

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB RECORD

### Computer Network Lab (23CS5PCCON)

Submitted by

Harbakshish Singh Arora (1BM23CS104)

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

September 2025 – January 2026

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 Harbakshish Singh Arora (1BM23CS104) 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

Sl. No.	Date	Experiment Title	Page No.
PART A			
1	19/08/25	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	4-7
2	9/09/25	Configure DHCP within a LAN and outside LAN.	8-12
3	9/09/25	Configure Web Server, DNS within a LAN.	13-16

4	09/09/25	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	17-19
5	23/09/25	Configure default route, static route to the Router.	19-25
6	23/09/25	Configure RIP routing Protocol in Routers.	26-28
7	14/10/25	Configure OSPF routing protocol.	28-30
8	14/10/25	To construct a VLAN and make the PC's communicate among a VLAN.	30-33
9	11/11/25	To construct a WLAN and make the nodes communicate wirelessly.	33-36
10	11/11/25	Demonstrate the TTL/ Life of a Packet.	36-38
11	18/11/25	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	38-40
12	18/11/25	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	40-42
		PART B	
1	28/10/25	Write a program for congestion control using Leaky bucket algorithm.	42-48
2	17/11/25	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.	48-50
3	17/11/25	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.	50-51
4	28/10/25	Write a program for error detecting code using CRCCCITT (16-bits).	51-55

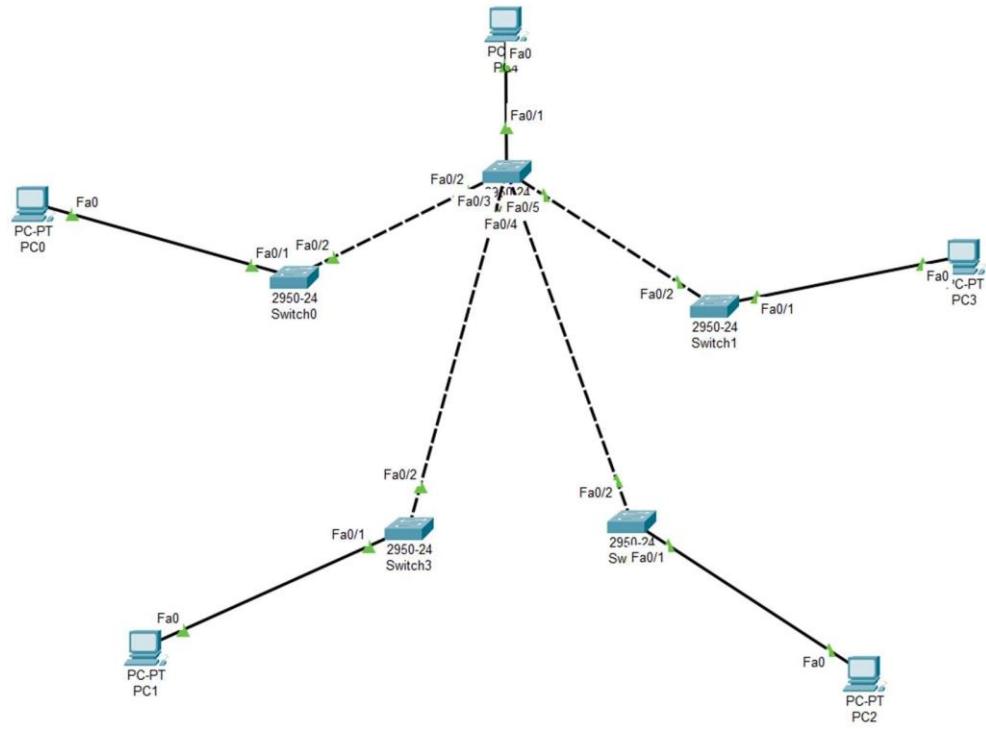
## PART-A

Program – 1:

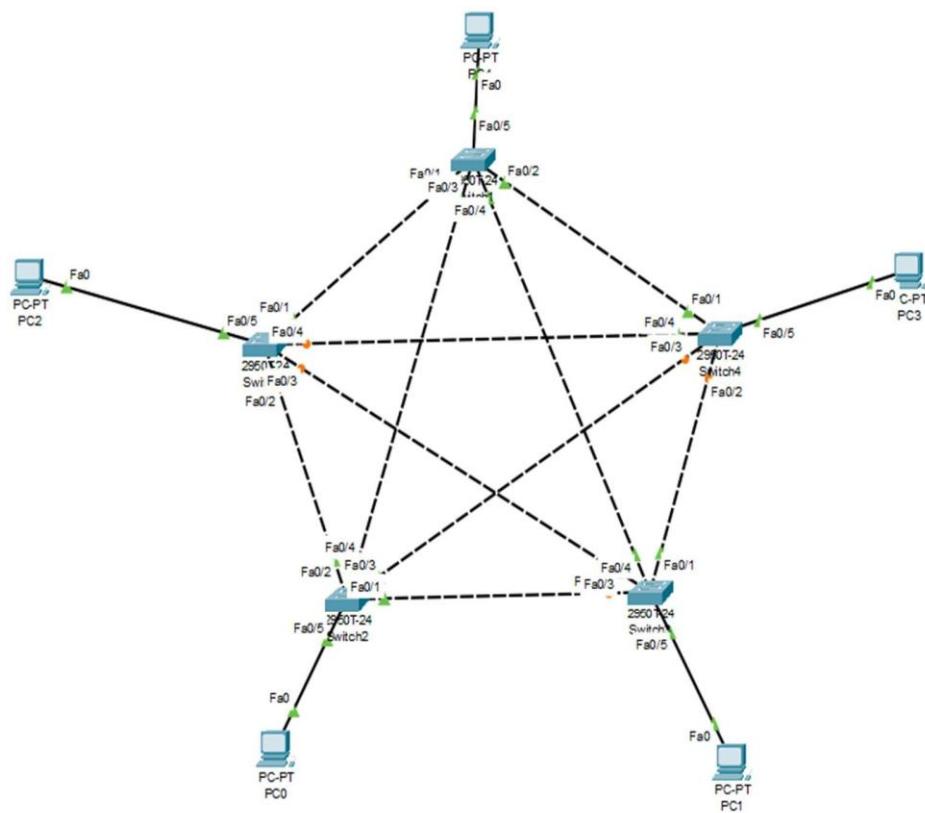
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network Diagram:

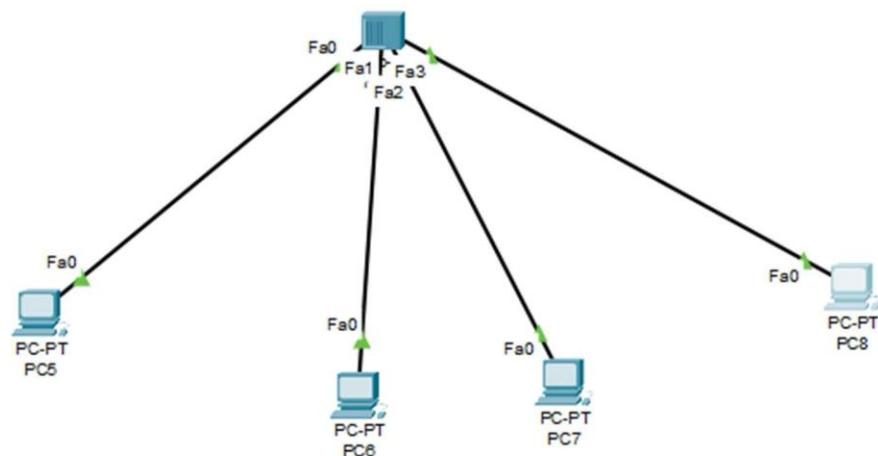
Star Topology:



Mesh Topology:



Hub-Based Network Topology:



## Configuration

LAB-1

Date \_\_\_\_\_  
Page \_\_\_\_\_

\* Create a Topology and stimulate sending a PDU destined to destination using hub and switch ad connecting devices and demonstrate ping message in Cisco packet Traces.

1. Star Topology using Switch

① Name : Implementation of star Topology using Cisco packet Traces.

**PROCEDURE:**

1. Open a new object project in Packet Traces
2. Drag and drop:
  - a. Central Switch (Switch 0)
  - b. 4 Edge switches (Switch 1, Switch 2, Switch 3, Switch 4, Switch 5)
  - c. 5 PCs (PC 1, PC 2, PC 3, PC 4, PC 5)
3. Connect each PC to its respective switch using cables straight-through cables
4. Connect all edge switches to the central switch.
5. Assign IP addresses:
  - PC 1 → 192.168.1.2
  - PC 2 → 192.168.1.3
  - PC 3 → 192.168.1.4
  - PC 4 → 192.168.1.5
  - PC 5 → 192.168.1.6
6. Subnet mask: 255.255.255.0 is already entered

**Simulation:**

1. Switch to simulation mode.
2. Click 'File' → Add Simple PDU (Converge icon) tool.
3. Select source PC → Then orientation PC.
4. In the event list, a packet creation dialog will appear.
5. Click on Auto Capture, Play or capture forward to watch packet flow.
6. With switch, the packet travels only to the intended destination PC, not to all.

destination PC, not to all

PC - P1  
PC - P2  
PC - P3  
PC - P4  
PC - P5

2950-24  
Switch 0

2950-24  
Switch 1

2950-24  
Switch 2

2950-24  
Switch 3

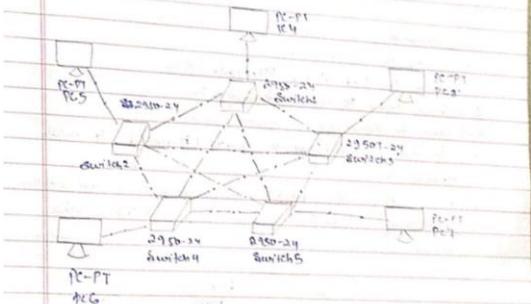
2950-9  
Switch 4

2950-9  
Switch 5

PC - P1  
PC - P2  
PC - P3  
PC - P4  
PC - P5

② Mesh Topology using Switches:

1. Stimulate.  
 2. Switch to simulation mode.  
 3. Click the Add simple PDU tool.  
 4. Choose source PC → then destination PC.  
 5. In the Event list → packet transmission will be logged.  
 6. Use capture / play or capture / forward to observe the packet flow.  
 7. In mesh topology, packets have multiple possible paths between switches, if one link fails, an alternate path is used.



### (b) Hub-based Network Topology:

Name : Implementation of Hub-based Network Topology in Cisco Packet Tracer.

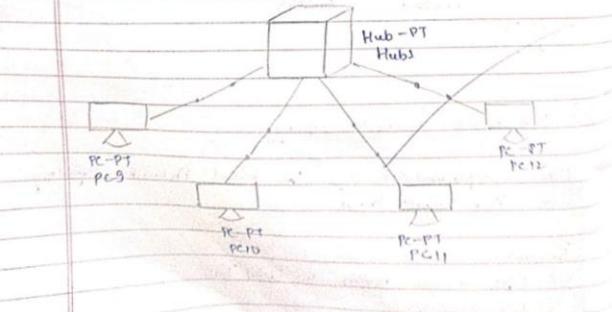
#### Procedure -

1. Open a new project in Packet Tracer.
2. Stage 2 dropdown:  
 1. Hub (Hub)

- (MS (PC9, PC10, PC11, PC12))
3. Connect all PCs to the hub using straight-through cables.
  4. Assign IP addresses:
    - PC9 → 192.168.1.2
    - PC10 → 192.168.1.3
    - PC11 → 192.168.1.4
    - PC12 → 192.168.1.5
  - (Subnet mask: 255.255.255.0)

#### Simulation:

1. Switch to Simulation mode.
2. Click the Add simple PDU tool.
3. Choose source PC → then destination PC.
4. Choose the Event list for packet creation.
5. Use auto capture / play or capture / forward to watch transmission.
6. With a hub, the packet is broadcast to all devices but only the destination PC accepts it while others discard it.



