VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Utkrisht Umang (1BM23CS355)

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 (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

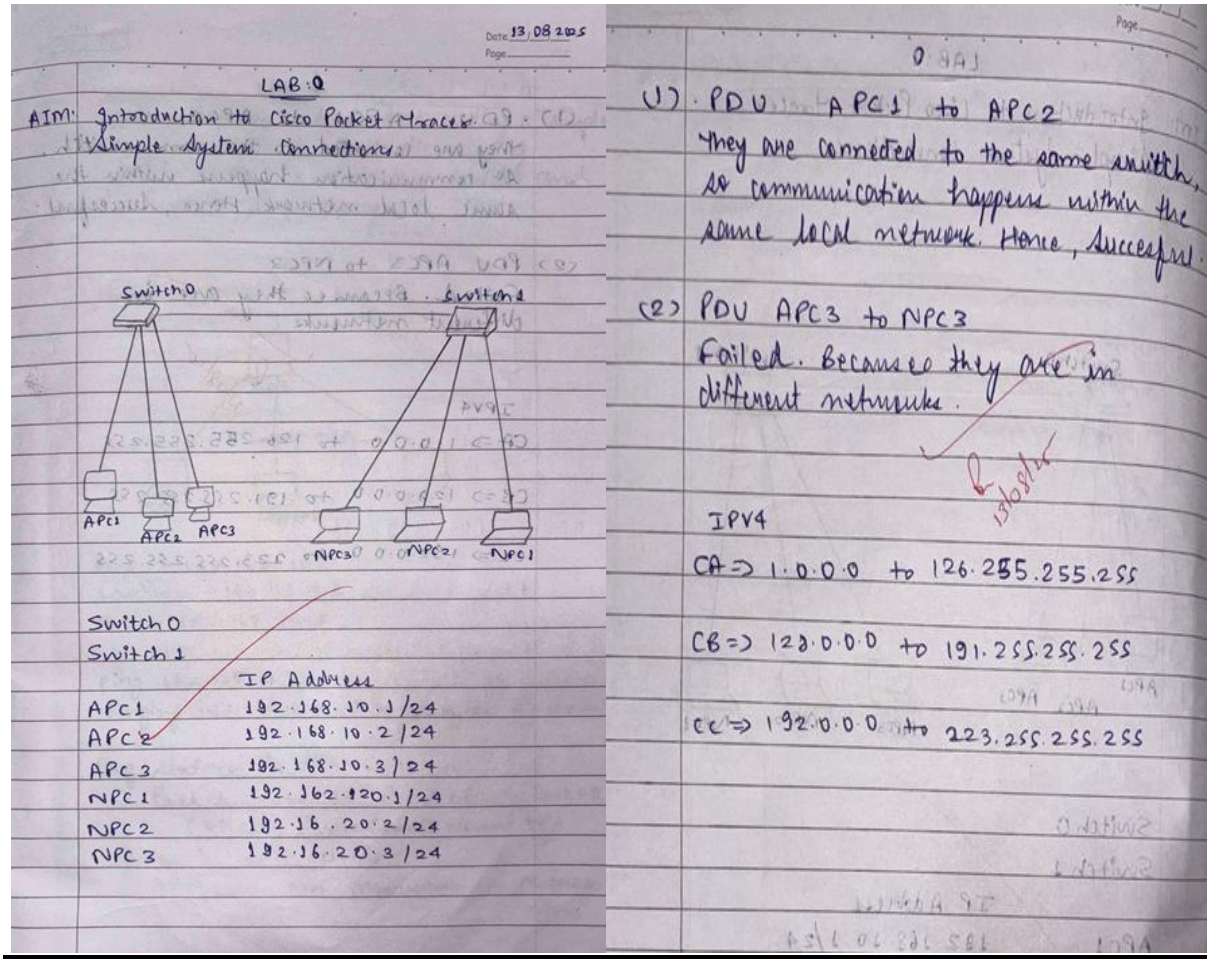
This is to certify that the Lab work entitled " Computer Network (23CS5PCCON)" carried out by **Utkrisht Umang (1BM23CS355),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of 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. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/utk1college/ComputerNetworks

Introduction:



Date 13/08/2025
Page

LAB:0

AIM: Introduction to Cisco Packet Tracer. Simple System Connections.

	IP Address
Switch 0	
Switch 1	
APC1	192.168.10.1/24
APC2	192.168.10.2/24
APC3	192.168.10.3/24
NPC1	192.162.120.1/24
NPC2	192.16.20.2/24
NPC3	192.16.20.3/24

Page

(1) PDU APC1 to APC2
they are connected to the same switch, so communication happens within the same local network. Hence, successful.

(2) PDU APC3 to NPC3
Failed. Because they are in different networks.

IPV4

CA => 1.0.0.0 to 126.255.255.255

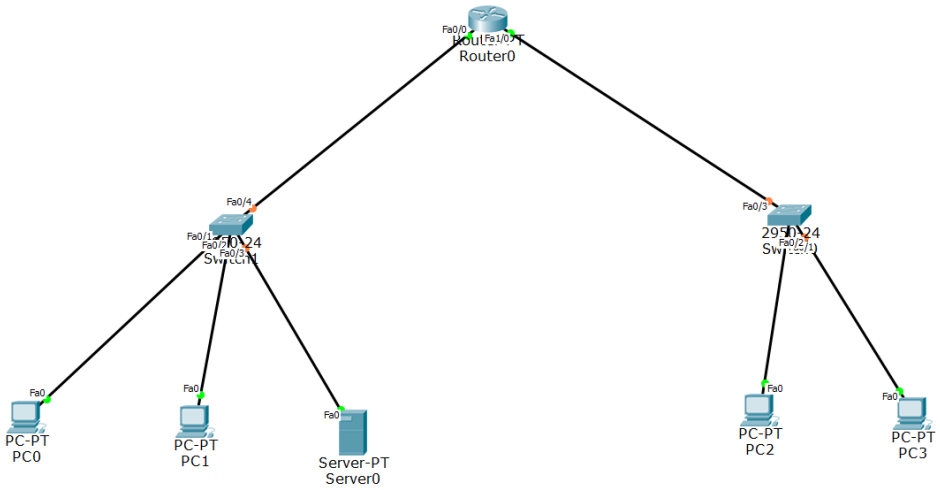
CB => 128.0.0.0 to 191.255.255.255

CC => 192.0.0.0 to 223.255.255.255

Program 1

Aim: Configure DHCP within a LAN and outside LAN.

Topology:



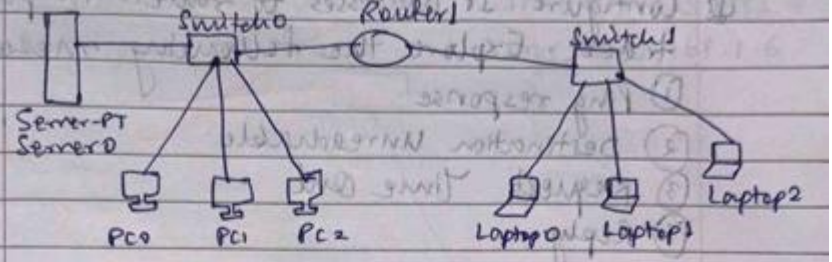
Procedure:

```
             Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706


Cisco Internetwork Operating System Software
IOS (tm) PT1000 Software (PT1000-I-M), Version 12.2(28), RELEASE SOFTWARE (fc5)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2005 by cisco Systems, Inc.
Compiled Wed 27-Apr-04 19:01 by miwang

PT 1001 (PTSC2005) processor (revision 0x200) with 60416K/5120K bytes of memory
.
Processor board ID PT0123 (0123)
PT2005 processor: part number 0, mask 01
Bridging software.
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)

Press RETURN to get started!


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

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up
```

Observation:

LAB-2

Q: Configure DHCP within a LAN and outside a LAN.



ON ON

Pool Name	SwitchOne	SwitchTwo
Gateway	192.168.10.1	192.168.20.1
Start IP Address	192.168.10.3	192.168.20.2
Subnet Mask	255.255.255.0	255.255.255.0
Max No. of Users	20	20
	Add	Add

for Laptop 2:

DHCP request successful

Router : → CLI → no.

