

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB RECORD

### Computer Network Lab (23CS5PCCON)

*Submitted by*

**Hitish Rao P (1BM23CS116)**

*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 2025-26 (even)**

# B.M.S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

## Department of Computer Science and Engineering



### CERTIFICATE

This is to certify that the Lab work entitled “ Computer Network (23CS5PCCON)” carried out by **Hitish Rao P (1BM23CS116)**, who is bonafide student of **B.M.S. College of Engineering**. It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Rashmi H Assistant Professor Department of CSE, BMSCE	Dr. Selva Kumar Professor & HOD Department of CSE, BMSCE
---	--

# Index

<b>Sl. No.</b>	<b>Date</b>	<b>Experiment Title</b>	<b>Page No.</b>
1	13-08-2025	Simple PDU from source to destination using hub and switch as connecting devices.	4-8
2	17-09-2025	Default route and static route to the Router	9-13
3	03-09-2025	DHCP within a LAN and outside LAN	14-15
4	10-09-2025	Web Server, DNS within a LAN	16-18
5	08-10-2025	Operation of TELNET to access the router in server room from a PC in IT office	19-21
6	19-11-2025	RIP routing Protocol in Routers	22-23
7	15-10-2025	VLAN to make the PCs communicate among a VLAN	24-25
8	15-10-2025	WLAN to make the nodes communicate wirelessly	26-27
9	19-11-2025	Simple LAN to understand the concept and operation of ARP	28-29
10	08-10-2025	OSPF routing protocol	30-31
11	15-10-2025	TTL/ Life of a Packet	32-33
12	15-10-2026	Ping responses, destination unreachable, request timed out, reply	34-36
13	29-10-2025	Congestion control using Leaky bucket algorithm	37-38
14	29-10-2025	Error detecting code using CRC-CCITT	39-40
15	12-11-2025	TCP File Request–Response Using Client–Server Socket Program	41-42
16	12-11-2025	UDP File Request–Response Using Client–Server Socket Program	43-44

Github Link: <https://github.com/Hitish-Rao-P/CN-1BM23CS116>

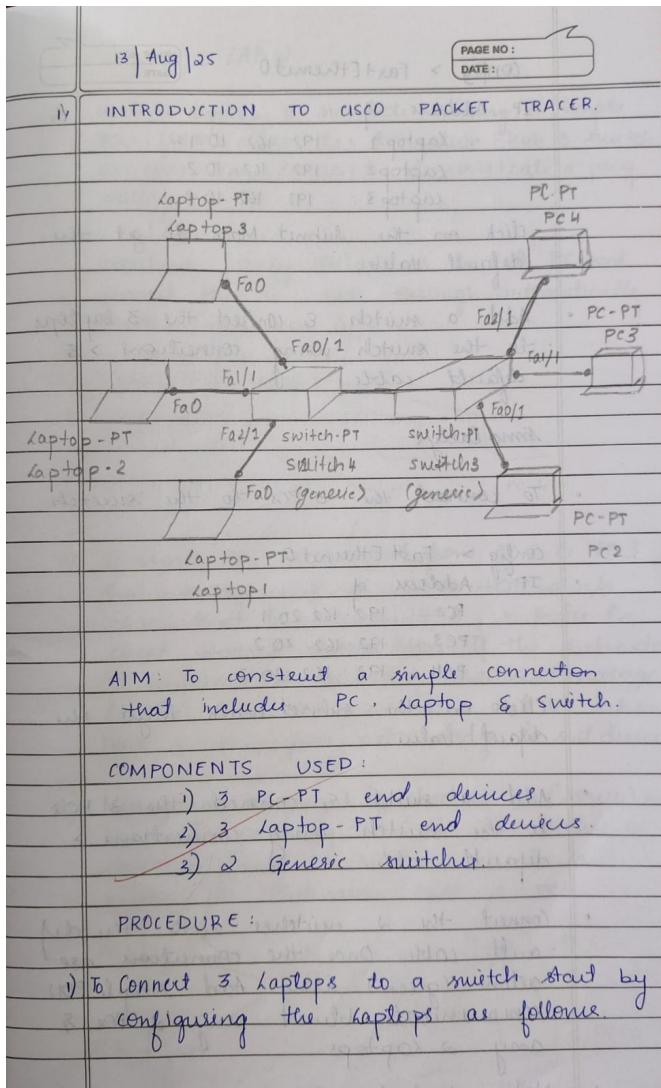
## CYCLE 1:

### Program 1

Aim of the program:

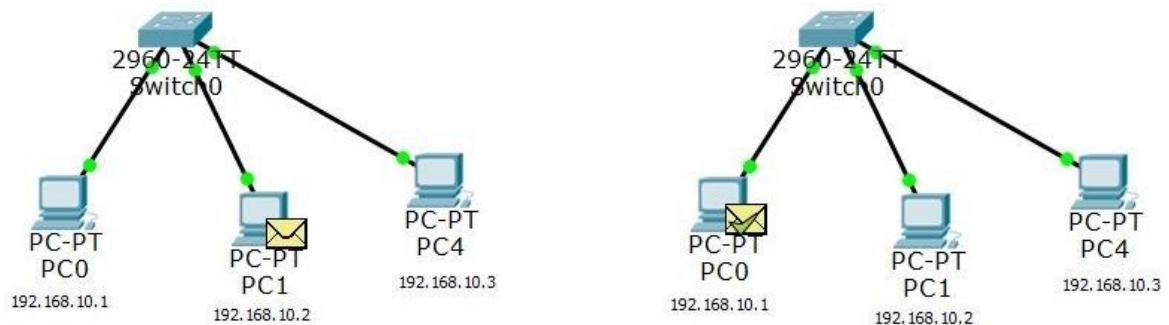
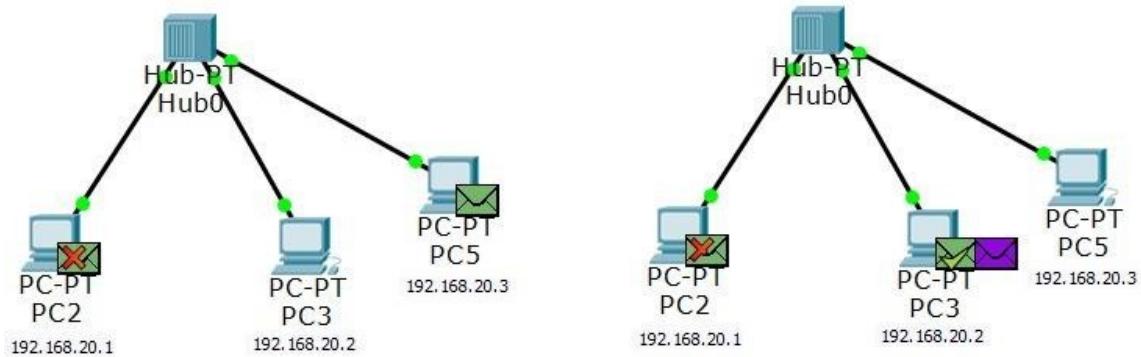
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Procedure and topology:

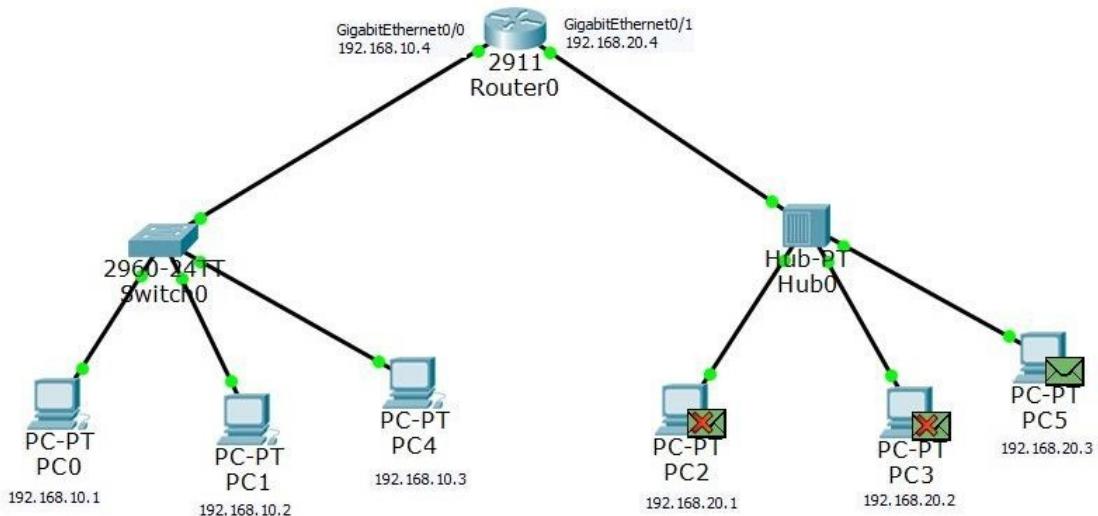


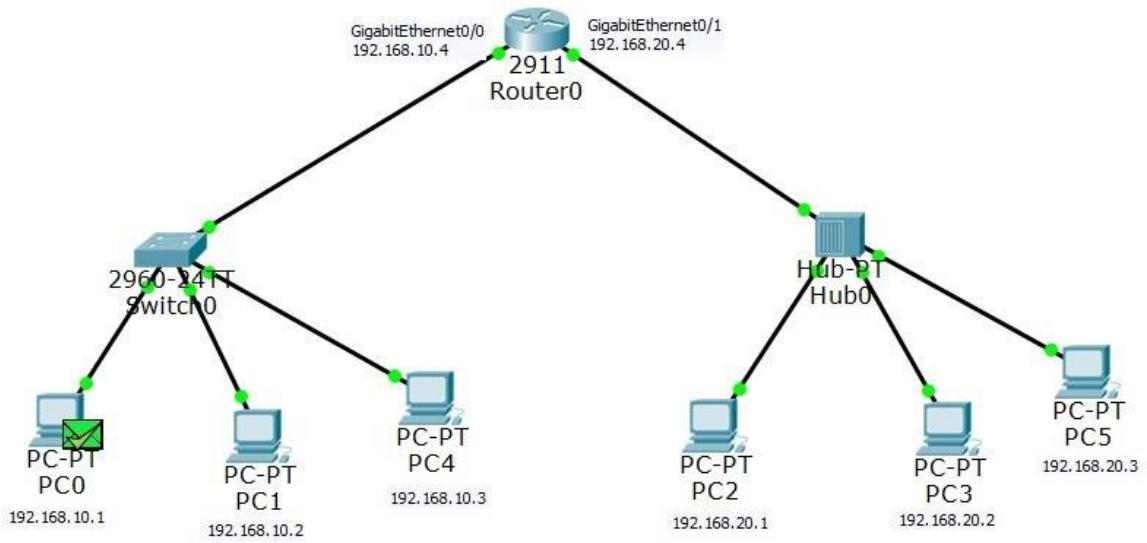
- PAGE NO :  
DATE :
- config > Fast Ethernet 0
- IP Address of  
 Laptop 1 : 192.162.10.1  
 Laptop 2 : 192.162.10.2  
 Laptop 3 : 192.162.10.3  
 Click on the subnet mask to get the default value.
  - Add a switch & connect the 3 laptops to the switch using connections > & default cable (mark like red & yellow).
  - Similarly,
  - To connect the 3 PCs to the switch
- config > Fast Ethernet 0
- IP Address of  
 PC2 : 192.162.20.1  
 PC3 : 192.162.20.2  
 PC4 : 192.162.20.3  
 Click on the subnet mask to get the default value.
- Add a switch & connect the 3 PCs to the switch using connections > & default cable.
  - Connect the 2 switches with the default cable. Once the connections are active (green) using Add simple PDU communicate between any 2 PCs & any 2 laptops.
  - Notice it fails between a laptop & PC.

Screenshots/ Output:



Updated topology





Observation:

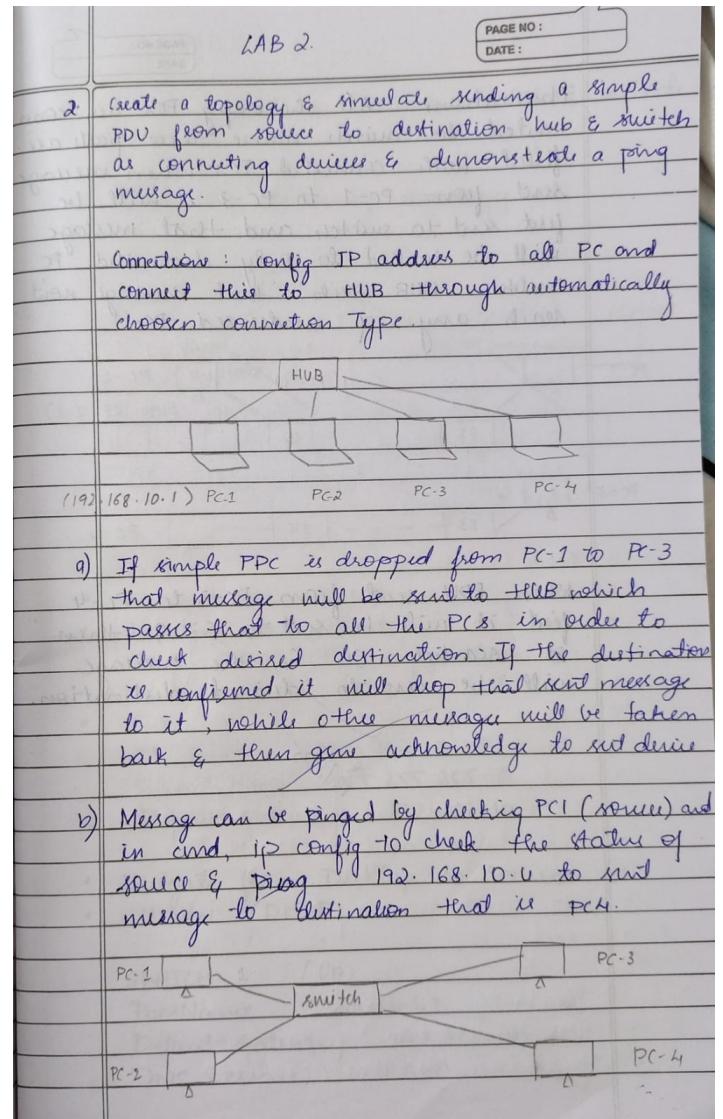
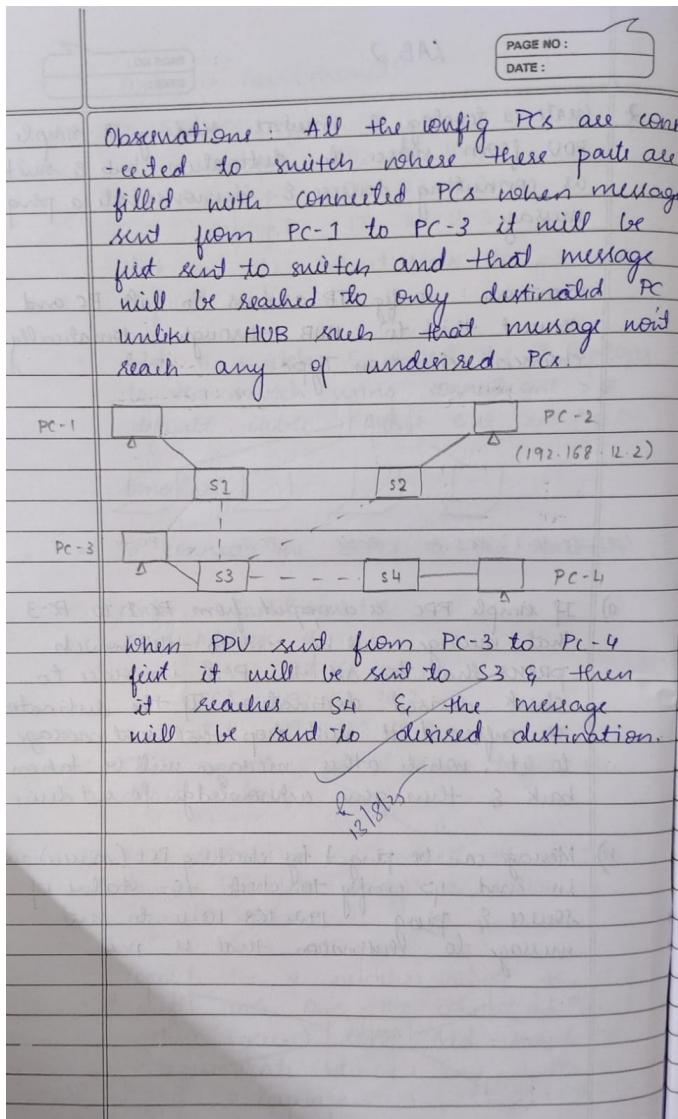
- In the hub-based topology, the PDU was broadcast to all ports, while the switch forwarded the PDU only to the destination MAC after learning addresses from incoming frames.
- Successful ICMP echo and echo-reply messages confirm that both devices enabled connectivity, with the switch demonstrating selective unicast forwarding and reduced unnecessary traffic.

## **Program 2**

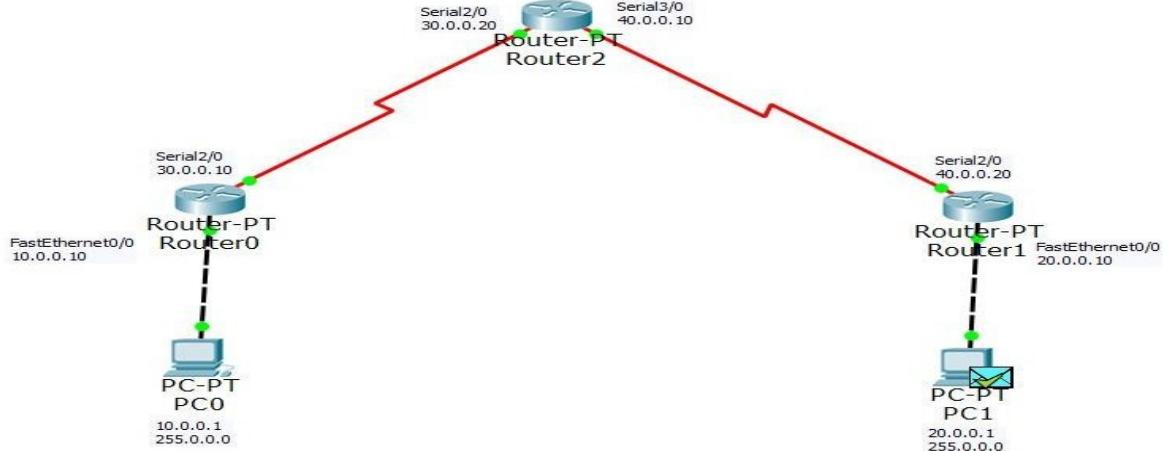
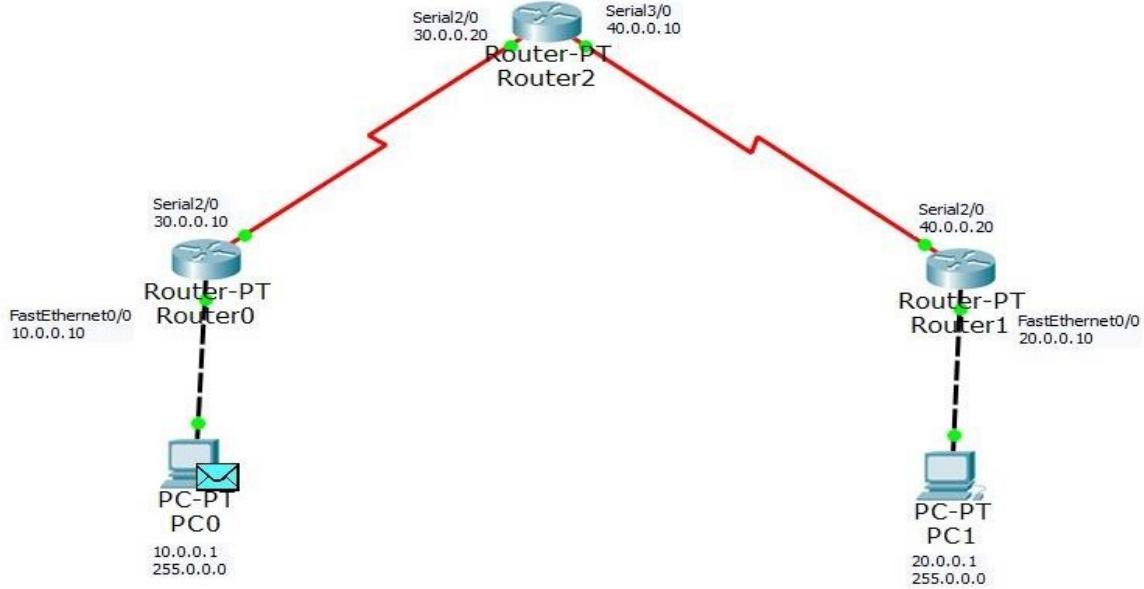
Aim of the program:

Configure default route, static route to the Router

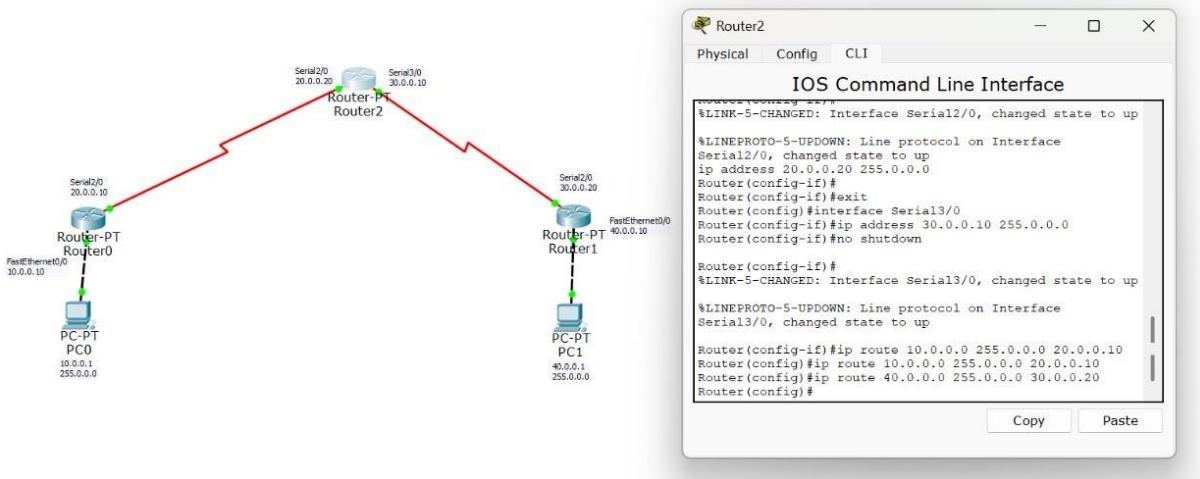
Procedure and topology:



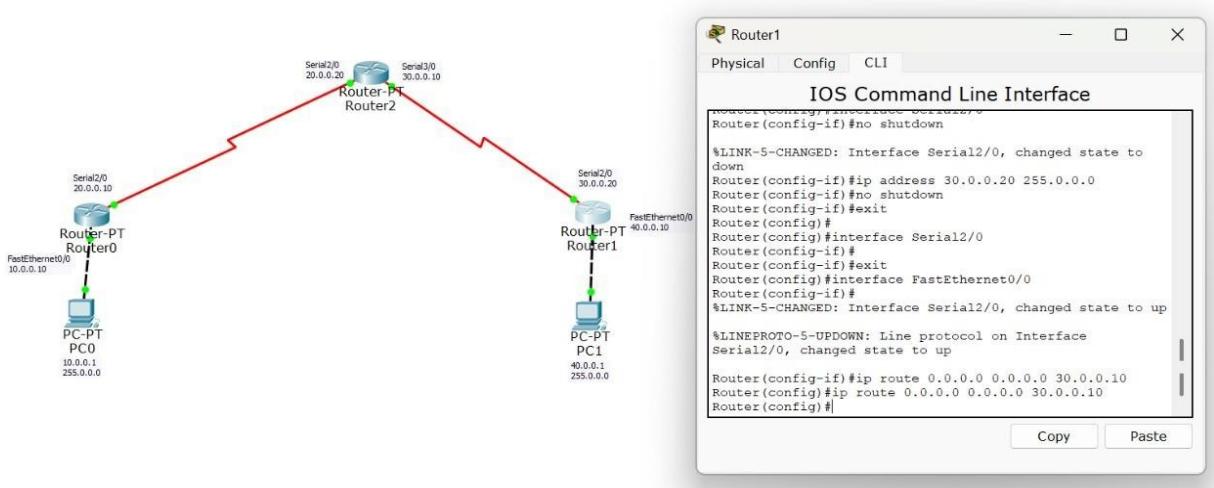
Screenshots/ Output:



Static routing CLI commands:



## Default routing CLI commands:



### Observation:

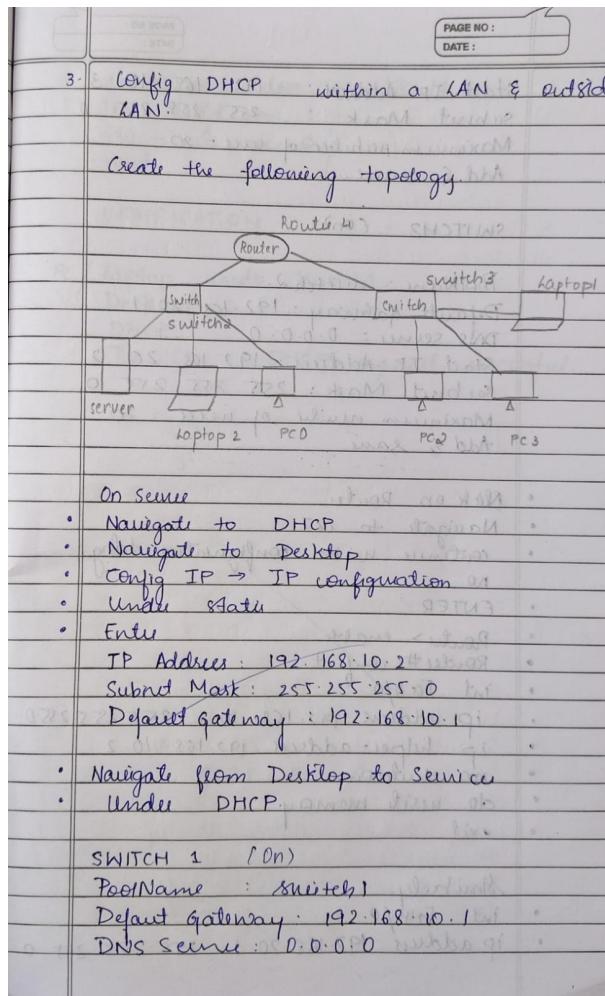
- The configured static and default routes correctly updated the router's routing table, enabling deterministic next-hop selection for remote networks.
  - Successful ping tests verified that traffic was forwarded according to the static/default route entries, ensuring end-to-end reachability across different network segments.

## Program 3

Aim of the program:

Configure DHCP within a LAN and outside LAN.

Procedure and topology:



PAGE NO : DATE :

Start IP Address : 192.168.10.3  
Subnet Mask : 255.255.255.0  
Maximum number of user = 20  
Add & save.

SWITCH2 (on)

PeerName : switch2.  
Default Gateway : 192.168.20.1  
DNS Server : 0.0.0.0  
Start IP Address : 192.168.20.2  
Subnet Mask : 255.255.255.0  
Maximum number of user : 20  
Add & save

- Click on Router
- Navigate to CLI
- continue with configuration dialog
- no
- ENTER
- Router> enable
- Router# conf +
- int Fa 0/0
- ip address 192.168.10.1 255.255.255.0
- ip helper-address 192.168.10.2
- no shutdown
- do write memory
- exit

Similarly.

- int Fa 0/0
- ip address 192.168.20.1 255.255.255.0

PAGE NO : DATE :

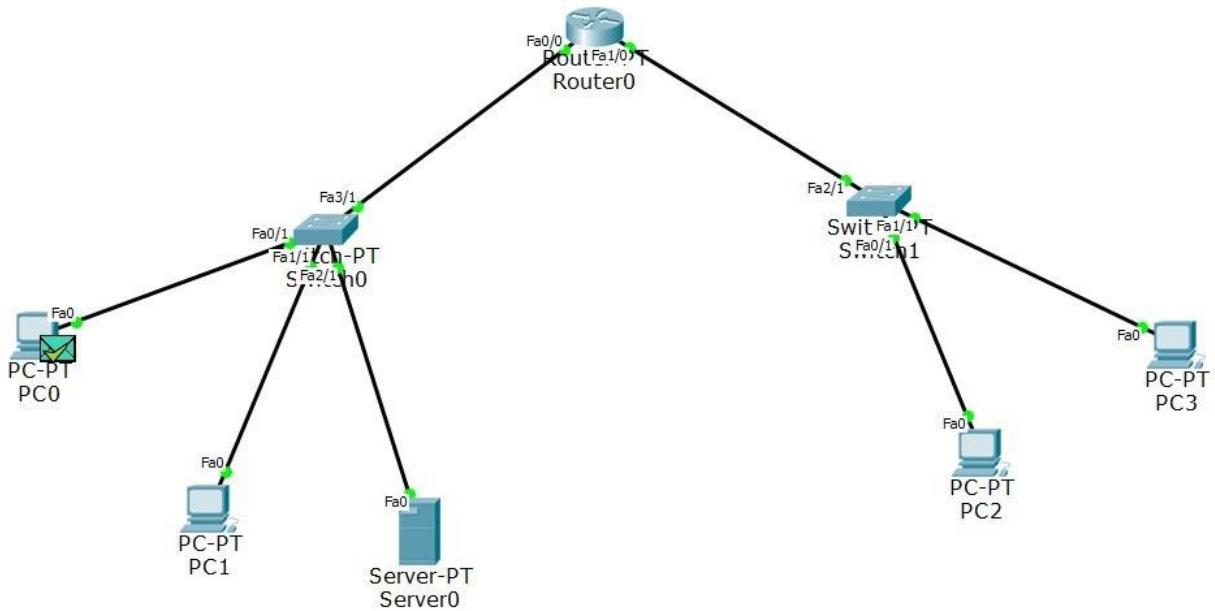
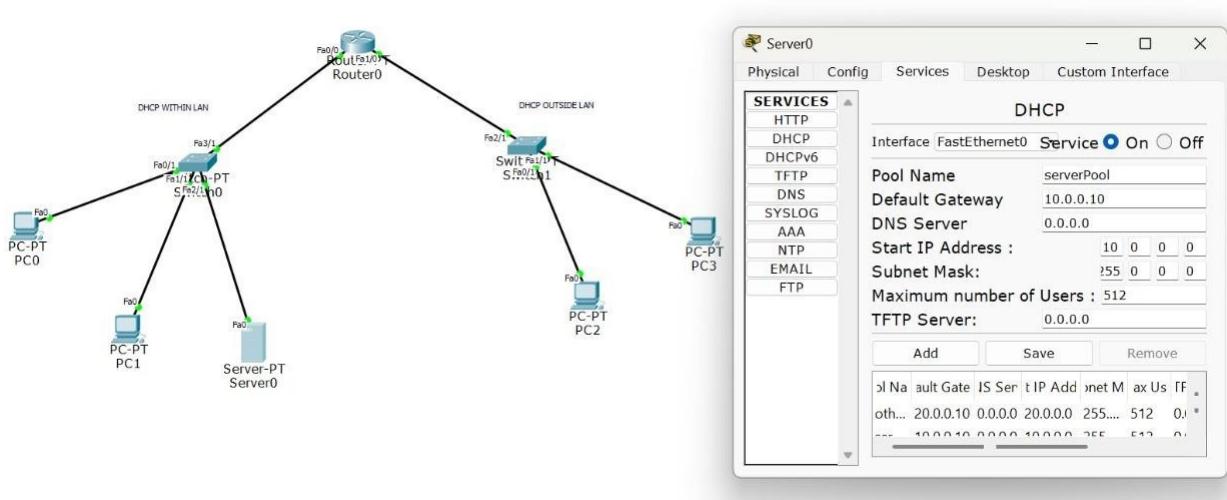
ip helper-address 192.168.10.2  
no shutdown  
do write memory  
exit

VERIFICATION

PC / Laptop under switch1  
Desktop  
DHCP  
The OP : DHCP request successful.

B  
3

## Screenshots/ Output:



## Observation:

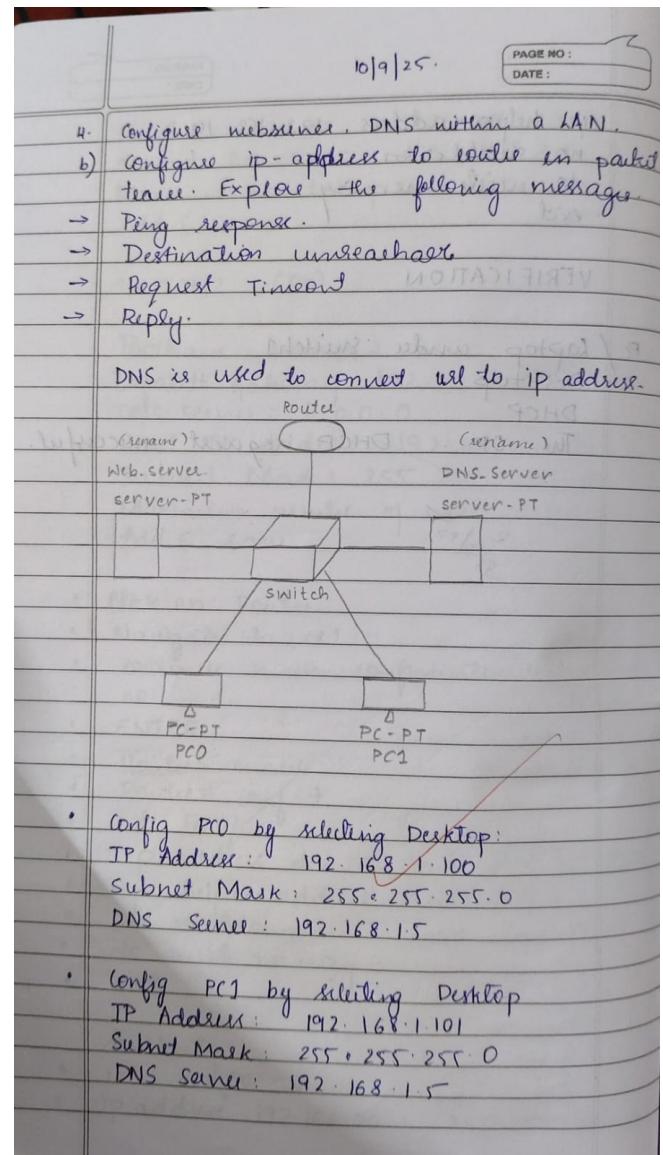
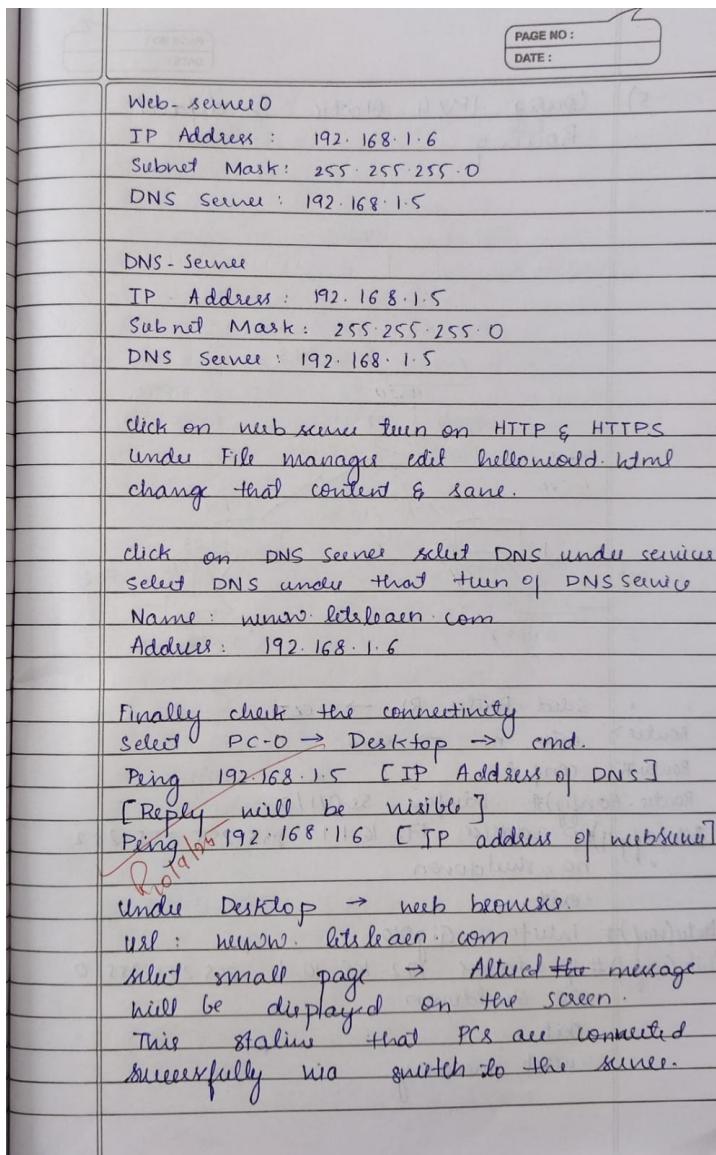
- The DHCP server successfully allocated IP addresses to clients within the LAN, confirming proper scope configuration and automatic distribution of network parameters.
- DHCP relay (IP Helper) enabled clients outside the LAN to obtain leases from the central DHCP server, demonstrating correct inter-network forwarding of DHCP Discover and Offer messages.

## Program 4

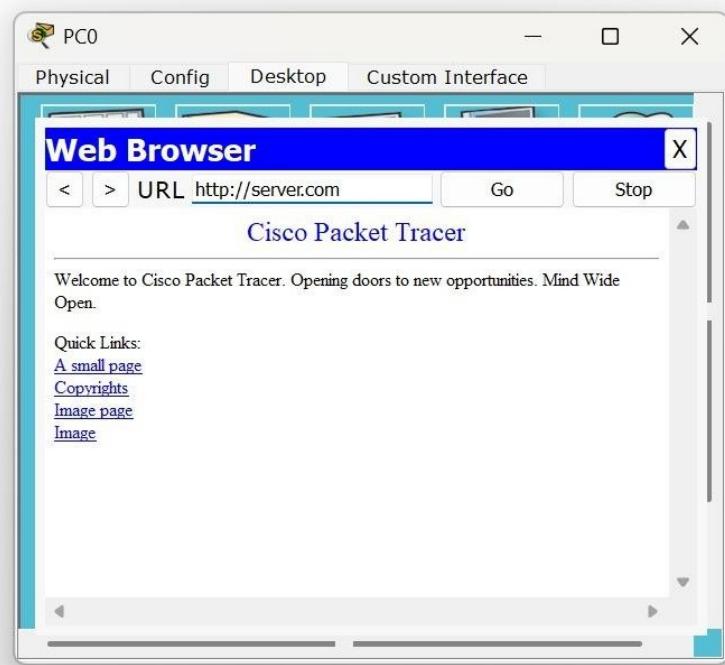
Aim of the program:

Configure Web Server, DNS within a LAN.

Procedure and topology:



## Screenshots/ Output:



## Observation:

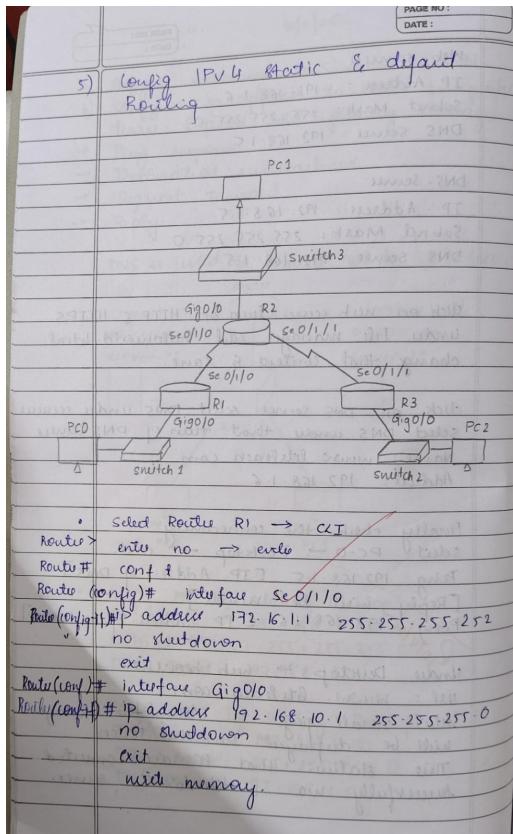
- The DNS server successfully resolved domain names to the corresponding web server's IP address, confirming proper hostname-to-IP mapping within the LAN.
- HTTP requests reached the web server using the DNS-resolved address, validating correct server configuration and internal LAN communication.

## Program 5

Aim of the program:

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

Procedure and topology:



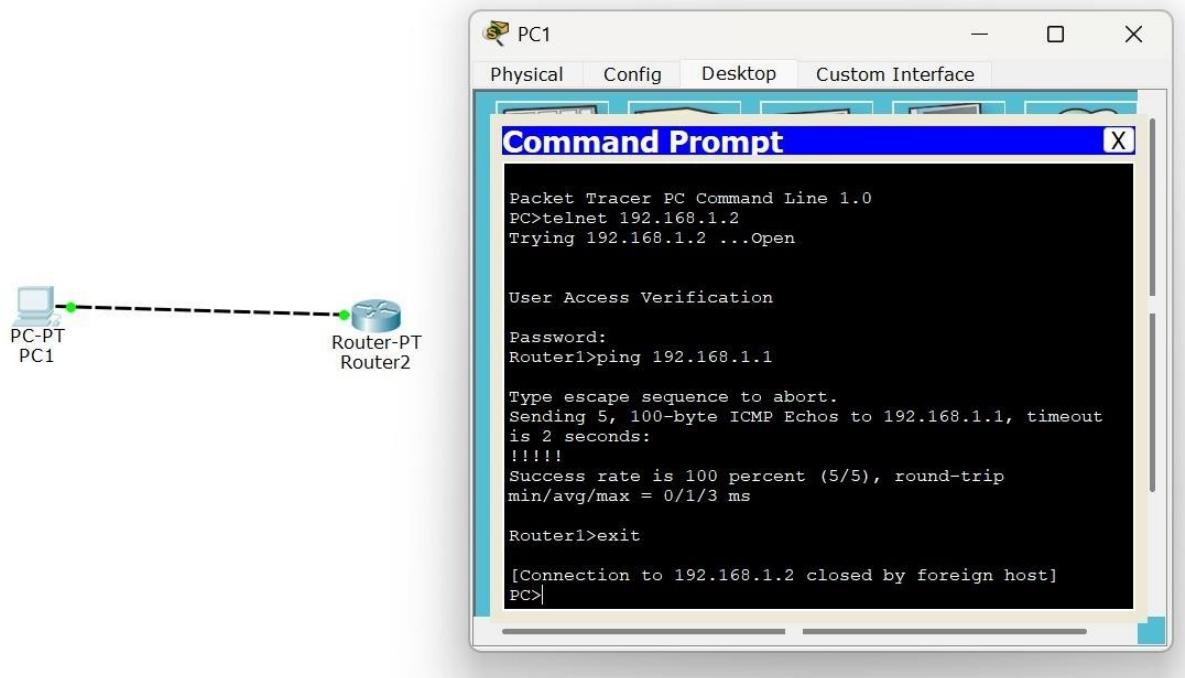
- Select R2 : CLI.  
 no → enable  
 Router # config +  
 Router (config) # hostname R2.  
 Router (config) # interface Se 0/1/0  
 # ip address 172.16.1.2 255.255.255.252  
 # no shutdown # exit  
 Router (config) # interface Gig 0/0  
 # ip address 192.168.20.1 255.255.0  
 # no shutdown # exit  
 Router (config) # interface Se 0/1/1  
 # ip address 172.16.2.1 255.255.255.252  
 Router (config) # no shutdown # exit  
 Router (config) # write memory.  
 Router (config) # no shutdown R3.
- Select R3 : CLI  
 no → enable.  
 Router # config +  
 Router (config) # hostname R3.  
 # interface Se 0/1/1  
 # ip address 172.16.2.2 255.255.255.252  
 # no shutdown # exit  
 # int Gig 0/0  
 # ip address 192.168.30.1 255.255.255.0  
 # no shutdown # exit  
 Router (config) # write memory.
- Click on PC0 → Desktop → CLI  
 ↳ Desktop  
 ↳ IP configuration

PC0	PC1	PC2
IP Address Gateway	192.168.10.10 192.168.10.1	192.168.30.10 192.168.30.1
		192.168.30.1

- Click on R1  
 R1 # config  
 R1 (config) # ip route 192.168.20.0 255.255.255.252 172.16.1.2.  
 # ip route 192.16.2.0 255.255.255.252 172.16.1.1  
 # ip route 192.168.30.0 255.255.255.0 172.16.1.2  
 exit  
 write memory.
- Click on R2.  
 R2 # config  
 R2 (config) # ip route 192.168.10.0 255.255.255.0 172.16.1.1  
 # ip route 192.168.30.0 255.255.255.0 172.16.2.2  
 exit  
 write memory.
- Click on R3  
 R3 # config +  
 R3 (config) # ip route 0.0.0.0 0.0.0.0 Se 0/1/0  
 exit, write memory.
- To see Routing table  
 click on R1 | Click on R2 | Click on R3  
 ↳ Show IP route | ↳ ↳ ↳

Click on PC0 → Desktop → CLI  
 → ping 192.168.10.1  
 → ping 192.168.20.1  
 → ping 192.168.30.1  
 One ping is successful send PDU from  
 PC0 to PC1  
 10/10s

Screenshots/ Output:



The screenshot shows the Router2 CLI interface with the following tabs: Physical, Config, and CLI. The CLI tab is active and displays the IOS Command Line Interface.

The terminal window shows the following session:

```

Router(config-if)#npno shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state
to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to up

Router(config-if)#exit
Router(config)#hostname Router1
Router1(config)#enable secret p1
Router1(config)#line vty 0 4
Router1(config-line)#login
% Login disabled on line 132, until 'password' is set
% Login disabled on line 133, until 'password' is set
% Login disabled on line 134, until 'password' is set
% Login disabled on line 135, until 'password' is set
% Login disabled on line 136, until 'password' is set
Router1(config-line)#password cisco
Router1(config-line)#exit
  
```

At the bottom of the window are 'Copy' and 'Paste' buttons.

Observation:

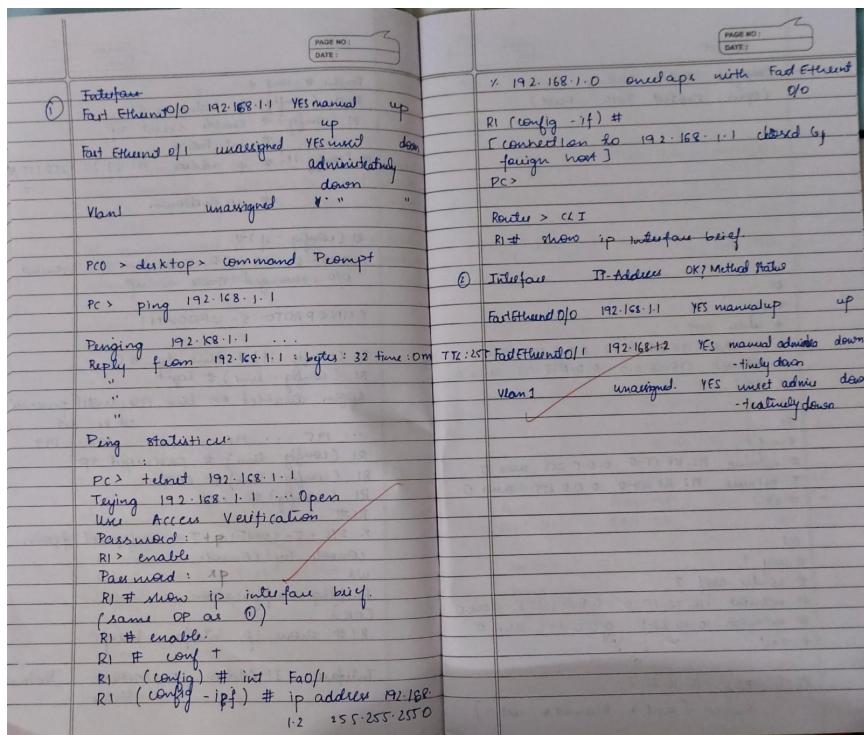
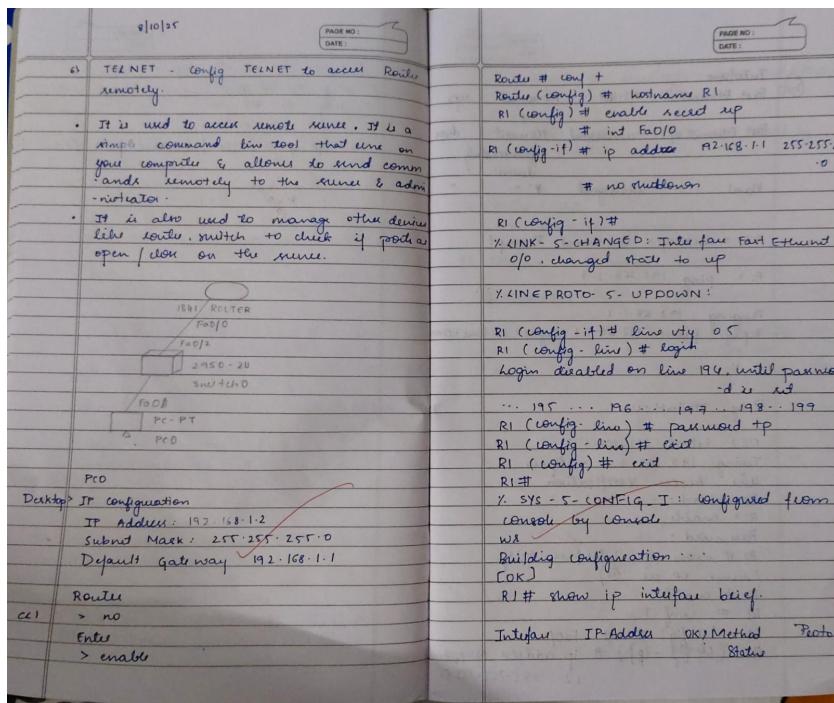
- The Telnet session successfully established a remote CLI connection to the router, confirming proper VTY line configuration and IP reachability between the IT office PC and the server room router.
- Command execution over the Telnet session demonstrated reliable remote device management.

## Program 6

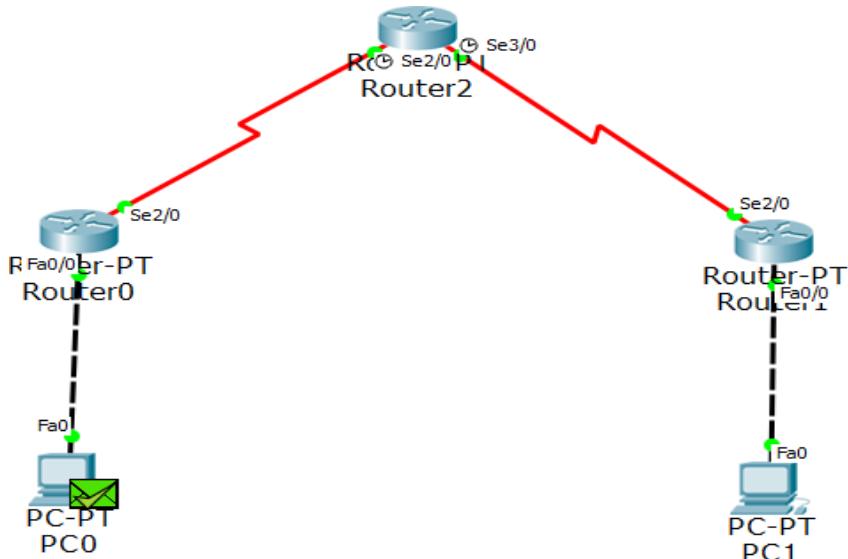
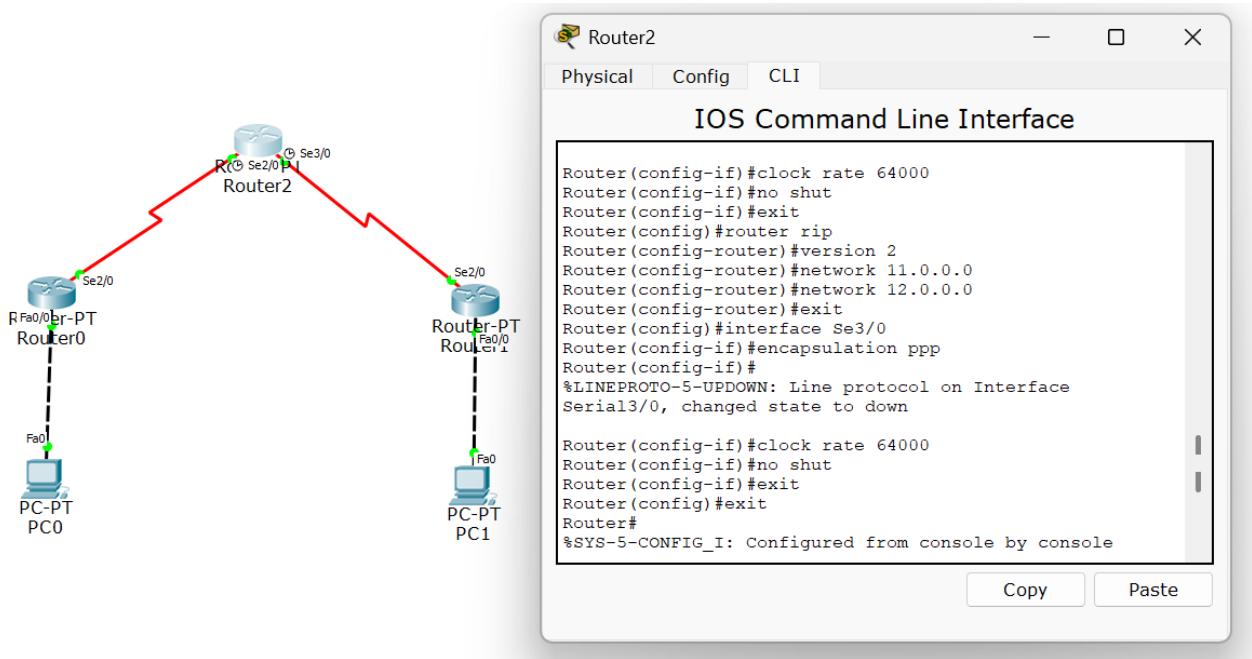
Aim of the program:

Configure RIP routing Protocol in Routers

Procedure and topology:



## Screenshots/ Output:



## Observation:

- RIP routing updates were successfully exchanged between routers, allowing each router to dynamically learn remote network routes through hop-count-based distance vector advertisements.
- The routing tables converged correctly, and successful ping tests confirmed end-to-end

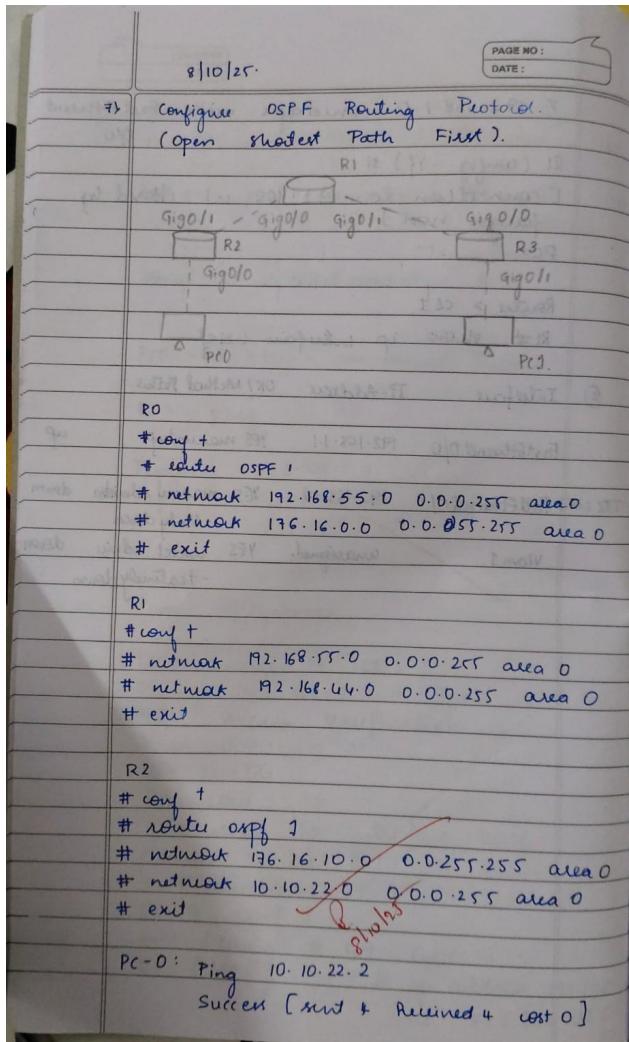
connectivity maintained by periodic RIP updates and route propagation.

## Program 7

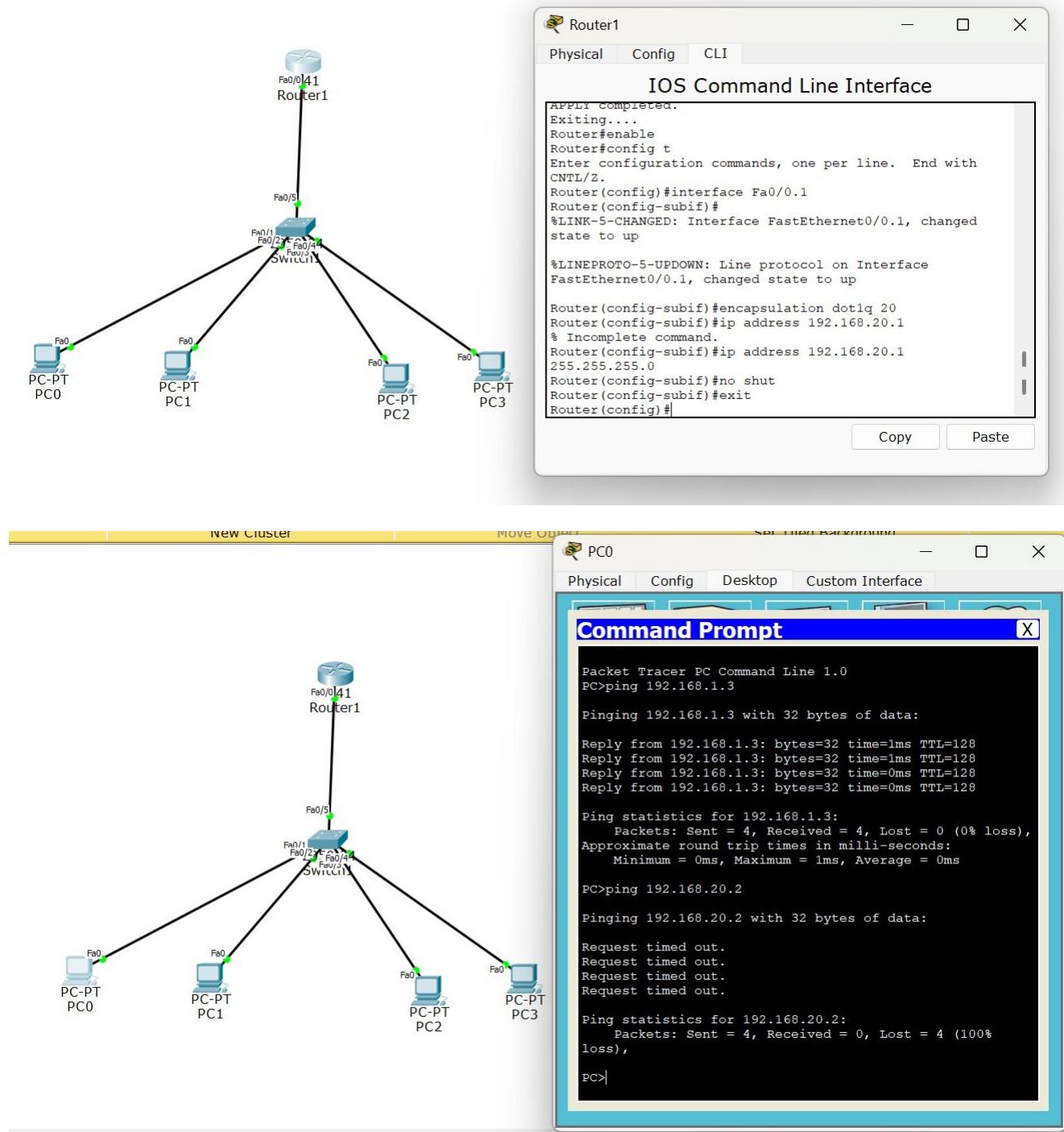
Aim of the program:

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

Procedure and topology:



## Screenshots/ Output:



## Observation:

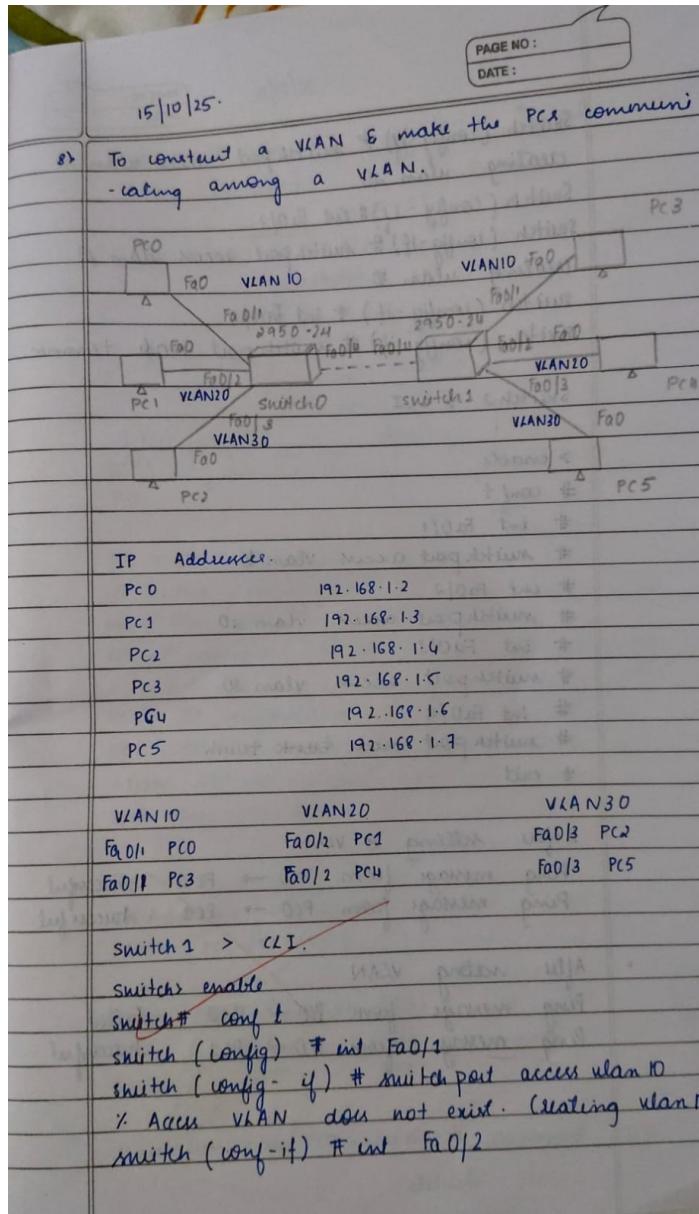
- VLAN segmentation successfully separated broadcast domains, and switch ports were correctly assigned to their respective VLAN IDs using access mode configuration.
- Inter-VLAN communication was achieved through the Layer-3 device, and successful ping tests confirmed proper VLAN membership, tagging, and routing functionality.

## Program 8

Aim of the program:

To construct a WLAN and make the nodes communicate wirelessly

Procedure and topology:



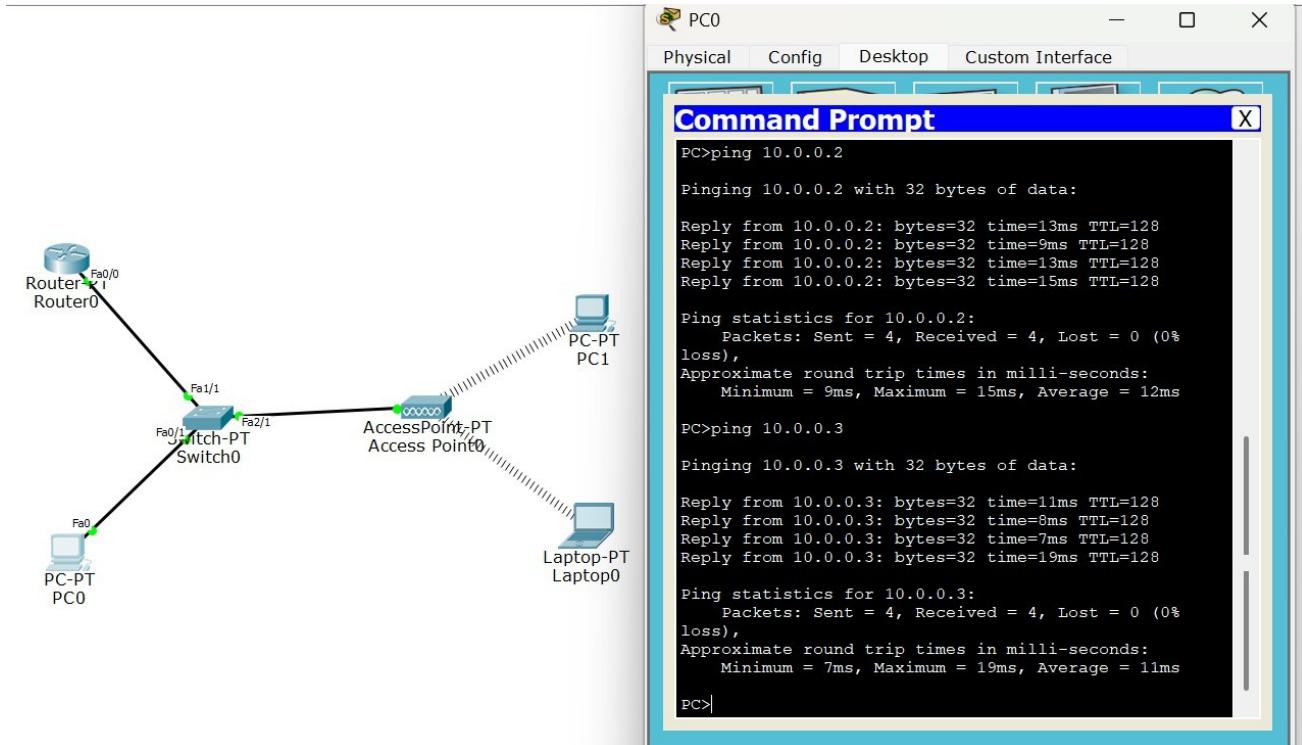
PAGE NO: \_\_\_\_\_  
DATE: \_\_\_\_\_

```

Switch (config-if) # switchport access vlan 20
creating wlan 20
Switch (config-if) # int Fa0/3
switch (config-if) # switchport access vlan 30
creating wlan 30
switch (config-if) # int Fa0/4
switch (config-if) # switchport mode trunk
switch2 > CLI
> enable
# config t
# int Fa0/1
# switchport access vlan 10
# int Fa0/2
# switchport access vlan 20
# int Fa0/3
# switchport access vlan 30
# int Fa0/4
# switchport mode trunk
# exit
• By you setting a VLAN
Ping message from PC0 → PC2 successful
Ping message from PC0 → PC1 successful
• After setting VLAN
Ping message from PC0 → PC2 Failed
Ping message from PC0 → PC1 successful

```

## Screenshots/ Output:



## Observation:

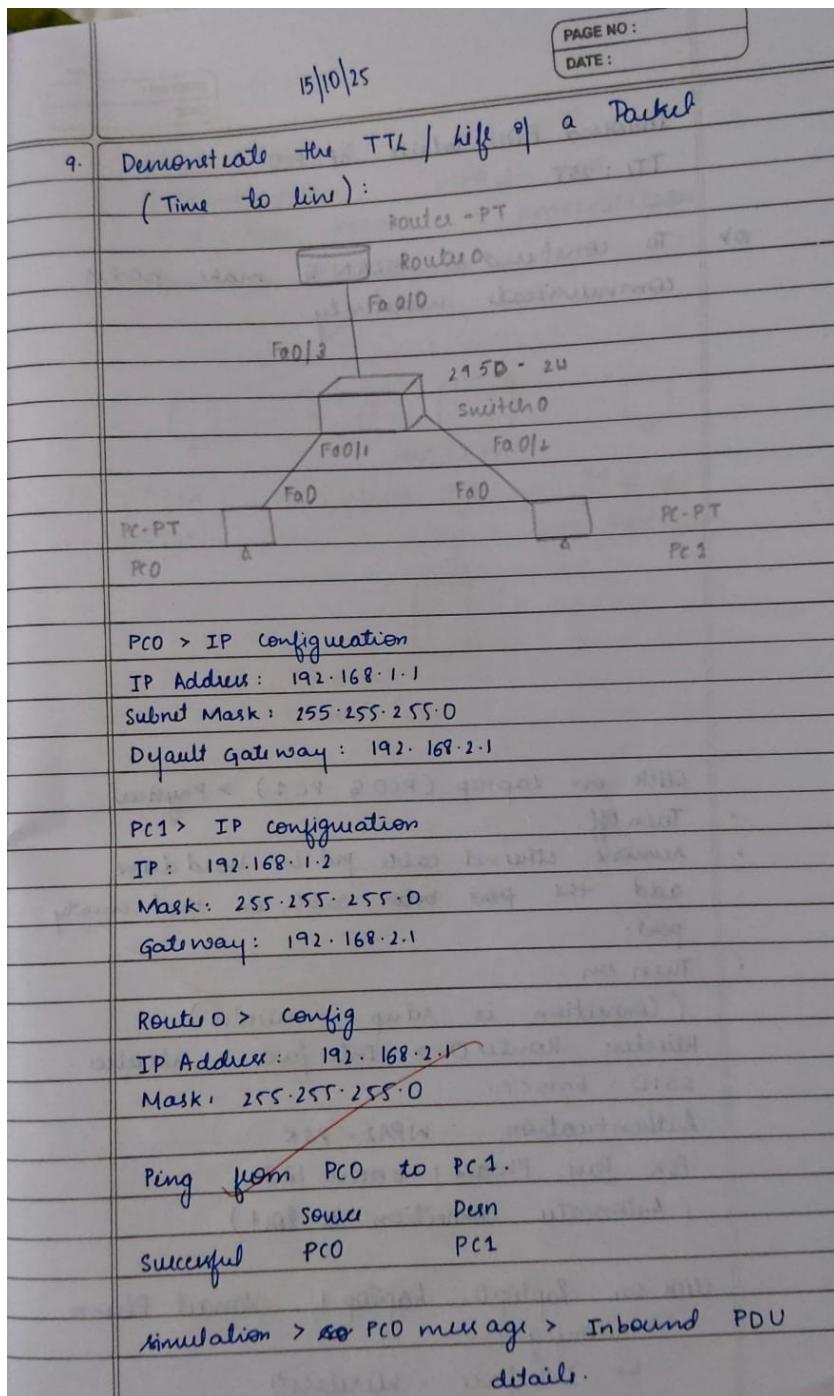
- The WLAN configuration enabled wireless nodes to associate with the access point using the configured SSID and security settings, confirming proper authentication and signal coverage.
- Successful ping communication between wireless devices verified stable wireless Connectivity.

## Program 9

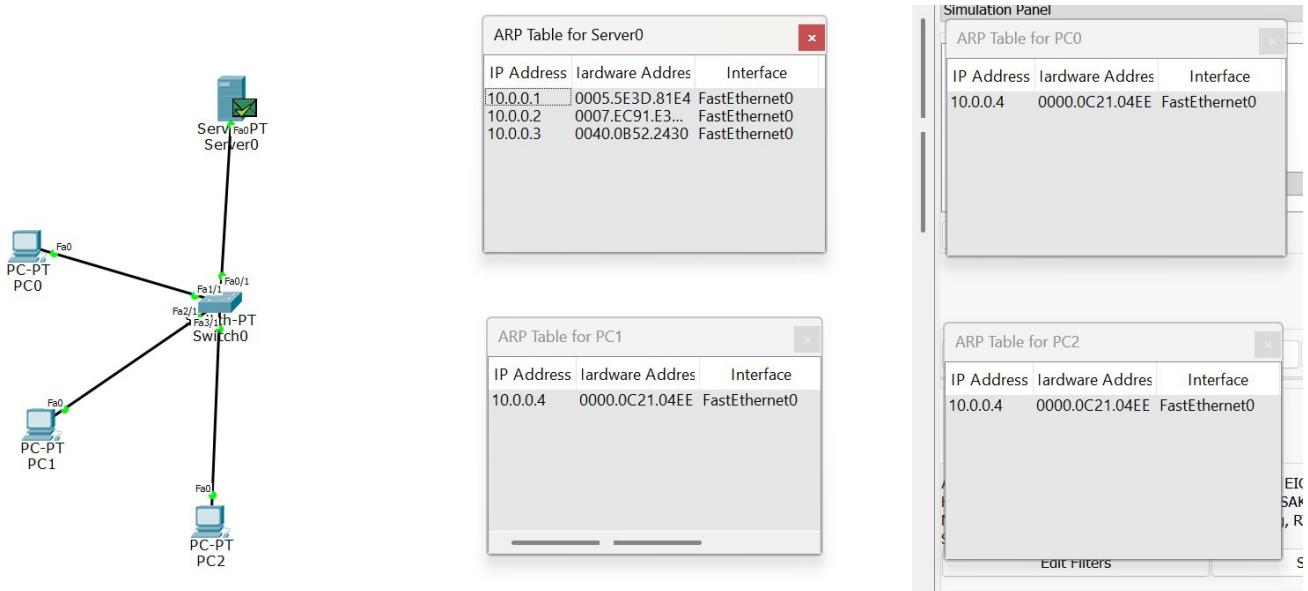
Aim of the program:

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

Procedure and topology:



## Screenshots/ Output:



Observation:

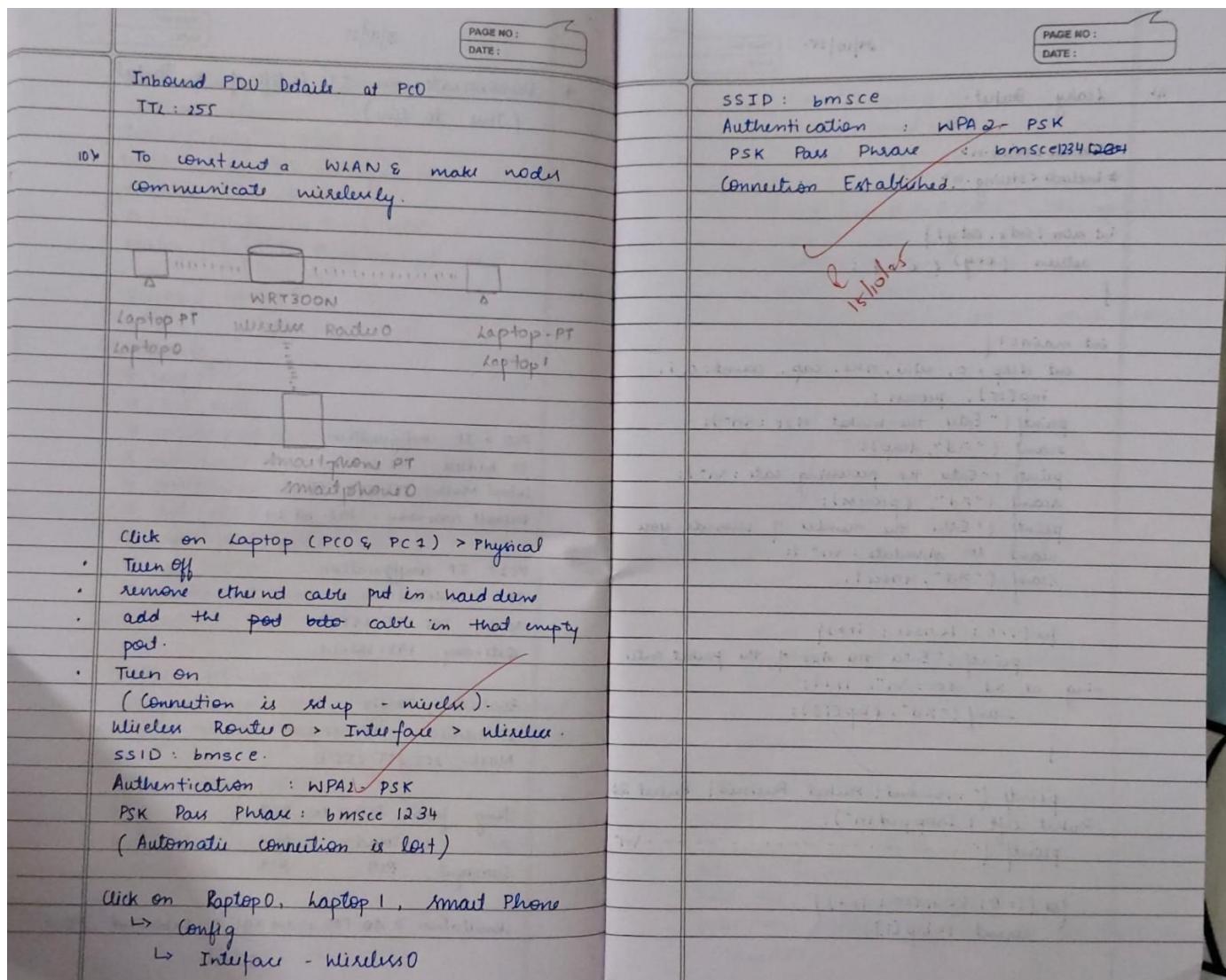
- ARP successfully resolved the destination IP address to its corresponding MAC address, as seen from ARP request and reply exchanges between LAN hosts.
- The populated ARP tables and successful ping communication confirmed correct layer-2 addressing, frame forwarding, and basic LAN operation.

## Program 10

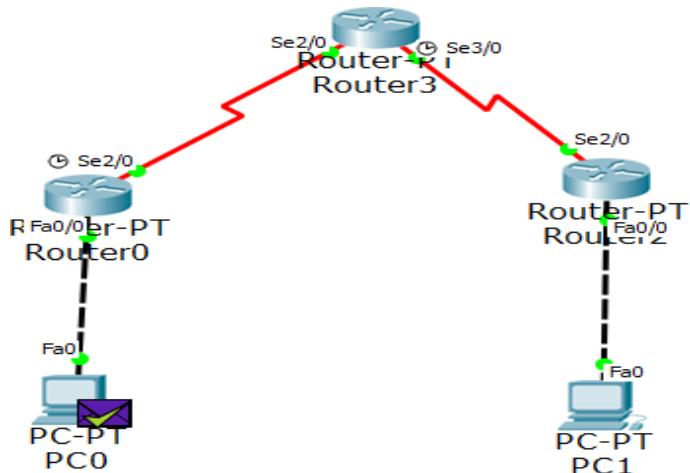
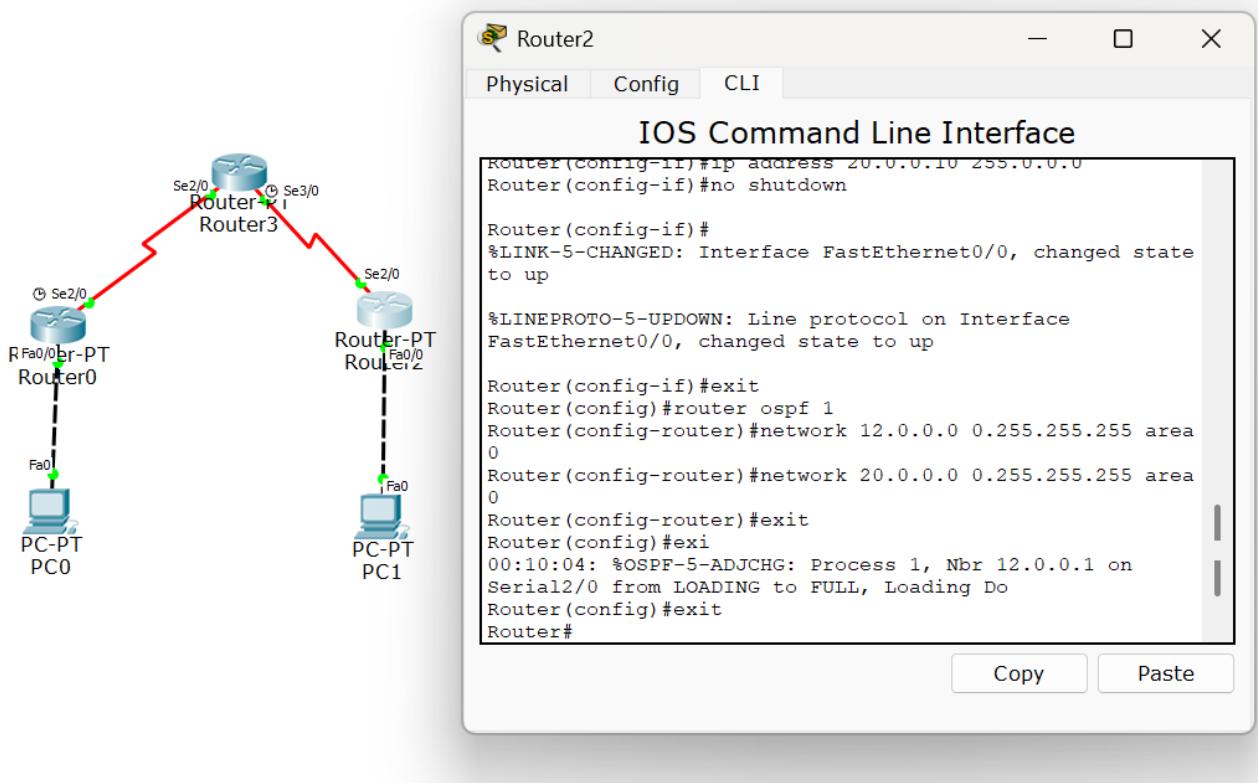
Aim of the program:

Configure OSPF routing protocol

Procedure and topology:



## Screenshots/ Output:



## Observation:

- OSPF successfully established neighbour adjacencies and exchanged LSAs, allowing routers to build a synchronized link-state database across the OSPF area.
- The routing tables converged using SPF calculations, and successful pings confirmed efficient

path selection and dynamic route learning through OSPF.

## Program 11

Aim of the program:

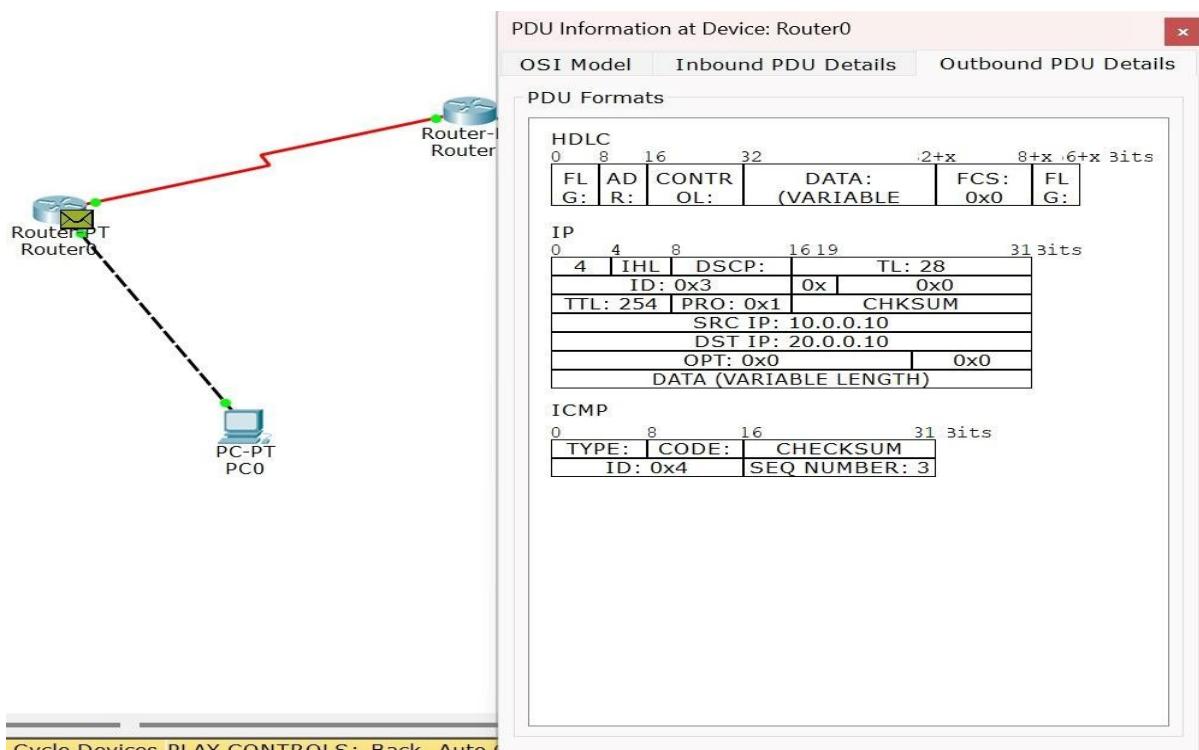
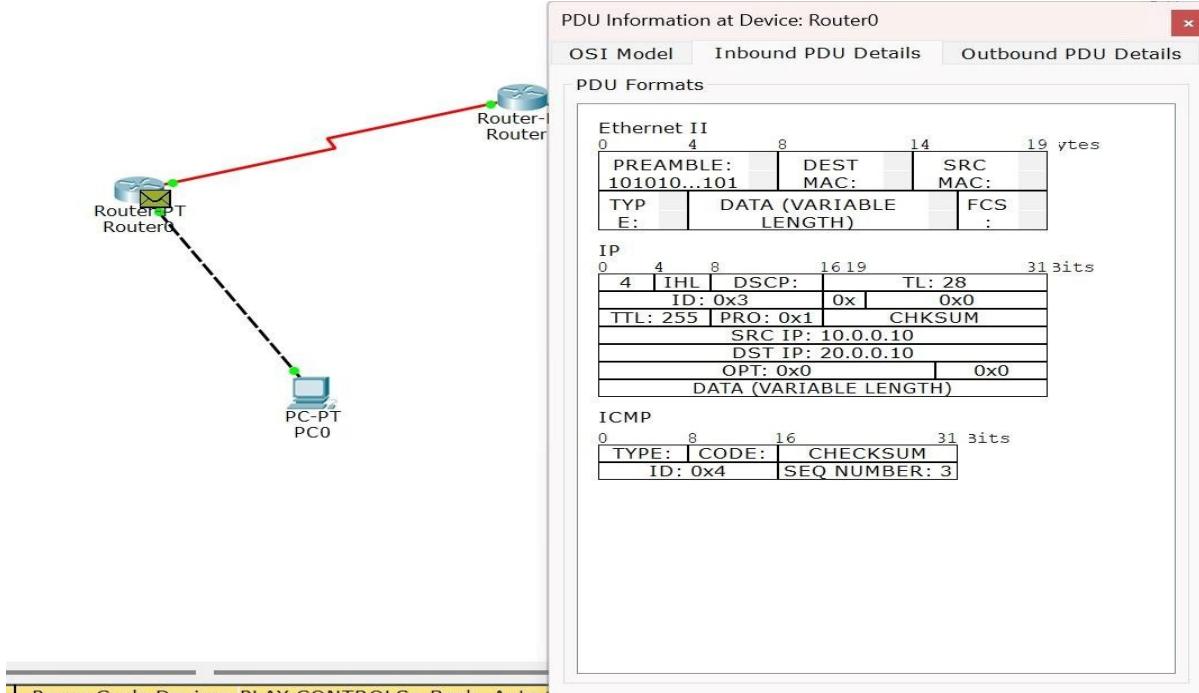
Demonstrate the TTL/ Life of a Packet

Procedure and topology:

<pre> 29/10/25. PAGE NO: DATE: 11. Leaky Bucket. #include &lt;stdio.h&gt; #include &lt;string.h&gt;  int min ( int x, int y ) {     return (x &lt; y) ? x : y ; }  int main() {     int deep = 0, mini, max, cap, count = 0, i,         inp[25], process;     printf ("Enter the bucket size : \n");     scanf ("%d", &amp;cap);     printf ("Enter the processing rate : \n");     scanf ("%d", &amp;process);     printf ("Enter the number of seconds you want to simulate : \n");     scanf ("%d", &amp;nsec);      for (i=0; i&lt;nsec; i++) {         printf ("Enter the size of the packet entering at sec %d : \n", i+1);         scanf ("%d", &amp;inp[i]);         count += inp[i];     } } </pre>	<pre> if (count &gt; cap) {     deep = count - cap;     count = cap; } printf ("%d   %d   ", i+1, inp[i]); mini = min (count, process); printf ("%d   %d   ", mini); count -= mini; printf ("%d   %d   \n", count, deep); deep = 0; }  while (count != 0) {     if (count &gt; cap) {         deep = count - cap;         count = cap;     }     printf ("%d   %d   ", ++i, 0);     mini = min (count, process);     printf ("%d   %d   ", mini);     count -= mini;     printf ("%d   %d   \n", count, deep);     deep = 0; } return 0; }  OUTPUT: Enter the bucket size : 5 Enter the processing rate : 2 Enter the number of seconds you want to simulate : 3 </pre>
--	---

Second	Packet Received	Packet sent	Packet left	Deep
1	5	2	3	0
2	4	2	3	2
3	3	2	1	1
4	0	2	1	0
5	0	0	0	0

## Screenshots/ Output:



Observation:

- The TTL field in the IP header decreased by one at each router hop, demonstrating its role in preventing packets from looping indefinitely in the network.

## Program 12

Aim of the program:

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

Procedure and topology:

PAGE NO :  
DATE :

Q. Write a P. Error detection using CRC - CCITT  
(16 bit)

F: 1101011011      G: 10011  
Add (G-1) zeros.      (XOR for subtraction)

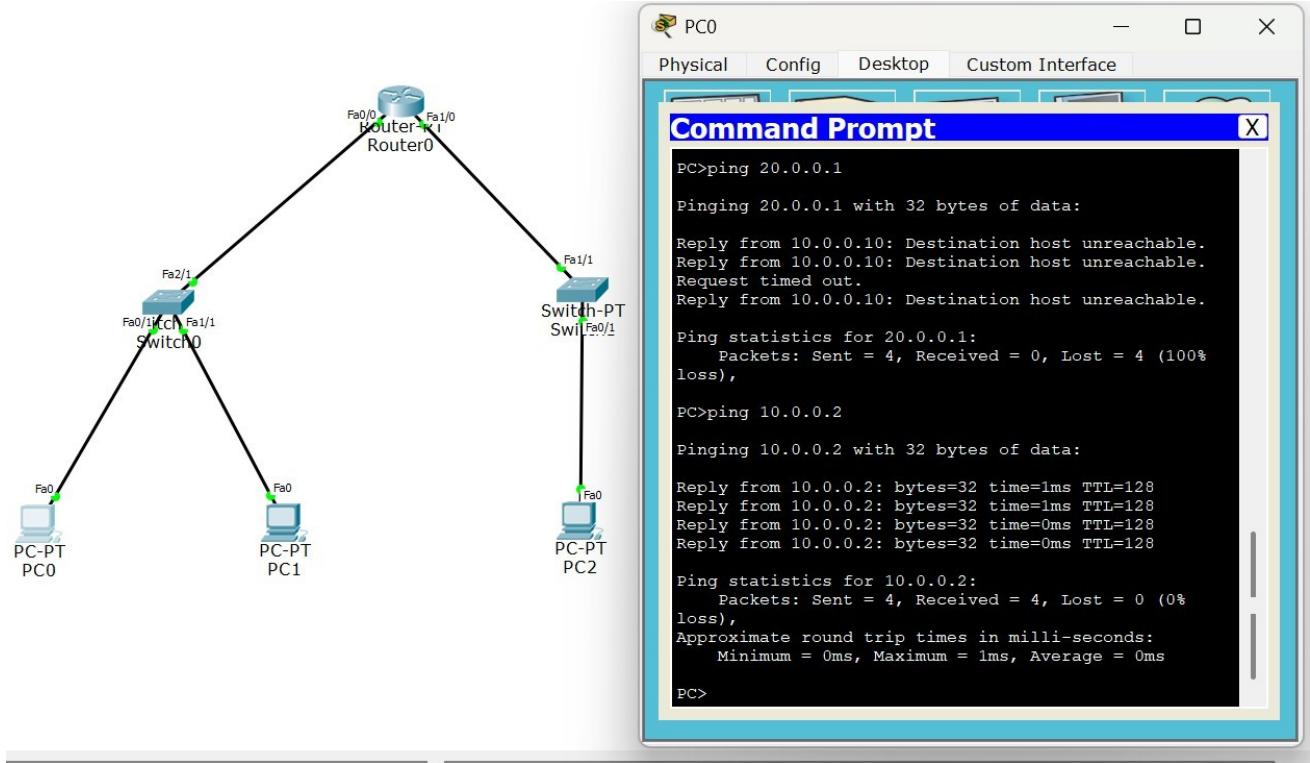
10011 | 11010110110000  
- 10011 | | | | | |  
010011 | | | | | |  
10011 | | | | | |  
0000010110 | | | | | |  
- 10011 | | | | | |  
0010100 | | | | | |  
10011 | | | | | |  
001110 - Remainder.