## Output:

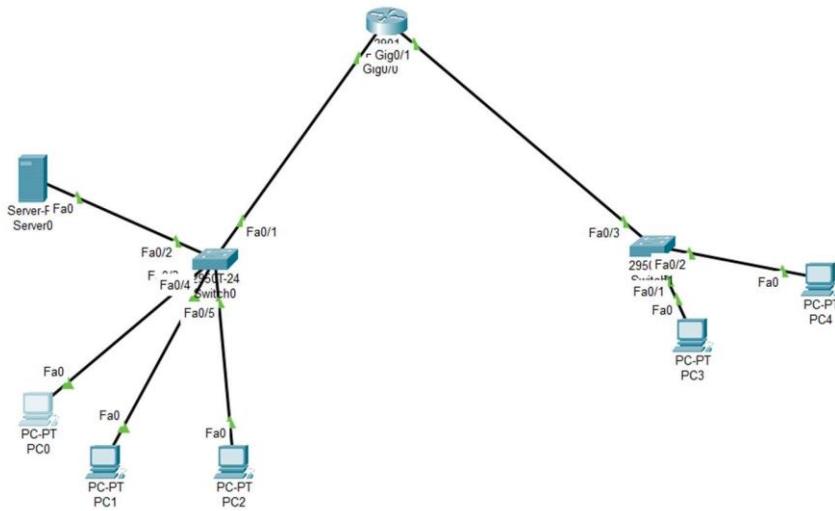


Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
Successful		PC2	PC4	ICMP	Green	0.000	N	0	(edit)	(delete)
Successful		PC3	PC2	ICMP	Green	0.000	N	1	(edit)	(delete)
Successful		PC4	PC3	ICMP	Green	0.000	N	2	(edit)	(delete)

## Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



### Configuration:

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

Steps:

- ① Click on DHCP server → Desktop → IP config. → Static →
- ② DHCP 192.168.10.2. and Gateway → 192.168.10.1

Services → DHCP → Pool name : SwitchOne      SwitchTwo  
 Gateway : 192.168.10.2      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

[ADD] [ADD]

Routes:

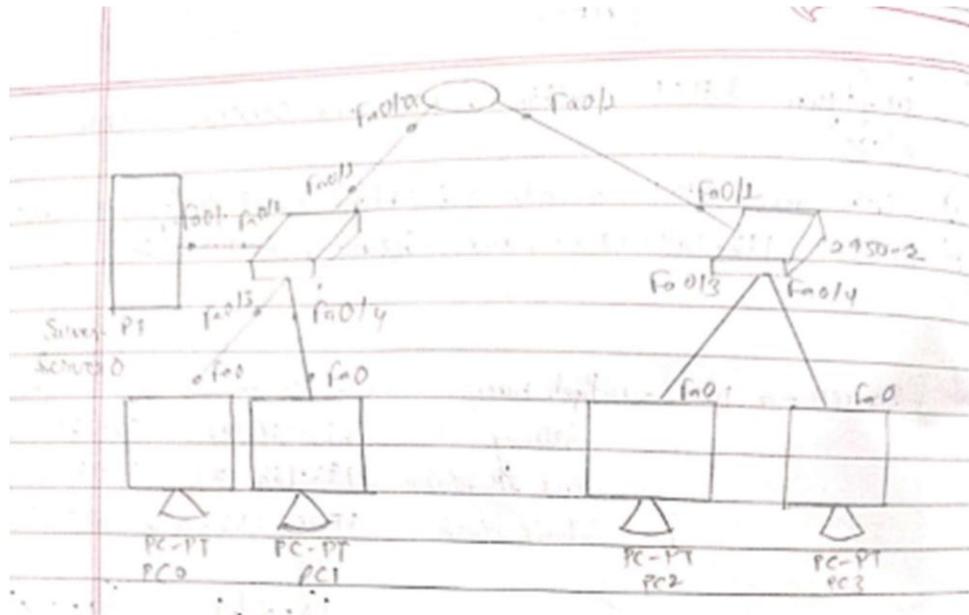
```

  ↗ CLI
  ↗ no
  Router # enable.
  # config 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 shutdown
  end
  write memory
  
```

Note : - IP-helper-add is also called Relay Agent



### DHCP Concepts:

- **DHCP Server:** The device that assigns IP config.
- **DHCP Client:** A device that requests unknown setting from DHCP server.
- **IP lease:** The temporary assignment of an IP address to client
- **Scope / pool:** A range of IP address available for assignment

Output:

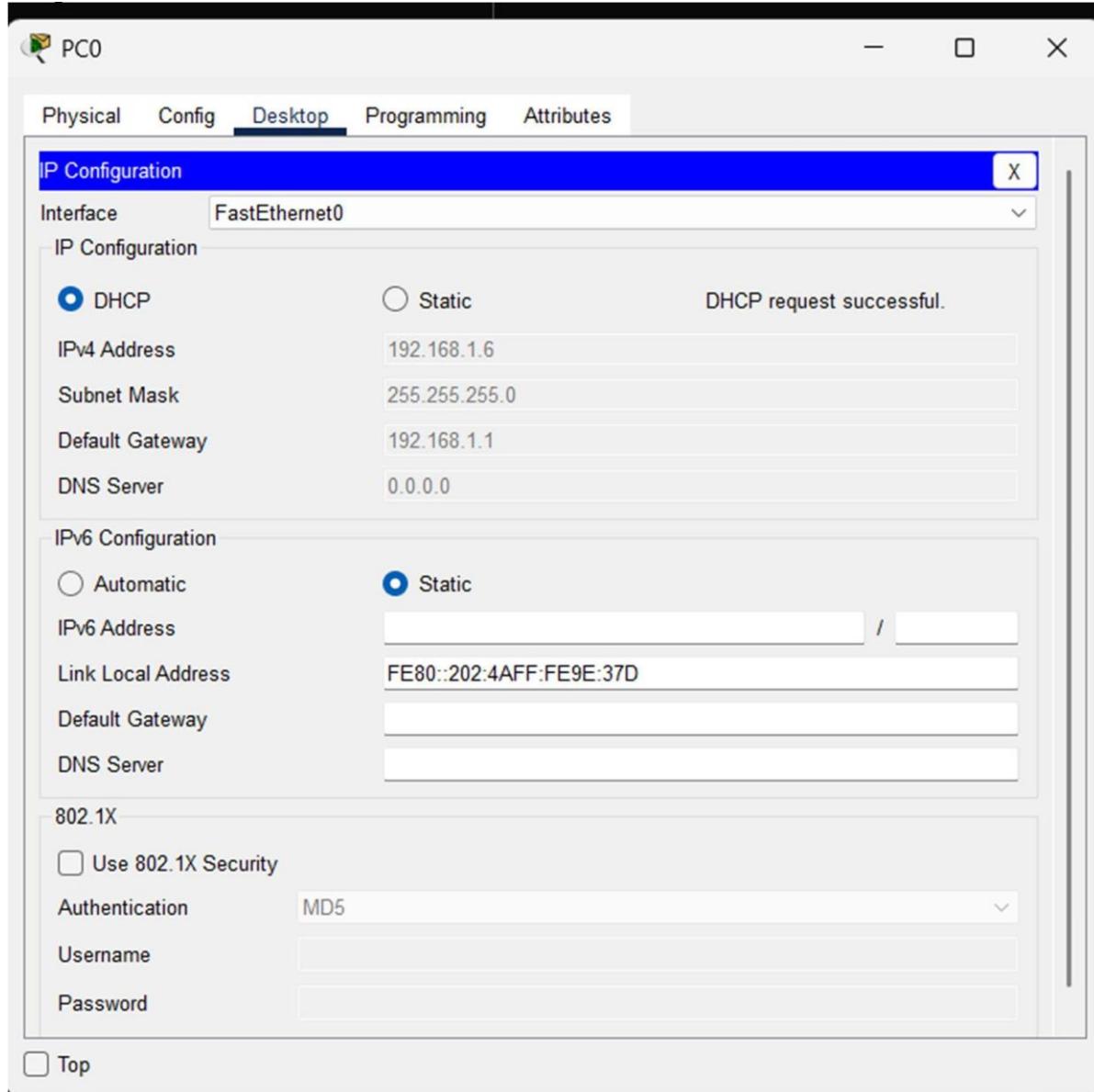


Fig 1. Ip address assigned by DHCP server within LAN (PC1)

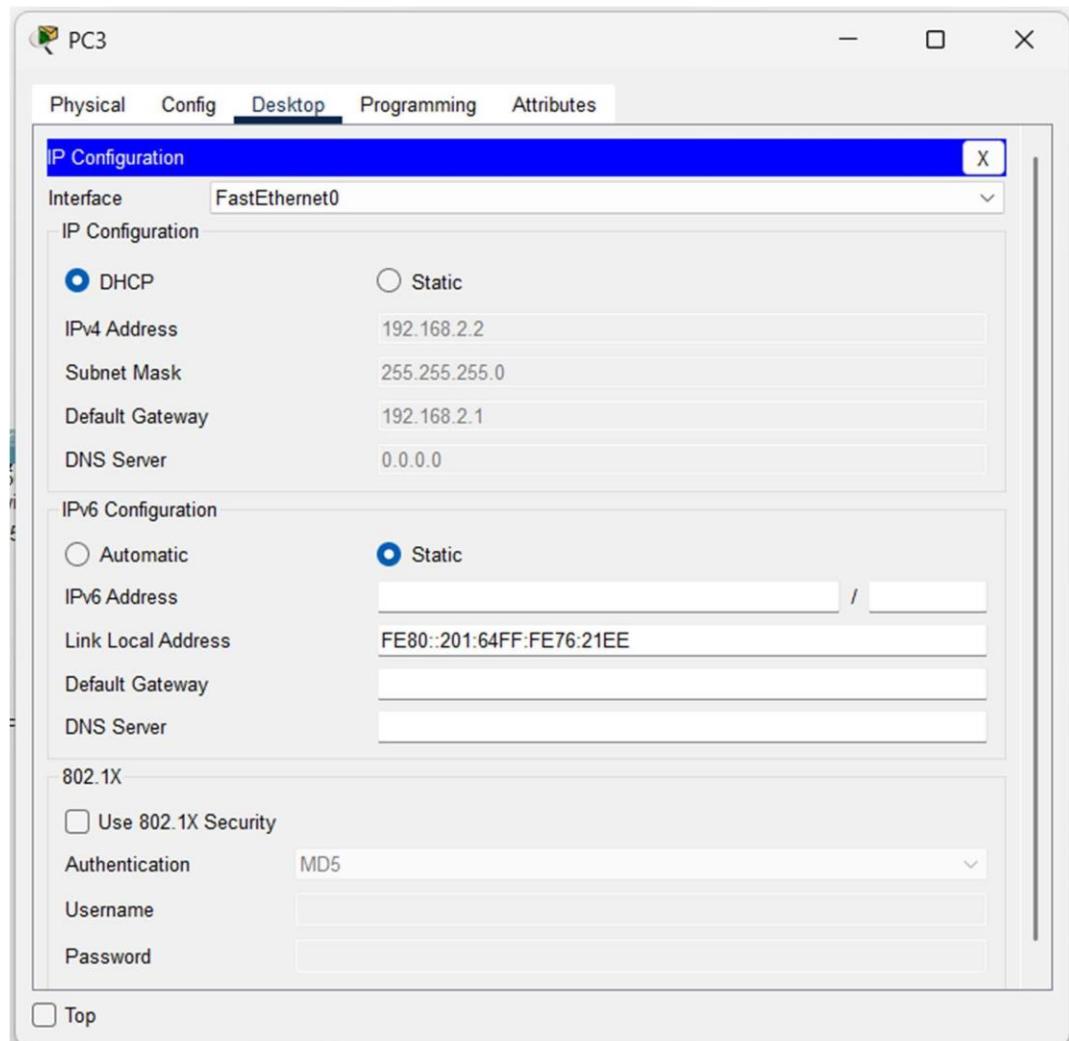
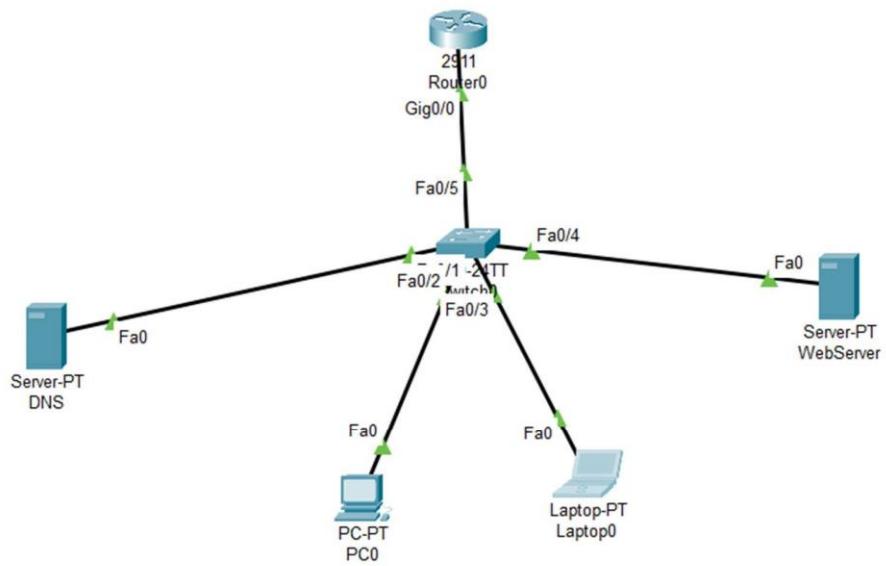


Fig 2. Ip address assigned by DHCP server outside LAN

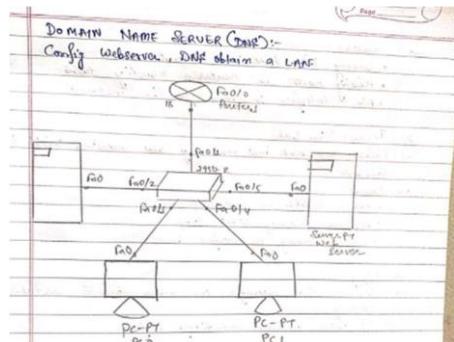
Program 3:

Aim: Configure Web Server, DNS within a LAN.

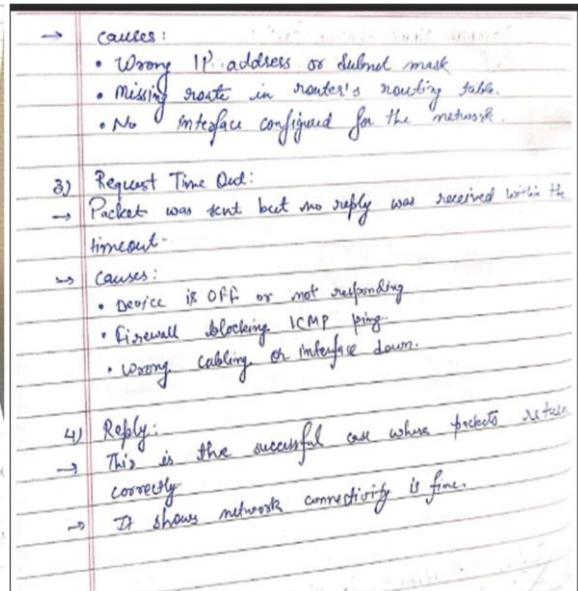
Network diagram:



Configuration:



- i) Configure IP addresses to the router in packet tracer.  
Explore the following:
  - (i) Ping Response
  - (ii) Destination Unreachable
  - (iii) Request Time Out
  - (iv) Reply
- ii) Ping Response:  
The destination is reachable. If addresses are correct and the network path works.  
Both source & destination are configured properly in network.
- iii) Destination Unreachable:  
Router / device knows the destination network does not exist or route to route to it.