Router > enable

conf t

int fa0/0

ip address 192.168.10.1 255.255.255.0

ip helper-address 192.168.10.2

no shutdown

do write memory

exit

int fa0/1

ip address 192.168.20.1 255.255.255.0

ip helper-address 192.168.10.2

no shut down

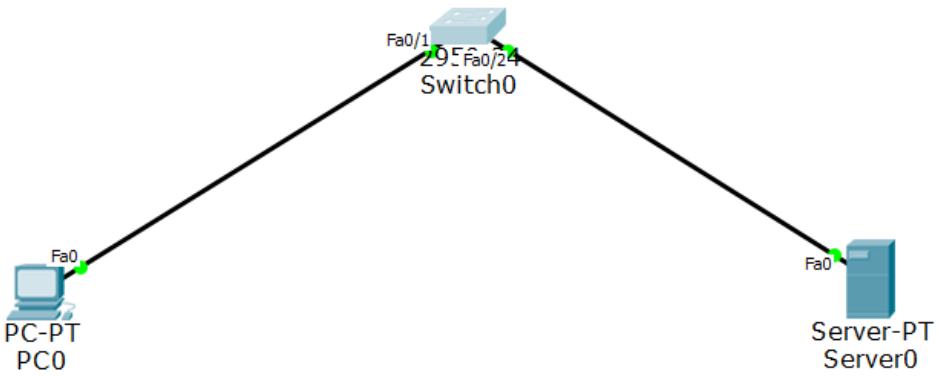
do write memory

exit

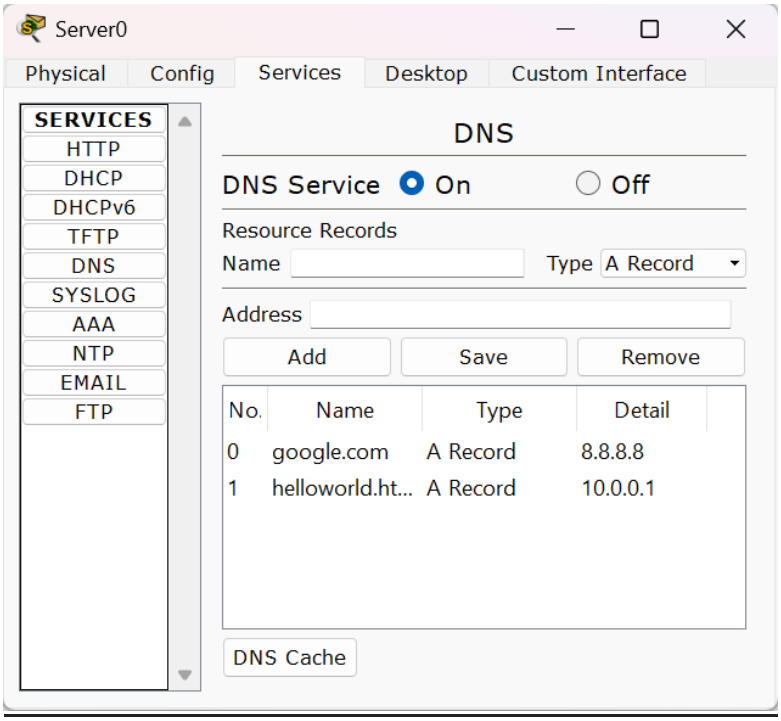
Program 2

Aim: Configure Web Server, DNS within a LAN

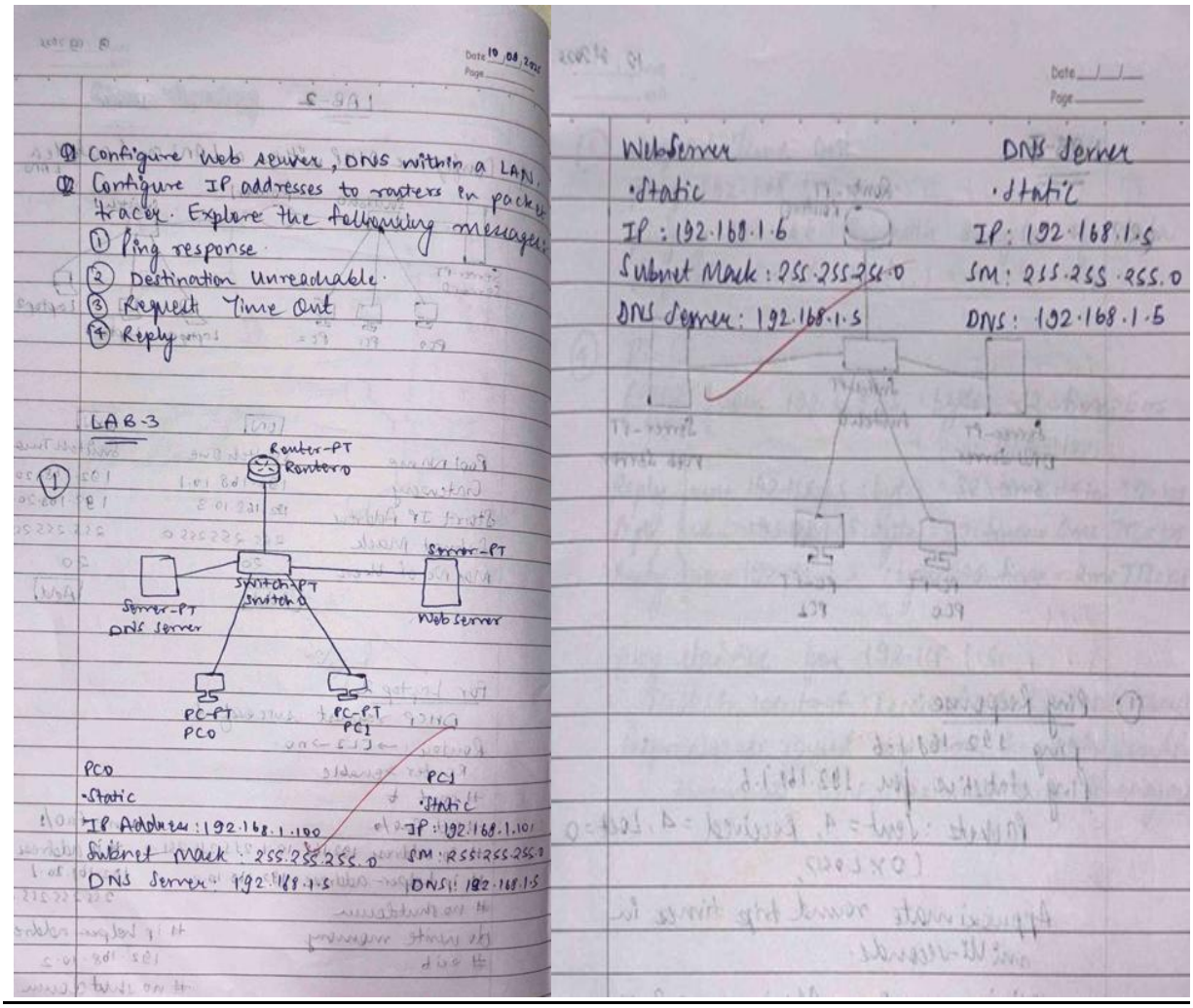
Topology:



Procedure:



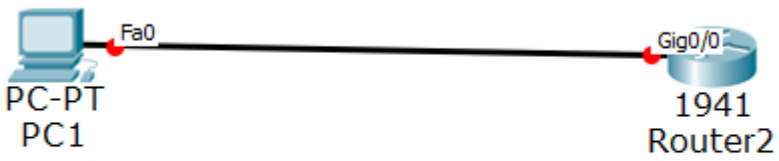
Observation:



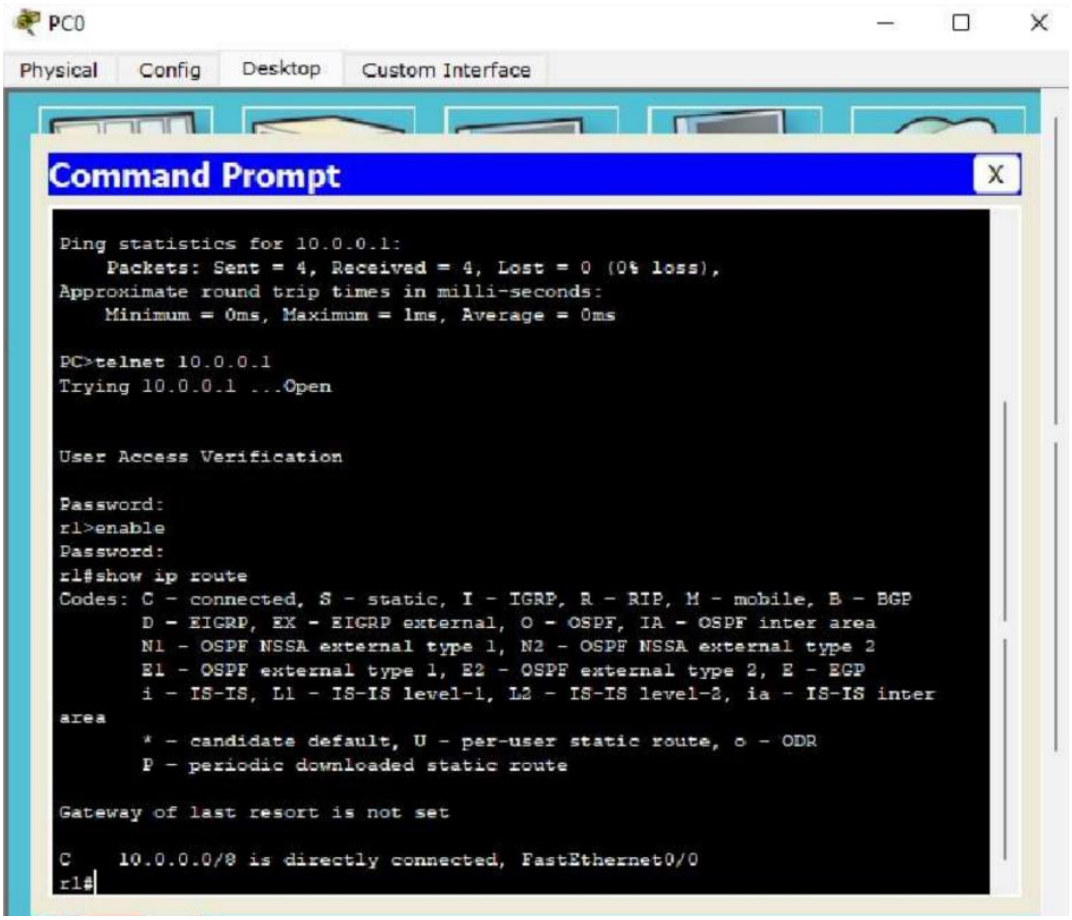
Program 3

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

Topology:



Procedure:

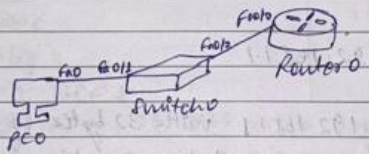


Observation:

Configure TELNET to access router remotely.

TELNET

- It is used to access remote servers.
- It is a simple command line tool that runs on your computer and it allows you to send commands remotely to a server and administrator.
- It is also used to manage other devices like router, switch to check if ports are open or close on the server



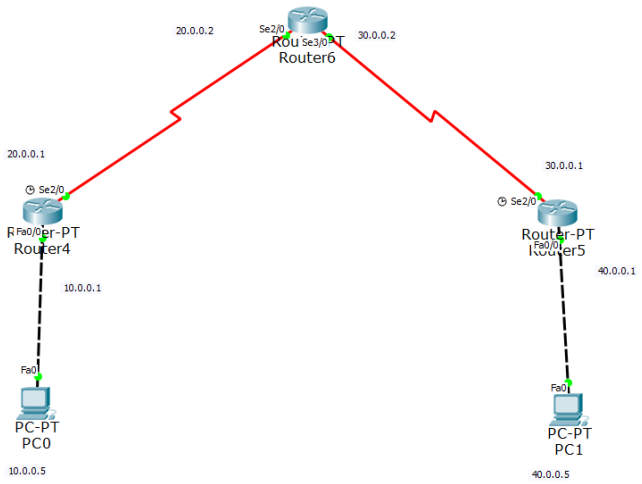

```
Router >enable
conf t
hostname R1
enable secret rp
int Fa0/0
ip address 192.168.1.1 255.255.255.0
no shutdown

line vty 0 5
login
Login disabled on line 194, until 'password' is set
  "   "   "  "  195,  "   "   "  "
  "   "   "  "  196,  "   "   "  "
  "   "   "  "  197,  "   "   "  "
  "   "   "  "  198,  "   "   "  "
  "   "   "  "  199,  "   "   "  "
```

```
password tp
exit
exit
wr
show ip interface brief
Interface   IP-Address   OK? Method Status
fast Ethernet0  192.168.1.1  YES manual up
  FE 0/1    unass.       YES unset
  Vlan1     unass.       YES unset

show ip int brief
Fa0/1   192.168.12.1  YES manual

PC0/0
ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from ..  bytes=32 time=0ms TTL=255

telnet 192.168.1.1
Trying 192.168.1.1 ... Open
User Access Verification
Password: tp
enable
Password: rp
ip interface brief
Interface   IP-Address   OK? Method Status
  Fa0/1     unassigned   YES
ip add 192.168.12.1 255.255.255.0
(ip gets updated)
```

