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

GithubLink: https://github.com/HithaHarishCS/CN

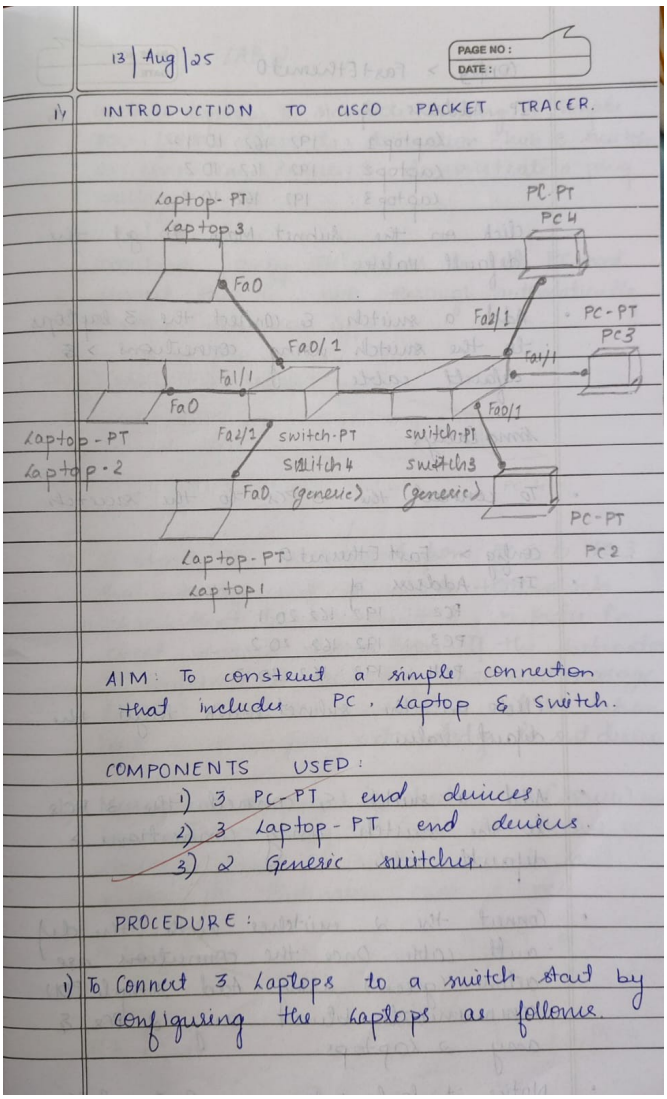
CYCLE 1:

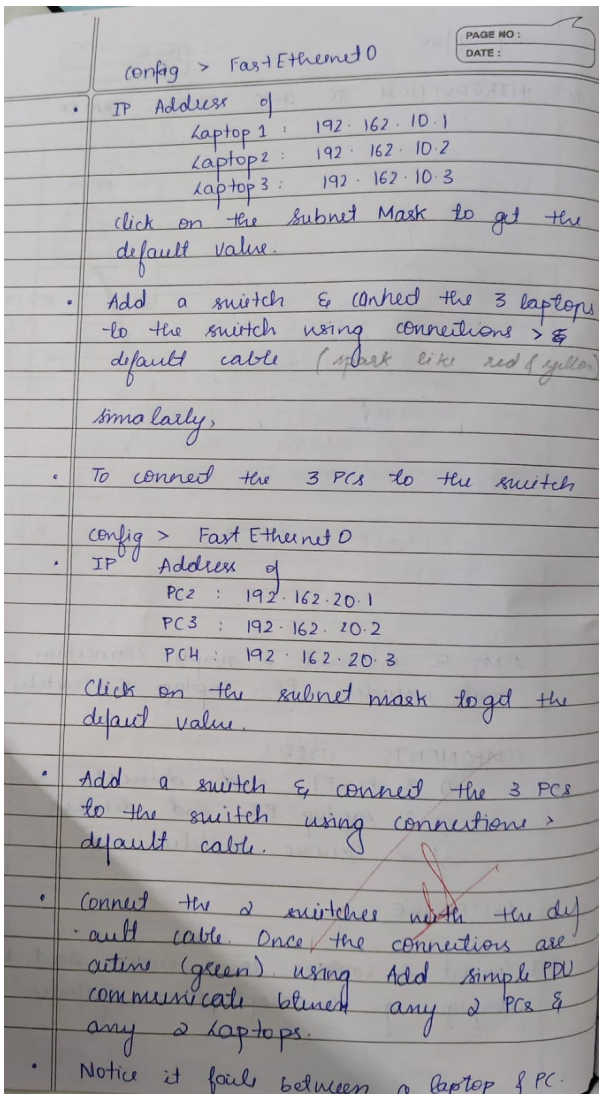
Program 1

Aim of the program:

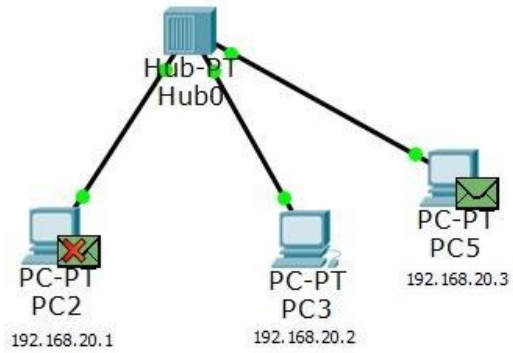
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

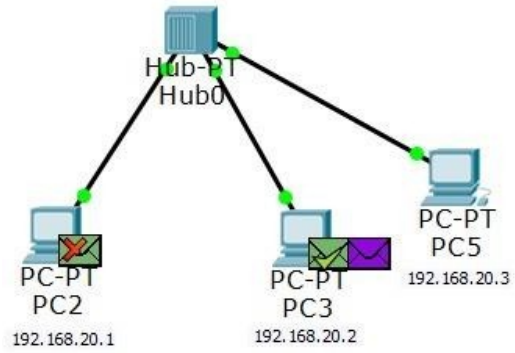
Procedure and topology:

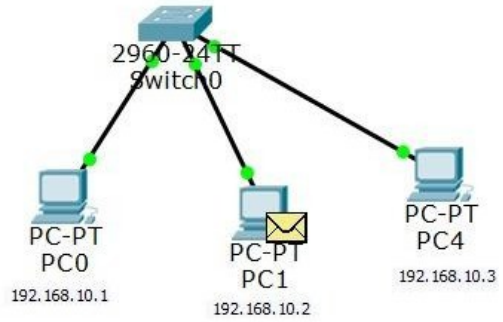


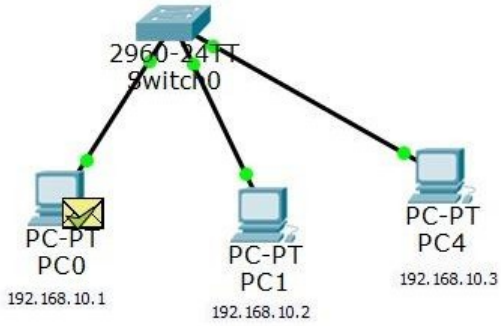


Screenshots/ Output:

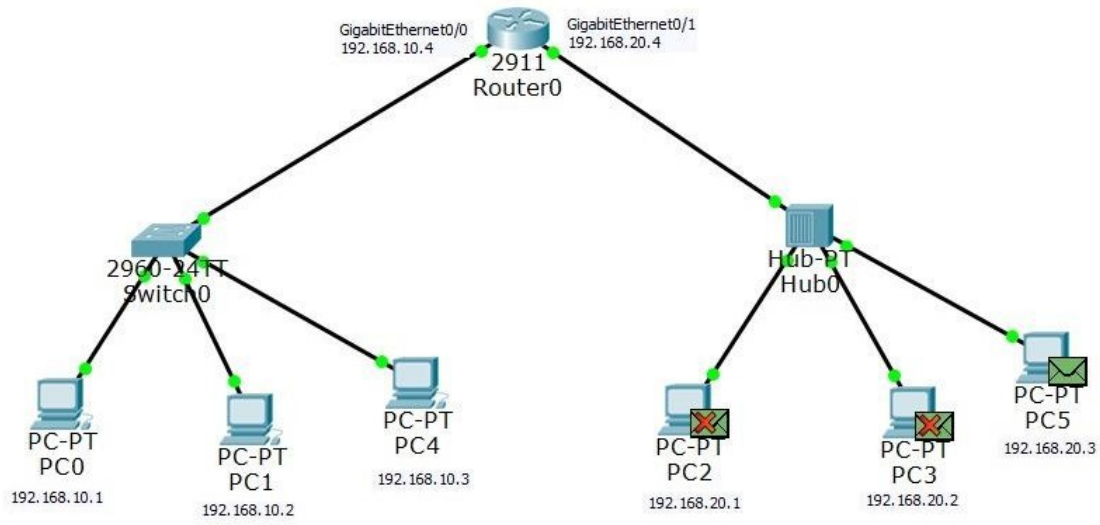


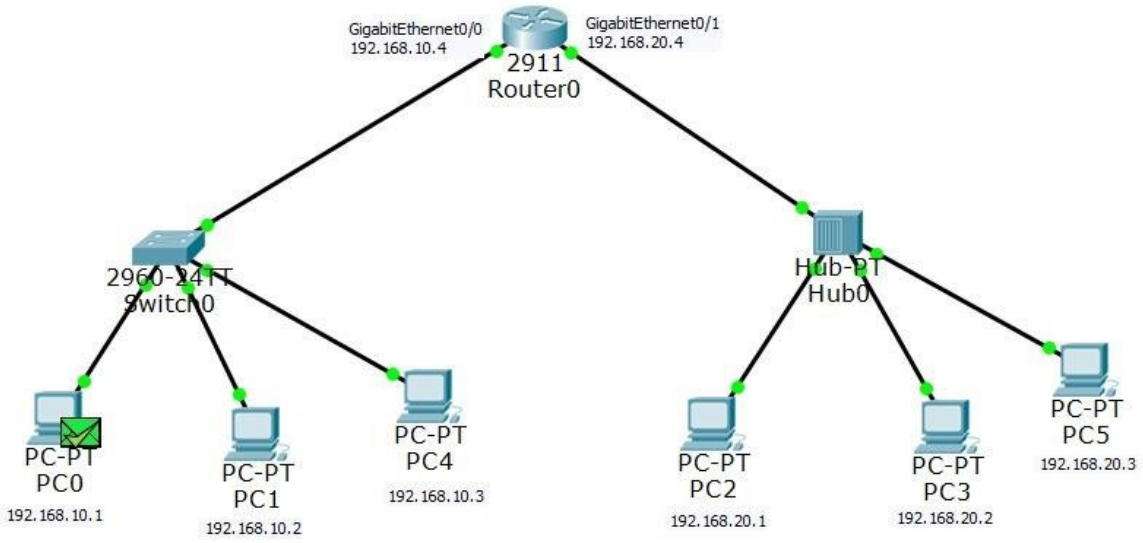






Updated topology





Observation:

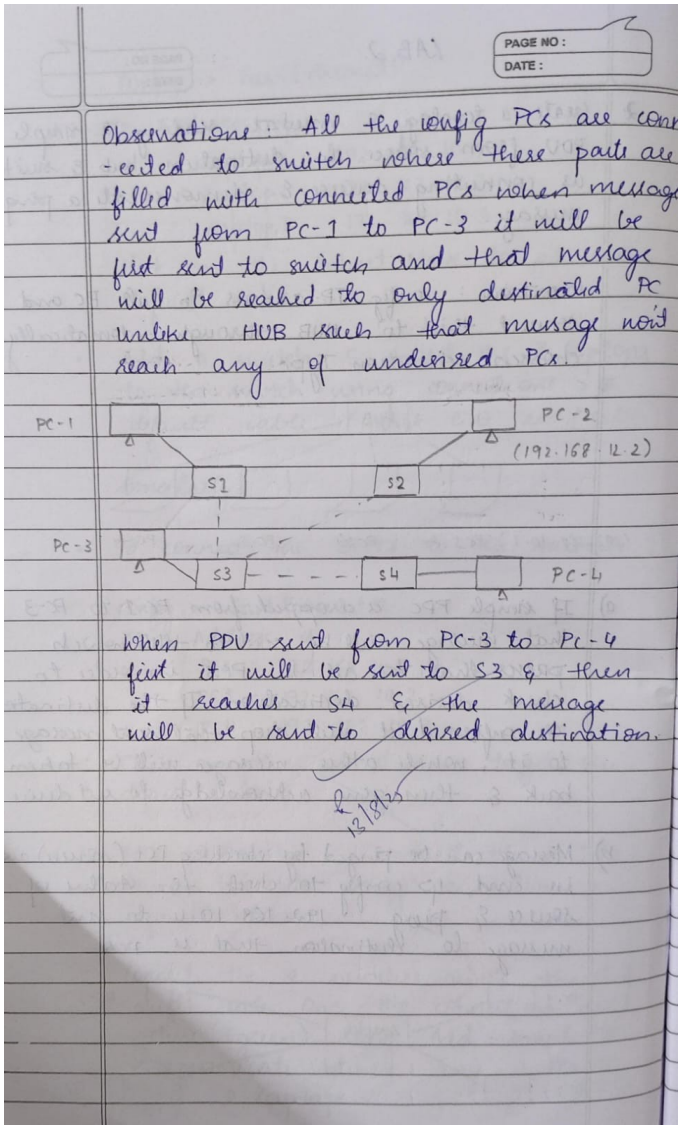
- In the hub-based topology, the PDU was broadcast to all ports, while the switch forwarded the PDU only to the destination MAC after learning addresses from incoming frames.
- Successful ICMP echo and echo-reply messages confirm that both devices enabled connectivity, with the switch demonstrating selective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

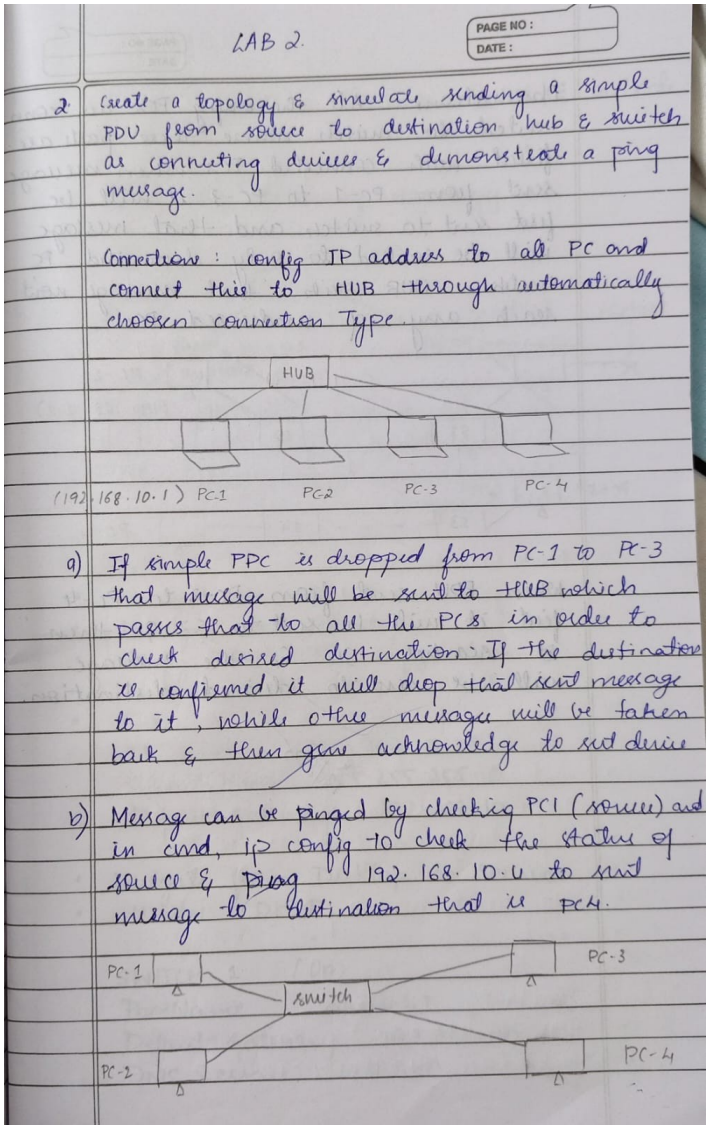
Program 2

Aim of the program:

