VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Gopika Pushparajan (1BM23CS101)

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 **Gopika Pushparajan (1BM23CS101),** 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

Part - B

Github Link:

https://github.com/gopikapushparajan/cnlab

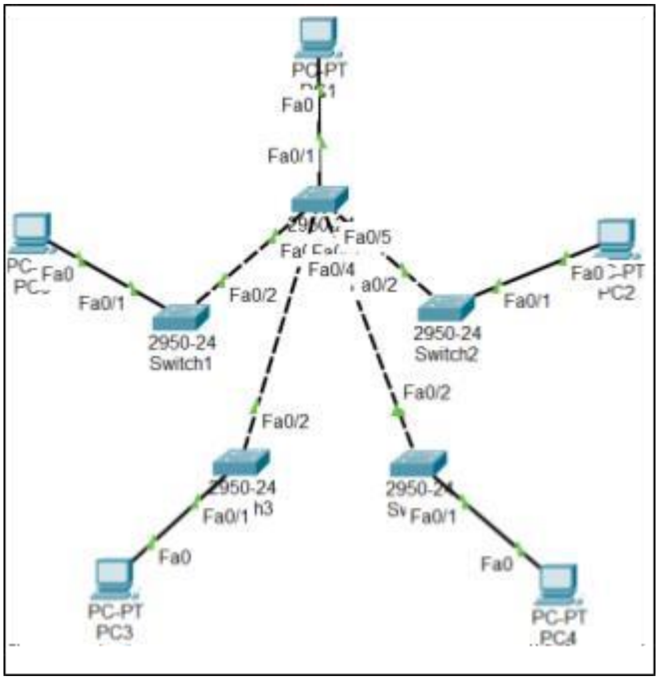
PART - A

Program 1:

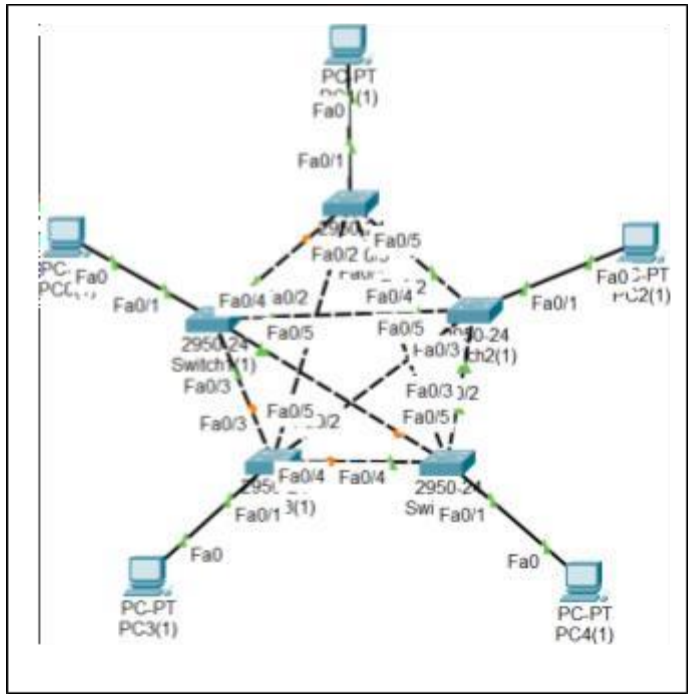
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Topology:

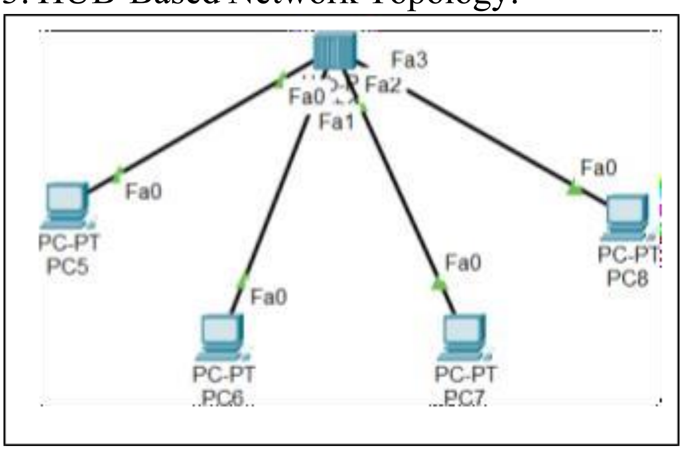
1. STAR Topology with Switch:



2. MESH Topology with Switch:



3. HUB-Based Network Topology:



Procedure:

1. Create STAR Topology Using a Switch

1. Open Cisco Packet Tracer and go to the End Devices section.
2. Drag and drop PCs (PC0, PC1, PC2, PC3, PC4) into the workspace.
3. From Switches, drag a 2950-24 switch to the center.
4. Connect each PC to the switch using Copper Straight-Through cables:
   - PC0 → Switch (Fa0/1)
   - PC1 → Switch (Fa0/2)
   - PC2 → Switch (Fa0/5)
   - PC3 → Switch (Fa0/3)
   - PC4 → Switch (Fa0/4)
5. Assign IP addresses to PCs:
   - Go to PC → Desktop → IP Configuration
   - Enter the IP address/subnet for each PC (any address in same network).
6. Test connectivity:
   - Use Add Simple PDU tool to send a ping from one PC to another.

2. Create MESH Topology Using Switches

1. Drag and drop PCs (PC0, PC1, PC2, PC3, PC4).
2. Add two 2950-24 switches to the workspace.
3. Create mesh-style interconnections:
   - Connect each PC to the nearest switch.
   - Connect Switch1 ↔ Switch2 with multiple redundant links (e.g., Fa0/1 ↔ Fa0/3, Fa0/2 ↔ Fa0/4).
4. Assign IP addresses to all PCs within the same network.
5. Verify STP operation automatically blocks redundant paths.
6. Use Simple PDU (ICMP) to test ping between:
   - PC0 → PC3
   - PC1 → PC4
   - PC2 → any node
7. View packet movement under Simulation Mode.

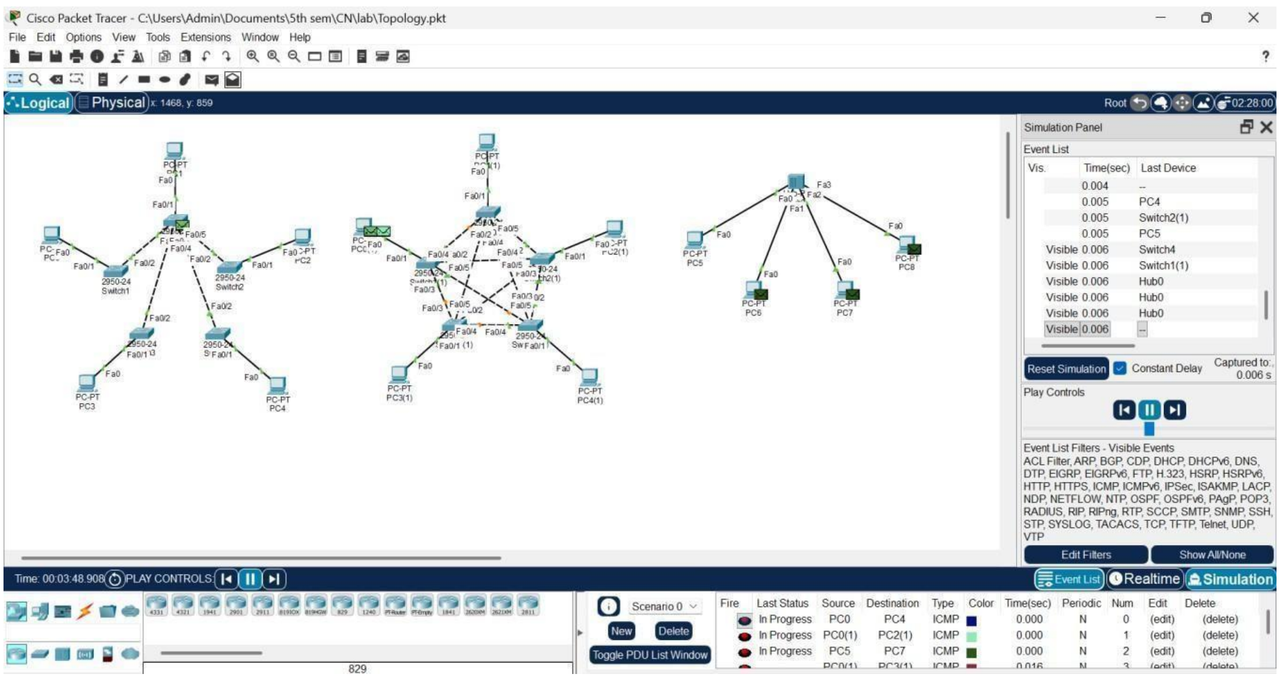
3. Create HUB-Based Topology

1. Drag and drop PCs (PC5, PC6, PC7, PC8, PC9).
2. From Hubs section, drag a Generic Hub (Hub0).
3. Connect each PC to the hub using Copper Straight-Through cable:
   - PC5 → Hub Fa0
   - PC6 → Hub Fa1
   - PC7 → Hub Fa2
   - PC8 → Hub Fa3
   - PC9 → Hub Fa4
4. Assign IP addresses within the same network for all PCs.
5. Use Simulation mode to send Simple PDU.
6. Observe broadcast behavior:
   - Hub sends the packet to all devices.

4. Demonstrate Ping Message (ICMP)

1. Switch to Simulation Mode from bottom-right corner.
2. Select the Simple PDU Tool (envelope icon).
3. Click on Source PC, then Destination PC.
4. Playback controls:
   - Play to observe step-by-step
   - Fast Forward for quick simulation
5. Watch the ICMP request and reply in the Event List window

Output:



Observation:

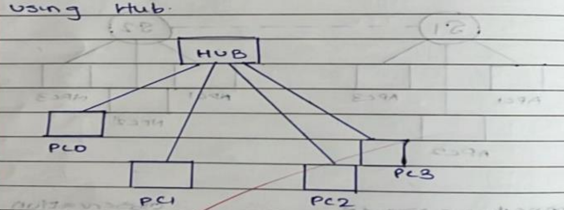
20/8/25

LAB 2

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

→ Components used
1. Generic Hub - PT
2. Generic Switch - PT
3. Generic PC-PT PC0 - 192.168.11.1
4. Generic PC-PT PC1 - 192.168.11.2
5. Generic PC-PT PC2 - 192.168.11.3
6. Generic PC-PT PC3 - 192.168.11.4

a) using Hub



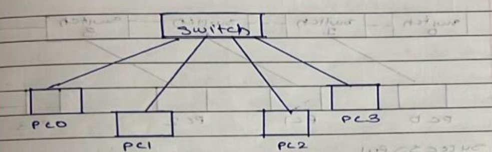
Connections made.

1. PDU PC0 to PC2

- Observation
- PDU send to hub which is broadcasted to all other devices
- PDU accepted by PC2 and rejected by other devices

- signal sent back to hub which is broadcasted to other devices
- PC0 accepts the signal rest rejects.

b) using switch.

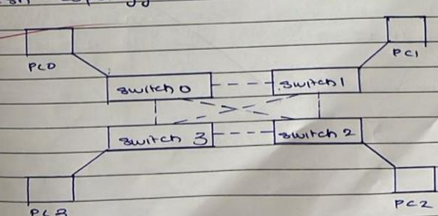


Connections made
- PDU from PC0 to PC2

Observation
- PDU sent from PC0 to switch
- switch sent PDU to PC2 only (and not broadcast)
- signal sent back to switch from PC2
- signal sent back to PC0

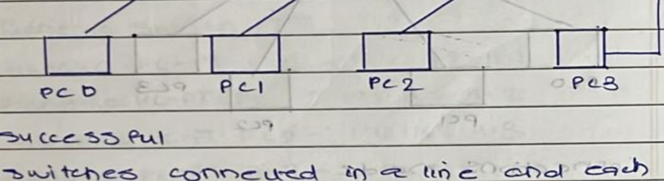
2) Topology

→ Mesh topology



→ Successful connection
- switches connected in a mesh manner along with end devices

→ Bus topology

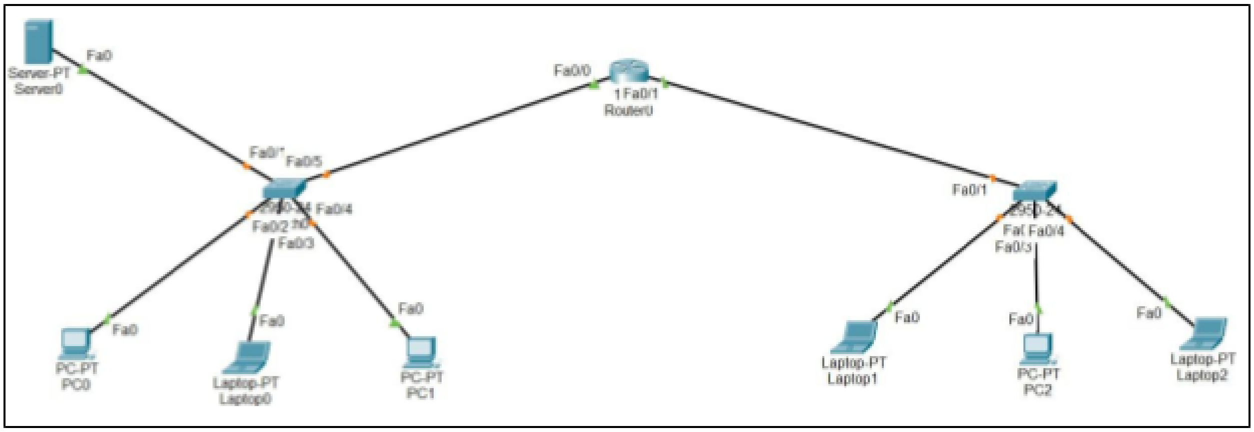


- successful
- switches connected in a line and each switch connected to an end device

Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Topology:



Procedure:

1. **Configure DHCP Server:**
   in DHCP server go to Desktop>IP-Config, assign static IP – 192.168.10.2 and gateway 192.168.10.1

2. **Open Services>DHCP and add following two dhcp pool:**

   (a) Pool Name: switch1

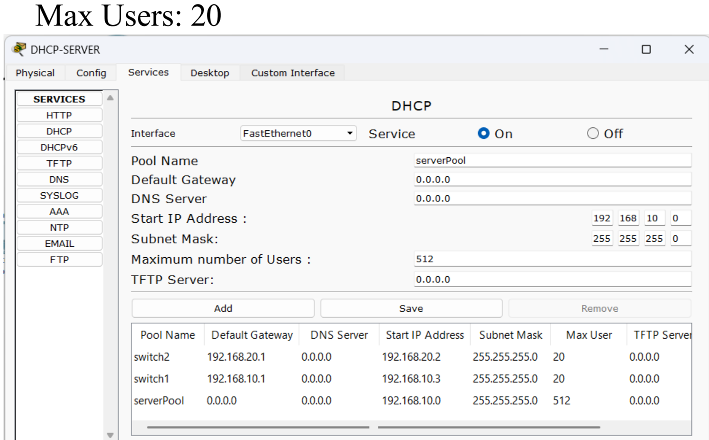
   Gateway: 192.168.10.1

   Start Ip: 192.168.10.3
   Subnet Mask: 255.255.255.0
   Max Users: 20

   (b) Pool Name: switch2

   Gateway: 192.168.20.1

   Start Ip: 192.168.10.2
   Subnet Mask: 255.255.255.0
   Max Users: 20



3. **Configure Router**

Router>enable

i. Router#configure terminal

(Within Lan)

ii. Router(config)# int fa0/0

iii. Router(config-if)# ip address 192.168.10.1 255.255.255.0

iv. Router(config-if)# ip helper-address 192.168.10.2

v. Router(config-if)# no shutdown

vi. Router(config-if)# exit

(Outside Lan)

vii. Router(config)# int fa0/1

viii. Router(config-if)# ip address 192.168.20.1 255.255.255.0

ix. Router(config-if)# ip helper-address 192.168.10.2

x. Router(config-if)# no shutdown

xi. Router(config-if)# exit

xii. Router(config)# exit

xiii. Router# write memory

Observation:

Bafna Gold

Date: Page:

LAB Experiment I:

~~what is the~~

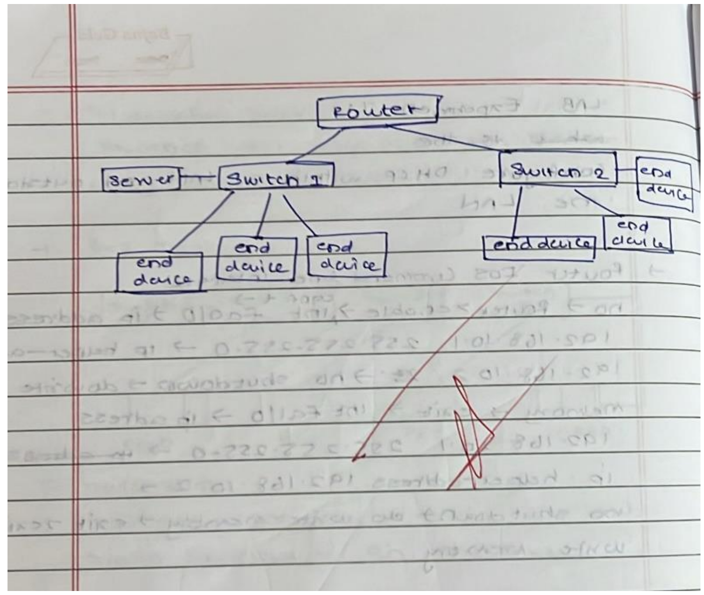
configure DHCP within a LAN and outside the LAN

→ Router IOS Command line interface

no → Router > enable → conf t → int fa0/0 → ip address 192.168.10.1 255.255.255.0 → ip helper-adaress 192.168.10.2 → no shutdown → do write memory → exit → int fa1/0 → ip adress 192.168.20.1 255.255.255.0 → ~~ip adress~~ ip helper-address 192.168.10.2 → no shutdown → do write memory → exit → exit → write memory.