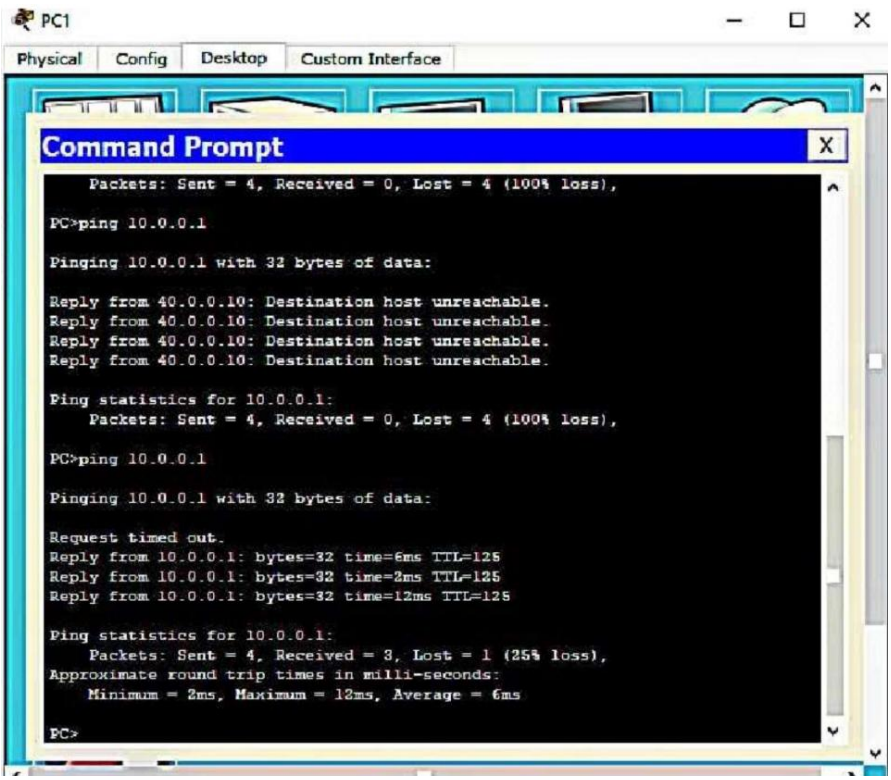
Program 4

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

Topology:



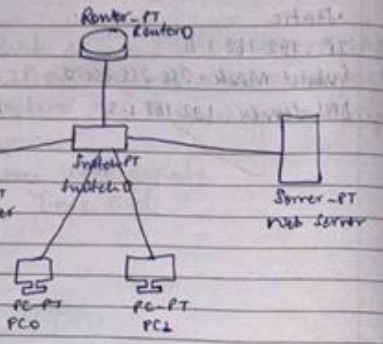
Output:



Observation:

LAB-3

(2)



(1) Ping Response:

ping 192.168.1.6

Ping statistics for 192.168.1.6:

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

Approximate round trip times in milli-seconds:

Minimum: 0ms, Maximum = 2ms, Average = 1ms

(2) Destination Unreachable

PC0 was connected through a router to a web server. An Access Control List was applied to the router to block PC1's traffic to the server. PC1 - 192.168.1.101, Web Server - 10.0.0.2

ping 10.0.0.2

Reply from 192.168.1.1: Destination host unreachable

Ping statistics for 10.0.0.2:

Packages: Sent = 4, Received = 0, Lost = 4 (100% Loss)

(3) Request Time Out

ping 192.168.1.2

pinging 192.168.1.2 with 32 bytes of data

Request timed out.

(4) Reply

Reply from 192.168.1.5: bytes=32 time=5ms TTL=128.

Reply from 192.168.1.5: bytes=32 time=4ms TTL=128

Reply from 192.168.1.5: bytes=32 time=0ms TTL=128

Reply from 192.168.1.5: bytes=32 time=2ms TTL=128

ping statistics for 192.168.1.5:

Packets: sent = 4, Received = 4, Lost = 0 (0% Loss),

Approximate round trip times in milli-seconds:

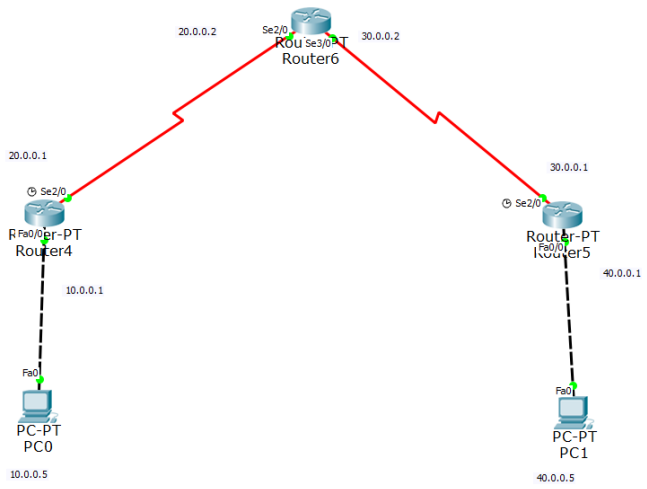
Minimum = 0ms, Maximum = 5ms, Average = 2ms

10/11/2022

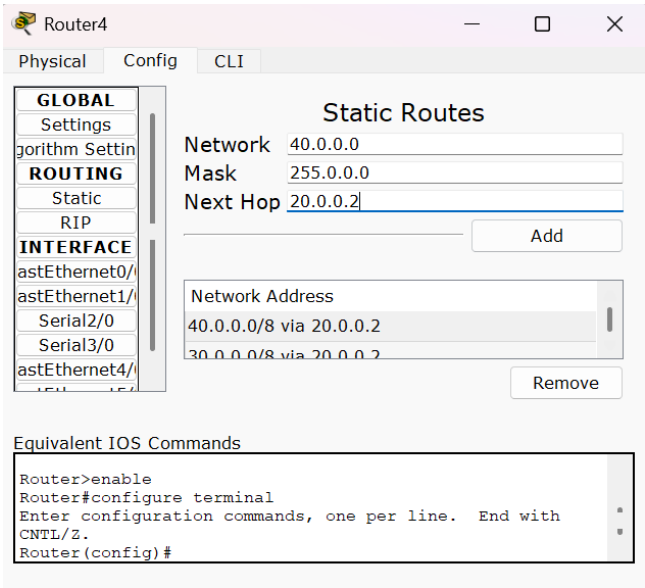
Program 5

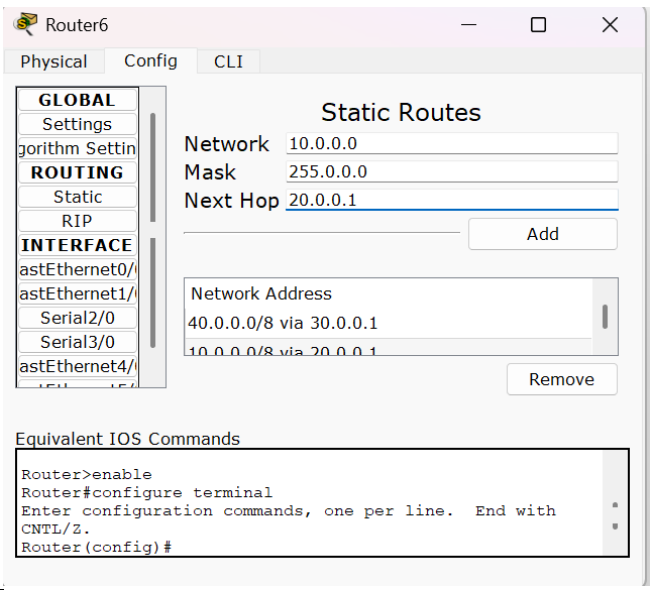
Aim: Configure default route, static route to the Router

Topology:

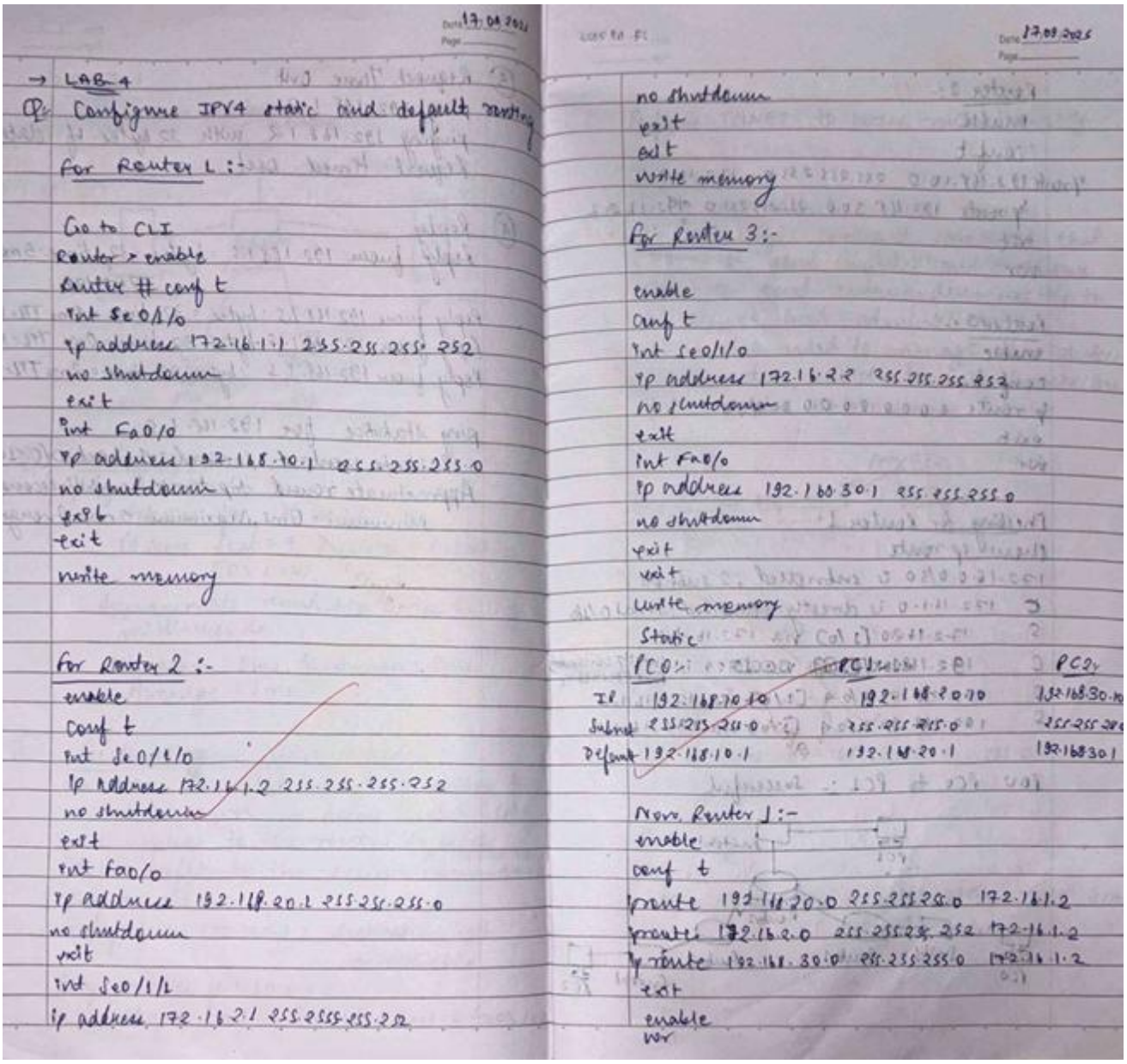


Procedure:





Observation:



→ LAB 4

Q: Configure IPV4 static and default routing

For Router 1 :-

Go to CLI
Router > enable
Router # conf t
int Se0/1/0
ip address 172.16.1.1 255.255.255.252
no shutdown
exit
int Fa0/0
ip address 192.168.10.1 255.255.255.0
no shutdown
exit
exit
write memory

for Router 2 :-

enable
Conf t
int Se0/1/0
ip address 172.16.1.2 255.255.255.252
no shutdown
exit
int fa0/0
ip address 192.168.20.1 255.255.255.0
no shutdown
exit
int Se0/1/1
ip address 172.16.2.1 255.255.255.252
no shutdown
exit
exit
write memory

for Router 3:-

enable
conf t
int se0/1/0
ip address 172.16.2.2 255.255.255.252
no shutdown
exit
int Fa0/0
ip address 192.168.30.1 255.255.255.0
no shutdown
exit
exit
write memory

Static

PC0:		PC1:	PC2:
IP	192.168.10.10	192.168.20.10	192.168.30.10
Subnet	255.255.255.0	255.255.255.0	255.255.255.0
Default	192.168.10.1	192.168.20.1	192.168.30.1

Now, Router 1 :-
enable
conf t
ip route 192.168.20.0 255.255.255.0 172.16.1.2
ip route 172.16.2.0 255.255.255.252 172.16.1.2
ip route 192.168.30.0 255.255.255.0 172.16.1.2
exit
enable
wr

Date 17/09/2024
Page

Router 2:-

enable
conf t
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
wr

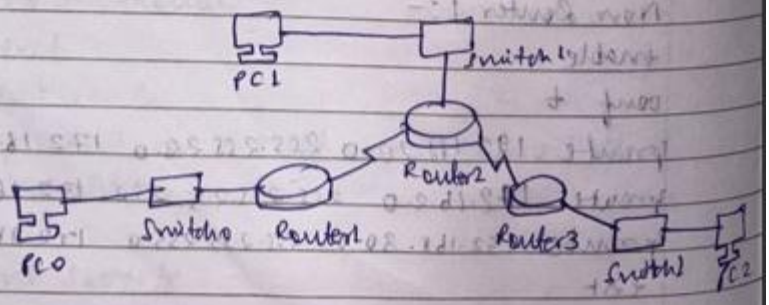
Router 3:-

enable
conf t
ip route 0.0.0.0 0.0.0.0 se0/1/0
exit
wr

Checking for Router 1:-

show ip route
172.16.0.0/30 is subnetted, 2 subnets
C 172.16.1.0 is directly connected, Serial 0/1/0
S 172.16.2.0 [1/0] via 172.16.1.2
C 192.168.20.0/24 is directly connected, FastEthernet0/0
S 192.168.20.0/24 [1/0] via 172.16.1.2
S 192.168.30.0/24 [1/0] via 172.16.1.2

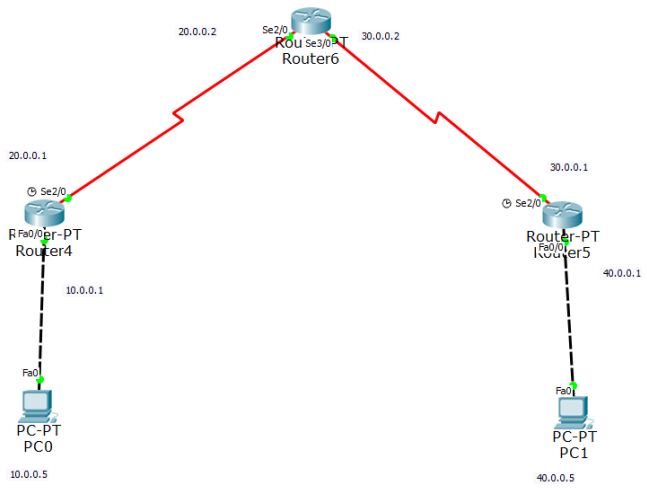
PDU PC0 to PC1 :- Successful



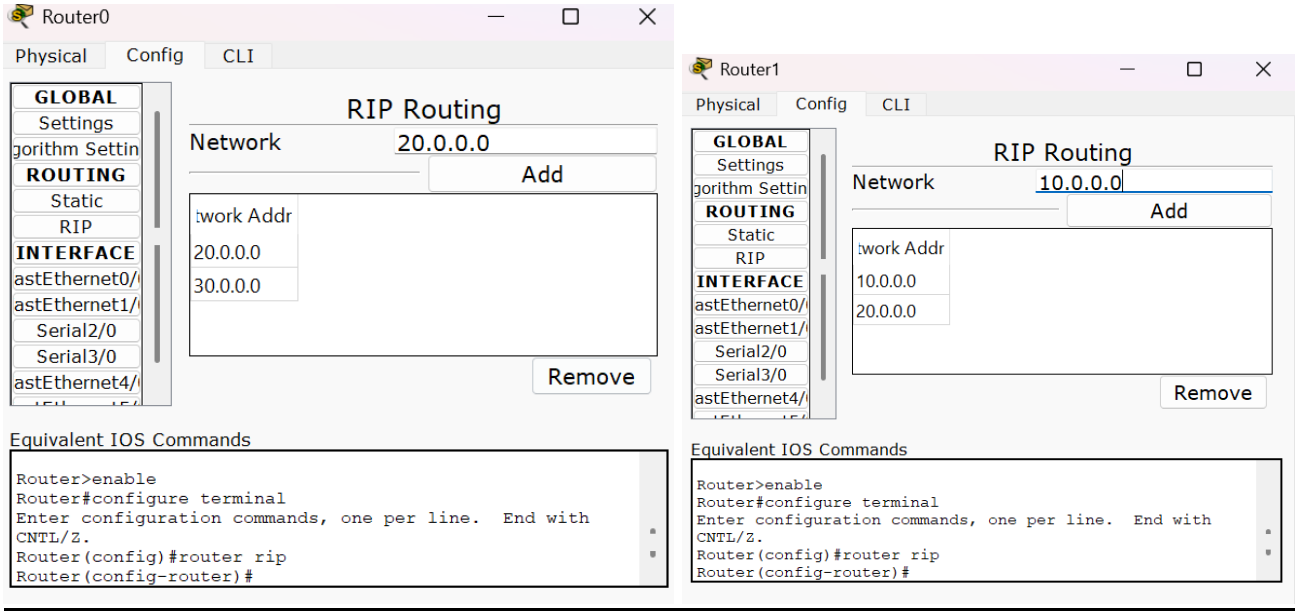
Program 6

Aim: Configure RIP routing Protocol in Routers

Topology:



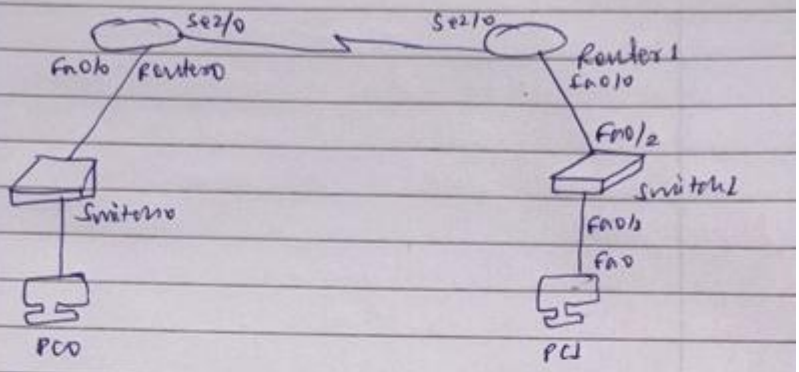
Procedure:



Observation:

Date 19/11/2025
Page

→ Configure RIP routing protocol in router to transfer packet from node A to node B.



	PC 0	PC1
IP	192.168.1.2	192.168.2.2
Def Gat	192.168.1.1	192.168.2.1

	Router 0	Router 1
Fa0/0	IP : 192.168.1.1	192.168.2.1
Serial 2/0	IP: 10.10.0.2	10.10.0.3
Clock Rate	64000	64000

RIP Routing

Network Address

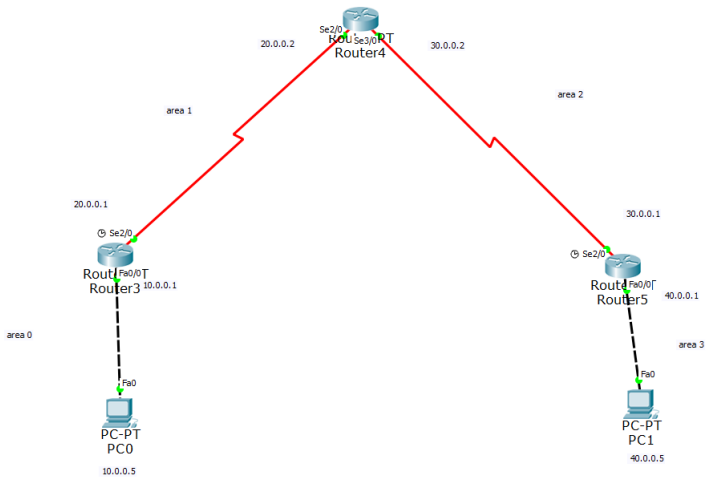
10.0.0.0

192.168.2.0.

