

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Darshna Mimrot(1BM22CS081)

in partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING**



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Academic Year 2024-25(Odd)

B.M.S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled “**COMPUTER NETWORKS**” carried out by DARSHNA MIMROT(1BM22CS081), 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 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Computer Networks Lab - (23CS5PCCON) work prescribed for the said degree.

Sandhya A Kulkarni
Assistant Professor
Department of CSE, BMSCE

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

INDEX

CYCLE 1

Sl. No.	Date	Experiment Title	Page No.
1	09-10-24	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping messages.	1-2
2	09-10-24	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	3-5
3	16-10-24	Configure default route, static route to the Router (Part 1).	6-9
4	23-10-24	Configure default route, static route to the Router (Part 2).	10-11
5	13-11-24	Configure DHCP within a LAN and outside LAN.	12-14
6	20-11-24	Configure RIP routing Protocol in Routers .	15-16
7	20-11-24	Demonstrate the TTL/ Life of a Packet.	17-19
8	27-11-24	Configure OSPF routing protocol.	20-22
9	18-12-24	Configure Web Server, DNS within a LAN.	23-24
10	18-12-24	To construct a simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	25-27
11	18-12-24	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	28-30
12	18-12-24	To construct a VLAN and make the PC's communicate among a VLAN.	31-34
13	18-12-24	To construct a WLAN and make the nodes communicate wirelessly.	35-38

INDEX

CYCLE 2

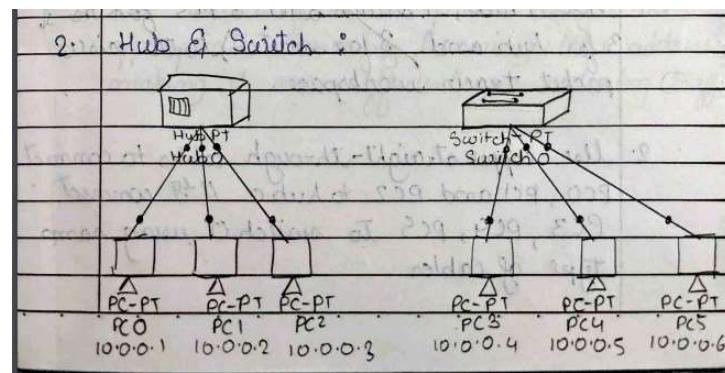
SI NO.	Date	Experiment Title	Page No.
14	18-12-24	Write a program for congestion control using Leaky bucket algorithm.	39-42
15	18-12-24	Using TCP/IP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.	43-45
16	18-12-24	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.	46-49
17	18-12-24	Write a code for error detecting code using CRC-CCITT	40-51
18	18-12-24	Tool Exploration- Wireshark	52-55

Github Link: <https://github.com/Darshna27/cn-lab->

Program 1

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

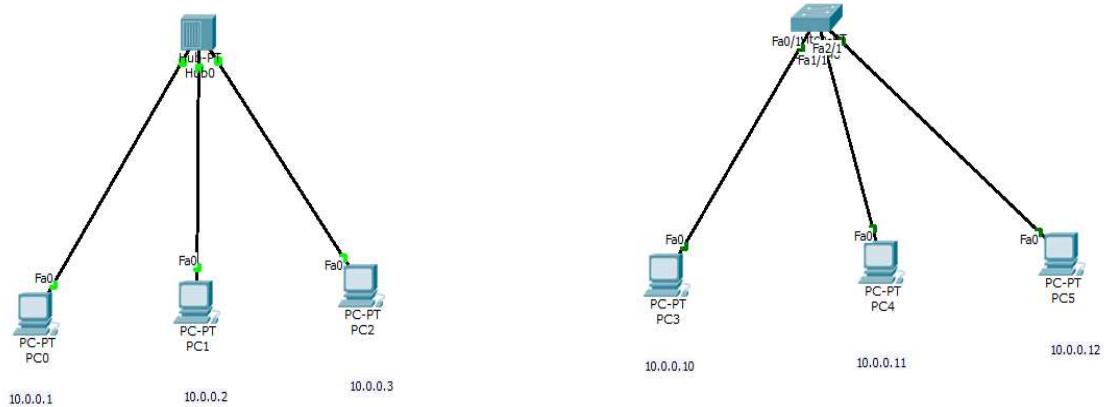
Topology , Procedure and Observation:



→ Aim : To create a simple network consisting of 3 PCs connected to a central hub and another network with 3 PCs connected to a switch. This configuration will help observe the behaviour of data transmission using hub & switch devices.
→ Topology :
1. Hub network : Three PCs (PC0, PC1, PC2) are connected to a hub (Hub0) using straight-through Ethernet cables. IP addresses : PC0=10.0.0.1, PC1=10.0.0.2 PC2=10.0.0.3
2. Switch Network : Three PCs (PC3, PC4, PC5) are connected to a switch (Switch0) using straight-through Ethernet cables. IP addresses : PC3=10.0.0.4, PC4=10.0.0.5 PC5=10.0.0.6
→ Procedure :
1. Add 1 hub, 1 switch and 6 PCs for the (3 for hub and 3 for switch) to the Cisco packet tracer workspace. 2. Use copper straight-through cables to connect PC0, PC1 and PC2 to hub0. Then connect PC3, PC4, PC5 to switch0 using same type of cables.

<p>Date 1/1 Page 5</p> <p>a. Assign IP addresses to each PC & obtain subnet mask.</p> <p>4. Switch to simulation mode to observe data traffic behaviour when packets are sent between the devices.</p> <p>5. In the hub network, notice how the hub broadcasts packets to all devices, causing potential traffic overload. In the switch network, observe how the switch forwards packets only to the intended recipient reducing unnecessary traffic.</p> <p>6. The hub broadcasts data to all connected devices leading to more network congestion, while the switch efficiently sends data only to the correct device, optimizing performance.</p> <p>Observations:</p> <ul style="list-style-type: none"> 1. The hub broadcasts packets to all devices, which may cause unnecessary traffic. 2. The switch forwards packets only to the appropriate device by learning MAC address, making it more efficient in reducing traffic. 	<p>Difference b/w Hubs & switches</p> <table border="1"> <thead> <tr> <th>HUBS</th><th>SWITCHES</th></tr> </thead> <tbody> <tr> <td>1. Hub Broadcast data to all devices</td><td>Switches send it only to the destination</td></tr> <tr> <td>2. Hubs create more traffic</td><td>Switches reduces traffic by directing data.</td></tr> <tr> <td>3. Hubs work at physical layer</td><td>Switches operate at the data link layer</td></tr> <tr> <td>4. Hubs are slower</td><td>Switches are faster due to shared bandwidth with dedicated bandwidth.</td></tr> <tr> <td>5. Hubs are cheaper</td><td>Switches are more expensive but more efficient</td></tr> </tbody> </table>	HUBS	SWITCHES	1. Hub Broadcast data to all devices	Switches send it only to the destination	2. Hubs create more traffic	Switches reduces traffic by directing data.	3. Hubs work at physical layer	Switches operate at the data link layer	4. Hubs are slower	Switches are faster due to shared bandwidth with dedicated bandwidth.	5. Hubs are cheaper	Switches are more expensive but more efficient
HUBS	SWITCHES												
1. Hub Broadcast data to all devices	Switches send it only to the destination												
2. Hubs create more traffic	Switches reduces traffic by directing data.												
3. Hubs work at physical layer	Switches operate at the data link layer												
4. Hubs are slower	Switches are faster due to shared bandwidth with dedicated bandwidth.												
5. Hubs are cheaper	Switches are more expensive but more efficient												

Screen Shots:



Program 2

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

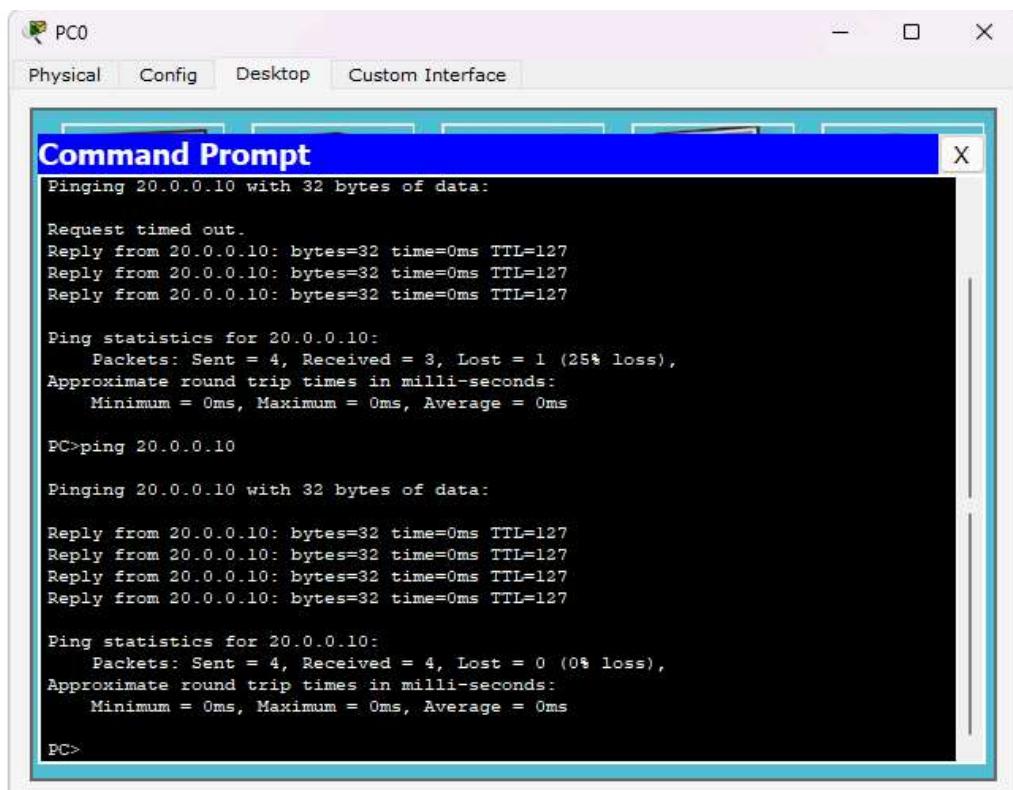
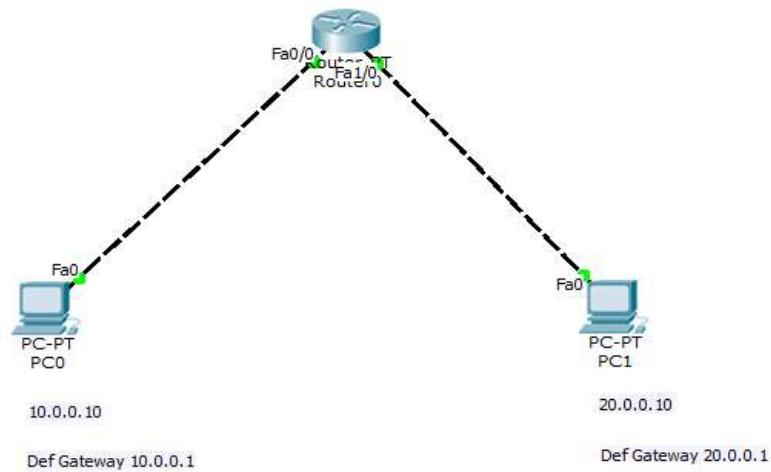
Topology , Procedure and Observation:

<p><u>LAB-2</u></p> <p>Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.</p> <p>→ Aim: To create a network connectivity of 2 PCs connected to a central router. The configuration will help us observe the behaviour of data transition using 2 PCs and server, router devices.</p> <p>→ Topology:</p> <p>1. PC0 - Two PC's are connected to router. PC0 → configured with gateway 10.0.0.1 IP address - 10.0.0.10</p> <p>2. Router PT - configured router through CLI user to run the commands Router > enable</p>	<p>Config terminal</p> <pre>interface fastethernet 0/0 IP address 10.0.0.1 255.0.0.0 no shutdown exit</pre> <p>Interface fastethernet 1/0</p> <pre>IP address 20.0.0.1 255.0.0.0 no shutdown</pre> <p>Interface eth fastethernet 1/0 IT changed state to up</p> <p>exit</p> <p>Router config is now available</p> <p>→ Procedure:</p> <ol style="list-style-type: none"> Two PC's are selected - PC0 and PC1 A generic router is selected. The Two PC's and Router are connected using Copper-Cross over It is labelled with its gateway and IP address. Procedure is selected in Realtime mode in order to do PING. After the Router is configured in CLI the
---	--