## Output:

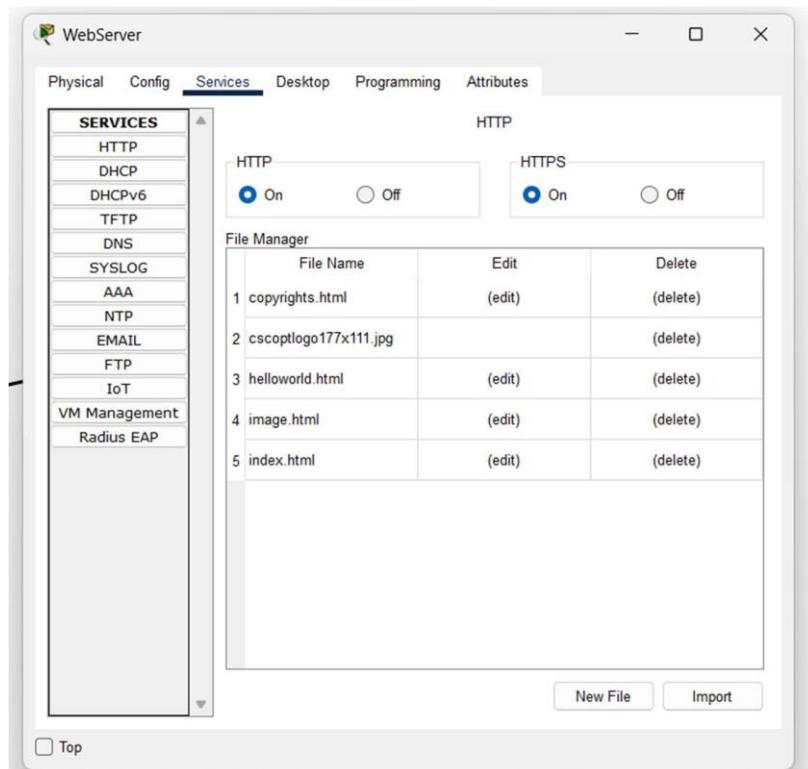


Fig 1. WEB server – HTTP Services

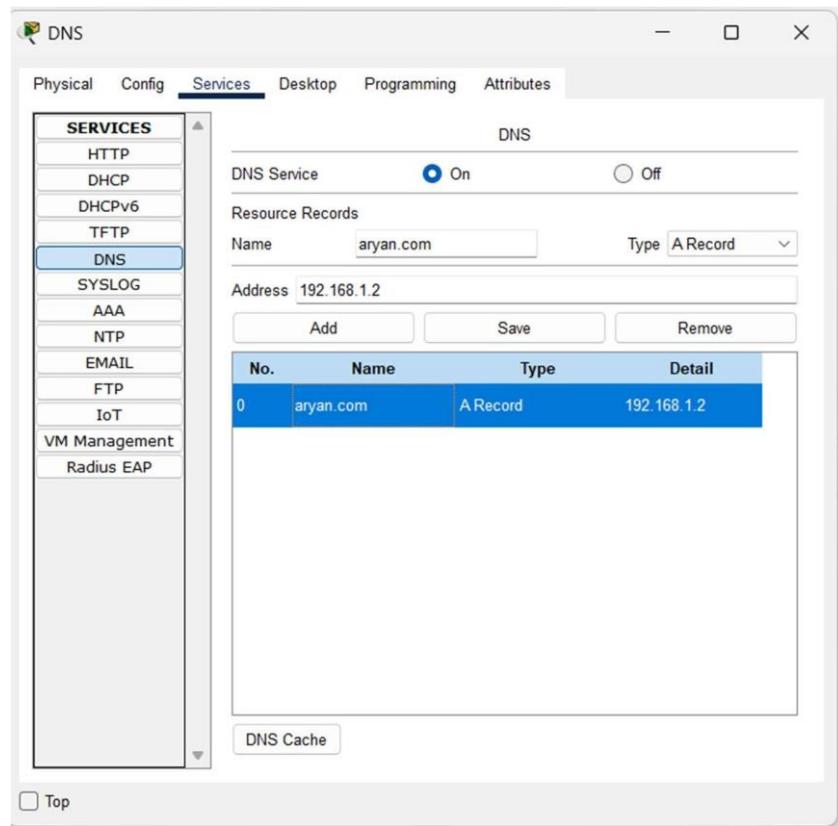


Fig 2. DNS server – DNS Services

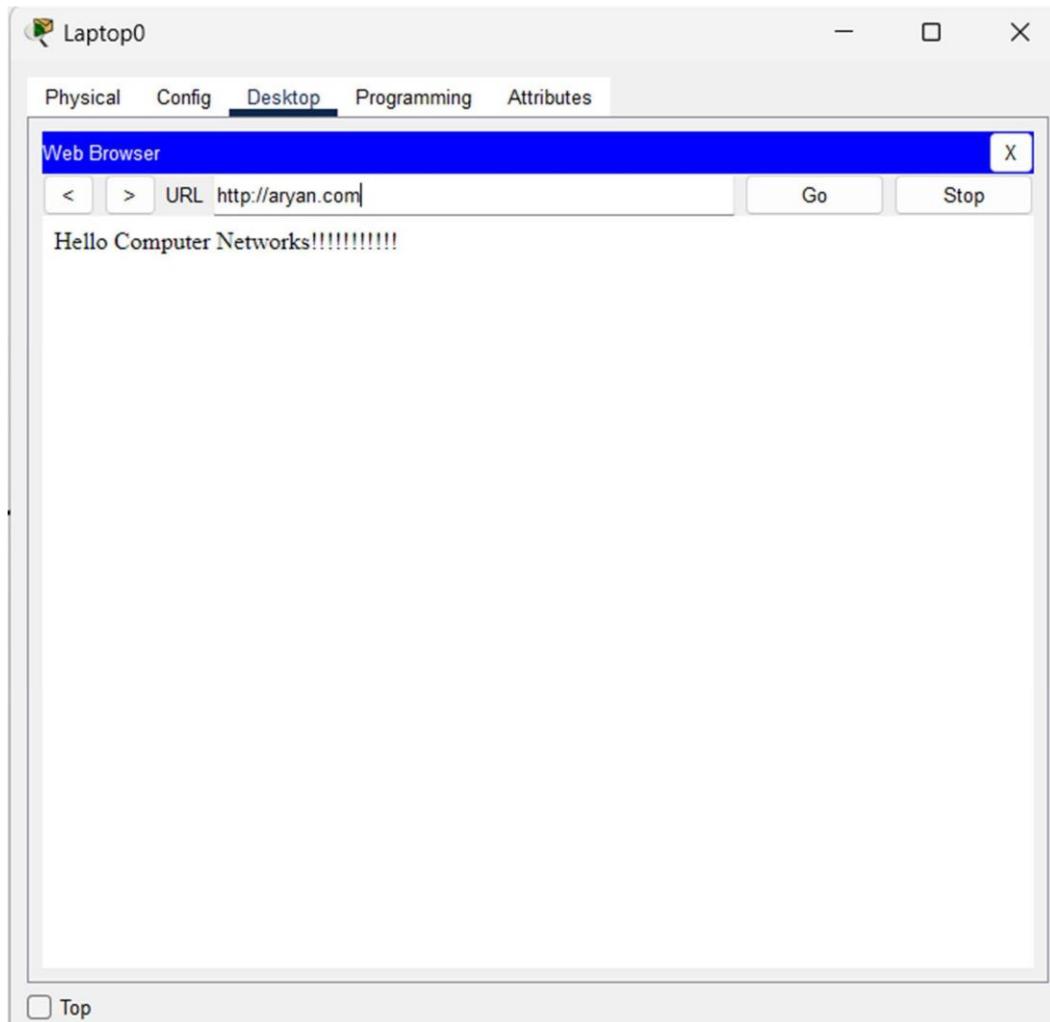
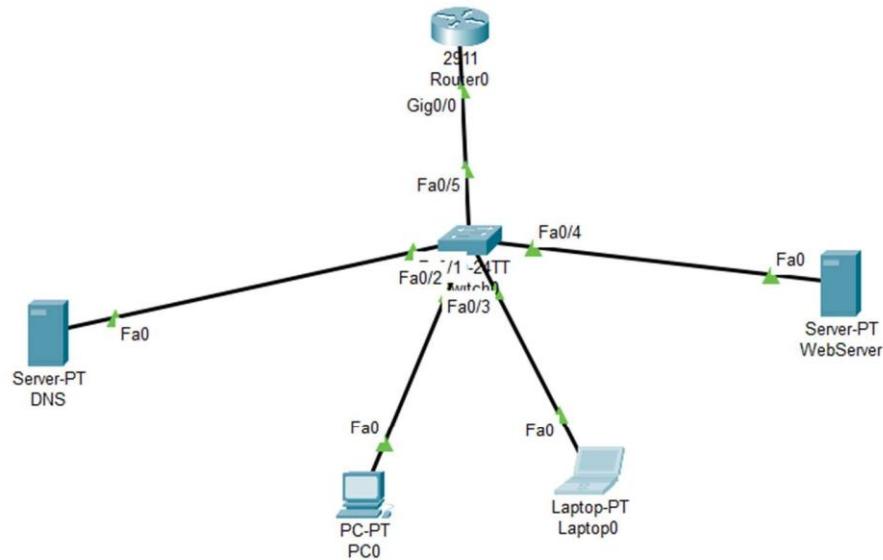


Fig 3. Laptop0 – accessing data from web browser

#### Program 4:

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

Network diagram:



Configuration:

- PC 0
- Q) Configure IP address to the router in packet tracer.  
Explore the following:
- Ping Response
  - Destination Unreachable
  - Request Time Out
  - Reply.
- B1P(i) Ping Response:**  
 The destination is reachable, IP addresses are correct and the network path works.  
 Both source & destination are configured properly by connected.
- B1P(ii) Destination Unreachable:**  
 Router / device knows the destination network does not exist or can't speak to it.
- Causes:
  - Wrong IP address or subnet mask
  - Missing route in router's routing table
  - No interface configured for the network
- Request Time Out:  
 → Packet was sent but no reply was received within the timeout.
- Causes:
  - Device is OFF or not responding
  - Firewall blocking ICMP ping
  - Wrong cabling or interface down.
- Reply:  
 → This is the successful case where packets return correctly.  
 → It shows network connectivity is fine.

Output:

Laptop0

Physical Config Desktop Programming Attributes

Command Prompt X

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<lms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.1.6

Pinging 192.168.1.6 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

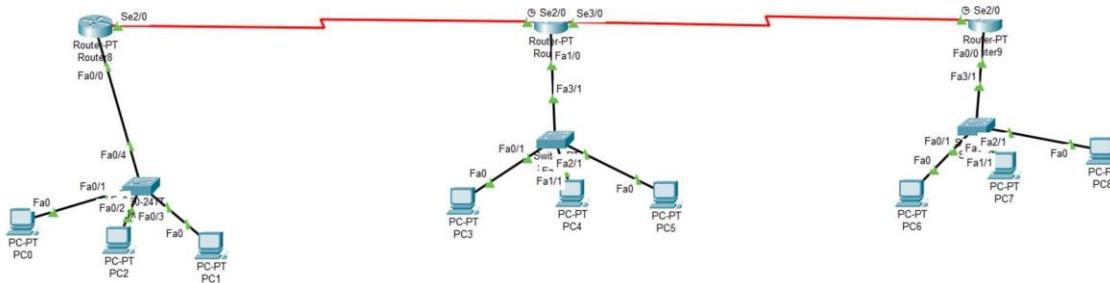
Ping statistics for 192.168.1.6:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

## Program 5:

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

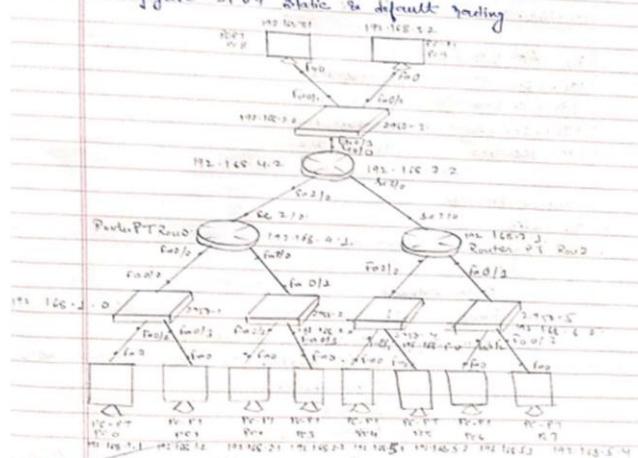
Network diagram:



## Configuration:

1. Static  $\rightarrow$  known data2. Dynamic  $\rightarrow$  unknown data

Q) Configure IPv4 static &amp; default routing.