Program 7

Aim: Configure OSPF routing protocol

Topology:



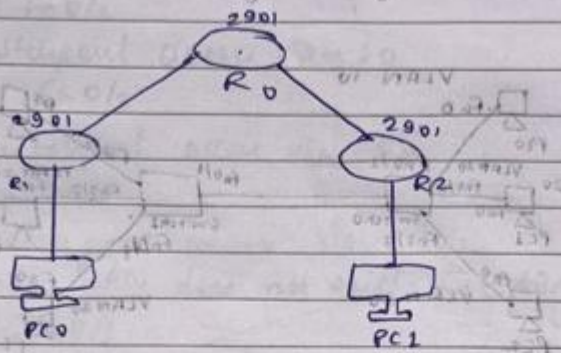
Procedure:

```
Cisco Internetwork Operating System Software
IOS (tm) PT1000 Software (PT1000-I-M), Version 12.2(28), RELEASE SOFTWARE (fc5)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2005 by cisco Systems, Inc.
Compiled Wed 27-Apr-04 19:01 by miwang

PT 1001 (PTSC2005) processor (revision 0x200) with 60416K/5120K bytes of memory
.
Processor board ID PT0123 (0123)
PT2005 processor: part number 0, mask 01
Bridging software.
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)

Press RETURN to get started!


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

%LINK-5-CHANGED: Interface Serial2/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up

00:00:10: %OSPF-5-ADJCHG: Process 1, Nbr 30.0.0.2 on Serial2/0 from LOADING to FULL, Loading Done
```

Observation:

Date 08/11/2025
Page

Configure OSPF routing protocol



R0 :-

enable
config t
router OSPF 1
network 192.168.55.0 0.0.0.255 area 0
network 176.16.0.0 0.0.255.255 area 0
exit
process1

R2:-

ospf 1
network 176.0.0.0 0.0.0.255 area 0
network 10.0.0.0 0.255.255.255 area 0
exit

R1:-

ospf 1
nw 192.168.55.0 0.0.0.255 area 0
nw 192.168.44.0 0.0.0.255 area 0
exit

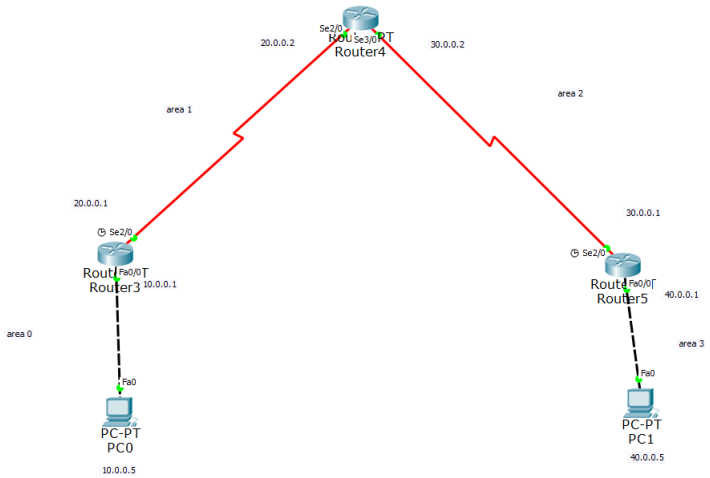
PC0:-

Ping 10.10.22.2
Ping statistics for 10.10.22.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% Loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

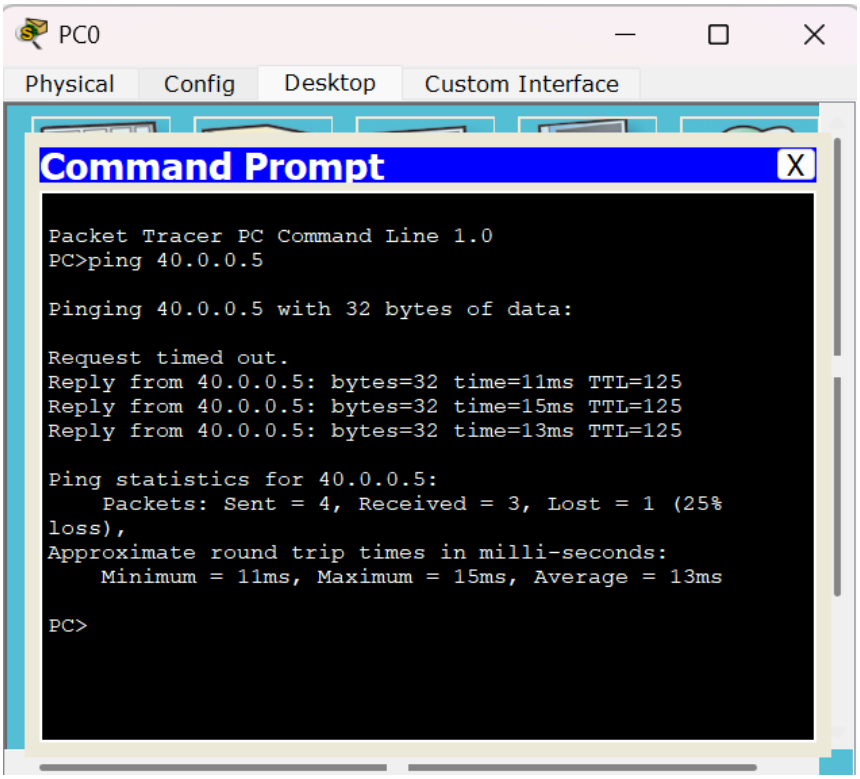
Program 8

Aim: Demonstrate the TTL/ Life of a Packet

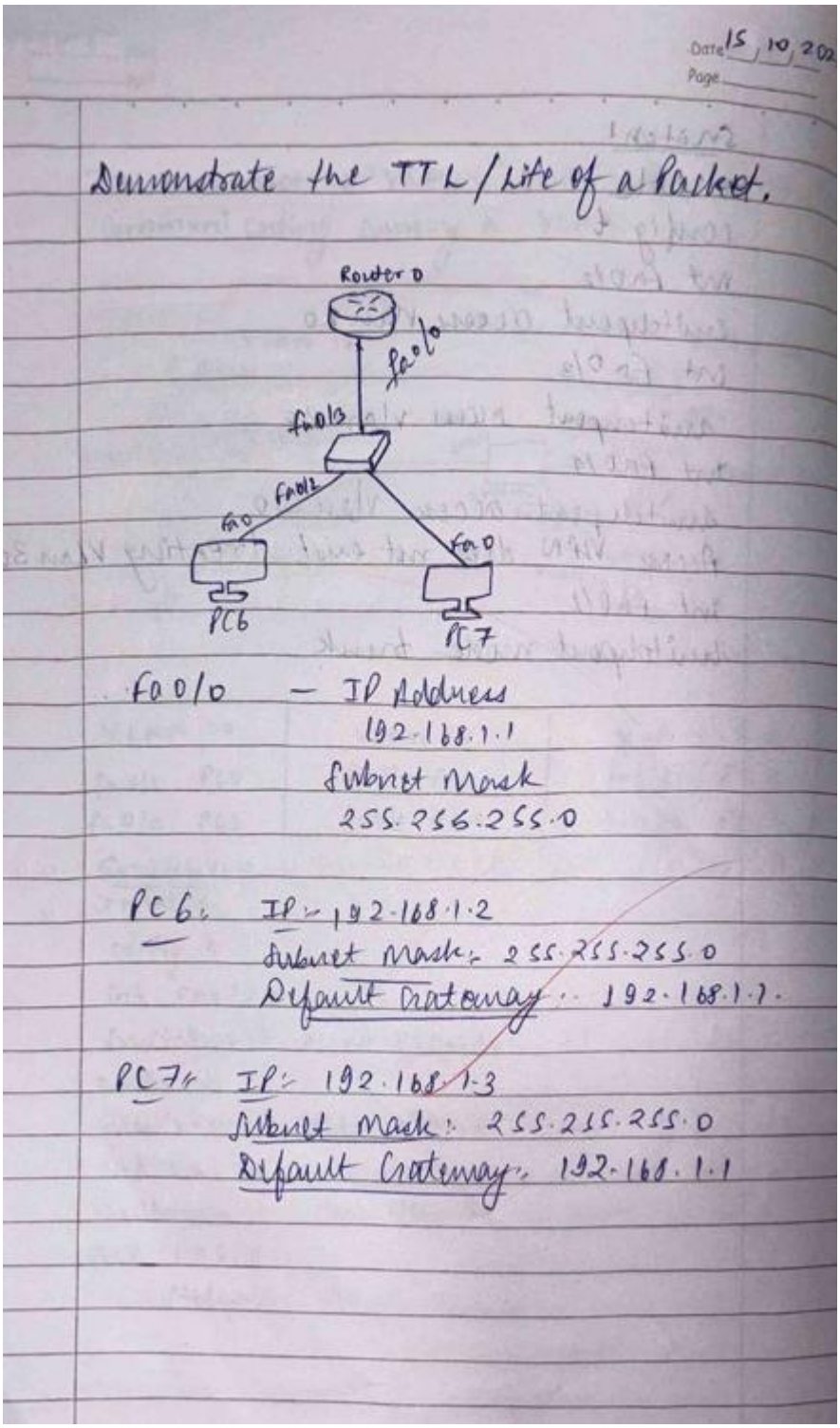
Topology:



Output:



Observation:



Date 15/10/202
Page

Demonstrate the TTL / Life of a Packet.

fa0/0 — IP Address
192.168.1.1
Subnet Mask
255.255.255.0

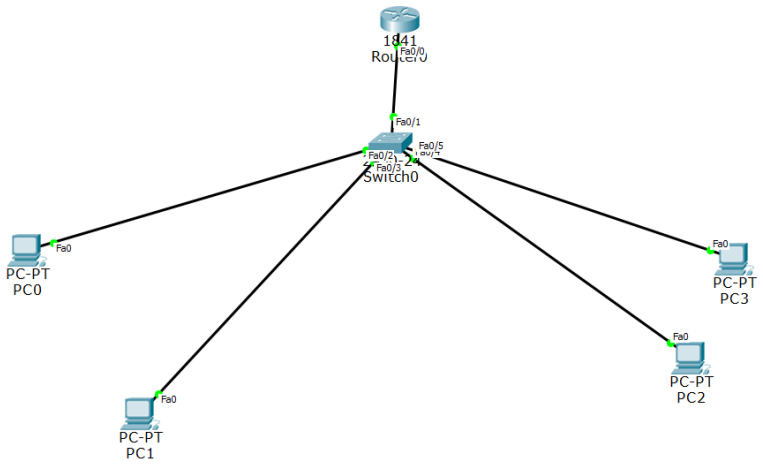
PC6: IP: 192.168.1.2
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.1.1