There is error in the transmission

F: 100100      G:  $x^3 + x^2 + 1 \Rightarrow 1101$   
(G-1) : 3 zeros.

1101 | 100100000 | 11101  
1101 | | | | | |  
01000 | | | | | |  
2910101 | | | | | |  
01010 | | | | | |  
1101 | | | | | |  
01110 | | | | | |  
1101 | | | | | |  
001100 | | | | | |  
1101 | | | | | |  
0001 | | | | | |

## Screenshots/ Output:



## Observation:

- Proper IP addressing on router interfaces enabled successful ICMP echo and echo-reply communication, confirming correct Layer-3 configuration and reachability.
- “Destination Unreachable” and “Request Timed Out” responses occurred when routes or interfaces were misconfigured, demonstrating how routers handle missing paths and non-responsive hosts.

## CYCLE 2:

### Program 13

Aim of the program:

Write a program for congestion control using Leaky bucket algorithm

Procedure and topology:

PAGE NO.:	DATE:
13. Write a program in C for client server communication using TCP or IP sockets to make client send the name of the file of server to send back the contents of the requested file if present.	3) listen for connections with the listen (sfds, ...) sys call 4) sfds = accept a connection with the accept (sfds, ...) sys call. 5) read the file name by socket 6) open the file by fd = open (buff) 7) read the contents of file 8) write the file content to socket 9) go to step 7 if m > 0 10) close (sfds)
<b>ALGORITHM (client side)</b> 1) sfds = create a socket with the socket (...) system call. 2) connect the socket to the address of the server using connect (sfds, ...) system call. The IP address of the server machine & port no. of the server service need to be provided. 3) Read file name for standard input by n: read (stdio, buffer, sizeof (buffer)) 4) write file name to the socket using write (sfds, buffer, n) 5) read file contents from the socket by m: read (sfds, buffer, sizeof (buffer)) 6) display file contents to standard output by write (stdio, buffer, m) 7) go to step 5 if m > 0 8) close socket by close (sfds)	<b>ALGORITHM (server side)</b> 1) sfds = create a socket with socket (...) sys call. 2) bind the socket to an address using the bind (sfds, ...) system call. If not sure of machine IP address, keep the structure member s-addr to INADDR_ANY. Assign a port no bw 3000 & 5000 to in-port

Screenshots/ Output:

**Output**

```
Enter bucket size: 4
Enter outgoing size: 1
Enter number of inputs: 7
Enter the incoming packet size: 3
Bucket buffer size 3 out of 4
After outgoing 2 packets left out of 4 in buffer
Enter the incoming packet size: 2
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 1
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 4
Dropped 3 packets
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 3 out of 4
After outgoing 2 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 2 out of 4
After outgoing 1 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 1 out of 4
After outgoing 0 packets left out of 4 in buffer
```

Observation:

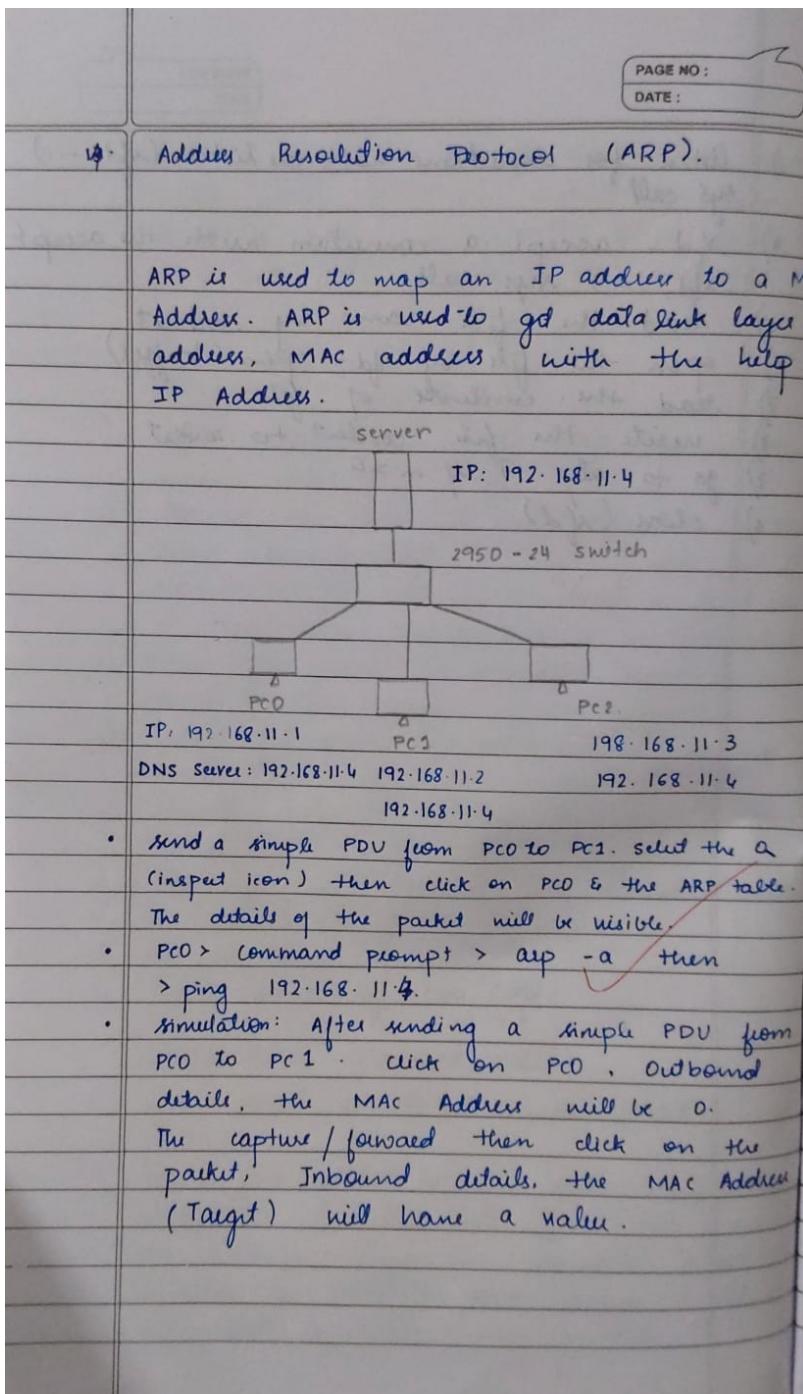
- The leaky bucket mechanism regulated the outgoing packet flow at a constant rate, preventing sudden traffic bursts from overwhelming the network.
- Packets exceeding the bucket capacity were dropped, demonstrating effective congestion control by smoothing traffic and enforcing rate-limiting.

## Program 14

Aim of the program:

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

Procedure and topology:



Screenshots/ Output:

## Output

```
Enter message bits: 101100
```

```
Enter polynomial g(x): 1001
```

```
Padded data (Message + zeros): 101100000
```

```
CRC bits (remainder): 001
```

```
Transmitted message: 101100001
```

```
Enter received bits: 101100001
```

```
No Error detected. Message OK.
```

```
==== Code Execution Successful ===
```

Observation:

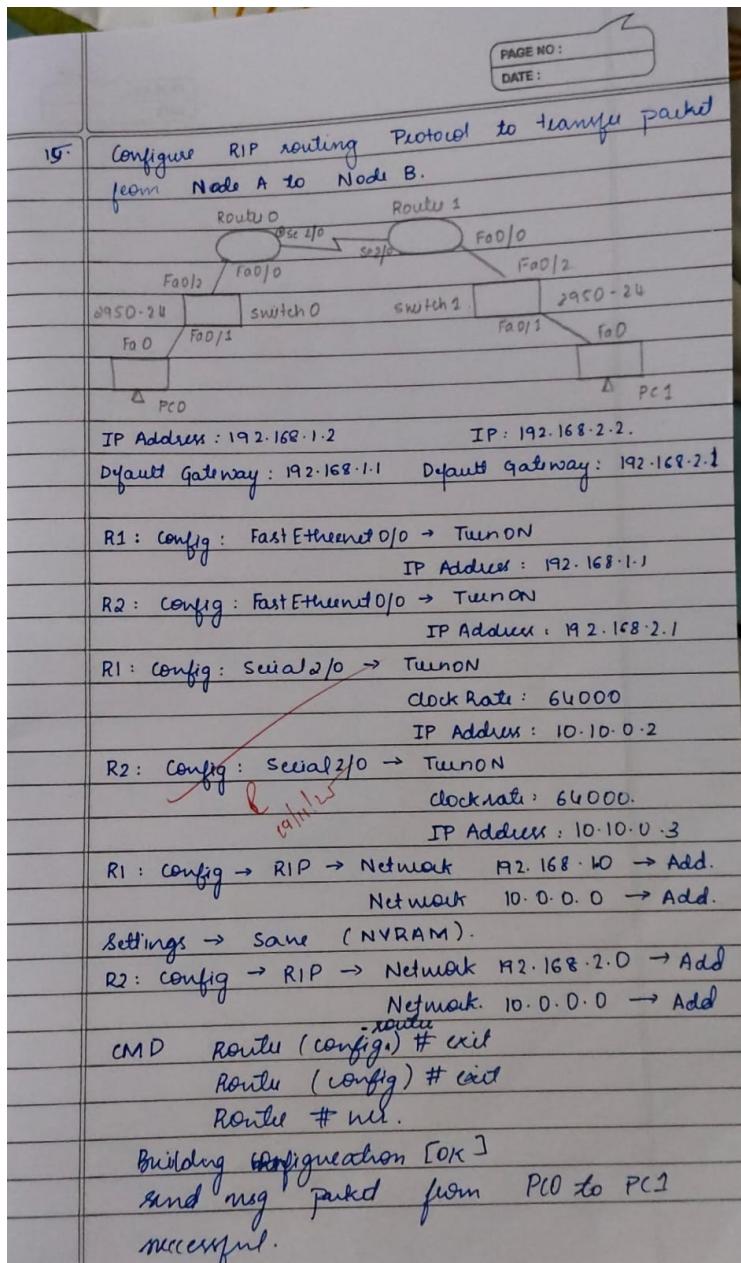
- The CRC-CCITT (16-bit) algorithm correctly generated a checksum for the transmitted data, ensuring reliable detection of single-bit and burst errors.
- Intentional error tests produced mismatched CRC values at the receiver, confirming accurate error detection through polynomial division.

## Program 15

Aim of the program:

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.

Procedure and topology:



Screenshots/ Output:

```
● PS D:\CN stuff\TCP> python client.py
Enter filename to request: test.txt

--- File Content ---

Hello from server side
○ PS D:\CN stuff\TCP> □
```

```
PS D:\CN stuff\TCP> python --version
● Python 3.12.6
● PS D:\CN stuff\TCP> python server.py
Server is listening on port 8080 ...
Connected by: ('127.0.0.1', 65258)
○ PS D:\CN stuff\TCP> □
```

Observation:

- The TCP client successfully established a reliable connection with the server and transmitted the requested filename using stream-oriented communication.
- The server correctly retrieved and returned the file contents over the same TCP session, demonstrating reliable data transfer, acknowledgment handling, and error-free delivery.

## Program 16

Aim of the program:

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.

Procedure and topology:

12.11.25

Write a program in C/C++ for client Server communication using TCP or IP sockets

**Algorithm (Client side)**

- \* sfd = Create a socket with socket(... ) System call.
- \* Connect the socket to the address of the server using the connect(sfd, system call).
- \* Read file name from standard input by n = read (stdio, buffer, sizeof(buffer))
- \* Write file name to the socket using write (sfid, buffer, n)
- \* Read file contents from the socket by m = read (sfid, buffer, sizeof(buffer))
- \* Display file contents to standard output by write (stdio, buffer, m)
- \* Close socket by close(sfd).

**Algorithm (Server side)**

- \* sfid = Create a socket with socket( ) System call
- \* Bind the socket to an address using the bind(sockfd, system call).
- \* Listen for connections with the listen(sfid, ...) system call.
- \* sfid = Accept a connection with the accept (sfid, ..) system call.
- \* Read the filename from the socket by n = read (sfid, buffer, sizeof(buffer))
- \* Open the file by fd = open (buffer)
- \* Read the contents of the file by m = read (fd, buffer, sizeof(buffer))

Screenshots/ Output:

```
● PS D:\CN stuff\UDP> python server.py
  UDP Server is ready ...
  Requested file: test.txt
○ PS D:\CN stuff\UDP> █
```

```
● PS D:\CN stuff\UDP> python client.py
  Enter filename to request: test.txt

  --- File Content ---

  Welcome to UDP file server
○ PS D:\CN stuff\UDP> █
```

Observation:

- The UDP client successfully sent the filename as a connectionless datagram, demonstrating non-reliable, low-overhead message transfer without session establishment.
- The server responded with the file contents using UDP packets, and correct reception validated functional data exchange despite the absence of acknowledgment and retransmission mechanisms.