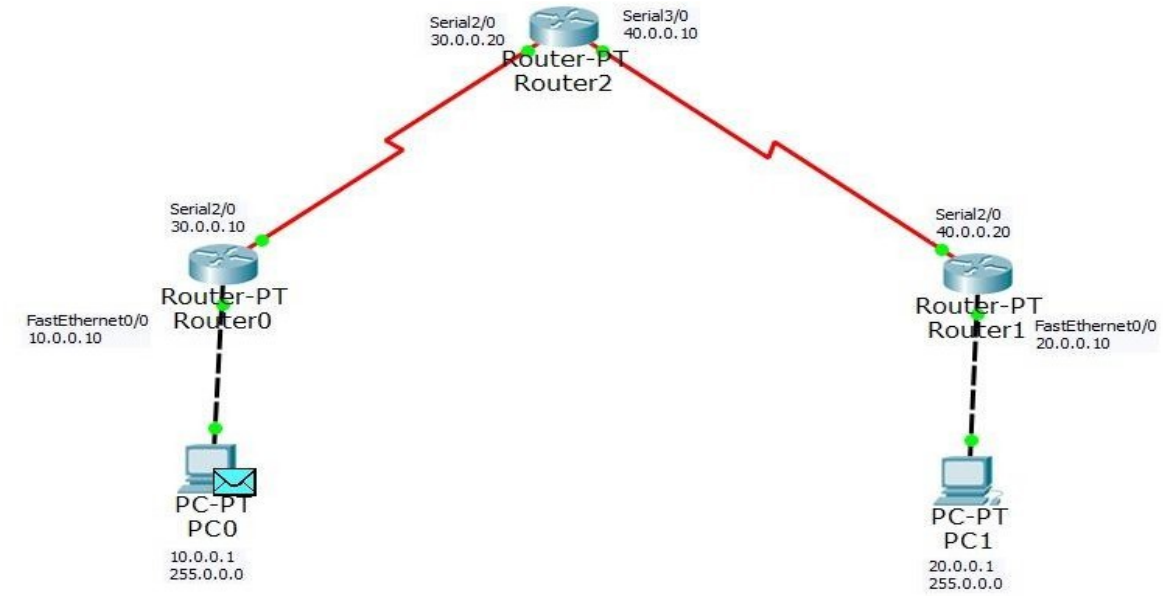
Configure default route, static route to the Router

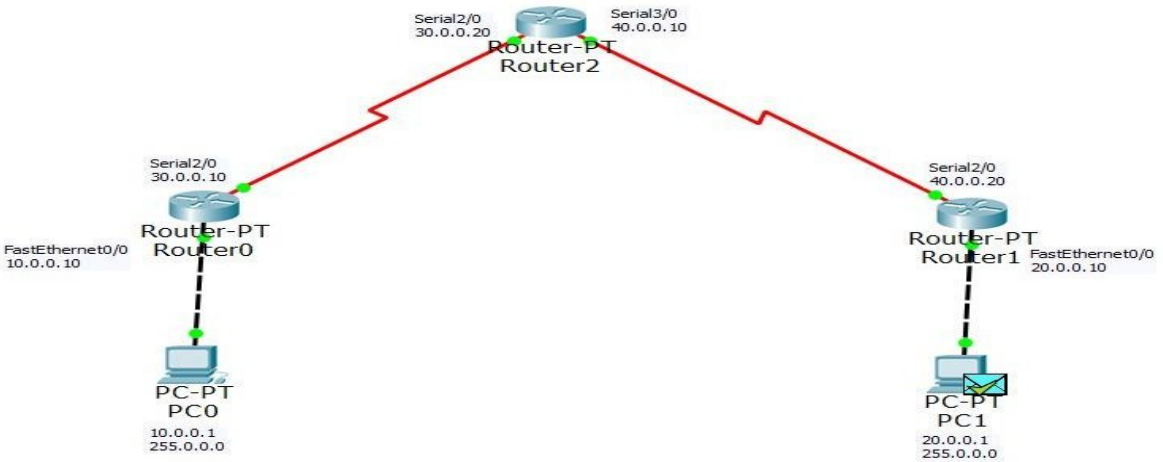
Procedure and topology:



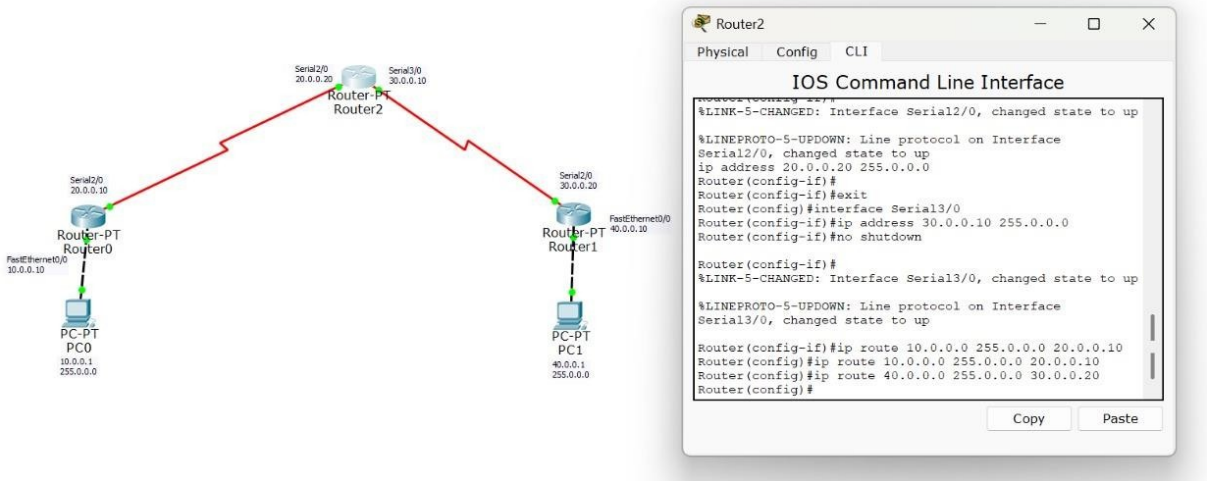


Screenshots/ Output:

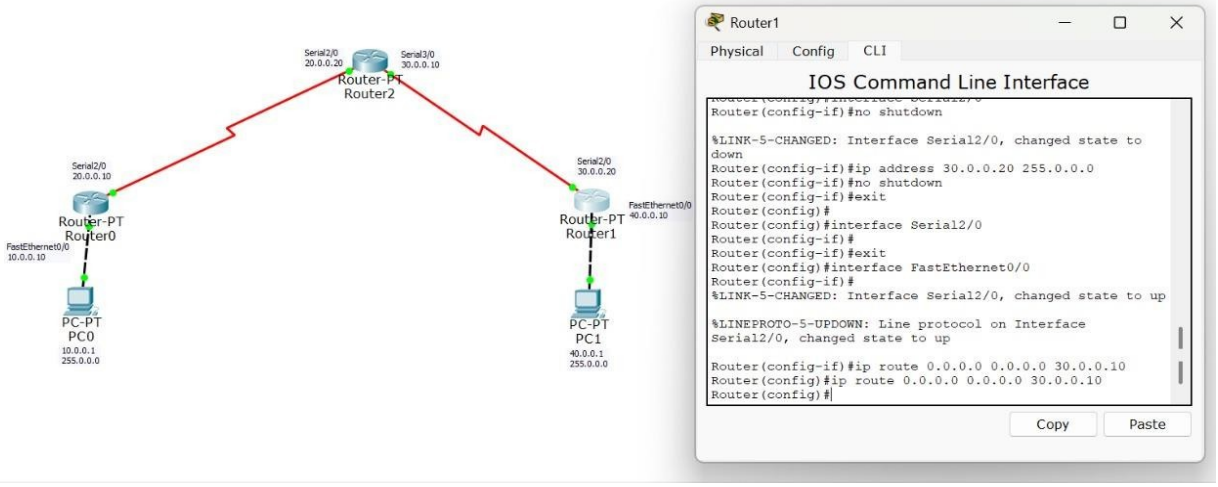




Static routing CLI commands:



Default routing CLI commands:



Observation:

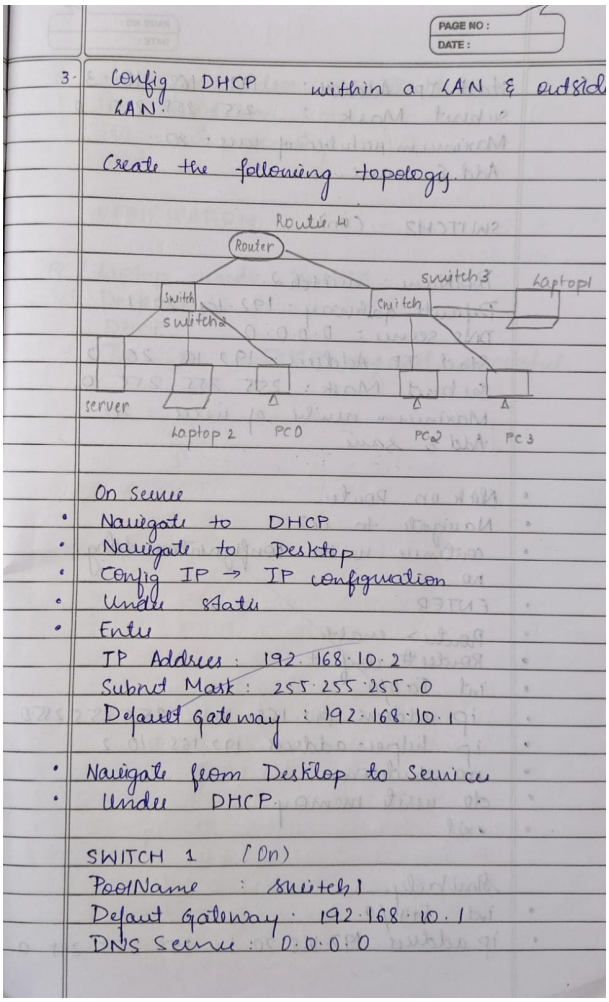
- The configured static and default routes correctly updated the router's routing table, enabling deterministic next-hop selection for remote networks.
- Successful ping tests verified that traffic was forwarded according to the static/default route entries, ensuring end-to-end reachability across different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

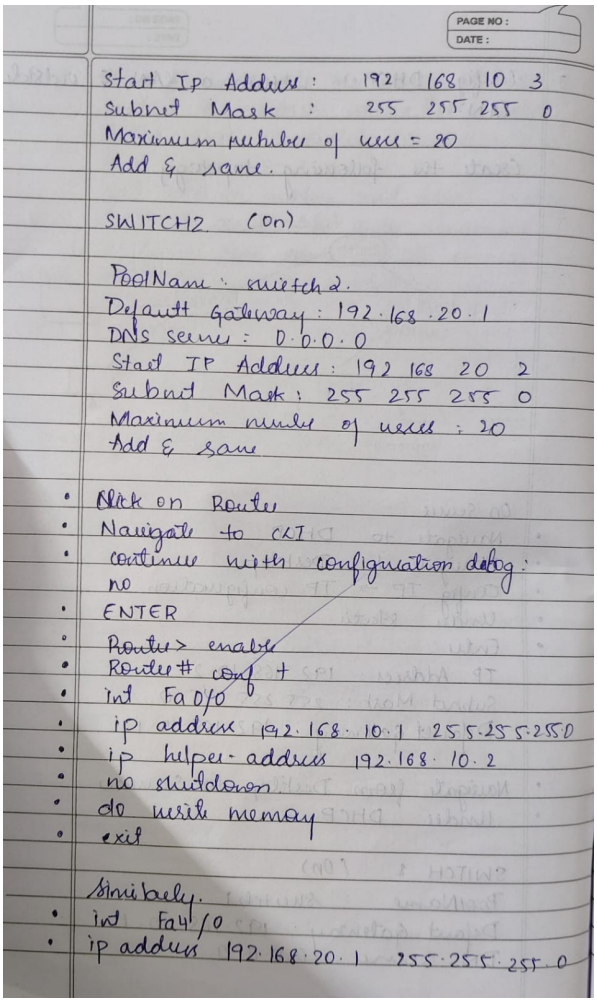
Program 3

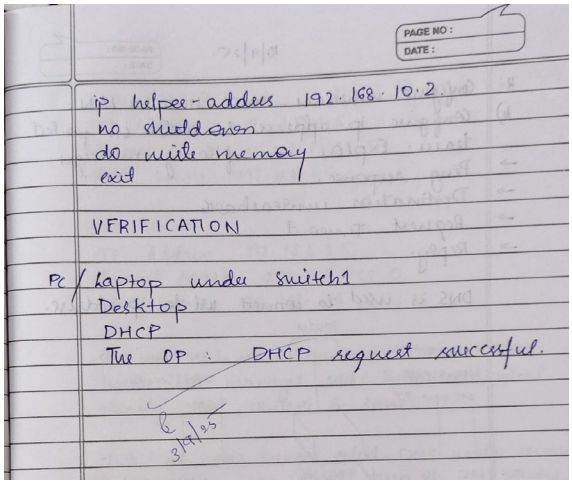
Aim of the program:

Configure DHCP within a LAN and outside LAN.

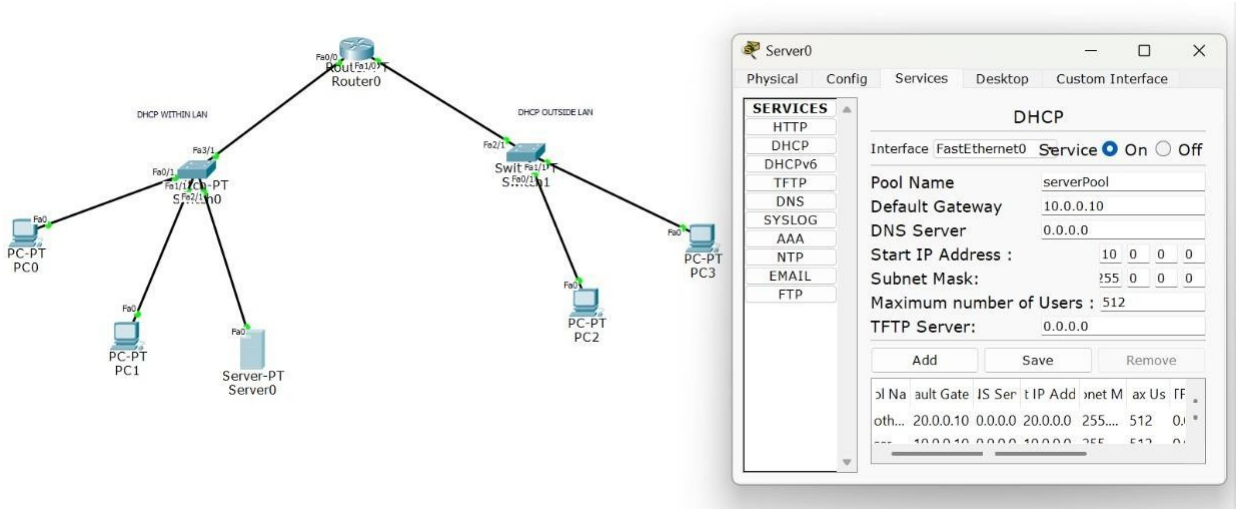
Procedure and topology:

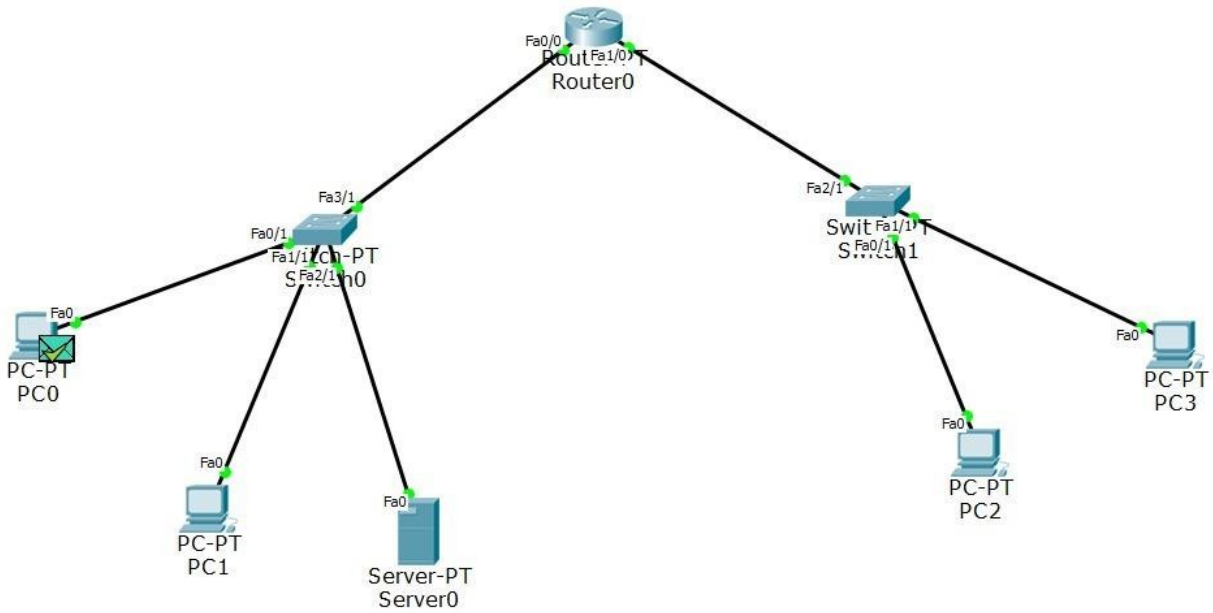






Screenshots/ Output:





Observation:

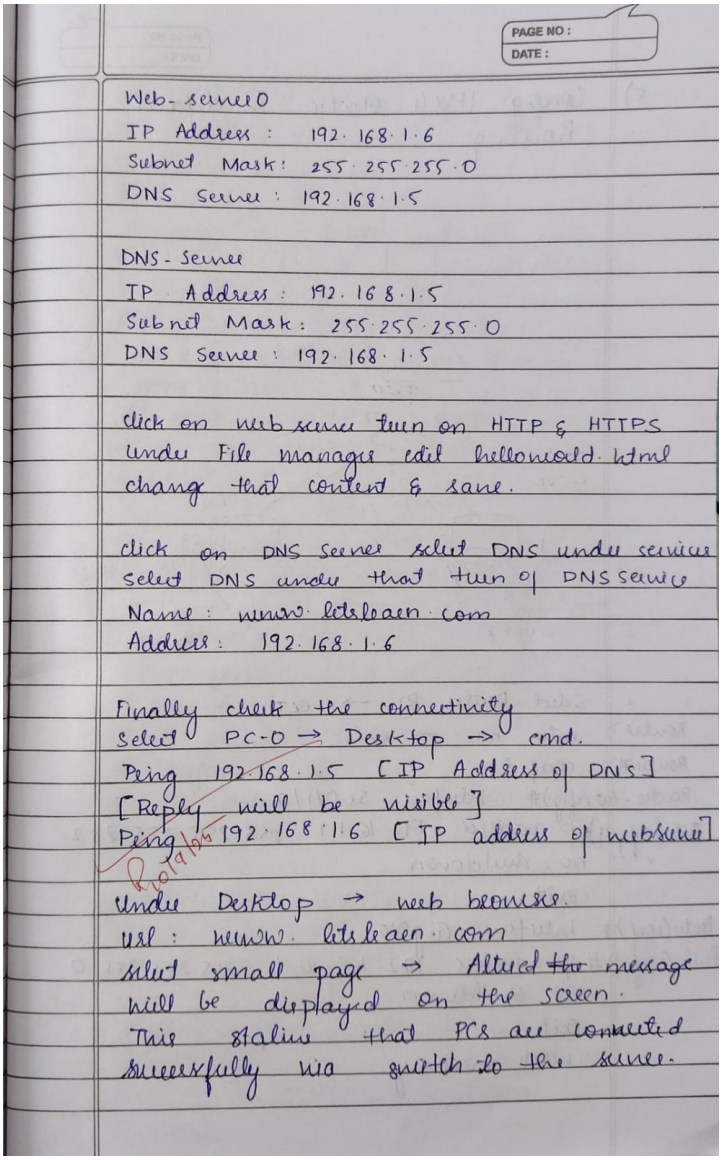
- The DHCP server successfully allocated IP addresses to clients within the LAN, confirming proper scope configuration and automatic distribution of network parameters.
- DHCP relay (IP Helper) enabled clients outside the LAN to obtain leases from the central DHCP server, demonstrating correct inter-network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

Program 4

Aim of the program:

Configure Web Server, DNS within a LAN.

Procedure and topology:



Web-server0
IP Address : 192.168.1.6
Subnet Mask: 255.255.255.0
DNS Server : 192.168.1.5

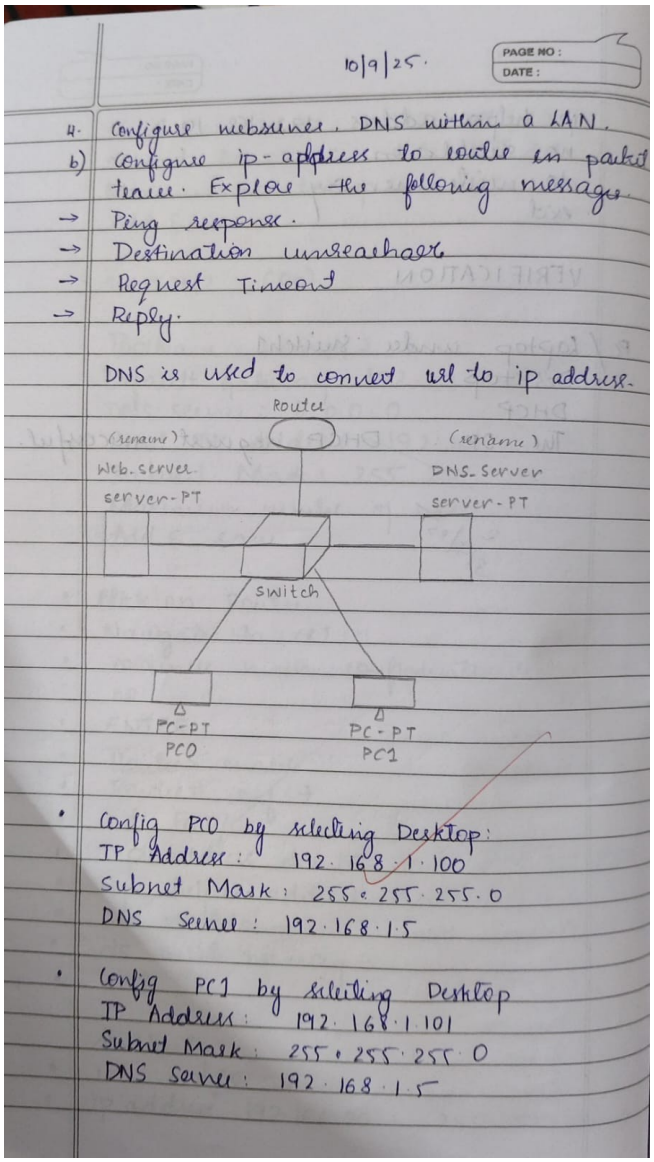
DNS-Server
IP Address : 192.168.1.5
Subnet Mask: 255.255.255.0
DNS Server : 192.168.1.5

click on web server turn on HTTP & HTTPS
Under File manager edit helloworld.html
change that content & save.

click on DNS Server select DNS under services
select DNS under that turn of DNS service
Name : www.bitsbaen.com
Address : 192.168.1.6

Finally check the connectivity
select PC-0 → Desktop → cmd.
Ping 192.168.1.5 [IP Address of DNS]
[Reply will be visible]
Ping www.bitsbaen.com 192.168.1.6 [IP address of webserver]

Under Desktop → web browser.
url : www.bitsbaen.com
click small page → Altered the message
will be displayed on the screen.
This status that PCs are connected
successfully via switch to the server.



10/9/25.

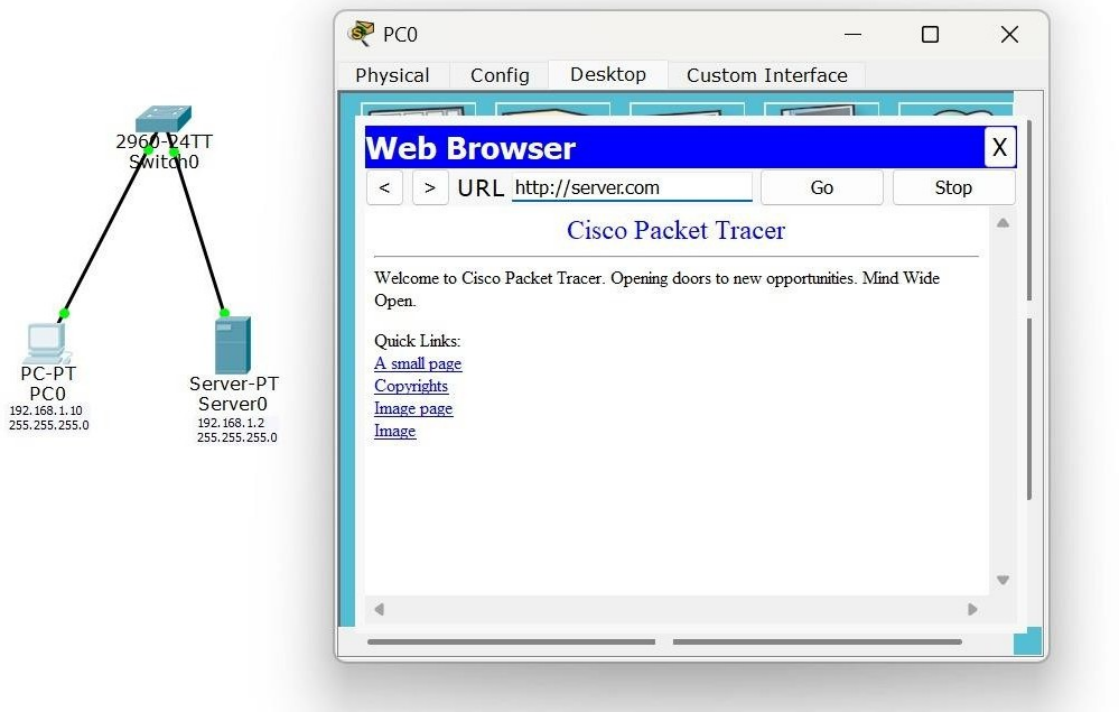
4. Configure webserver, DNS within a LAN.
b) Configure ip-address to route in packet tracer. Explore the following messages.
→ Ping response.
→ Destination unreachable.
→ Request Timeout
→ Reply.

DNS is used to convert url to ip address.

- Config PC0 by selecting Desktop:
IP Address : 192.168.1.100
Subnet Mask : 255.255.255.0
DNS Server : 192.168.1.5

- Config PC1 by selecting Desktop
IP Address : 192.168.1.101
Subnet Mask : 255.255.255.0
DNS Server : 192.168.1.5

Screenshots/ Output:



Observation:

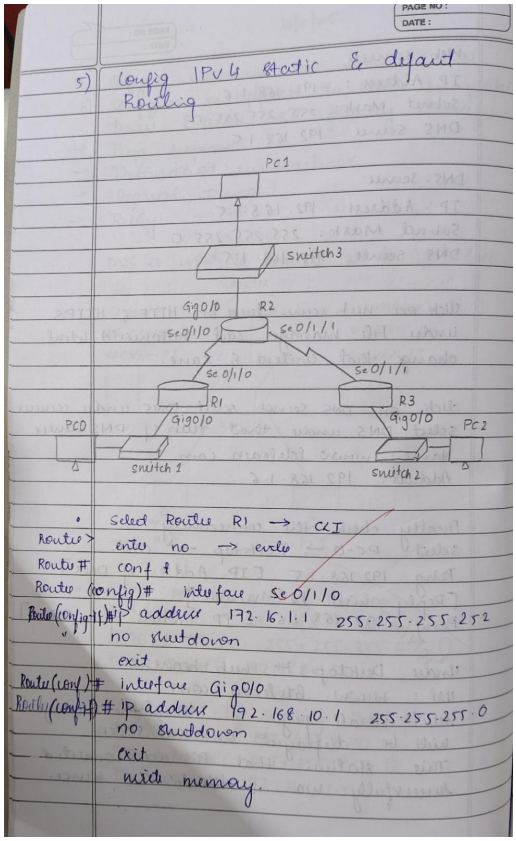
- The DNS server successfully resolved domain names to the corresponding web server's IP address, confirming proper hostname-to-IP mapping within the LAN.
- HTTP requests reached the web server using the DNS-resolved address, validating correct server configuration and internal LAN communication.

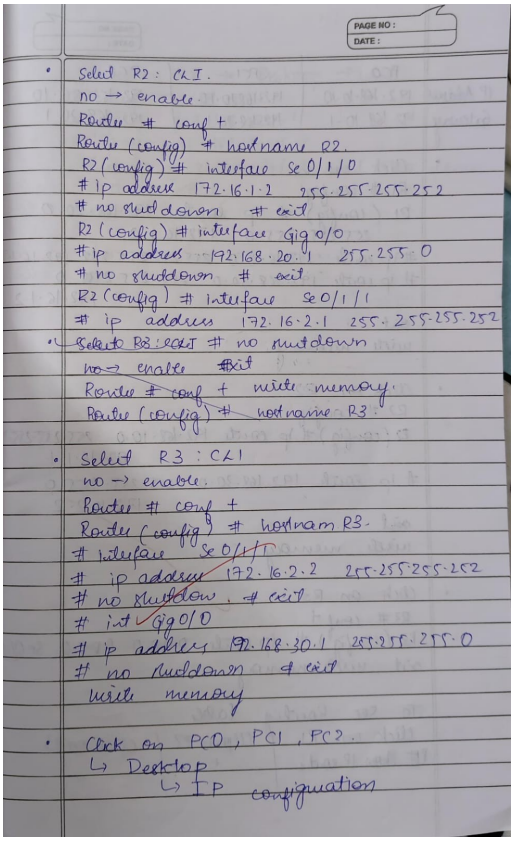
Program 5

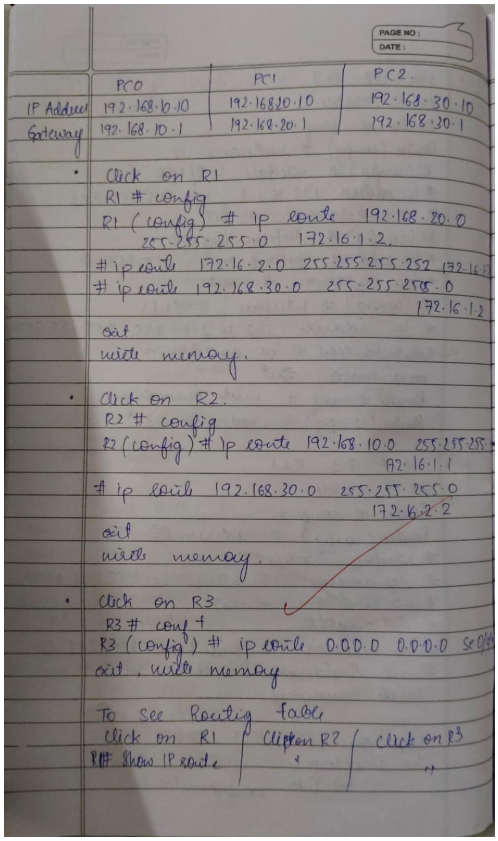
Aim of the program:

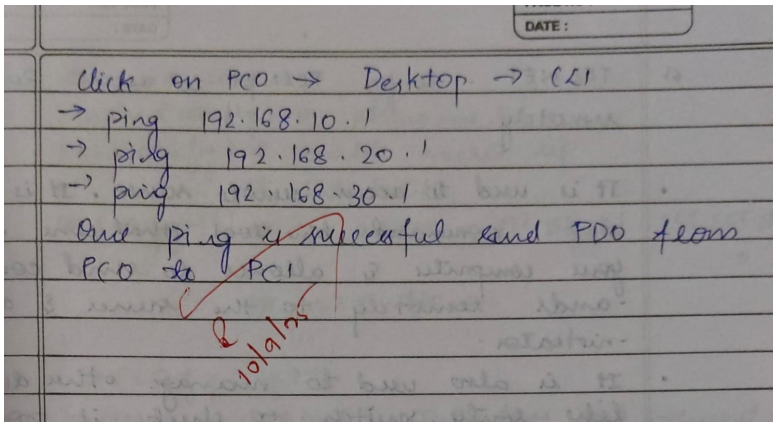
To understand the operation of TELNET by accessing the router in server room from a PC in IT office

Procedure and topology:

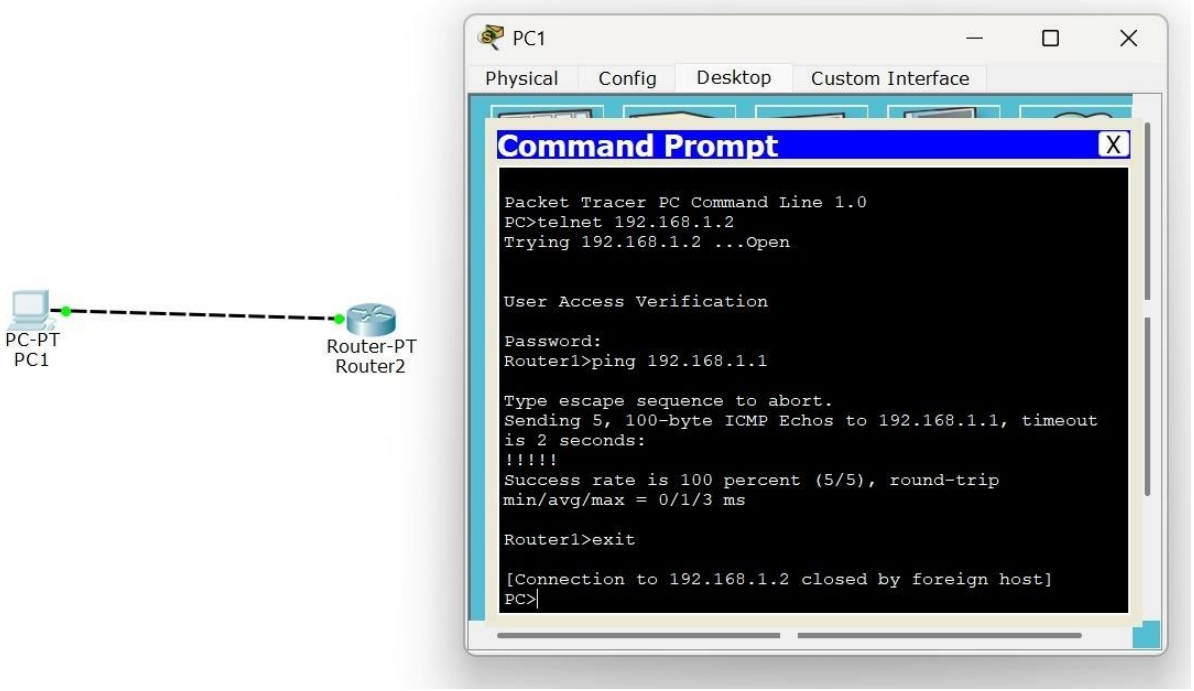








Screenshots/ Output:



Observation:

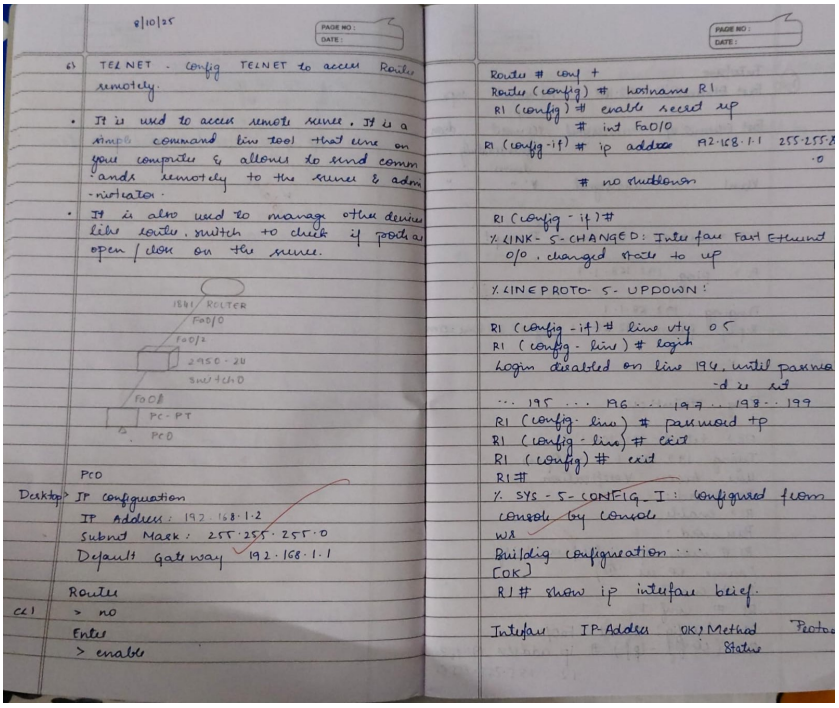
- The Telnet session successfully established a remote CLI connection to the router, confirming proper VTY line configuration and IP reachability between the IT office PC and the server room router.
- Command execution over the Telnet session demonstrated reliable remote device management.

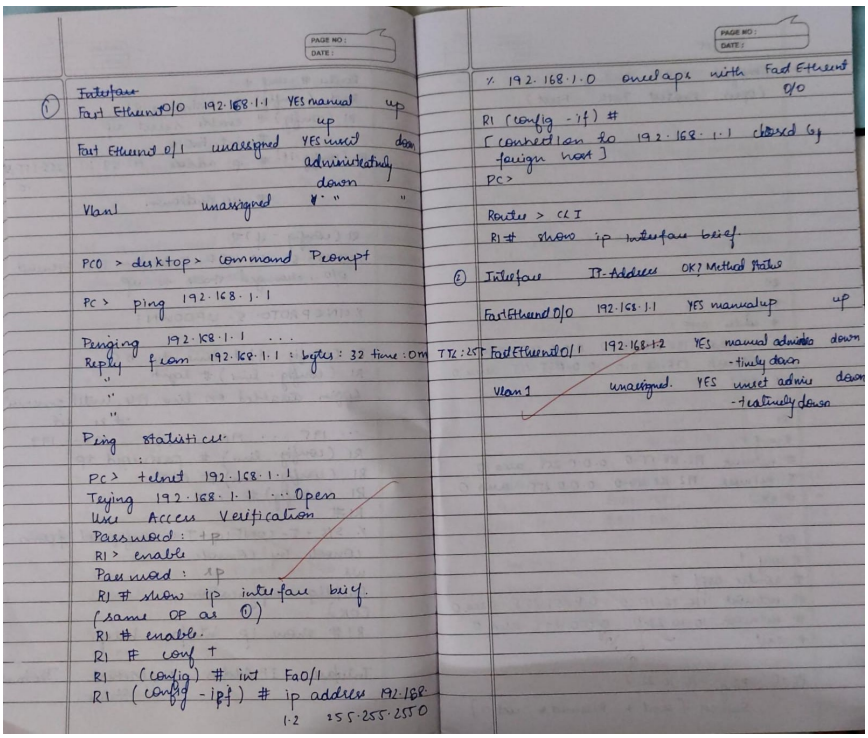
Program 6

Aim of the program:

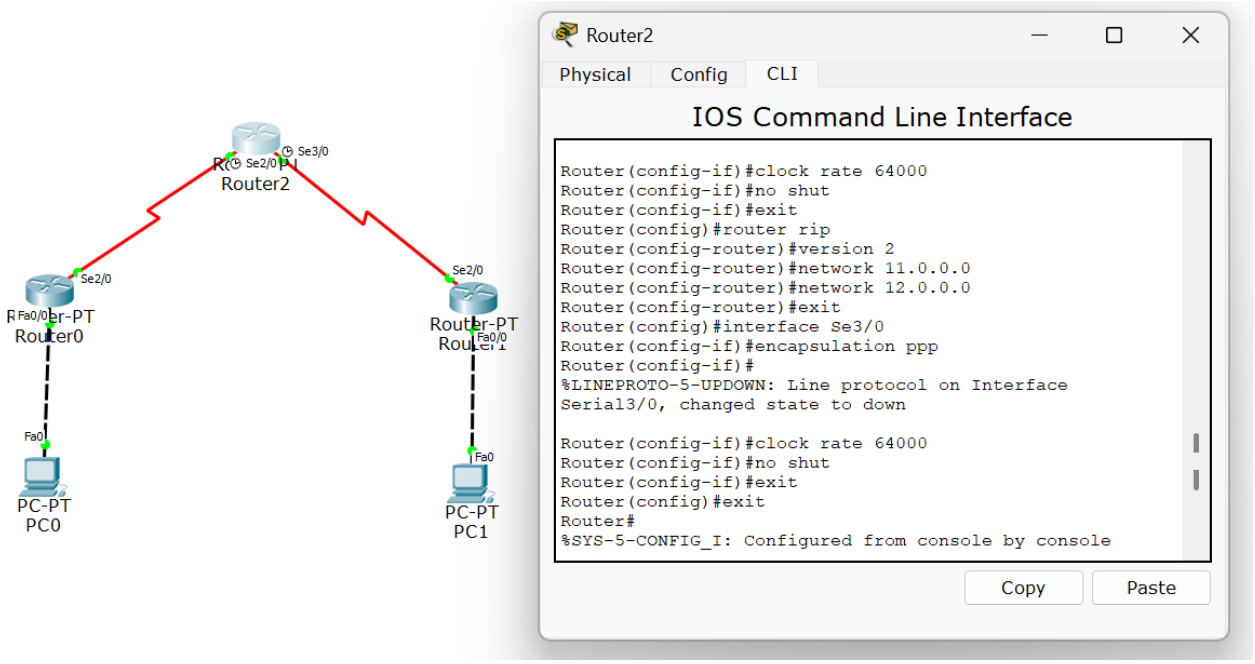
Configure RIP routing Protocol in Routers

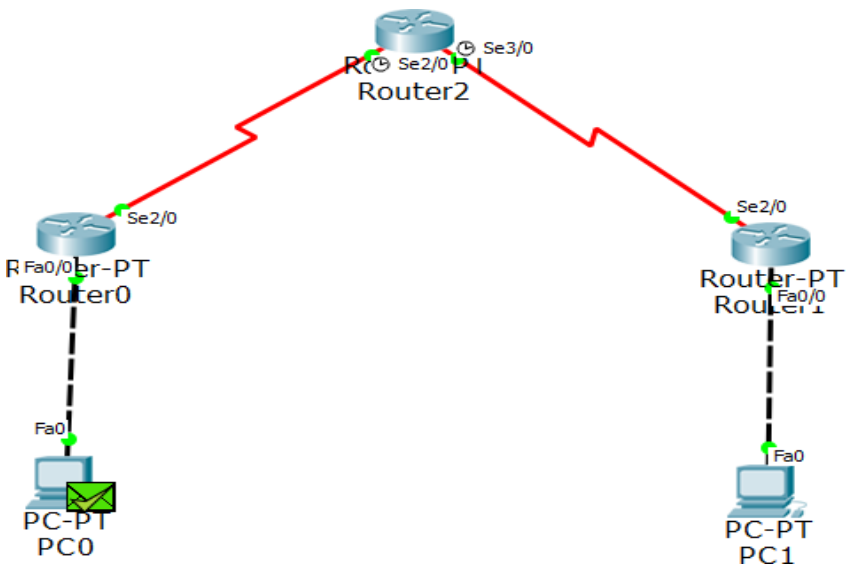
Procedure and topology:





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

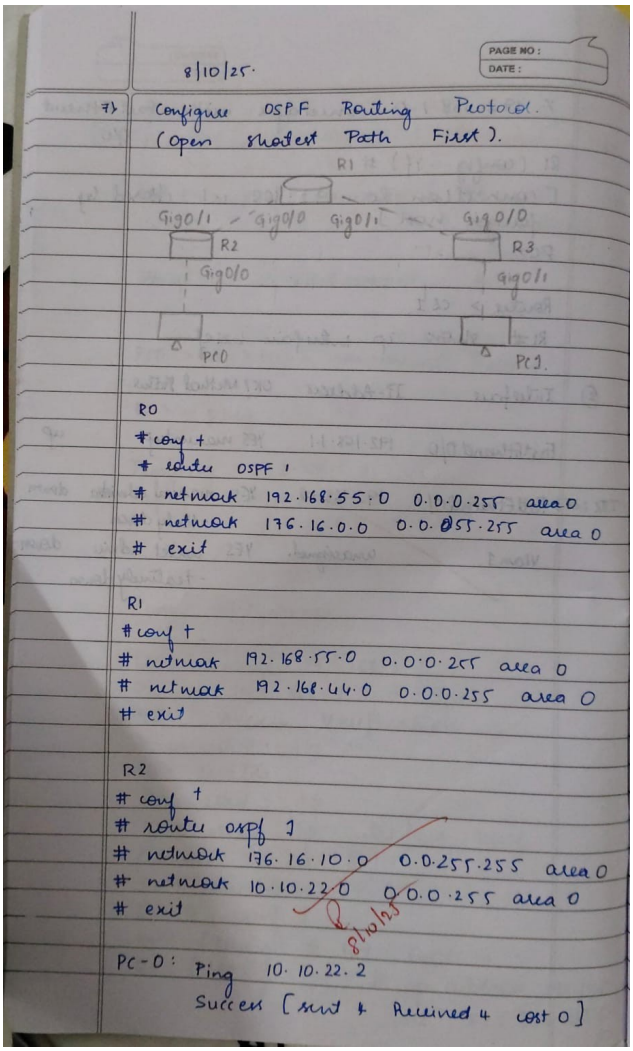
connectivity maintained by periodic RIP updates and route propagation.

Program 7

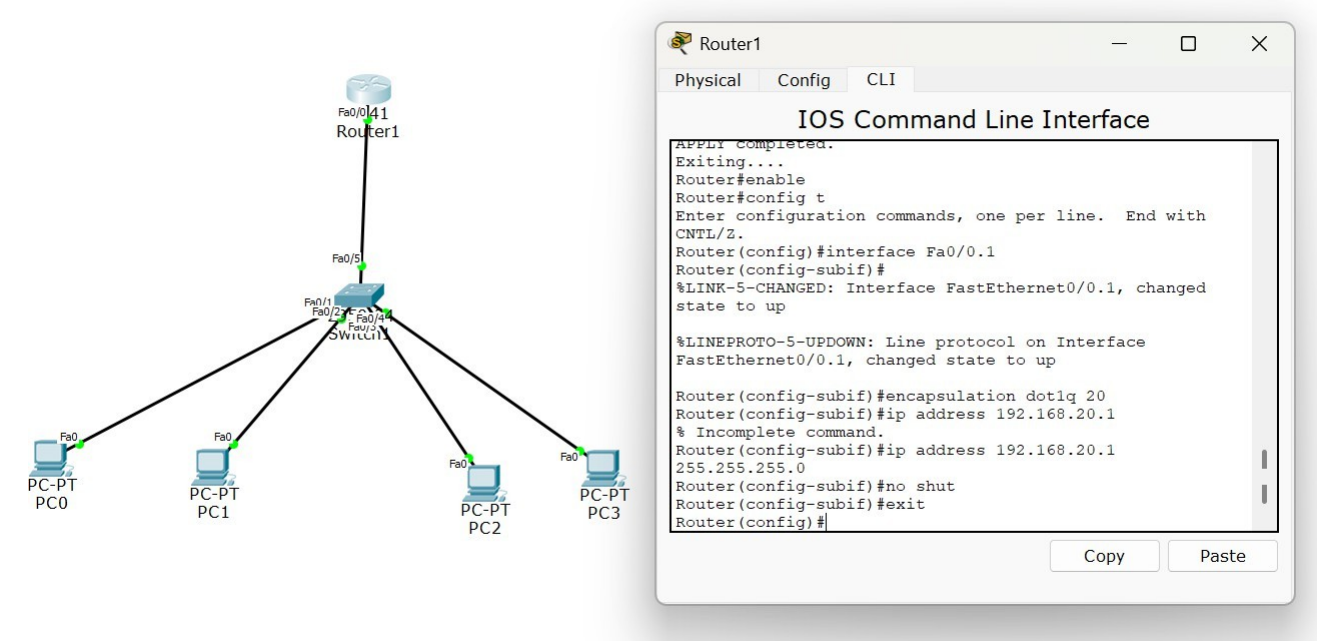
Aim of the program:

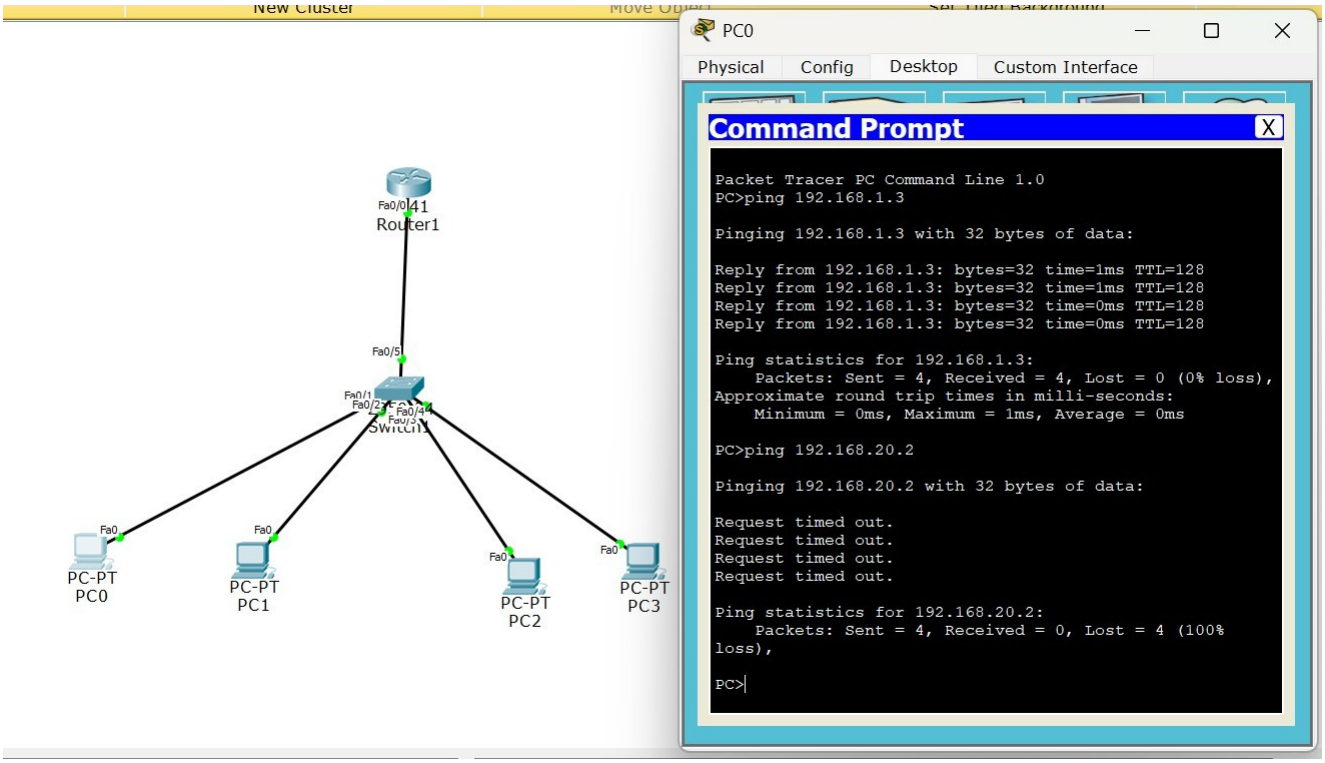
To construct a VLAN and make the PCs communicate among a VLAN

Procedure and topology:



Screenshots/ Output:





Observation:

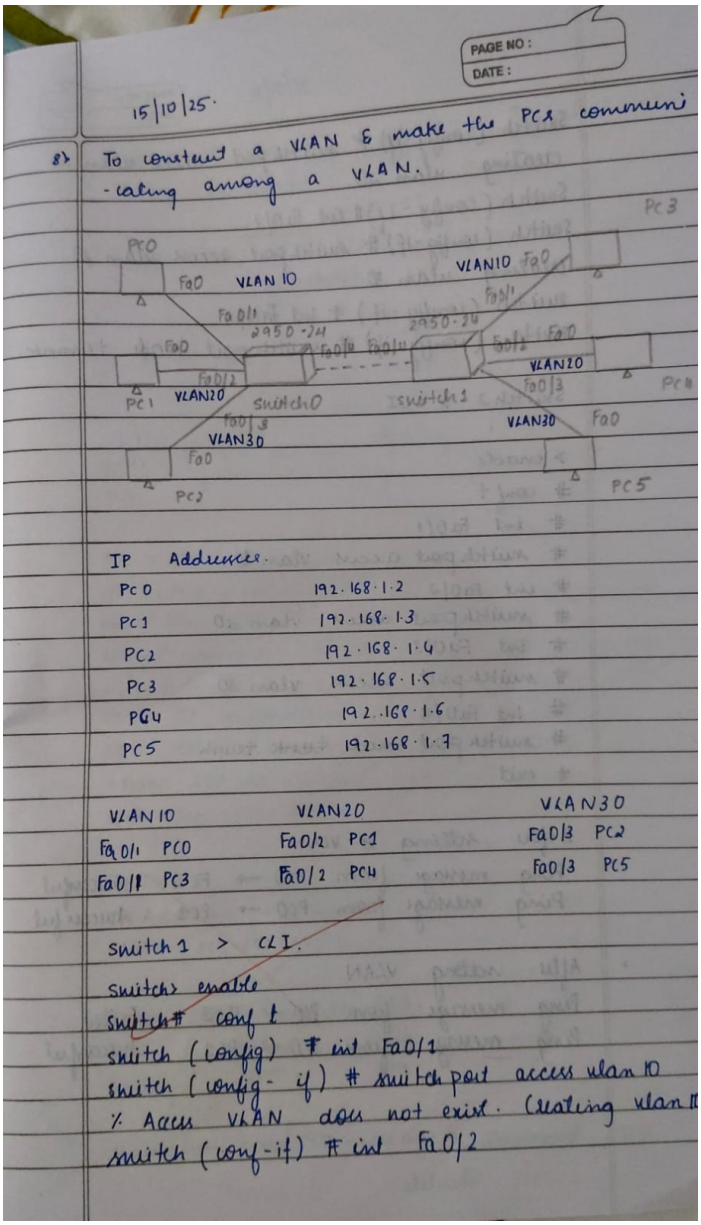
- VLAN segmentation successfully separated broadcast domains, and switch ports were correctly assigned to their respective VLAN IDs using access mode configuration.
- Inter-VLAN communication was achieved through the Layer-3 device, and successful ping tests confirmed proper VLAN membership, tagging,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

Program 8

Aim of the program:

To construct a WLAN and make the nodes communicate wirelessly

Procedure and topology:



15/10/25.

8) To construct a VLAN & make the PCs communicating among a VLAN.

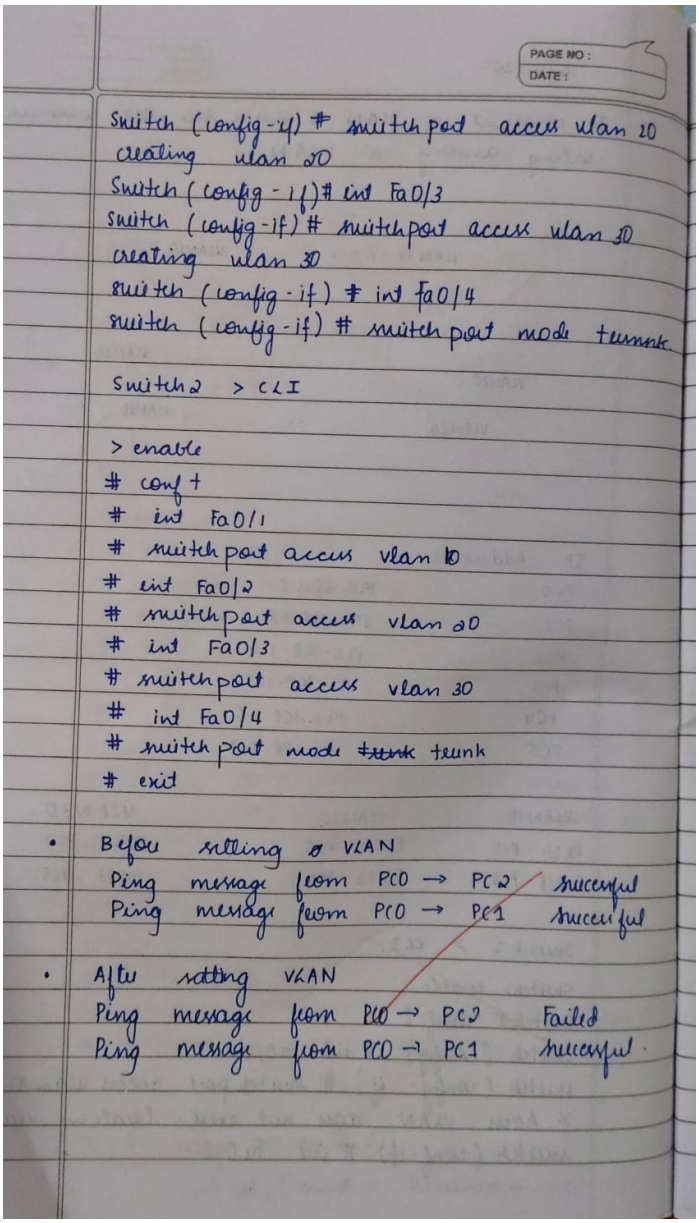
PC3
PC0
Fa0 VLAN 10
VLAN10 Fa0
Fa0/1
2950-24
Fa0 Fa0/4 Fa0/1 2950-24
Fa0
Fa0/2
VLAN20
PC1 VLAN20 Switch0 Switch1
Fa0/3
VLAN30
Fa0
Fa0
PC2
PC5

IP Addresses.

PC0	192.168.1.2
PC1	192.168.1.3
PC2	192.168.1.4
PC3	192.168.1.5
PC4	192.168.1.6
PC5	192.168.1.7

VLAN 10	VLAN 20	VLAN 30
Fa0/1 PC0	Fa0/2 PC1	Fa0/3 PC2
Fa0/1 PC3	Fa0/2 PC4	Fa0/3 PC5

Switch 1 > CLI.
Switch> enable
Switch# conf t
Switch (config) # int Fa0/1
Switch (config-if) # switchport access vlan 10
% Access VLAN does not exist. Creating vlan 1
Switch (conf-if) # int Fa0/2



Switch (config-if) # switchport access vlan 20
creating vlan 20
Switch (config-if)# int Fa0/3
Switch (config-if) # switchport access vlan 30
creating vlan 30
Switch (config-if) # int Fa0/4
Switch (config-if) # switchport mode trunk

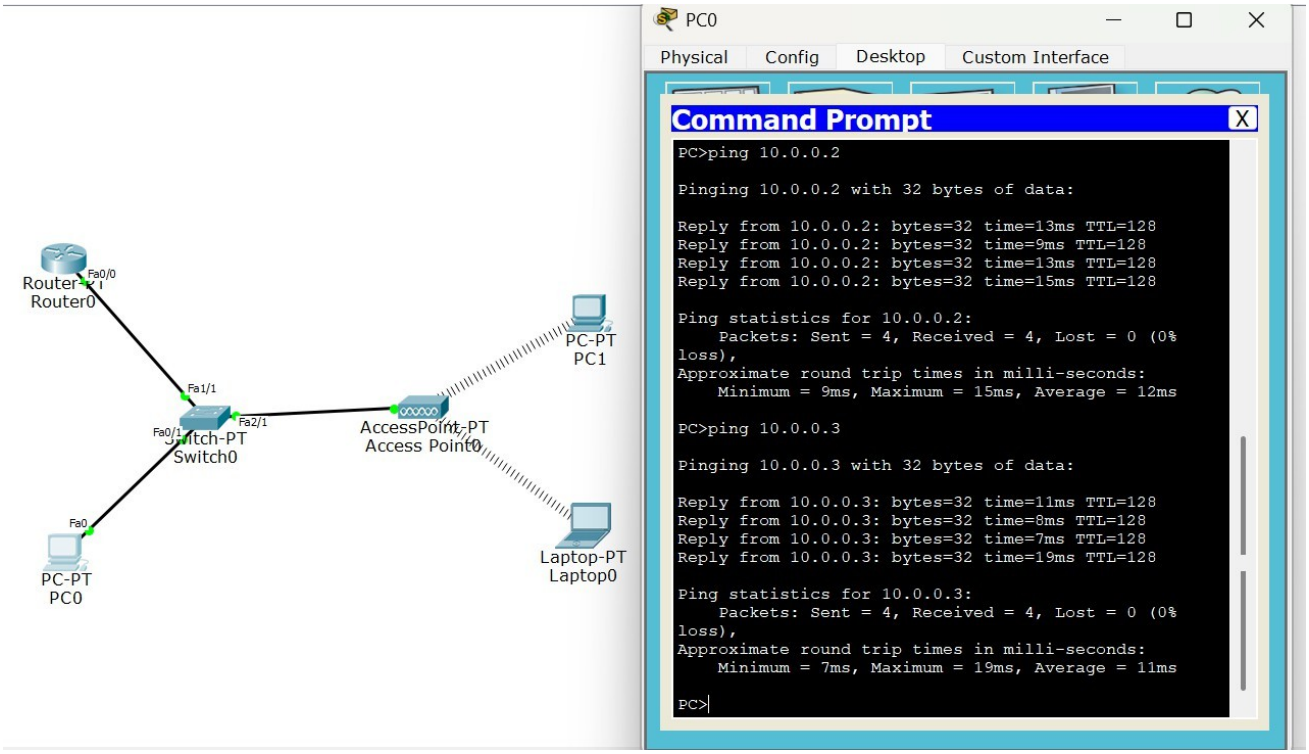
Switch2 > CLI

> enable
conf 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~~ trunk
exit

- Before 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:

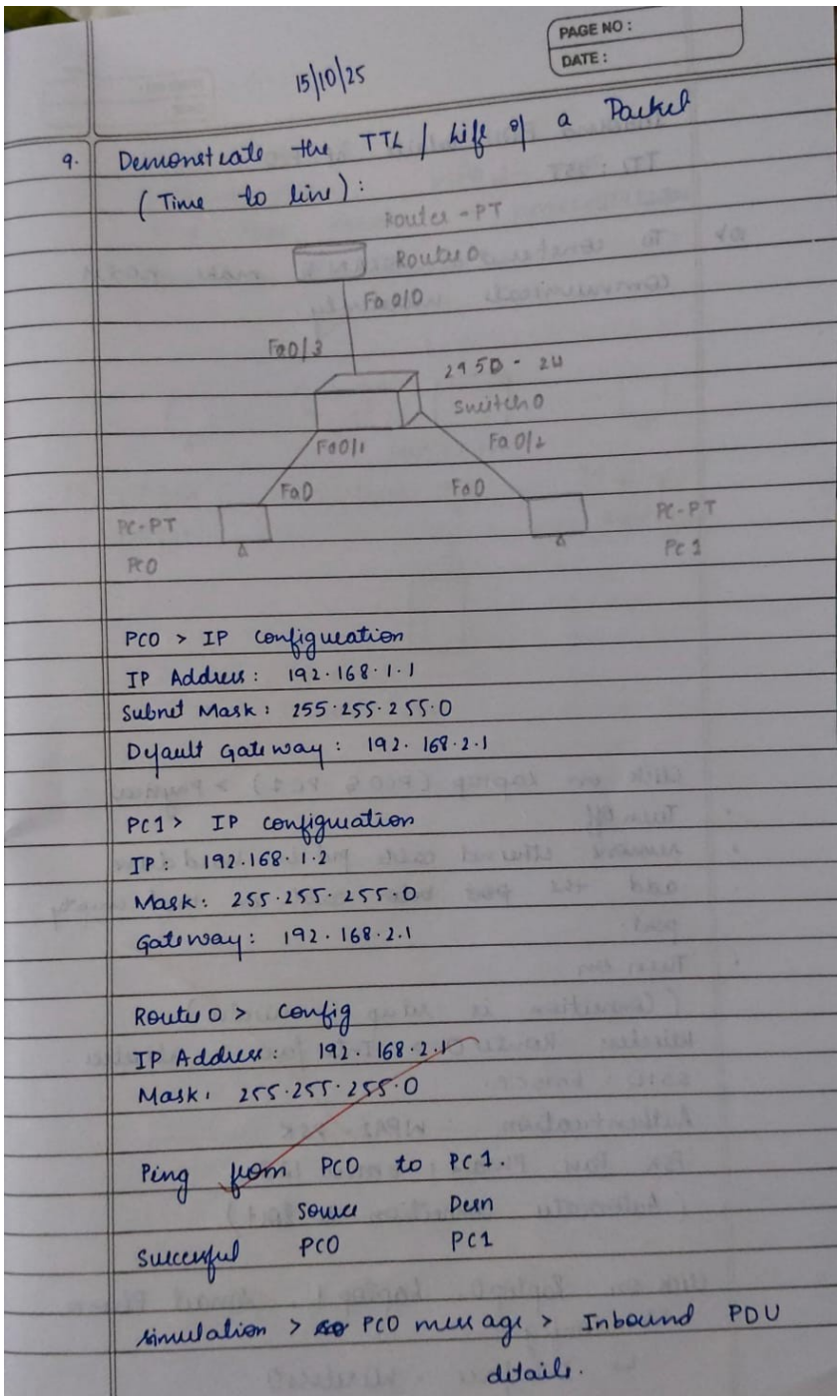
- The WLAN configuration enabled wireless nodes to associate with the access point using the configured SSID and security settings, confirming proper authentication and signal coverage.
- Successful ping communication between wireless devices verified stable wireless Connectivity.