PC7: IP: 192.168.1.3
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.1.1

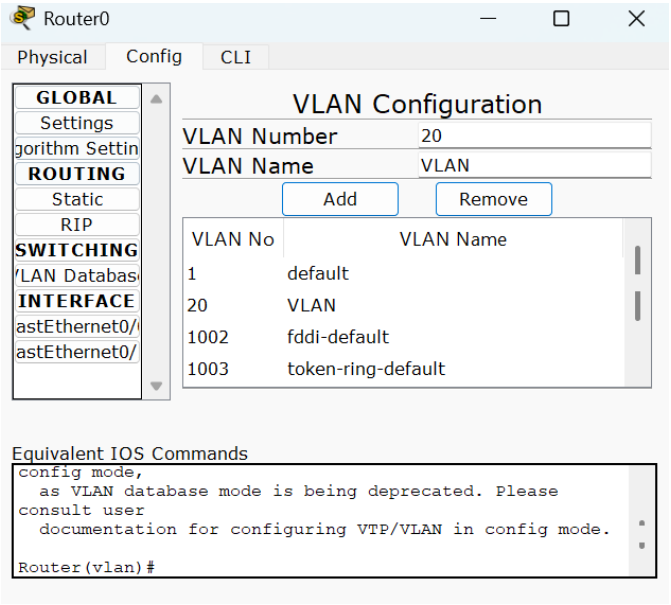
Program 9

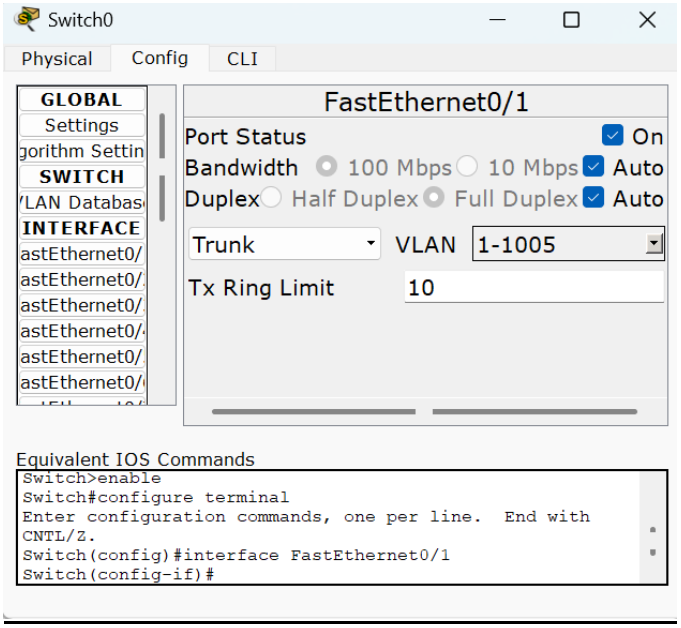
Aim: To construct a VLAN and make the PC's communicate among a VLAN

Topology:



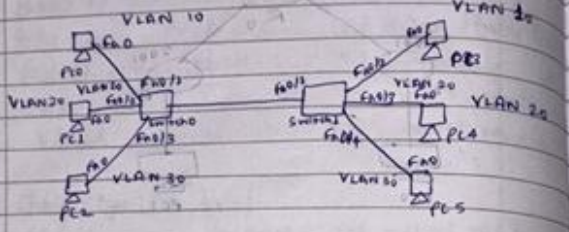
Procedure:





Observation:

To construct a VLAN and make the PCs communicating among a VLAN.



VLAN 10	VLAN 20	VLAN 30
fa0/1 PC0	fa0/2 PC1	fa0/3 PC2
fa0/2 PC3	fa0/3 PC4	fa0/4 PC5

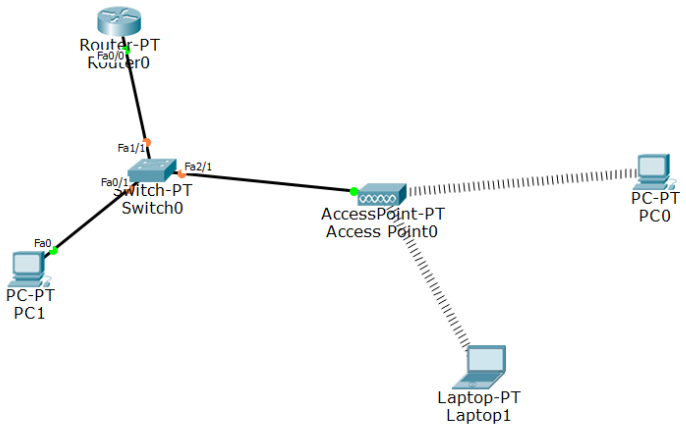
Switch0
>enable
config t
int Fa0/1
switchport access vlan 10
int Fa0/2
switchport access vlan 20
int Fa0/3
switchport access vlan 30
int fa0/4
switchport mode trunk

Switch1
>enable
config t
int Fa0/2
switchport access vlan 10
int Fa0/3
switchport access vlan 20
int Fa0/4
switchport access vlan 30
Access VLAN does not exist. Creating vlan 30
int Fa0/1
switchport mode trunk

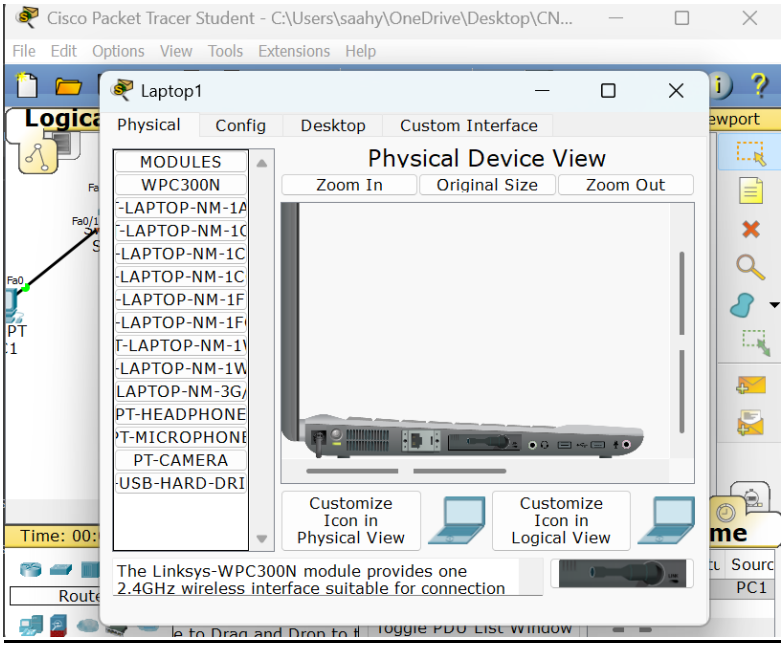
Program 10

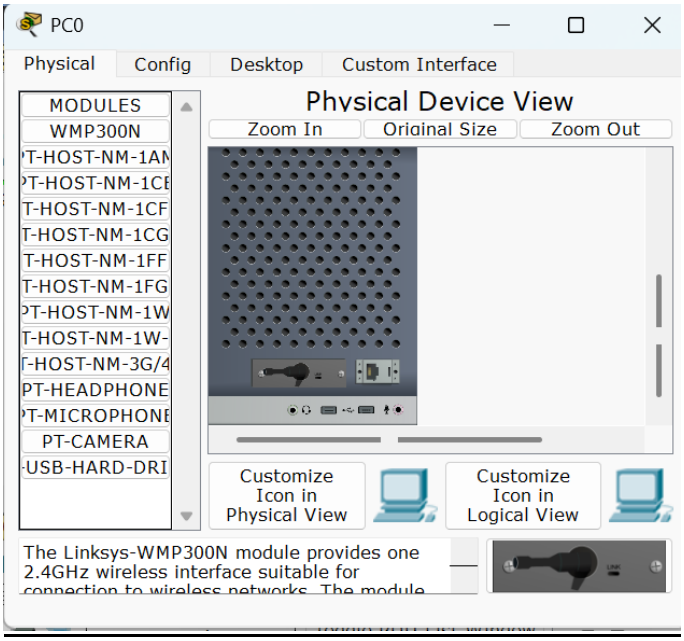
Aim: To construct a WLAN and make the nodes communicate wirelessly

Topology:

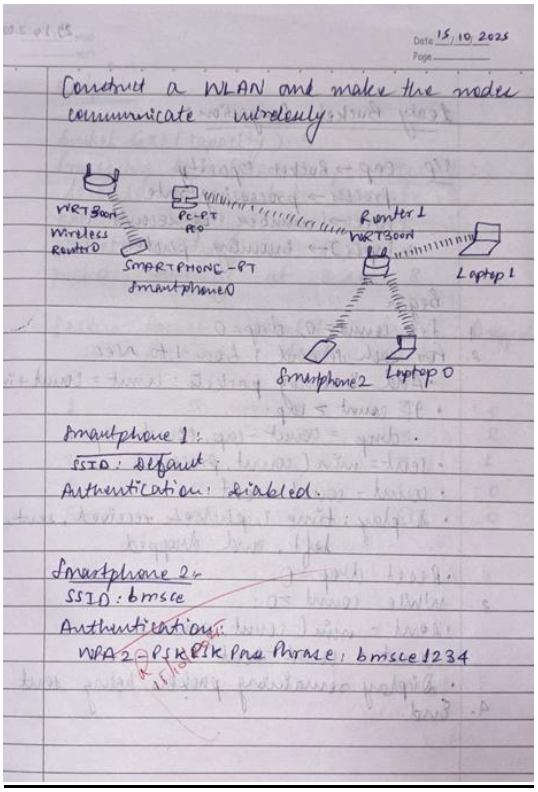


Procedure:





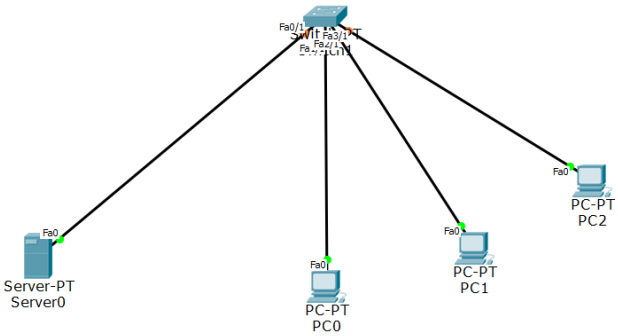
Observation:



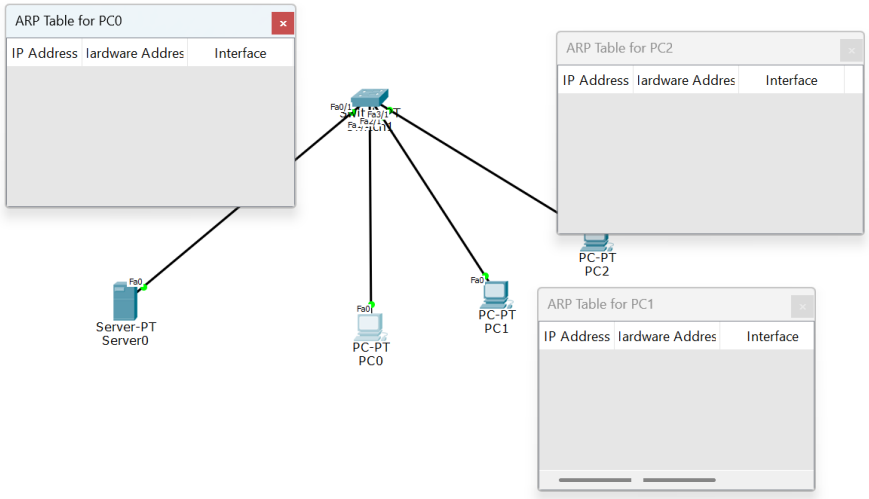
Program 11

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

Topology:



Procedure:

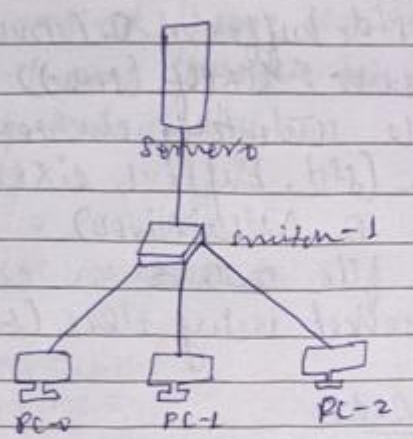


Observation:

Date 19/11/2025
Page

→ ARP (Address Resolution Protocol)

- It is used to map an IP Address to a MAC address.
- It is used to get Data Link layer address MAC address with the help of IP address.