	<p>connection turns from red to green indicating that the route is configured and connection is stabilised.</p> <p>Then you Ping PC0 to PC1 and PC1 to PC0.</p> <p>→ <u>Observation :</u></p> <p><u>Setup :</u></p> <p>1. <u>Ping Results :</u> The PC's are pinged from PC0 to PC1.</p> <p>2. <u>IP results & routes :</u></p> <p>Codes: C-connected, S-static, I-IGRP, R-RIP, M-Mobile, B-BGP D-EIGRP, E-Ex-EIGRP internal, O-OSPF IA-OSPF intra area, N1-OSPF NSSA. internal Type 1, N2-OSPF NSSA internal Type 2 E1-OSPF external Type 1, E2-OSPF external Type 2, F-EGP, J-IS-IS, L-TS-TS level 1 L2-IS-TS level 2, i0-IS-TS intra and -candidate & default, U-peer-learn static route, O-ODR. P-periodic downloaded static routes.</p> <p>Gateway of last resort is not set. C 10.0.0.0/8 is directly connected, FastEthernet 0/0 C 20.0.0.0/8 is directly connected, FastEthernet 1/0.</p>
--	---

	<p>Date 1/1 Page 10</p> <p>3. If the configurations and cabling are correct, you will receive successful ping b/w two PC's</p> <p>4. If there is no connection, troubleshoot by verifying: correct IP addressing, cabling type, both router interfaces are up & running.</p> <p>5. Then you can view the IP route data via CLI.</p>
--	---

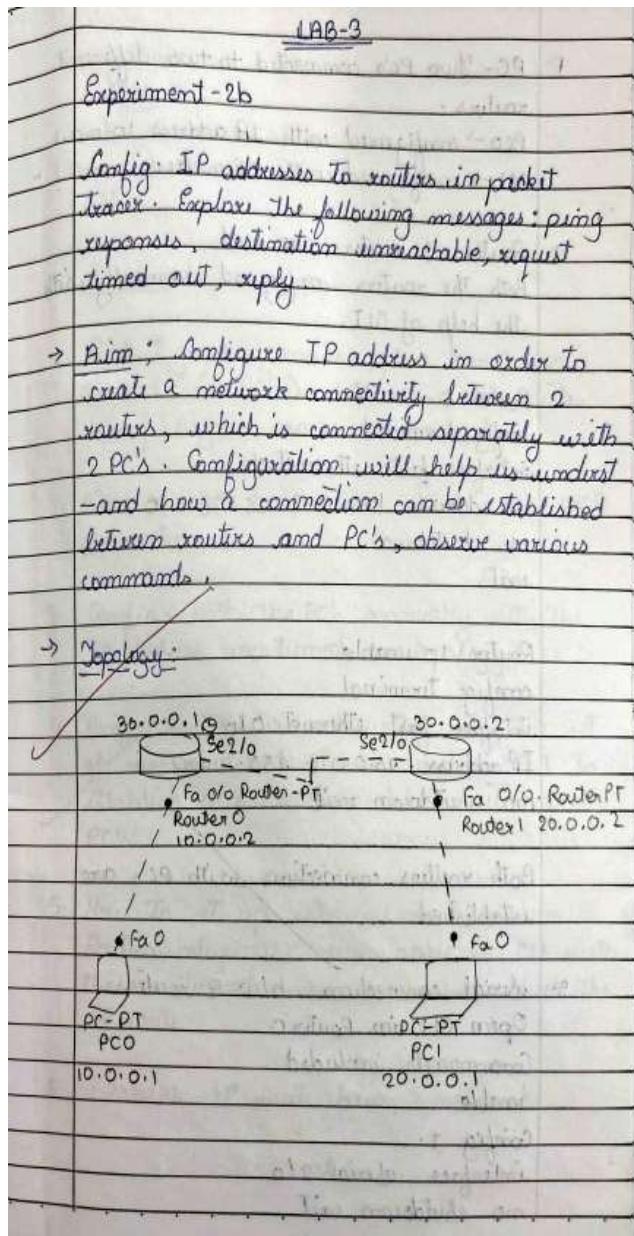
Screen Shots:



Program 3

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

Topology , Procedure and Observation:



- 1 PC - Two PCs connected to two different routers.
PC0 → configured with IP address 10.0.0.1
PC1 → configured with IP address 20.0.0.1
- 2 Router - 2 routers are used
Both the routers configured manually with the help of CLI.
- Router 0 - enable
config terminal
interface fast ethernet 0/0
IP address 10.0.0.1 255.0.0.0
no shutdown
exit.
- Router 1 - enable
config terminal
interface fast ethernet 0/0
IP address 20.0.0.1 255.0.0.0
no shutdown exit
- Both routers connection with PCs are established
- serial connection b/w 2 routers
Open CLI in Router 0.
Commands included
enable
Config t.
interface serial 2/0
no shutdown exit

Router 1
unable
config.t
interface serial 2/0
no shutdown
exit.

→ Procedure :

1. Pick 2 generic routers, place them, and connect (2 PCs), 1 PC to each router using copper cross-over connection.
2. Now connect two routers using serial DCE connection.
3. Config. both the PCs manually with the IP address mentioned in topology.
4. Config. the Routers manually using the set of commands to be used in CLI To establish a connection b/w router and PCs
5. How to set up establishment between 2 routers setup the server cable in CLI with the mentioned commands then check the IP route
6. Establish : IP with b/w 2 networks.

→ Observation :

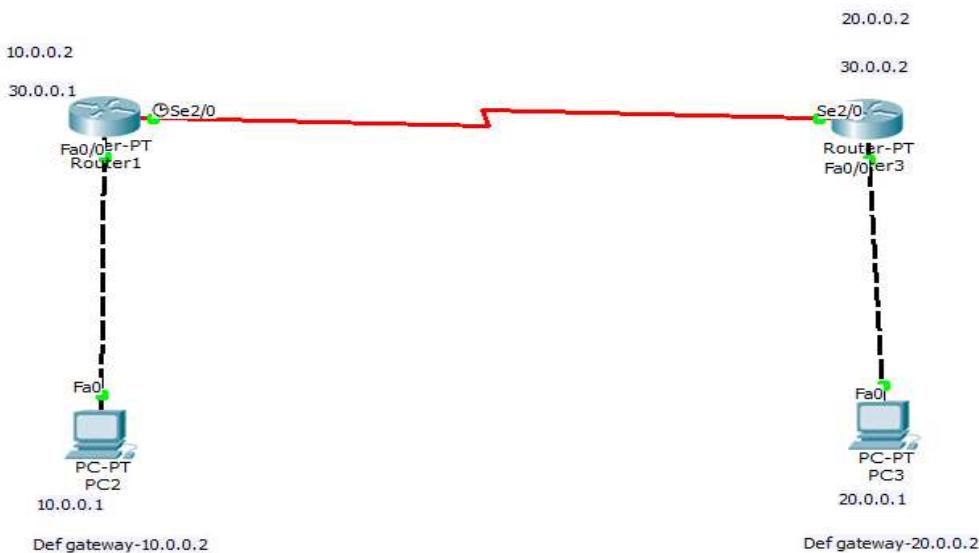
* Setup :

1. Ping Results:
When PC 0 is pinged to PC 1 in command prompt, we get a message / reply that the destination host is unreachable.
2. When PC 1 is pinged to PC 0, we get a request timed out message as right connection isn't established.
3. After establishing IP route b/w the 2 routers ping b/w 2 PCs is working.
4. Ping 10.0.0.1
PC1 → pinging 10.0.0.1 with 32 bytes of data
Reply from 10.0.0.2: destination host unreachable. Request timed out.
5. Ping 20.0.0.1
PC0 → Pinging 20.0.0.1 with 32 bytes of data
Reply from 20.0.0.2: destination host unreachable.
6. Ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1 bytes = 32 time= 4 ms

Reply from 20.0.0.1 bytes=32 time=4ms TTL=12
 Reply from 20.0.0.1 bytes=32 time=4ms TTL=12
 Reply from 20.0.0.1 bytes=32 time=4ms TTL=12
 Pinging ~~elatellis~~ for 20.0.0.1
 Packets: sent=24, received=4, lost=0 (0% loss)
 7. Pinging 10.0.0.1
 Pinging 10.0.0.1 with 32 bytes of data.
 Reply from 10.0.0.1 bytes=32 time=3ms TTL=12
 Pinging ~~elatellis~~ for 10.0.0.1
 Packets: sent=4, received=4, lost=0 (0% loss)

23/10

Screen Shots:



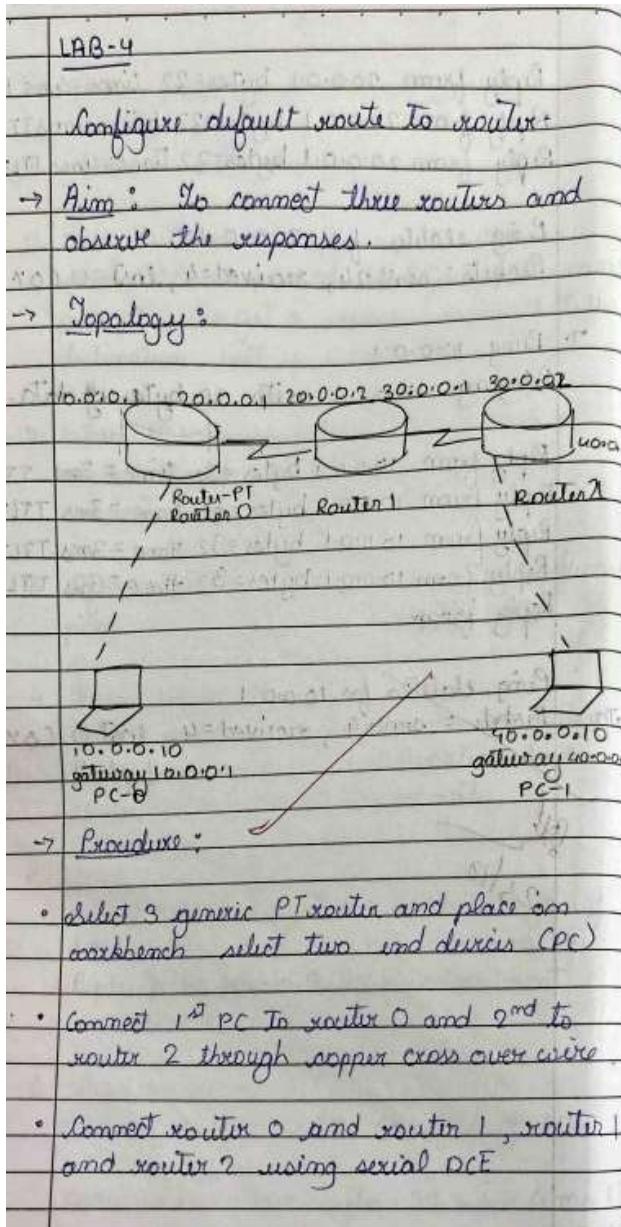
The screenshot shows a software interface titled "PC2" with tabs for "Physical", "Config", "Desktop", and "Custom Interface". A "Command Prompt" window is open, displaying the following terminal session:

```
Reply from 10.0.0.2: Destination host unreachable.  
Reply from 10.0.0.2: Destination host unreachable.  
Reply from 10.0.0.2: Destination host unreachable.  
  
Ping statistics for 20.0.0.1:  
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),  
  
PC>ping 20.0.0.1  
  
Pinging 20.0.0.1 with 32 bytes of data:  
  
Reply from 10.0.0.2: Destination host unreachable.  
  
Ping statistics for 20.0.0.1:  
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),  
  
PC>ping 20.0.0.2  
  
Pinging 20.0.0.2 with 32 bytes of data:  
  
Reply from 10.0.0.2: Destination host unreachable.  
  
Ping statistics for 20.0.0.2:  
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),  
  
PC>
```

Program 4

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

Topology , Procedure and Observation:



Configure IP addresses 10.0.0.10 to PC1 and 10.0.0.11 to PC2

R(config)# ip route 10.0.0.0 255.0.0.0
R(config)# ip route 90.0.0.0 255.0.0.0 30.0.0.2
exit

Now the 2 PCs can send messages to each other.

Now to configure default route.

