

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



## **LAB RECORD**

### **Computer Network Lab (23CS5PCCON)**

*Submitted by*

**PRADEEP G (1BM24CS414)**

*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 **Pradeep G (1BM24CS414)**, 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.

Sarala D V Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
---------------------------------------------------------------	------------------------------------------------------------------

# **Index**

## **Part - A**

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

## Part - B

Sl. No.	Date	Experiment Title	Page No.
1	28/10/25	Write a program for congestion control using Leaky bucket algorithm.	31 – 34
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.	35 – 36
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.	37 – 38
4	28/10/25	Write a program for error detecting code using CRC-CCITT (16-bits).	39 - 43

Github Link:

[https://github.com/Pradeepg06/Computer-Networks/tree/CN\\_lab](https://github.com/Pradeepg06/Computer-Networks/tree/CN_lab)

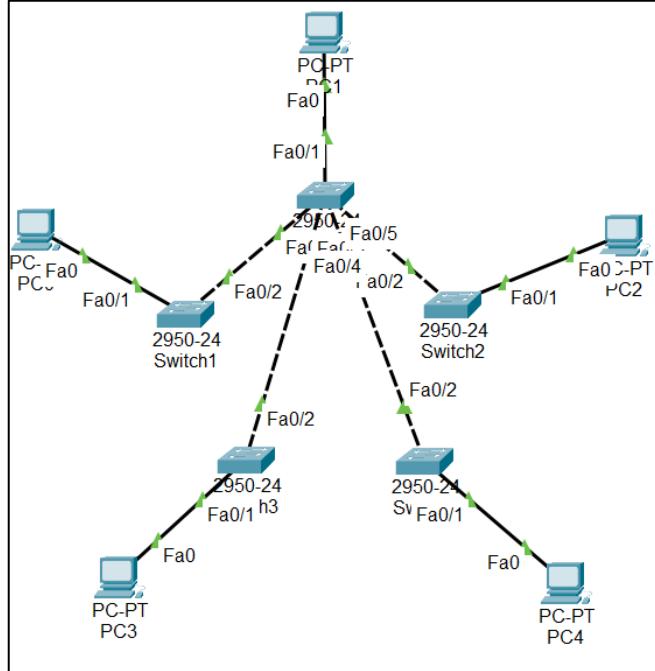
## PART - A

### Program 1:

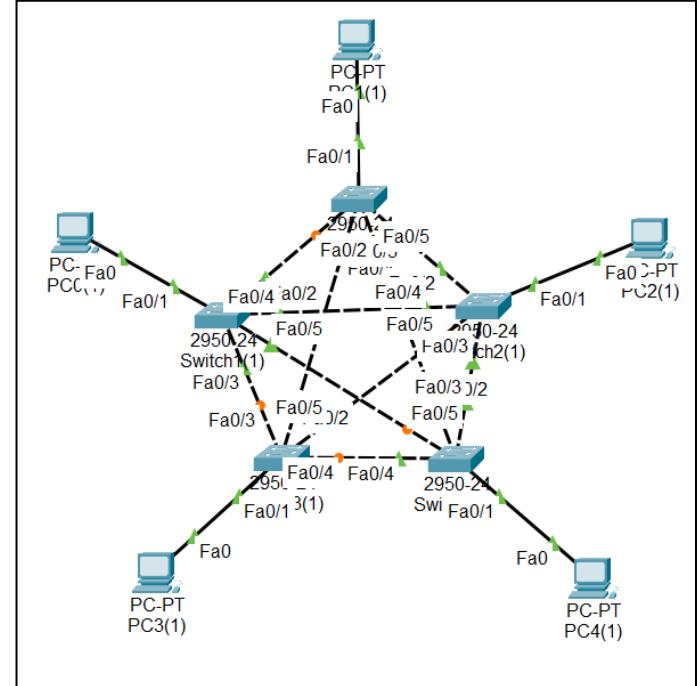
**Aim:** Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

### Network diagram:

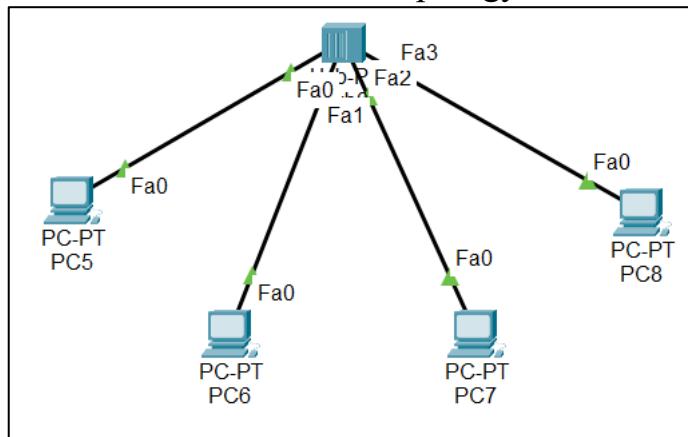
#### 1. STAR Topology with Switch:



#### 2. MESH Topology with Switch:



#### 3. HUB-Based Network Topology:



# Configuration:

**Procedure:**  
 1. Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message using CISCO packet tracer.

**Star Topology using Switch:**

**Name:** Implementation of Star Topology using Cisco Packet Tracer.

**Procedure:**

1. Open a new project in Packet Tracer.
2. Drag and drop:
  - 1 Central Switch (Switch 0)
  - 4 Edge switches (Switch 1, Switch 2, Switch 3, Switch 4, Switch 5)
  - 5 PCs (PC4, PC5, PC6, PC7, PC8)

3. Connect each PC to its respective switch using Copper Straight-through cables.
4. Connect all edge switches to the central switch.

5. Assign IP addresses: [Click on PC icon/device  $\rightarrow$  Desktop  $\rightarrow$  IP configuration  $\rightarrow$  enter IP address]

- PC4  $\rightarrow$  192.168.1.2
- PC5  $\rightarrow$  192.168.1.3
- PC6  $\rightarrow$  192.168.1.4
- PC7  $\rightarrow$  192.168.1.5
- PC8  $\rightarrow$  192.168.1.6

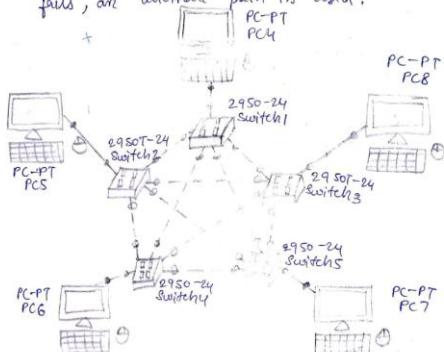
Subnet mask: 255.255.255.0 is directly entered

**Simulation:**

1. Switch to Simulation Mode (bottom-right corner).
2. Click the Add Simple PDU (envelope icon) tool.
3. Select source PC  $\rightarrow$  then destination PC.
4. In the Event list, a packet creation entry will appear.

**Simulation:**

1. Switch to Simulation Mode.
2. Click the Add Simple PDU tool.
3. Choose source PC  $\rightarrow$  then destination PC.
4. In the Event list, packet transmission will be logged.
5. Use Auto capture/Play or Capture/Forward to observe the packet flow.
6. In mesh topology, packets have multiple possible paths between switches; if one link fails, an alternate path is used.



**③ Hub-Based Network Topology:**

**Name:** Implementation of hub-Based Network Topology in Cisco Packet Tracer.

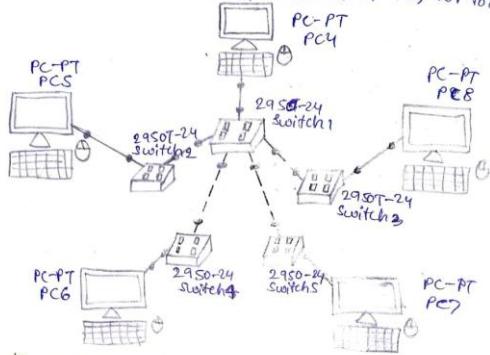
**Procedure:**

1. Open a new project in Packet Tracer.

2. Drag and drop:
  - 1 Hub (Hub1)
  - 4 PCs (PC9, PC10, PC11, PC12)

3. Click Auto capture/Play or Capture/Forward to watch packet flow.

4. With Switch, the packet travels only to the intended destination PC, not to all



**② MESH Topology using Switches:**

**Name:** Implementation of Mesh Topology using Switches in Cisco packet tracer.

**Procedure:**

1. Open a new project in Packet Tracer.

2. Drag and Drop:
  - 5 switches (Switch 1, Switch 2, Switch 3, Switch 4, Switch 5)
  - 5 PC's (PC4, PC5, PC6, PC7, PC8)

3. Connect each PC to a switch.
4. Interconnect all switches (mesh connection).

5. Assign IP addresses:

- PC4  $\rightarrow$  192.168.1.2
- PC5  $\rightarrow$  192.168.1.3
- PC6  $\rightarrow$  192.168.1.4
- PC7  $\rightarrow$  192.168.1.5
- PC8  $\rightarrow$  192.168.1.6

6. Connect all PCs to the hub using copper straight-through cables.

7. Assign IP addresses:

- PC9  $\rightarrow$  192.168.1.2
- PC10  $\rightarrow$  192.168.1.3
- PC11  $\rightarrow$  192.168.1.4
- PC12  $\rightarrow$  192.168.1.5

(Subnet Mask: 255.255.255.0)

**Simulation:**

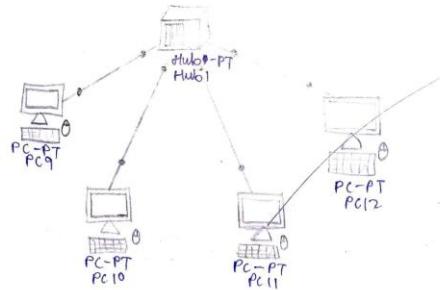
1. Switch to Simulation Mode.
2. Click the Add Simple PDU tool.

3. Choose Source PC  $\rightarrow$  then Destination PC.

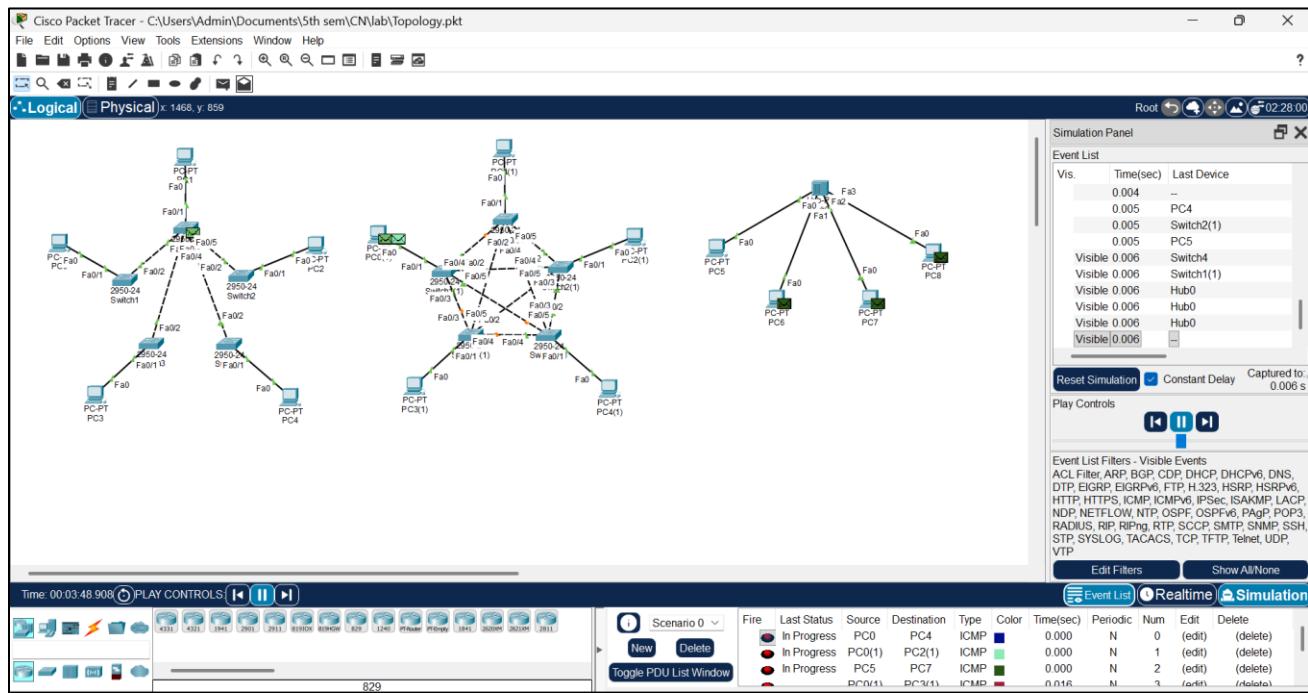
4. Check the Event list for packet creation.

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.



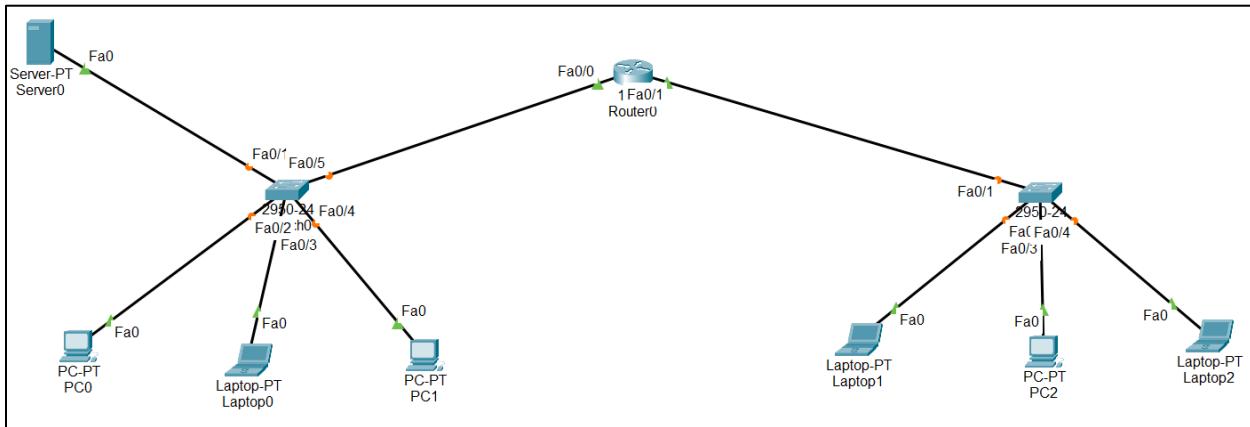
## Output:



## Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

7/9/25 program 2  
\* Configure DHCP within a LAN & outside a LAN

Procedure:

- ① click on Server
  - ② Go to desktop → IP configuration → select static  
Ex give (192.168.10.2) → Ex give gateway  
(192.168.10.1).
  - ③ Go to Services → DHCP → desktop IP config →  
Static IP → 192.168.10.2  
Gateway → 192.168.10.1
  - ④ Services → DHCP → Pool Name : switchOne  
Gateway : 192.168.20.1  
Start IP addl : 192.168.10.3  
Subnet Mask : 255.255.255.0
  - pool name: switch Two  
Gateway : 192.168.20.1  
Start IP addl : 192.168.20.2  
Subnet Mask : 255.255.255.0
- (Add) ↳

⑤ Router ↳ CLI ↳ ↳ enter.

```

    ↳ mo ↳ ↳
    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
    exit
    do write memory.
  