PC-0 ⇒ 192.168.11.1 → IP
255.255.255.0 → Subnet
DNS = 192.168.11.4

PC-1 ⇒ 192.168.11.2
255.255.255.0
192.168.11.4

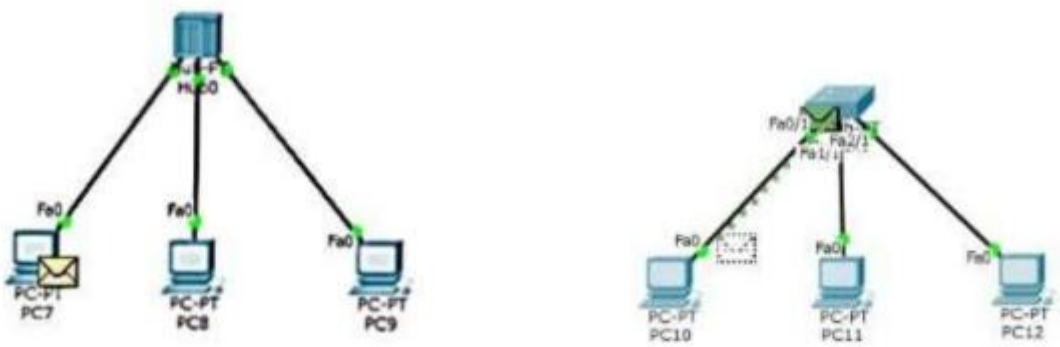
PC-2 ⇒ 192.168.11.3 → IP
255.255.255.0 ⇒ Subnet Mask
DNS ⇒ 192.168.11.4

Server ⇒ 192.168.11.4.

Program 12

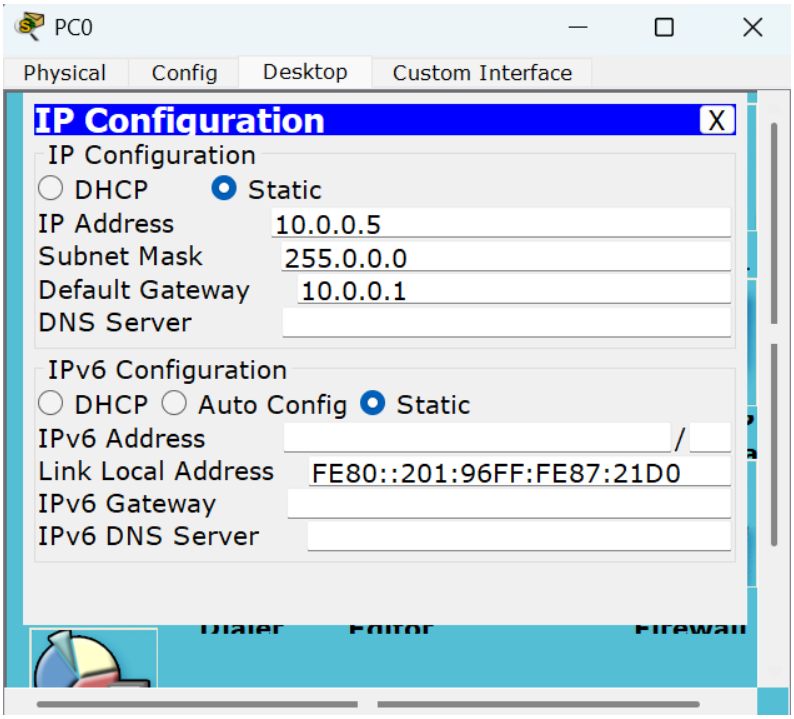
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Topology:



-

Procedure:



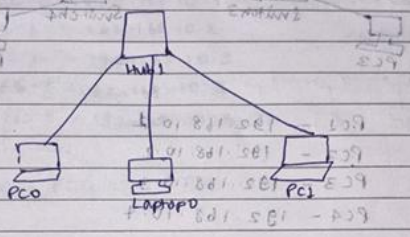
Observation:

Date 20/08/2025
Page

LAB-1

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

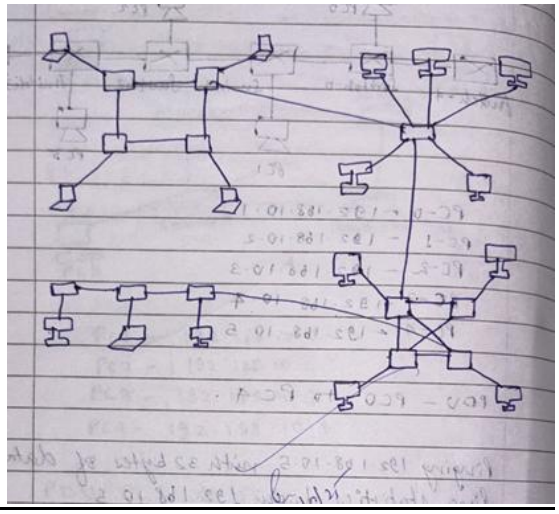
Hub:



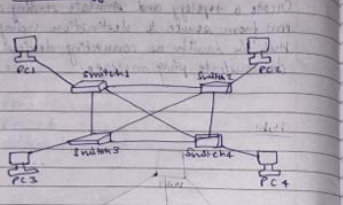
PC-0 – 192.168.10.1
Laptop0 – 192.168.10.2
PC1 – 192.168.10.3

ping 192.168.10.3
Pinging 192.168.10.3 with 32 bytes of data:

Ping Statistics for 192.168.10.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% Loss), Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 4ms, Average = 4ms



Mesh Topology:

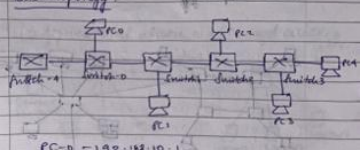


PC1 – 192.168.10.1
PC2 – 192.168.10.2
PC3 – 192.168.10.3
PC4 – 192.168.10.4

PDU PC1 to PC4

Pinging 192.168.10.4 with 32 bytes of data:
Ping Statistics for 192.168.10.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% Loss), Approximate round trip times in milliseconds:
Minimum = 6ms, Maximum = 6ms, Average = 6ms

Bus Topology:



PC-0 – 192.168.10.1
PC-1 – 192.168.10.2
PC-2 – 192.168.10.3
PC-3 – 192.168.10.4
PC-4 – 192.168.10.5

PDU – PC0 to PC4.

Pinging 192.168.10.5 with 32 bytes of data
Ping Statistics for 192.168.10.5
Packets: Sent = 4, Received = 4, Lost = 0 (0% Loss).
Approximate round trip times in milliseconds:
Minimum = 10 ms, Maximum = 10ms, Average = 10ms

Program 13

Write a program for congestion control using Leaky bucket algorithm.

Code:

```
#include <stdio.h>

int main() {
    int incoming, outgoing, bucket_size, n;
    int store = 0;

    printf("Enter bucket size: ");
    scanf("%d", &bucket_size);

    printf("Enter outgoing rate: ");
    scanf("%d", &outgoing);

    printf("Enter number of inputs: ");
    scanf("%d", &n);

    while (n > 0) {
        printf("\nEnter the incoming packet size: ");
        scanf("%d", &incoming);

        if (incoming <= (bucket_size - store)) {
            // store incoming packets
            store += incoming;
            printf("Bucket buffer size = %d out of %d\n", store, bucket_size);
        } else {
            // drop extra packets
            int dropped = incoming - (bucket_size - store);
            printf("Dropped packets = %d\n", dropped);

            store = bucket_size;  // bucket is full now
            printf("Bucket buffer size = %d out of %d\n", store, bucket_size);
        }

        // Leak/outgoing
        store = store - outgoing;
        if (store < 0)
            store = 0;

        printf("After outgoing, %d packets left in buffer out of %d\n",
               store, bucket_size);
```

```
        n--;
    }

    return 0;
}
```

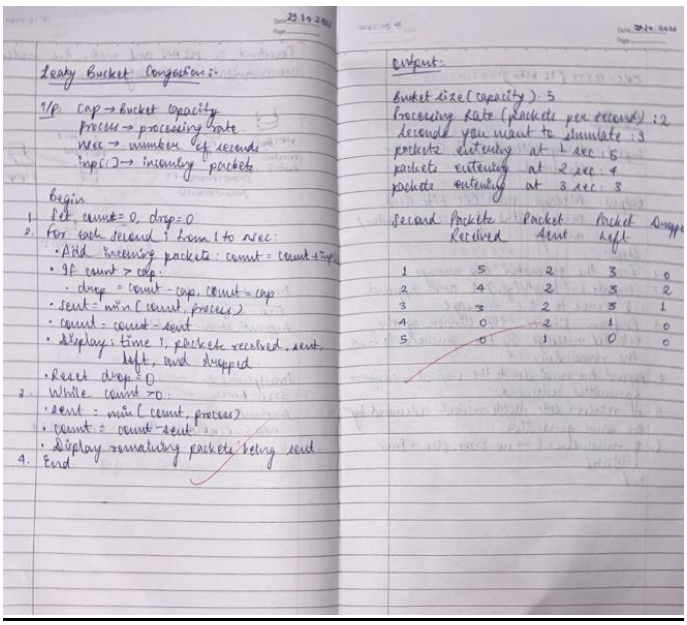
Input:

```
Enter bucket size: 10
Enter outgoing rate: 4
Enter number of inputs: 1
Enter the incoming packet size: 8
```

Output:

```
Enter bucket size: 10
Enter outgoing rate: 4
Enter number of inputs: 1

Enter the incoming packet size: 8
Bucket buffer size = 8 out of 10
After outgoing, 4 packets left in buffer out of 10
```

Observation:



Program 14

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

Code:

TCP server

```
# tcp_server.py
import socket

# Step 1: Create a TCP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Step 2: Bind it to an address and port
server_socket.bind(('localhost', 8080))

# Step 3: Listen for client connection
server_socket.listen(1)
print("Server is listening on port 8080...")

# Step 4: Accept client connection
conn, addr = server_socket.accept()
print("Connected by:", addr)

# Step 5: Receive file name from client
filename = conn.recv(1024).decode()

try:
    # Step 6: Open file and read contents
    with open(filename, 'r') as f:
        data = f.read()
        conn.send(data.encode())  # Send file contents
except FileNotFoundError:
    conn.send(b"File not found on server.")

# Step 7: Close connection
conn.close()
server_socket.close()
```