## # STATIC ROUTES

Ro (1,2,4)

192.168.3.0 192.168.4.2

192.168.5.0 192.168.4.2

192.168.6.0 192.168.4.2

192.168.7.0 192.168.4.2

CLASSMATE  
Done Page

<b>R<sub>1</sub> (3,4,7)</b>	
192.168.1.0	192.168.4.1
192.168.2.0	192.168.4.2
192.168.5.0	192.168.2.1
192.168.6.0	192.168.2.2

<b>R<sub>2</sub> (5,6,7)</b>	
192.168.1.0	192.168.7.2
192.168.2.0	192.168.7.2
192.168.3.0	192.168.7.2
192.168.4.0	192.168.7.2

CONFIGURATIONS

→ Router Config  
   ↳ CLI  
     ↳ mode

Router > enable  
 Router > Config  
 Router (Config) # interface serial 2/0  
 # ip address 192.168.4.2 255.255.255.0  
 # no shutdown  
 # exit  
 # interface fast0/0  
 # ip address 192.168.1.0 255.255.255.0  
 # no shutdown  
 # exit  
 # interface fast1/0  
 # ip address 192.168.2.0 255.255.255.0  
 # no shutdown  
 # exit  
 write memory.

classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

→ R2 Config  
 ↳ CLT  
 ↳ Line 2

```

Router > enable
Router # config
Router (config) # host name R2
# interface Serial 2/0
# ip address 192.168.1.42.0
# 255.255.255.0
# no shutdown
# exit
# int Fa0/0
# ip address 192.168.2.0 255.255.255.0
# no shutdown
# exit
# exit
# Fa0/0
write memory
  
```

→ R3 Config  
 ↳ CLT  
 ↳ Line 3

```

Router > enable
Router # config
Router (config) # hostname R3
# interface Serial 2/0
# ip address 192.168.1.3 255.255.255.0
# no shutdown
# exit
# int Fa0/0
# ip address 192.168.3.0 255.255.255.0
  
```

WINDMILL  
Date \_\_\_\_\_  
Page \_\_\_\_\_

→ PC1 Config  
 ↳ Desktop

```

↳ IP Config
# IP Address 192.168.1.1 255.255.255.0
# Default Gateway 192.168.1.0
  
```

→ PC1 Config  
 ↳ Laptop

```

↳ IP Config
# IP Address 192.168.1.2 255.255.255.0
# Default Gateway 192.168.1.0
  
```

→ PC2 Config  
 ↳ Desktop

```

↳ IP Config
# IP Address 192.168.2.1 255.255.255.0
# Default Gateway 192.168.2.0
  
```

→ PC3 Config  
 ↳ Desktop

```

↳ IP Config
# IP Address 192.168.2.2 255.255.255.0
# Default Gateway 192.168.2.0
  
```

## Output:

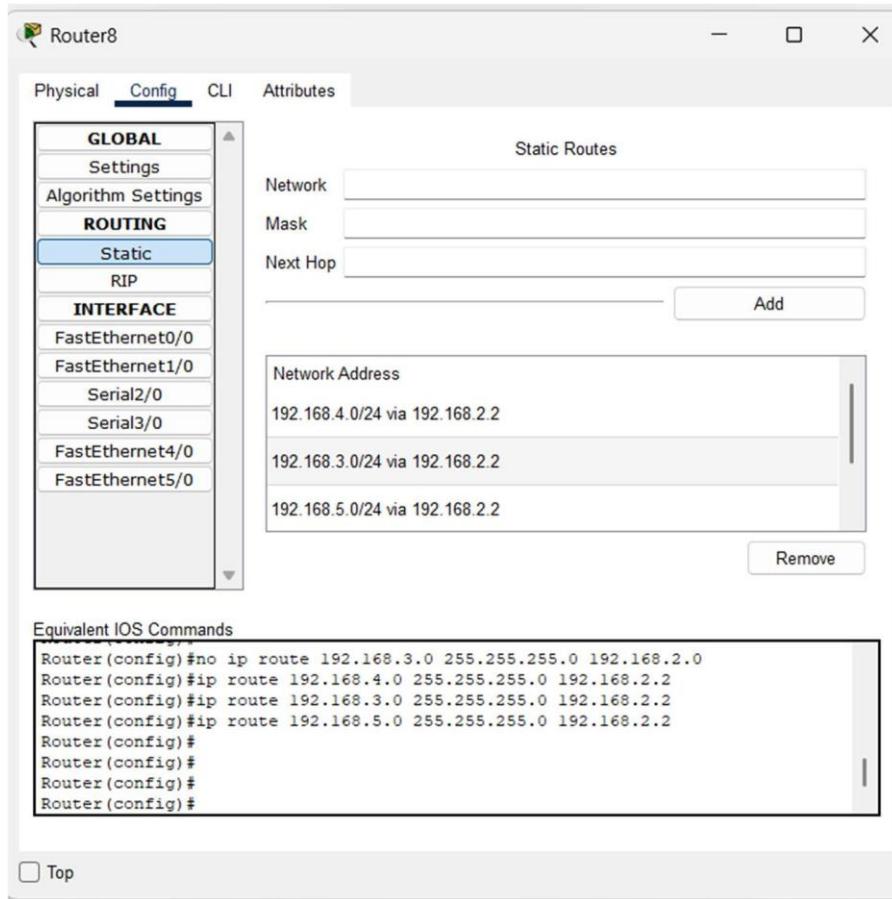


Fig 1. Router— Static routing

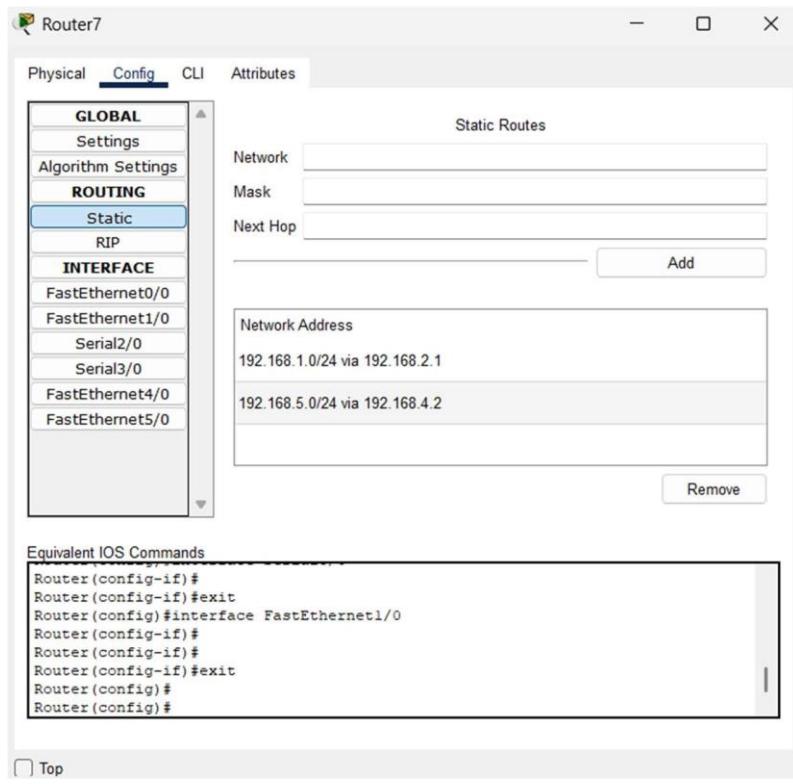


Fig 2. Router– Static routing

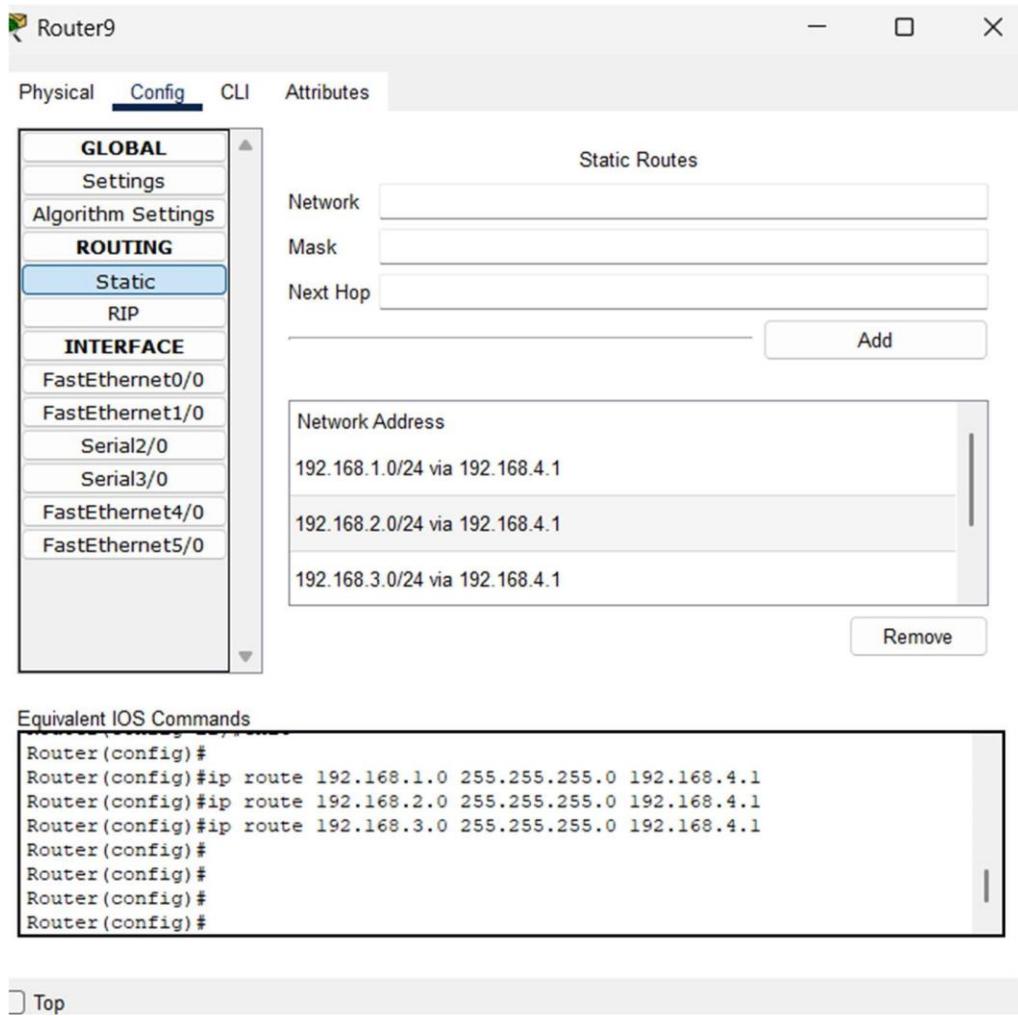
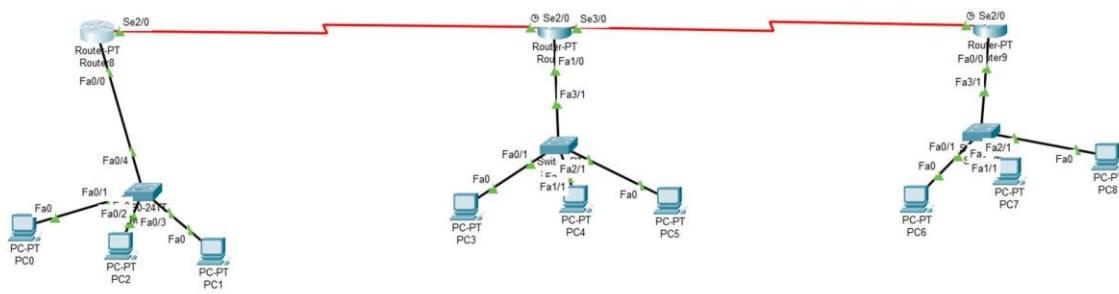


Fig 3. Router— Static routing

Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:

PC 4 Config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.3.1	255.255.255.0
# Default Gateway 192.168.3.0	
PC 5 Config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.3.2	255.255.255.0
# Default Gateway 192.168.3.0	
PC 6 config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.5.1	255.255.255.0
# Default Gateway 192.168.5.0	
PC 7 Config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.5.2	255.255.255.0
# Default Gateway 192.168.5.0	
PC 8 Config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.6.1	255.255.255.0
# Default Gateway 192.168.6.0	
PC 9 Config	
↳ Desktop	
↳ IP Config	
# IP Address 192.168.6.2	255.255.255.0
# Default Gateway 192.168.6.0	

Output:

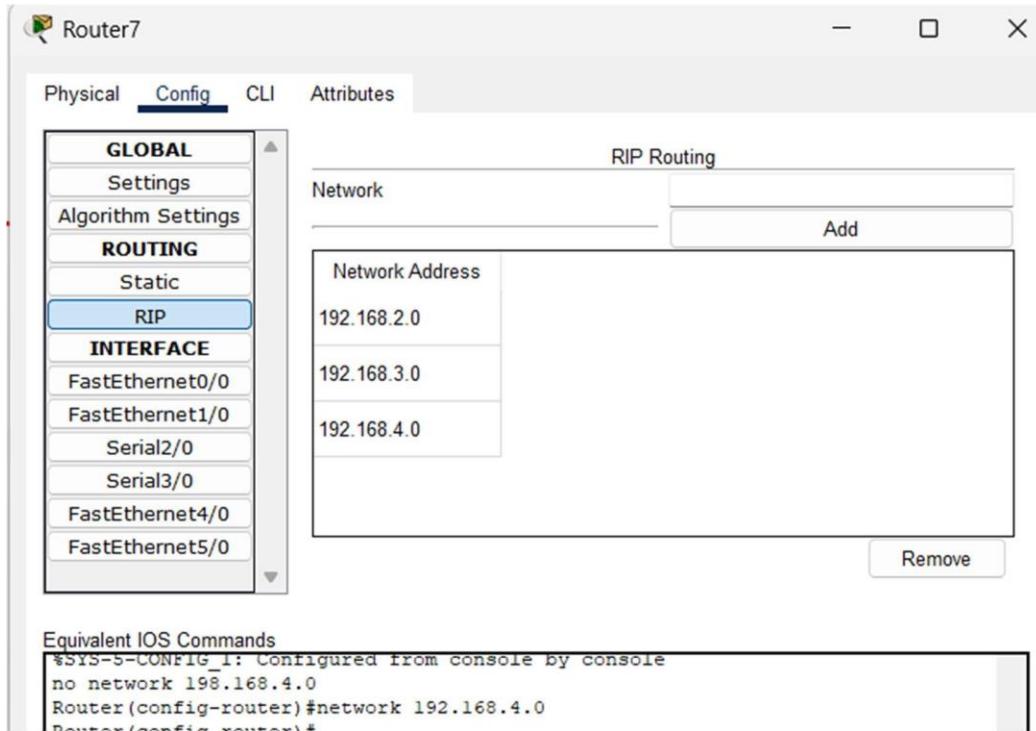


Fig 1. Router – RIP routing

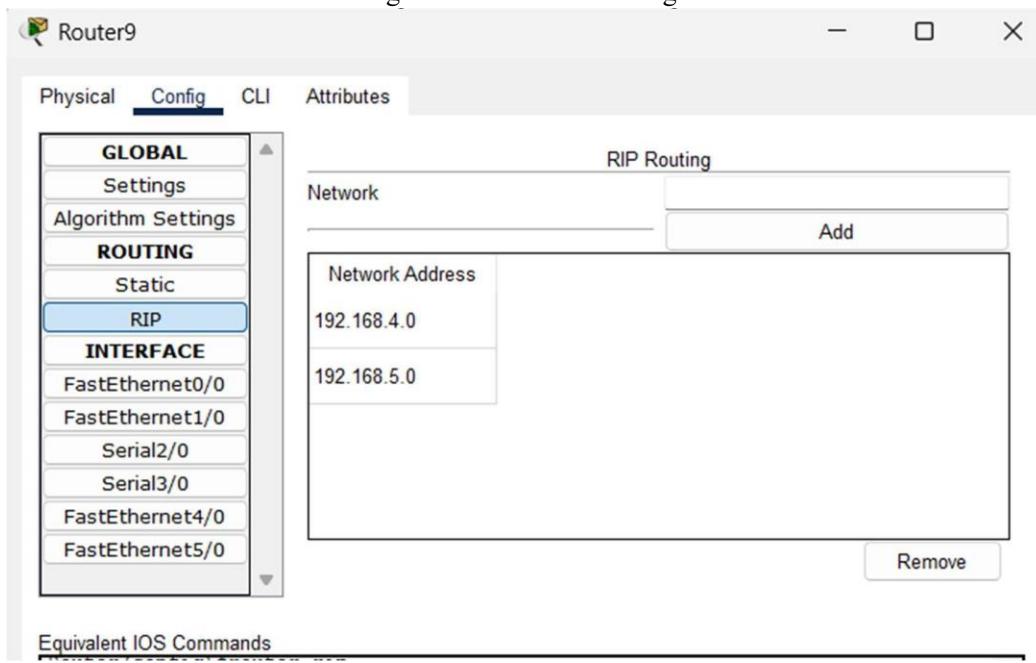


Fig 2. Router – RIP routing

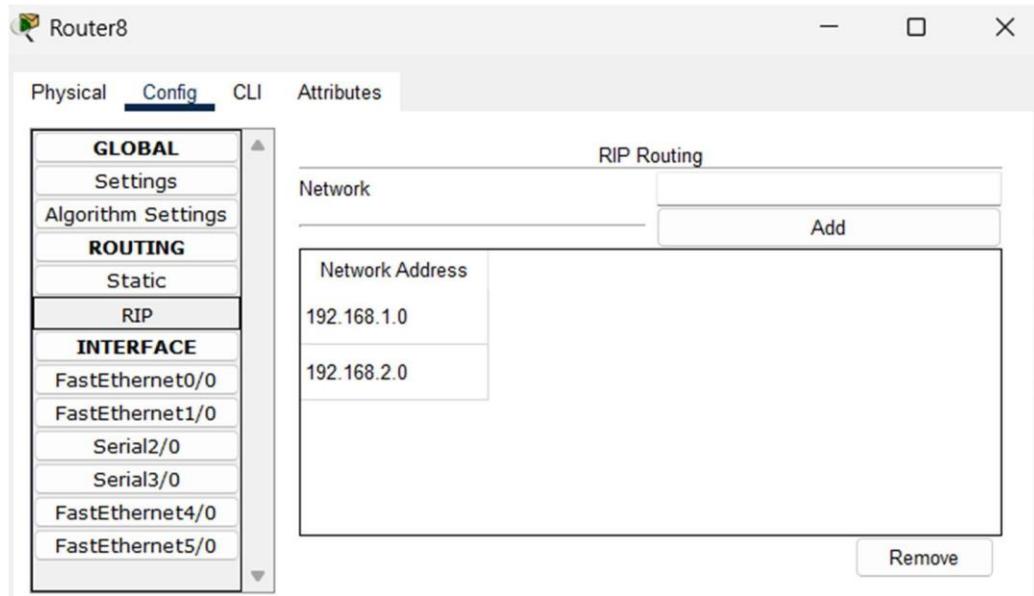
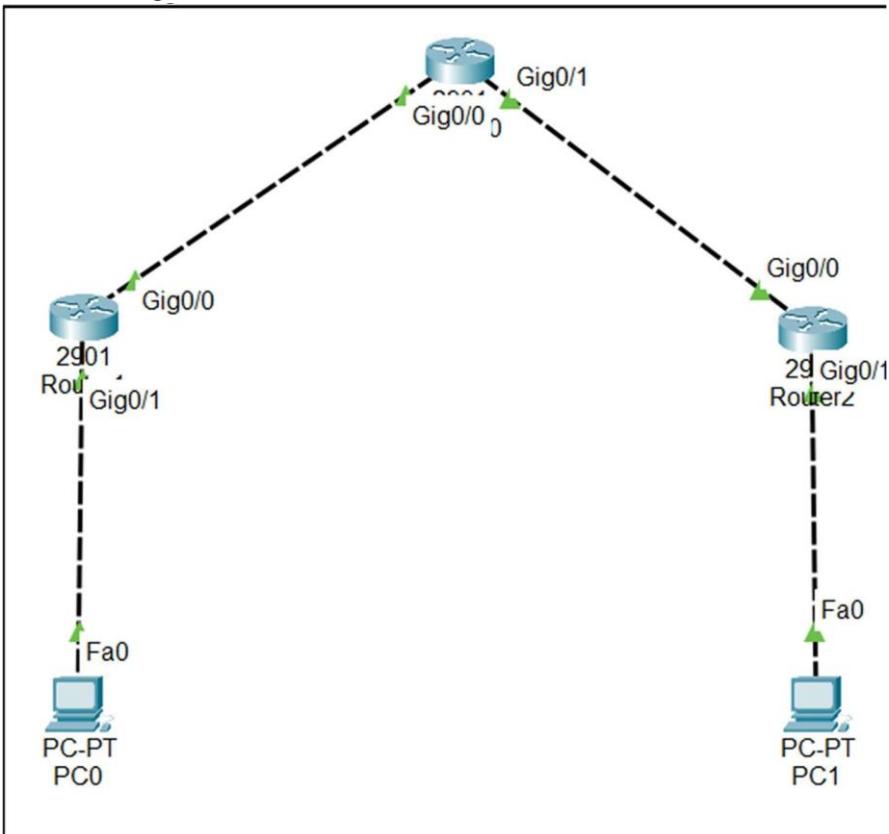


Fig 3. Router – RIP routing

Program 7:

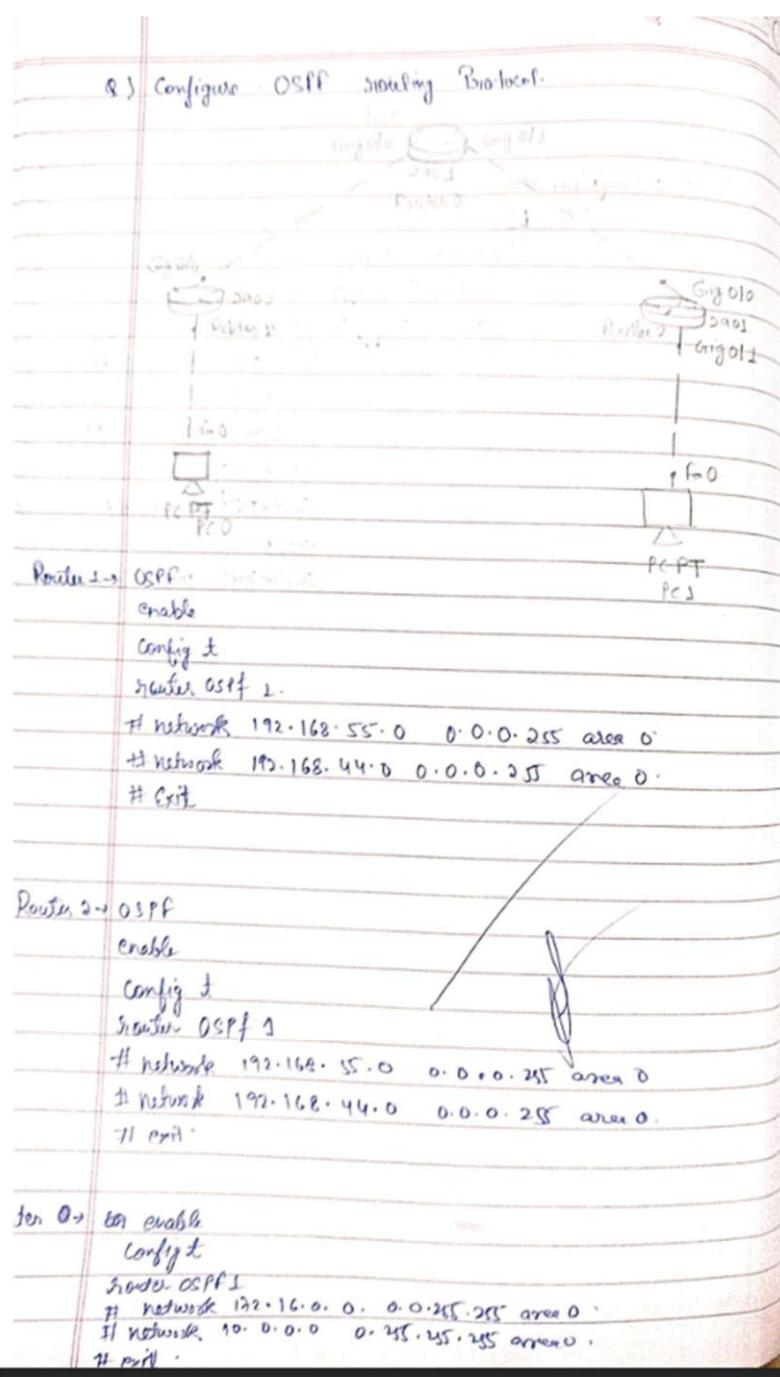
Aim: Configure OSPF routing protocol.

Network diagram:



Configuration:

Q3) Configure OSPF routing Protocol.



Output:

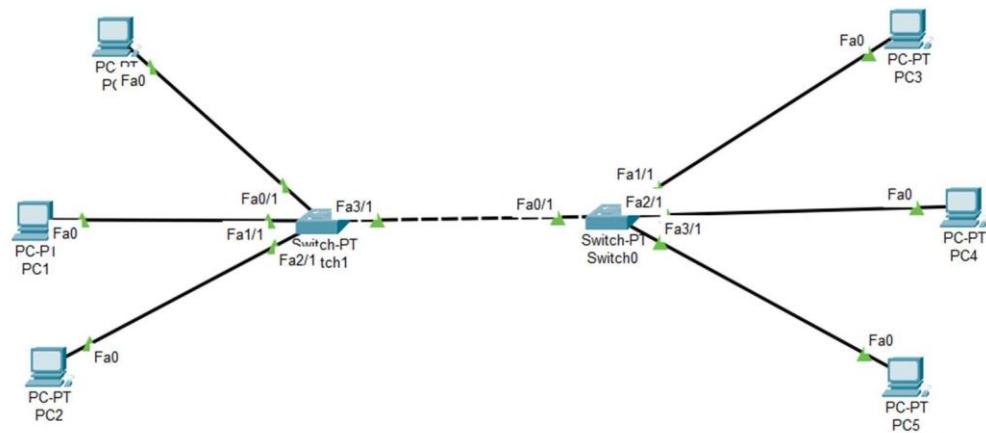
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
Successful	PC2	PC5	ICMP	■	0.000	N	3	(edit)	(delete)	
Successful	PC1	PC6	ICMP	■	0.000	N	4	(edit)	(delete)	
Successful	PC1	PC6	ICMP	■	0.000	N	5	(edit)	(delete)	

Checking PDU messages

Program 8:

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

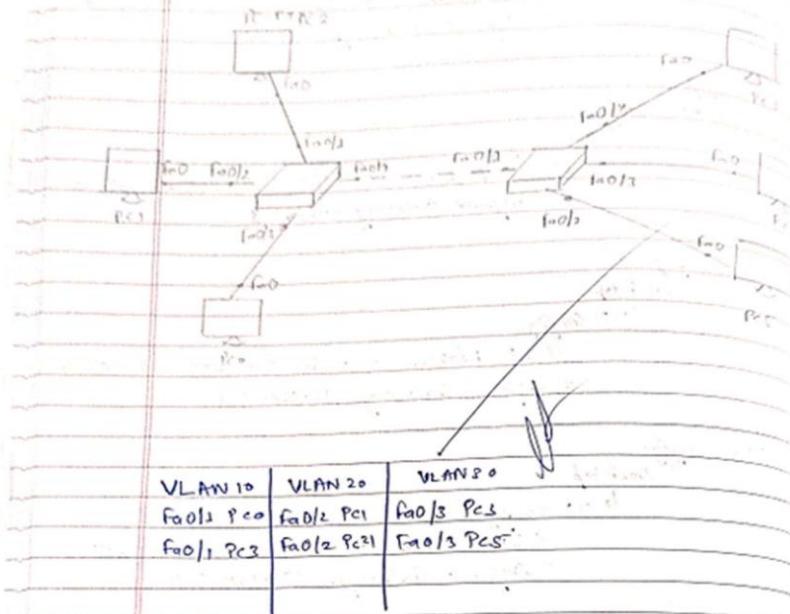
Network diagram:



Configuration:

Week - 4

a) Construct a VLAN to do make the PC communicate w/o VLAN



Switch configuration

↳ CLI

↳ Switch > enable

switch # configt

switch (config-if) int fa0/1.

switchport access VLAN10

int fa0/2

switchport access VLAN20

int fa0/3

switchport access VLAN30

int fa0/4

Switch 3 configuration

↳ CLI

switch > enable

switch # configt

switch configt int fa0/1

switchport access VLAN10

int fa0/2

switchport access VLAN20

int fa0/3

switchport access VLAN30

int fa0/4

switchport mode trunk

Output:

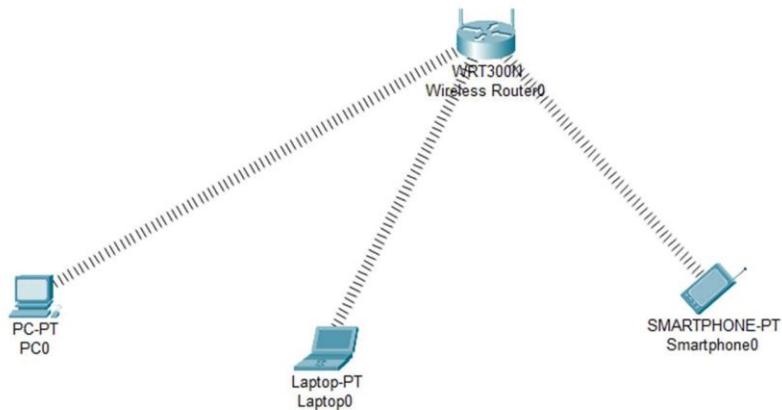
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
Successful	PC0	PC3	ICMP	Green	0.000	N	0	(edit)	(delete)	
Successful	PC1	PC4	ICMP	Magenta	0.000	N	1	(edit)	(delete)	
Failed	PC1	PC3	ICMP	Yellow	0.000	N	2	(edit)	(delete)	

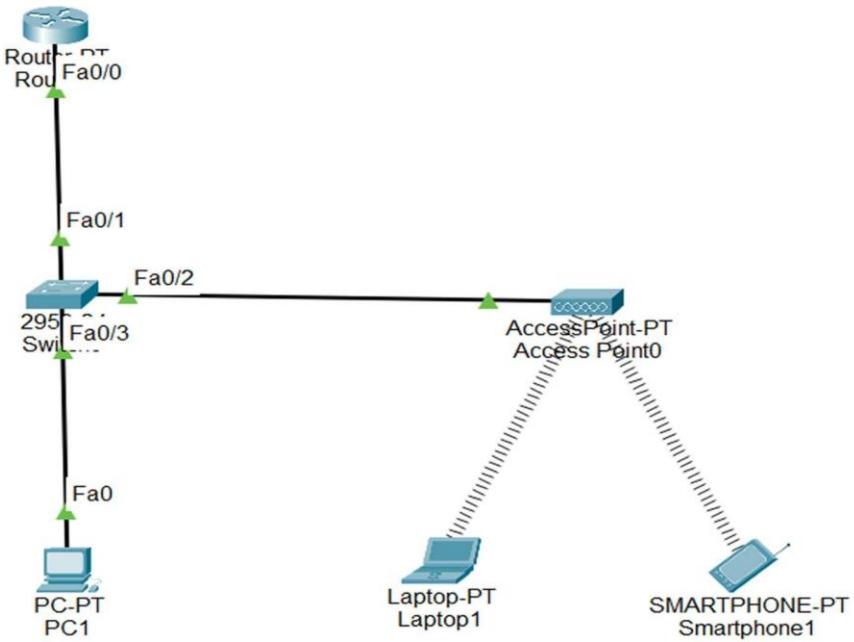
Checking PDU messages

Program 9:

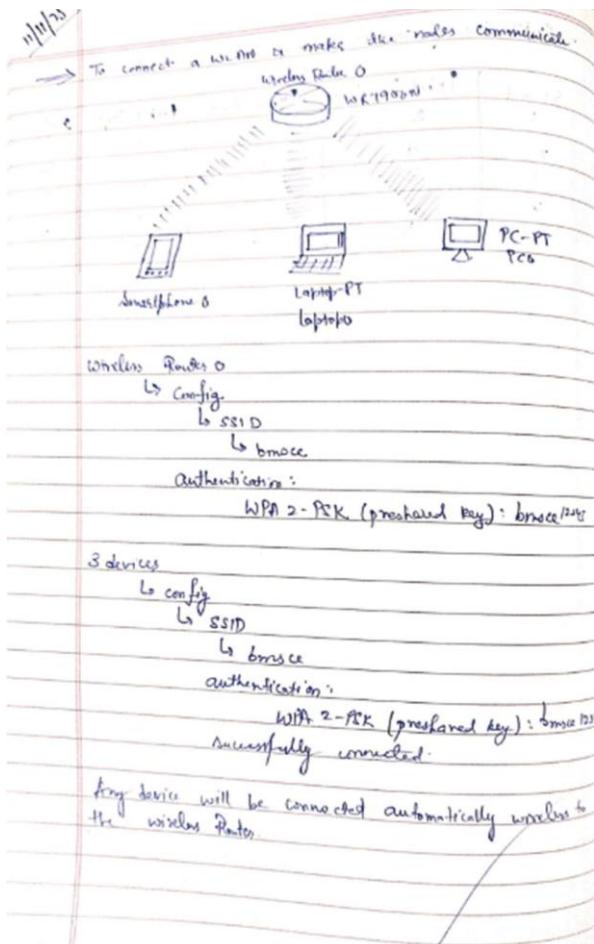
Aim: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:





Configuration:



Output:

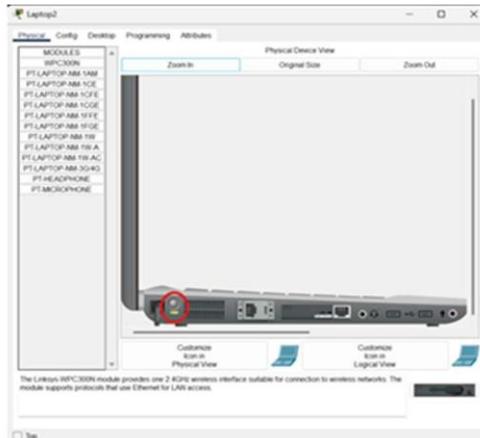


Fig 1.1 Step1: Turn off light / Power off laptop

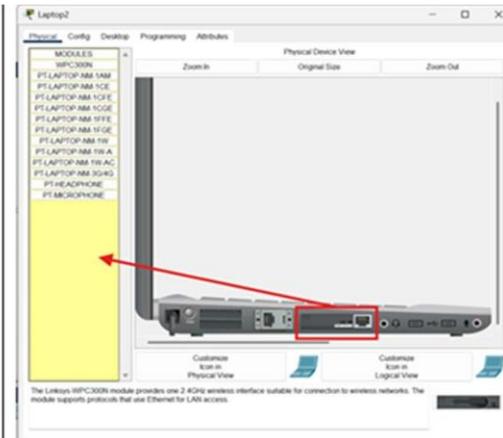


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

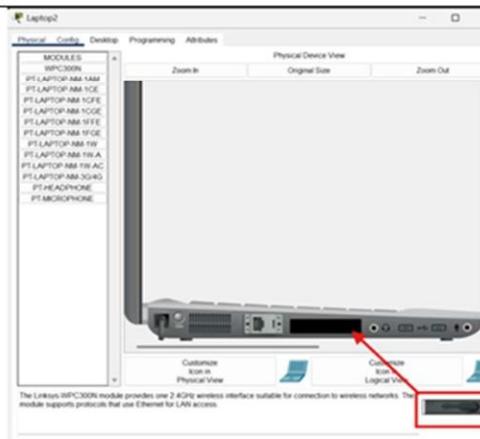


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

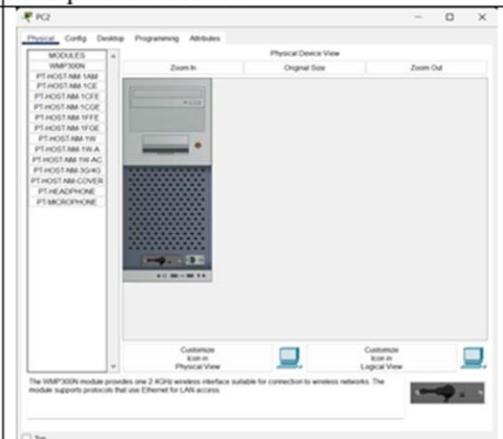


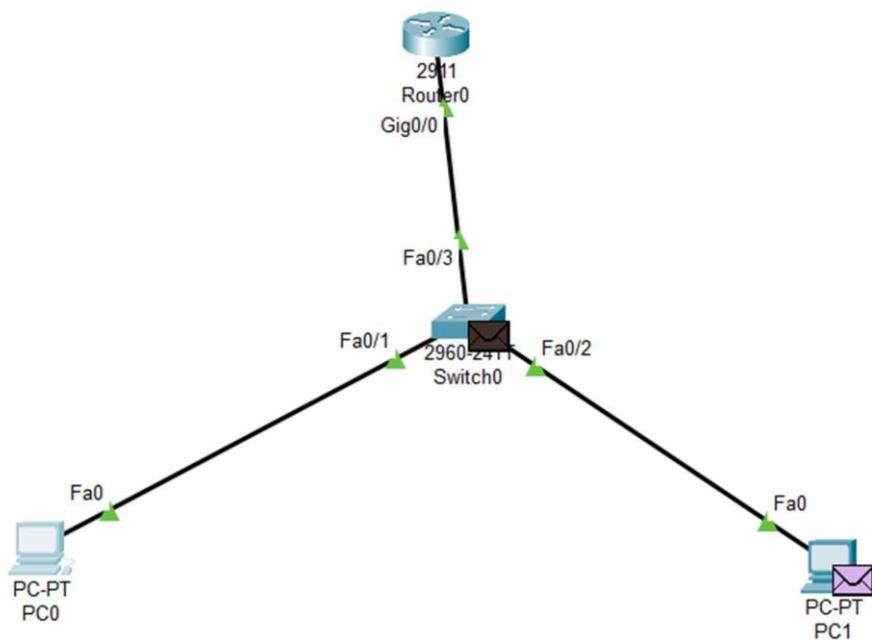
Fig 2. PC physical connection (combined 3 steps)

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
Successful	Wireless...	Smartphone...	ICMP	■	0.000		N	0	(edit)	(delete)
Successful	PC0	Wireless ...	ICMP	■	0.000		N	1	(edit)	(delete)
Successful	Laptop0	Smartpho...	ICMP	■	0.000		N	2	(edit)	(delete)

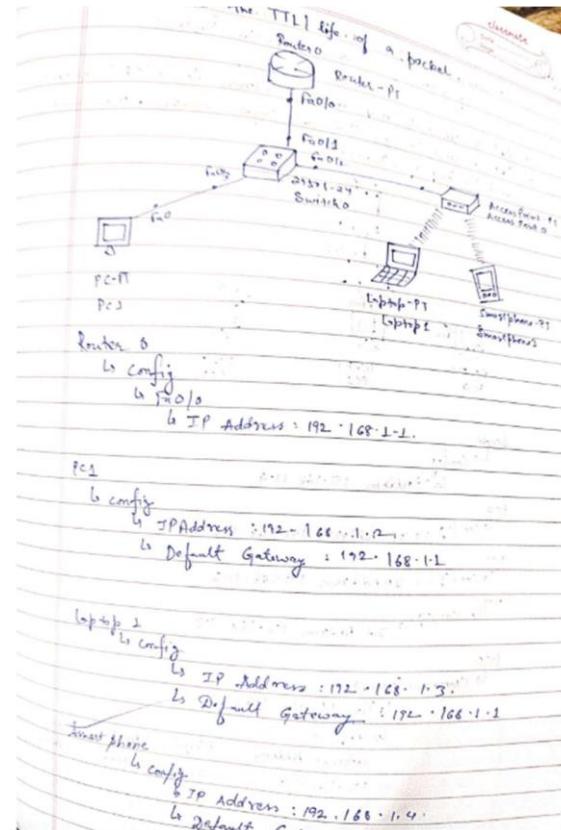
Program 10:

Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:



## Output:

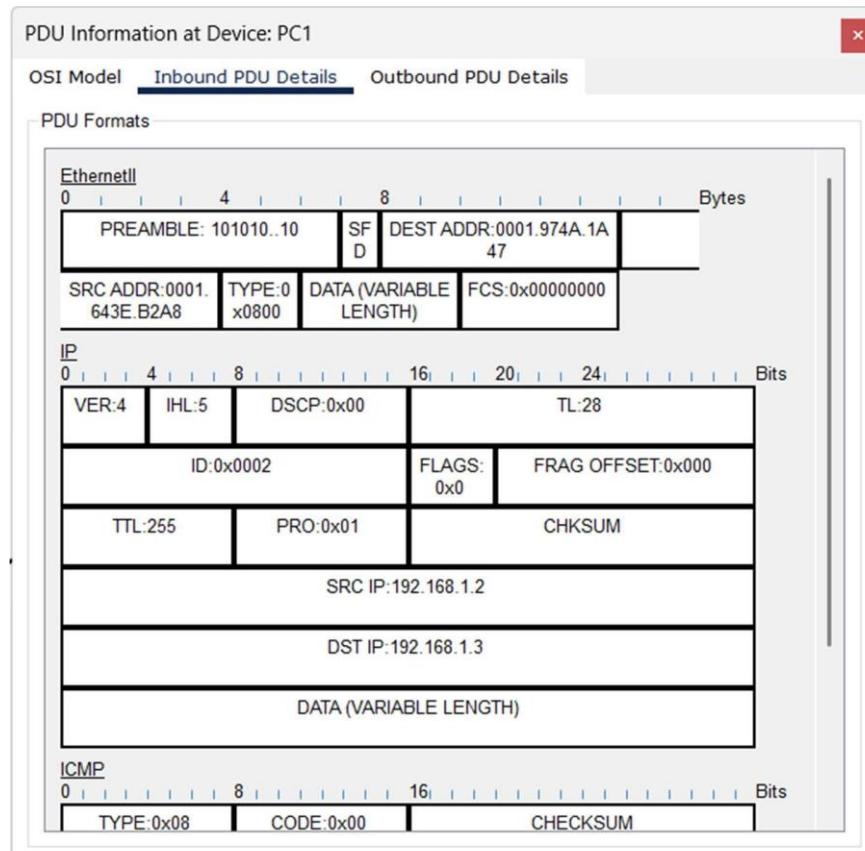


Fig 1. Inbound PDU Details at Device PC1

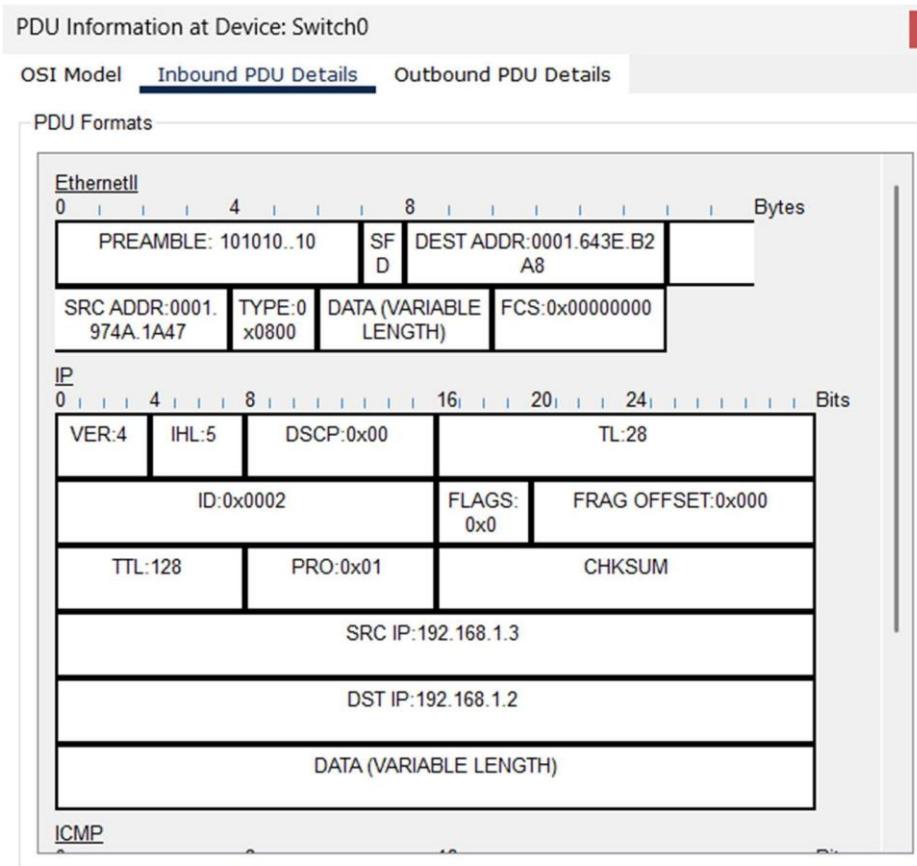


Fig 1. Inbound PDU Details at Device Switch0

#### Program 11:

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

Network diagram:

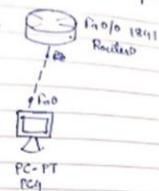


Configuration:

In introduce a topology to demonstrate the concept of Telnet.

It is used to access remote boxes. It is a simple command line tool that runs on computer. It will allow you to send command remotely to your server.

Telnet is also used to manage other devices like printer, switch to check if ports are open or close on a device.



In Router:

R1>en  
R1#conf t

Router (config) # hostname R1

# enable secret 5\$

# int F0/0

R1 (config)# ip address 192.168.1.1 255.255.255.0

# no shutdown

# line vty 0 5

# login

# password tp

# exit

In pc :

TC> Ping 192.168.1.1

>select 192.168.1.1

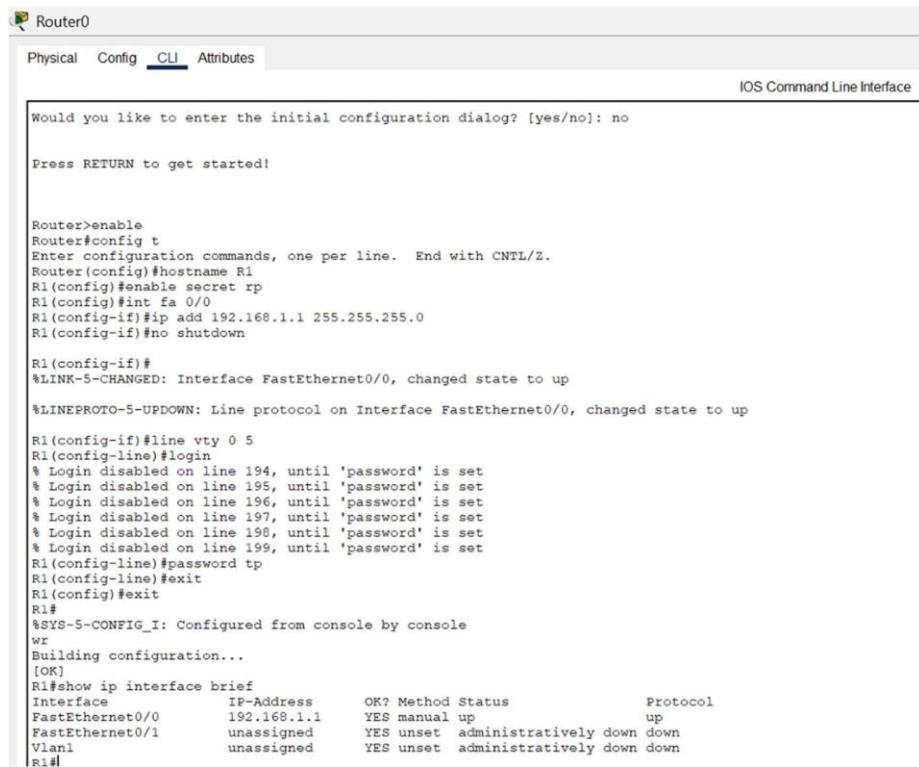
Password:

R1>en

R1# show ip interface brief

Interface	IP Address	OK?	Method	Status	Protocol
F0/0	192.168.1.1	Yes	manual	up	up
F0/0/3	unsigned	Yes	unset	administratively down	

Output:



The screenshot shows the Router0 CLI interface. The title bar says "Router0". Below it is a menu bar with "Physical", "Config", "CLI" (which is selected), and "Attributes". A sub-menu "IOS Command Line Interface" is open. The main window displays the following text:

```

Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!

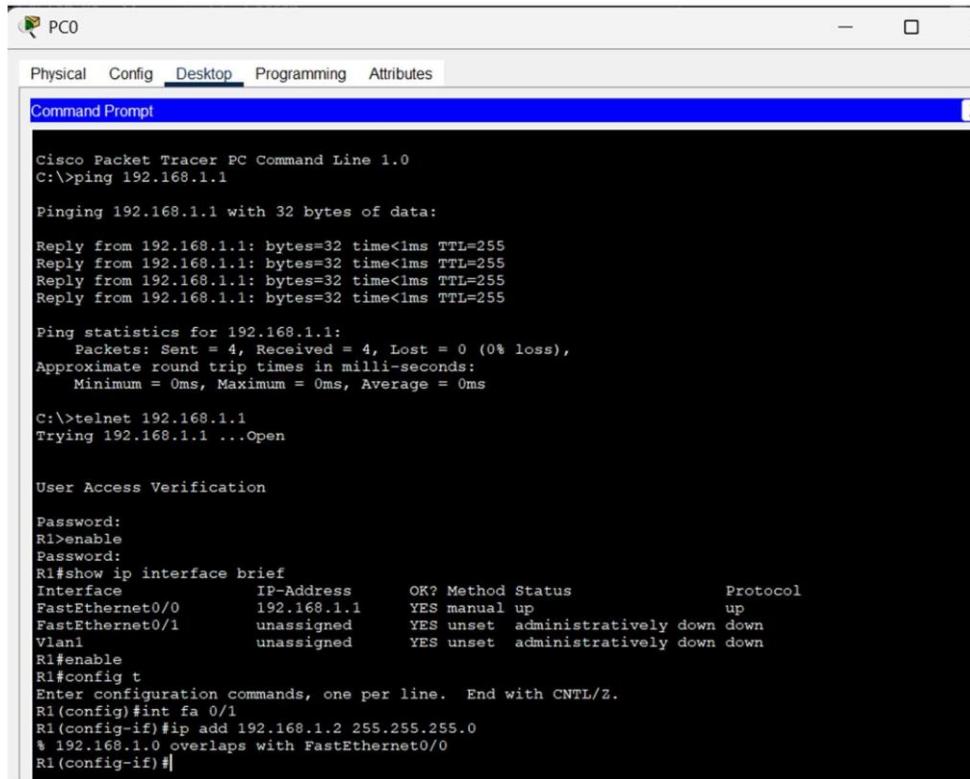
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname R1
R1(config)#enable secret rp
R1(config)#int fa 0/0
R1(config-if)#ip add 192.168.1.1 255.255.255.0
R1(config-if)#no shutdown

R1(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

R1(config-if)#line vty 0 5
R1(config-line)#login
% Login disabled on line 194, until 'password' is set
% Login disabled on line 195, until 'password' is set
% Login disabled on line 196, until 'password' is set
% Login disabled on line 197, until 'password' is set
% Login disabled on line 198, until 'password' is set
% Login disabled on line 199, until 'password' is set
R1(config-line)#password tp
R1(config-line)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
wr
Building configuration...
[OK]
R1#show ip interface brief
Interface          IP-Address      OK? Method Status        Protocol
FastEthernet0/0    192.168.1.1    YES manual up           up
FastEthernet0/1    unassigned      YES unset administratively down down
Vlan1             unassigned      YES unset administratively down down
R1#

```

Fig 1. Router0 – CLI commands



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The title bar says "PC0". Below it is a menu bar with "Physical", "Config", "Desktop" (which is selected), "Programming", and "Attributes". The main window displays the following text:

```

Cisco Packet Tracer PC Command Line 1.0
C:>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:>telnet 192.168.1.1
Trying 192.168.1.1 ...Open

User Access Verification

Password:
R1>enable
Password:
R1#show ip interface brief
Interface          IP-Address      OK? Method Status        Protocol
FastEthernet0/0    192.168.1.1    YES manual up           up
FastEthernet0/1    unassigned      YES unset administratively down down
Vlan1             unassigned      YES unset administratively down down
R1#enable
R1#config t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#int fa 0/1
R1(config-if)#ip add 192.168.1.2 255.255.255.0
% 192.168.1.0 overlaps with FastEthernet0/0
R1(config-if)#

```

Fig2. PC Command Line Prompt

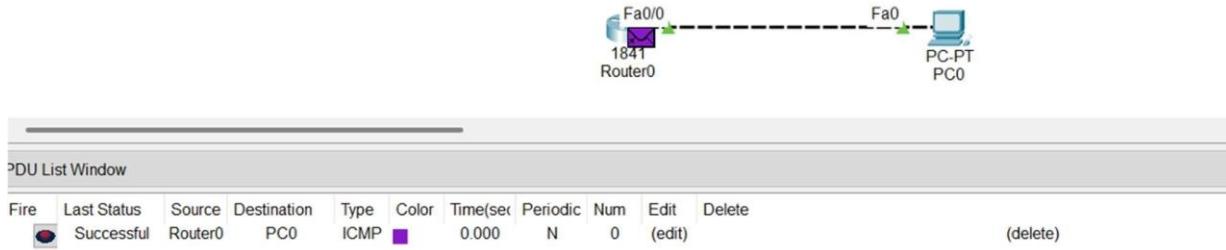
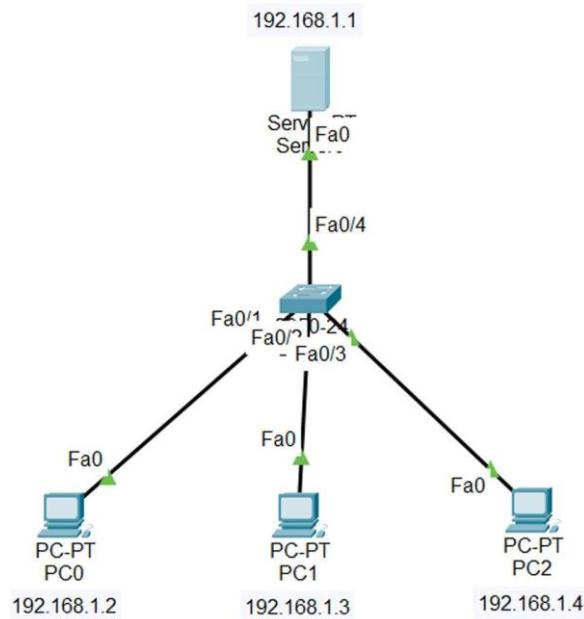


Fig3. PDU Message successful

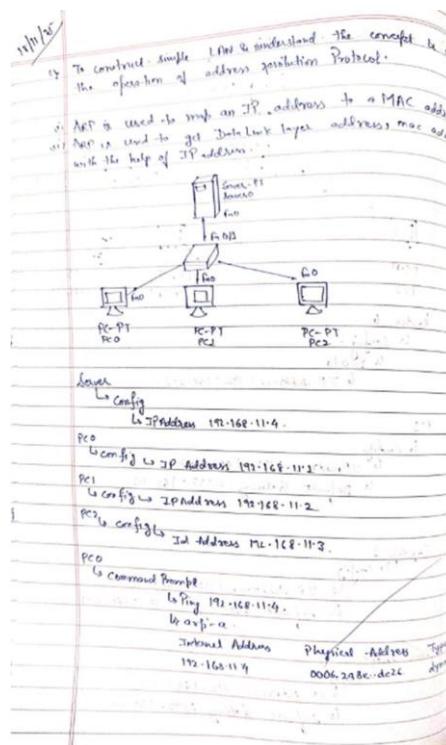
## Program 12:

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

Network diagram:



Configuration:



ARP Table for Server0		
IP Address	Hardware Address	Interface
192.168.1.2	00E0.F736.0126	FastEthernet0
192.168.1.3	0090.0C24.1CCC	FastEthernet0
192.168.1.4	00D0.D396.D2B5	FastEthernet0

Fig 1.1 ARP table at Server0

```
Cisco Packet Tracer SERVER Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.2            00e0.f736.0126      dynamic
192.168.1.3            0090.0c24.1ccc      dynamic
192.168.1.4            00d0.d396.d2b5      dynamic
c:\>
```

Fig 1.2 Command Prompt at Server0

ARP Table for PC0		
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 2.1 ARP table at PC0

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=3ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 8ms, Average = 5ms

C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.1            00e0.f7c6.ac93      dynamic
c:\>
```

Fig 2.2 Command Prompt at PC0

ARP Table for PC1		
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 3.1 ARP table at PC1

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=8ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.1            00e0.f7c6.ac93      dynamic
c:\>
```

Fig 3.2 Command Prompt at PC1

## PART – B Program

1:

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

PART - B

Program 1:

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

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

```
int min(int x, int y)
{    if (x < y)
    return x;    else
    return 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 entering at %d sec:\n", i + 1);
32
```

```
        scanf("%d", &inp[i]);
    }
```

```
    printf("\nSecond | Packet Received | 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("%d\t %d\t", i + 1, inp[i]);

    mini = min(count, process);
    printf("%d\t", mini);

    count = count - mini;
    printf("%d\t %d\n", count, drop);

    drop = 0;
}

// Remaining packets after time
ends    for (; count != 0; i++) {
if(count > cap) {
33

    drop = count - cap;
    count = cap;
}

printf("%d\t 0\t", i + 1);

mini = min(count, process);
printf("%d\t", mini);

count = count - mini;
printf("%d\t %d\n", count, drop);

drop = 0;
}

return 0;
}

```

OUTPUT:

```
Enter the bucket size:  
8  
Enter the processing rate:  
2  
Enter the number of seconds you want to simulate:  
5  
Enter the size of the packet entering at 1 sec:  
3  
Enter the size of the packet entering at 2 sec:  
1  
Enter the size of the packet entering at 3 sec:  
4  
Enter the size of the packet entering at 4 sec:  
3  
Enter the size of the packet entering at 5 sec:  
5  
  
Second | Packet Received | Packet Sent | Packet Left | Dropped  
-----  
1    3    2    1    0  
2    1    2    0    0  
3    4    2    2    0  
4    3    2    3    0  
5    5    2    6    0  
6    0    2    4    0  
7    0    2    2    0
```

Observation:

→ Congestion Control using token bucket algorithm

→ Code:

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

```
int main () {  
    if (x > y)  
        return x;  
    else  
        return y;  
}
```

```
int main () {  
    int drop = 0, min, max, cap, count = 0, i, m[5],  
    process;  
    printf ("Enter the buffer size : \n");  
    scanf ("%d", &cap);  
    printf ("Enter the processing rate : \n");  
    scanf ("%d", &process);  
    printf ("Enter the number of records you want to  
simulate : \n");  
    scanf ("%d", &max);  
    for (int i=0; i<max; i++){  
        printf ("Enter size of packet entering at %d sec  
i.e. : ");  
        scanf ("%d", &inp[i]);  
    }  
    printf ("In second 1Packet received 1Packet Sent /  
Packet left Unsentd \n");  
    printf (".....\n");  
    for (int i=0; i<max; i++) {  
        count += inp[i];  
    }
```

```

    count = info[1];
    count = cap;
}

printf("%d\n", info[1]);
printf("%d\n", min);
min = count - process;
info[1] = ("1t %d", min);
count = count - min;
printf("%d %d\n", count);
printf("%d\n", drop);
drop = 0;
}

for (count = 0; i <= 5) {
    if (count > cap) {
        drop = count - cap;
        count = cap;
    }
    printf("%d\n", info[1]);
    printf("%d\n", drop);
    min = min + process;
    printf("%d %d\n", min);
    count = count - min;
    printf("%d\n", count);
    printf("%d\n", drop);
}

```

Output:

Enter bucket size: 5  
 Enter processing rate: 2  
 Enter the number of second you want to simulate:  
 Enter size of packet entering at 3 sec: 5  
 Enter size of packet entering at 2 sec: 4

Enter size of packet entering at 3 sec: 3

Second	Packet Received	Packet Sent	Packet Left	Dropped
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

Drop 1  
 Drop 2  
 Drop 3

Drop 1  
 Drop 2  
 Drop 3

## Program 2:

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

Code:

# tcp_client.py	# tcp_server.py
import socket	import socket
# Step 1: Create TCP socket	# Step 1: Create a TCP socket
client_socket =	server_socket =
socket.socket(socket.AF_INET,	socket.socket(socket.AF_INET,
socket.SOCK_STREAM)	socket.SOCK_STREAM)
# Step 2: Connect to server	# Step 2: Bind to address and port
client_socket.connect(('localhost',	server_socket.bind('localhost',
8080))	8080))
# Step 3: Send filename	# Step 3: Listen for client
filename = input("Enter filename to	connections
request: ")	server_socket.listen(1)
	print("Server is listening on port
client_socket.send(filename.encode())	8080...")
# Step 4: Receive file contents	# Step 4: Accept connection
data =	conn, addr = server_socket.accept()
client_socket.recv(4096).decode()	print("Connected by:", addr)
print("\n--- File Content ---\n")	# Step 5: Receive file name
print(data)	filename =
# Step 5: Close connection	conn.recv(1024).decode().strip()
client_socket.close()	try:
	# Step 6: Open and read file
	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()

Observation:

Client - server communication using TCP or IP sockets  
to make client send the name of the file to server  
to send back the contents of the requested file if present

### Algorithm (Client side)

1. Sfd = Create a socket with the socket (...) system call.
2. Connect the socket to the address of the server using the connect (Sfd, ...) system call. The IP address of the server machine and port number of the server service need to be provided.
3. Read file name from standard input by:  
 $n = \text{read} (\text{fdin}, \text{buffer}, \text{sizef}(\text{buffer}))$
4. Write file name to the socket using write (Sfd, buffer, n).
5. Read file contents from the socket by  $m = \text{read} (\text{Sfd}, \text{buffer}, \text{sizef}(\text{buffer}))$
6. Display file contents to Standard output by write (fdin, 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. If not sure of machine IP address, keep the structure member S-addr to INADDRANY.
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  $n = \text{read} (\text{Sfd}, \text{buffer}, \text{sizef}(\text{buffer}))$
6. Open the file by fd = open (buffer).

7. Read the contents of the file by  $m = \text{read} (\text{fd}, \text{buffer}, \text{sizef}(\text{buffer}))$
8. Write the file content to socket by write (Sfd, buffer, m)
9. Go to step 7 if  $m > 0$
10. close (Sfd)

### Program 3:

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

Code:

<pre># udp_client.py</pre> <hr/> <pre>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.en code(), server_address)  # Step 3: Receive response data, addr = client_socket.recvfrom(4096)  print("\n--- File Content ---\n") print(data.decode())  # Step 4: Close socket client_socket.close()</pre>	<pre># udp_server.py</pre> <hr/> <pre>import socket  # Step 1: Create UDP socket server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Step 2: Bind to address and port 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().strip()      print(f"Requested file: {filename}")      try:         # Step 4: Open file and         send content         with open(filename, 'r') as f:             data = f.read()              server_socket.sendto(data. encode(), addr)      except FileNotFoundError:         server_socket.sendto(b"Fil e not found on server.", addr)</pre>
--	--

#### Program 4:

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

Code:

```

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

int main() {    char rem[50], a[50], s[50], c,
msj[50], gen[30];
    int i, genlen, t, j, flag = 0, k, n;

    printf("Enter the generation polynomial:\n");
    gets(gen);    printf("Generator polynomial is CRC-
CCITT: %s\n", gen);

    genlen = strlen(gen);
    k = genlen - 1;

    printf("Enter          the
message:\n");    n = 0;    while
((c = getchar()) != '\n') {
        msj[n] =
c;    n++;
}    msj[n] =
'\0';

    for (i = 0; i < n; i++)
        a[i] = msj[i];

    for (i = 0; i < k; i++)
a[n + i] = '0';
a[n + k] = '\0';

    printf("\nMessage polynomial appended with zeros:\n");
puts(a);

    for (i = 0; i < n; i++) {
if (a[i] == '1') {
    t = i;          for (j = 0;
j <= k; j++) {
            if
(a[t] == gen[j])
a[t] = '0';          else
a[t] = '1';          t++;
}
}
}

    for (i = 0; i < k; i++)
rem[i] = a[n + i];
rem[k] = '\0';

    printf("Checksum (remainder):\n");
puts(rem);

```

```

printf("\nMessage with checksum
appended:\n");    for (i = 0; i < n; i++)      a[i] =
msj[i];
=
for (i = 0; i < k; i++)
a[n + i] = rem[i];    a[n +
k] = '\0';
puts(a);

n = 0;
printf("Enter the received message:\n");
while ((c = getchar()) != '\n') {
    s[n] =
c;        n++;
}    s[n] =
'\0';

for (i = 0; i < n; i++) {
if (s[i] == '1') {
    t = i;            for (j = 0; j <=
k; j++, t++) {        if (s[t] ==
gen[j])            s[t] = '0';
else            s[t] = '1';
}
}
}

for (i = 0; i < k; i++)
rem[i] = s[n + i];
rem[k] = '\0';

for (i = 0; i < k; i++) {
if (rem[i] == '1')
flag = 1;
}

if (flag == 0)    printf("Received polynomial
is error-free \n");
else    printf("Received
polynomial contains error \n");

return 0;
}

```

OUTPUT:

```

Enter the generation polynomial:
101
Generator polynomial is CRC-CCITT: 101
Enter the message:
1101010101010100

Message polynomial appended with zeros:
1101010101010000
Checksum (remainder):
11

Message with checksum appended:
1101010101010011
Enter the received message:
1101010101010011
Received polynomial is error-free

Process returned 0 (0x0) execution time : 33.192 s
Press any key to continue.

```

### Observation:

CRC

- CRC is most powerful & easy to implement
- Based on binary division
- Calculate the remainder if it 0, then data is error free.

Data frame: 11001  
Polynomial Generator  $\bar{P} = 101$  ( $x^3 + 1$ ) ( $\oplus$  101)

i) Compute CRC at sender side

$$\begin{array}{r}
 11001 \\
 101 ) \underline{11001} \\
 \underline{101} \\
 \underline{0110} \\
 \underline{101} \\
 \underline{0011} \\
 \underline{101} \\
 \underline{0010} \quad \text{CRC} = 10
 \end{array}$$

ii) Frame + CRC

$$\begin{array}{r}
 11001 + 10 \\
 \hline
 110010
 \end{array}
 \quad (\text{append})$$

→ transmit this

→ Receiver Side:

$$\begin{array}{r}
 110010 \\
 101 ) \underline{110010} \\
 \underline{101} \\
 \underline{110} \\
 \underline{101} \\
 \underline{101} \\
 \underline{0}
 \end{array}$$

Frame: 100100  
Polynomial  $x^3 + x^2 + x + 1 \rightarrow 1101 = 2 \cdot 4 + 3 = 3$

$$\begin{array}{r}
 1101 \\
 101 ) \underline{10010000} \\
 \underline{101} \\
 \underline{01000} \\
 \underline{101} \\
 \underline{101} \\
 \underline{110} \\
 \underline{101} \\
 \underline{010} \\
 \underline{101} \\
 \underline{101} \\
 \underline{0}
 \end{array}$$

R

CRC + Frame = 1001001011

Receiver Side:

$$\begin{array}{r}
 11011 \\
 1101 ) \underline{1001001011} \\
 \underline{1101} \\
 \underline{1000} \\
 \underline{1101} \\
 \underline{1010} \\
 \underline{1101} \\
 \underline{1101} \\
 \underline{1011} \\
 \underline{1011} \\
 \underline{0}
 \end{array}$$

→ No error

$\Rightarrow$  Encoding procedure at Sender.

Message is represented as information polynomial  $i(x)$  of length  $k$ . The polynomial generator is represented as  $g(x)$  of length  $m$ .

1. Multiply  $i(x)$  by  $x^{(n-k)}$  by putting zeros in  $n-k$  low order positions.
  2. Divide  $x^{(n-k)} i(x)$  by  $g(x)$  to get  $r(x)$  using Euclidean division algorithm with  $n-k$  bits left in register.
  3. Add remainder  $r(x)$  to  $x^{(n-k)} i(x)$  by putting zeros in  $n-k$  lower order positions.
  4. Based on circumstances, the message can be transmitted with error or without error.
  5. For transmission with error, introduced an error at random position to the message  $x^{(n-k)} i(x)$  to display the position of the error.
  6. Transmitted message is  $b(x) = x^{(n-k)} i(x) + r(x)$
- $\Rightarrow$  Reducing operations at the Receiver.
1. The received message  $b(x)$  is divided by  $g(x)$  using Euclidean division algorithm.
  2. If the remainder is 0, then there is no error in the transmission, else there is error in transmission.

OUTPUT:

1. Enter the generator polynomial : 101

Enter the message : 110101

Message appended with zeros : 11010100

Check sum (Remainder) : 11

Transmitted Message : 11010111

Enter the received message : 11010111

Received polynomial is error-free.

2. generator polynomial : 101

Message : 110101

Message appended with zeros : 11010100

check sum : 11

Transmitted Message : 11010111

Enter Received Message : 11101111

Received polynomial contains errors.

3. generator polynomial : 101

Message : 11010101010100

Message appended with zeros : 1101010101010000

check sum = 11

Transmitted Message : 1101010101010011

Enter received message : 1101010101010011

Received polynomial is error free.