```

## Output:

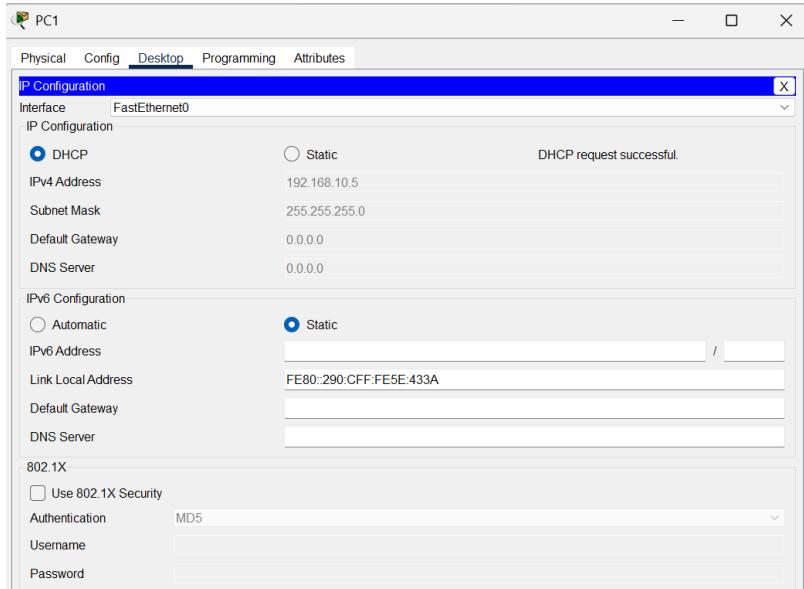


Fig 1. Ip address assigned by DHCP server within Lan (PC1)

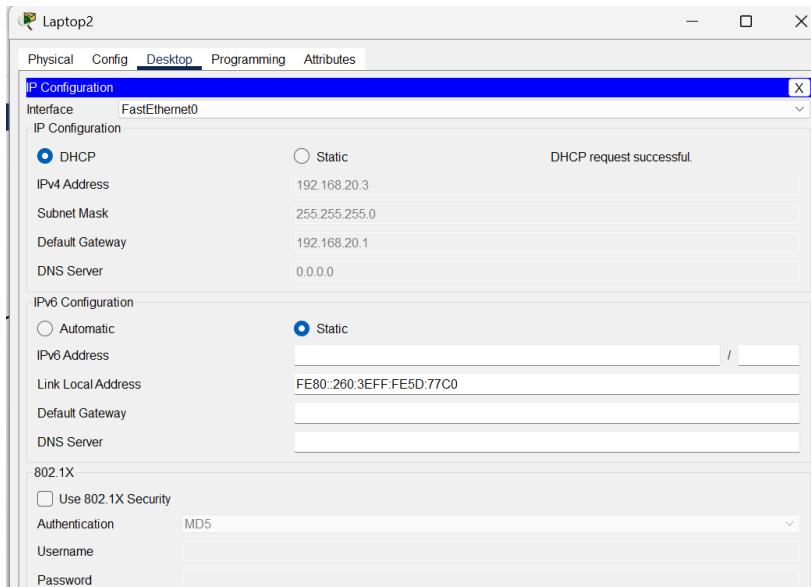
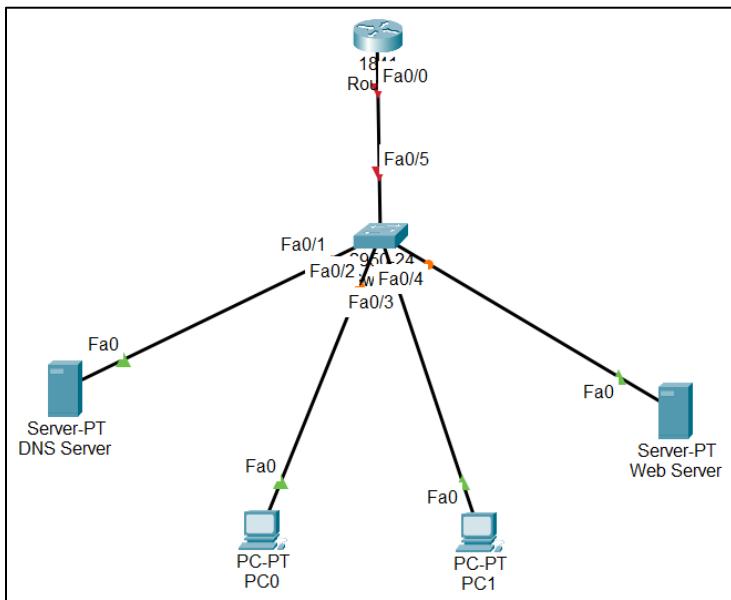


Fig 2. Ip address assigned by DHCP server outside Lan (laptop2)

### Program 3:

Aim: Configure Web Server, DNS within a LAN.

#### Network diagram:



#### Configuration:

Programm: Configure web server, DNS within a LAN

Procedure:

① DNS Setup

↳ Desktop → IP Configuration

↳ IP Address : 192.168.1.2  
Subnet Mask : 255.255.255.0  
Gateway : 192.168.1.1  
DNS Server : 192.168.1.2

② PC0

↳ IP Configuration

↳ IP Address : 192.168.1.100  
Subnet Mask : 255.255.255.0  
Gateway : 192.168.1.1  
DNS Server : 192.168.1.2

PC1

↳ IP Configuration

↳ IP Address : 192.168.1.101  
Subnet Mask : 255.255.255.0  
Gateway : 192.168.1.1  
DNS Server : 192.168.1.2

③ webserver

↳ IP Configuration

↳ IP Address : 192.168.1.6  
Subnet Mask : 255.255.255.0  
Gateway : 192.168.1.1  
DNS Server : 192.168.1.2

↳ Services

↳ HTTP

↳ HTTP [ON] HTTPS [OFF]

↳ Newfile → save

④ DNS Server

↳ Services

↳ DNS

↳ [ADD]  
Name: www.letslearn.com Type:A Record  
Address: 192.168.1.6  
click on [Add]

⑤ DNS Server

↳ Desktop

↳ web Browser

↳ URL: www.letslearn.com

↳ click on [Go]

## Output:

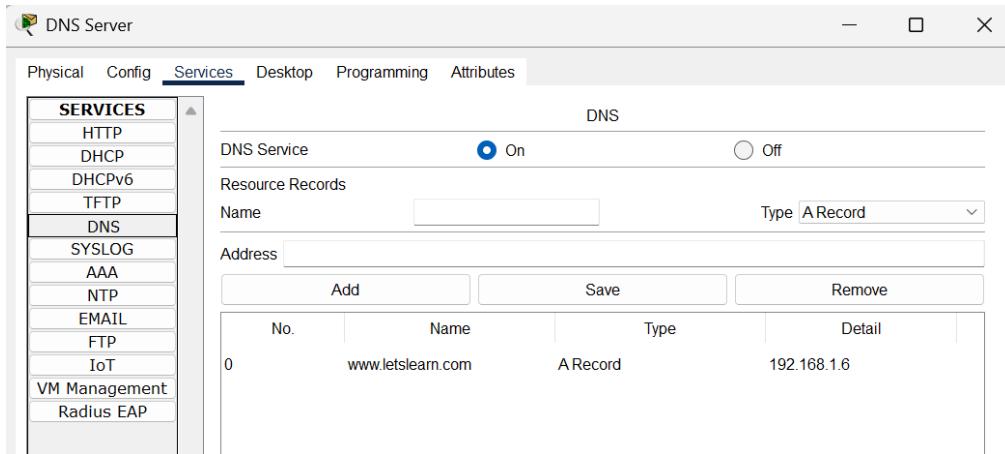


Fig 1. DNS server – DNS Services

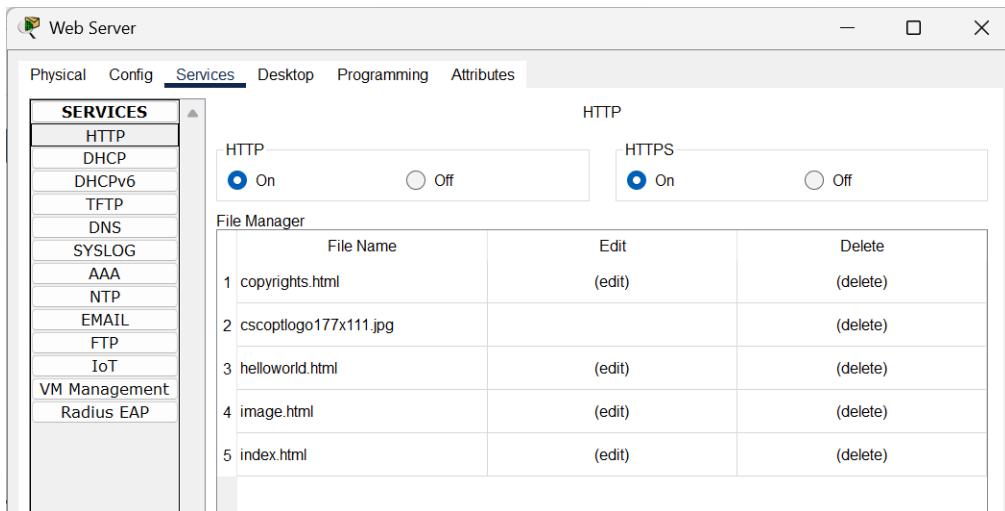


Fig 2. WEB server – HTTP Services

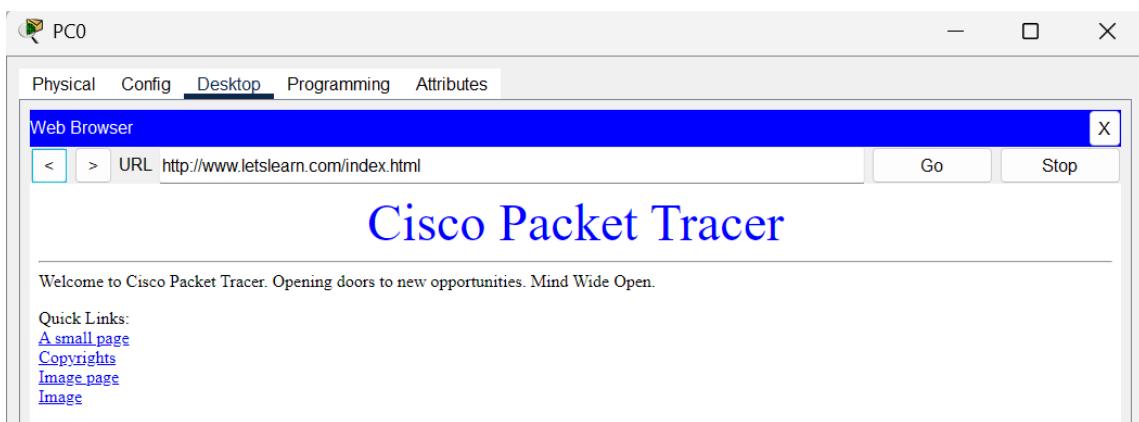
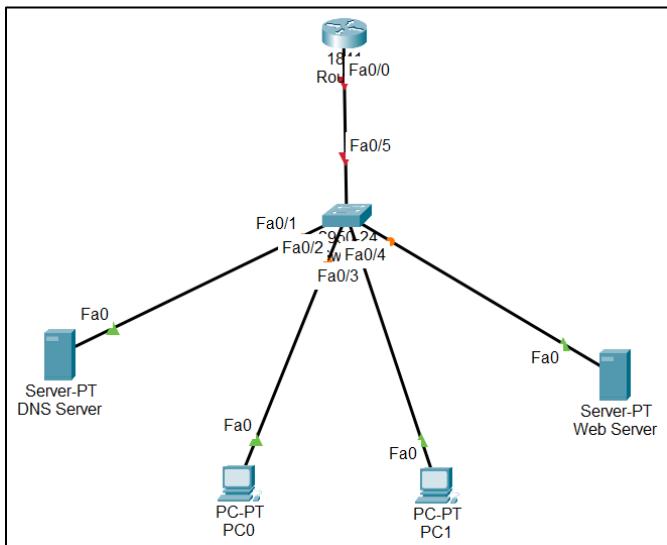


Fig 3. PC0 – accessing data from web browser

## 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:

Program 4: Configure IP address to the router in packet tracer explore the following messages

1. Ping response
2. Destination unreachable
3. Request timeout
4. Reply

#### Network connection (Diagram):

Same as previous experiment (dns -> web server)

#### Procedure

1. Assign IP address as follows

Device	Interface	IP Address	Subnet Mask	Gateway
Router (Fa0/0)	—	192.168.1.1	255.255.255.0	—
PC0	Fa0	192.168.1.10	255.255.255.0	192.168.1.1
PC1	Fa0	192.168.1.20	255.255.255.0	192.168.1.1
DNS Server	Fa0	192.168.1.100	255.255.255.0	192.168.1.1
Web Server	Fa0	192.168.1.200	255.255.255.0	192.168.1.1

#### 2. Configure Router interface

```

Router# enable
Router# configure terminal
Router(config)# interface fa0/0
Router(config-if)# ip address 192.168.1.1 255.255.255.0
Router(config-if)# no shutdown
  
```

#### 3. Save Configuration:

```
Router# write
```

#### 4. Configure IP & Default Gateway in PCs/Servers

(Laptop → IP configuration)

#### 5. Test connectivity using ping command from PCs.

Change conditions (wrong IP, wrong gateway, shut interface, power off device) to observe diff. ping messages.

#### Observations:

##### Call 1: Ping Response

Ping Command: ping 192.168.1.10  
(PC0 → PC1 in same network)

Message observed: Ping Response

Reason: ICMP Echo Request and Echo reply exchanged successfully b/w two active devices.

##### Call 2: Reply

Ping Command: ping 192.168.1.100  
(PC0 → DNS Server)

Message observed: Reply from 192.168.1.100

Reason: Destination device is active, reachable & properly configured.

##### Call 3: Destination Unreachable

Ping Command: ping 192.168.1.200  
(towards webserver)

Message observed: Destination Host unreachable

Reason: Router cannot be reached due to missing/incorrect gateway, so no route exists.