TCP Client

```
# tcp_client.py
import socket

# Step 1: Create a TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Step 2: Connect to the server
client_socket.connect(('localhost', 8080))

# Step 3: Send filename
filename = input("Enter filename to request: ")
client_socket.send(filename.encode())

# Step 4: Receive file content
data = client_socket.recv(4096).decode()
print("\n--- File Content ---\n")
print(data)

# Step 5: Close connection
client_socket.close()
```

Input:

```
Enter filename to request: sample.txt
```

```
Hello TCP File Transfer!
This is a sample file.
```

Output:

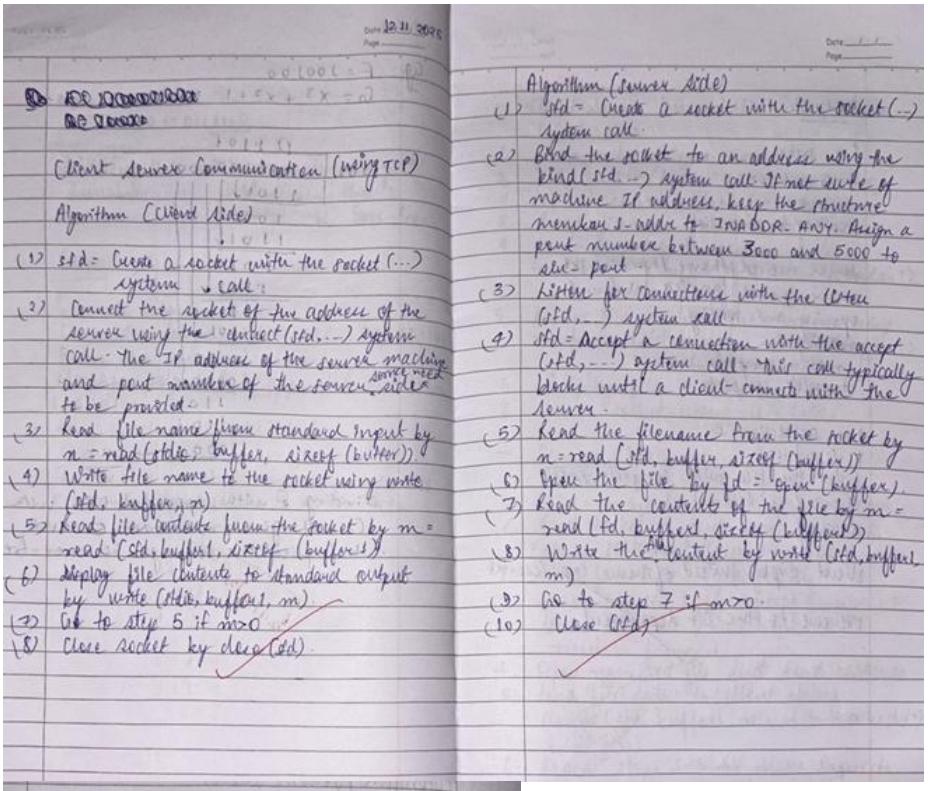
Server output

```
Server is listening on port 8080...
Connected by: ('127.0.0.1', 53412)
```

Client output

```
Enter filename to request: sample.txt

--- File Content ---

Hello TCP File Transfer!
This is a sample file.
```

Observation:



Client Server Communication (using TCP)

Algorithm (Client side)

(1) sfd = Create a socket with the socket(...) system call
(2) Connect the socket of the address of the server using the connect(sfd,...) system call. The IP address of the server machine and port number of the server application to be provided.
(3) Read file name from standard input by n = read(sfile, buffer, sizeof(buffer))
(4) Write file name to the socket using write (sfd, buffer, n)
(5) Read file contents from the socket by m = read(sfd, buffer, sizeof(buffer))
(6) Display file contents to standard output by write(sfile, buffer, m)
(7) Go to step 5 if m>0
(8) Close socket by close(sfd)

Algorithm (Server side)

(1) sfd = Create a socket with the socket(...) system call
(2) Bind the socket to an address using the bind(sfd,...) system call. For most of machine, if address, keep the structure member s_addr to INADDR_ANY. Assign a port number between 3000 and 5000 to sin_port
(3) Listen for connections with the listen(sfd,...) system call
(4) sfd = Accept a connection with the accept(sfd,...) system call. This call typically blocks until a client connects with the server.
(5) Read the filename from the socket by n = read(sfd, buffer, sizeof(buffer))
(6) Open the file by fd = open(buffer)
(7) Read the contents of the file by m = read(fd, buffer, sizeof(buffer))
(8) Write the content by write(sfd, buffer, m)
(9) Go to step 7 if m>0
(10) Close(sfd)

Output:

```
$ cc socketserver.c
$ ./a.out 1025
server:
waiting for connection

server received: /home/aps/cse.txt
server: /home/aps/cse.txt found.
opening and reading..
reading..
...reading complete
transfer complete

$ cc socketclient.c
$ ./a.out 1025
Enter the file with complete path
/home/aps/cse.txt
Reading..

client: display content of /home/aps/cse.txt
...
Welcome to the CSE department
.
```

Program 15

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.

Code:

UDP Server

```
# udp_server.py
import socket

# Step 1: Create UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Step 2: Bind it to an address
server_socket.bind(('localhost', 8081))
print("UDP Server is ready...")

while True:
    # Step 3: Receive filename from client
    filename, addr = server_socket.recvfrom(1024)
    filename = filename.decode()
    print(f"Requested file: {filename}")

    try:
        # Step 4: Read file and send data
        with open(filename, 'r') as f:
            data = f.read()
            server_socket.sendto(data.encode(), addr)
    except FileNotFoundError:
        server_socket.sendto(b"File not found on server.", addr)
```

UDP Client

```
# udp_client.py
import socket

# Step 1: Create UDP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 8081)
filename = input("Enter filename to request: ")

# Step 2: Send filename to server
client_socket.sendto(filename.encode(), server_address)
```

```
# Step 3: Receive response
data, _ = client_socket.recvfrom(4096)
print("\n--- File Content ---\n")
print(data.decode())

# Step 4: Close socket
client_socket.close()
```

Input:

```
Enter filename to request: notes.txt
```

```
This is a UDP file transfer example.
Hello from server!
```

Output:

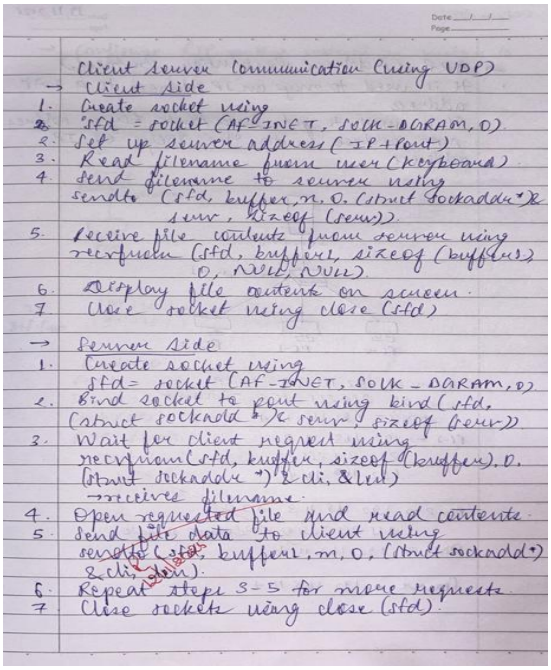
Server output

```
UDP Server is ready...
Requested file: notes.txt
```

Client output

```
Enter filename to request: notes.txt

--- File Content ---

This is a UDP file transfer example.
Hello from server!
```

Observation:


Client Server Communication Using UDP)
→ Client Side
1. Create socket using
sfd = socket (AF_INET, SOCK_DGRAM, 0)
2. Set up server address (IP+Port)
3. Read filename from user (keyboard)
4. Send filename to server using
sendto (sfd, buffer, n, 0, (struct sockaddr*)&
serv, sizeof (serv))
5. Receive file contents from server using
recvfrom (sfd, buffer, sizeof (buffer)
0, NULL, NULL)
6. Display file content on screen.
7. Close socket using close (sfd)
→ Server Side
1. Create socket using
sfd = socket (AF_INET, SOCK_DGRAM, 0)
2. Bind socket to port using bind (sfd,
(struct sockaddr *)& serv, sizeof (serv))
3. Wait for client request using
recvfrom(sfd, buffer, sizeof (buffer), 0,
(struct sockaddr *)& cli, &len)
→ receive filename
4. Open requested file and read contents
5. Send file data to client using
sendto (sfd, buffer, n, 0, (struct sockaddr*)
&cli, len).
6. Repeat steps 3-5 for more requests
7. Close sockets using close (sfd)

Program 16

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

Code:

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

char data[50], poly[50], check_value[50];
int data_length, poly_length;

void XOR() {
    for (int i = 1; i < poly_length; i++) {
        check_value[i] = (check_value[i] == poly[i]) ? '0' : '1';
    }
}

void crc() {
    // Copy initial bits to check_value
    for (int i = 0; i < poly_length; i++)
        check_value[i] = data[i];

    int i = poly_length;

    do {
        if (check_value[0] == '1')
            XOR();

        // Shift left
        for (int j = 0; j < poly_length - 1; j++)
            check_value[j] = check_value[j + 1];

        check_value[poly_length - 1] = data[i];
        i++;

    } while (i <= data_length + poly_length - 1);
}

void receiver() {
    printf("\nEnter the received data: ");
    scanf("%s", data);

    printf("Data received: %s\n", data);
```

```
    crc();

    int error = 0;
    for (int i = 0; i < poly_length - 1; i++) {
        if (check_value[i] == '1') {
            error = 1;
            break;
        }
    }

    if (error)
        printf("❌ Error detected in received data!\n");
    else
        printf("✅ No error detected (data correct).\n");
}

int main() {
    printf("Enter data to be transmitted: ");
    scanf("%s", data);

    printf("Enter the divisor polynomial: ");
    scanf("%s", poly);

    data_length = strlen(data);
    poly_length = strlen(poly);

    // Pad with n-1 zeros
    for (int i = data_length; i < data_length + poly_length - 1; i++)
        data[i] = '0';

    data[data_length + poly_length - 1] = '\0';

    printf("\nData padded with n-1 zeroes: %s\n", data);

    crc();

    printf("CRC value is: ");
    for (int i = 0; i < poly_length - 1; i++)
        printf("%c", check_value[i]);

    // Append CRC to data
    for (int i = data_length, j = 0; i < data_length + poly_length - 1; i++, j++)
```

```
        data[i] = check_value[j];

    printf("\nFinal codeword to be sent: %s\n", data);

    receiver();
    return 0;
}
```

Input:

```
Enter data to be transmitted: 1101
Enter the divisor polynomial: 1011
```

Output:

```
Enter data to be transmitted: 1101
Enter the divisor polynomial: 1011

Data padded with n-1 zeroes: 1101000
CRC value is: 011
Final codeword to be sent: 1101011

Enter the received data: 1101011
Data received: 1101011
✅ No error detected (data correct).
```

Observation: