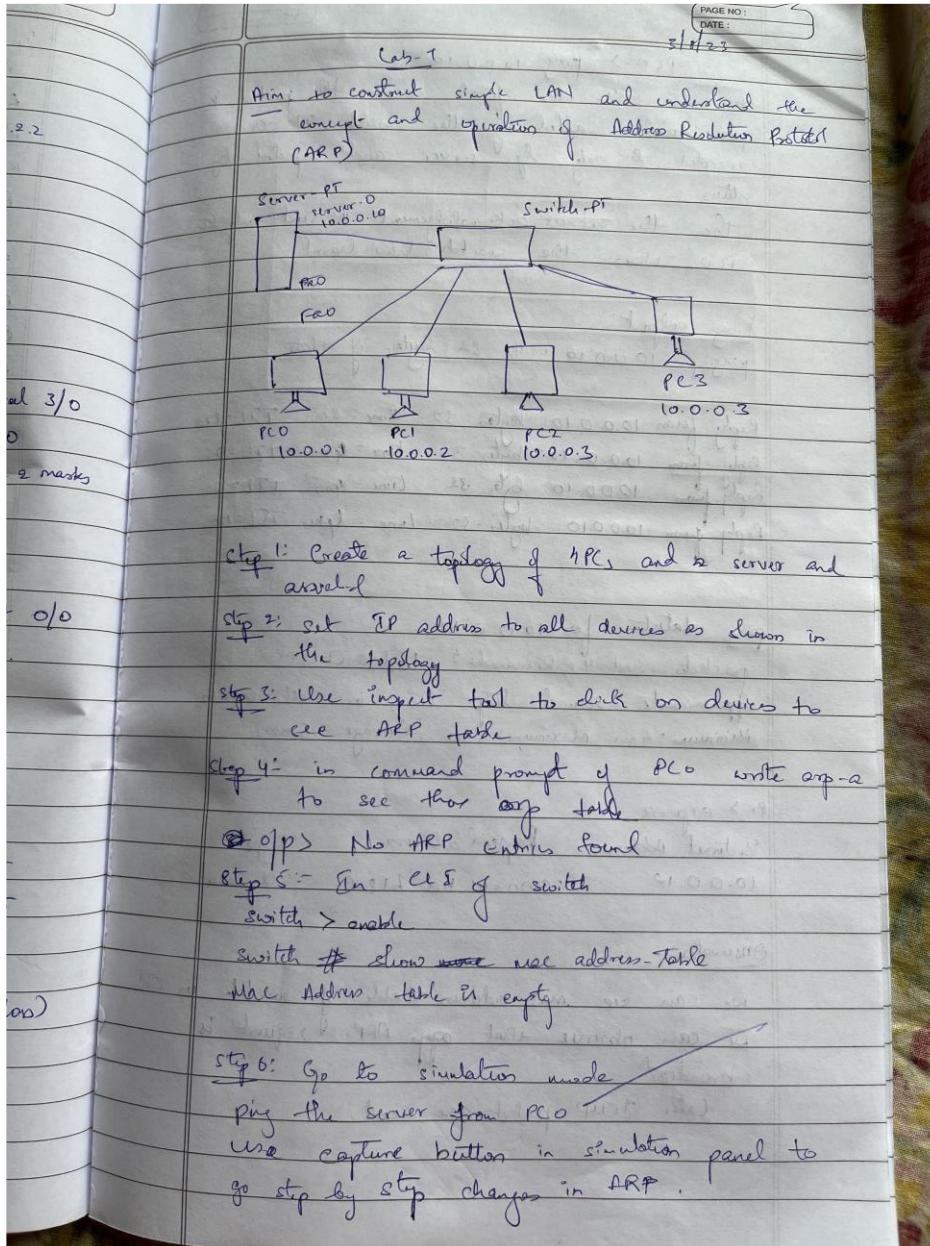
PC>
```



LAB 8

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

OBSERVATION:



PC0 > ping 10.0.0.10

Observation: In the first turn the packet is
also broadcasted by the switch. It is
accepted only by server and rejected by
others.

Then the network acknowledgement is sent only to
PC0. Now, the switch has learnt

Ping output:

pinging 10.0.0.10 with 32 bytes of data,

Reply from 10.0.0.10: bytes = 32 time = 8ms TTL = 128

Reply from 10.0.0.10: bytes = 32 time = 8ms TTL = 128

Reply from 10.0.0.10: bytes = 32 time = 8ms TTL = 128

Reply from 10.0.0.10: bytes = 32 time = 8ms TTL = 128

ping statistics for 10.0.0.10

packets sent = 4, Received = 4, lost = 0, %

Approximate round trip times in milli seconds

Minimum = 4 ms, Maximum = 8 ms, Average = 5 ms

pc > arp -a

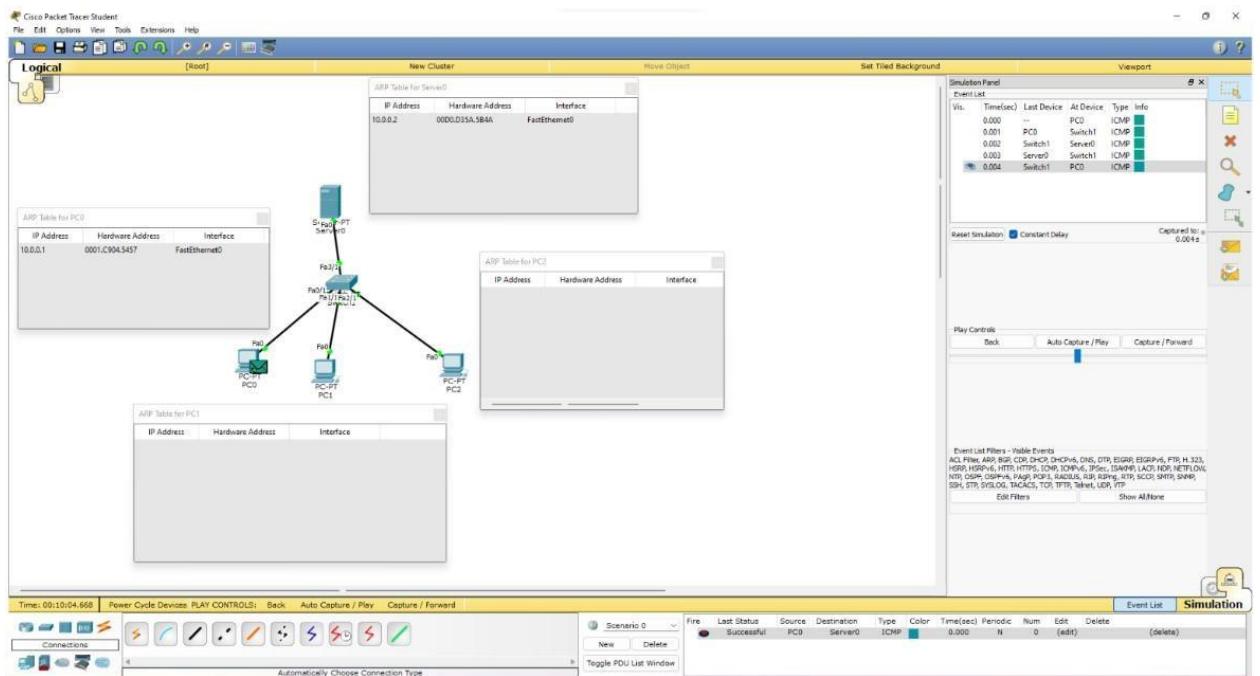
Internet Address	Physical Address	Type
10.0.0.10	00:0c:11:b1:8d	DYNAMIC

Observation:

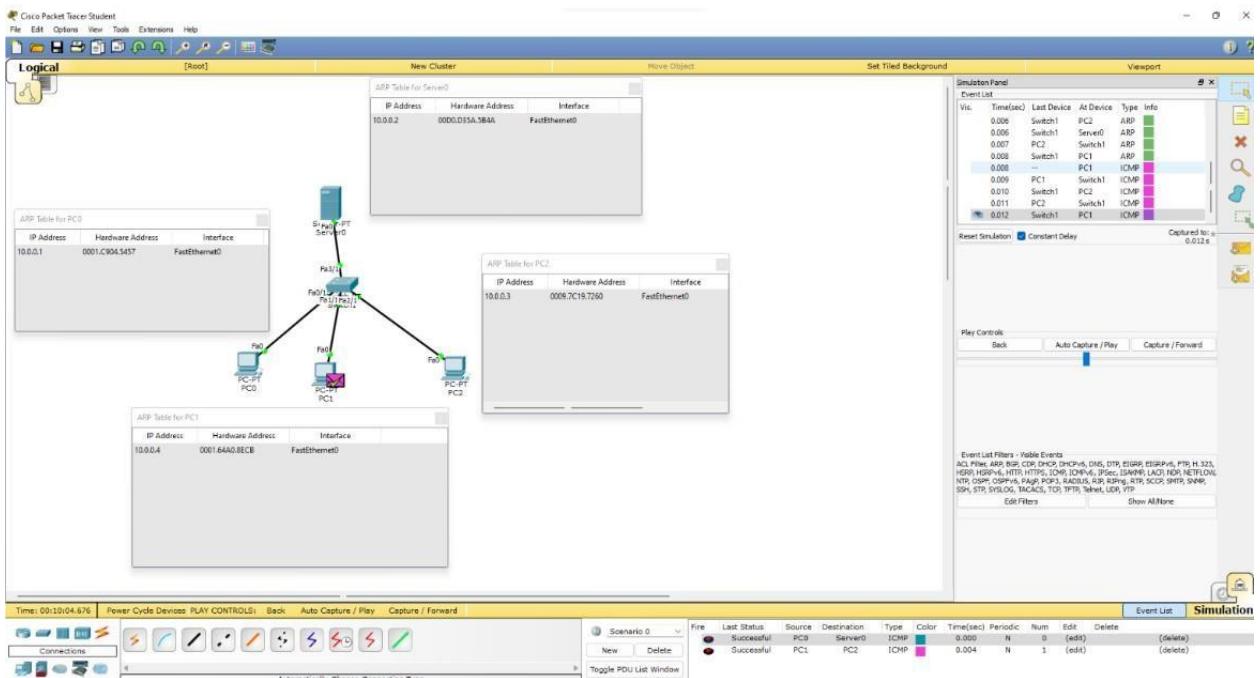
We can see mac address table of switch in CII
we can observe that ARP request is
broadcast

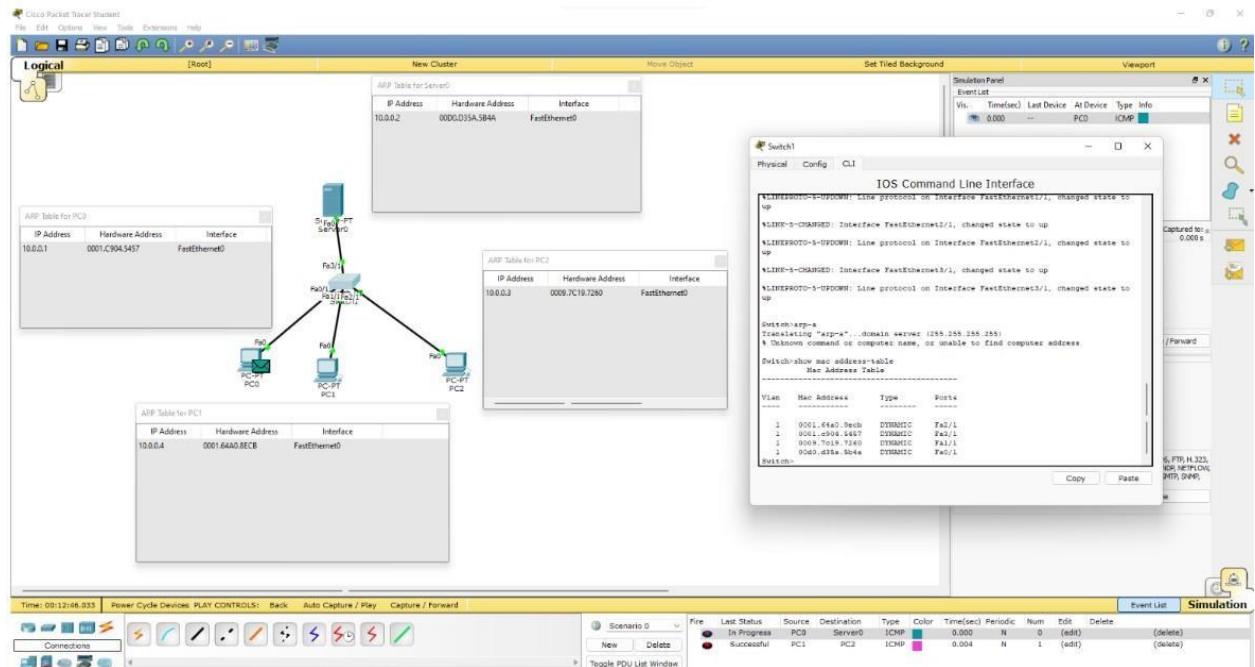
Later TCP packets are sent,

TOPOLOGY:



OUTPUT:





LAB 9

To construct a VLAN and make a pc communicate among VLAN.

OBSERVATION:

Lab 9
Aim: To construct a VLAN & make PCs communicate among a VLAN
PAGE NO: _____
DATE: _____

Topology:

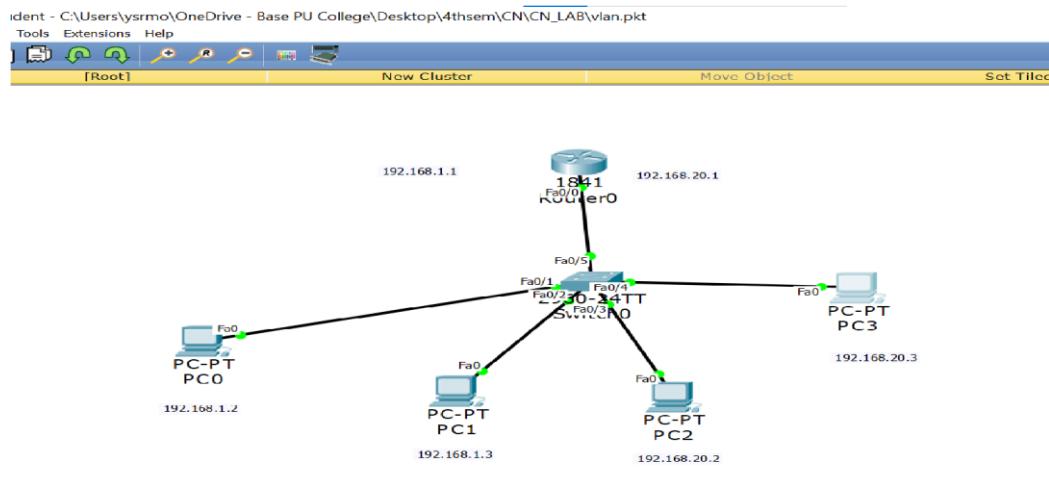
command center.

Procedure:

1. Create above topology with 2 PCs in one set & other set with IP address 192.168.1.2, 192.168.1.3 if one set and 192.168.20.2, 192.168.20.3 for other set.
2. Connect all 4 PCs to switch & now connect this router to switch.
3. Now create a VLAN database in switch with a VLAN number and VLAN name.
4. Now go to routers CLI mode.
5. Select the interface fast-ethernet 0/5 which is towards or between switch & router and make VLAN Trunk.
6. Go to the PCs which are set of VLAN & set a ethernet to new VLAN.
7. Go to CLI of router change it to config mode
Router (config) # interface fast-ethernet 0/0.1
Router (config-sub) # encapsulation dot1q 2

		DATE : _____
	Router (config-subif) # ip address 192.168.20.1 255.255.255.0	
	Router (config-subif) # no shut	
8.	This will create a sub interface 0/0 with setting 192.168.20.1 as its gateway.	
9.	Now ping a PC from one network to another	
	<u>Output:</u>	
	PC > ping 192.168.20.3	
	pinging 192.168.20.3 with 32 bytes of data:	
	Reply from 192.168.20.3: bytes=32 time=0ms TTL=122	
	Reply from 192.168.20.3: bytes=32 time=1ms TTL=122	
	Reply from 192.168.20.3: bytes=32 time=4ms 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=4, Lost=0 (0% loss),	
	Approximate round trip in milliseconds:	
	Minimum=0ms, Maximum=4ms, Average=1ms.	
	<u>Observation:</u>	
	From a network which is connected to one interface can be further divided into sub networks called virtual LAN. When created a new sub interface devices connected to it act like a different network, now after setting up, we can communicate between two networks.	

TOPOLOGY:



OUTPUT:

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.3

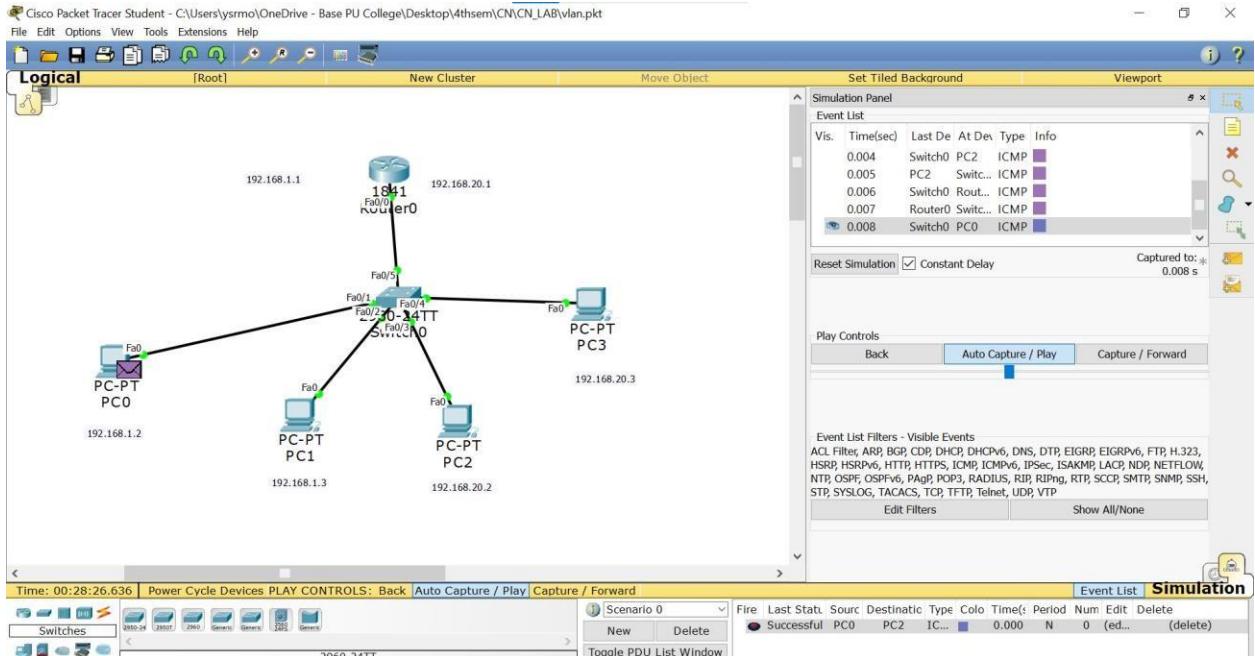
Pinging 192.168.20.3 with 32 bytes of data:

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 milli-seconds:
        Minimum = 0ms, Maximum = 5ms, Average = 1ms

PC>

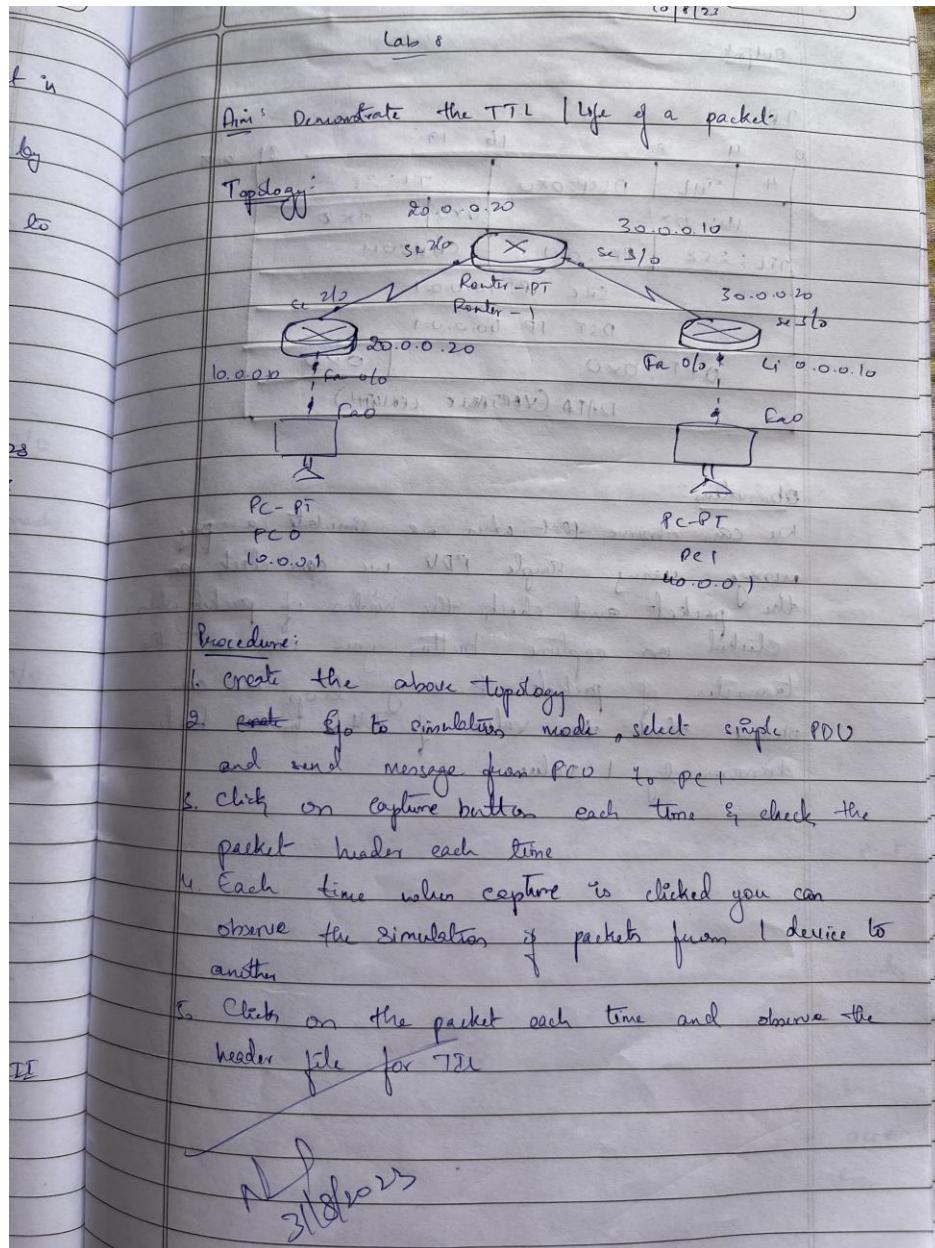
```



LAB 10

Demonstrate the TTL/ Life of a Packet.

OBSERVATION:



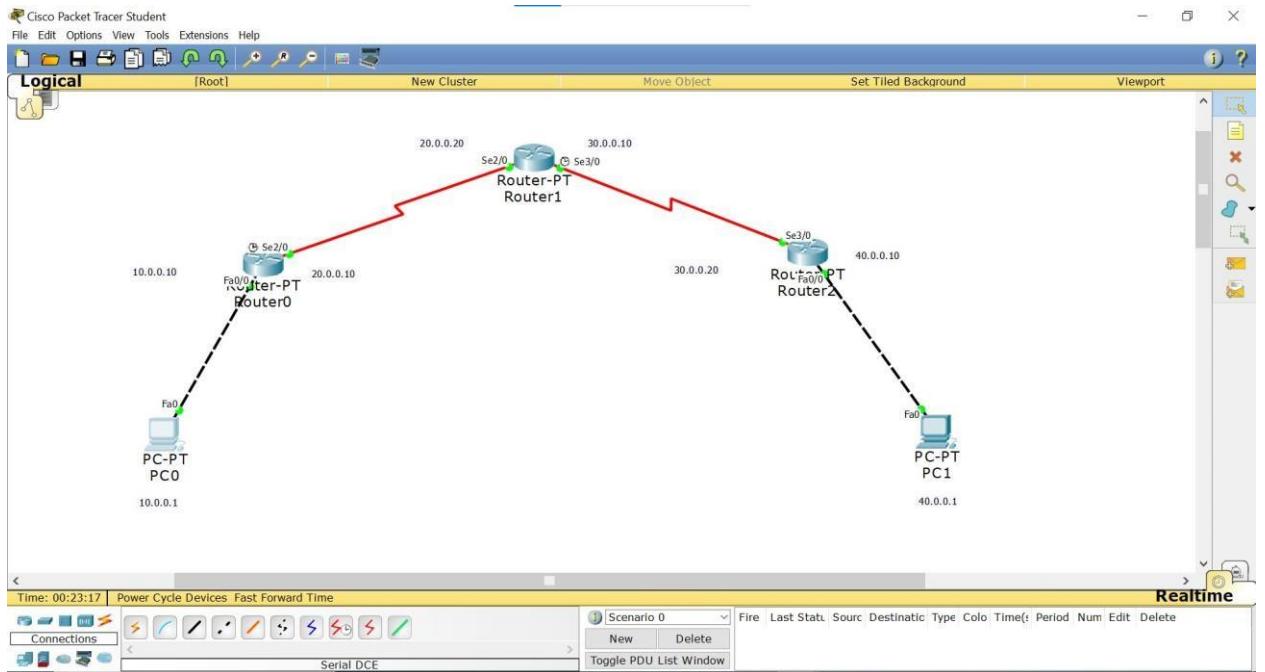
Output:

IP										31 bits	
0	4	8	16	19							
4	2HL	DSRP: 0x0	TTL: 28								
1D: 0x4			0x0	0x0							
TTL: 252	PRO: 0x1		CHECKSUM								
	SRC IP: 10.0.0.1										
	DST IP: 40.0.0.1										
OPT: 0x0			0x0								
DATA (VARIABLE LENGTH)											

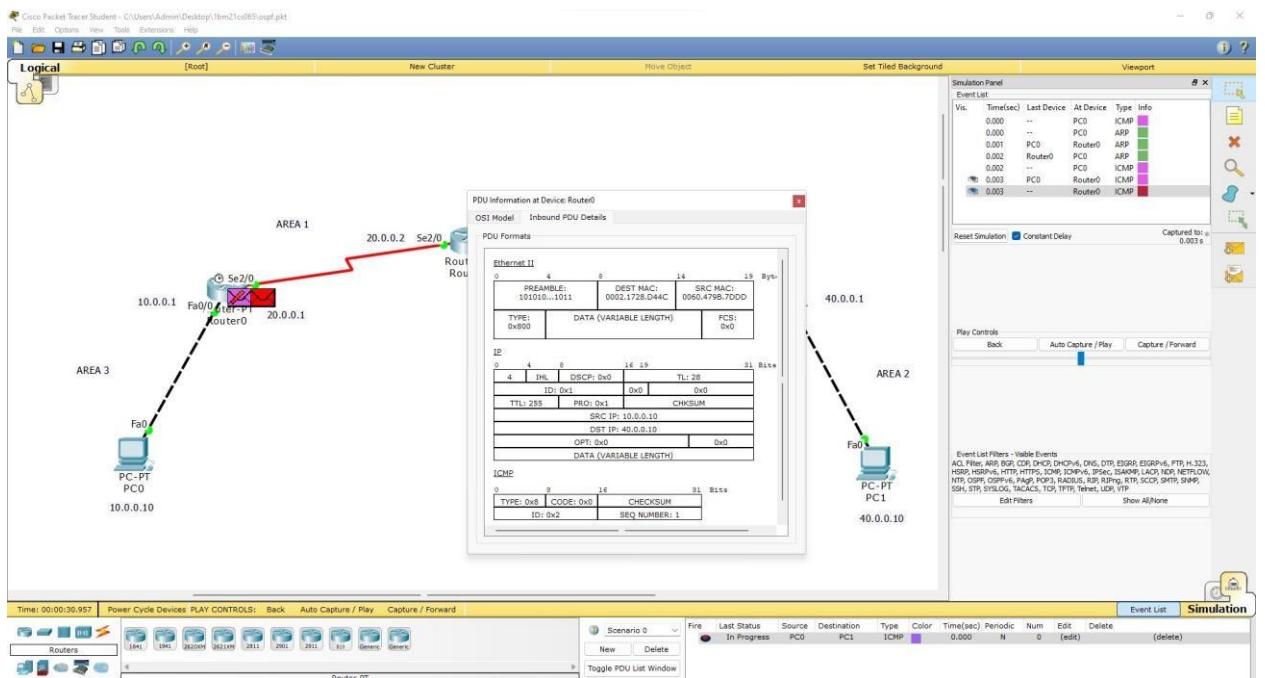
Observation:

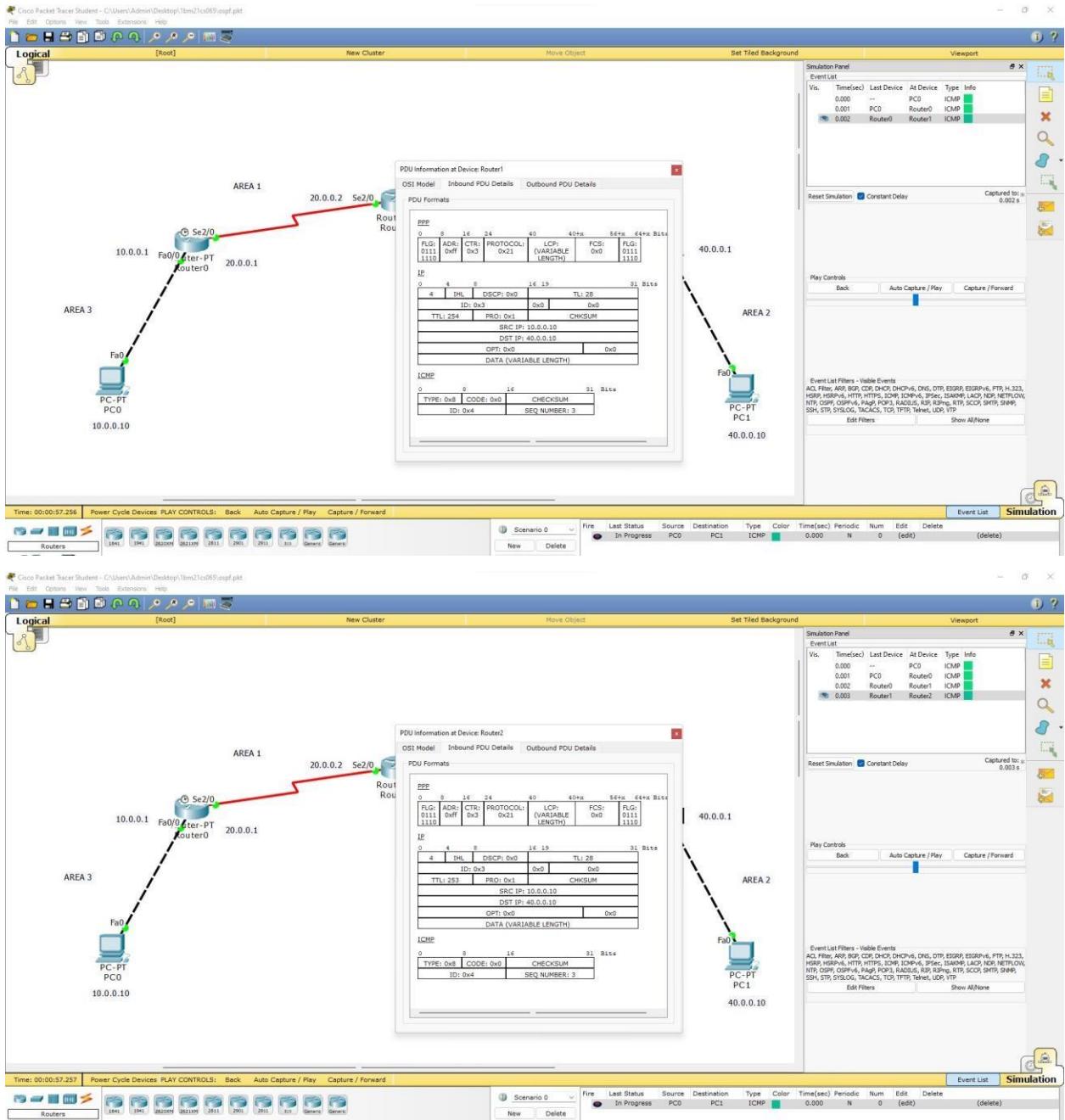
We can observe that when we simulate a ping message using simple PDU we can click on the packet and check the header of packet when clicked on capture button you can see the transitions of packet. For every loop we can observe that the value of TTL in IP header decreases by 1 value.

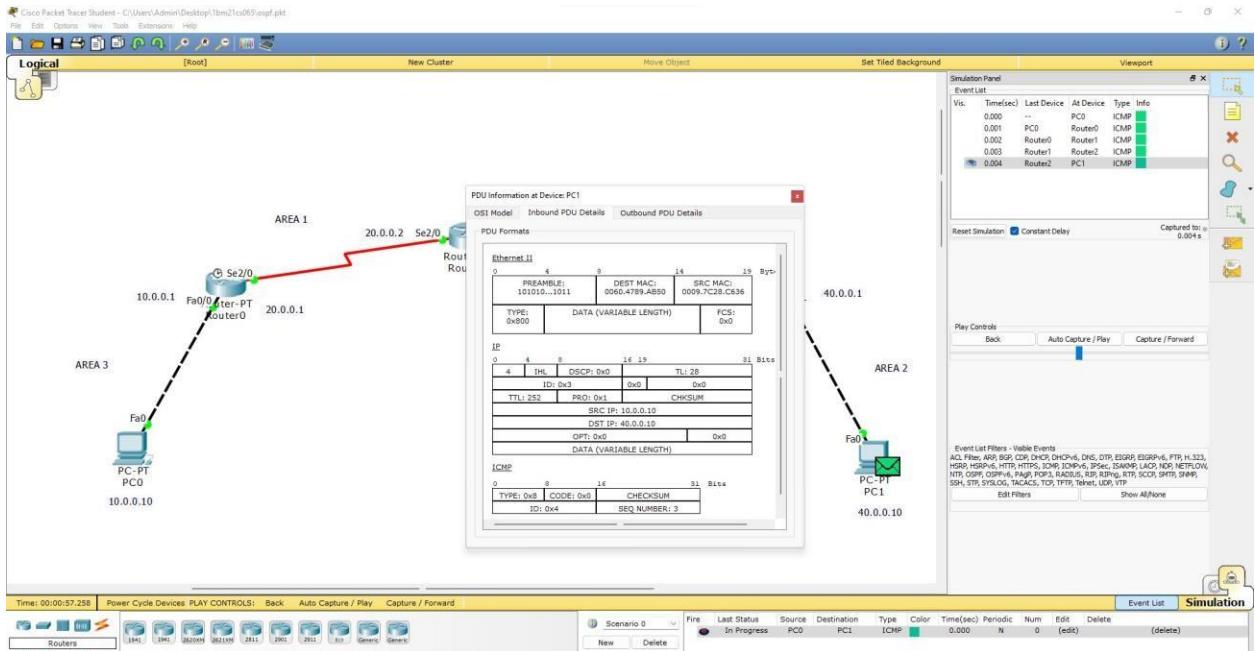
TOPOLOGY:



OUTPUT:



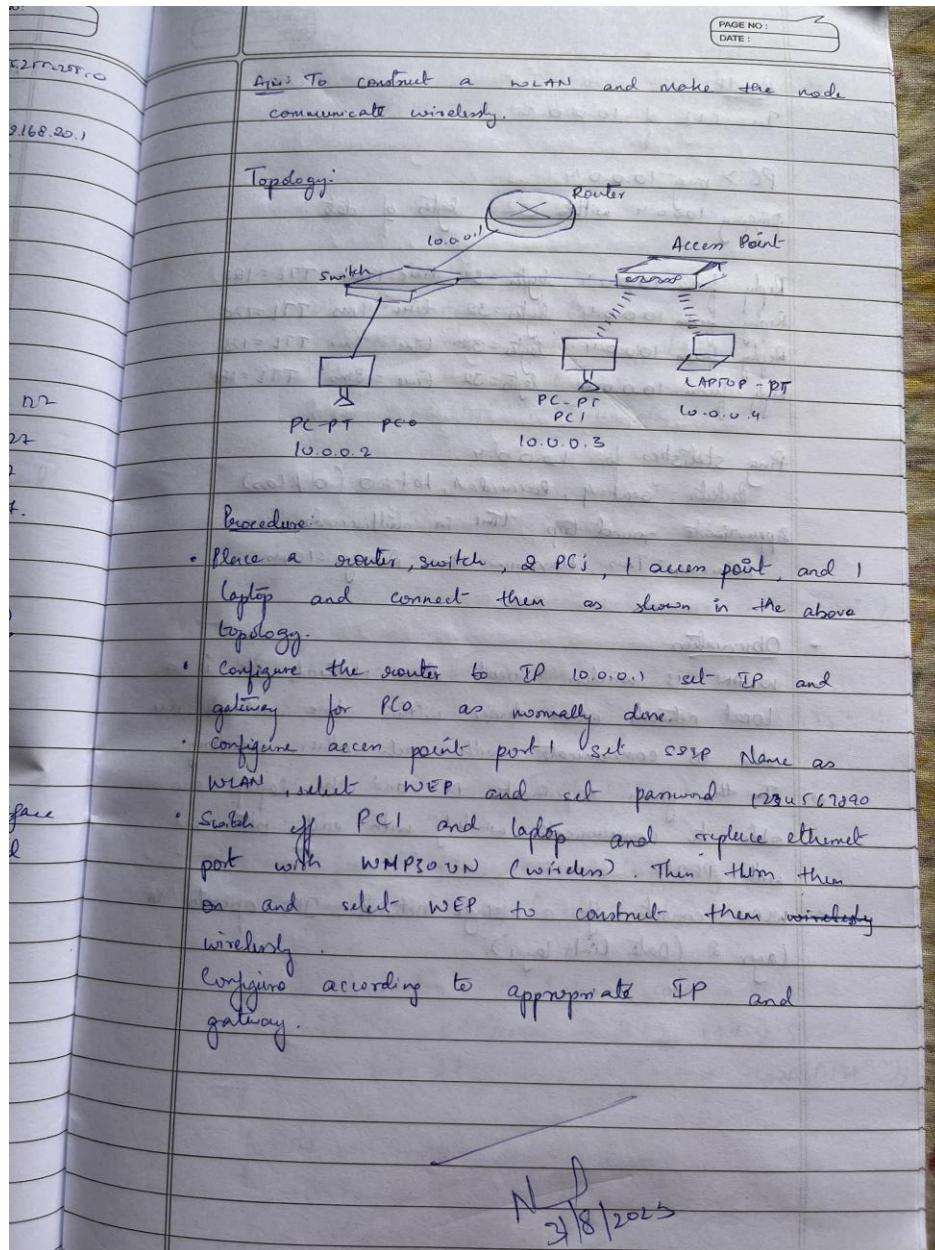




LAB 11

To construct a WLAN and make the nodes communicate wirelessly

OBSERVATION:



Result:

In CTS of 10.0.0.2

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 = 19ms 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 = 8ms TTL = 128

Ping statistics for 10.0.0.4

Packets: sent = 4, received = 4, lost = 0 (0% loss)

Approximate round-trip time in milliseconds

Minimum = 7ms, Maximum = 19ms, Average = 10ms.

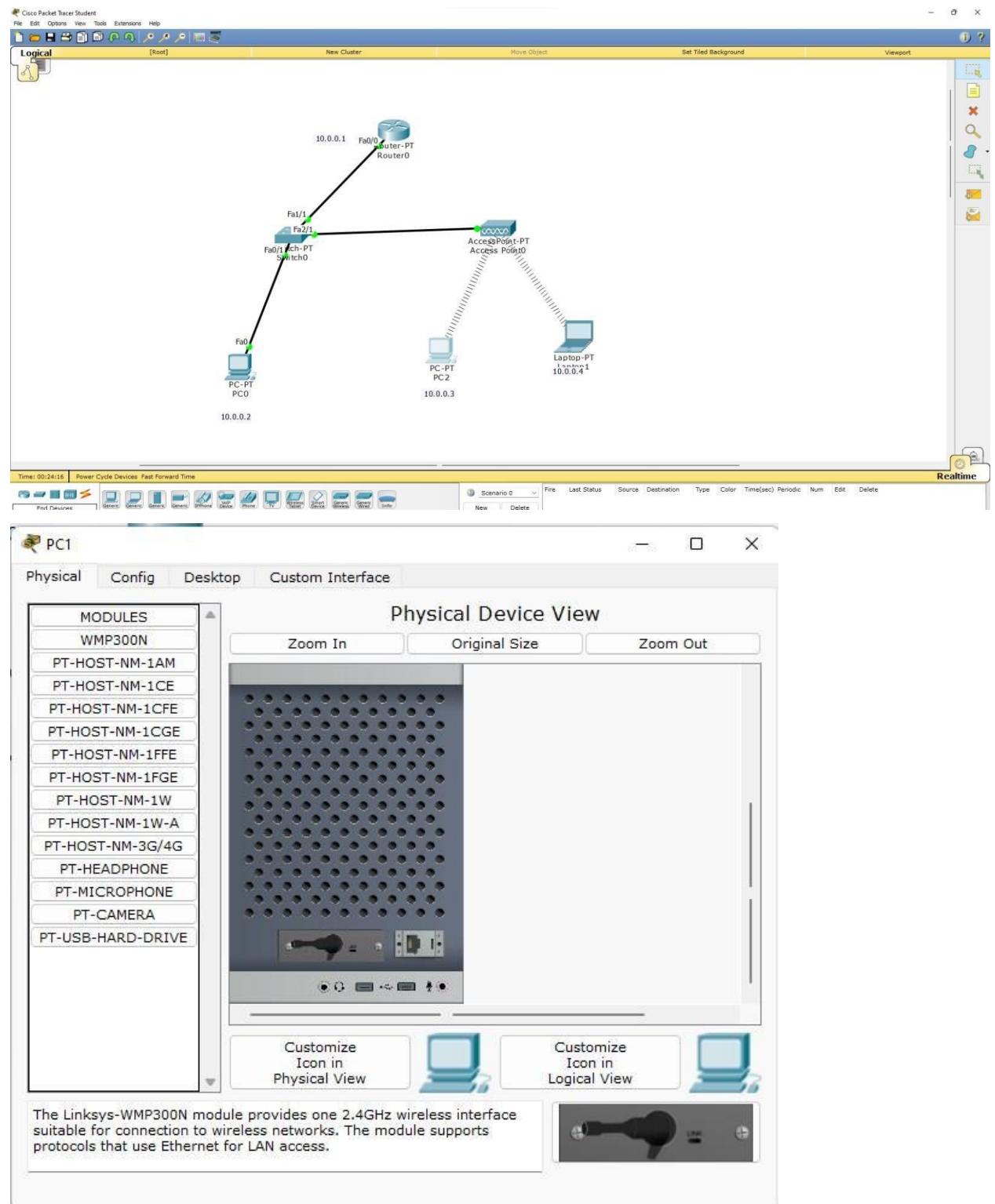
* Observation:

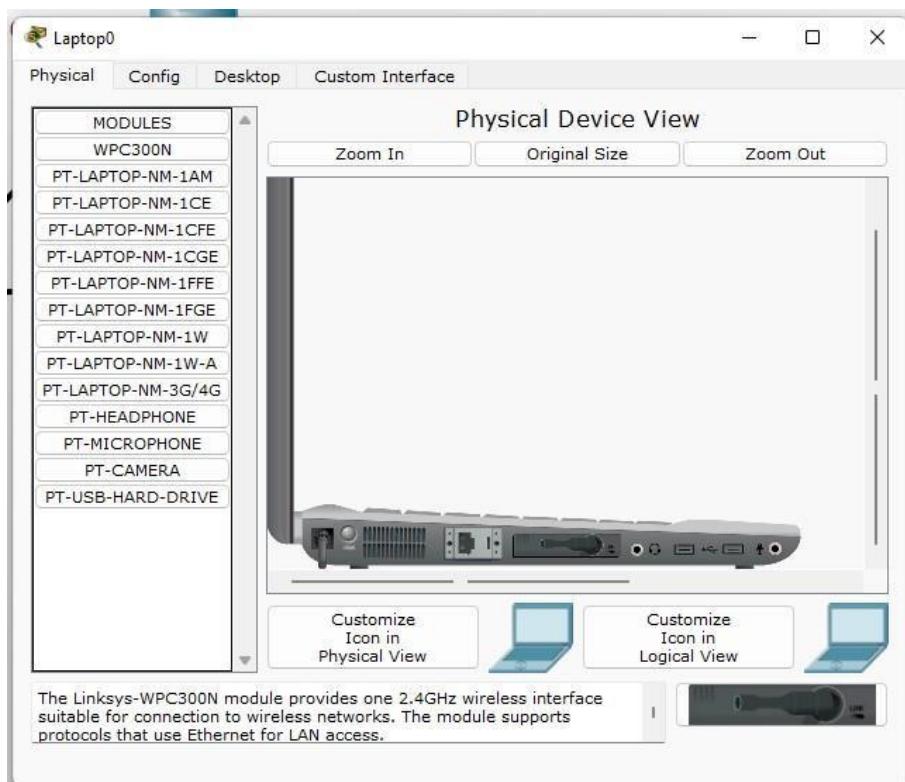
→ WLAN is wireless Local Area Network. It is a local network where devices within the network are able to communicate with each other wirelessly.

→ In the given experiment, PC1 and the laptop are able to communicate wirelessly with each other and with PC0.

→ WLAN consists of a single network. It operates in Layer 2 (Data Link layer).

TOPOLOGY:





OUTPUT:

```
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

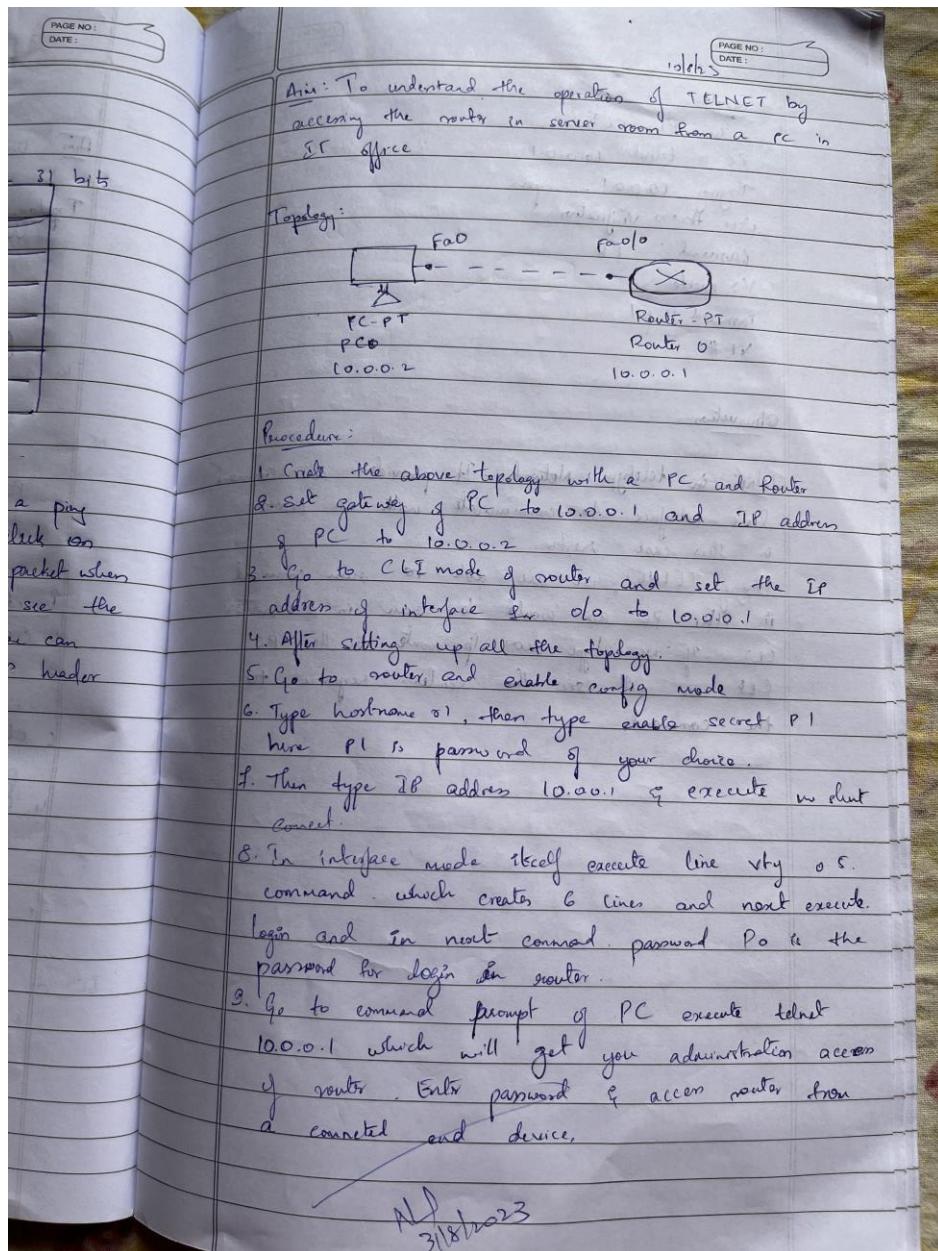
Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=21ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=10ms 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 = 7ms, Maximum = 21ms, Average = 11ms
PC>
```

LAB 12

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

OBSERVATION:



-> Result

In PC CLI

PC > telnet 10.0.0.1

Trying 10.0.0.1...open
User Access Verification

Password : PO

V> enable

Password : pl

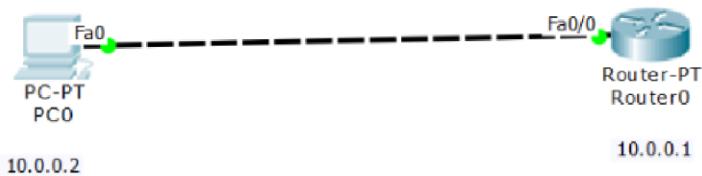
V1 #

Observation:

Telnet in tele-type Network. It provides a command line interface to communicate with a server, in this case router.

Using TELNET, we are able to run command in the PC as would be run in a router CLI. If we type show ip route in the PC CLI, we will see the routers response to the command.

TOPOLOGY:



OUTPUT:

```
PC0
Physical Config Desktop Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=1ms TTL=255

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

User Access Verification
Password:
* Password: timeout expired!
[Connection to 10.0.0.1 closed by foreign host]
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

User Access Verification
Password:
* Password:
* Password:
[Connection to 10.0.0.1 closed by foreign host]
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

User Access Verification
Password:
* Password:
* Password:
[Connection to 10.0.0.1 closed by foreign host]
PC>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

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

The screenshot shows a terminal window titled "Command Prompt" running on a "PC0" host. The session starts with a ping command to the router at 10.0.0.1, which receives four replies. It then attempts to establish a telnet connection to the router, but the connection is closed due to a password timeout. This process repeats three more times. Finally, the user runs the "show ip route" command, which displays the routing table. The table shows a single entry for the network 10.0.0.0/8, which is directly connected via the FastEthernet0/0 interface.

LAB 13

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

CODE:

```
#include<stdio.h> int arr[17];
```

```
void xor(int x[], int y[])
{
    int k=0;
    for(int i=1;i<16;i++)
    { if(x[i]==y[i]) arr[k++]=0;
      else arr[i]=1;
    }
}
```

```
void main()
{ int dd[17],div[33],ze[17],i,k;

printf("Enter the dataword \n");
for(i=0;i<17;i++) scanf("%d",&div[i]);

for(i=i;i<33;i++) div[i]=0;

for(i=0;i<17;i++) ze[i]=0;
printf("Enter dividend \n"); for(i=0;i<17;i++)
scanf("%d",&dd[i]);

i=0; k=0;

for(i=i;i<17;i++)
arr[k++]=div[i];

while(i<33)
{ if(arr[0]==0) xor(arr,ze);
  else
```

```

xor(arr,dd); arr[16]=div[i++];

}

k=0; for(i=17;i<33;i++) div[i]=arr[k++];
printf("Codeword: "); for(i=0;i<33;i++)
printf("%d",div[i]);

for(i=0;i<17;i++) arr[i]=0; printf("\nAt receiver end
\n");

k=0;

for(i=i;i<17;i++)
    arr[k++]=div[i];

while(i<33)
{ if(arr[0]==0) xor(arr,ze); else xor(arr,dd);
    arr[16]=div[i++];
}

k=0;
for(i=17;i<33;i++) div[i]=arr[k++]; printf("Codeword: ");
for(i=0;i<33;i++) printf("%d",div[i]);
}

```

OUTPUT:

```
C:\Users\Admin\Desktop\1BM21CS047\ADA\CRC16\bin\Debug\CRC16.exe
Enter the dataword
1 0 1 1 0 0 1 1 1 0 0 1 0 1 1 1
Enter dividend
1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1
Codeword: 101100111100101110000000000011011
At receiver end
Codeword: 10110011110010111000000000000000
Process returned 1 (0x1) execution time : 49.507 s
Press any key to continue.
```

OBSERVATION:

Date: 17/8/23

Labs 1.0 :

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

Java program:

```

import java.util.Scanner;
class Main {
    public static void main (String args) {
        Scanner in = new Scanner (System.in);
        System.out.println ("Enter message /data bits:");
        String message = in.nextLine ();
        System.out.println ("Enter Generator:");
        String generator = in.nextLine ();
        int data [] = new int [generator.length ()];
        int divisor [] = new int [generator.length ()];
        for (int i=0; i<message.length (); i++) {
            data [i] = Integer.parseInt (message.charAt (i));
        }
        for (int i=0; i<generator.length (); i++) {
            divisor [i] = Integer.parseInt (generator.charAt (i));
        }
        for (int i=0; i<message.length (); i++) {
            if (data [i] == 1)
                for (int j=0; j<divisor.length(); j++)
                    data [i+j] = data [i+j] ^ divisor [j];
        }
        System.out.println ("The checksum code is ");
        for (int i=0; i<message.length (); i++) {
            data [i] = Integer.parseInt (message.charAt (i));
        }
        for (int i=0; i<data.length; i++)
            System.out.println (data [i]);
        System.out.println ();
    }
}

```

N P
3/8/2023

```
System.out.println("Enter checksum code : ");
message = in.nextLine();
System.out.println("Enter generator : ");
generator = in.nextLine();

for (int i=0; i<message.length(); i++)
    data[i] = Integer.parseInt(message.charAt(i) + " ")
for (int i=0; i<generator.length(); i++)
    divisor[i] = Integer.parseInt(generator.charAt(i) + " ")
for (int i=0; i<message.length(); i++) {
    if (data[i] == 1)
        for (int j=0; j<divisor.length(); j++)
            data[i+j] = divisor[j];
}
boolean valid = true;
for (int i=0; i<data.length; i++) {
    if (data[i] == -1) {
        valid = false;
        break;
    }
}
if (valid)
    System.out.println("Data stream is valid!");
else
    System.out.println("Data stream is invalid! CRC error");
in.close();
return 0;
```

Output:

Enter message/data bits: 1110000

Enter generator: 1001

The checksum code is: 1110000111

Enter checksum code: 1110000111

Enter generator: 1001

Data stream is valid

Enter message/data bits: 11110000

Enter generator: 1001

The checksum code is: 1110000111

Enter checksum code: 1110000110

Enter generator: 1001

Data stream is not valid! CRC error.

3/8/2023

Lab 14

Write a program for congestion control using Leaky bucket algorithm.

CODE:

```
#include <stdio.h>
#include <stdlib.h> // Include this for the rand() function int main()
{
    int buckets, outlets, k = 1, num, remaining;
    printf("Enter Bucket size and outstream size\n"); scanf("%d %d", &buckets, &outlets);
    remaining = buckets; while (k)
    {
        num = rand() % 1000; // Generate a random number between 0 and 999 if (num < remaining)
        {
            remaining = remaining - num; printf("Packet of %d bytes
accepted\n", num); // Added missing
variable
        }
        else
        {
            printf("Packet of %d bytes is discarded\n", num);
        }
        if (buckets - remaining > outlets)
        {
            remaining += outlets; // Fixed the calculation
        }
        else remaining = buckets; printf("Remaining
bytes: %d \n", remaining); printf("If you want to stop input, press 0, otherwise, press
1\n"); scanf("%d", &k);
    }
    while (remaining < buckets) // Fixed the condition
    {
        if (buckets - remaining > outlets)
```

```

{
    remaining += outlets; // Fixed the calculation
}

else remaining = buckets;
printf("Remaining bytes: %d \n", remaining);

}

return 0; // Added a return statement to indicate successful completion
}

```

OUTPUT:

```

PS D:\VS Code> cd "d:\VS Code\OS\" ; if ($?) { gcc bucket.c -o bucket } ; if ($?) { ./bucket }
Enter Bucket size and outstream size
2000
100
Packet of 41 bytes accepted
Remaining bytes: 2000
If you want to stop input, press 0, otherwise, press 1
1
Packet of 467 bytes accepted
Remaining bytes: 1633
If you want to stop input, press 0, otherwise, press 1
1
Packet of 334 bytes accepted
Remaining bytes: 1399
If you want to stop input, press 0, otherwise, press 1
1
Packet of 500 bytes accepted
Remaining bytes: 999
If you want to stop input, press 0, otherwise, press 1
1
Packet of 169 bytes accepted
Remaining bytes: 930
If you want to stop input, press 0, otherwise, press 1
1
Packet of 724 bytes accepted
Remaining bytes: 306
If you want to stop input, press 0, otherwise, press 1
1
Packet of 478 bytes is discarded
Remaining bytes: 406
If you want to stop input, press 0, otherwise, press 1
1
Packet of 358 bytes accepted
Remaining bytes: 148
If you want to stop input, press 0, otherwise, press 1
1
Packet of 962 bytes is discarded
Remaining bytes: 248
If you want to stop input, press 0, otherwise, press 1
0
Remaining bytes: 348
Remaining bytes: 448
Remaining bytes: 548
Remaining bytes: 648
Remaining bytes: 748

```

```
Remaining bytes: 348
Remaining bytes: 448
Remaining bytes: 548
Remaining bytes: 648
Remaining bytes: 748
Remaining bytes: 848
Remaining bytes: 948
Remaining bytes: 1048
Remaining bytes: 1148
Remaining bytes: 1248
Remaining bytes: 1348
Remaining bytes: 1448
Remaining bytes: 1548
Remaining bytes: 1648
Remaining bytes: 1748
Remaining bytes: 1848
Remaining bytes: 1948
Remaining bytes: 2000
PS D:\VS Code\OS> □
```

OBSERVATION:

17/10/23

Aim: write a program for configuration control using Leaky bucket algorithm

```

C Program:
#include <stdio.h>
#include <stdlib.h>
#define capacity 20
void main () {
    int timeLimit = 10, bucketCapacity = 0, OutputRate = 5;
    while (timeLimit < 20) {
        int newPacket;
        printf ("Enter new packet size = ");
        scanf ("%d", &newPacket);
        if (newPacket < capacity) {
            bucketCapacity += newPacket;
            printf ("In Bucket capacity = %d, bucket capacity = %d", bucketCapacity, capacity);
            bucketCapacity -= OutputRate;
            printf ("In Bucket capacity after output = %d, bucket capacity = %d", bucketCapacity, capacity);
            timeLimit++;
        } else if (newPacket > capacity || (newPacket + bucketCapacity) > capacity) {
            printf ("New packet cannot be added to packet");
            bucketCapacity -= OutputRate;
            printf ("In Bucket capacity after output = %d, bucket capacity = %d", bucketCapacity, capacity);
            timeLimit++;
        } bucketCapacity = 0;
    }
}

```

→ Output
Enter 1
Bucket
Enter
New
Bucket
Enter
Bucket
Enter
Bucket
Enter
Bucket
Bucket
18

```
PAGE NO: _____  
DATE: _____  
  
of using  
pointy ("In Bucket capacity after output -> l,d",  
        bucket capacity);  
timeLimit ++;  
exit (0);  
3  
3  
  
→ Output:  
→ Enter New packet size = 15  
bucket capacity : 10  
Enter new packet size = 27  
New packet cannot be added to bucket  
bucket capacity after output = 5  
Enter new packet size = 0  
bucket capacity : 5  
bucket capacity after output : 0  
Enter new packet size : 0  
Bucket capacity after output : -5  
P  
  
(Ch 10) Queuing mechanism  
(under + Priority queuing in IP)  
11/8/2023
```

WEEK15

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.

CODE:

```
ClientTCP.py from socket import *  
serverName = "127.0.0.1"  
serverPort = 12000  
clientSocket = socket(AF_INET, SOCK_STREAM)  
clientSocket.connect((serverName, serverPort))  
sentence = input("\nEnter file name: ")  
clientSocket.send(sentence.encode())  
filecontents = clientSocket.recv(1024).decode()  
print("\nFrom 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("\nSent contents of " + sentence)  
    file.close()  
    connectionSocket.close()
```

OUTPUT:

The image shows two separate Python IDLE shells running simultaneously. Both shells are titled "IDLE Shell 3.11.4".

Left Shell (ClientTCP.py):

```
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\Admin\Desktop\lkm21cs065\ClientTCP.py =====
Enter file name:ServerTCP.py

From server:

from socket import *
serverName="172.0.0.1"
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('\nSent contents of' + sentence)
    file.close()
    connectionSocket.close()

>>>
```

Right Shell (ServerTCP.py):

```
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\Admin\Desktop\lkm21cs065\ServerTCP.py =====
The server is ready to receive
Sent contents ofServerTCP.py
The server is ready to receive
```

OBSERVATION:

Lab - II

DATE: 24/8/23

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 required file of present

Server TCP.py:

```

from socket import *
serverName = "127.0.0.1" → logbook address
serverPort = 12000
serverSocket = Socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen()
while True:
    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("In sent contents of " + sentence)
    file.close()
    connectionSocket.close()

```

Client TCP.py:

```

from socket import *
serverName = "127.0.0.1"
serverPort = 12000

```

PAGE NO: _____
DATE: _____

- server file

```
Client Socket = socket (AF_INET, SOCK_STREAM)
Client Socket . connect ((serverName, serverPort))
sentence = input ("In Enter file name:")
Client Socket . send (sentence . encode ())
file contents = Client Socket . recv (1024) . decode ()
print ("In From Server :\n")
print (file contents)
Client Socket . close ()
```

Procedure:

- Create 2 IDLE instance and write client and server files
- Run server first and then the client.

Output: *ND off 17/02*

Server Instance:
The server is ready to receive

client instance:-
Enter file name: ServerTCP.py

from server:
the contents of server TCP.py is displayed here

Server Instance:-
The server is ready to receive

Sent contents of server TCP.py
The server is ready to receive!

LAB 16

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

CODE:

ClientUDP.py

```
from socket import *  
serverName = "127.0.0.1"  
serverPort = 12000  
clientSocket = socket(AF_INET, SOCK_DGRAM)  
sentence = input("\nEnter file name: ")  
clientSocket.sendto(bytes(sentence,"utf8"),(serverName, serverPort))  
filecontents,serverAddress = clientSocket.recvfrom(2048)  
print ("\nReply from Server:\n")  
print (filecontents.decode("utf-8")) # for i in filecontents: # print(str(i), end = "  
")  
clientSocket.close()  
clientSocket.close()  
  
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 ("\nSent contents of ", end = " ")  
    print (sentence) # for i in sentence: # print (str(i), end = " ")  
    file.close()
```

OUTPUT:

The image shows two windows of the Python IDLE Shell 3.11.4 running on Windows 10. Both windows have identical titles: "IDLE Shell 3.11.4".

Left Window (Client Side):

```
>>> Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:\Users\Admin\Desktop\lkm2lcs065\ClientUDP.py
Enter file name: ServerUDP.py
Reply from Server:
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ("\nSent contents of ", end = " ")
    print (sentence)
    # for i in sentence:
    #     print (str(i), end = '')
    file.close()

>>>
```

Right Window (Server Side):

```
>>> Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:\Users\Admin\Desktop\lkm2lcs065\ServerUDP.py
The server is ready to receive
Sent contents of  ServerUDP.py
```

OBSERVATION:

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.

Server UDP.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")
```

```
fde = open(sentence, "r")
```

```
con = fde.read(2048)
```

```
serverSocket.sendto(bytes(con, "utf-8"), clientAddress)
```

```
print("In sent contents of ", end="")
```

```
print(sentence)
```

```
fde.close()
```

Client UDP.py

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Program and

```
sentence = input("In Enter file name = ")  
client:  
clientSocket = socket(AF_INET, SOCK_DGRAM)  
  
sentence = input("In Enter file name = ")  
client:  
clientSocket.sendto(sentence.encode("utf-8"),  
(serverName, serverPort))  
fileContents, serverAddress = clientSocket.recvfrom(1024)  
print("In Reply from server: " + fileContents.decode("utf-8"))  
  
clientSocket.close()  
clientSocket.close()  
  
Output:  
MP  
07/01/2023  
Server Instance:  
The server is ready to receive  
client instance:  
Enter file name: server UDP.py  
  
Reply from server:  
contents of server UDP.py displayed here.  
  
Server Instance:  
The server is ready to receive.  
Sent contents of server UDP.py.  
The server is ready to receive.
```

LAB 17

Tool Exploration -Wireshark OBSERVATION:

PAGE NO.:
DATE:

Lab-12 3/8/23

Wireshark

Aim: Exploring functionalities of wireshark

Overview: wireshark is an open-source application that captures and displays data traveling back and forth on a network. wireshark is a packet sniffer and network analyzer.

Capturing packets:

- First select a network from which we require to sniff packets.
- wireshark begins capturing packet from selected network. All captured packets are shown in the top section of the panel.
- The following packet details are shown
 - Time stamp
 - Source IP
 - Destination IP
 - Protocol name
 - Length of packet

Filtering in wireshark:

wireshark provides a filter function to better analyze network data. wireshark also allows creating custom filters.

An example of a filter is to select only packets from HTTP. `http`

PAGE NO.:
DATE:

tcp - port = 80 / 11 udp. port = 80

Packet details:

The middle section of the panel, detail panel, presents the protocol and port number fields of the selected packet in a collapsible format. We can apply additional filters and switch click on protocol for a detailed view.

At the bottom panel, raw data of the selected packet is seen in hexadecimal format. It is called as hex dump. It contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset.

AD
07/09/2023