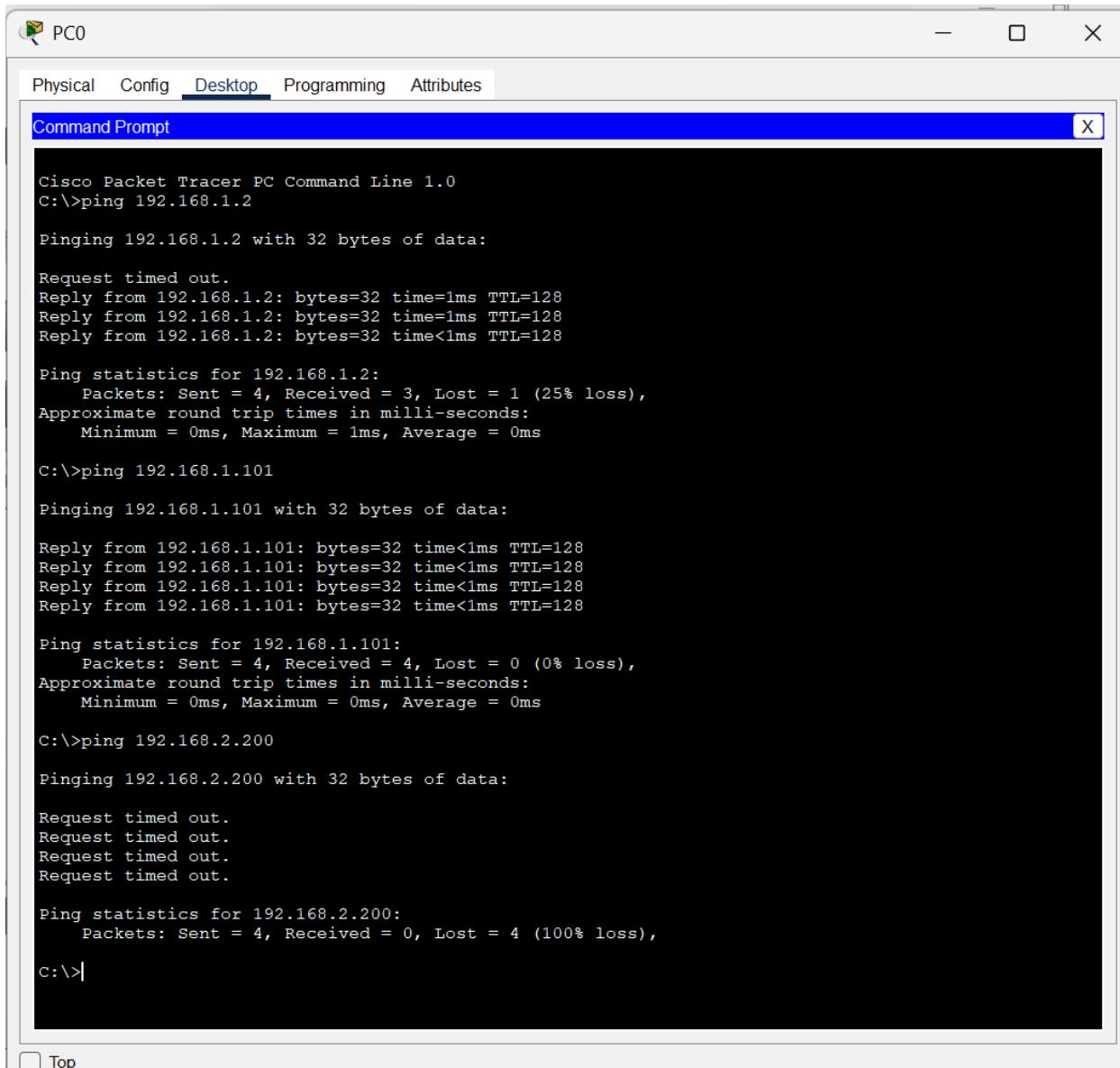
##### Call 4: Request Timedout

Ping Command: ping 192.168.1.1500  
(non-existent device)

Message observed: Request timed out

Reason: No reply received since IP does not exist/ device is off.

## Output:



The screenshot shows a window titled "Command Prompt" from "Cisco Packet Tracer PC Command Line 1.0". The window contains the following command-line output:

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

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.1.2: bytes=32 time=1ms TTL=128
Reply from 192.168.1.2: bytes=32 time=1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 192.168.1.101

Pinging 192.168.1.101 with 32 bytes of data:

Reply from 192.168.1.101: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.2.200

Pinging 192.168.2.200 with 32 bytes of data:

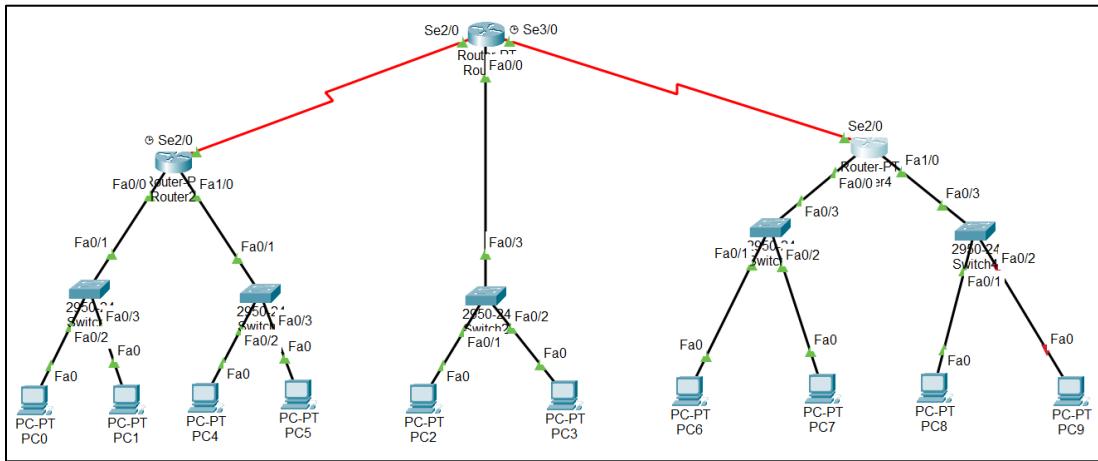
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.2.200:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>
```

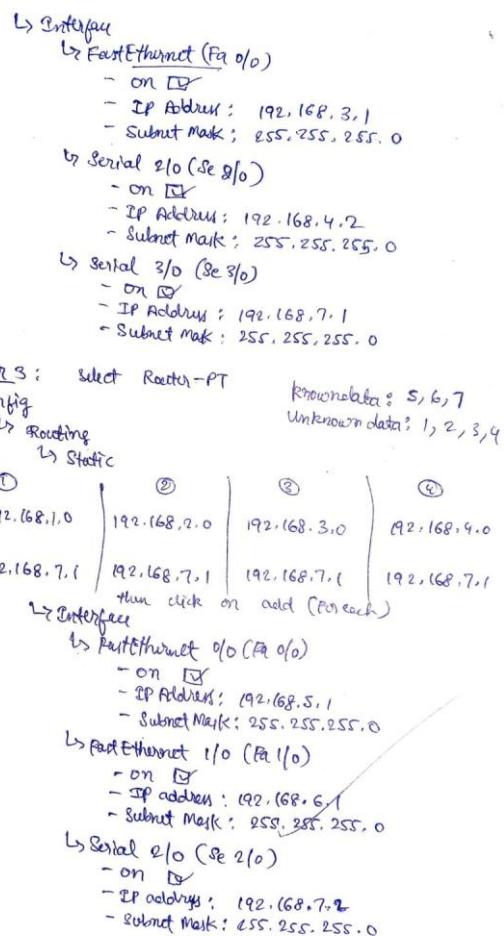
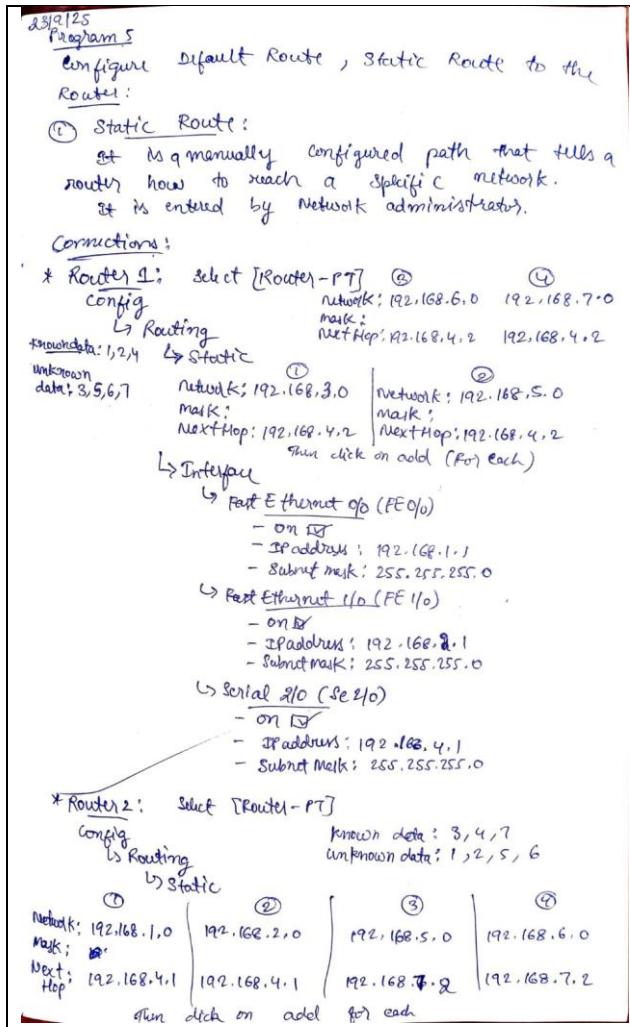
## Program 5:

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

Network diagram:



Configuration:



<ul style="list-style-type: none"> <li>* PC 0 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.1.2</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway : 192.168.1.1</li> </ul> </li> </ul> </li>   <li>* PC 1 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.1.3</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway : 192.168.1.1</li> </ul> </li> </ul> </li>   <li>* PC2 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.2.2</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway : 192.168.2.1</li> </ul> </li> </ul> </li>   <li>* PC3 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.2.3</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway : 192.168.2.1</li> </ul> </li> </ul> </li>   <li>* PC4 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.3.2</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway : 192.168.3.1</li> </ul> </li> </ul> </li>   <li>* PC5 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.3.3</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default gateway : 192.168.3.1</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>* PCG (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>IP configuration               <ul style="list-style-type: none"> <li>↳ IP address: 192.168.5.2</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway: 192.168.5.1</li> </ul> </li> </ul> </li>   <li>* PC7 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP configuration               <ul style="list-style-type: none"> <li>↳ IP address: 192.168.5.3</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway: 192.168.5.1</li> </ul> </li> </ul> </li>   <li>* PC8 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.6.2</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default Gateway: 192.168.6.1</li> </ul> </li> </ul> </li>   <li>* PC9 (Select PC-PT)           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>↳ IP address : 192.168.6.3</li> <li>Subnet Mask : 255.255.255.0</li> <li>Default gateway: 192.168.6.1</li> </ul> </li> </ul> </li> </ul>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Output:

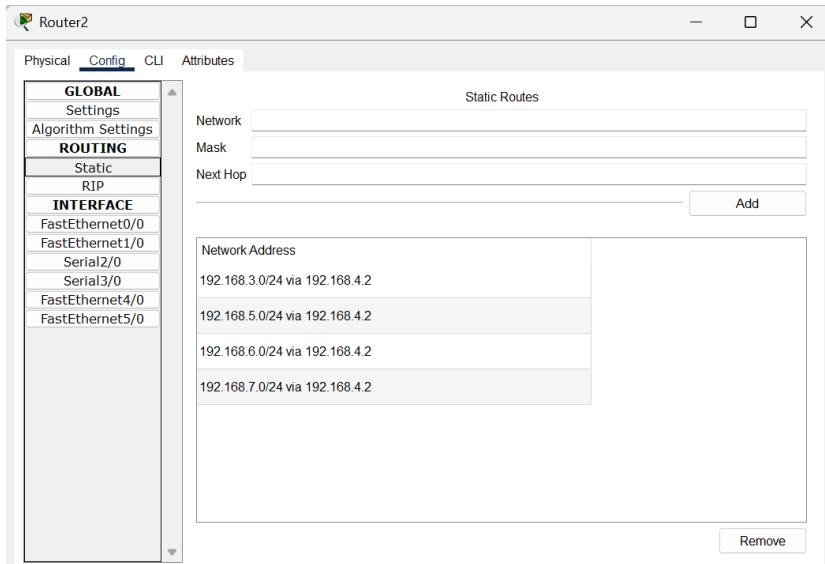


Fig 1. Router 2 – Static routing

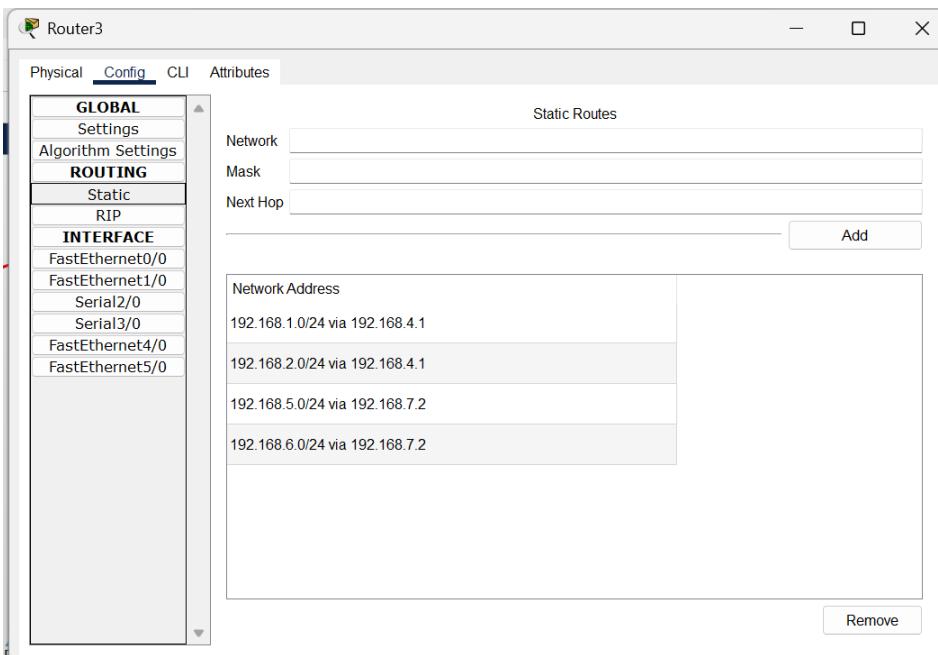


Fig 2. Router 3 – Static routing

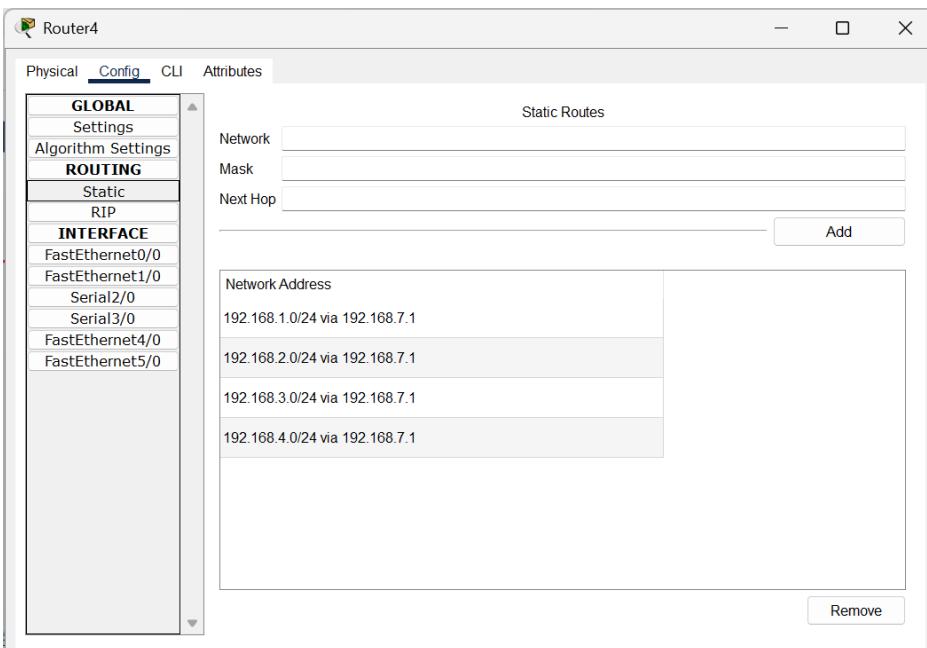
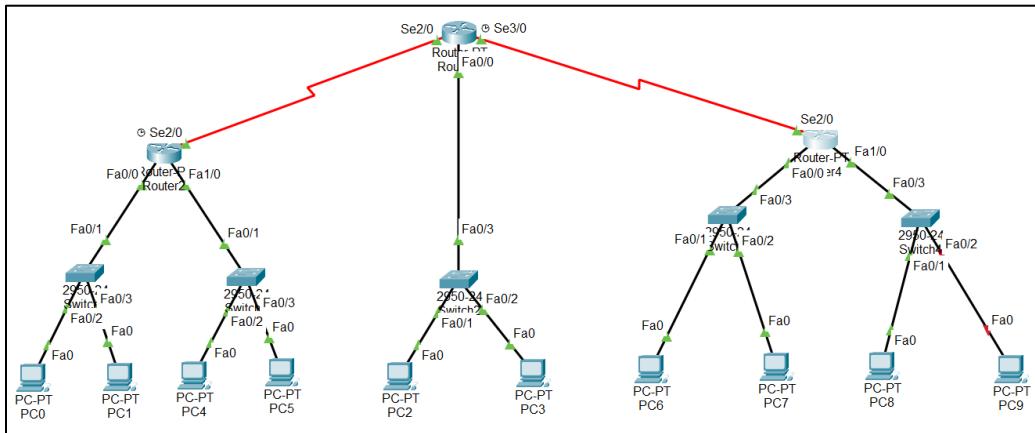


Fig 3. Router 4 – Static routing

## Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:

### ② Dynamic Routing

Dynamic Routing is a networking technique where routers automatically and adaptively share routing information using protocols to find the best path for data to travel across a network.

#### Connections:

Same as static Routing, but we have to remove all static Routes [under Routing] from all routers & assign the Dynamic Routing, i.e.,

\* Router 1: (Select Router-PT)

↳ Config

↳ Routing

↳ RIP Routing

↳ Network(s):  
192.168.1.0  
192.168.2.0  
192.168.4.0

then click on add [For each]

\* Router 2:

↳ Config

↳ Routing

↳ RIP Routing

↳ Network:  
192.168.3.0  
192.168.4.0  
192.168.7.0

then click on add [For each]

\* Router 3:

↳ Config

↳ Routing

↳ RIP Routing

↳ Network:  
192.168.5.0  
192.168.6.0  
192.168.7.0

then click on add [For each]

## Output:

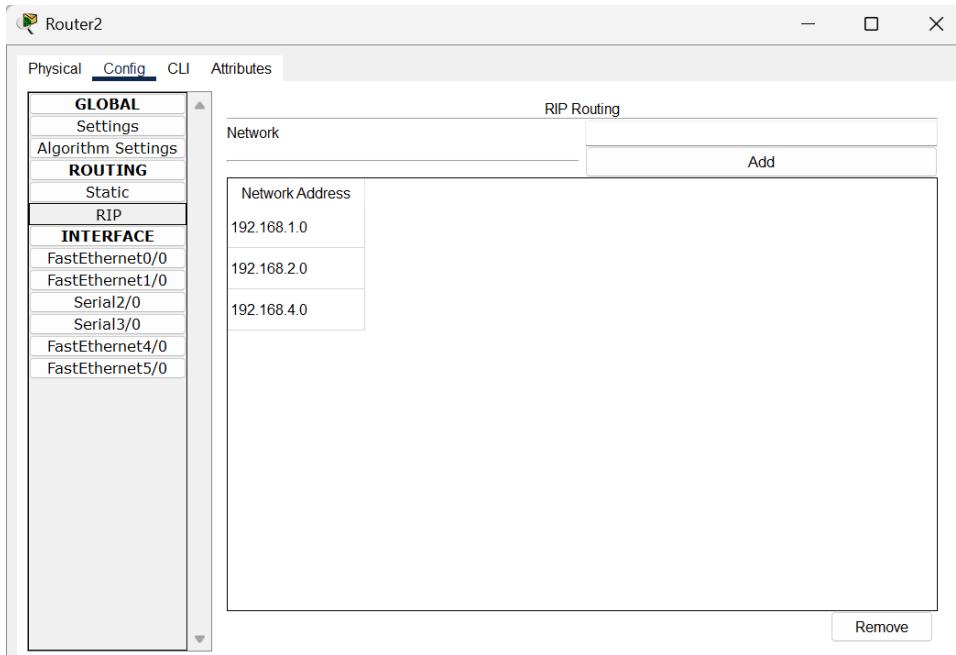


Fig 1. Router 2 – RIP routing

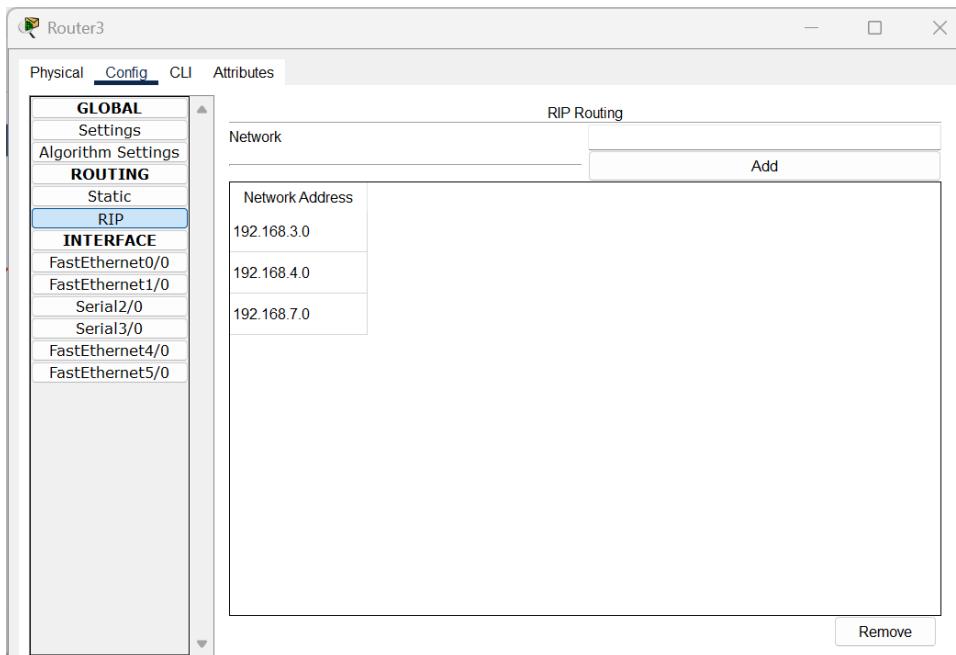


Fig 2. Router 3 – RIP routing

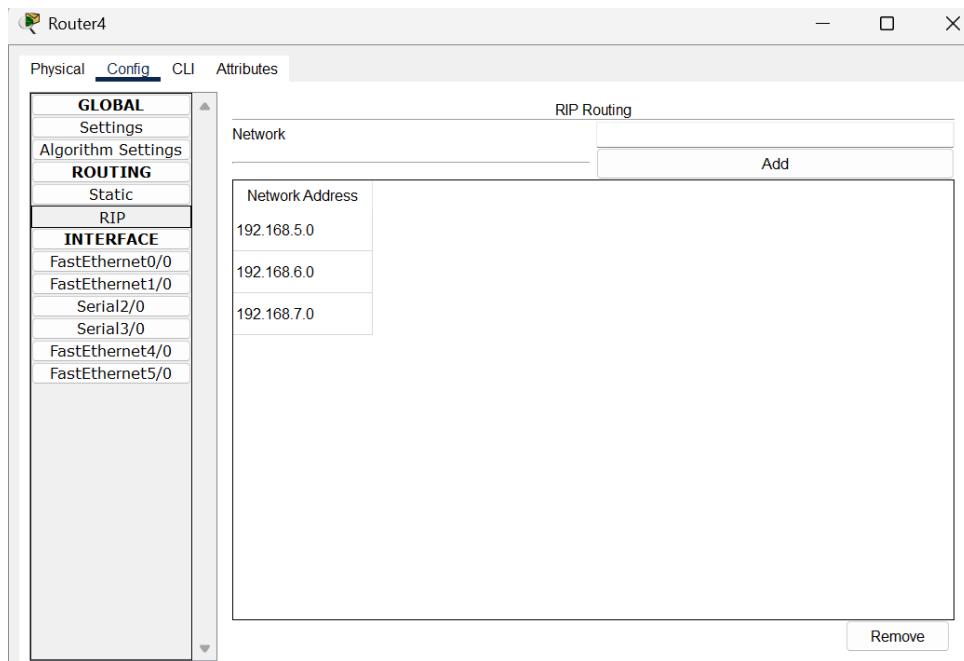
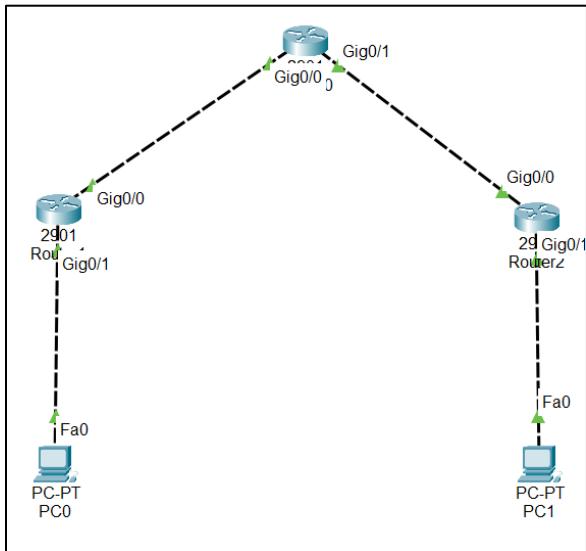


Fig 3. Router 4 – RIP routing

## Program 7:

**Aim:** Configure OSPF routing protocol.

**Network diagram:**



**Configuration:**

<p>* Configure OSPF (Open shortest path first) Routing Protocol.</p> <p>Note: If <math>\leq 5</math> Router devices then use RIP Routing Protocol.</p> <p>Connections:</p> <ul style="list-style-type: none"> <li>* PC0           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>- IP address: 192.168.44.2</li> <li>- Subnet Mask: 255.255.255.0</li> <li>- Default Gateway: 192.168.44.1</li> </ul> </li> </ul> </li> <li>* PC1           <ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP Configuration               <ul style="list-style-type: none"> <li>- IP address: 10.10.22.2</li> <li>- Subnet Mask: 255.0.0.0</li> <li>- Default Gateway: 10.10.22.1</li> </ul> </li> </ul> </li> <li>* Router1 select 2901 Router           <ul style="list-style-type: none"> <li>↳ config</li> <li>↳ Interface               <ul style="list-style-type: none"> <li>↳ GigabitEthernet 0/0 (Ge 0/0)                   <ul style="list-style-type: none"> <li>- on</li> <li>- IP Address: 192.168.44.1</li> <li>- Subnet Mask: 255.255.255.0</li> </ul> </li> <li>↳ GigabitEthernet 0/1 (Ge 0/1)                   <ul style="list-style-type: none"> <li>- on</li> <li>- IP Address: 192.168.55.2</li> <li>- Subnet Mask: 255.255.255.0</li> </ul> </li> </ul> </li> <li>↳ CLI               <ul style="list-style-type: none"> <li>↳ [Enter the following commands one by one]</li> <li>enable</li> <li>config +</li> <li>router ospf 1</li> <li># network 192.168.55.0 0.0.0.255 area 0</li> <li># network 