Route 0
IP route 0.0.0.0 0.0.0.0 20.0.0.2

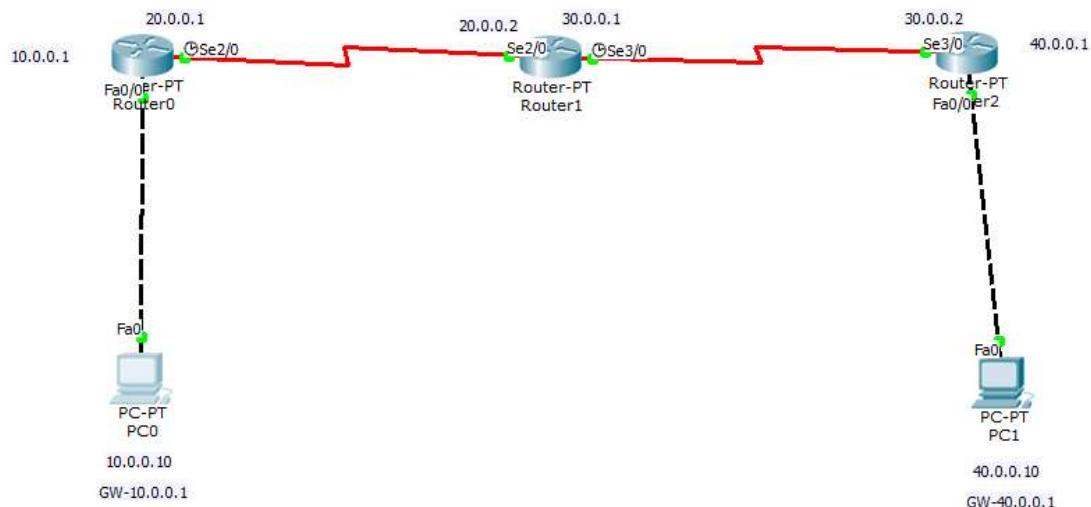
Route 2
IP route 0.0.0.0 0.0.0.0 30.0.0.1

Ping:
PC1> ping 40.0.0.10
Packets sent = 4 ; received = 4 , lost = 0 (0% loss)

→ Observation :

3 routers are connected with modes configured and they have communication

Screen Shots:



PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=5ms 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 = 5ms, Maximum = 7ms, Average = 6ms

PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

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

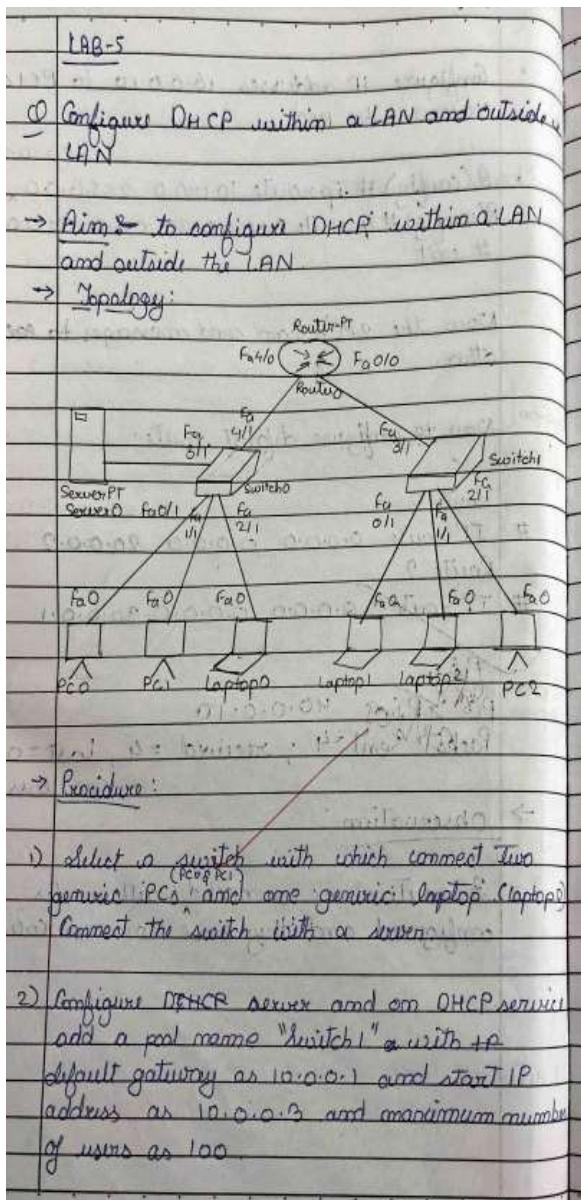
Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 6ms, Maximum = 9ms, Average = 7ms

PC>
    
```

Program 5

Aim: Configure DHCP within a LAN and outside LAN.

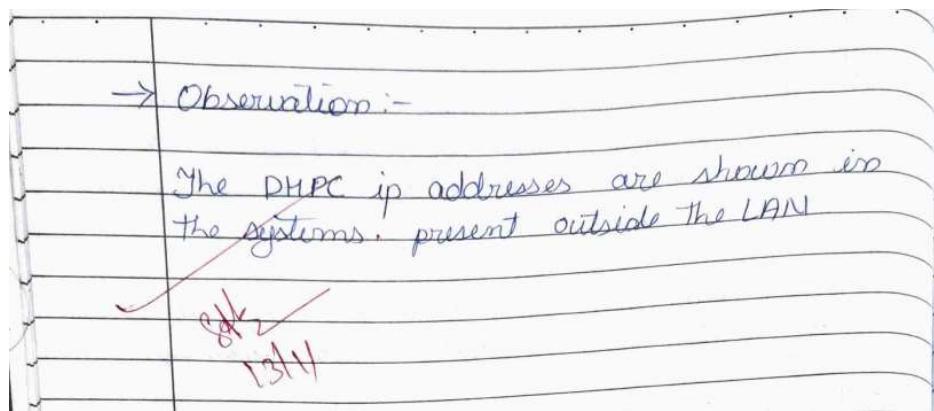
Topology , Procedure and Observation:



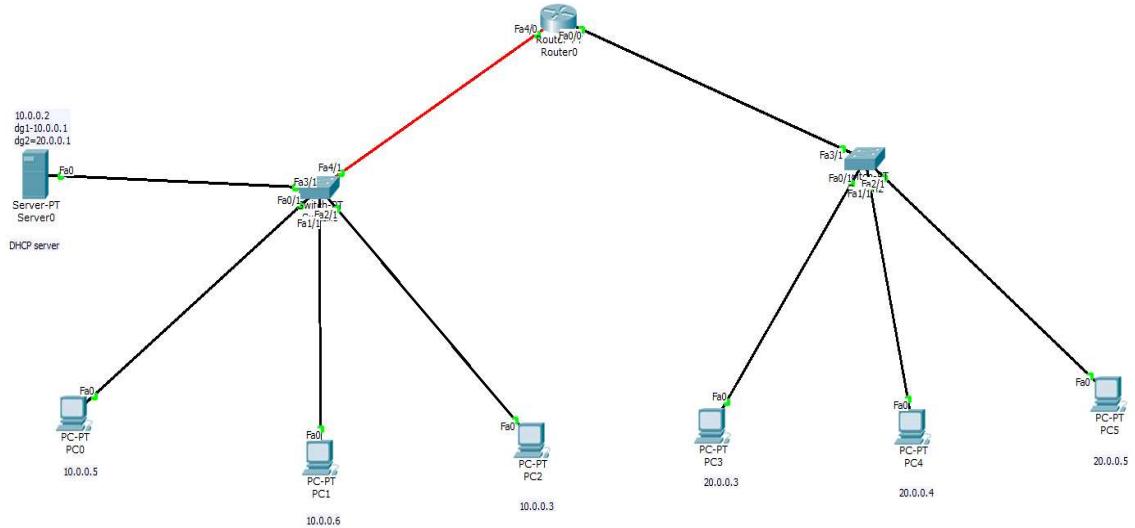
- 3) In the server add another pool name "switch2" with default gateway as 20.0.0.1 and start IP address as 20.0.0.3 with maximum number of users as 100.
 - 4) Add Configure the PCs and laptop IP address as DHCP so addresses are generated successfully.
 - 5) Add the switch to check the config in command prompt.
 - 6) Add the switch to a generic router and connect the router to another generic switch which is connected to 2 laptop (laptop1 & laptop2) and 1 generic PC (PC2).
 - 7) In Router CLI.
- ```

enable
config terminal
interface fastethernet 4/0
ip address 10.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shutdown
exit
interface fastethernet 0/0
ip address 20.0.0.1 255.0.0.0
ip helper-address 20.0.0.2
no shutdown
exit

```



### Screen Shots:



PC0

Physical Config Desktop Custom Interface

**Command Prompt**

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128

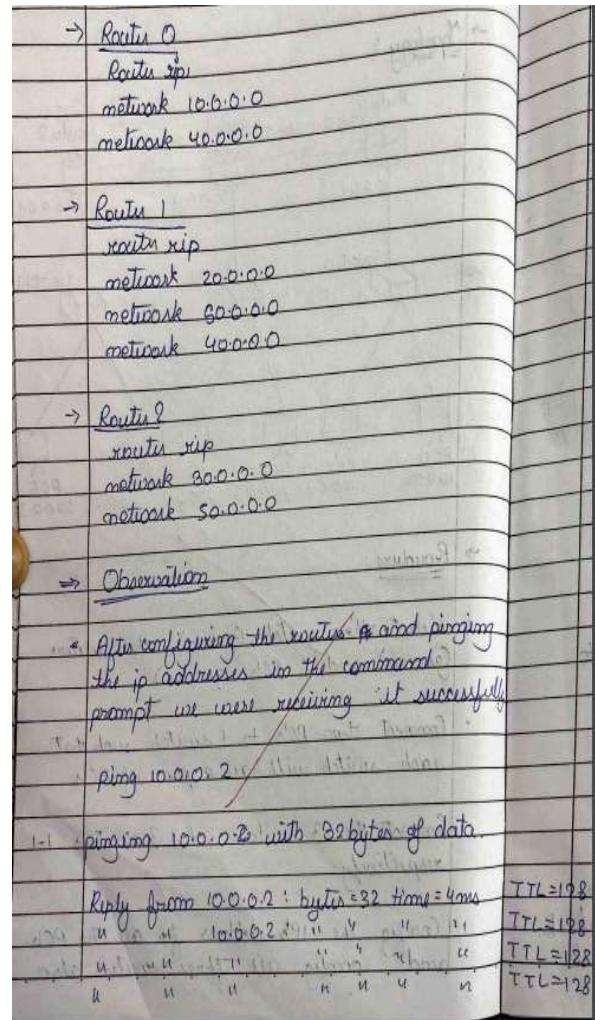
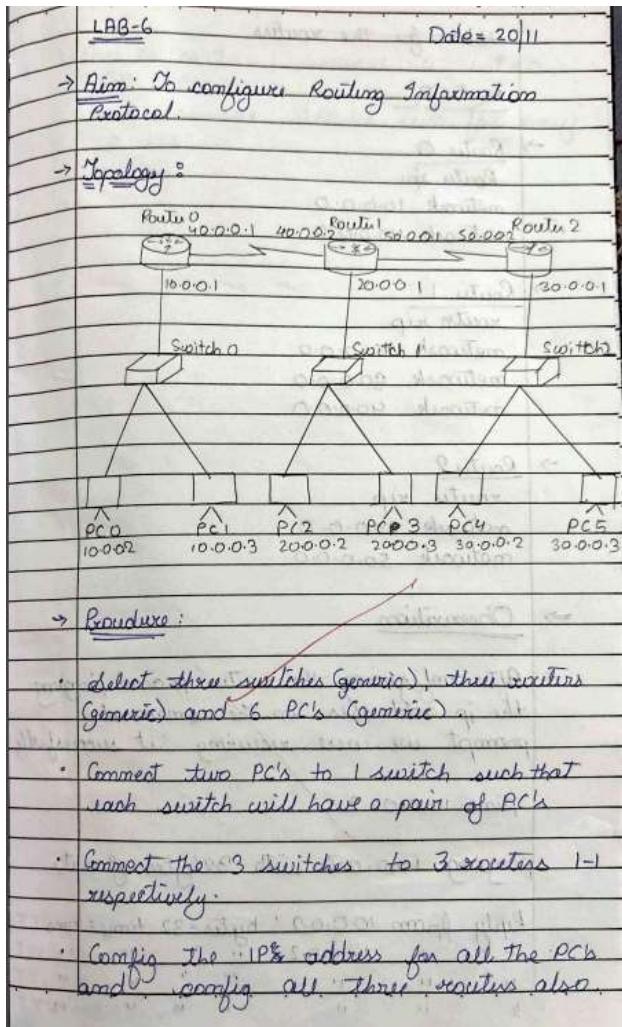
Ping statistics for 10.0.0.4:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
 Approximate round trip times in milli-seconds:
 Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>
```

## Program 6

**Aim:**Configure RIP routing Protocol in Routers .

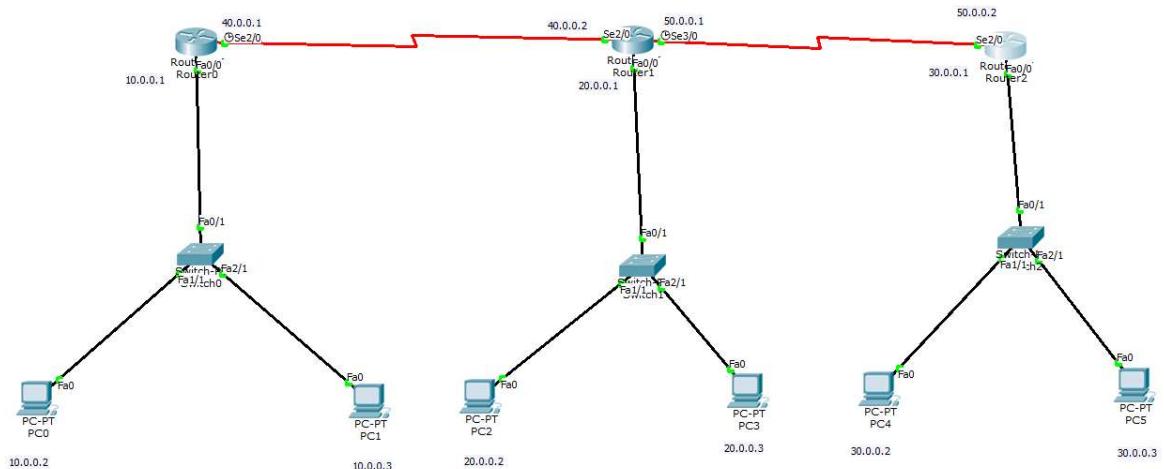
### **Topology , Procedure and Observation:**



Page \_\_\_\_\_

such message will be seen for every IP address.