→ Server

1. set service → DHCP → pool name = switchone
2. Default Gateway: 192.168.10.1
3. Start IP address: 192 168 10 3
4. Subnet mask: 255 255 255 0
5. Maximum number of users: 20
6. Service 'on'
7. add
8. repeat for switch two
9. Default Gateway: 192.168.20.1



Output:

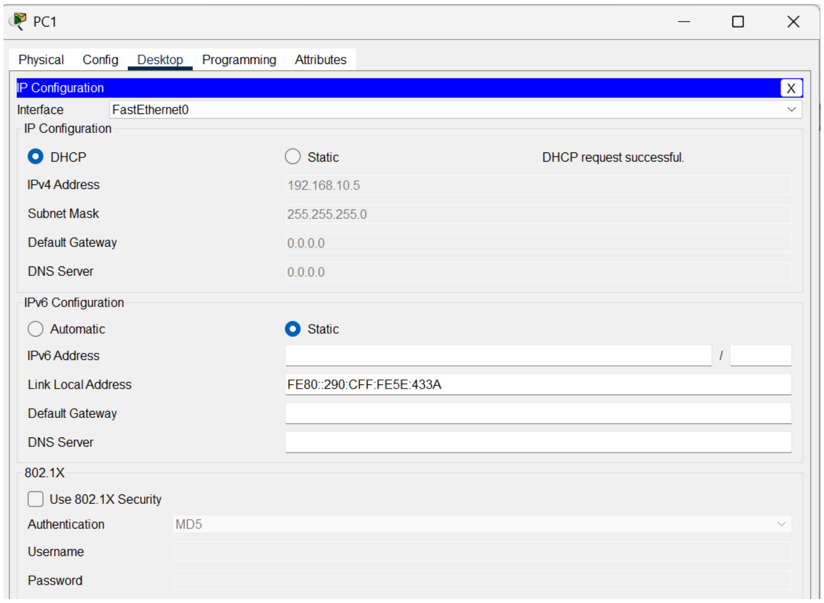


Fig 1. Ip address assigned by DHCP server within Lan (PC1)

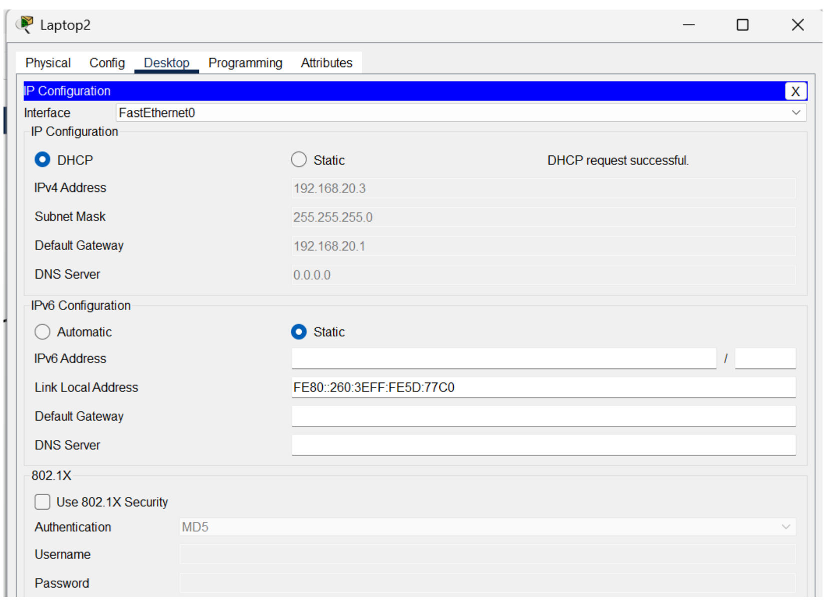
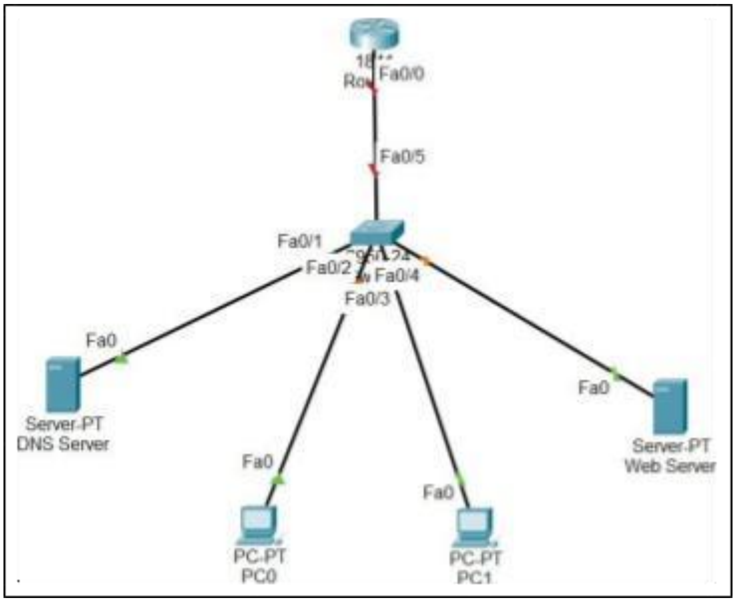


Fig 2. Ip address assigned by DHCP server outside Lan (laptop2)

Program 3:

Aim: Configure Web Server, DNS within a LAN.

Topology:



Procedure:

1. Create the Network

1. Place 1 Router, 1 Switch, 1 DNS Server, 1 Web Server, and two PCs.
2. Connect all devices using Copper Straight-Through cables.

2. Assign IP Addresses

1. On each device: Desktop → IP Configuration
   - Assign IPs in same network (e.g., 192.168.1.x).
   - Set Gateway = Router's interface IP.

3. Configure DNS Server

1. Open DNS Server → Services → DNS.
2. Turn DNS Service = On.
3. Add A-Record:
   - Name: letslearn.com
   - Address: IP of Web Server

4. Click Add → Save.

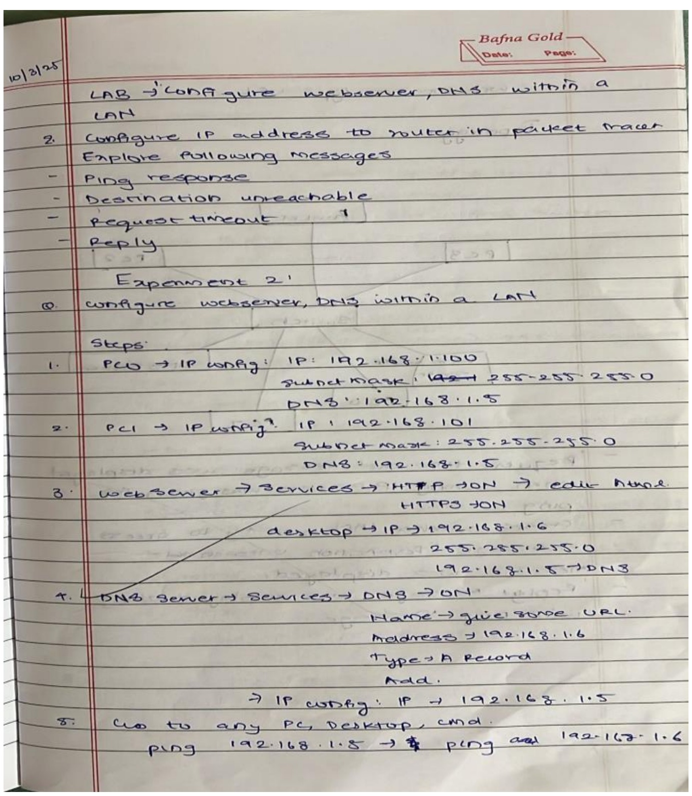
4. Configure Web Server

1. Open Web Server → Services → HTTP.
2. Turn HTTP = On (HTTPS optional).
3. Ensure index.html exists (default file is fine).
4. Edit HTML if needed.

5. Test from PC

1. Open PC → Desktop → Web Browser.
2. Enter URL:
3. http://www.letslearn.com/index.html
4. The webpage should load, confirming DNS + Web Server working.

Observation:



Bafna Gold
Date: Page:

10/3/25

LAB → Configure webserver, DNS within a LAN

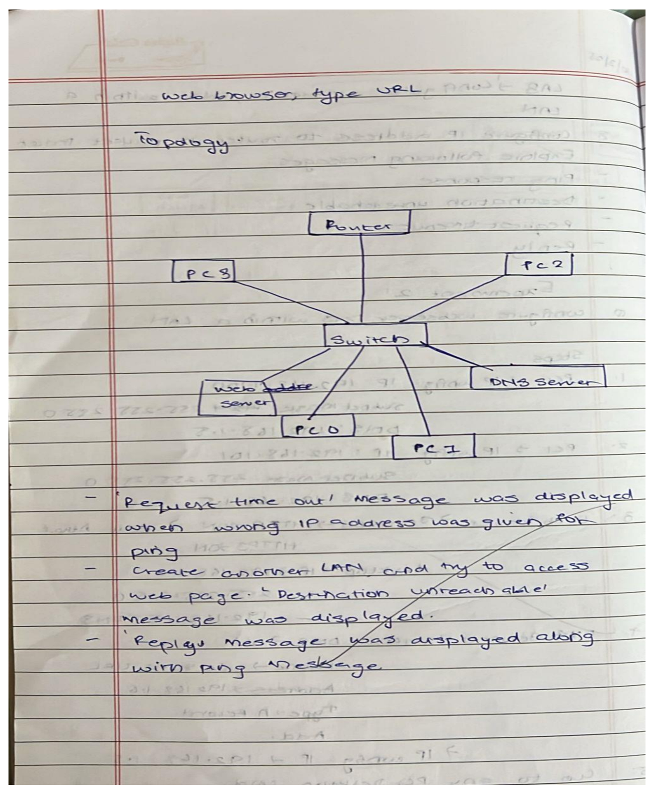
2. Configure IP address to router in packet tracer
Explore following messages
- Ping response
- Destination unreachable
- Request timeout
- Reply

Experiment 2:

Q. configure webserver, DNS within a LAN

Steps:

1. PC0 → IP config: IP: 192.168.1.100
subnet mask: ~~192~~ 255.255.255.0
DNS: 192.168.1.5
2. PC1 → IP config: IP: 192.168.1.101
Subnet mask: 255.255.255.0
DNS: 192.168.1.5
3. web server → services → HTTP → ON → edit html
HTTPS → ON
desktop → IP → 192.168.1.6
255.255.255.0
192.168.1.5 → DNS
4. DNS server → services → DNS → ON
Name → give some URL.
address → 192.168.1.6
Type → A Record
Add.
→ IP config: IP → 192.168.1.5
5. Go to any PC, Desktop, cmd.
ping 192.168.1.5 → ping ~~and~~ 192.168.1.6



web browser, type URL

Topology:

- 'Request time out' message was displayed when wrong IP address was given for ping
- Create another LAN and try to access web page. 'Destination unreachable' message was displayed.
- 'Reply' message was displayed along with ping message.

Output:

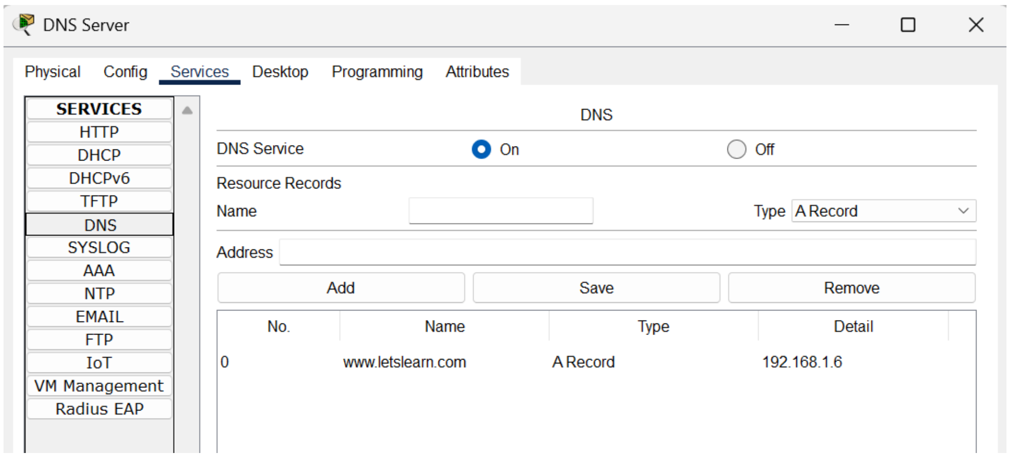


Fig 1. DNS server – DNS Services

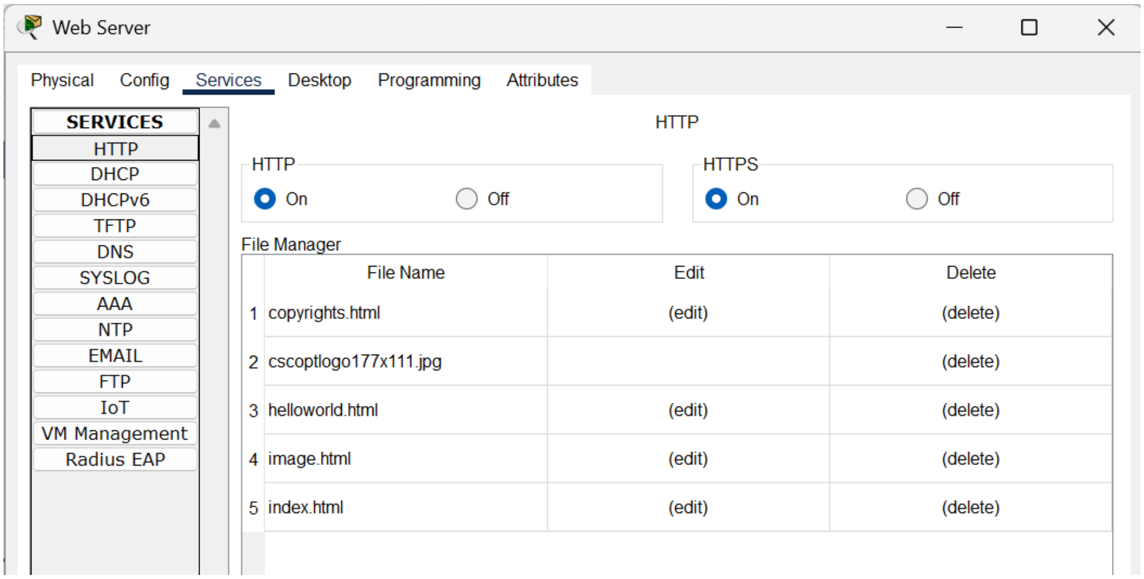


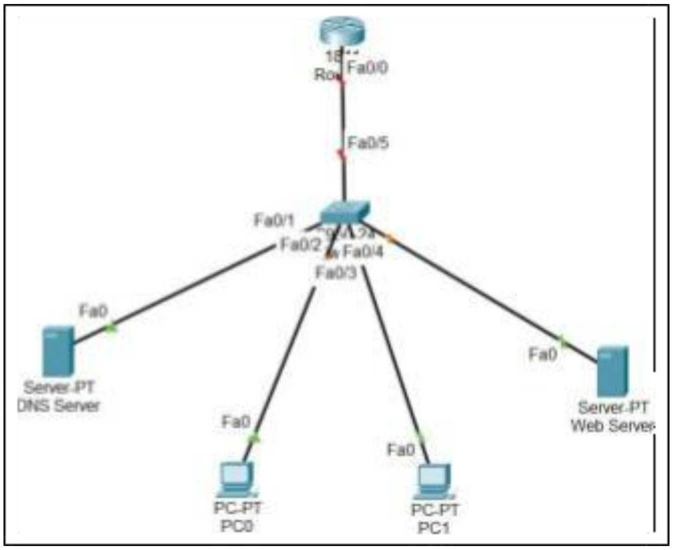
Fig 2. WEB server – HTTP Services

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.