Program 9

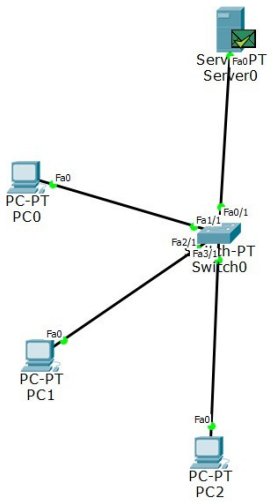
Aim of the program:

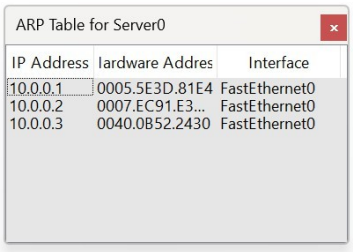
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

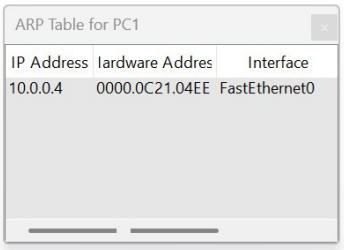
Procedure and topology:

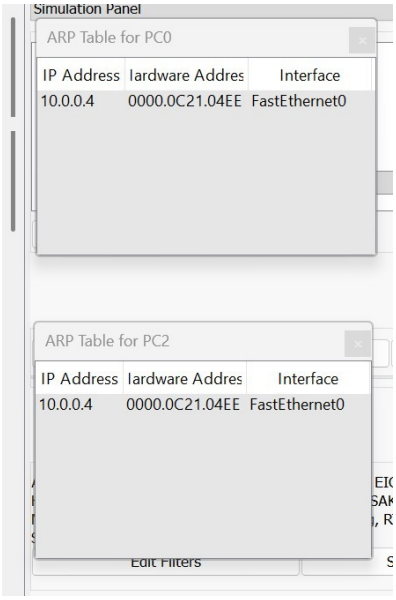


Screenshots/ Output:









Observation:

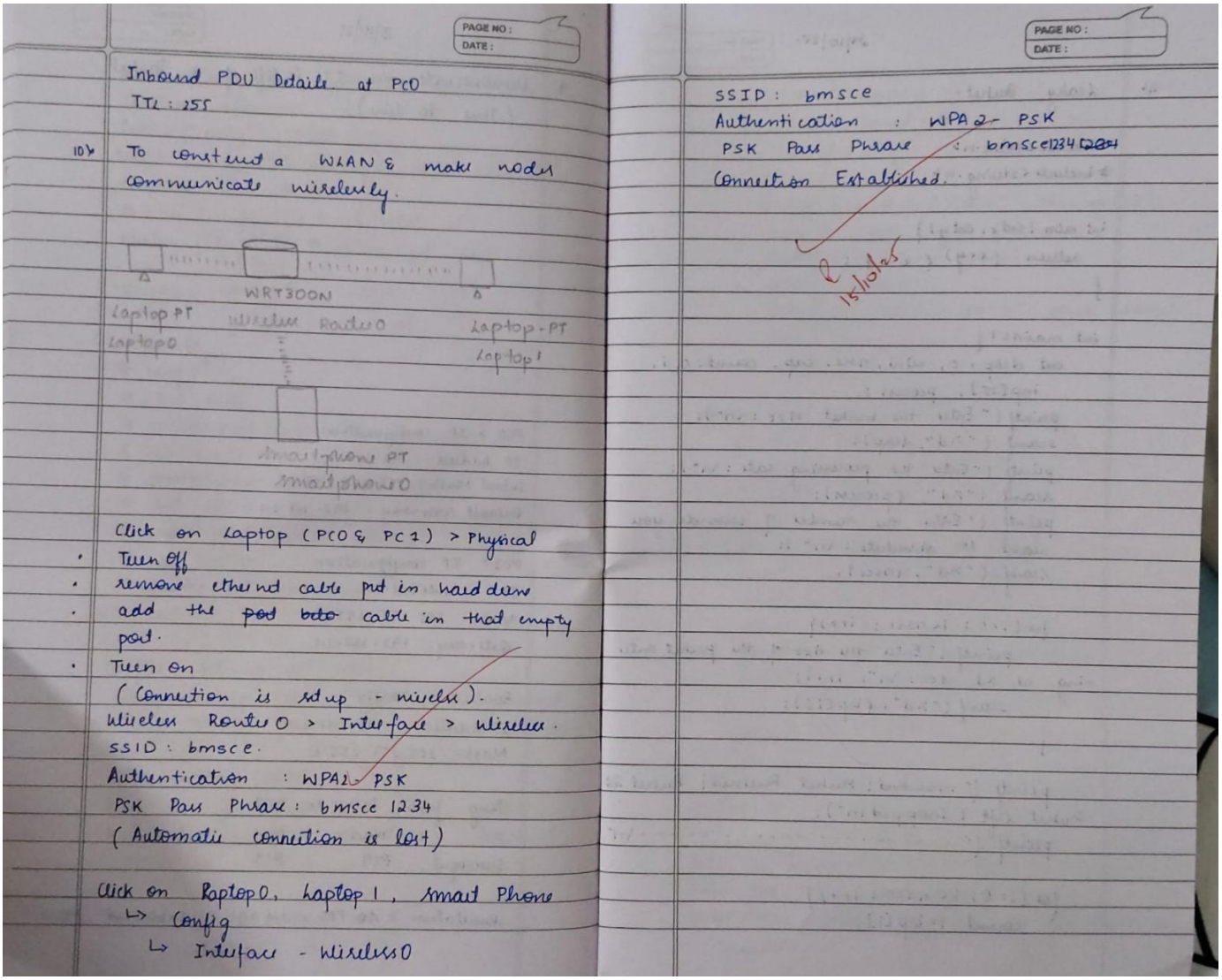
- ARP successfully resolved the destination IP address to its corresponding MAC address, as seen from ARP request and reply exchanges between LAN hosts.
- The populated ARP tables and successful ping communication confirmed correct layer-2 addressing, frame forwarding, and basic LAN operation.

Program 10

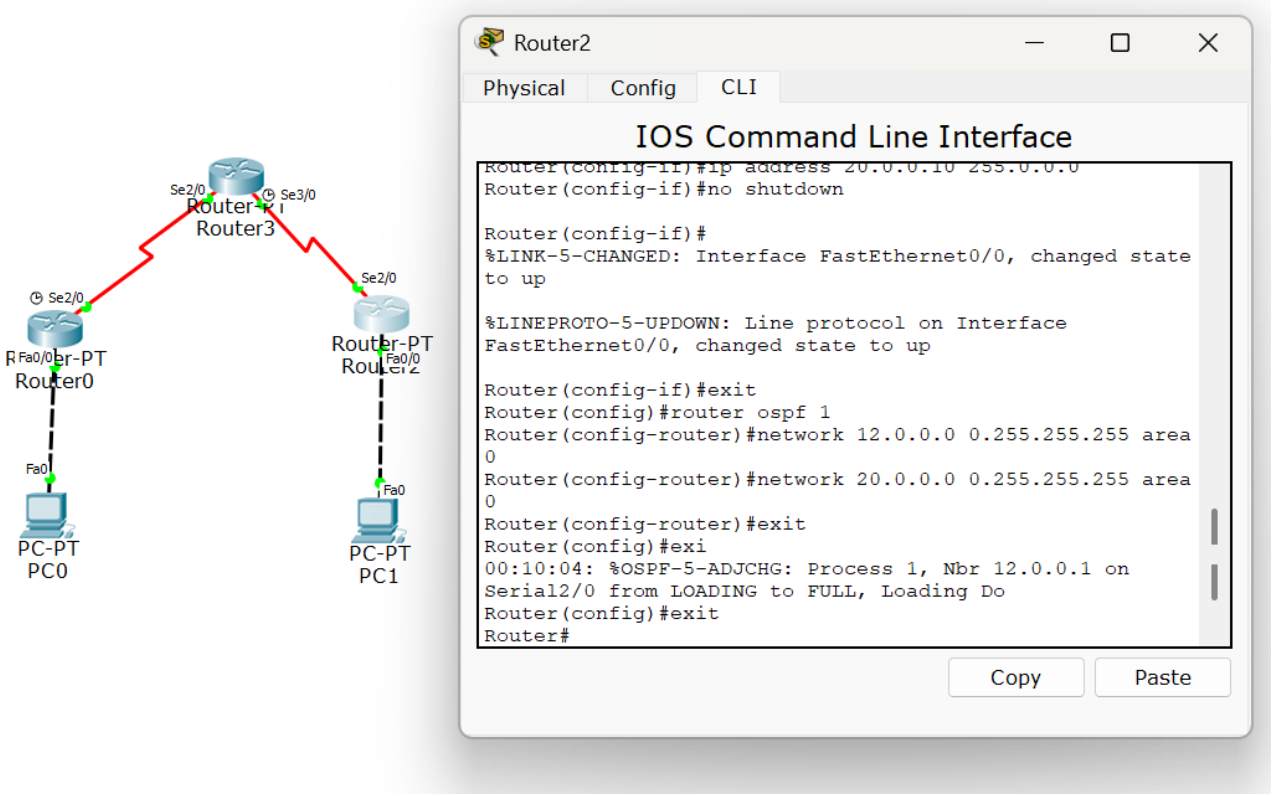
Aim of the program:

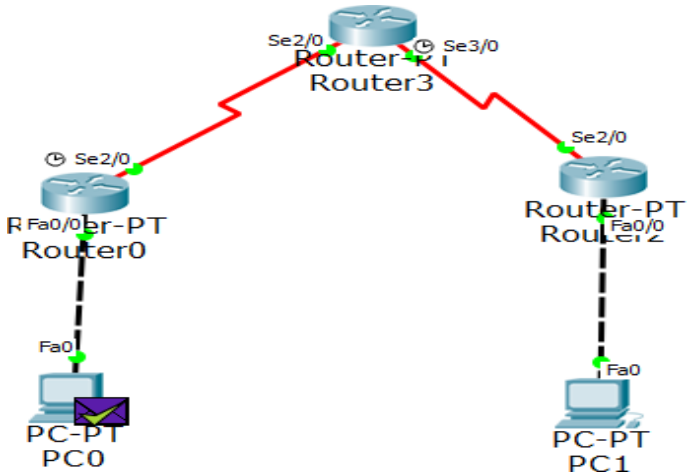
Configure OSPF routing protocol

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

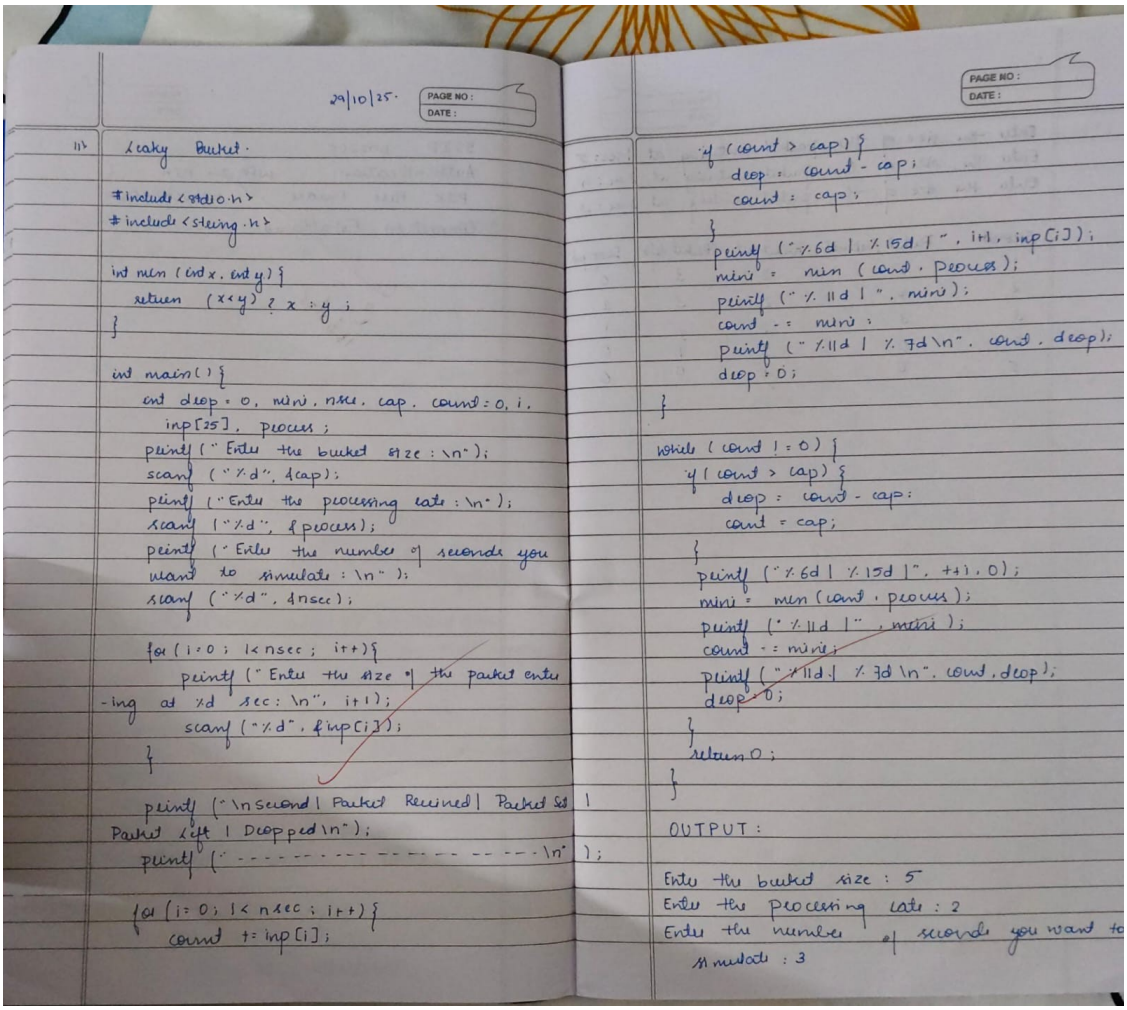
path selection and dynamic route learning through OSPF.

Program 11

Aim of the program:

Demonstrate the TTL/ Life of a Packet

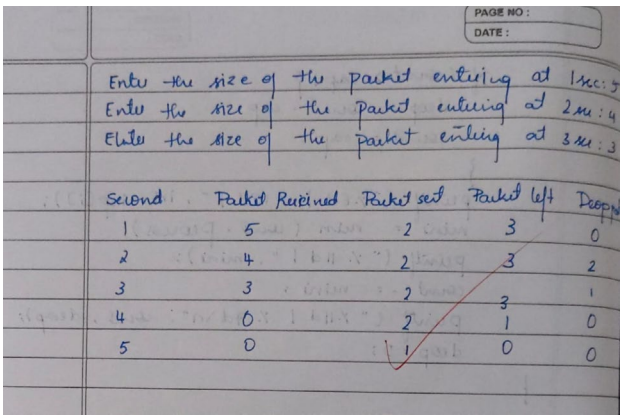
Procedure and topology:



```
29/10/25.

11) Leaky Bucket.

#include <stdio.h>
#include <stdlib.h>

int min (int x, int y){
  return (x<y) ? x : y ;
}

int main(){
  int drop = 0, mini, nsec, cap, count = 0, i,
    inp[25], process ;
  printf (" Enter the bucket size : \n");
  scanf (" %d", &cap);
  printf (" Enter the processing rate : \n");
  scanf (" %d", &process);
  printf (" Enter the number of seconds you
  want to simulate : \n" );
  scanf (" %d", &nsec);

  for (i=0 ; i<nsec ; i++){
    printf (" Enter the size of the packet enter
-ing at %d sec: \n", i+1);
    scanf (" %d", &inp[i]);
  }

  printf (" \n Second | Packet Recieved | Packet Sent |
Packet Left | Dropped \n");
  printf (" ---------- ---------- ---------- ---------- ---------- \n" );

  for (i=0; i<nsec ; i++){
    count += inp[i];
    if (count > cap){
      drop = count - cap;
      count = cap;
    }
    printf (" %6d | %15d | ", i+1, inp[i]);
    mini = min (count, process);
    printf (" %11d | ", mini);
    count -= mini ;
    printf (" %11d | %7d \n", count, drop);
    drop = 0;
  }

  while (count != 0){
    if (count > cap){
      drop = count - cap;
      count = cap;
    }
    printf (" %6d | %15d | ", i+1, 0);
    mini = min (count, process);
    printf (" %11d | ", mini);
    count -= mini;
    printf (" %11d | %7d \n", count, drop);
    drop = 0;
  }
  return 0;
}
```

OUTPUT :