## Screen Shots:



PC0

Physical Config Desktop Custom Interface

**Command Prompt**

```

Pinging 30.0.0.2 with 32 bytes of data:

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

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

PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:

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

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

PC>

```

## Program 7

**Aim:** Demonstrate the TTL/ Life of a Packet .

**Topology , Procedure and Observation:**

20-11-20-24

Demonstrate the TTL or life of a packet.

**Objective**

Demonstrate the TTL or life of a packet

**Procedure**

- Add a Simple PDU across the PC's of different networks
- Consider PC0 to PC5

**Observation**

- While Auto Capture and observing the TTL across each pc, it was observed as follows:

|                                     |           |
|-------------------------------------|-----------|
| PDU information at Device : PC1     | TTL : 255 |
| PDU information at Device : Router1 | TTL : 254 |
| PDU information at Device : Router2 | TTL : 253 |

- Cisco packet tracer has the maximum TTL as 255
- It is observed that the TTL decrements on the message is being passed stop by stop (router to router)
- The figure of OSI model of switch demonstrates flow of packets in 2 layers while 3 layers in the router
- The TTL reaches 250 once all the re-packets are received.

20-11-2024.

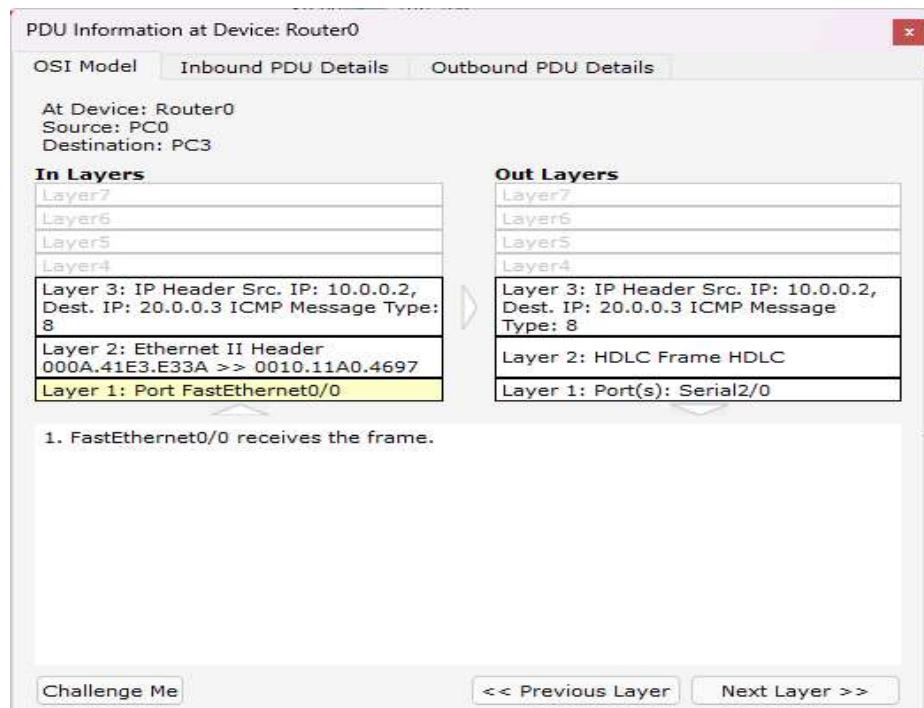
> ping 20.0.0.2

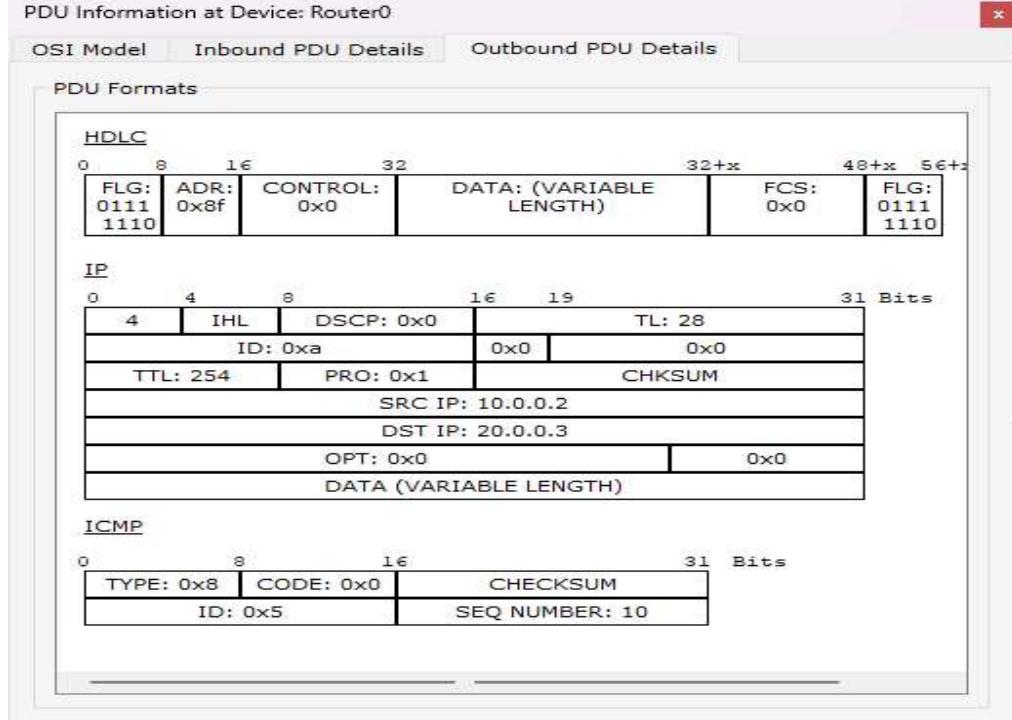
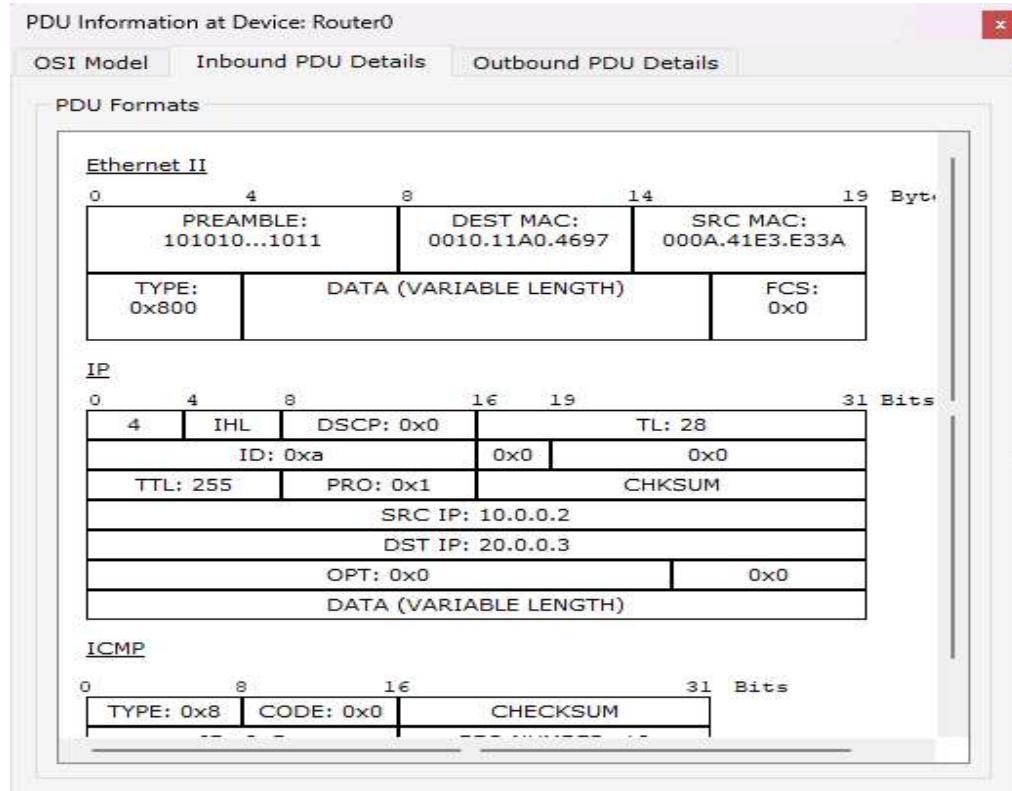
ping statistics for 20.0.0.2  
packet sent = 4 received = 3 lost = 1 (25% loss)

> ping 30.0.0.2

ping statistics for 30.0.0.2  
packet sent = 4 received = 3 lost = 1 (25% loss).

### Screen Shots:





## Program 8

**Aim:** Configure OSPF routing protocol .

**Topology , Procedure and Observation:**

| LAB-4         |                                                                                                                                                                                                                                                                                                        |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| → Aim:        | Configure OSPF routing protocol.                                                                                                                                                                                                                                                                       |
| → Topology:   |                                                                                                                                                                                                                                                                                                        |
| → Procedure:  | <ol style="list-style-type: none"> <li>1) Connect three routers to each other and first router to one PC(PC0) and last router to 1 PC (PC1)</li> <li>2) Configure the ip addresses to all interfaces</li> </ol>                                                                                        |
| For Router 0: | <pre>R(config)# inteface fastethernet 0/0 R(config-if)# ip address 10.0.0.1 255.0.0.0 R(config-if)# no shut R(config-if)# exit</pre><br><pre>R(config)# inteface serial 2/0 R(config-if)# ip address 20.0.0.1 255.0.0.0 R(config-if)# encapsulation ppp R(config-if)# no shut R(config-if)# exit</pre> |

|                                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| R(config-if)# : clock rate 64000                                                                                                                                          |
| R(config-if)# : no shut                                                                                                                                                   |
| R(config-if)# : exit                                                                                                                                                      |
| In Router 1 :                                                                                                                                                             |
| <pre>R(config)# inteface serial 3/0 R(config-if)# ip address 30.0.0.1 255.0.0.0 "      : encapsulation ppp "      : clock rate 64000 "      : no shut "      : exit</pre> |
| <pre>R(config)# inteface serial 2/0 R(config-if)# ip address 30.0.0.2 255.0.0.0 "      : encapsulation ppp "      : no shut "      : exit</pre>                           |
| In Router 2 :                                                                                                                                                             |
| <pre>R(config)# inteface fastethernet 0/0 R(config-if)# ip address 40.0.0.1 255.0.0.0 "      : no shut "      : exit</pre>                                                |

4) Router 0:

R(config)# : router ospf 1  
R(config-router) : router-id 1.1.1.1  
: network 10.0.0.0 0.255.255.255 area 0  
: network 10.0.0.0 0.255.255.255 area 0  
: exit

Router 1:

R(config)# : router ospf 1  
: router-id 2.2.2.2  
: network 20.0.0.0 0.255.255.255 area 0  
: network 30.0.0.0 0.255.255.255 area 0

Router 2:

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

5) Router 0:

R(config-if) : interface loopback 0  
: ip add 172.16.1.252 255.255.0.0  
: no shut

Router 1:

R(config-if) : interface loopback 0  
: ip add 172.16.1.253 255.255.0.0

Router 2:

R(config): interface loopback0  
           : ip add 172.16.1.254 255.255.255.0  
           : no shut

5) Router 0:

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

Router 1:

R(config): router ospf 1  
           : area 1 virtual-link 1.1.1.1  
           : no shut

Router 2:

R(config): exit shout ip route

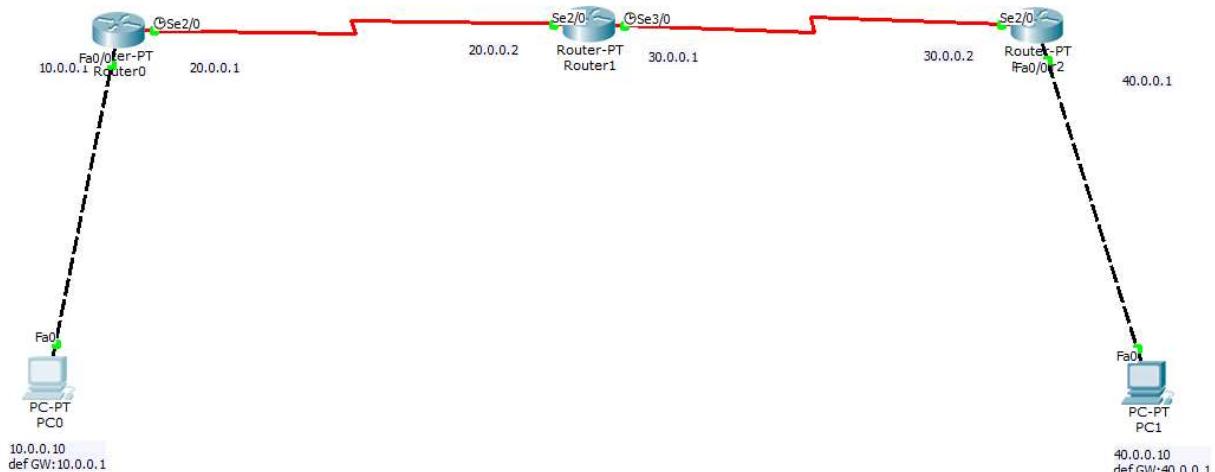
7) Observation:

ping 40.0.0.10  
 Request timed out  
 reply from 40.0.0.10 by test32 time 0ms

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| u | u | u | u | u | u | u | u |
| u | u | u | u | u | u | u | u |

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

## Screen Shots:



PC0

Physical Config Desktop Custom Interface

**Command Prompt**

```

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=8ms 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 = 7ms, Maximum = 8ms, Average = 7ms

PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

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

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

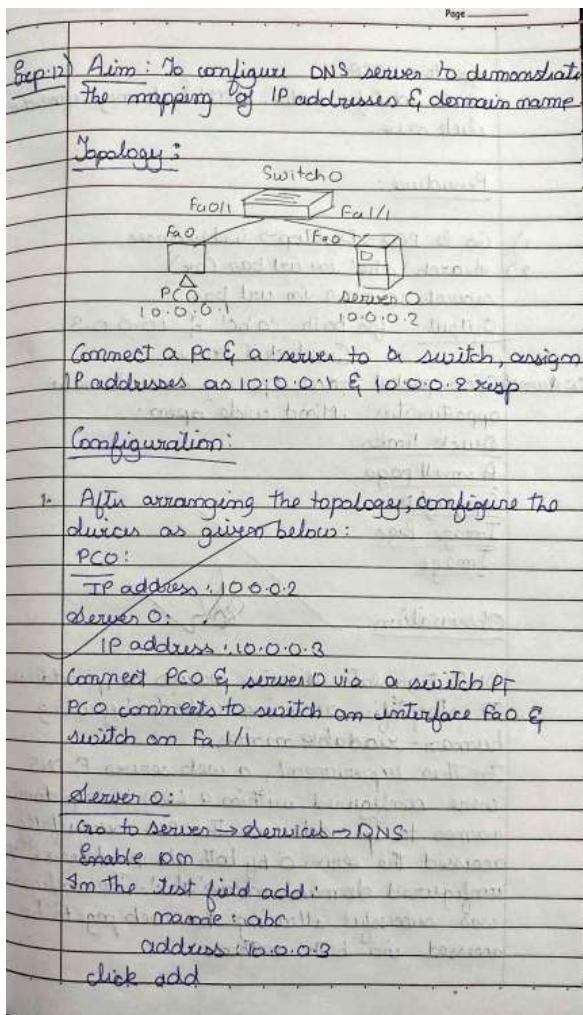
PC>

```

## Program 9

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

**Topology , Procedure and Observation:**



Procedure:

- Go to PC0 → Desktop → Web browser
- search 'abc' in url bar (or)
- search 10.0.0.3 in url bar

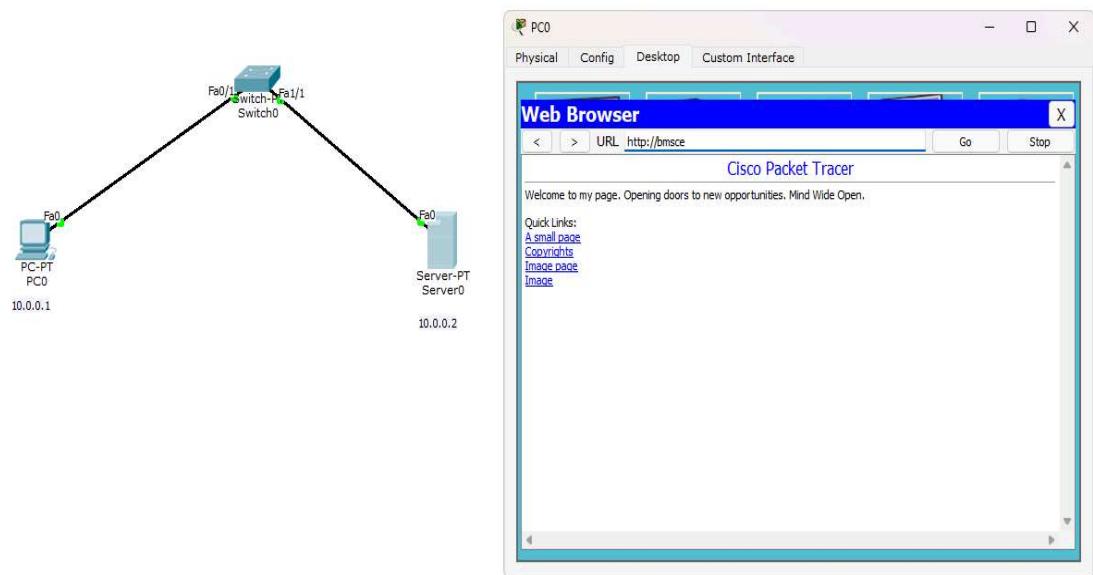
Output: for both 'abc' & 10.0.0.3

Observation:

DNS translates domain names to ip addresses. It simplifies accessing websites by using human-readable names.

In This experiment, a web server & DNS were configured within a LAN to map domain names to ip addresses. The PC successfully accessed the server by both its ip address & the reconfigured domain name 'abc'. The configuration was successful allowing the web page to be accessed via both methods.

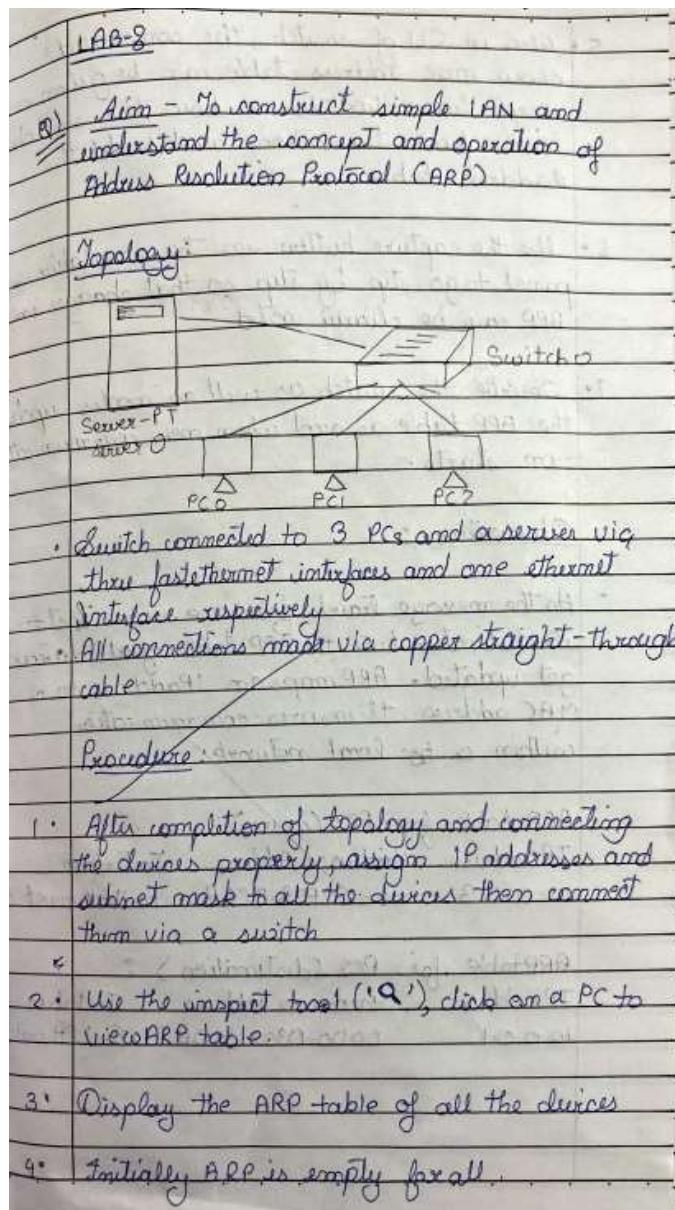
## Screen Shots:



## Program 10

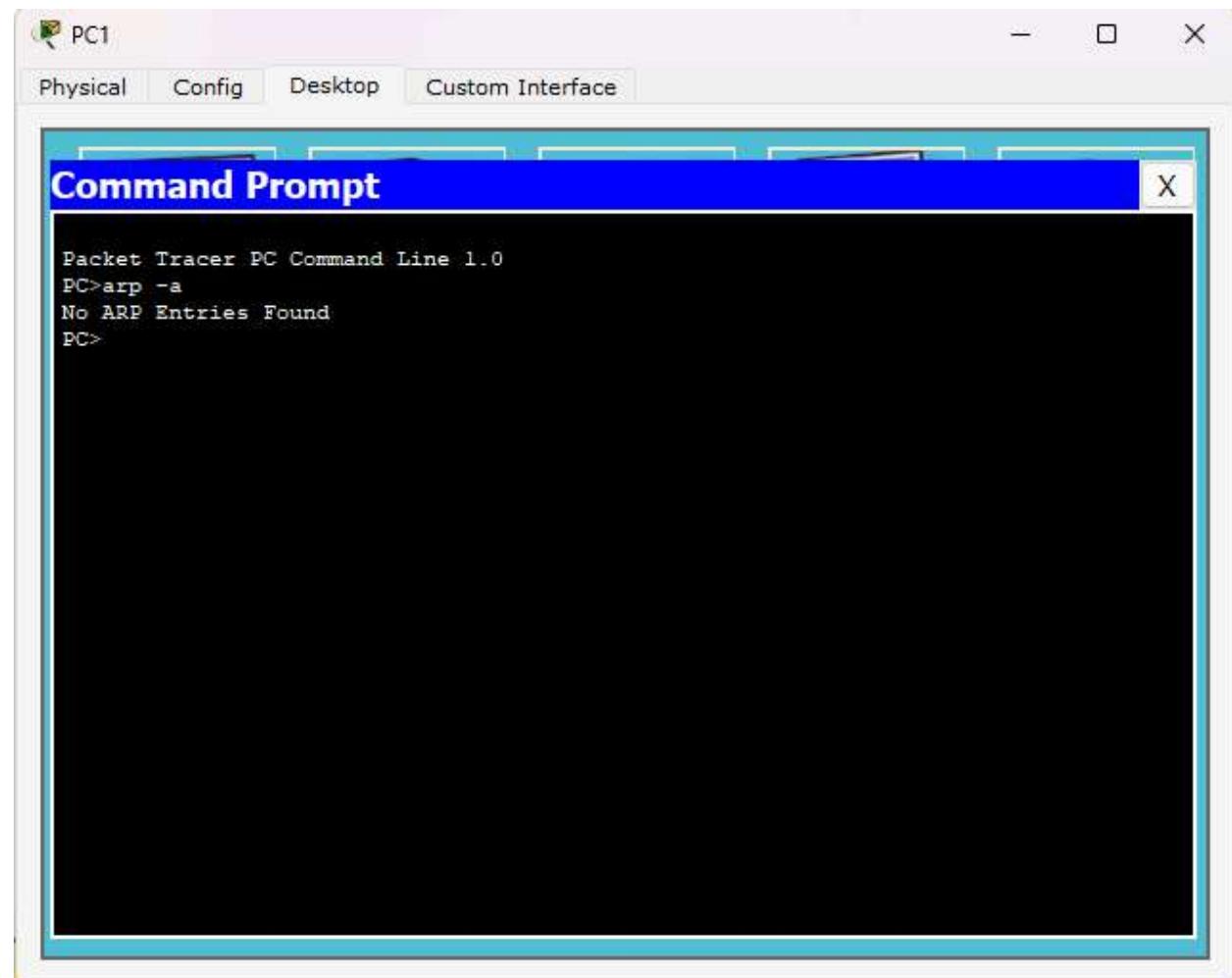
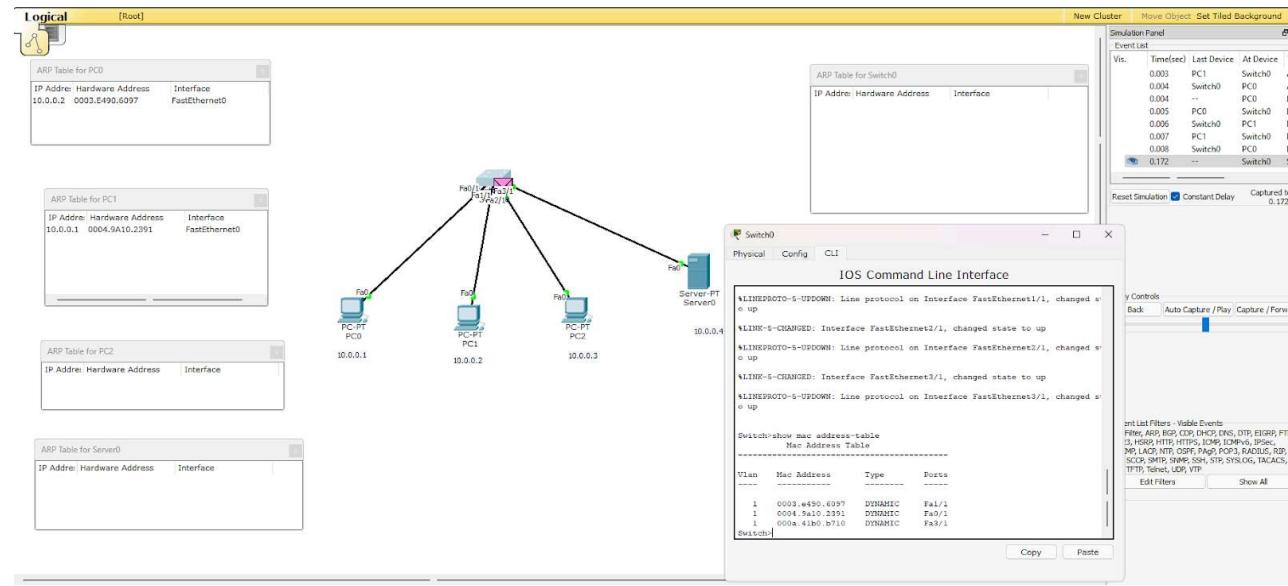
**Aim:** To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

**Topology , Procedure and Observation:**



| 5.                                                                                                                                                                                                                                                         | Also in CLI of switch, the command = show mac Address-table can be given on every transaction to see how the switch learns from transactions and build the address table. |                 |                  |           |          |                   |                 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------------------|-----------|----------|-------------------|-----------------|
| 6.                                                                                                                                                                                                                                                         | Use the capture button in the simulation panel to go step by step so that changes in ARP can be clearly noted                                                             |                 |                  |           |          |                   |                 |
| 7.                                                                                                                                                                                                                                                         | Observe the switch as well as nodes upbit the ARP table as and when new communication starts.                                                                             |                 |                  |           |          |                   |                 |
| <u>Observation :</u>                                                                                                                                                                                                                                       |                                                                                                                                                                           |                 |                  |           |          |                   |                 |
| <ul style="list-style-type: none"> <li>As the message travels from one source host to its destination host, the ARP table of all devices got updated. ARP maps an IP address to a MAC address. It ensures communication within a local network.</li> </ul> |                                                                                                                                                                           |                 |                  |           |          |                   |                 |
| ARP table for PC0 (source) :                                                                                                                                                                                                                               |                                                                                                                                                                           |                 |                  |           |          |                   |                 |
| <table border="1"> <thead> <tr> <th>IP address</th> <th>Hardware Address</th> <th>Interface</th> </tr> </thead> <tbody> <tr> <td>10.0.0.3</td> <td>00:60:7F:09:9C:B8</td> <td>Fast Ethernet 0</td> </tr> </tbody> </table>                                 |                                                                                                                                                                           | IP address      | Hardware Address | Interface | 10.0.0.3 | 00:60:7F:09:9C:B8 | Fast Ethernet 0 |
| IP address                                                                                                                                                                                                                                                 | Hardware Address                                                                                                                                                          | Interface       |                  |           |          |                   |                 |
| 10.0.0.3                                                                                                                                                                                                                                                   | 00:60:7F:09:9C:B8                                                                                                                                                         | Fast Ethernet 0 |                  |           |          |                   |                 |
| ARP table for PC1 (destination) :                                                                                                                                                                                                                          |                                                                                                                                                                           |                 |                  |           |          |                   |                 |
| <table border="1"> <thead> <tr> <th>IP address</th> <th>Hardware Address</th> <th>Interface</th> </tr> </thead> <tbody> <tr> <td>10.0.0.1</td> <td>00:D0:D3:02:36:0B</td> <td>Fast Ethernet 0</td> </tr> </tbody> </table>                                 |                                                                                                                                                                           | IP address      | Hardware Address | Interface | 10.0.0.1 | 00:D0:D3:02:36:0B | Fast Ethernet 0 |
| IP address                                                                                                                                                                                                                                                 | Hardware Address                                                                                                                                                          | Interface       |                  |           |          |                   |                 |
| 10.0.0.1                                                                                                                                                                                                                                                   | 00:D0:D3:02:36:0B                                                                                                                                                         | Fast Ethernet 0 |                  |           |          |                   |                 |

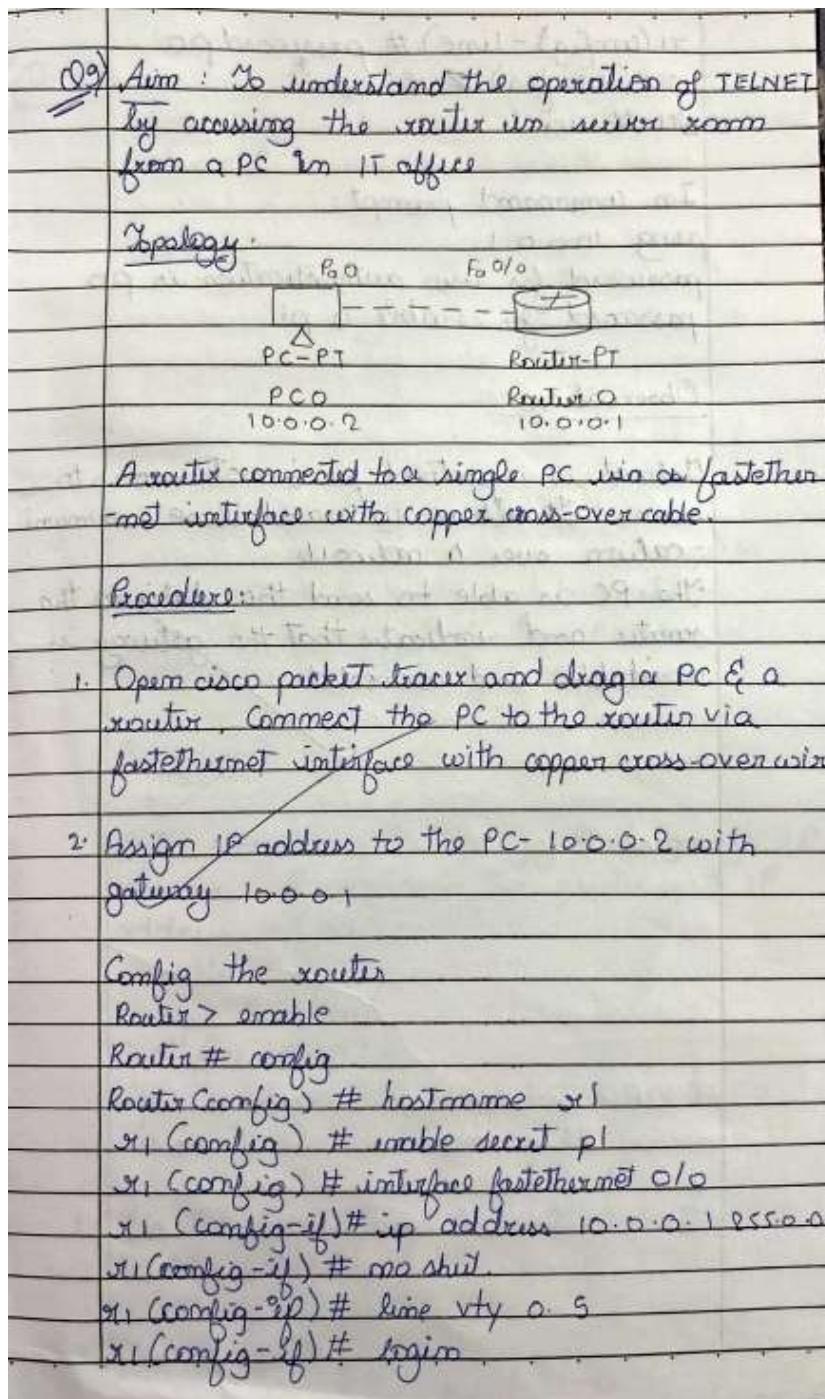
## Screen Shots:

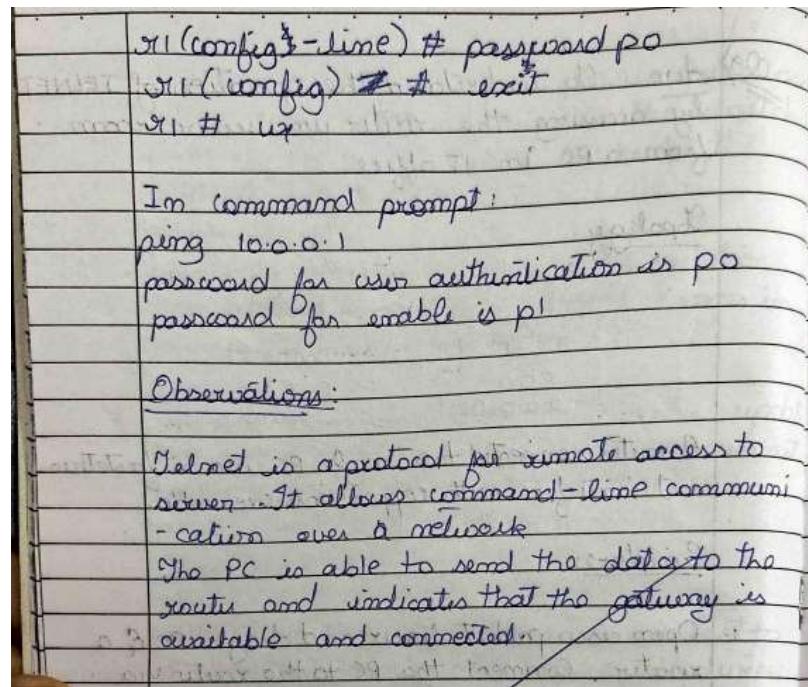


## Program 11

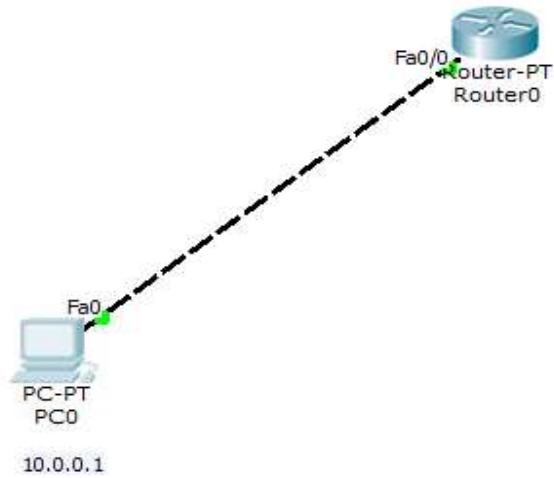
**Aim:** To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.

### **Topology , Procedure and Observation:**





### Screen Shots:



## Command Prompt

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

Pinging 10.0.0.2 with 32 bytes of data:

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

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

PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open

User Access Verification

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

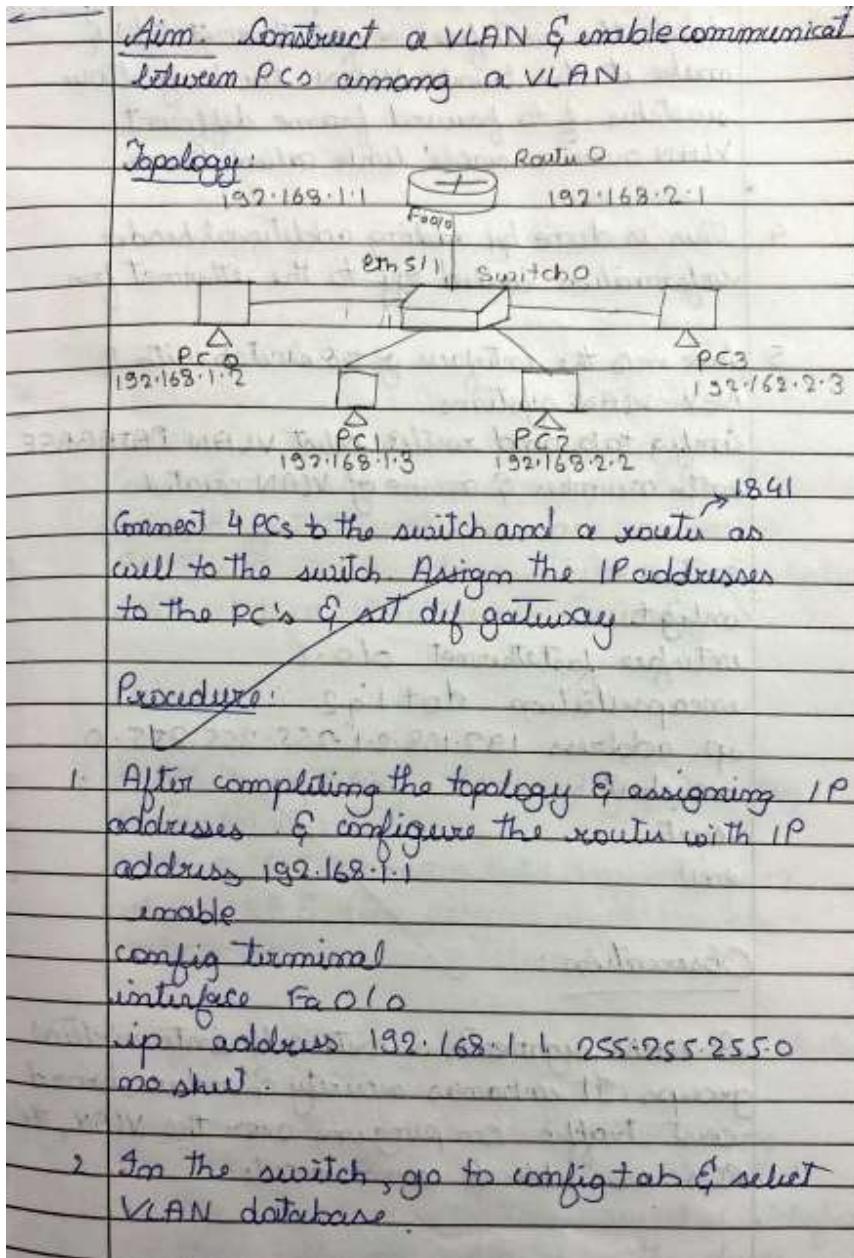
Gateway of last resort is not set

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

## Program 12

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

**Topology , Procedure and Observation:**

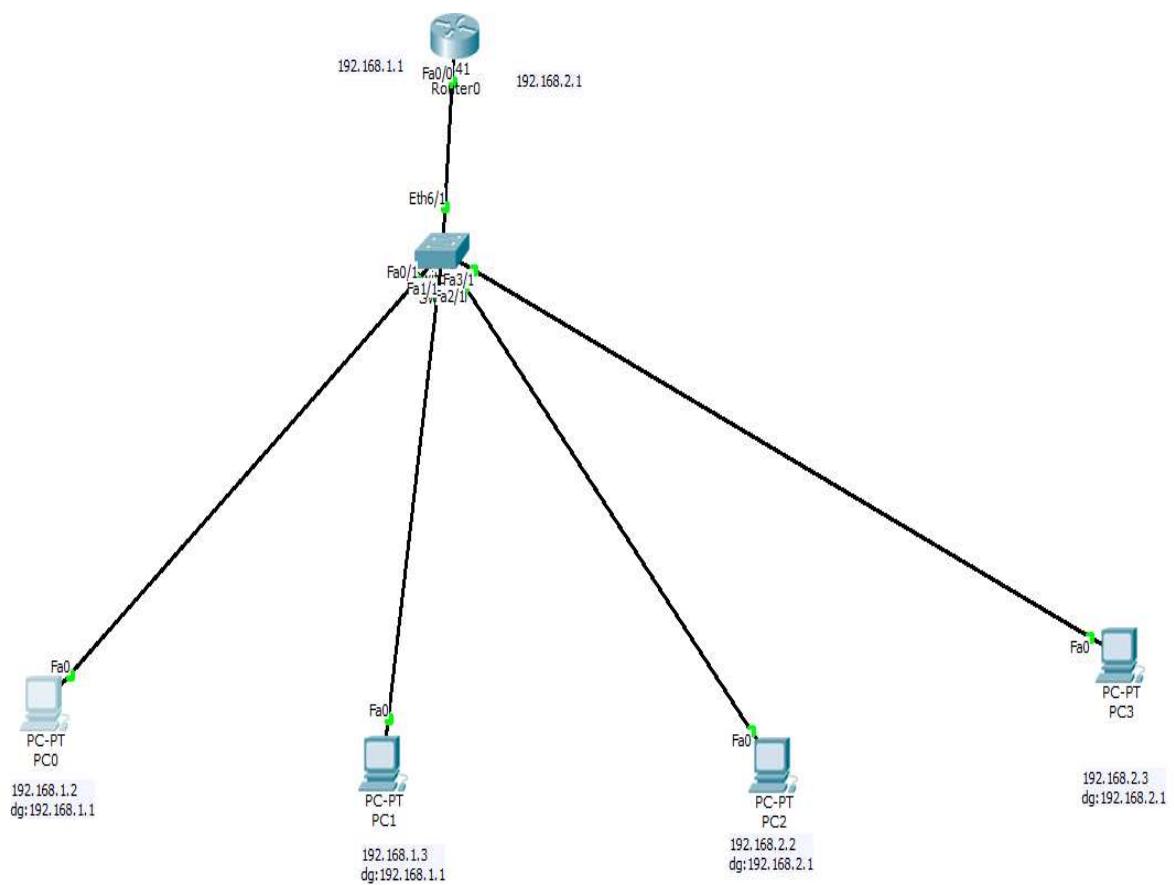


3. Set the VLAN number & VLAN name.  
 Select the interface i.e. fastethernet s1/1/9  
 make it the trunk VLAN. Trunking allows  
 switches to forward frame different  
 VLAN over a single link called trunk.
4. This is done by adding additional header  
 information called tag to the ethernet frame
5. Look into the interface of the switch with 2  
 NEW VLAN systems  
 Config tab and you'll select VLAN DATABASE  
 with number & name of VLAN created
- ```

exit
config
interface fastethernet 0/0.1
encapsulation dot1q 2
ip address 192.168.2.1 255.255.255.0
no shutdown
exit
exit
  
```

Observation:

A VLAN segment or network into virtual groups. It enhances security & reduces broadcast traffic. On pinging over the VLAN, the PCs are able to communicate with each other.

Screen Shots:

Command Prompt

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

Pinging 192.168.2.2 with 32 bytes of data:

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

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

PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

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

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

PC>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

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

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

PC>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

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

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

PC>
```

Program 13

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

Topology , Procedure and Observation:

Exp-11) Aim: To construct WLAN and make nodes communicate wirelessly.

Topology :

```

    graph LR
        Router[Router 10.0.0.2] --- FE[Fast Ethernet]
        FE --- Switch[Switch]
        Switch --- AP[Access Point 10.0.0.3]
        Switch --- PC1[PC 10.0.0.1]
        AP --- Laptop[Laptop 10.0.0.4]
    
```

Connect a router and access point to a switch through fastethernet interface connect a PC & set its IP address Take a PC & a laptop and set their IP addresses (wireless)

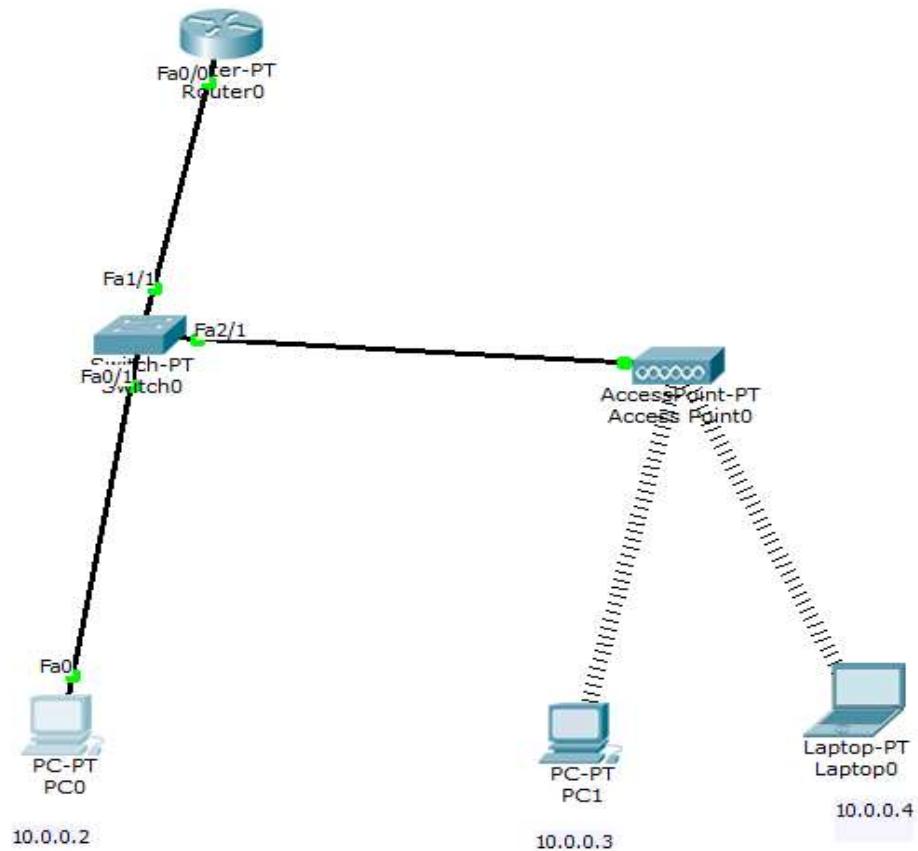
Procedure:

1. After completing the topology, config Access Point:
Port1 → SSID Name → Enter any name → select WEP & give any 10 digit hex key - 1234567890
2. Configure PC1 & laptop with wireless station
3. Turn off the device. Drag the existing PT-HOST-NM-LAM to the component listed in the LH. Drag WMP300N wireless interface to the empty port, turn on the device

4. In the config tab, a new wireless interface would have been added. Now Configure SSID, WEP, WEP key, IP addresses & gateway to the device.
5. Ping from every device to every other device and see the results.

Observation :

WLAN enables wireless m/w comm. It uses radio waves for connectivity. WLAN connects devices wirelessly within a local area. It eliminates the need for physical cables.

Screen Shots:

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=22ms TTL=128
Reply from 10.0.0.3: bytes=32 time=6ms TTL=128
Reply from 10.0.0.3: bytes=32 time=3ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 22ms, Average = 9ms
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=5ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms 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 = 5ms, Maximum = 19ms, Average = 9ms
PC>

PART-B

Program 14

Write a program for congestion control using Leaky bucket algorithm.

Luby Bucket Program

Q Write a program for congestion control using leaky bucket algorithm;

```

program: lbucket.cc
#include < stdio.h >
#include < stdlib.h >
#include < unistd.h >
#define NOF_PACKETS
int rand (int a) {
    int rm=(random() % 16) / a;
    return rm == 0 ? 1 : rm;
}
#include < stdlib.h >
long int random (void);
int main ()
{
    int packet_sz[NOF_PACKETS], i, clk, b_size,
        o_rate, p_sg, rm=0, p_sg, p_time_op;
    for(j=0; j<NOF_PACKETS; j++)
        packet_sz[j]=random() % 16;
    for(i=0; i<NOF_PACKETS; i++)
        printf ("\\npacket[%d]: %d bytes \\t", i, packet_sz[i]);
    printf ("\\nEnter the output rate: ");
    scanf ("%d", &o_rate);
    printf ("\\nEnter the Bucket Size: ");
    scanf ("%d", &b_size);
    for(i=0; i<NOF_PACKETS; i++)
    {
        if (packet_sz[i] > o_rate)
            packet_sz[i] = o_rate;
        else
            o_rate -= packet_sz[i];
        if (o_rate <= 0)
            break;
    }
    for(i=0; i<NOF_PACKETS; i++)
        printf ("\\npacket[%d]: %d bytes \\t", i, packet_sz[i]);
}

```

```

if (packet_sg[i] + p_sg_rm) > b_size)
if (packet_sg[i] > b_size) {
    packet_sg[i] = b_size;
    cout << "Incoming packet size (" << d_bytes << " bytes) is
    Greater than bucket capacity (" << d_bytes << " - PACKED
    REJECTED", packet_sg[i], b_size);
} else
    cout << "n Bucket capacity exceeded =
    PACKETS REJECTED!!";
else {
    p_sg_rm += packet_sg[i];
    cout << "n Incoming packet size : " << d;
    packet_sg[i];
    cout << "n Bytes remaining to Transmit : " << d;
    p_sg_rm;
    // p_time = random() * 10;
    // cout << "n Time left for transmission : " << d
    // units, p_time;
    // for (clk = 10; clk <= p_time; clk += 10)
    while (p_sg_rm > 0)
        if (p_sg_rm >= max_f)
            sleep(1);
        if (p_sg_rm <= 0.5 * max_f)
            op = p_sg_rm, p_sg_rm = 0;
        else
            op = o_rate, p_sg_rm -= o_rate;
    cout << "n Packt of size " << d << " Transmitted ";
    cout << "n Bytes Remaining to Transmit : " << d;
    p_sg_rm;
}

```

Page _____

```
else
{
    printf ("In No packets to transmit!");
}
}
?
?
```

Output :

packet[0]: 83 bytes
packet[1]: 86 bytes
packet[2]: 77 bytes
packet[3]: 15 bytes
packet[4]: 93 bytes
Enter the output rate : 30
Enter the Bucket size : 85

Incoming Packet size : 83
Bytes remaining to Transmit : 83
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 53
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 23
Packet of size 23 Transmitted --- Bytes Remaining to Transmit : 0

Incoming packet size (86 bytes) is Greater than bucket capacity (85 bytes) - PACKET REJECTED

Incoming packet size : 77
Bytes remaining to Transmit : 77
Packet of size 30 Transmitted --- Bytes Remaining

	To Transmit: 47
	Packet of size 30 Transmitted---Bytes Remaining
	To Transmit: 17
	Packet of size 17 Transmitted---Bytes Remaining
	To Transmit: 0
	Incoming Packet size: 15
	Bytes remaining to Transmit: 15
	Packet of size 15 Transmitted---Bytes Remaining
	To Transmit: 0

```

Output Clear

packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:30
Enter the Bucket Size:85

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 30 Transmitted---Bytes Remaining to Transmit: 53
Packet of size 30 Transmitted---Bytes Remaining to Transmit: 23
Packet of size 23 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 86
Incoming packet size (86bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 30 Transmitted---Bytes Remaining to Transmit: 47
Packet of size 30 Transmitted---Bytes Remaining to Transmit: 17
Packet of size 17 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 93
Incoming packet size (93bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

```

Program 15

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

Code and Output:

①	<p>using TCP/IP sockets, write a client-server program to make a client sending the file name and the server to send back the contents of the requested file if present.</p>
→	<pre> <u>Client TCP.py</u> 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('From Server:', filecontents) clientSocket.close() </pre>
→	<pre> <u>Server TCP.py</u> from socket import * serverName = '127.0.0.1' serverPort = 12000 serverSocket = socket(AF_INET, SOCK_STREAM) serverSocket.bind((serverName, serverPort)) serverSocket.listen(1) 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) </pre>

Date / /
Page / /

```

connectionSocket.send(l.encode())
print('In sent contents of ' + sentence)
file.close()
connectionSocket.close()

Output

server → The source is ready to receive
sent contents of server.py
client ⇒ enter file name : Server.py
From server: ntpd[1]: listening on port 123
from socket import *
serverName = "127.0.0.1"
serverPort = 12300
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, address = 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()

```

```
PS C:\Users\I AM HP\CN> python ClientTCP.py
Enter file name :ServerTCP.py
From Server:
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()
PS C:\Users\I AM HP\CN> 
```

DEBUG CONSOLE PROBLEMS TERMINAL PORTS

```
PS C:\Users\I AM HP\CN> python ServerTCP.py
The server is ready to receive
Sent contents ofServerTCP.py
The server is ready to receive

```

Program 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 and Output:

<p>Q 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.</p>	<p>→ ClientUDP.py</p> <pre>from socket import * serverName = "127.0.0.1" serverPort = 12000 clientSocket = socket(AF_INET, SOCK_DGRAM) sentence = input("Enter file name: ") clientSocket.sendto(sentence.encode("utf-8"), (serverName, serverPort)) fileContent, serverAddress = clientSocket.recvfrom(2048) print("Reply from server: ") print(fileContent.decode("utf-8")) # for i in range(len(fileContent)): # print(fileContent[i], end="") clientSocket.close() clientSocket.close()</pre> <p>ServerUDP.py</p> <pre>from socket import * serverPort = 12000 serverSocket = socket(AF_INET, SOCK_DGRAM) serverSocket.bind(("127.0.0.1", serverPort)) print("The server is ready to receive") while True: sentence, clientAddress = serverSocket.recvfrom(2048)</pre>
---	--

```

sentence = sentence.decode("utf-8")
file = open(sentence, "r")
com = file.read(2048)
serverSocket.sendto(com, clientAddress)

print('I'm sending contents of', end=' ')
print(sentence)
# for i in sentence:
#     print(i, end=' ')
file.close()

```

Output

```

server → The server is ready to receive
sent contents of Server.py
The source is ready to receive
client → client.py
Enter file name: Server.py
Reply from server:
from socket import*
serverPort=12000
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(("127.0.0.1",serverPort))
while 1:
    print("The server is ready to receive")
    sentence, clientAddress=serverSocket.recvfrom(2048)
    sentence=sentence.decode("utf-8")
    file=open(sentence,"r")
    l=file.read(2048),

```

```
serverSocket.sendto(l.bytes(l, "Utf-8"),  
clientAddress)  
print('Im sent contents of ', end=' ')  
print(sentence)  
#for i in sentence:  
#    print(str(i), end=' ')  
file.close()
```

DEBUG CONSOLE PROBLEMS TERMINAL PORTS

- PS C:\Users\I AM HP\CN> **python** ServerUDP.py
The server is ready to receive
Sent contents of ServerUDP.py
The server is ready to receive
█

- PS C:\Users\I AM HP\CN> **python 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))
while 1:
    print("The server is ready to receive")
    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 "+sentence)
    file.close()
```

- PS C:\Users\I AM HP\CN> □

Program 17:

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

```

Q Write a program for error detection
code using CRC-CCITT (16 bits)

def xor(a, b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)

def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0 : pick]
    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]
        else:
            tmp = xor('0' * pick, tmp) +
            dividend[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)
    checkword = tmp
    return checkword

```

```

def encode(data, key):
    key_len = len(key)
    append_data = data + '0' * (key_len - 1)
    remainder = mod2div(append_data, key)
    codeword = data + remainder
    print(f"Encoded Data : {codeword}")
    return codeword

def decode(data, key):
    remainder = mod2div(data, key)
    print(f"Remainder after decoding : {remainder}")
    if '1' not in remainder:
        print("No error detected in received data")
    else:
        print("Error detected in received data")

if __name__ == "__main__":
    data = input("Enter the data bits: ")
    key = input("Enter the key(divisor): ")
    encoded_data = encode(data, key)
    print("Decoding the encoded data...")
    decode(encoded_data, key)

```

Output

Enter the data bits: 100100100100100

Enter the key (divisor): 10101

Encoded Data: 1001001001001000001

Decoding the encoded data...

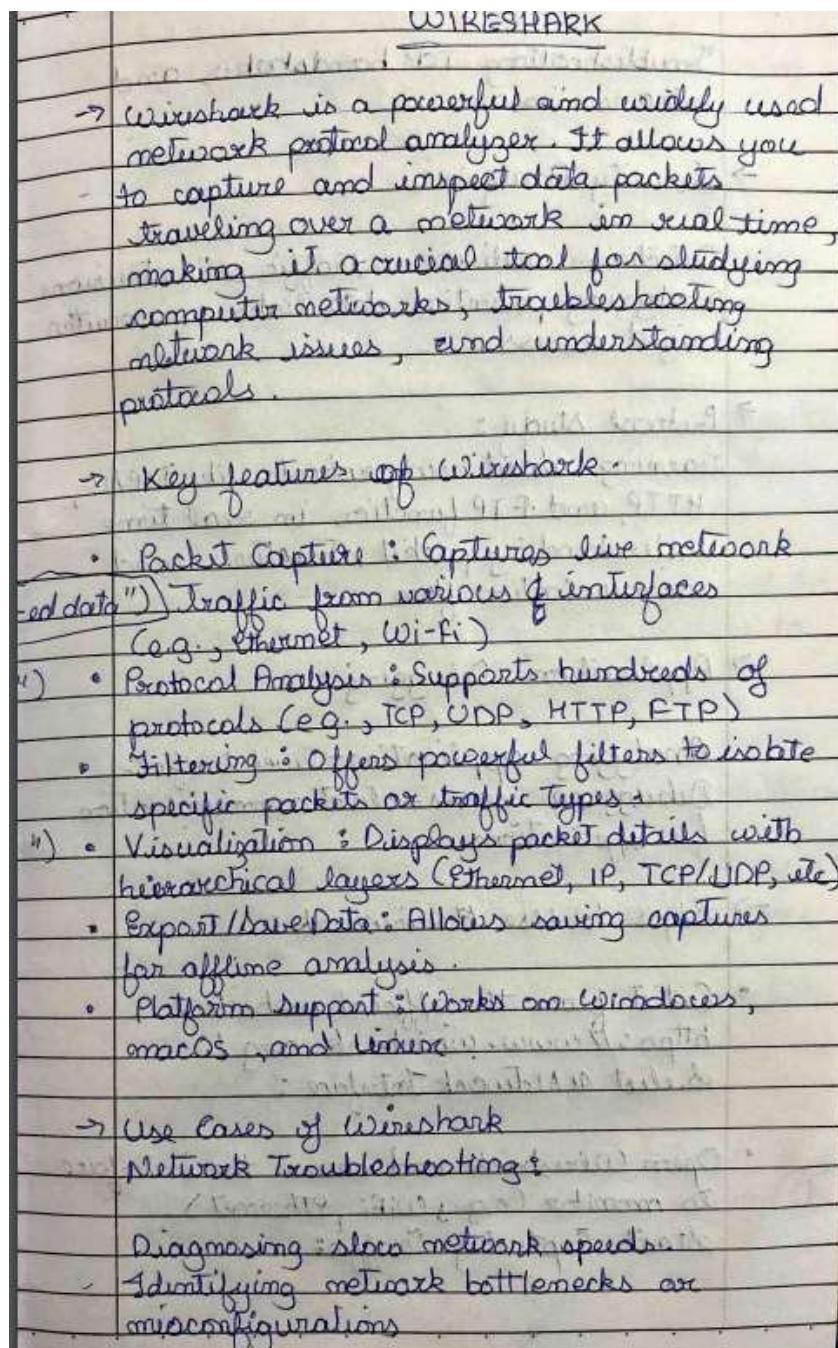
Remainder after decoding: 0000

No error detected in received data

== Code Execution Successful ==

Program 18:

Tool exploration : wireshark



Troubleshooting TCP handshakes and retransmissions	
→ Security Analysis:	Detecting malicious traffic or intrusions Analyzing potential data leaks or unauthorized access
→ Protocol Study:	Learning about how protocols like TCP/IP, HTTP, and FTP function in real time Understanding packet structures and communication flow
→ Application Debugging:	Analyzing application-level traffic Debugging server-client communication for applications
→ Steps to Use Wireshark:	<ul style="list-style-type: none">Get it from the official website https://www.wireshark.org/ and select a Network Interface:Open Wireshark and choose the interface to monitor (e.g., WiFi, Ethernet).Start Capturing Traffic.

	<ul style="list-style-type: none"> Click the start button to begin capturing packets on the selected interface. <p>Analyze Captured Data :</p> <ul style="list-style-type: none"> View the real-time data packets. use the filter bar to narrow down traffic, e.g. http, ip.addr == 192.168.1.1. Stop and save the capture. <p>Stop the capture by clicking the stop button.</p> <p>Save the captured data for further analysis.</p> <p>Common filters</p> <p>https : Show only HTTP traffic.</p> <p>tcp.port == 80 : Show traffic on TCP port 80.</p> <p>ip.addr == 192.168.1.1 : Show packets to or from a specific IP address.</p> <p>udp : Show only UDP traffic.</p> <p>Example scenarios.</p> <p>Capture UDP Client -> Server Communication</p> <p>Use Wireshark to observe the traffic b/w the UDP client and server program you created.</p> <p>Filter by udp to see the relevant packets.</p> <p>Inspect HTTP Traffic :</p> <p>Use Wireshark to monitor HTTP GET and POST requests when accessing a website.</p> <p>Study DNS Queries :</p>
--	---