Topology:



Procedure:

1. Assign IP Addresses to Router Interfaces

1. Click the Router → Config → Interfaces.
2. **Configure and enable:**
    - Fa0/0 → IP: 192.168.1.1 /24
    - Fa0/5 → IP: 192.168.2.1 /24
3. **Turn Port Status = On for each interface**.

2. Assign IP Addresses to PCs and Servers

1. On each device → Desktop → IP Configuration.
2. Use matching networks:
    - Devices connected to Fa0/0 → IP: 192.168.1.x, Gateway: 192.168.1.1
    - Devices connected to Fa0/5 → IP: 192.168.2.x, Gateway: 192.168.2.1

3. Verify Connectivity with Ping

1. Open PC → Desktop → Command Prompt.
2. Test different responses:

- Ping reply → reachable IP
- Request timed out → device powered off / link down
- Destination unreachable → wrong gateway or missing route

3. Observe the output for each case.

   Example commands:

   ping 192.168.1.2

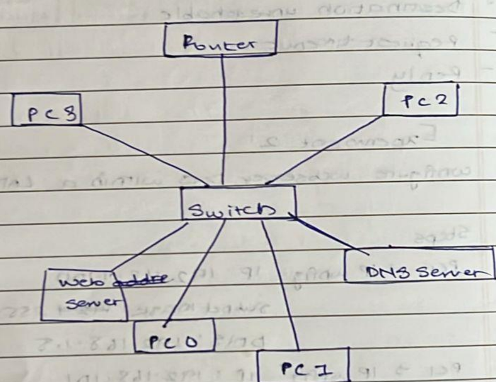
   ping 192.168.1.101

   ping 192.168.2.200

Observation:

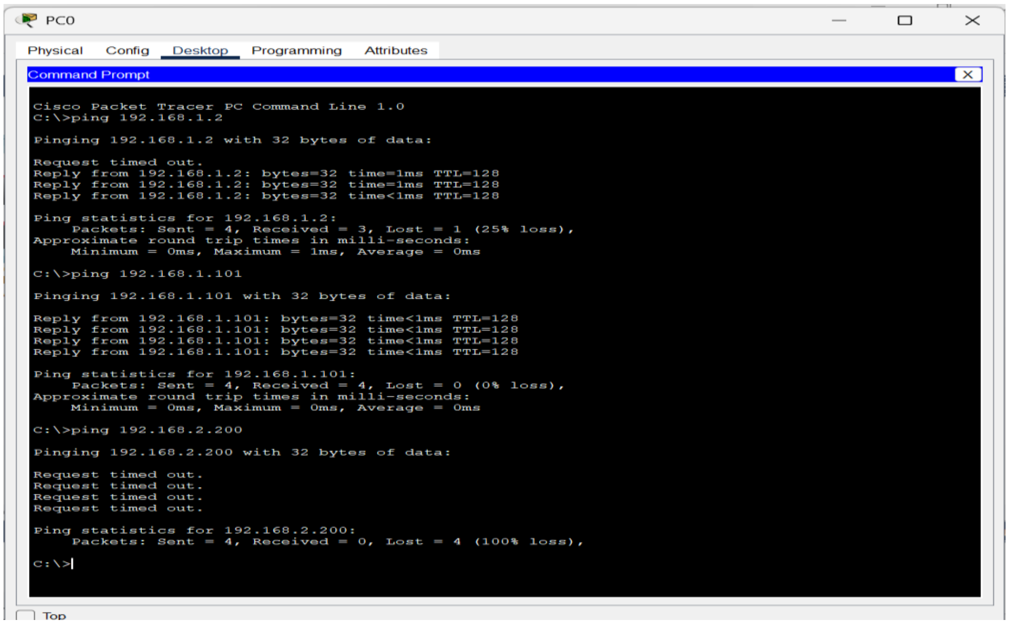
web browser, type URL

Topology



- 'Request time out' message was displayed when wrong IP address was given for ping
- Create another LAN and try to access web page. 'Destination unreachable' message was displayed.
- 'Reply' message was displayed along with ping message.

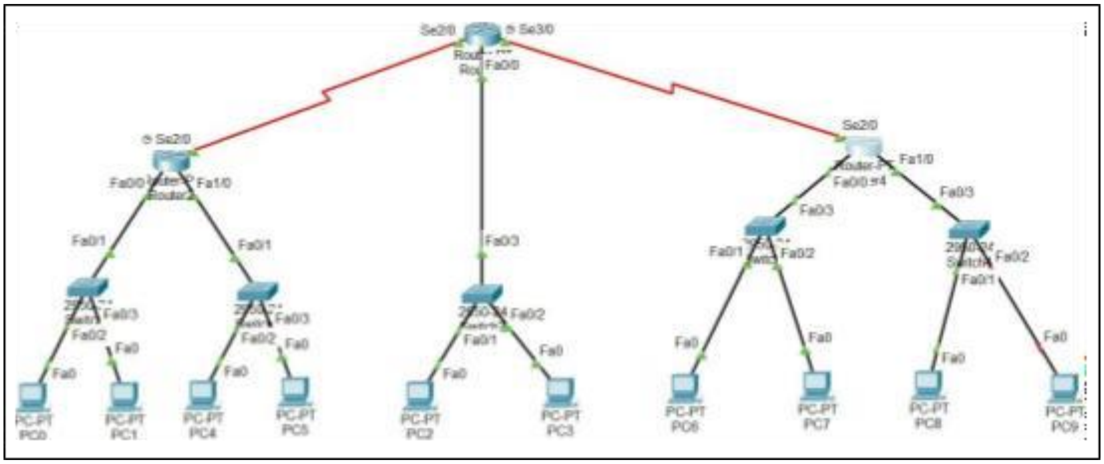
Output:


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
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
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
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:\>
Top

Program 5:

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

Topology:



Procedure:

1. Assign IP Addresses

1. On each router → Config → Interfaces
2. Set IP addresses for all FastEthernet and Serial interfaces according to the network diagram.
3. Turn Port Status = On for each interface.

2. Configure Static Routes

Perform on each router:

Router 2

1. Go to Config → Routing → Static
2. Add routes for networks behind Router 3 and Router 4:
   - Network: 192.168.3.0 /24 → Next Hop: 192.168.4.2
   - Network: 192.168.5.0 /24 → Next Hop: 192.168.4.2
   - Network: 192.168.6.0 /24 → Next Hop: 192.168.4.2
   - Network: 192.168.7.0 /24 → Next Hop: 192.168.4.2

Router 3

1. Go to Config → Routing → Static
2. Add routes toward Router 2 and Router 4:
   - 192.168.1.0 /24 → via 192.168.4.1
   - 192.168.2.0 /24 → via 192.168.4.1
   - 192.168.5.0 /24 → via 192.168.7.2
   - 192.168.6.0 /24 → via 192.168.7.2

Router 4

1. Go to Config → Routing → Static
2. Add routes toward Router 2 and Router 3:
   - 192.168.1.0 /24 → via 192.168.7.1

- 192.168.2.0 /24 → via 192.168.7.1
- 192.168.3.0 /24 → via 192.168.7.1
- 192.168.4.0 /24 → via 192.168.7.1

3. Configure Default Route (Optional)
If needed, add:
0.0.0.0 /0 → next-hop IP
(from each router toward the main/central router)

4. Test Connectivity

1. On any PC → Command Prompt
2. Use ping to reach devices in other networks.
3. Successful reply = routing configured correctly.

Observation

17/9/25

LAB→ Configure IPv4 static and default routing

1. Create topology
- Use normal connection for end device to switch and switch to router
- connect routers using serial DCE switch

2. Configure router 1
no
↳ enable
↳ conf t
↳ int se2/0
↳ ip address 172.16.1.1 (router address) 255.255.255.252
↳ no shutdown
↳ exit
↳ int fa0/0 (for switch)
↳ ip address 192.168.10.1 (IP address for switch) 255.255.255.0
↳ no shutdown
↳ ~~do~~ exit → write memory.

3. Repeat for R2 & R3.
R2: se 2/0 ip address 172.16.1.2 255.255.255.252
fa0/0 ip address 192.168.20.1 255.255.255.0
se3/0 ip address 172.16.2.1 255.255.255.252
3 interfaces

R3: Se 3/0 ip address 172.16.2.2 255.255.255.252
fa0/0 ↳ 192.168.30.1 255.255.255.0

4. Configure PC0, PC1, PC2
↳ ip address 192.168.20.10
default gateway 192.168.20.1
repeat for all 3

5. Configure router pain
R1 → conf t
ip route 192.168.20.0 255.255.255.0 172.16.1.2
ip route ~~192.16~~ 172.16.2.0 255.255.255.252 172.16.1.2
ip route 192.168.30.0 255.255.255.0 172.16.1.2
exit → do write memory

R2 → conf t
ip route 192.168.10.0 255.255.255.0 172.16.1.1
ip route 192.168.30.0 255.255.255.0 172.16.2.2
exit → do write memory

R3 → making it default
conf t
ip route 0.0.0.0 0.0.0.0
~~conf~~ exit → write memory

6. check if it is connected.
go to router → enable → show ip route
(must be connected, do for all 3)

{Do through static method}

Output:

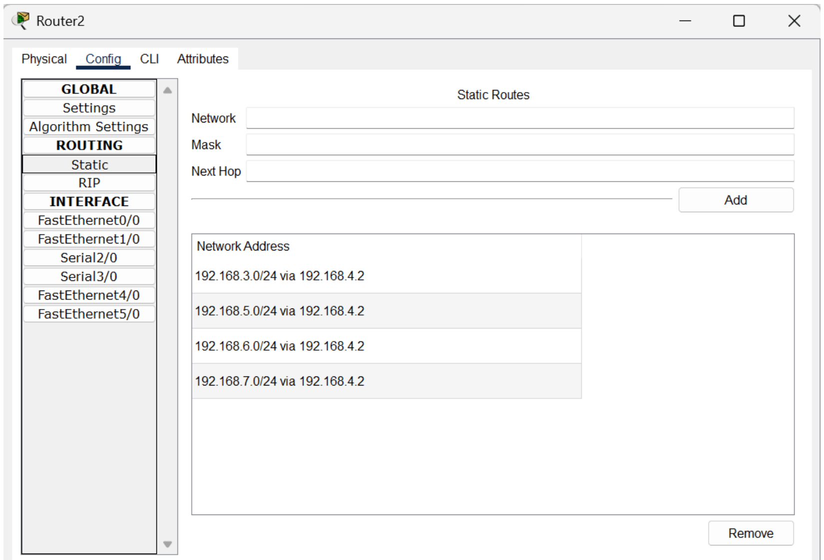


Fig 1. Router 2 – Static routing

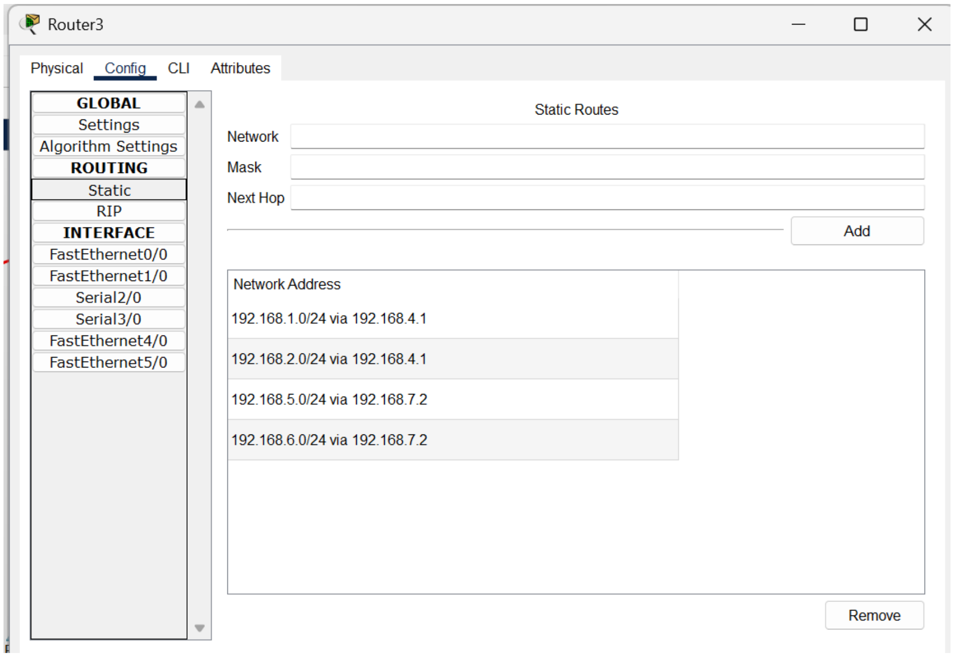


Fig 2. Router 3 – Static routing

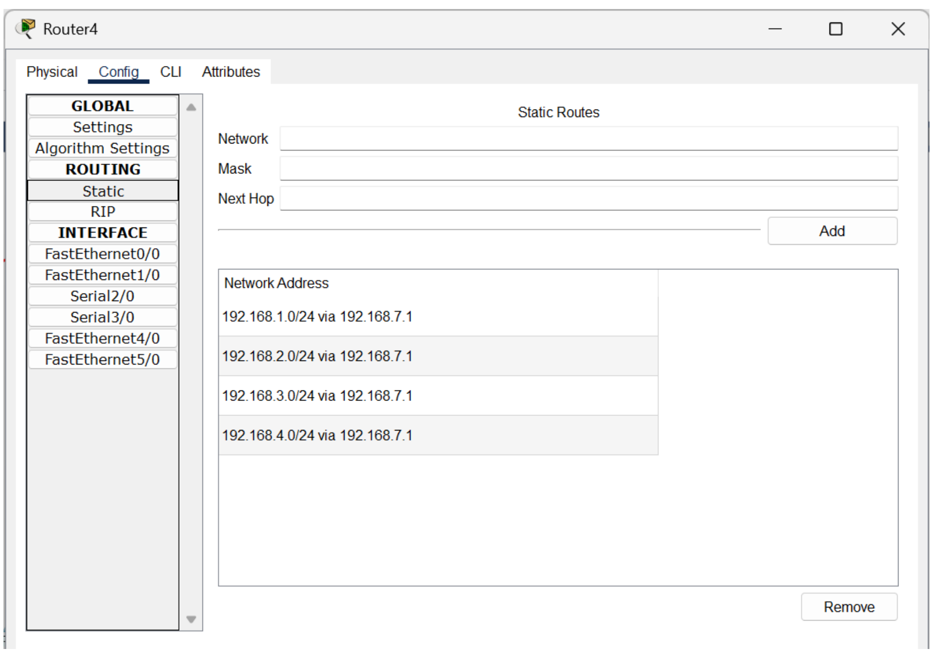
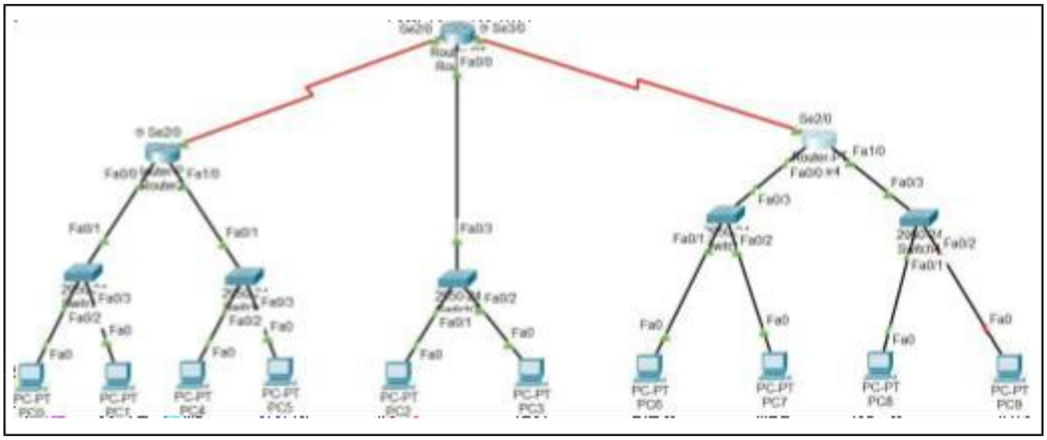


Fig 3. Router 4 – Static routing

Program 6:

Aim: Configure RIP routing Protocol in Routers.

Topology:



Procedure

1. Assign IP Addresses

1. On each router → Config → Interfaces
2. Configure IPs for all FastEthernet and Serial interfaces as per the network diagram.
3. Turn Port Status = On.

2. Enable RIP on Each Router

Router 2

1. Go to Config → Routing → RIP
2. Add directly connected networks:
    - 192.168.1.0
    - 192.168.2.0
    - 192.168.4.0

Router 3

1. Go to Config → Routing → RIP
2. Add networks:
    - 192.168.3.0
    - 192.168.4.0
    - 192.168.7.0

Router 4

1. Go to Config → Routing → RIP
2. Add networks:
    - 192.168.5.0
    - 192.168.6.0
    - 192.168.7.0

3. Verify Routing

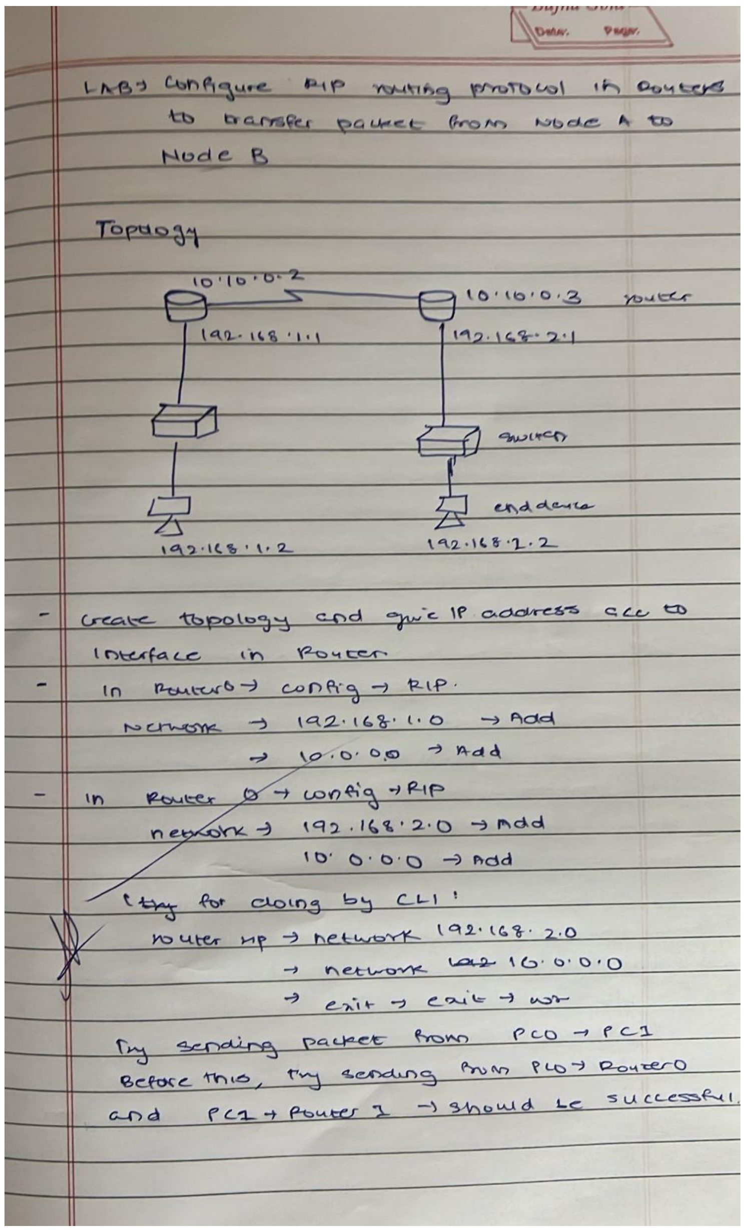
1. On any router → CLI
2. Use:
3. show ip route

→ RIP routes should appear with the letter R.

4. Test Connectivity

1. From PCs across different networks, use:
2. ping <destination IP>
3. Successful replies confirm RIP routing is working.

Observation



LAB 3 Configure RIP routing protocol in Routers to transfer packet from Node A to Node B

Topology

- Create topology and give IP address acc to interface in Router
- In Router0 → config → RIP.
  network → 192.168.1.0 → Add
  → 10.0.0.0 → Add
- In Router 0 → config → RIP
  network → 192.168.2.0 → Add
  10.0.0.0 → Add

(try for doing by CLI:
router rip → network 192.168.2.0
→ network ~~192~~ 10.0.0.0
→ exit → exit → wr

Try sending packet from PC0 → PC1
Before this, try sending from PC0 → Router0
and PC1 → Router 1 → should be successful.

Output:

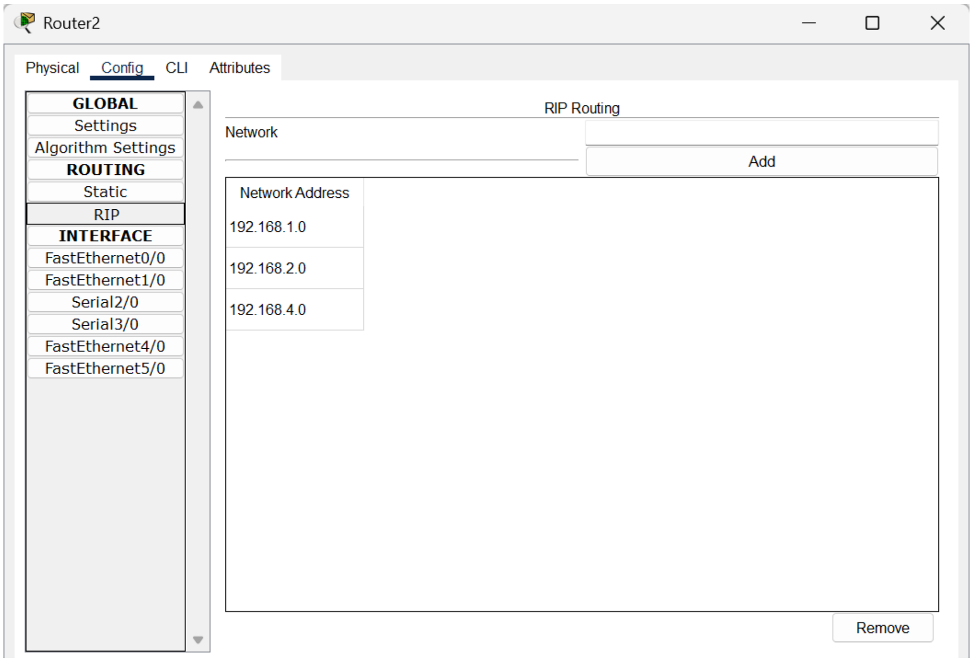


Fig 1. Router 2 – RIP routing

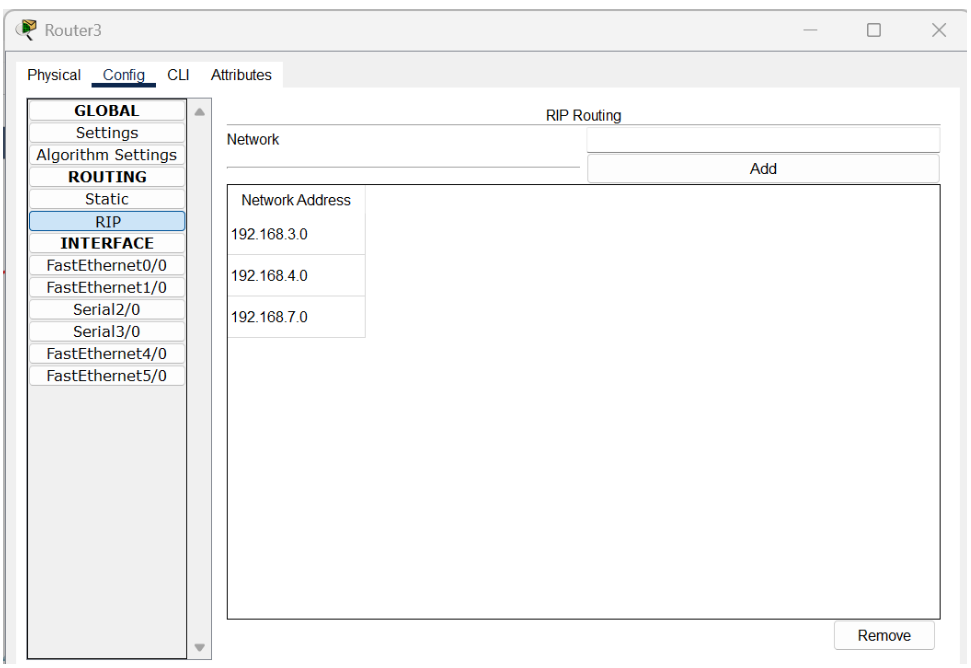


Fig 2. Router 3 – RIP routing

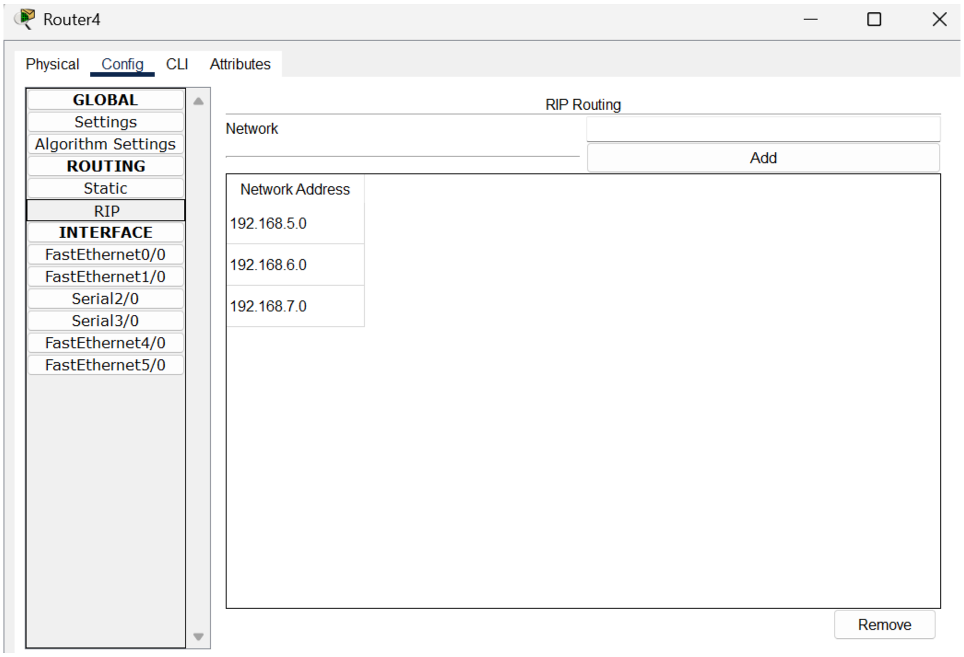
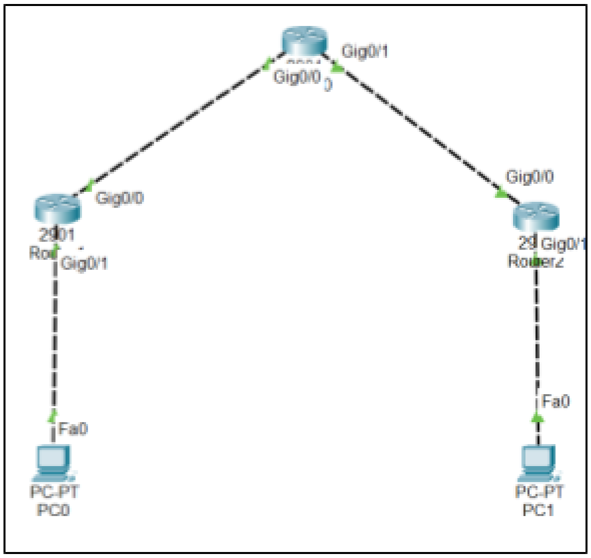


Fig 3. Router 4 – RIP routing

Program 7:

Aim: Configure OSPF routing protocol.

Topology:



Procedure:

1. Assign IP Addresses

1. On each router → Config → Interfaces
2. Assign IPs to Gig0/0, Gig0/1, and PC-facing interfaces as per the diagram.
3. Enable all interfaces (Port Status = On).

2. Configure OSPF on All Routers

Router 0

1. Go to Config → Routing → OSPF
2. Set Process ID = 1
3. Add networks:
   - 192.168.1.0 /24
   - 10.0.0.0 /30 (link to center router)

Router 1 (Center Router)

1. Process ID = 1
2. Add networks:
   - 10.0.0.0 /30 (left link)
   - 20.0.0.0 /30 (right link)

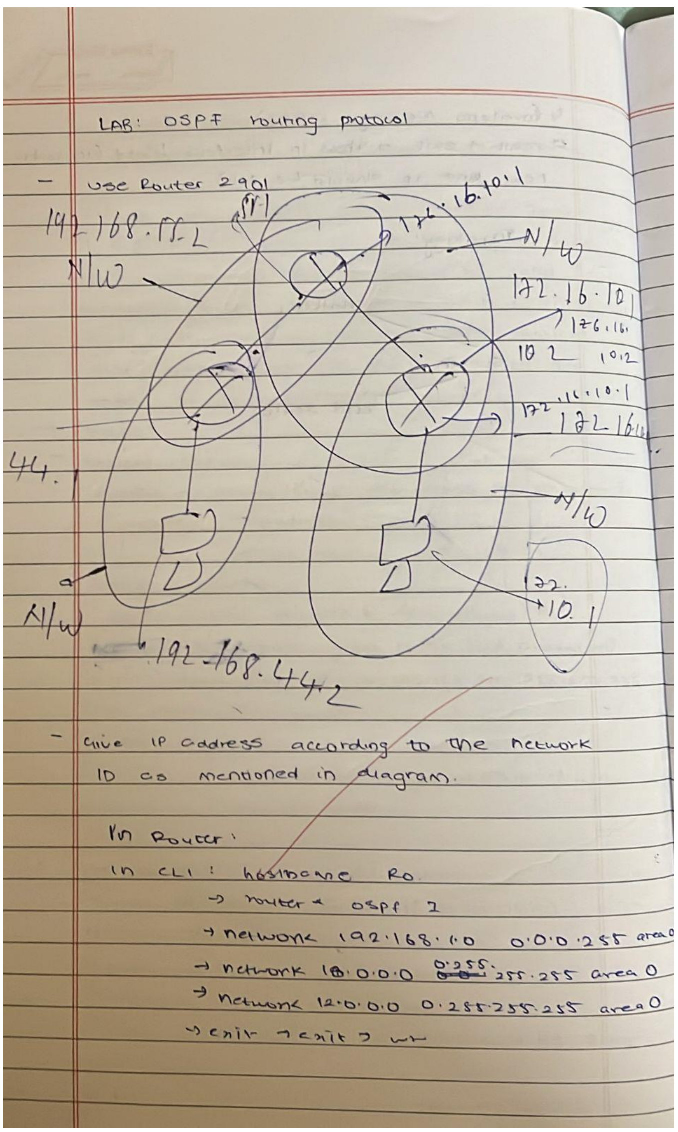
Router 2

1. Process ID = 1
2. Add networks:
   - 192.168.2.0 /24
   - 20.0.0.0 /30 (link to center router)

3. Test Connectivity

1. From PC0 → PC1, send PDU or use ping command.
2. Successful ICMP reply confirms OSPF is working.

3. Observation:


LAB: OSPF routing protocol

- use Router 2901

- Give IP address according to the network ID as mentioned in diagram.

In Router:

in CLI: hostname R0

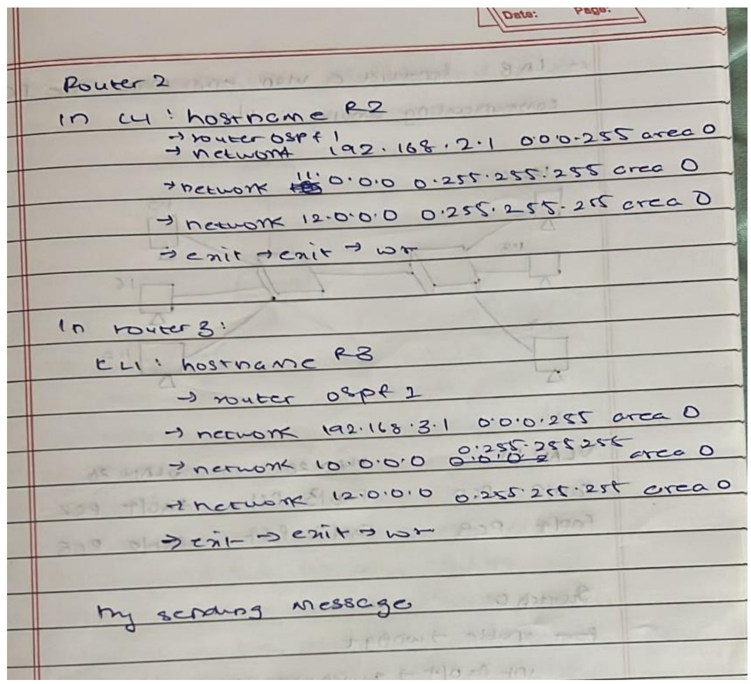
→ router ospf 1

→ network 192.168.1.0 0.0.0.255 area 0

→ network 10.0.0.0 0.255.255.255 area 0

→ network 12.0.0.0 0.255.255.255 area 0

→ exit → exit → wr


Router 2

in CLI: hostname R2

→ router ospf 1

→ network 192.168.2.1 0.0.0.255 area 0

→ network 11.0.0.0 0.255.255.255 area 0

→ network 12.0.0.0 0.255.255.255 area 0

→ exit → exit → wr

In router 3:

CLI: hostname R3

→ router ospf 1

→ network 192.168.3.1 0.0.0.255 area 0

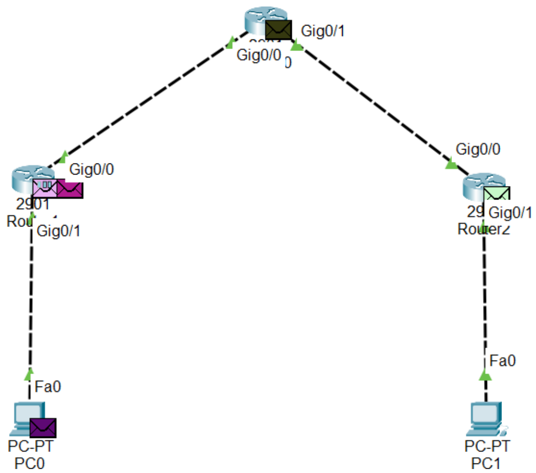
→ network 10.0.0.0 0.255.255.255 area 0

→ network 12.0.0.0 0.255.255.255 area 0

→ exit → exit → wr

my sending message

Output:



Gig0/1
Gig0/0
Gig0/0
Gig0/0
2901
Gig0/1
Gig0/1
Router2
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1

Fig 1. Sending PDU message from PC0 to PC1

PDU List Window

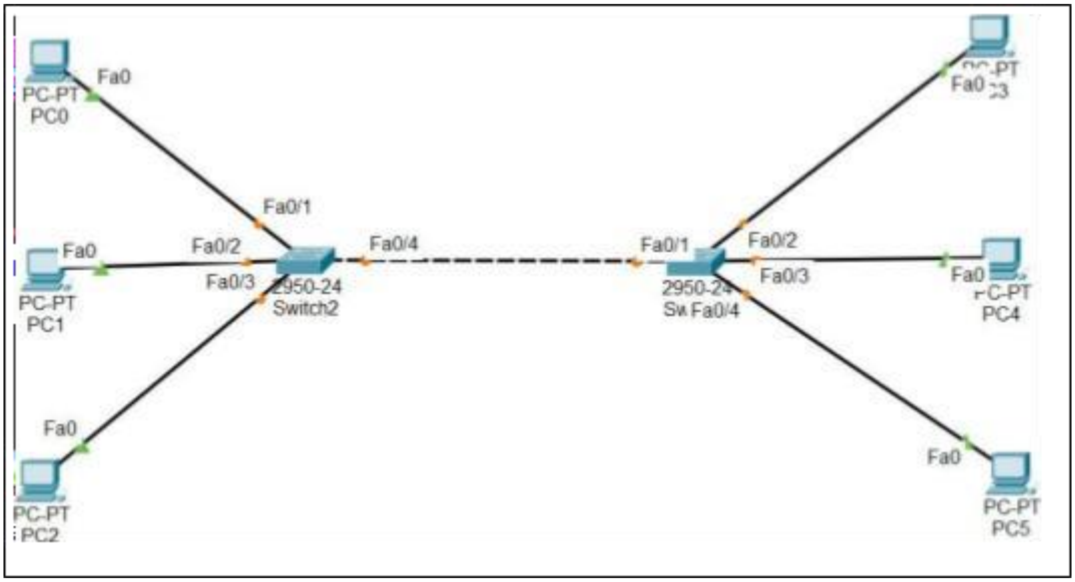
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Router2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	Router0	ICMP		0.000	N	2	(edit)	(delete)
	Successful	Router0	PC1	ICMP		0.000	N	3	(edit)	(delete)
	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.

Topology:



Procedure:

1. Create VLANs on Both Switches

1. Open each switch → Config → VLAN Database
2. Create VLANs (example):
   - VLAN 10
   - VLAN 20

2. Assign Ports to VLANs

Assign PCs to the required VLAN:

Switch 1 (Left Side)

- PC0 (Fa0/1) → VLAN 10
- PC1 (Fa0/2) → VLAN 10
- PC2 (Fa0/3) → VLAN 20

Switch 2 (Right Side)

- PC3 (Fa0/2) → VLAN 10
- PC4 (Fa0/3) → VLAN 10
- PC5 (Fa0/4) → VLAN 20

3. Configure Trunk Between Switches

1. Select the link between Fa0/4 (Switch1) ↔ Fa0/1 (Switch2)
2. On both ends → Config → Interface
3. Set Mode = Trunk
4. Allow VLANs 10 and 20 on the trunk.

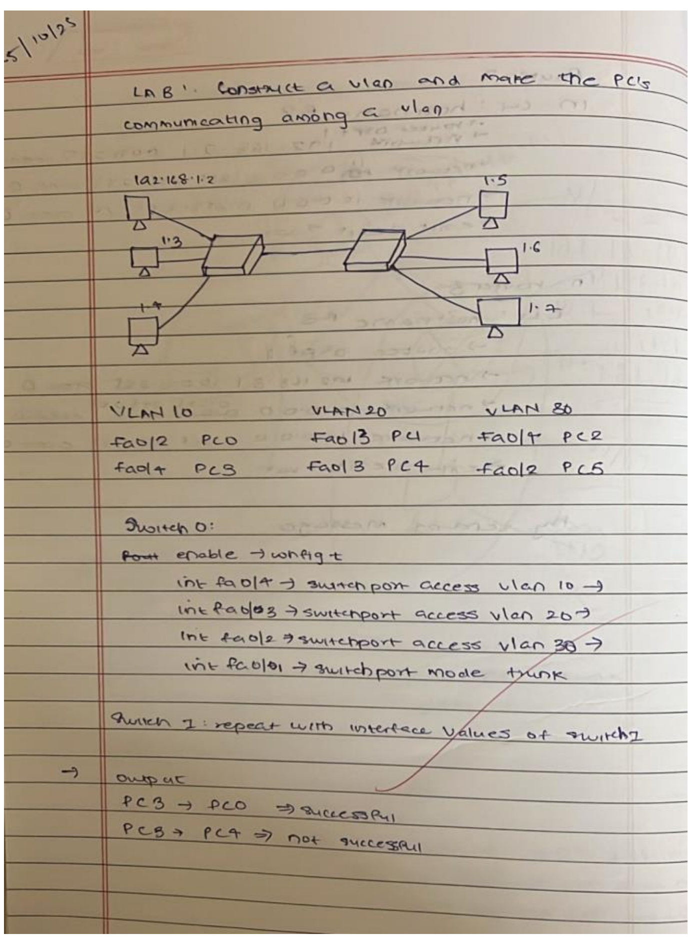
4. Assign IPs to PCs

1. On each PC → Desktop → IP Configuration
2. Assign IPs in VLAN-specific networks (example):
   - VLAN 10 → 192.168.10.x
   - VLAN 20 → 192.168.20.x

5. Test Connectivity

1. Use Add Simple PDU or ping:
   - Devices in the *same VLAN* should communicate.
   - Devices in *different VLANs* should not communicate.

Observation:



5/10/25

LAB : Construct a vlan and make the PC's communicating among a vlan

VLAN 10	VLAN 20	VLAN 30
fa0/2 PC0	Fa0/3 PC1	Fa0/4 PC2
fa0/4 PC3	Fa0/3 PC4	fa0/2 PC5

Switch 0:

~~Root~~ enable → config t

int fa0/4 → switchport access vlan 10 →

int fa0/3 → switchport access vlan 20 →

int fa0/2 → switchport access vlan 30 →

int fa0/1 → switchport mode trunk

Switch 1: repeat with interface values of switch1

→ output

PC3 → PC0 ⇒ successful

PC3 → PC4 ⇒ not successful

Output:

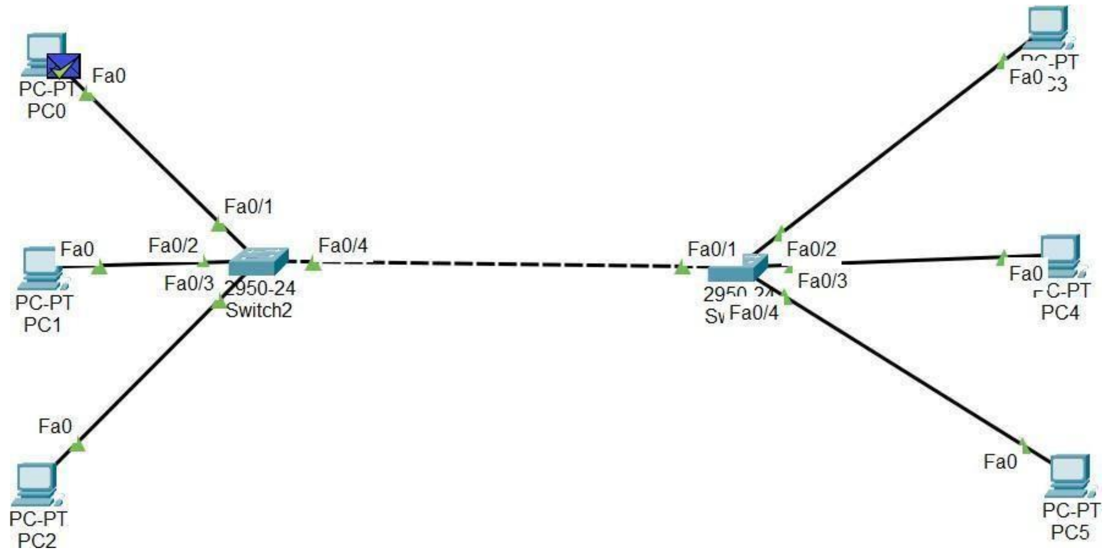


Fig 1. Sending PDU message from PC0 to PC5

PDU List Window

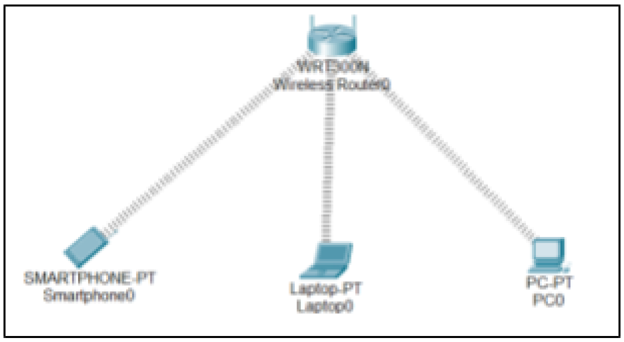
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC5	ICMP		0.000	N	2	(edit)	(delete)
	Successful	PC1	PC3	ICMP		0.000	N	3	(edit)	(delete)
	Successful	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

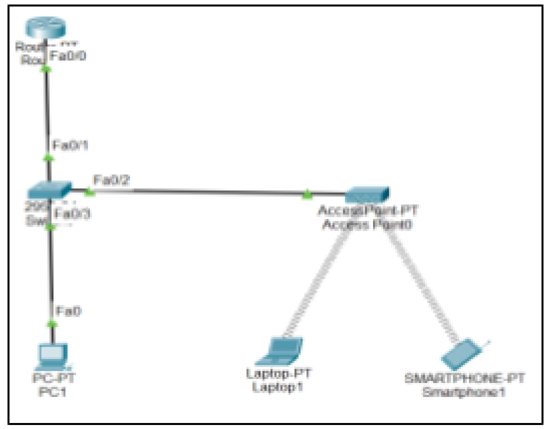
Program 9:

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

Topology:



Configuration 1



Configuration 2

Procedure:

1. Add Wireless Devices

1. Place Wireless Router, Access Point, Laptops, Smartphones, and PCs as shown.
2. For laptops/PCs without wireless modules →
    - Power off → Insert Wireless NIC → Power on.

2. Configure Wireless Router / Access Point

1. Click the Wireless Router / AP → Config → Wireless
2. Set:
    - SSID = BMSCE
    - Authentication = WPA2-PSK
    - Passphrase = bmsce123
3. Keep channel and encryption default.

3. Configure Wireless Settings on Laptop & Smartphone

1. Open device → Desktop → PC Wireless / Wi-Fi
2. Select SSID BMSCE
3. Enter password bmsce123
4. Connect.

4. Assign IP Addresses (if required)

1. Use DHCP (automatic) or manually assign from the same network.

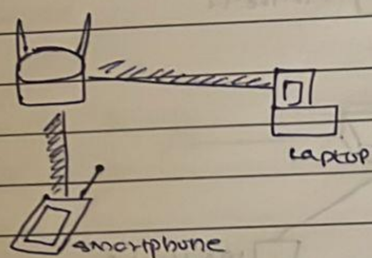
5. Test Wireless Communication

1. Use Add Simple PDU or ping between wireless devices.
2. Successful replies confirm WLAN communication.

Observation:

15/10/25

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



- Give a password to the wireless Router:

Steps:
config → wireless → SSID: bmsce → Authentication: WPA2-PSK
PSK Pass phrase: bmsce1234 (need to be minimum 8 characters) (All already connected devices will disconnect)

- Make laptop ready to connect to wireless:

Steps:
~~Turn off~~ Physical → remove the adaptor (drag and drop it to one of the modules) → take the wireless adaptor and place it

- Give password to laptop so it can connect:
config → wireless → SSID: bmsce → Authentication: WPA2-PSK PSK Pass phrase: bmsce1234

Laptop is connected, do it for the smartphone as well

Output:

1. Do Physical Connections In:

- Laptop
- PC

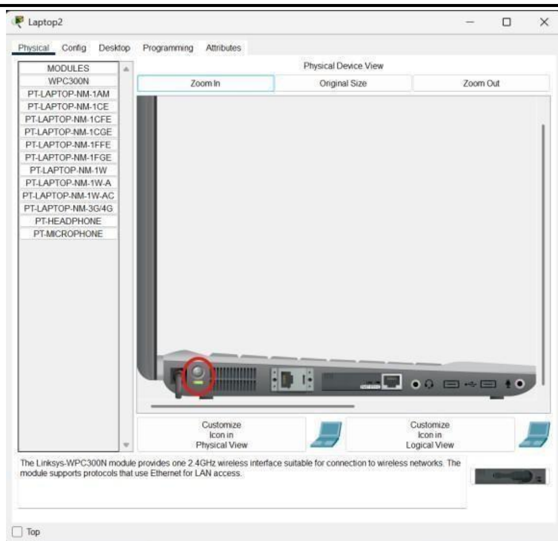


Fig 1.1 Step1: Turn off light / Power off laptop

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

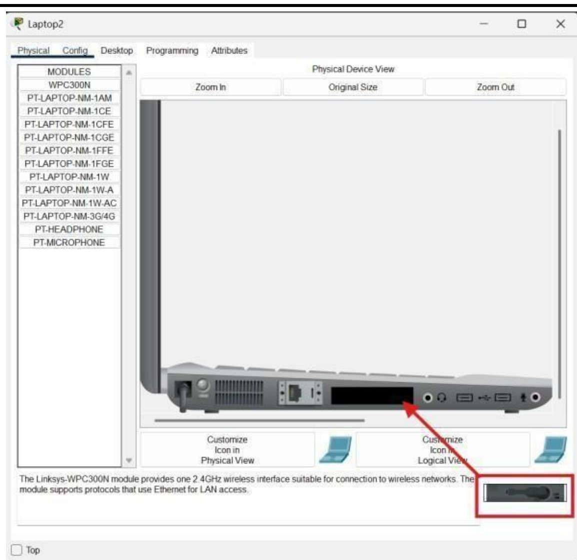


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

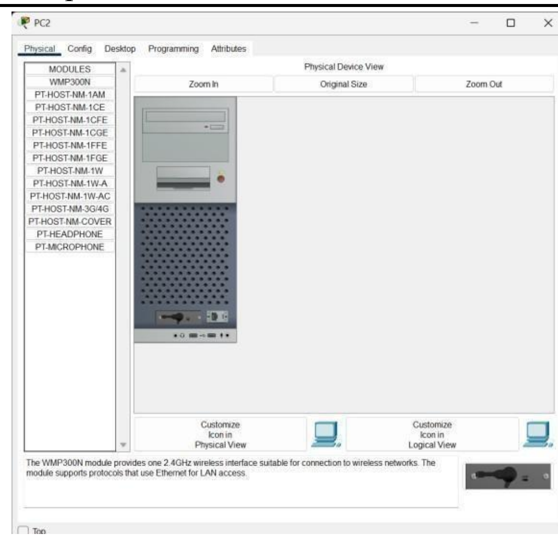


Fig 2. PC physical connection (combined 3 steps)

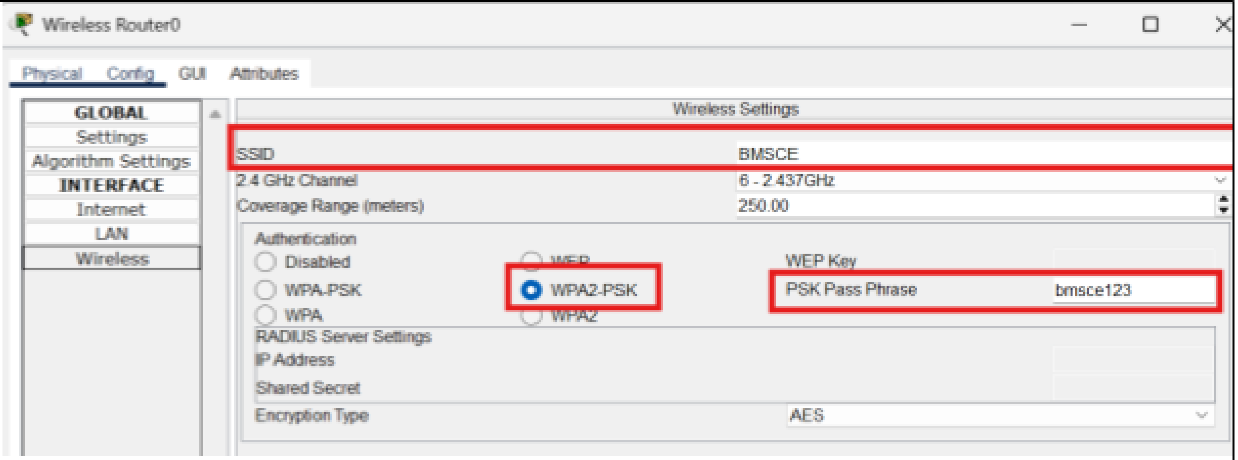


Fig 1. Config at Device Wireless Router0

2. Do Wireless Connection in:

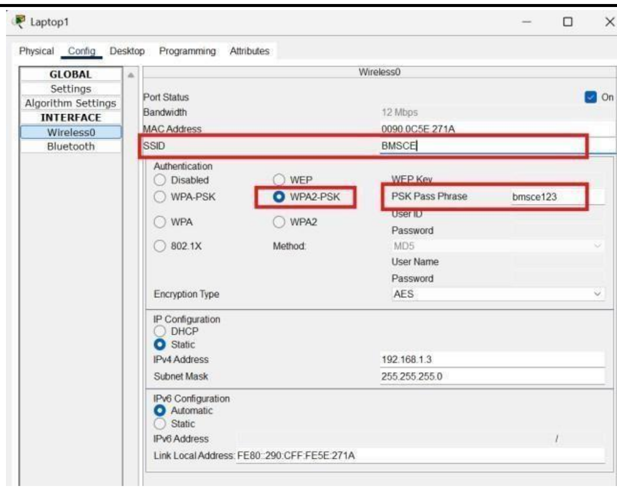


Fig 2. Config at Device Laptop0

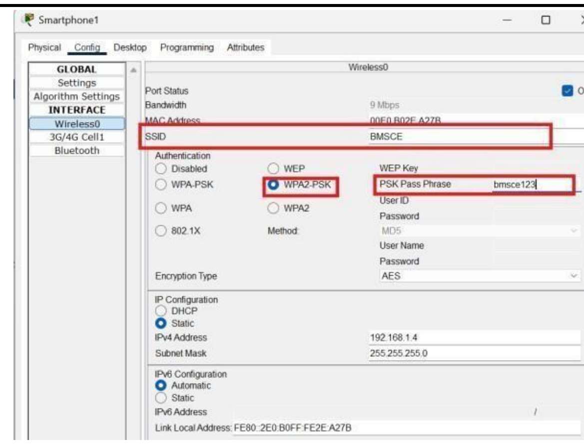


Fig 3. Config at Device Smartphone0

PDU List Window

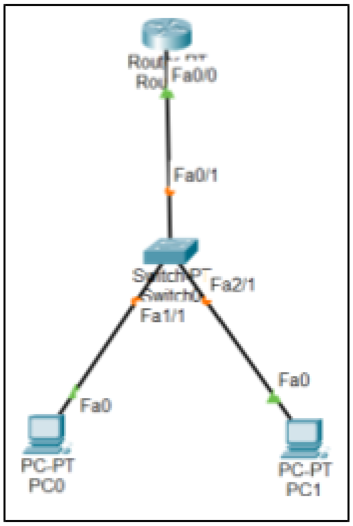
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Failed	Smar...	Laptop0	ICMP		0.000	N	0	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	1	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	2	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	3	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	4	(edit)
	Successful	Lapto...	Smartphone0	ICMP		0.000	N	5	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	6	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	7	(edit)
	Successful	Lapto...	PC1	ICMP		0.000	N	8	(edit)

Fig 3. Checking PDU messages

Program 10:

Aim: Demonstrate the TTL/ Life of a Packet.

Topology:



Procedure:

Create the Network

1. Place one Router, one Switch, and two PCs as shown in the topology.
2. Connect:
   - Router → Switch (Fa0/0 to Fa0/1)
   - Switch → PC0 (Fa1/1)
   - Switch → PC1 (Fa2/1)

2. Assign IP Addresses

1. On each PC → Desktop → IP Configuration
   - PC0: 192.168.1.2 /24
   - PC1: 192.168.1.3 /24
   - Gateway: 192.168.1.1
2. On Router → Config → Interface Fa0/0
   - IP: 192.168.1.1 /24
   - Turn Port Status = On

3. Switch to Simulation Mode

1. Click Simulation Mode (bottom right).
2. Select Add Simple PDU tool.

4. Send the Packet

1. Click PC0 → then click PC1 to send an ICMP (ping) PDU.
2. Observe packet movement step-by-step.

5. Check TTL (Time To Live)

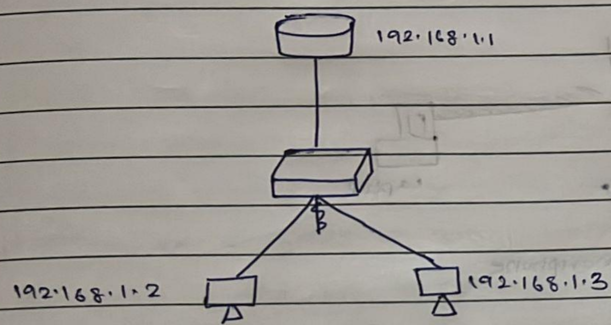
1. Click the PDU in the event list.
2. Open Inbound PDU Details and Outbound PDU Details.
3. Note the TTL value:
    - At source PC → TTL usually starts at 255
    - After passing Router → TTL reduces (example: 128)

6. Observe TTL Decrement

Each time a packet passes through a router, TTL decreases by 1, demonstrating the packet's lifespan on the network.

Observation:

LAB: Demonstrate the ~~TTL~~/ Life of a packet



- Give default gateway for end devices as 192.168.1.1
- Send a message from PC0 to PC1
- create simulation
- click on the message icon at PC0 and PC1
- The outbound from PC0 show TTL 255
- Inbound at PC1 ~~from~~ show TTL as 255
- Outbound at PC1 show TTL as 128

Output:

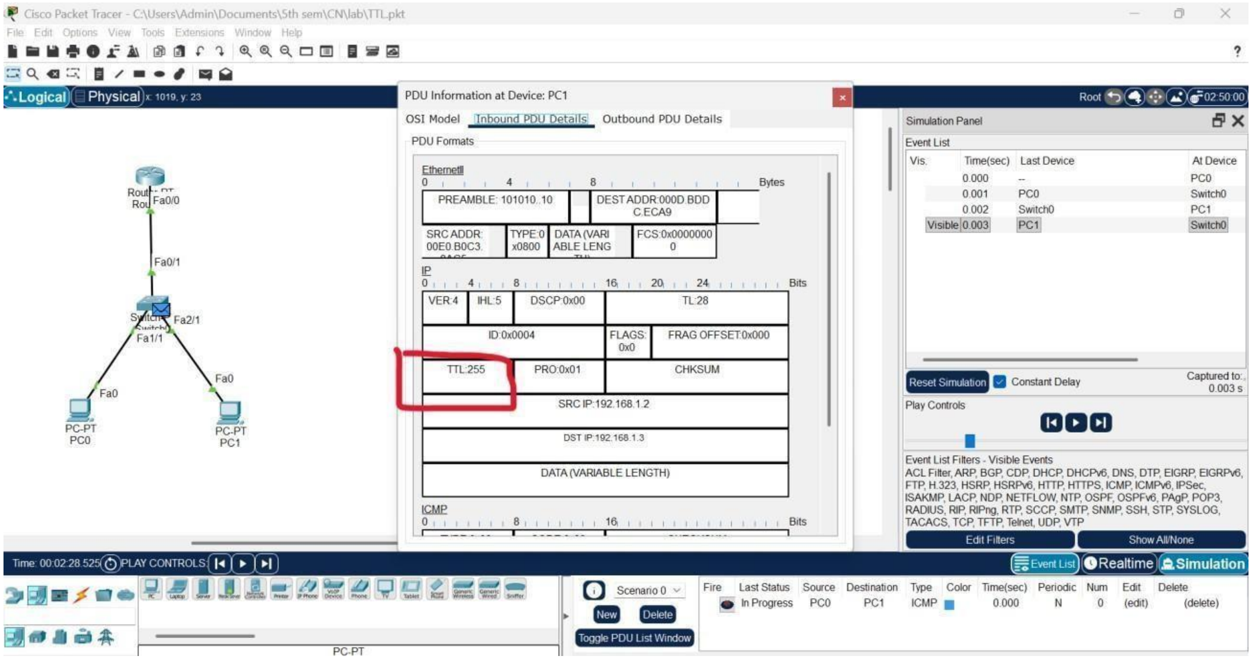


Fig 1. Inbound PDU Details at Device PC1

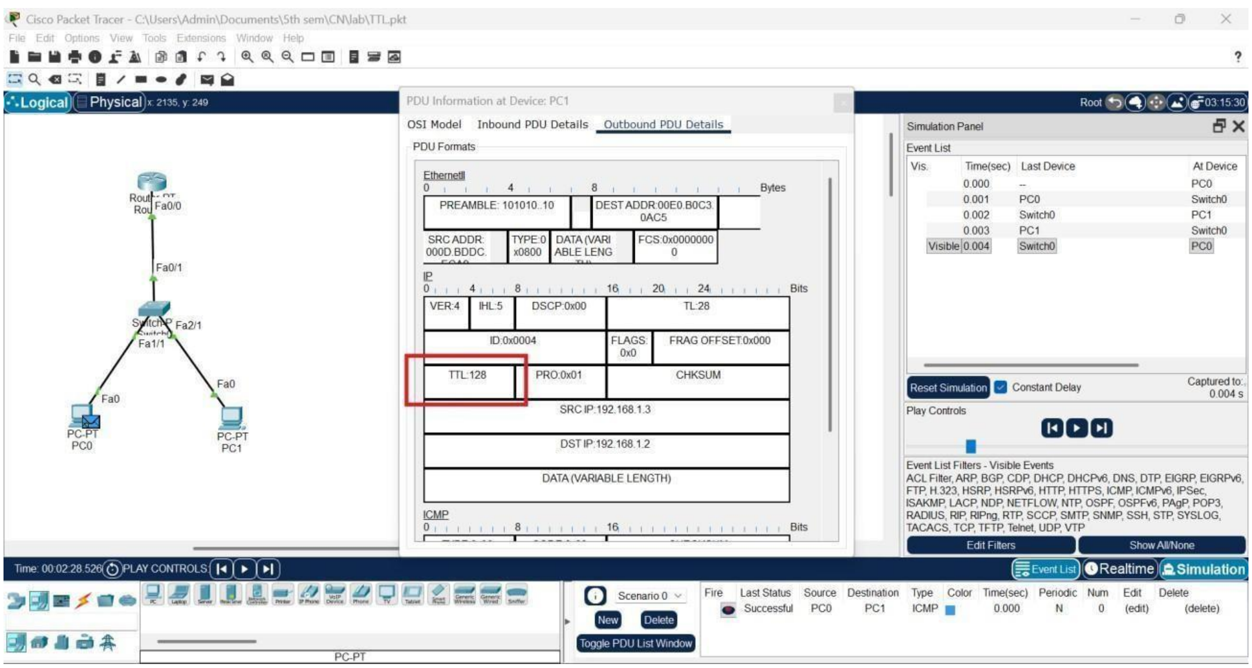
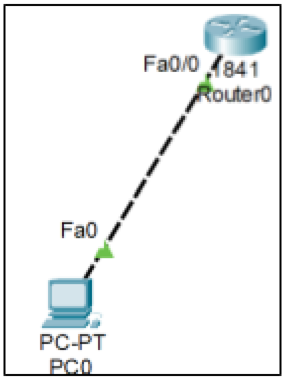


Fig 1. Outbound PDU Details at Device PC1

Program 11:

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

Topology:



Procedure:

Procedure

1. Configure the Router for Telnet

1. Open Router0 → CLI and enter:
2. enable
3. configure terminal
4. hostname R1
5. line vty 0 4
6. login
7. password cisco
8. enable secret tp
9. interface fa0/0
10. ip address 192.168.1.1 255.255.255.0
11. no shutdown
12. exit
13. end
14. Verify interface status:
15. show ip interface brief

2. Assign IP to PC

1. On PC0 → Desktop → IP Configuration:
   - IP Address: 192.168.1.2
   - Subnet Mask: 255.255.255.0
   - Gateway: 192.168.1.1

3. Test Connectivity

1. On PC0 → Command Prompt, ping the router:
2. ping 192.168.1.1

4. Access Router Using Telnet

1. On PC0 → Command Prompt:
2. telnet 192.168.1.1
3. Enter password: cisco to log in.
4. You now have remote access to the router.