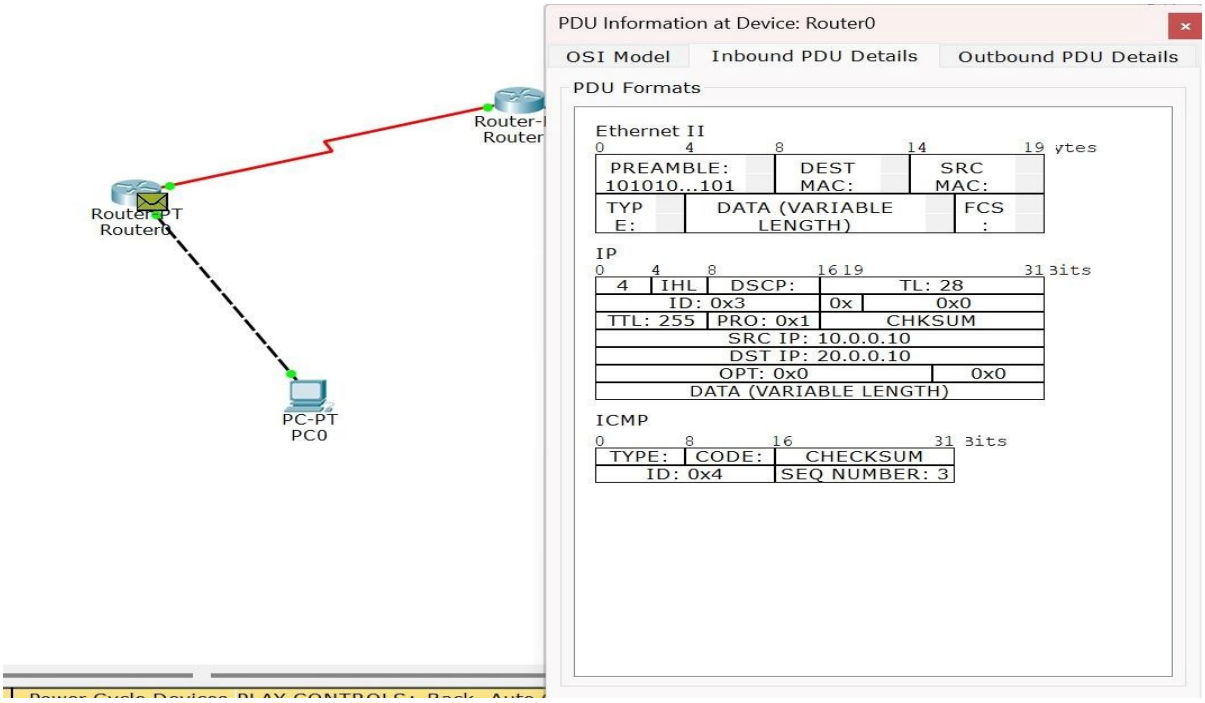
```
Enter the bucket size : 5
Enter the processing rate : 2
Enter the number of seconds you want to
simulate : 3
```

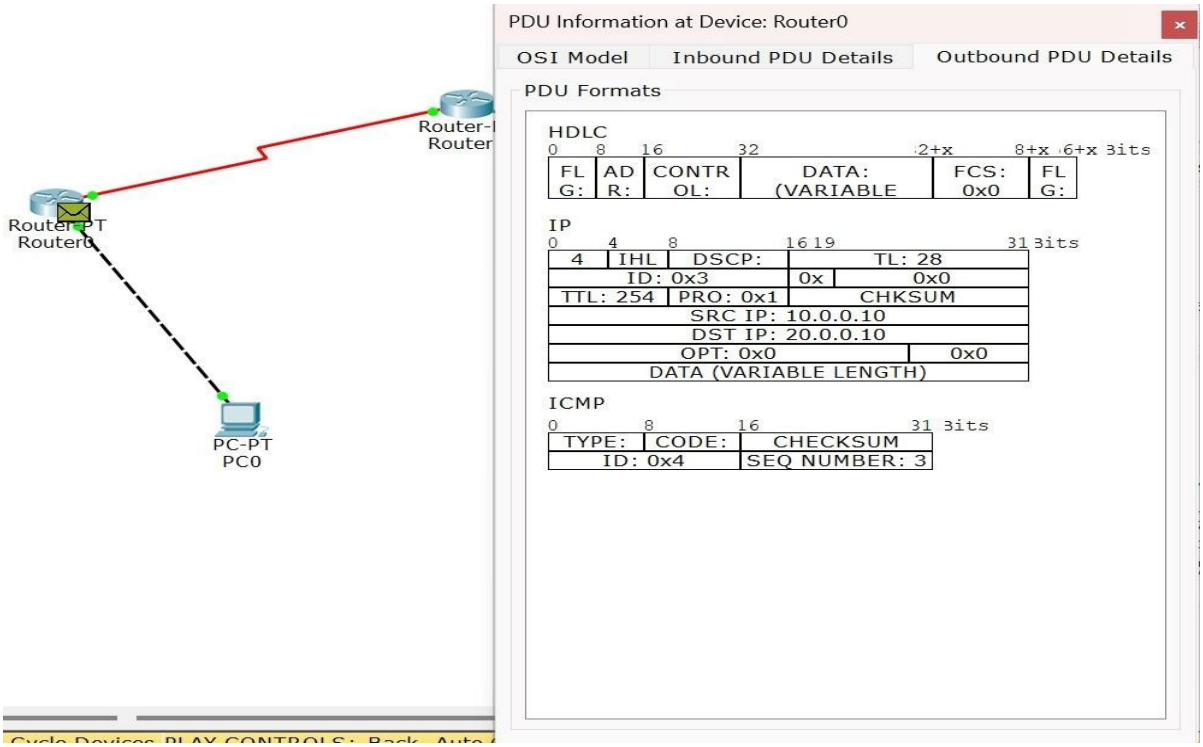


```
Enter the size of the packet entering at 1sec: 5
Enter the size of the packet entering at 2sec: 4
Enter the size of the packet entering at 3sec: 3
```

Second	Packet Recieved	Packet sent	Packet left	Dropp
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

Screenshots/ Output:





Observation:

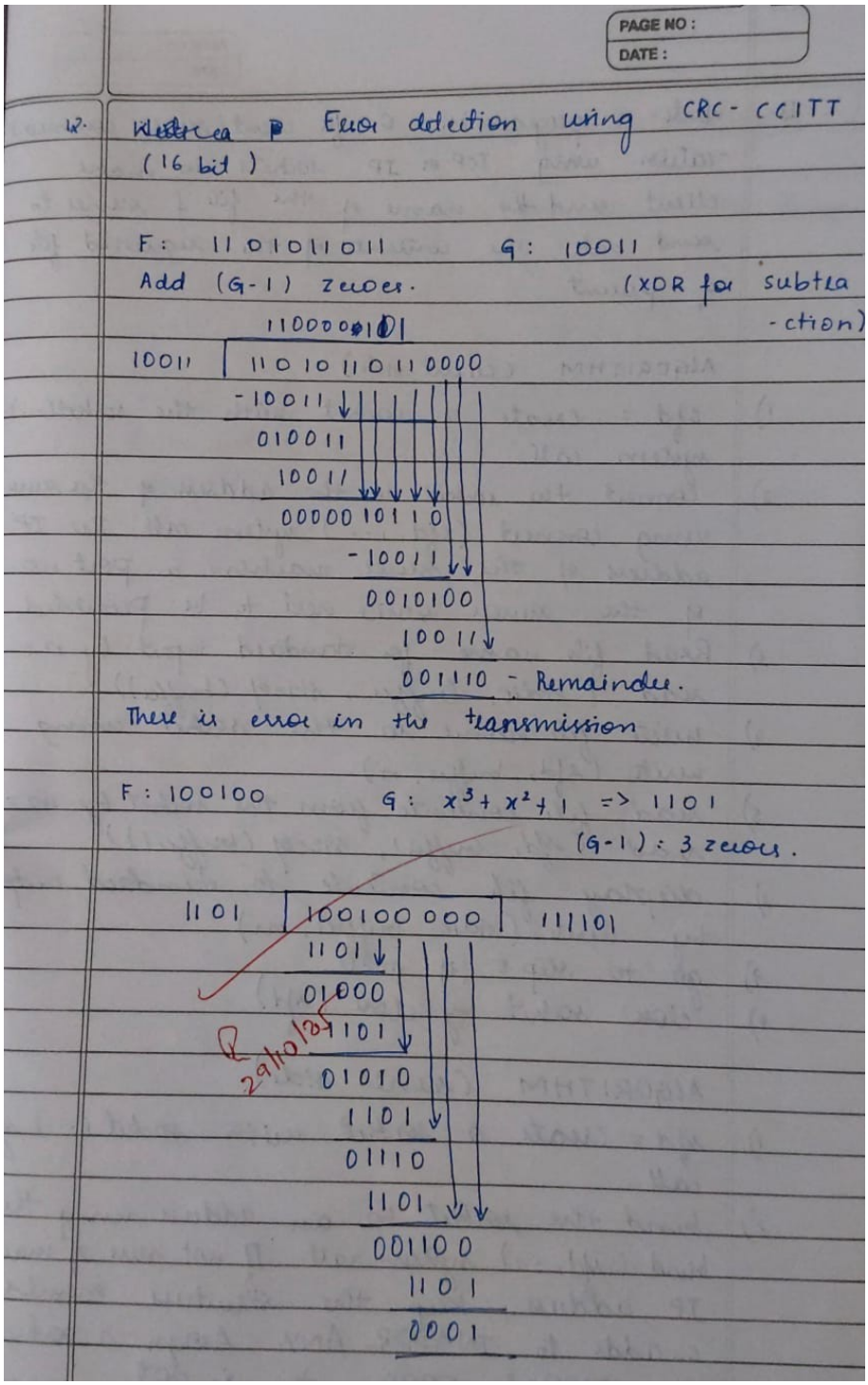
- The TTL field in the IP header decreased by one at each router hop, demonstrating its role in preventing packets from looping indefinitely in the network.

Program 12

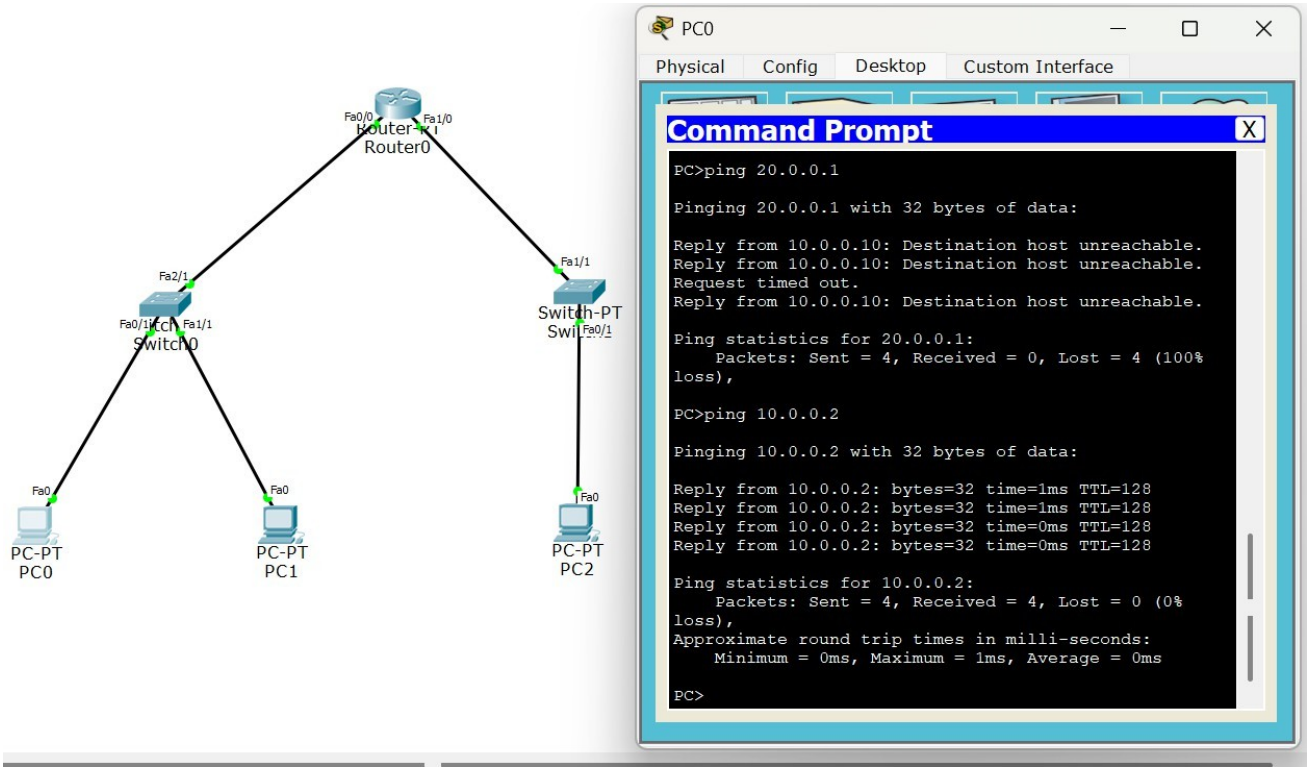
Aim of the program:

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

Procedure and topology:



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.

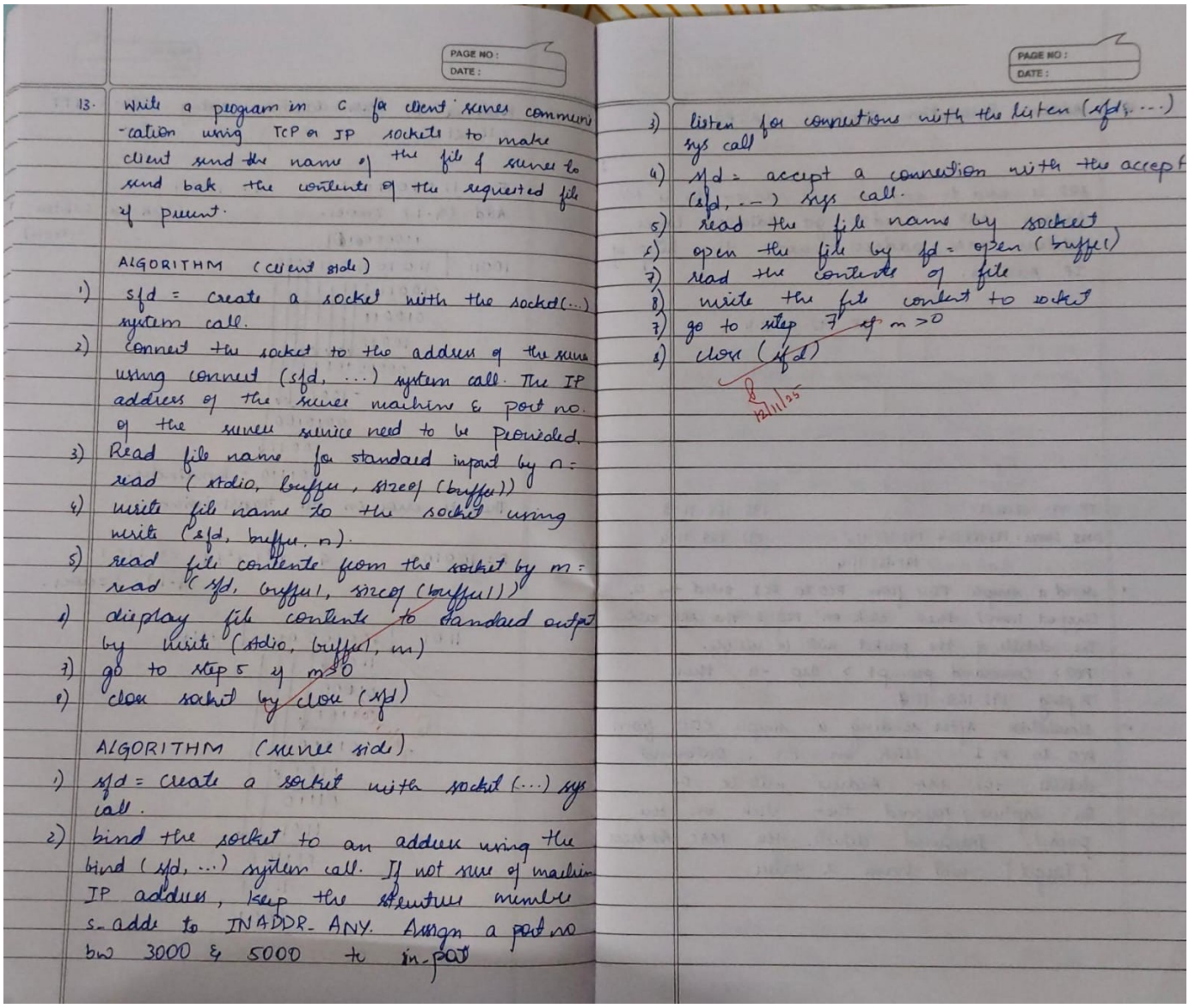
CYCLE 2:

Program 13

Aim of the program:

Write a program for congestion control using Leaky bucket algorithm

Procedure and topology:



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:

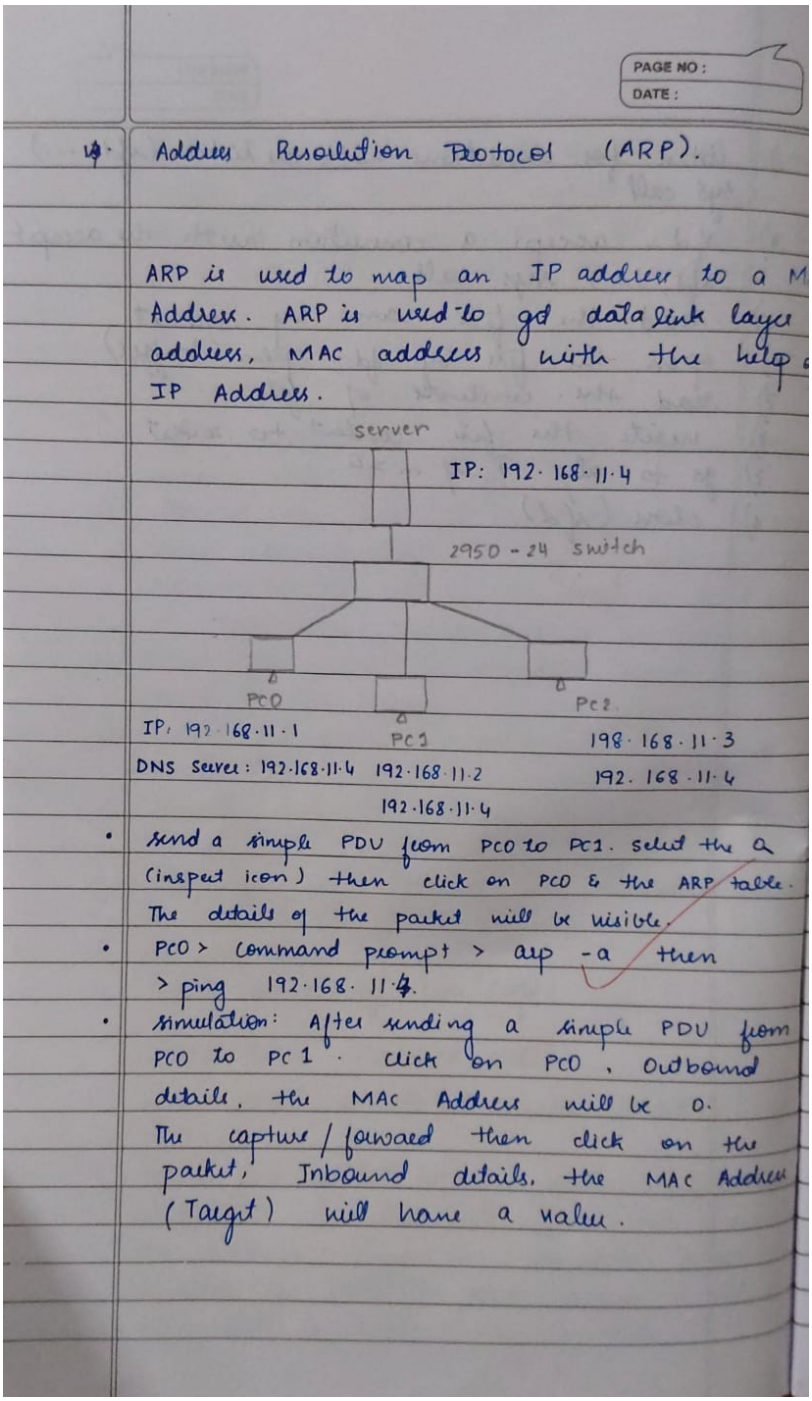
- The leaky bucket mechanism regulated the outgoing packet flow at a constant rate, preventing sudden traffic bursts from overwhelming the network.
- Packets exceeding the bucket capacity were dropped, demonstrating effective congestion control by smoothing traffic and enforcing rate-limiting.

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:

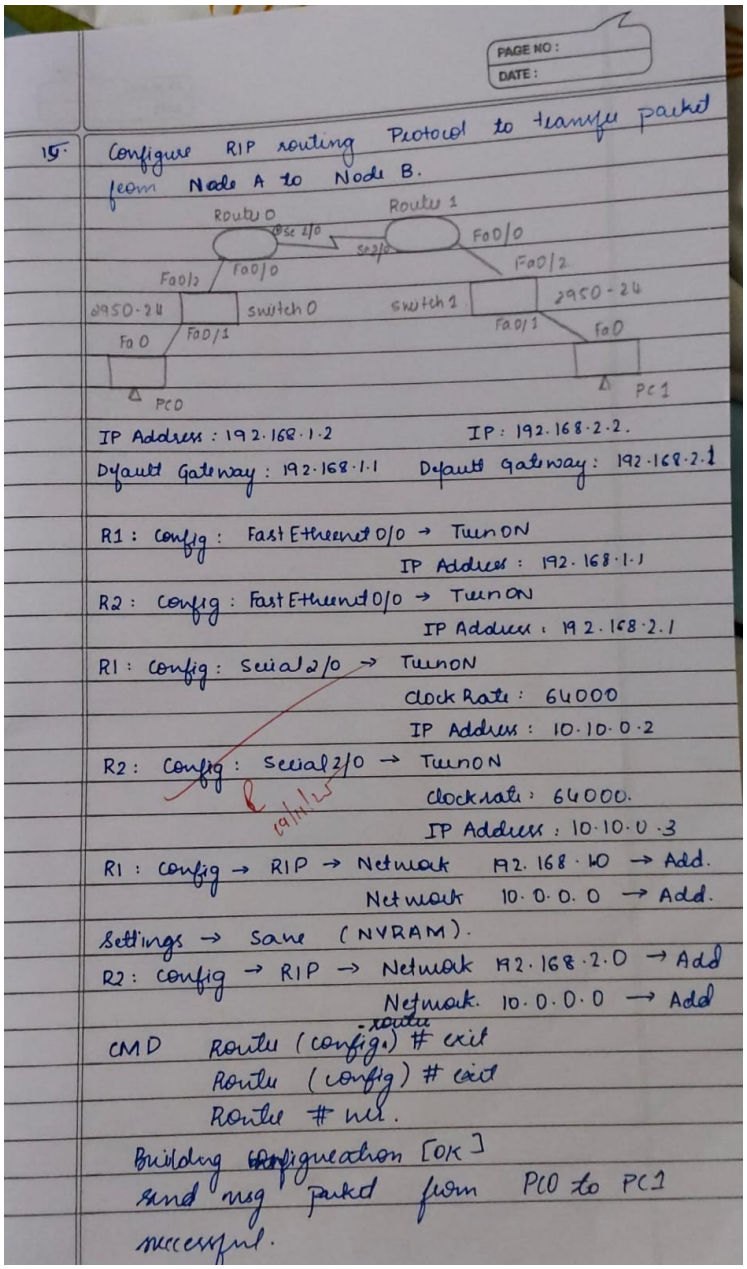
- The CRC-CCITT (16-bit) algorithm correctly generated a checksum for the transmitted data, ensuring reliable detection of single-bit and burst errors.
- Intentional error tests produced mismatched CRC values at the receiver, confirming accurate error detection through polynomial division.

Program 15

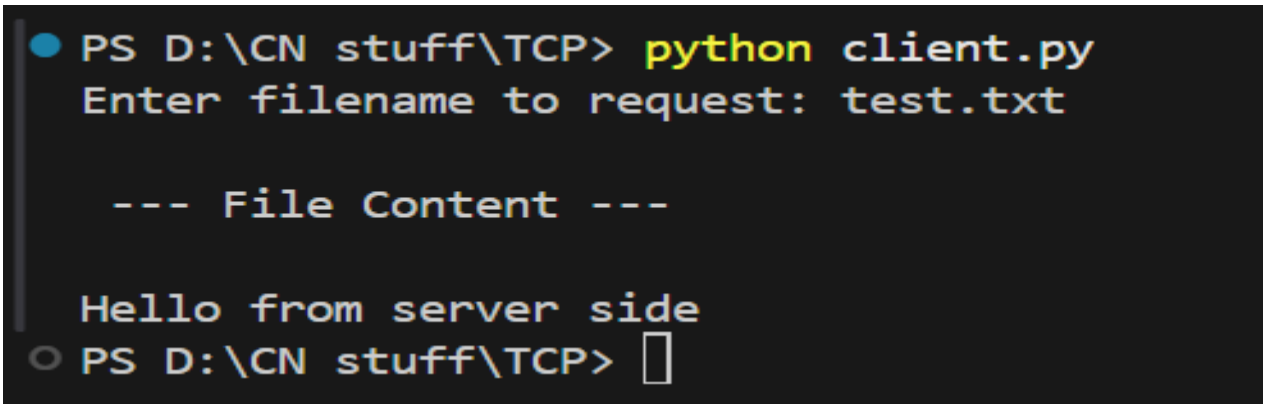
Aim of the program:

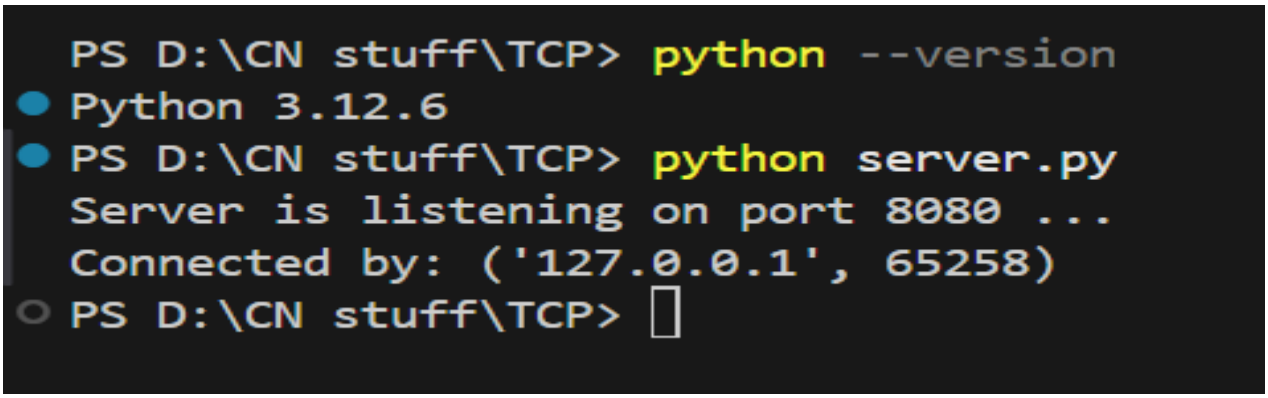
Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Procedure and topology:



Screenshots/ Output:





Observation:

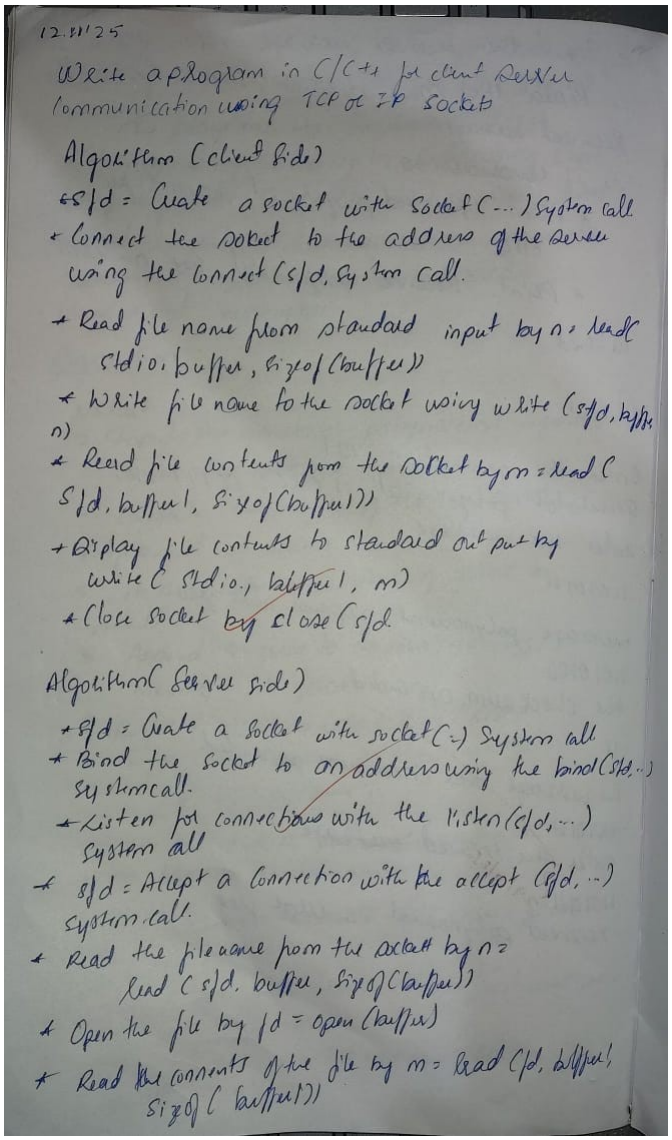
- The TCP client successfully established a reliable connection with the server and transmitted the requested filename using stream-oriented communication.
- The server correctly retrieved and returned the file contents over the same TCP session, demonstrating reliable data transfer, acknowledgment handling,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

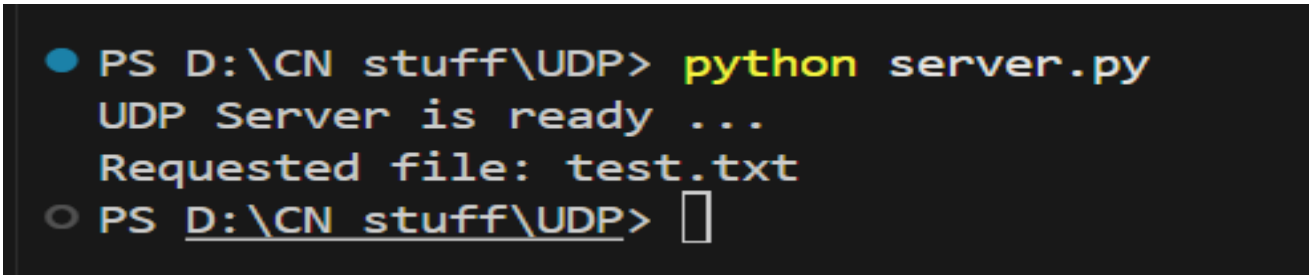
Algorithm (client side)

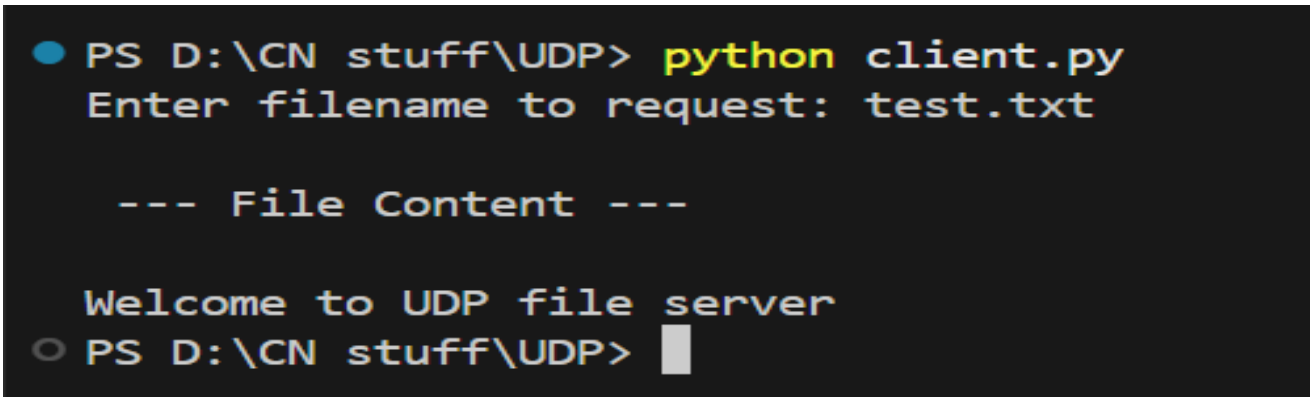
- s/d = Create a socket with socket(...) System call
- Connect the socket to the address of the Server using the connect(s/d, System call.
- Read file name from standard input by n = read(stdio, buffer, sizeof(buffer))
- Write file name to the socket using write(s/d, buffer, n)
- Read file contents from the socket by m = read(s/d, buffer, sizeof(buffer))
- Display file contents to standard output by write(stdio, buffer, m)
- Close socket by close(s/d)

Algorithm(Server side)

- s/d = Create a socket with socket(..) System call
- Bind the socket to an address using the bind(s/d, ..) system call.
- Listen for connections with the listen(s/d, ...) system call
- s/d = Accept a connection with the accept(s/d, ..) system call.
- Read the file name from the socket by n = read(s/d, 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:





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.