192.168.44.0 0.0.0.255 area 0</li> <li># exit</li> </ul> </li> </ul> </li> </ul>	<p>* Router2 (Select 2901 Router)</p> <p>↳ config</p> <p>↳ Interface</p> <ul style="list-style-type: none"> <li>↳ GigabitEthernet 0/0 (Ge 0/0)           <ul style="list-style-type: none"> <li>- on</li> <li>- IP address: 10.10.22.1</li> <li>- Subnet Mask: 255.0.0.0</li> </ul> </li> <li>↳ GigabitEthernet 0/1 (Ge 0/1)           <ul style="list-style-type: none"> <li>- on</li> <li>- IP address: 172.16.10.2</li> <li>- Subnet Mask: 255.255.0.0</li> </ul> </li> </ul> <p>↳ CLI</p> <p>↳ [Enter following commands one-by-one]</p> <pre># enable # config + # router ospf 1 # network 192.168.55.1 0.0.0.255 area 0 # network 10.10.22.1 0.0.0.255 area 0 # exit</pre> <p>* Router3 (Select 2901 Router)</p> <p>↳ config</p> <p>↳ Interface</p> <ul style="list-style-type: none"> <li>↳ GigabitEthernet 0/0 (Ge 0/0)           <ul style="list-style-type: none"> <li>- on</li> <li>- IP address: 192.168.55.1</li> <li>- Subnet Mask: 255.255.255.0</li> </ul> </li> <li>↳ GigabitEthernet 0/1 (Ge 0/1)           <ul style="list-style-type: none"> <li>- on</li> <li>- IP address: 172.16.10.1</li> <li>- Subnet Mask: 255.255.0.0</li> </ul> </li> </ul> <p>↳ CLI</p> <p>↳ [Enter following commands one-by-one]</p> <pre># enable # config + # router ospf 1 # network 192.168.55.0 0.0.0.255 area 0 # network 172.16.10.0 0.0.0.255 area 0 # exit</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Output:

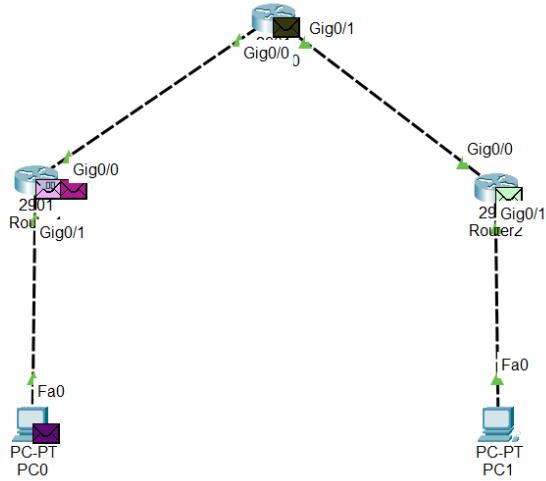


Fig 1. Sending PDU message from PC0 to PC1

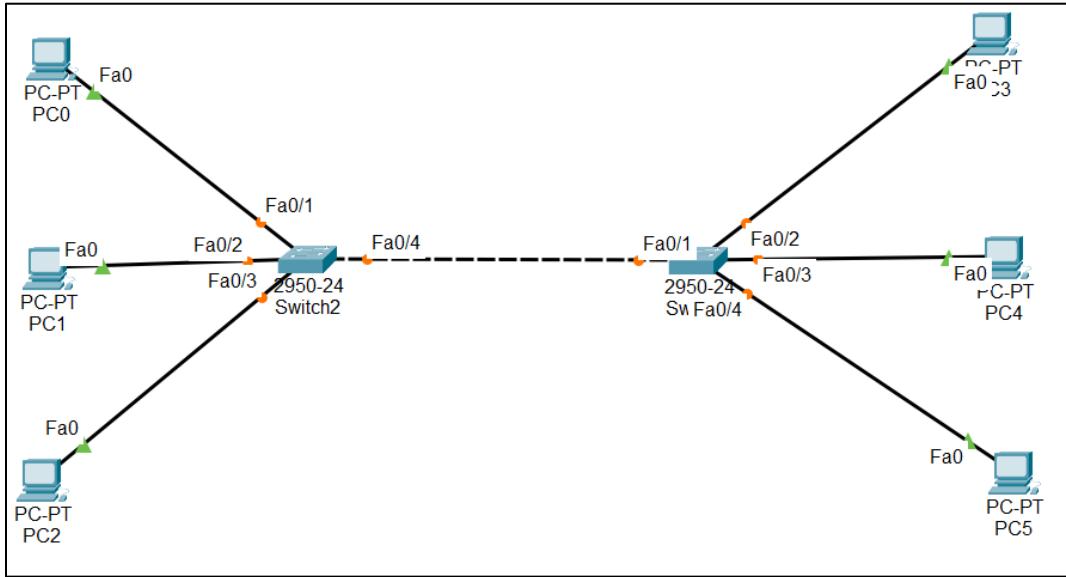
PDU List Window										
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	■	0.000	N	0	(edit)	(delete)
●	Successful	PC0	Router2	ICMP	■	0.000	N	1	(edit)	(delete)
●	Successful	PC0	Router0	ICMP	■	0.000	N	2	(edit)	(delete)
●	Successful	Router0	PC1	ICMP	■	0.000	N	3	(edit)	(delete)
●	Successful	Router1	PC1	ICMP	■	0.000	N	4	(edit)	(delete)
●	Successful	Router1	Router2	ICMP	■	0.000	N	5	(edit)	(delete)

Fig 2. Checking PDU messages

## Program 8:

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

**Network diagram:**



**Configuration:**

<p>16/10/25 To construct a VLAN &amp; make the PCs communicate among a VLAN.</p> <p>Connections:          * PC0:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.2          Subnet Mask: 255.255.255.0</p> <p>* PC1:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.3          Subnet Mask: 255.255.255.0</p> <p>* PC2:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.4          Subnet Mask: 255.255.255.0</p> <p>* PC3:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.5          Subnet Mask: 255.255.255.0</p> <p>* PC4:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.6          Subnet Mask: 255.255.255.0</p> <p>* PC5:          &gt; Desktop          &gt; IP Configuration          &gt; IP address: 192.168.1.7          Subnet Mask: 255.255.255.0</p> <table border="1"> <thead> <tr> <th>VLAN ID</th> <th>fa0/1</th> <th>fa0/2</th> <th>fa0/3</th> <th>fa0/4</th> <th>VLAN ID</th> <th>fa0/1</th> <th>fa0/2</th> <th>fa0/3</th> <th>fa0/4</th> </tr> </thead> <tbody> <tr> <td>vlan10</td> <td>PC0</td> <td>PC1</td> <td>PC2</td> <td></td> <td>vlan20</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> </tr> <tr> <td>fa0/2</td> <td></td> <td></td> <td></td> <td></td> <td>fa0/3</td> <td>PC3</td> <td>PC4</td> <td>PC5</td> <td></td> </tr> <tr> <td>fa0/3</td> <td></td> <td></td> <td></td> <td></td> <td>fa0/4</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	VLAN ID	fa0/1	fa0/2	fa0/3	fa0/4	VLAN ID	fa0/1	fa0/2	fa0/3	fa0/4	vlan10	PC0	PC1	PC2		vlan20															fa0/2					fa0/3	PC3	PC4	PC5		fa0/3					fa0/4					<p>* Switch1:          &gt; CLI (Command Line Interface)          &gt; Type this commands one by one ↴          &gt; enable          config t          int fa 0/1          switchport access vlan 10          int fa 0/2          switchport access vlan 20          int fa 0/3          switchport access vlan 30          int fa 0/4          switchport mode trunk          exit</p> <p>* Switch2:          &gt; CLI          &gt; enable          config t          int fa 0/2          switchport access vlan 10          int fa 0/3          switchport access vlan 20          int fa 0/4          switchport access vlan 30          int fa 0/1          switchport mode trunk          exit</p> <p>PC-PT</p>
VLAN ID	fa0/1	fa0/2	fa0/3	fa0/4	VLAN ID	fa0/1	fa0/2	fa0/3	fa0/4																																										
vlan10	PC0	PC1	PC2		vlan20																																														
fa0/2					fa0/3	PC3	PC4	PC5																																											
fa0/3					fa0/4																																														

## Output:

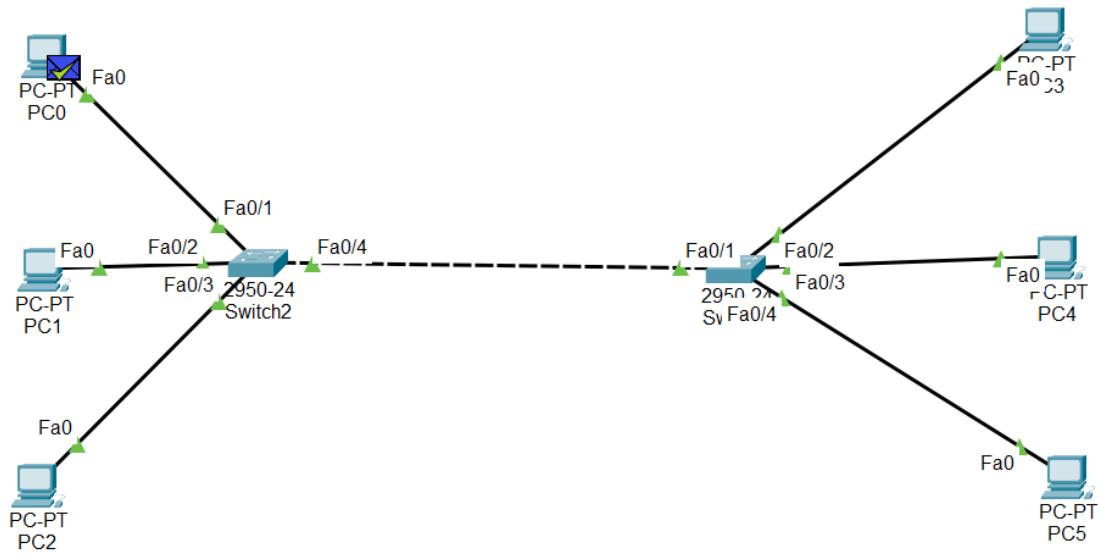


Fig 1. Sending PDU message from PC0 to PC5

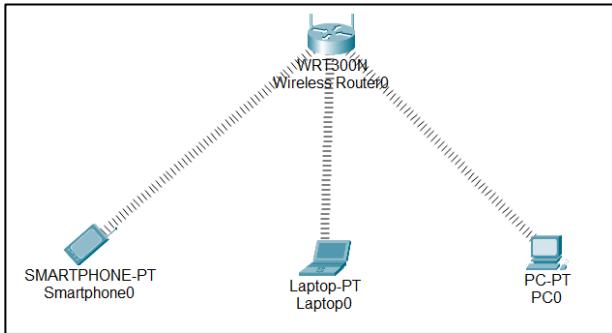
PDU List Window										
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC3	ICMP	■	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC4	ICMP	■	0.000	N	1	(edit)	(delete)
●	Successful	PC0	PC5	ICMP	■	0.000	N	2	(edit)	(delete)
●	Successful	PC1	PC3	ICMP	■	0.000	N	3	(edit)	(delete)
●	Successful	PC1	PC4	ICMP	■	0.000	N	4	(edit)	(delete)
●	Successful	PC1	PC5	ICMP	■	0.000	N	5	(edit)	(delete)
●	Successful	PC2	PC3	ICMP	■	0.000	N	6	(edit)	(delete)
●	Successful	PC2	PC4	ICMP	■	0.000	N	7	(edit)	(delete)
●	Successful	PC2	PC5	ICMP	■	0.000	N	8	(edit)	(delete)
●	Successful	PC3	PC2	ICMP	■	0.000	N	9	(edit)	(delete)

Fig 2. Checking PDU messages

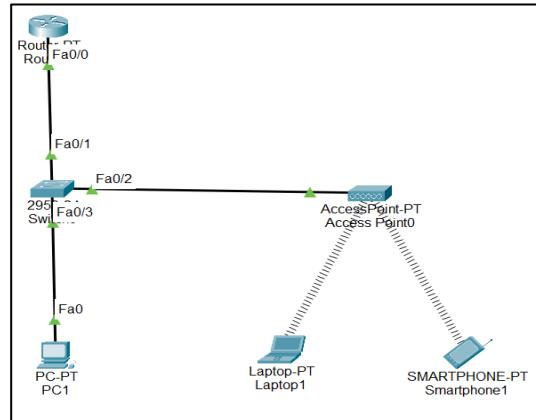
## Program 9:

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

**Network diagram:**

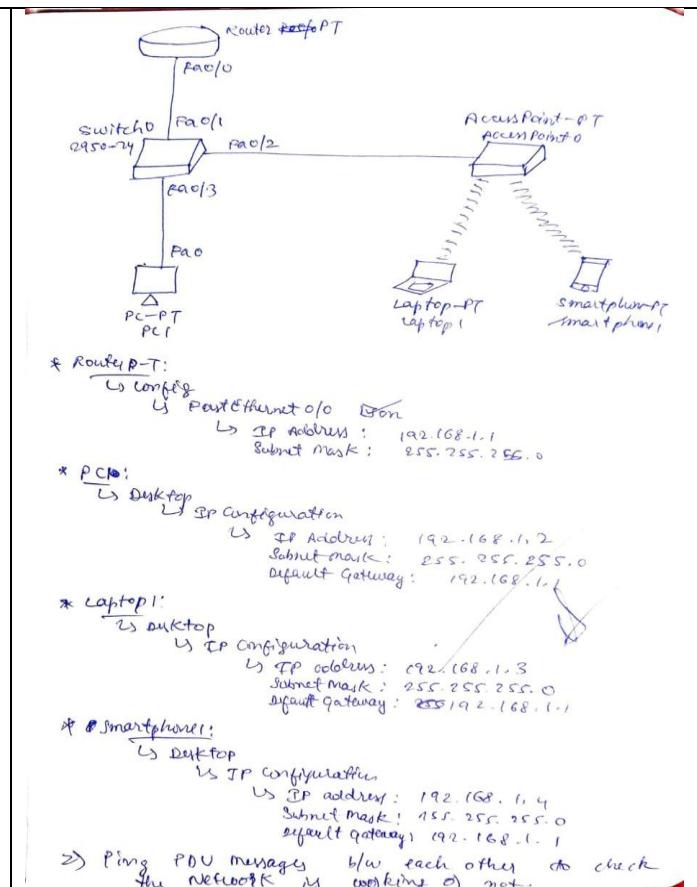
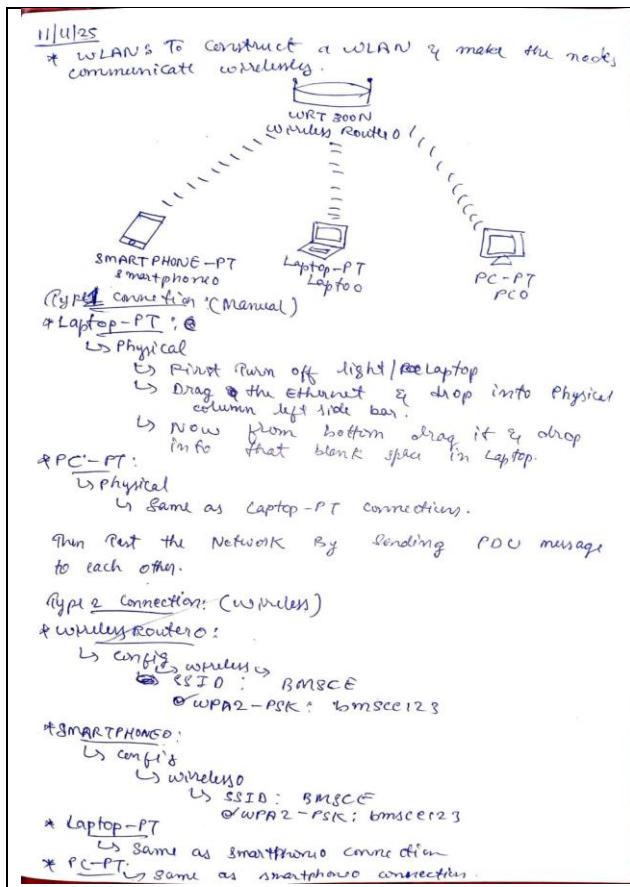


Configuration 1



Configuration 2

**Configuration:**



## Output:

### 1. Do Physical Connections In:

- Laptop
- PC

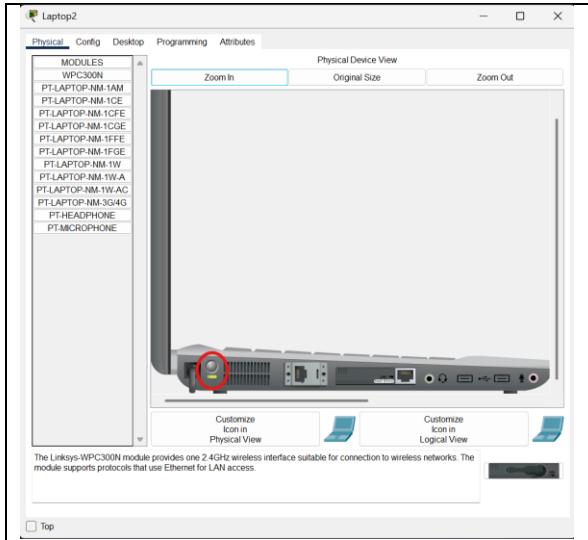


Fig 1.1 Step1: Turn off light / Power off laptop

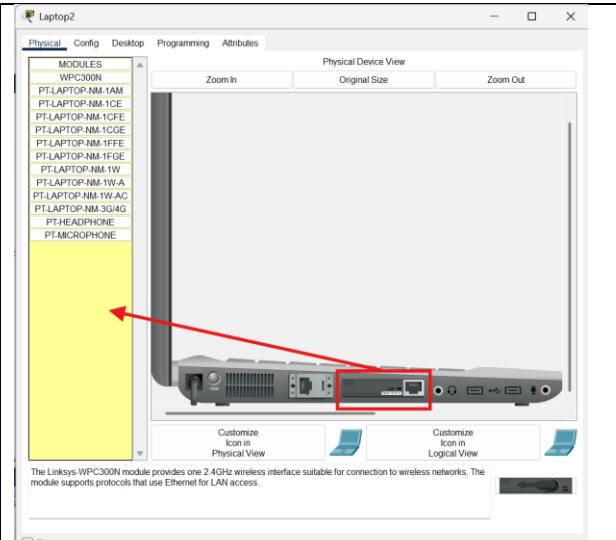


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

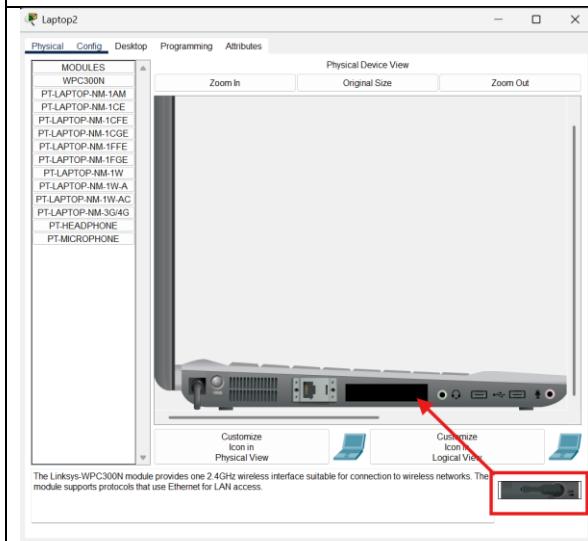


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

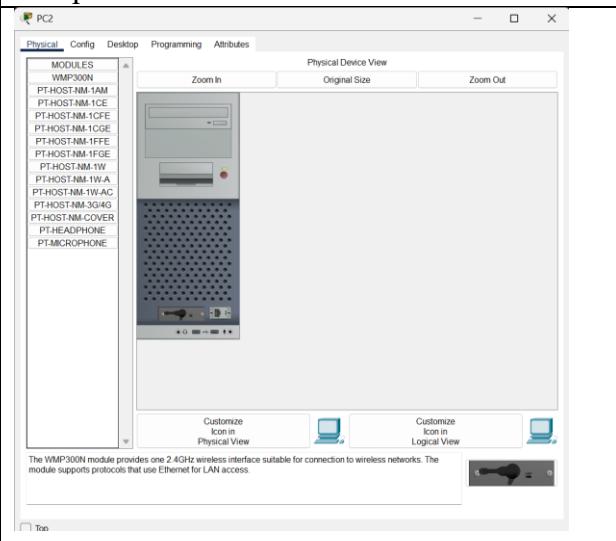


Fig 2. PC physical connection (combined 3 steps)

## 2. Do Wireless Connection in:

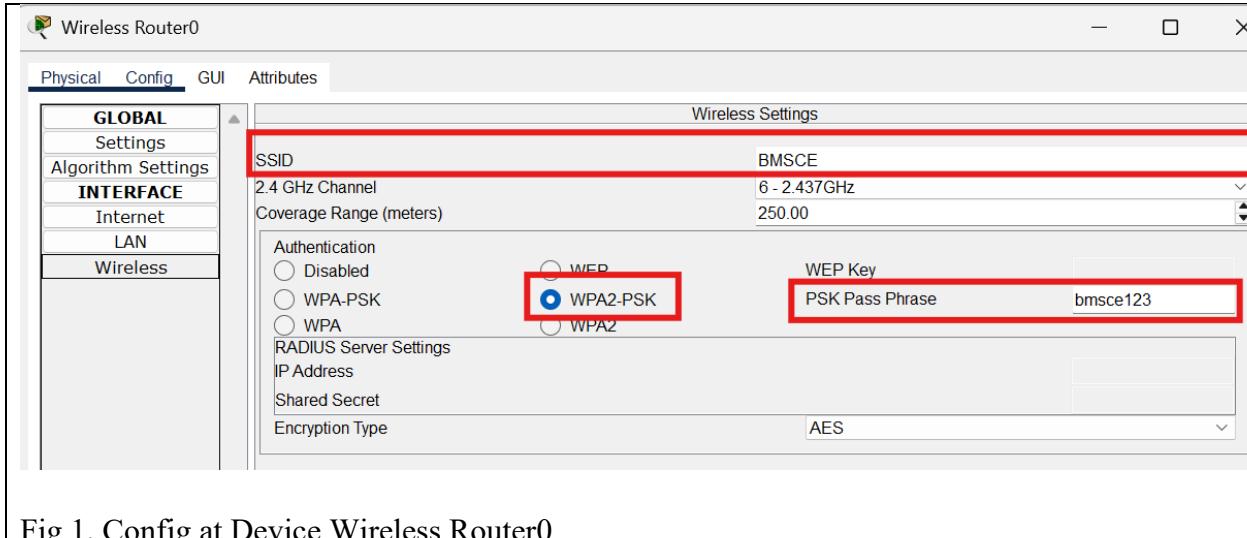


Fig 1. Config at Device Wireless Router0

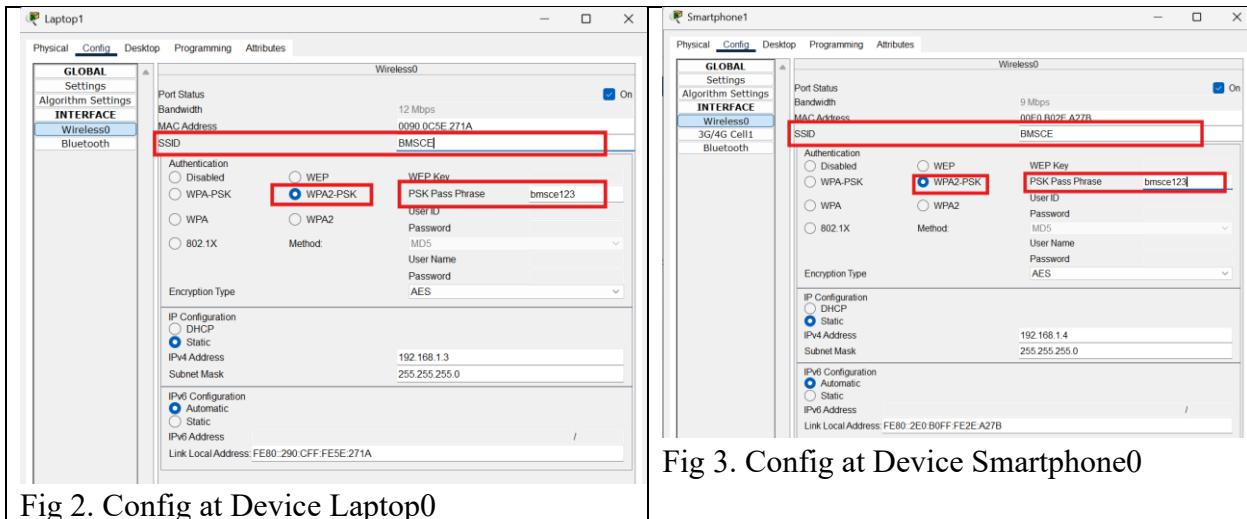


Fig 2. Config at Device Laptop0

Fig 3. Config at Device Smartphone0

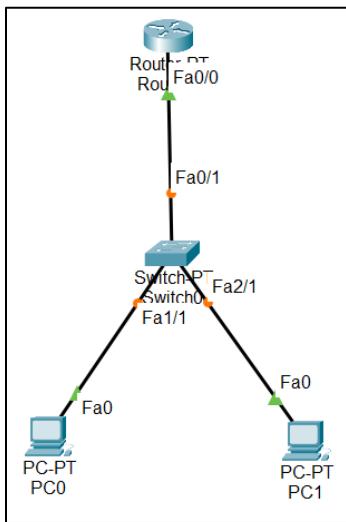
PDU List Window										
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	
●	Failed	Smar...	Laptop0	ICMP	purple	0.000	N	0	(edit)	
●	Successful	Laptop...	PC0	ICMP	dark purple	0.000	N	1	(edit)	
●	Failed	PC0	Laptop0	ICMP	dark green	0.000	N	2	(edit)	
●	Successful	PC0	Smartphone0	ICMP	light purple	0.000	N	3	(edit)	
●	Failed	PC0	Laptop0	ICMP	teal	0.000	N	4	(edit)	
●	Successful	Laptop...	Smartphone0	ICMP	purple	0.000	N	5	(edit)	
●	Successful	Laptop...	PC0	ICMP	green	0.000	N	6	(edit)	
●	Successful	PC0	Smartphone0	ICMP	pink	0.000	N	7	(edit)	
●	Successful	Laptop...	PC1	ICMP	yellow-green	0.000	N	8	(edit)	

Fig 3. Checking PDU messages

## Program 10:

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

**Network diagram:**



**Configuration:**

Demonstrate the TTL / life of a packet:		
	Router PT Router 0	
	Switch PT Switch 0	
* PC0 :	PC-PT PC0	PC-PT PC1
* Router PT:	<ul style="list-style-type: none"> <li>↳ Config</li> <li>↳ fastEthernet 0/0</li> <li>↳ IP address: 192.168.1.1</li> <li>↳ Subnet mask: 255.255.255.0</li> </ul>	
* PC0 :	<ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP configuration</li> <li>↳ Static</li> <li>IP Address: 192.168.1.2</li> <li>Subnet mask: 255.255.255.0</li> <li>Default Gateway: 192.168.1.1</li> </ul>	
* PC1 :	<ul style="list-style-type: none"> <li>↳ Desktop</li> <li>↳ IP configuration</li> <li>↳ Static</li> <li>IP address: 192.168.1.3</li> <li>Subnet mask: 255.255.255.0</li> <li>Default gateway: 192.168.1.1</li> </ul>	
⇒ Ping PDU messages b/w each other to check the network is working or not.		
⇒ In Simulation Mode;	<ul style="list-style-type: none"> <li>↳ while PDU messages are pinging/transferring</li> <li>↳ Tap on messages</li> <li>↳ Go to Outbound &amp; Inbound PDU details</li> <li>↳ check what is TTL value</li> <li>↳ TTL varies b/w 128 &amp; 255</li> </ul>	

TTL Value	Likely System	Max Hops
64	Linux / macOS	64
128	Windows	128
255	Cisco / unix	255
Input	TTL = 128	TTL = 255
Meaning	The packet can pass through up to 128 routers before being discarded.	The packet can pass through up to 255 routers before being discarded.
Typical Unit / Source	Common default TTL value used by Windows OS.	Common default TTL value used by Cisco / Linux, Cisco routers, and macOS.
Maximum Hops Allowed	128 hops	255 hops
Network Implication	Slightly lower lifetime, packet expires sooner if there's a long route.	Longer lifetime; packet can traverse more routers before expiring.
Tracing / Identification	Helps identify OS in network forensics or ping replies.	Same - helps detect source device type.
Security / Performance	No security advantage; just a design choice.	No major advantage - just allows for more hops.

## Output:

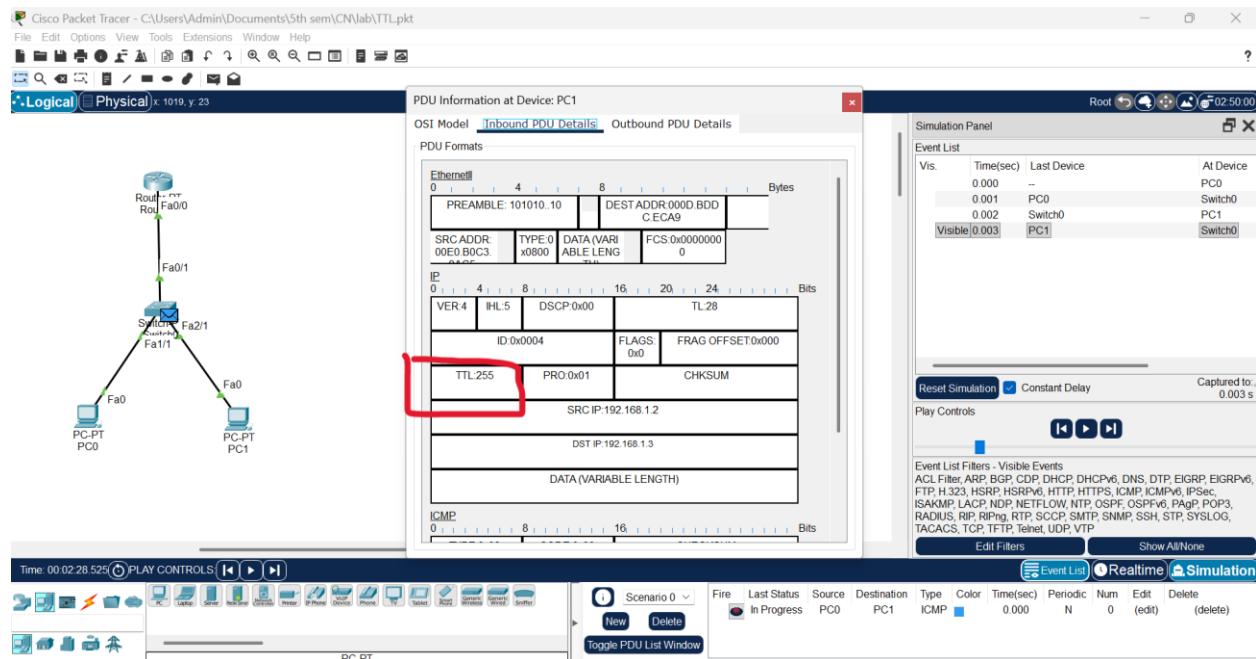


Fig 1. Inbound PDU Details at Device PC1

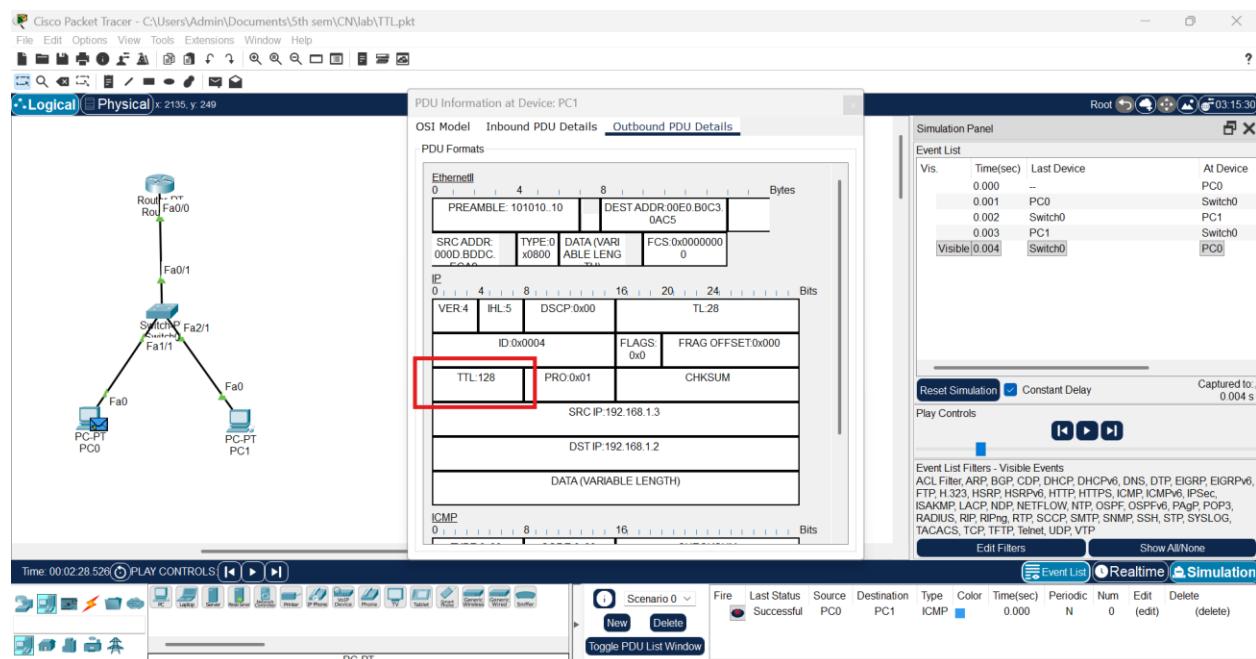
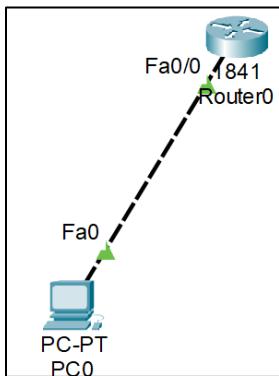


Fig 1. Outbound PDU Details at Device PC1

## Program 11:

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

### **Network diagram:**

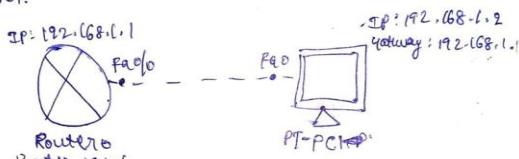


### **Configuration:**

Rules  
\* Construct a topology to demonstrate concept of Telnet.

#### TELNET:

- It is used to access remote servers.
- It's a simple command line tool that runs on your computer and it will allow you to send a command remotely to a server.
- It is also used managed devices like switches, routers ports are open or close on a server.



#### Router0:

- ↳ Go to CLI
- ↳ [Yes/No]: No
- ↳ config
- ↳ enable
- ↳ config t
- ↳ hostname R1
- ↳ enable secret rp
- ↳ int Fa0/0
- ↳ ip add 192.168.1.1 255.255.255.0
- ↳ no shutdown
- ↳ enter
- ↳ enter

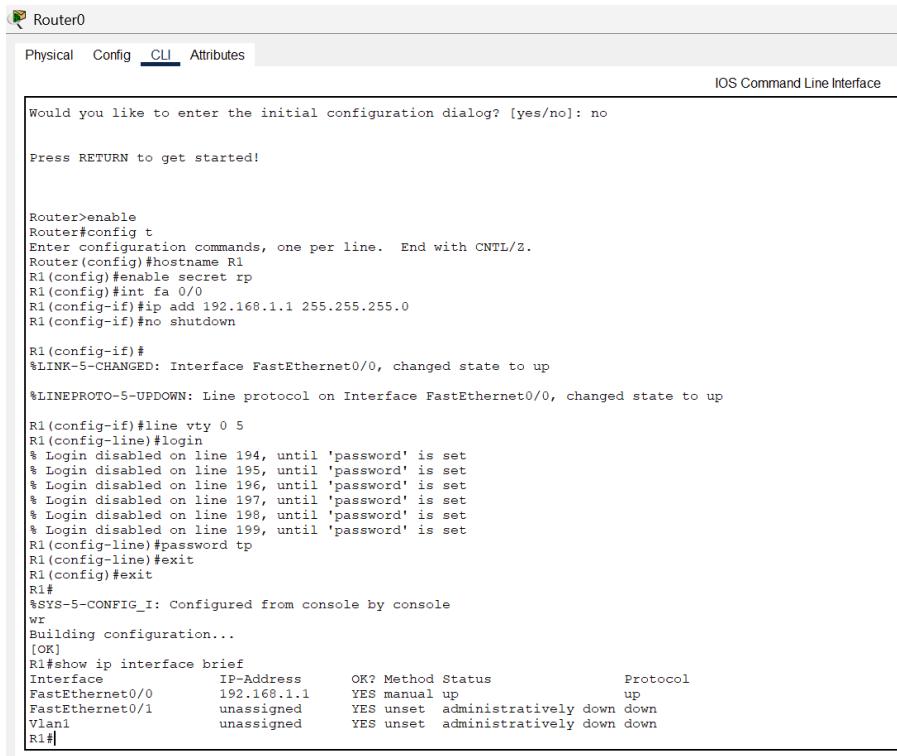
↳ link vty 0 5  
↳ login  
↳ password tp  
↳ exit  
↳ exit  
↳ user  
↳ show ip interface brief.

#### \*PC:

- PC > ping 192.168.1.1
- PC > telnet 192.168.1.1
- Dialing 192.168.1.1 open
- User Access verification
- password : tp
- R1 > enable
- Password : rp
- R1# show ip interface brief
- R1# en
- R1# config t
- R1(config)# int Fa0/1
- R1(config)# ip add 192.168.1.2 255.255.255.0
- ↳ 192.168.1.0 overlaps with FastEthernet 0/0

Then again go to Router 2  
R1 > show ip interface brief  
changes are made.

## Output:



Router0

Physical Config **CLI** Attributes

IOS Command Line Interface

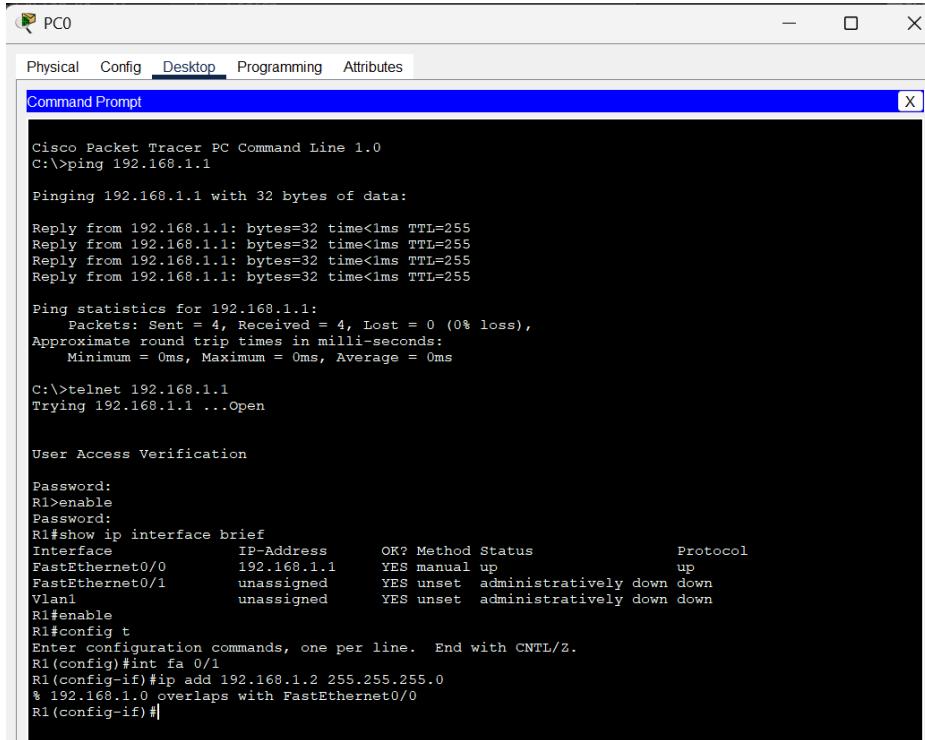
```
Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!

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)#
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



PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
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)#
R1#
```

Fig2. PC command line prompt

Router0

Physical Config CLI Attributes

IOS Command Line Interface

```

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)#
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#show ip interface brief
Interface          IP-Address      OK? Method Status       Protocol
FastEthernet0/0    192.168.1.1    YES manual up        up
FastEthernet0/1    192.168.1.2    YES manual administratively down down
Vlan1              unassigned      YES unset administratively down down
R1#

```

Top

Copy Paste

Fig 3. Updated the changes into Router0

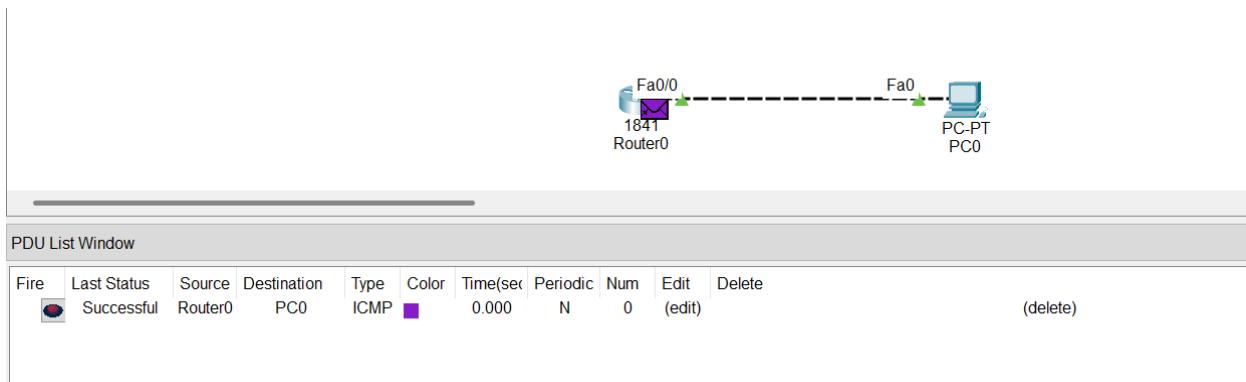
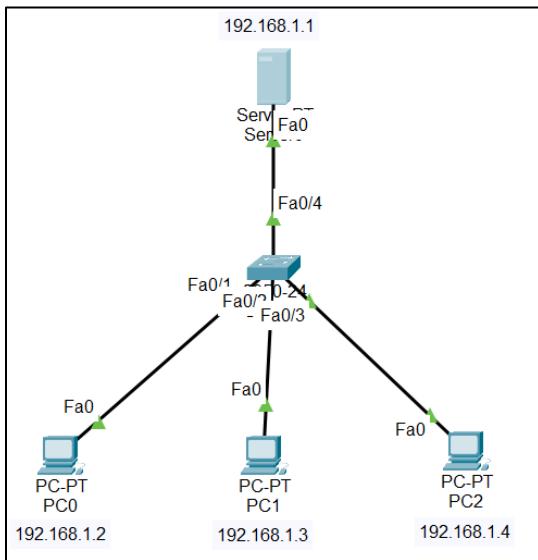


Fig 4. PDU message Successful

## Program 12:

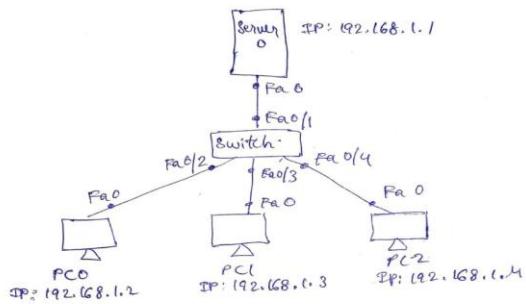
**Aim:** To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

**Network diagram:**



**Configuration:**

- \* To construct simple LAN to understand and operation of Address Resolution Protocol (ARP)
- ARP is used to map an IP address to a MAC address.
- ARP is used to get data-link layer address, MAC address with the help of IP address.



- Take a magnification glass and click on Server and PCs and get ARP table.
- Do the IP configuration for PCs & Server:  
 Server : 192.168.1.1  
 PC0 : 192.168.1.2  
 PC1 : 192.168.1.3  
 PC2 : 192.168.1.4

- Select PC0
  - ↳ Go to command prompt
  - ↳ Type → PC > arp -a
  - No ARP Entries Found

- Then go to Simulation Mode & send the PDU messages to the PCs and server.

→ Click on the magnifying glass and click on the outbound boundaries and check the ARP tables.

ARP table for Server:

IP: 192.168.1.4  
 HW add: 0060.3E88.4332  
 Interface: Fast Ethernet 0

ARP Table for ~~Server~~ PC0:

IP: 192.168.1.4  
 HW add: 0060.3E88.4332  
 Interface: Fast Ethernet 0

ARP Table for PC1:

IP: 192.168.1.4  
 HW add: 0060.3E88.4332  
 Interface: Fast Ethernet 0

ARP Table for PC2:

IP: 192.168.1.1, 192.168.1.2, 192.168.1.3  
 HW add: 0009.BD02.593E, 000C.CF4D.05BD,  
 0006.2A26.00339  
 Interface: Fast Ethernet 0, Fast Ethernet 0, Fast Ethernet 0

(D)

## Output:

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

```

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=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

```

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

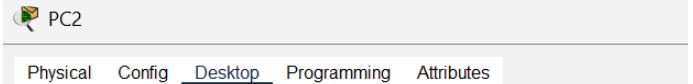
C:>arp -a
Internet Address      Physical Address      Type
192.168.1.1            00e0.f7c6.ac93    dynamic

```

Fig 3.2 Command Prompt at PC1

ARP Table for PC2		
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 4.1 ARP table at PC2



```

PC2
Physical Config Desktop Programming Attributes

Command Prompt
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 4.2 Command Prompt at PC2

## 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);
```

```

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

    }

    printf("\nSecond | Packet Received | Packet Sent | Packet
Left | Dropped\n");
    printf("-----\n-----\n");

    for (i = 0; i < nsec; i++) {
        count += inp[i];

        if (count > cap) {
            drop = count - cap;
            count = cap;
        }

        printf("%d\t %d\t\t", i + 1, inp[i]);

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

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

        drop = 0;
    }

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

```

```

        drop = count - cap;

        count = cap;

    }

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

mini = min(count, process);

printf("%d\t\t", mini);

count = count - mini;

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

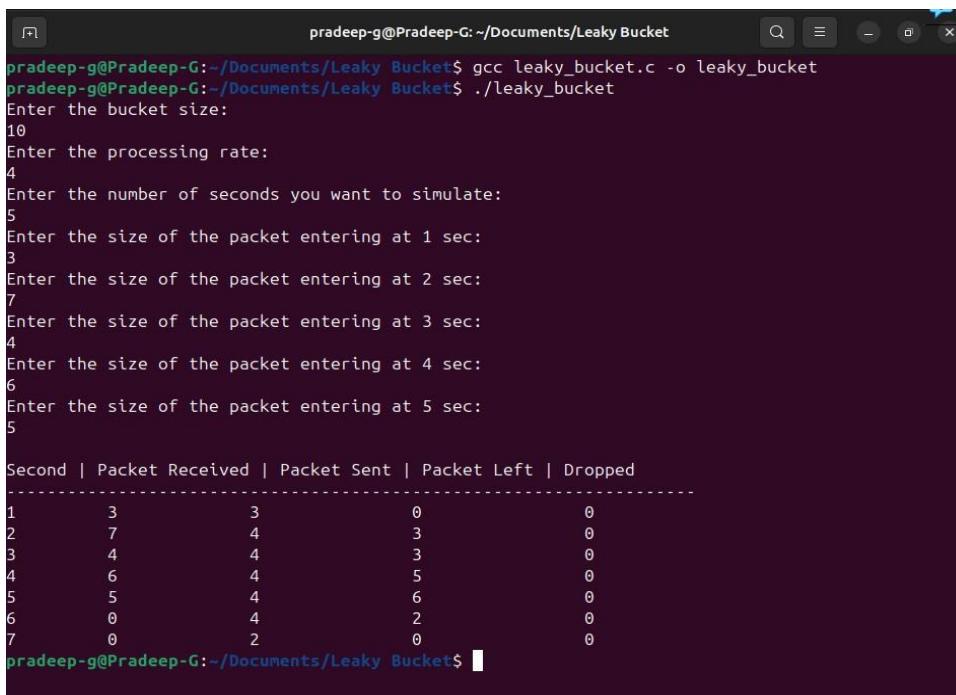
drop = 0;

}

return 0;
}

```

## Output:



```

pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ gcc leaky_bucket.c -o leaky_bucket
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ ./leaky_bucket
Enter the bucket size:
10
Enter the processing rate:
4
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:
7
Enter the size of the packet entering at 3 sec:
4
Enter the size of the packet entering at 4 sec:
6
Enter the size of the packet entering at 5 sec:
5

Second | Packet Received | Packet Sent | Packet Left | Dropped
-----
1      3              3              0              0
2      7              4              3              0
3      4              4              3              0
4      6              4              5              0
5      5              4              6              0
6      0              4              2              0
7      0              2              0              0
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ 

```

## Observation:

```

Code:
#include <stdio.h>

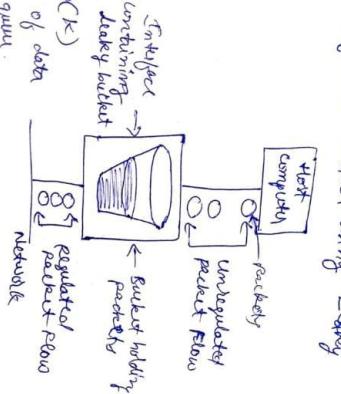
int min( int x, int y ) {
    int drop = 0, mini, msec, cap, count = 0, i;
    imprec;
    printf( "Enter the bucket size :\n" );
    scanf( "%d", &cap );
    printf( "Enter the processing rate :\n" );
    scanf( "%d", &msec );
    count = 0;
    printf( "Enter the no. of seconds you want to
simulate :\n" );
    scanf( "%d", &nsec );
    for( i = 0; i < nsec; i++ ) {
        count += imprec;
        if( count > cap ) {
            drop = count - cap;
            count = cap;
        }
        printf( "%d\t %d\t %d", i+1, imprec );
        mini = min( count, process );
        printf( "%d\t %d", i+1, mini );
        count = count - mini;
        printf( "%d\t %d\t %d", i+1, count, drop );
        drop = 0;
    }
}

Remaining packets after time ends
for( ; count != 0; i++ ) {
    if( count > cap ) {
        drop = count - cap;
        count = cap;
    }
    printf( "%d\t %d\t %d", i+1 );
    mini = min( count, process );
    printf( "%d\t %d", i+1, mini );
    count = count - mini;
    printf( "%d\t %d\t %d", i+1, count, drop );
    drop = 0;
}
return 0;

```

**Code:**  
~~Time complexity:  $O(k)$~~   
 Time in ~~Space complexity:  $O(k)$~~   
 where  $k$  = no. of data packets in the queue.

17/11/25  
 write a program for congestion control using Leaky Bucket algorithm.  
  
 Time interval containing leaky bucket



**Output:**  
 Enter the bucket size:  
 10  
 Enter the processing rate:  
 4  
 Enter the no. of seconds you want to simulate:  
 5

Enter the size of the packet entering at 1sec: 3  
 Enter the size of the packet entering at 2sec: 7  
 3sec: 4  
 4sec: 6  
 5sec: 5  

Second	Packet Received	Packet Sent	Packet Left	Dropped
1	3	3	0	0
2	7	4	3	0
3	4	4	3	0
4	6	4	4	2
5	5	5	0	0

## **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

import socket

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

# Step 2: Connect to 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 contents
data =
client_socket.recv(4096).decode()

print("\n--- File Content ---\n")
print(data)

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

```
# tcp_server.py

import socket

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

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

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

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

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

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()
```

## Output:

Server side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/TCP $ python3 server.py
Server is listening on port 8080...
Connected by: ('127.0.0.1', 47790)
pradeep-g@Pradeep-G: ~/Documents/TCP $
```

Client side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/TCP $ python3 client.py
Enter filename to request: hello.txt
--- File Content ---
Hi i am Pradeep G
Welcome to my WORLD!
pradeep-g@Pradeep-G: ~/Documents/TCP $
```

## Observation:

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

Code: (Python -> in Ubuntu)

client.py

```
# Step1: Create TCP Socket
client_socket = socket.socket(socket.AF_INET,
                             socket.SOCK_STREAM)

# Step2: Connect to server
client_socket.connect(('localhost', 8080))

# Step3: Send filename
filename = input("Enter filename to request: ")
client_socket.send(filename.encode())

# Step4: Receive file contents
data = client_socket.recv(4096).decode()
print("\n-- File Content --\n")
print(data)

# Step5: Close connection
client_socket.close()
```

server.py

```
# Step1: Create a TCP socket
server_socket = socket.socket(socket.AF_INET,
                             socket.SOCK_STREAM)

# Step2: Bind to address and port
server_socket.bind(('localhost', 8080))

# Step3: Listen for client connections
server_socket.listen(1)
print("Server is listening on port 8080...")

# Step4: Accept connection
conn, addr = server_socket.accept()
print("Connected by: ", addr)

# Step5: Receive file name
filename = conn.recv(1024).decode().strip()

try:
    # Step6: Open & 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.")

# Step7: Close connection
conn.close()
server_socket.close()
```

Output: Run/Compile in Ubuntu terminal.

& python3 server.py  
Server is listening on port 8080...  
Connected by: ('127.0.0.1', 47790)

Client terminal:

& python3 client.py  
Enter filename to request: hello.txt  
-- File Content --  
Hi i am Pradeep G  
Welcome to my WORLD !

### Program 3:

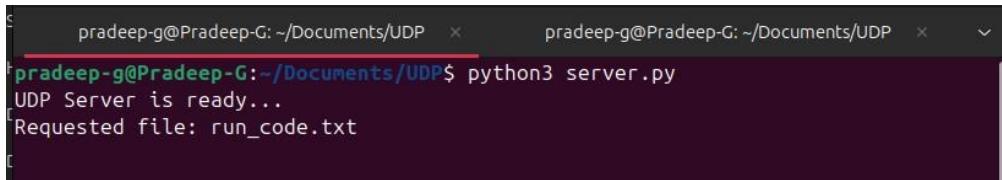
**Aim:** Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

#### **Code:**

# udp_client.py	# udp_server.py
import socket	import socket
# Step 1: Create UDP socket	# Step 1: Create UDP socket
client_socket =	server_socket =
socket.socket(socket.AF_INET,	socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)	socket.SOCK_DGRAM)
server_address = ('localhost',	# Step 2: Bind to address and port
8081)	server_socket.bind(('localhost',
filename = input("Enter filename	8081))
to request: ")	print("UDP Server is ready...")
# Step 2: Send filename to	while True:
server	# Step 3: Receive filename
client_socket.sendto(filename.en	from client
code(), server_address)	filename, addr =
# Step 3: Receive response	server_socket.recvfrom(1024)
data, addr =	filename =
client_socket.recvfrom(4096)	filename.decode().strip()
print("\n--- File Content ---	print(f"Requested file:
\n")	{filename}")
print(data.decode())	try:
# Step 4: Close socket	# Step 4: Open file and
client_socket.close()	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)

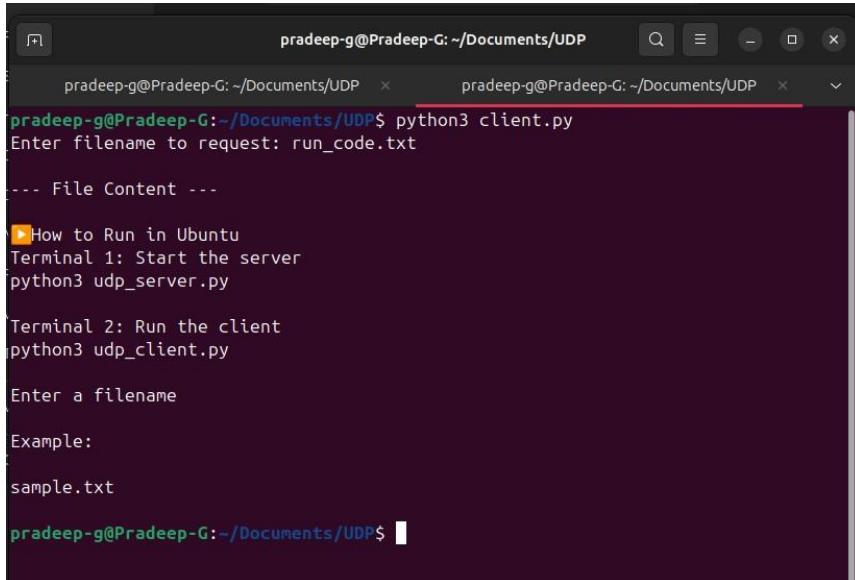
## Output:

Server side Terminal:



```
pradeep-g@Pradeep-G: ~/Documents/UDP$ python3 server.py
UDP Server is ready...
Requested file: run_code.txt
```

Client side Terminal:



```
pradeep-g@Pradeep-G: ~/Documents/UDP$ python3 client.py
Enter filename to request: run_code.txt
--- File Content ---
▶ How to Run in Ubuntu
Terminal 1: Start the server
python3 udp_server.py

Terminal 2: Run the client
python3 udp_client.py

Enter a filename
Example:
sample.txt

pradeep-g@Pradeep-G: ~/Documents/UDP$
```

## Observation:

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:  
[client.py]

```
# udp_client.py
import socket
#step1: Create UDP socket
client_socket = socket.socket(socket.AF_INET,
                             socket.SOCK_DGRAM)
server_address = ('localhost', 8081)
filename = input("Enter filename to request: ")
#step2: Send filename to server
client_socket.sendto(filename.encode(), server_address)
#step3: Receive response
data, addr = client_socket.recvfrom(4096)
print ("\\n-- File Content --\\n")
print (data.decode())
#step4: Close socket
client_socket.close()
```

[server.py]

```
# udp_server.py
import socket
#step1: Create UDP socket
server_socket = socket.socket(socket.AF_INET,
                             socket.SOCK_DGRAM)
#step2: Bind to address and port
server_socket.bind(('localhost', 8081))
print("UDP server is ready...")
while True:
    #step3: Receive filenames from client
```

filename, addr = server\_socket.recvfrom(1024)
filename = filename.decode().strip()
print(f"Requested file: {filename}")
try:
 # step 4: Open file & read content
 with open(filename, 'r') as f:
 data = f.read()
 server\_socket.sendto(data.encode(), addr)
except FileNotFoundError:
 server\_socket.sendto(b"File not found in", server, addr)

Output:  
Server Terminal:  
\$ python3 server.py  
UDP Server is ready...  
Requested file: run\_code.txt  
Client Terminal:  
\$ python3 client.py  
Enter filename to request: run\_code.txt  
--- File Content ---  
Johny Johny Yes Papa!  
Eating Sugar  
No Papa!  
Telling Lies  
No Papa!  
Open your mouth  
Hai! Hai! Hai!

## **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");

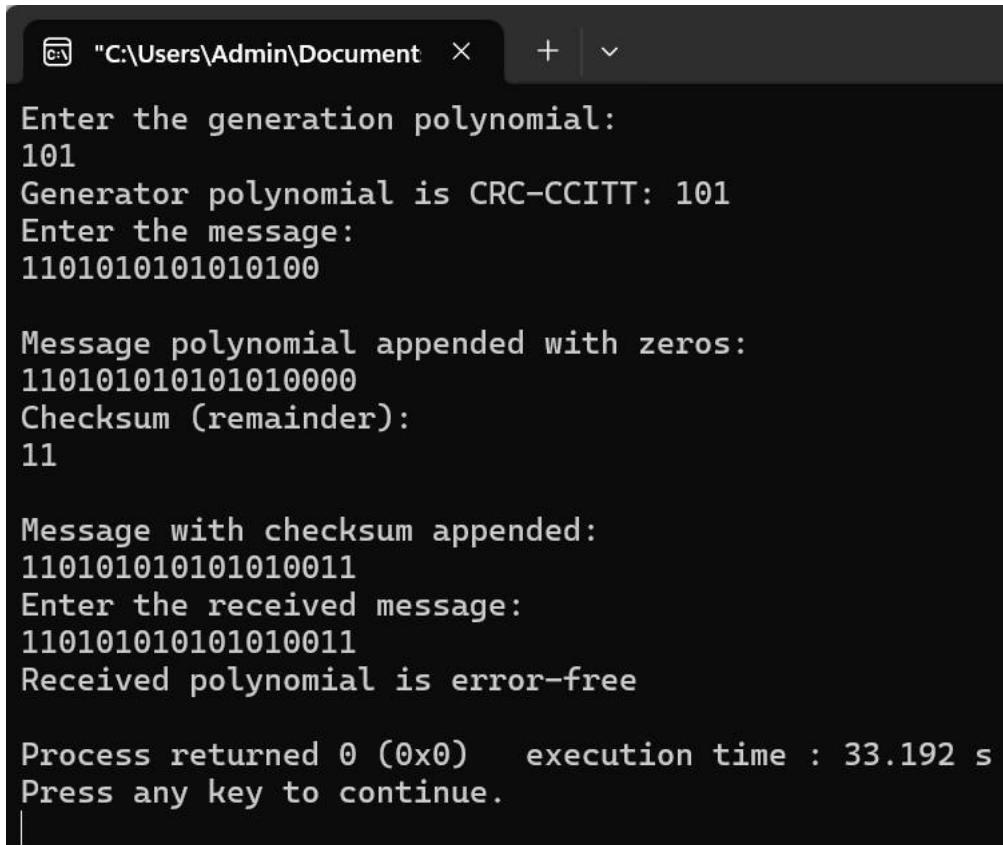


    return 0;

}

```

## Output:



```

C:\ "C:\Users\Admin\Document" × + | ▾
Enter the generation polynomial:
101
Generator polynomial is CRC-CCITT: 101
Enter the message:
1101010101010100

Message polynomial appended with zeros:
110101010101010000
Checksum (remainder):
11

Message with checksum appended:
110101010101010011
Enter the received message:
110101010101010011
Received polynomial is error-free

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

```

## Observation:

Ques/25

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

Pseudocode:

```

BEGIN
    INPUT generator polynomial  $\rightarrow g$ 
    genlen  $\leftarrow \text{LENGTH}(g)$ 
    K  $\leftarrow \text{genlen} - 1$ 
    INPUT message bits  $\rightarrow m$  [of lenm]
    a  $\leftarrow m$  + K zeros
    FOR each bit i in a[0..n-1]
        IF a[i] == '1' THEN
            FOR j  $\leftarrow 0$  TO K
                a[i+j]  $\leftarrow \text{xor}(a[i+j], g[j])$ 
        END FOR
    END IF
    END FOR
    sum  $\leftarrow$  last K bits of a
    transmitted_message  $\leftarrow m$  + sum
    PRINT transmitted_message

    INPUT received message  $\rightarrow s$ 
    FOR each bit i in s[0..n-1]
        IF s[i] == '1' THEN
            FOR j  $\leftarrow 0$  TO K
                s[i+j]  $\leftarrow \text{xor}(s[i+j], g[j])$ 
            END FOR
        END IF
    END FOR
    sum  $\leftarrow$  last K bits of s
    IF all bits of sum = '0' THEN
        Print "Received message is error-free"
    ELSE
        Print "Received message contains errors"
    ENDIF
END

```

Output:

Enter the generation polynomial:

101  
Generated Polynomial is CRC-CCITT: 101  
Enter the message:  
1101010101010100

message polynomial appended with zeros:  
110101010101010000

checksum (remainder):

11

message with checksum appended:

110101010101010011

Enter the received message:

110101010101010011

Received polynomial is error-free.

Error detection  
 1) Parity check  $\Rightarrow$  odd/even parity  
 Now total len of 11 is 1:4 which is even parity  
 2) Checksum  $\rightarrow$  identify error  
 3) CRC  $\rightarrow$  (Polynomial division)  
 when sum = 0  $\Rightarrow$  error-free  
 sum  $\neq$  0  $\Rightarrow$  errors

\* CRC:

- CRC is the most powerful & easy to implement
- CRC is based on binary division
- Calculate the Remainder
- If the Remainder is 0 at destination then it is error-free data if not then errored data.

Bob's logic:

① Frame: 11001

Polynomial Generator ( $G$ ) = 101

$x^3 + 1$

↓ Sender Side:

↑ Append into frame

$1 \oplus 1 = 0$

$1 \oplus 1 = 0$

1100100

↓ Receiver Side:

$1 \oplus 1 = 0$

$1 \oplus 1 = 0$

1100101

②

Frame: 100100

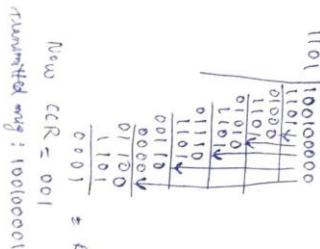
Polynomial Generator ( $G$ ):  $x^3 + x^2 + 1$   $\Rightarrow$  1101

Binary Shift

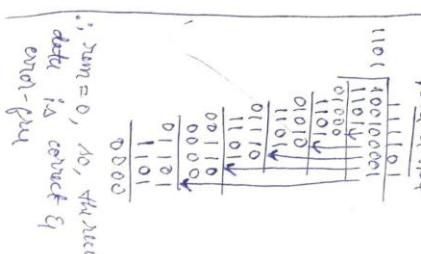
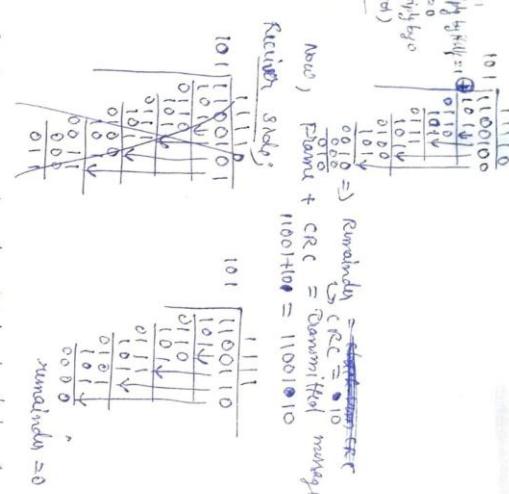
$$101 = 5 - 1 = 4 - 1 = 3$$

append zeros into frame

Now frame: 100100000



∴ sum = 0, so the received data is correct & error-free.



∴ sum = 0, so the received data is correct & error-free.