5. Verify Telnet Access

1. Execute any router command remotely, e.g.:

   show ip interface brief

Observation:

8/10/25

LAB - Telnet

Configure telnet to access router remotely

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

→ steps:

- Create topology.
- ~~Configure router~~ Give IP address for desktop → IP config. IP address: 192.168.1.2
  Default gateway: 192.168.1.1
- Configure router
  ↳ no → enable → conf t → hostname R1
  ↳ enable secret rp (this is the 2nd password)
  ↳ int fa0/0 → ip address 192.168.1.1 255.255.255.0
  ↳ no shutdown
  ↳ line vty 0 5
  ↳ login → password tp → exit → exit → wr
  ↳ show ip interface brief

- Go to desktop, command prompt
  ↳ ping 192.168.1.1 (check if successful)
  ↳ telnet 192.168.1.1
  ↳ password is tp (won't show)
  ↳ enable → password is rp
  ↳ show ip ~~brr~~ interface brief (should be same as router)
  ↳ conf t → int fa0/0 → ip add 192.168.1.2
  255.255.255.0

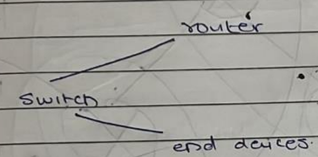
Bafna Gold

Date: Page:

↳ (overlaps message comes)

↳ exit → exit → show ip interface brief (in router, new ~~[illegible]~~ ip should be set)

Topology:



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)#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

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

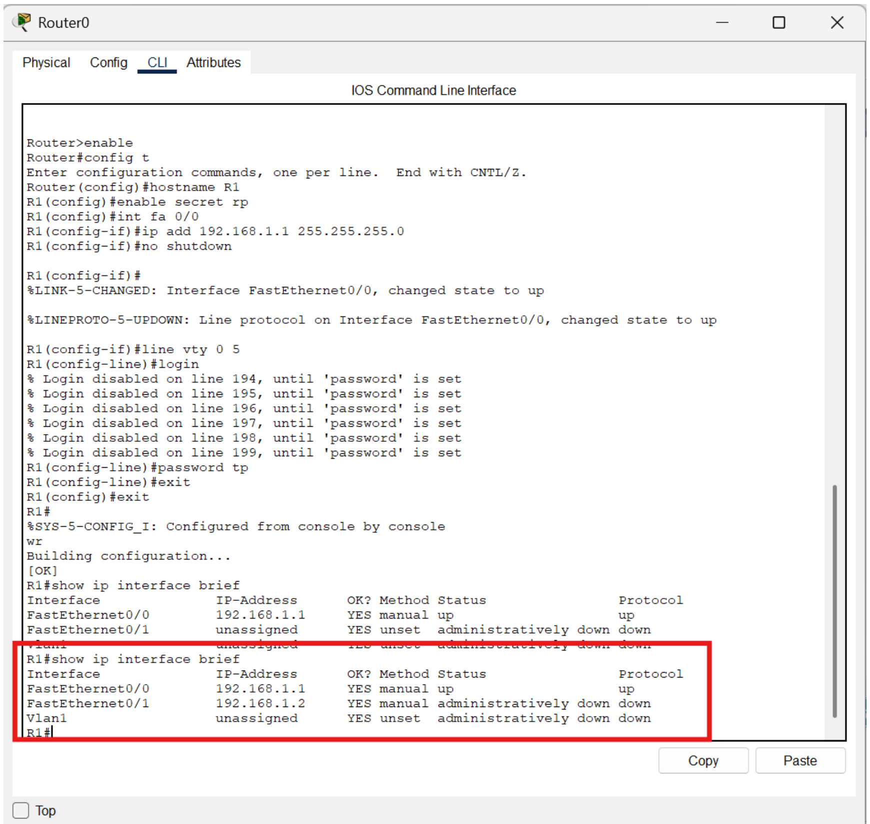


Fig 3. Updated the changes into Router0

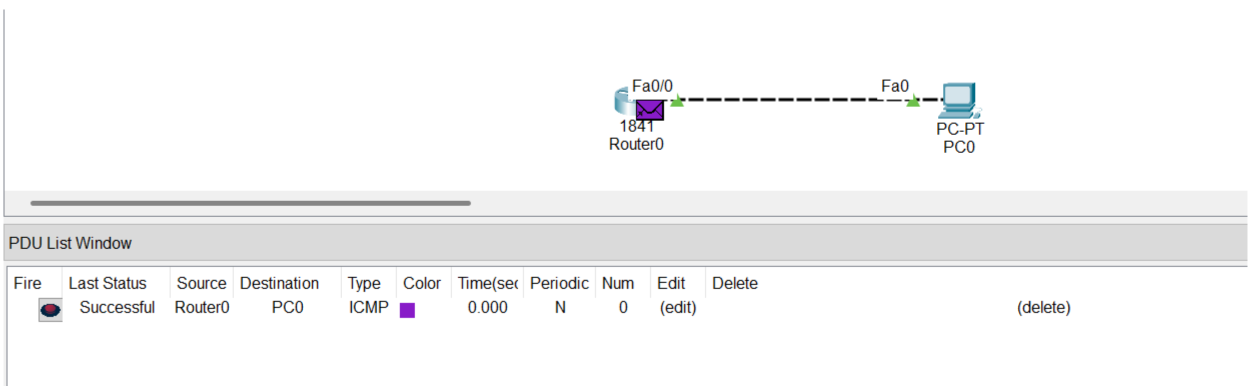
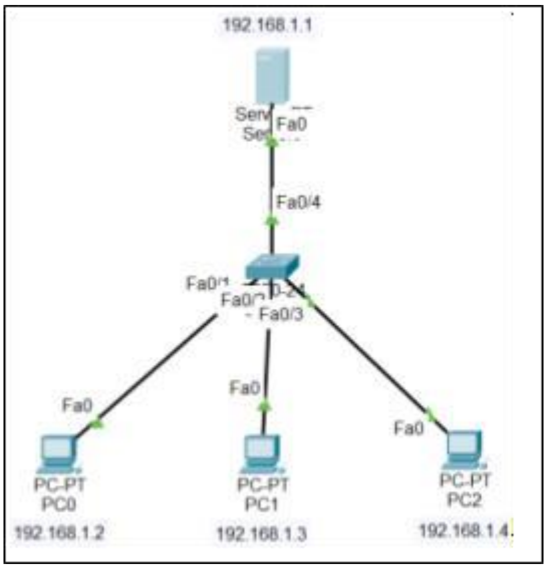


Fig 4. PDU message Successful

Program 12:

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

Topology:



Procedure:

1. Create the LAN

1. Place one server, one switch, and three PCs as shown.
2. Connect all devices to the switch using straight-through cables.

2. Assign IP Addresses

1. On each PC and the Server → Desktop → IP Configuration
    - Server: 192.168.1.1
    - PC0: 192.168.1.2
    - PC1: 192.168.1.3
    - PC2: 192.168.1.4
    - Subnet Mask: 255.255.255.0
    - Gateway: (none needed for LAN)

3. Check ARP Table (Before Communication)

1. On each device → Command Prompt
2. Type:
3. arp -a
4. The ARP table will be empty initially.

4. Generate Traffic (Ping)

1. On PC0 → Command Prompt:
2. ping 192.168.1.1

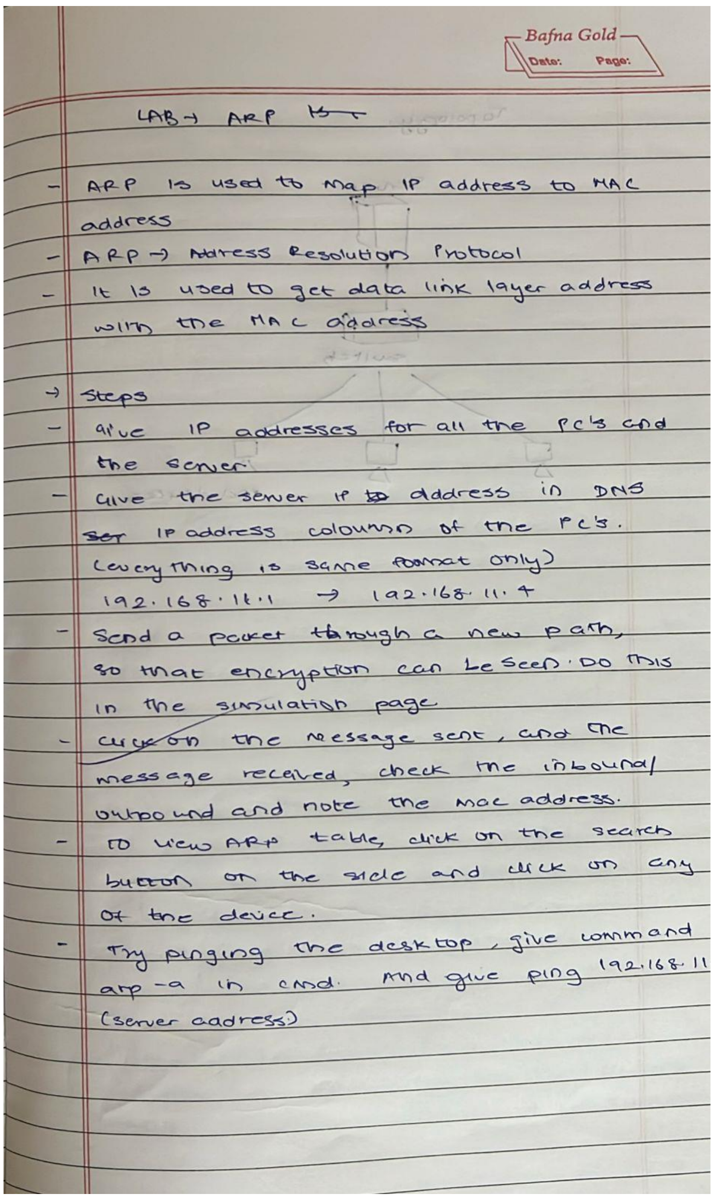
3. PC0 sends an ARP request → switch → server.
4. Server replies with its MAC address.

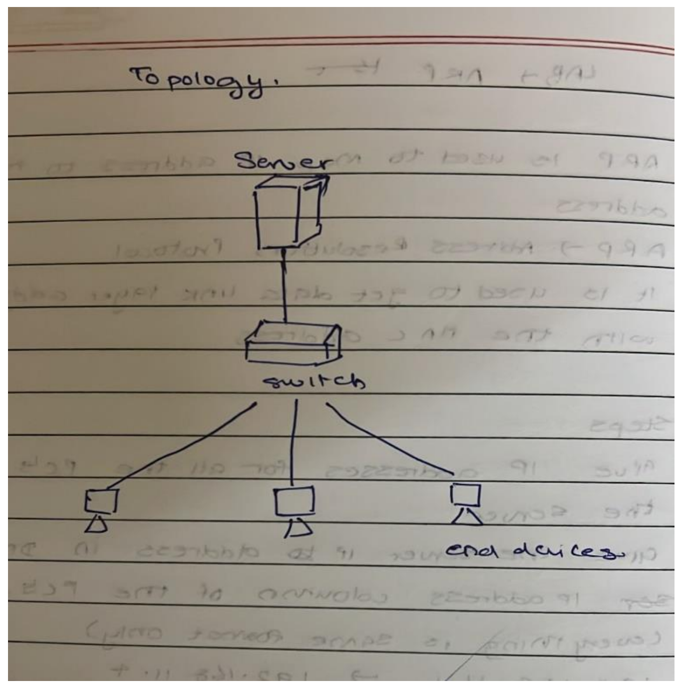
5. Check ARP Table (After Communication)

1. On each device, again run:
2. arp -a
3. Entries now appear showing:
    - IP Address
    - MAC Address
    - Interface

This demonstrates how ARP resolves IP → MAC mapping.

Observation:





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

Server0

Physical Config Services Desktop Programming Attributes

Command Prompt

```
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

PC0

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 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

PC1

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 3.2 Command Prompt at PC1

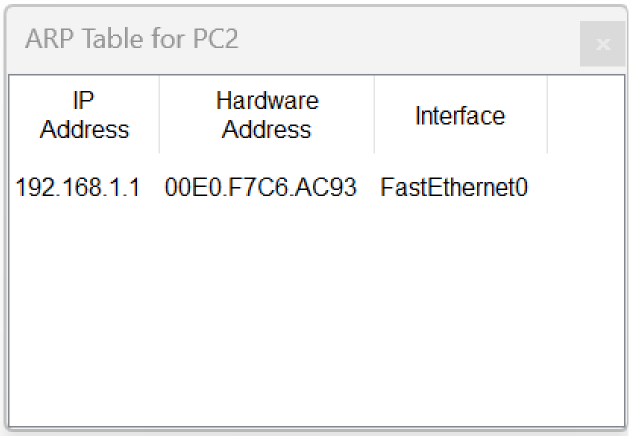


Fig 4.1 ARP table at PC2

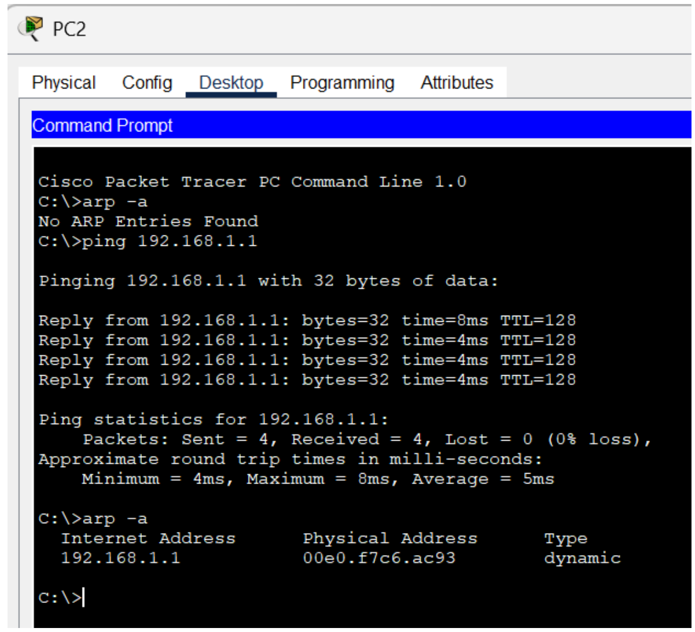


Fig 4.2 Command Prompt at PC2

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");

    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
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:

09/10/25

LAB: Leaky Bucket simulation

Algorithm:

1. Start
2. Input the following values:
   Bucket capacity → cap
   Processing rate → process
   Number of seconds to simulate → nsec
3. For each second i=1 to nsec:
   Input the number of packets arriving at second i → inp[i]
4. Initialise variables:
   count=0(packets currently in bucket)
   drop=0(packets dropped)
5. For each second i=1 to nsec:
   - Add incoming packets: count = count + inp[i]
   - if count > cap:
     drop= count-cap
     count=cap
   Compute packets to send: mini=min(count, process)
   - send packets: count=count-mini
   Print second number, packets received =0, packets sent(mini), Packets left(count), Packets dropped (drop)
   5- Reset drop=0
6. After all input seconds are processed, while count != 0:
   if count > cap:
     drop = count-cap
     count= cap
   Compute packets to send: mini=min(count, process)
   Send packets: count=count-mini
   Print the prev mentioned values
   Reset drop=0
7. Stop the program.

→ Output

Enter the bucket size: 5
Enter the processing rate: 2
Enter the number of seconds you want to simulate: 3
Enter the size of packet entering at 1 sec: 5
Enter the size of packet entering at 2 sec: 4
Enter the 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

<u>Program 2:</u>

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        pradeep-g@Pradeep-G: ~/Documents/TCP
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        pradeep-g@Pradeep-G: ~/Documents/TCP
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:

LAB: Write a program in C/C++ for client Server communication using TCP or IP sockets to make client send the name of the file and 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 = read(stdio, buffer, sizeof(buffer))
4. Write a file name to the socket using write(sfd, buffer, n)
5. Read file contents from the socket by m = read(sfd, buffer1, sizeof(buffer1))
6. Display file contents to standard output by write(stdio, buffer1, m)
7. Go to step 5 if m > 0
8. close socket by close(sfd)

→ Algorithm (Server Side)

- Create a socket with the socket() system call
- Bind the socket to ~~the~~ an address using the bind(sfd,..) system call. If not sure of machine IP address, keep the structure member s-addr to INADDR-ANY. Assign a port number between 3000 and 5000 to sin-port

Bafna Gold Date: Page:

- Listen for connections with the listen(sfd,...) system call. ~~If not sure of machine IP address, Keep the structure member s-addr to INADDR-ANY. Assign a port~~
- sfd = Accept a connection with the accept(sfd,...) system call. This call typically blocks until a client connects with the server
- Read the filename from the socket by n = read(sfd, buffer, size of (buffer))
- Open the file by fd = open(buffer)
- Read the contents of the file by m = read(fd, buffer1 size of (buffer1))
- Write the file content to socket by write(sfd, buffer1, m)
- Go to step 4 if m > 0
- close (sfd)

→ Output

$ cc socketserver.c$
$./a.out 1025
server:
waiting for connection
server received: /home/aps/cse.text
server: /home/aps/cse.text found
opening and reading --
reading --
.. reading complete
transfer complete

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.

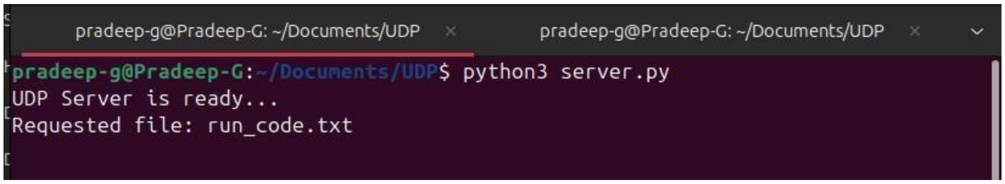
Code:

```python
# udp_client.py

import socket


# Step 1: Create UDP socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)


server_address = ('localhost', 8081)


filename = input("Enter filename to request: ")


# Step 2: Send filename to server

client_socket.sendto(filename.encode(), server_address)


# Step 3: Receive response

data, addr = client_socket.recvfrom(4096)


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

print(data.decode())


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

```python
# udp_server.py

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"File not found on server.", addr)
```

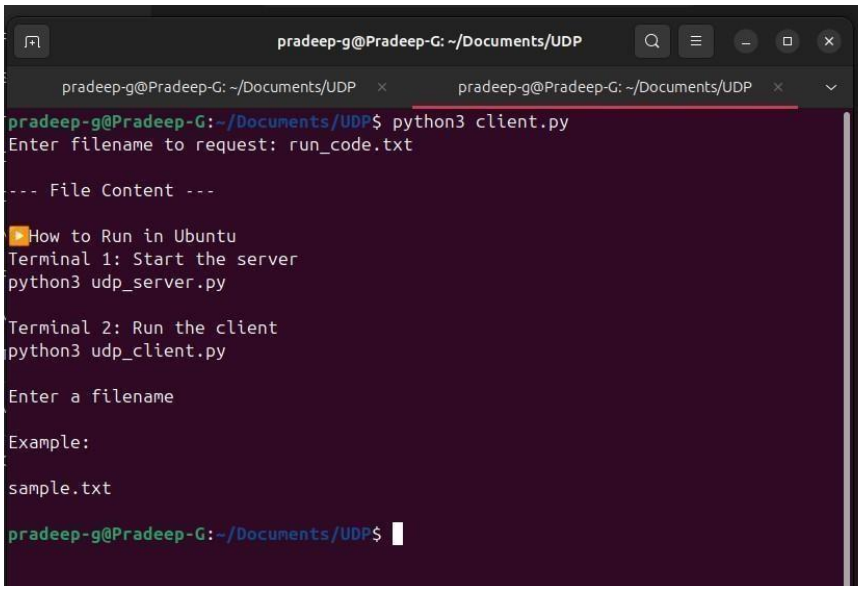
Output:

Server side Terminal:



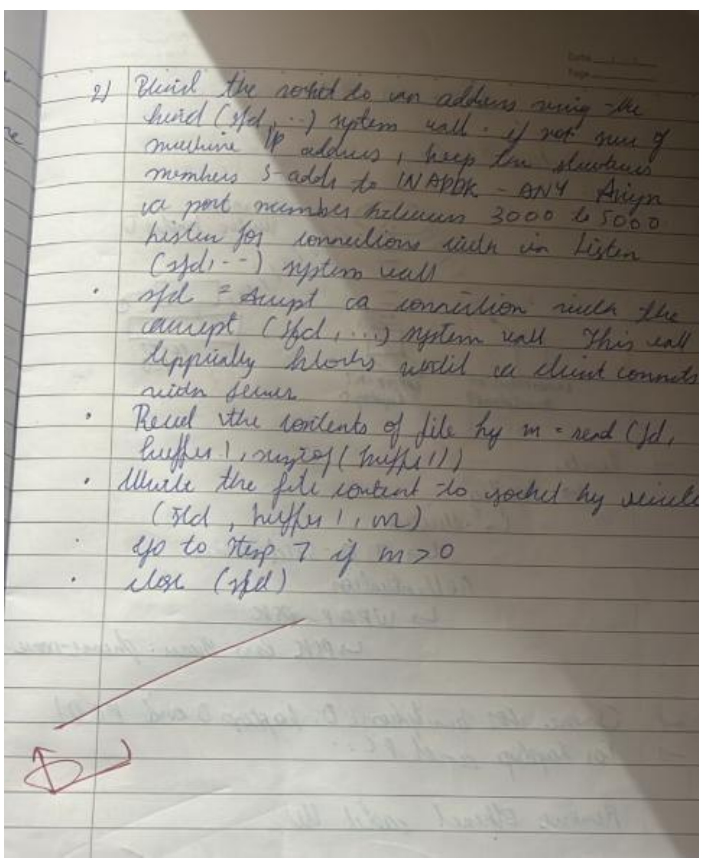
```
pradeep-g@Pradeep-G:~/Documents/UDP$ python3 server.py
UDP Server is ready...
Requested file: run_code.txt
```

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:



2) Build the socket to an address using the bind (sfd, ...) system call. if not sure of machine IP address, keep the structure members s-addr to INADDR_ANY Assign a port number between 3000 to 5000
listen for connections with in Listen (sfd, --) system call
• sfd = Accept a connection with the accept (sfd, ...) system call This call typically blocks until a client connects with server
• Read the contents of file by n = read (fd, buffer, sizeof(buffer))
• Write the file content to socket by send (sfd, buffer, n)
• go to step 7 if n > 0
• close (sfd)

Program 4:

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

Code:

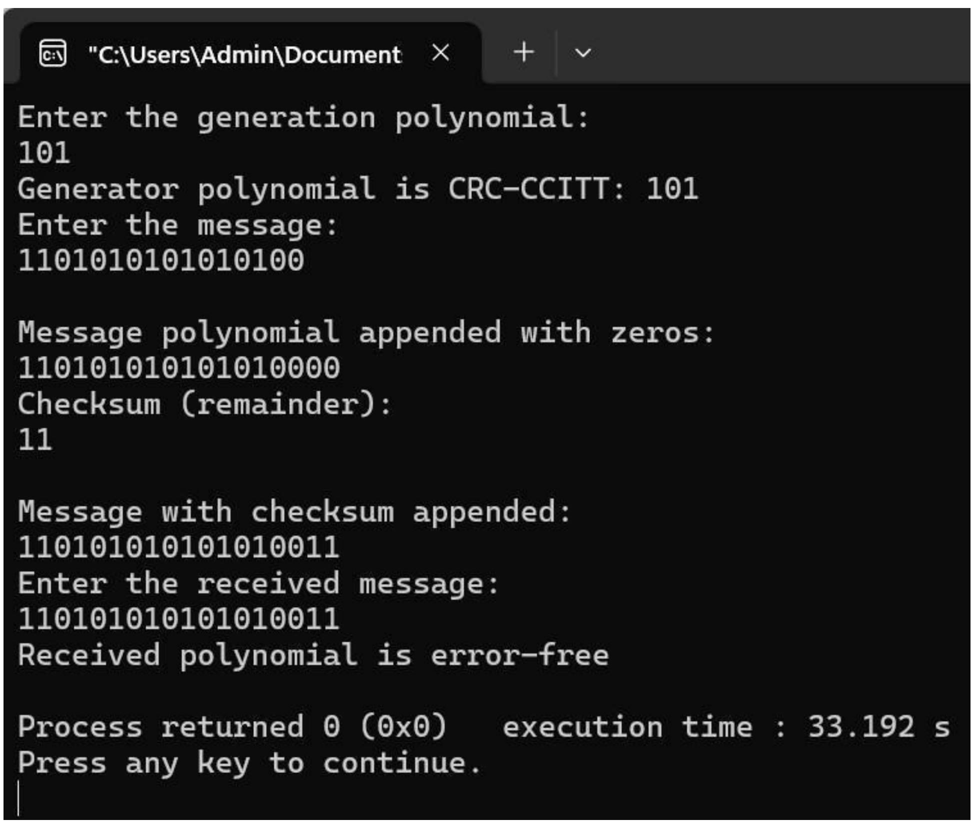
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

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++)
```

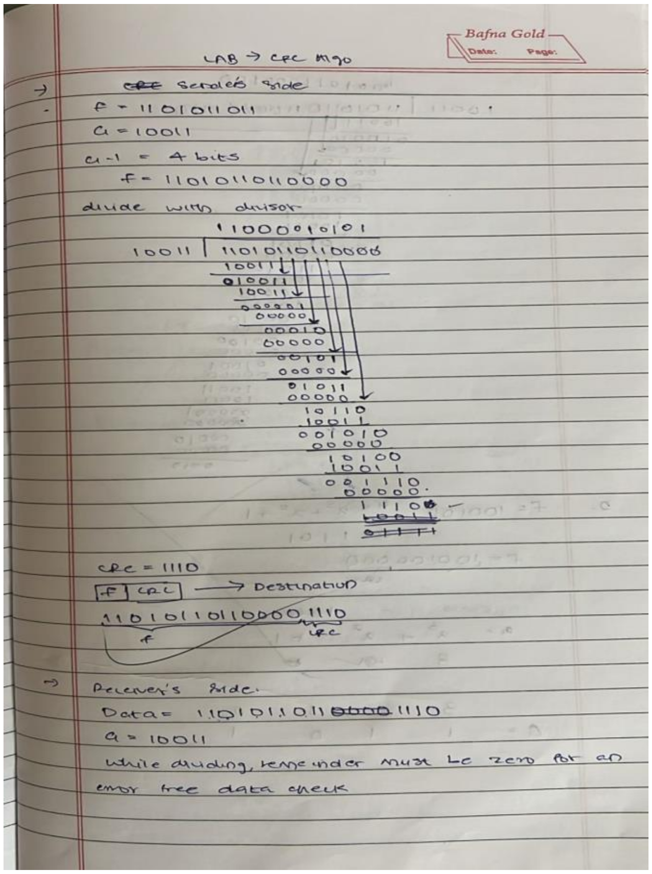
```c
if (rem[i] == '1') flag = 1;
        }

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

        return 0;
    }
```

Output:



```
"C:\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:


LAB → CRC Algo

→ ~~CRC~~ Sender's side

f = 1101011011

G = 10011

G-1 = 4 bits

f = 11010110110000

divide with divisor

1100001010

10011 | 11010110110000

CRC = 1110

$\boxed{f \mid CRC}$ → Destination

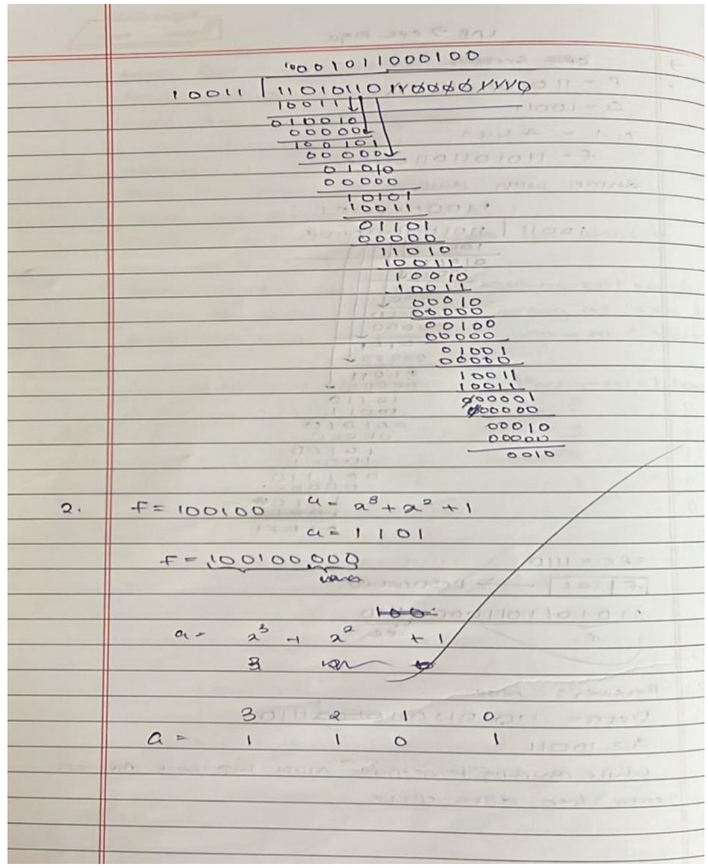
$\underbrace{11010110110000}_{f}\underbrace{1110}_{CRC}$

→ Receiver's side.

Data = 1101011011~~0000~~1110

G = 10011

while dividing, remainder must be zero for an error free data check


1000101100010

10011 | 11010110110000

0010

2. f = 100100 G = x^3+x^2+1

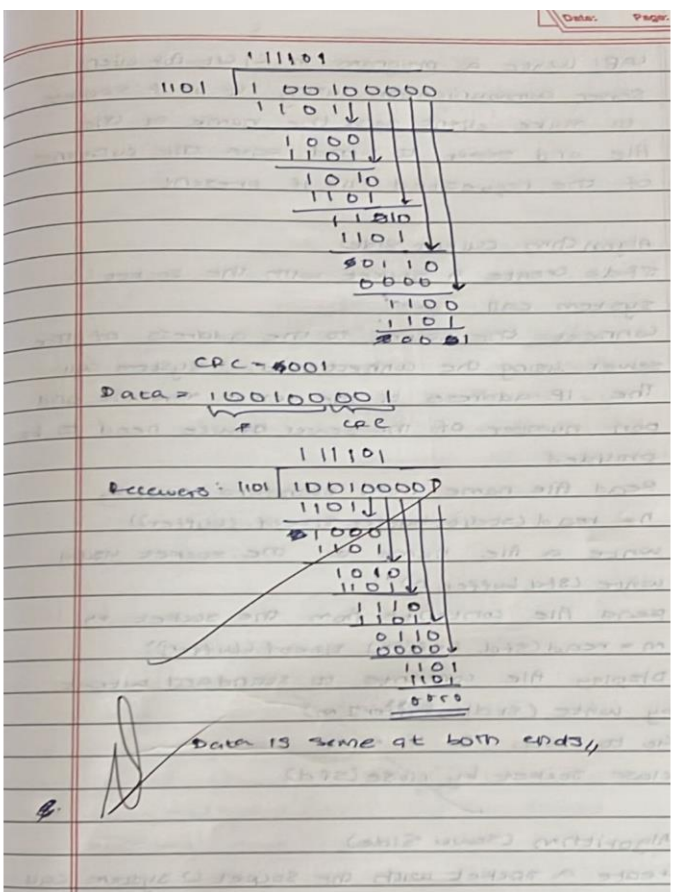
G = 1101

f = 100100000

G = $x^3 + x^2 + 1$

3 2 1 0

G = 1 1 0 1


111101

1101 | 100100000

CRC = 001

Data = $\underbrace{100100}_{f}\underbrace{001}_{CRC}$

111101

Receivers: 1101 | 100100001

000

Data is same at both ends.