VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Abhishek patil (1BM23CS013)

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 **Abhishek patil(1BM23CS013),** 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.

Praveen N	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE, BMSCE	Department of CSE, BMSCE

Index

Part - A

Part - B

Github Link:

https://github.com/abhishekpatil-bms/Computer-Networks

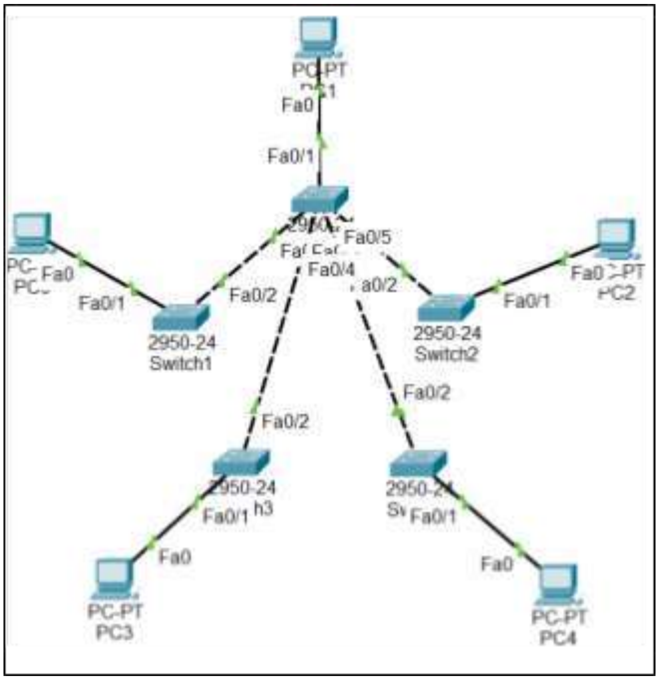
PART - A

<u>Program 1:</u>

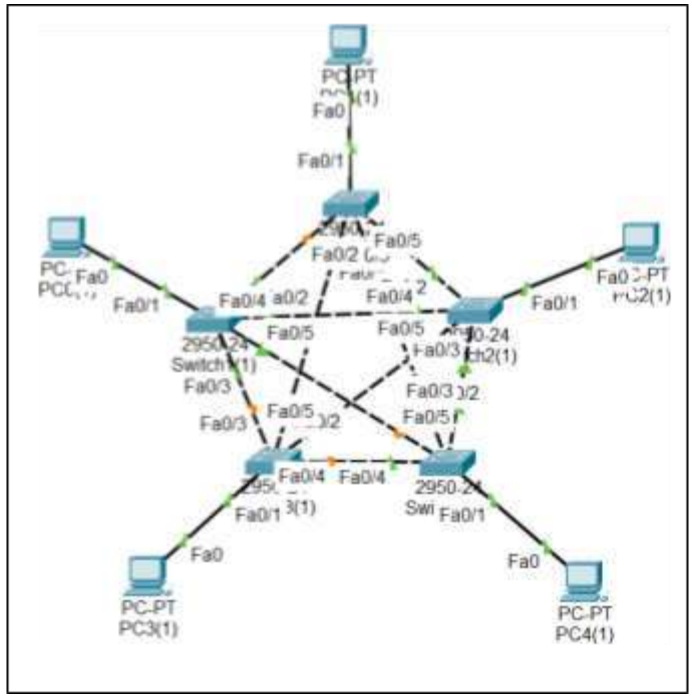
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Topology:

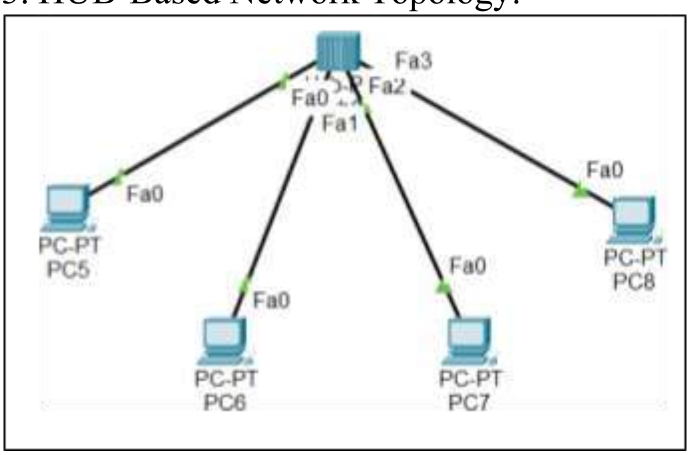
1. STAR Topology with Switch:



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.

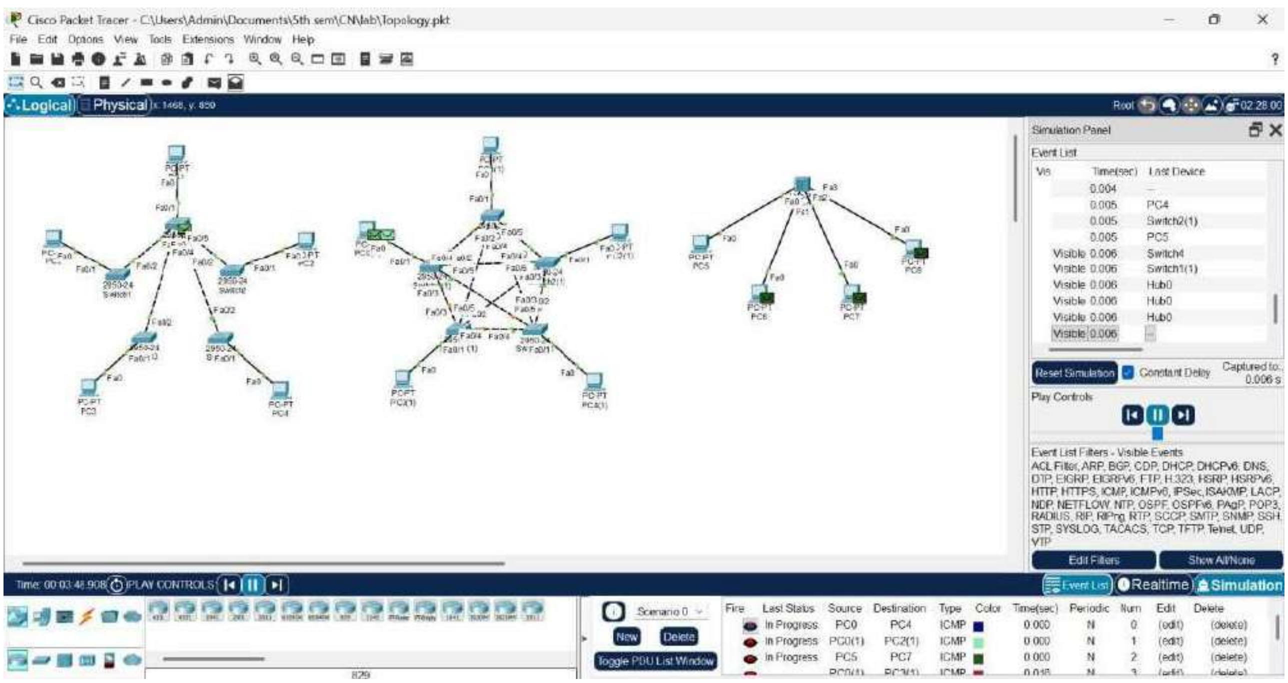
3. Create HUB-Based Topology

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.

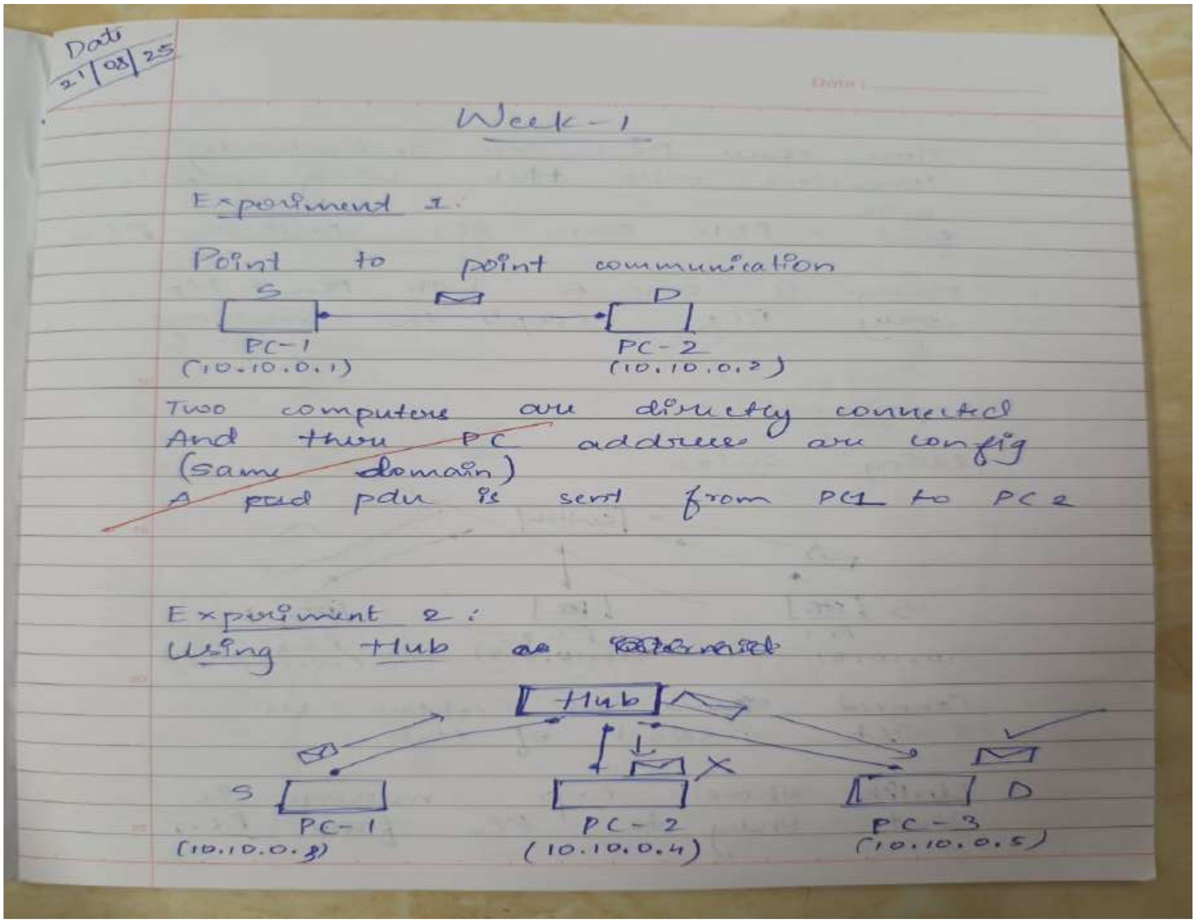
4. Demonstrate Ping Message (ICMP)

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:



Date 21/08/25

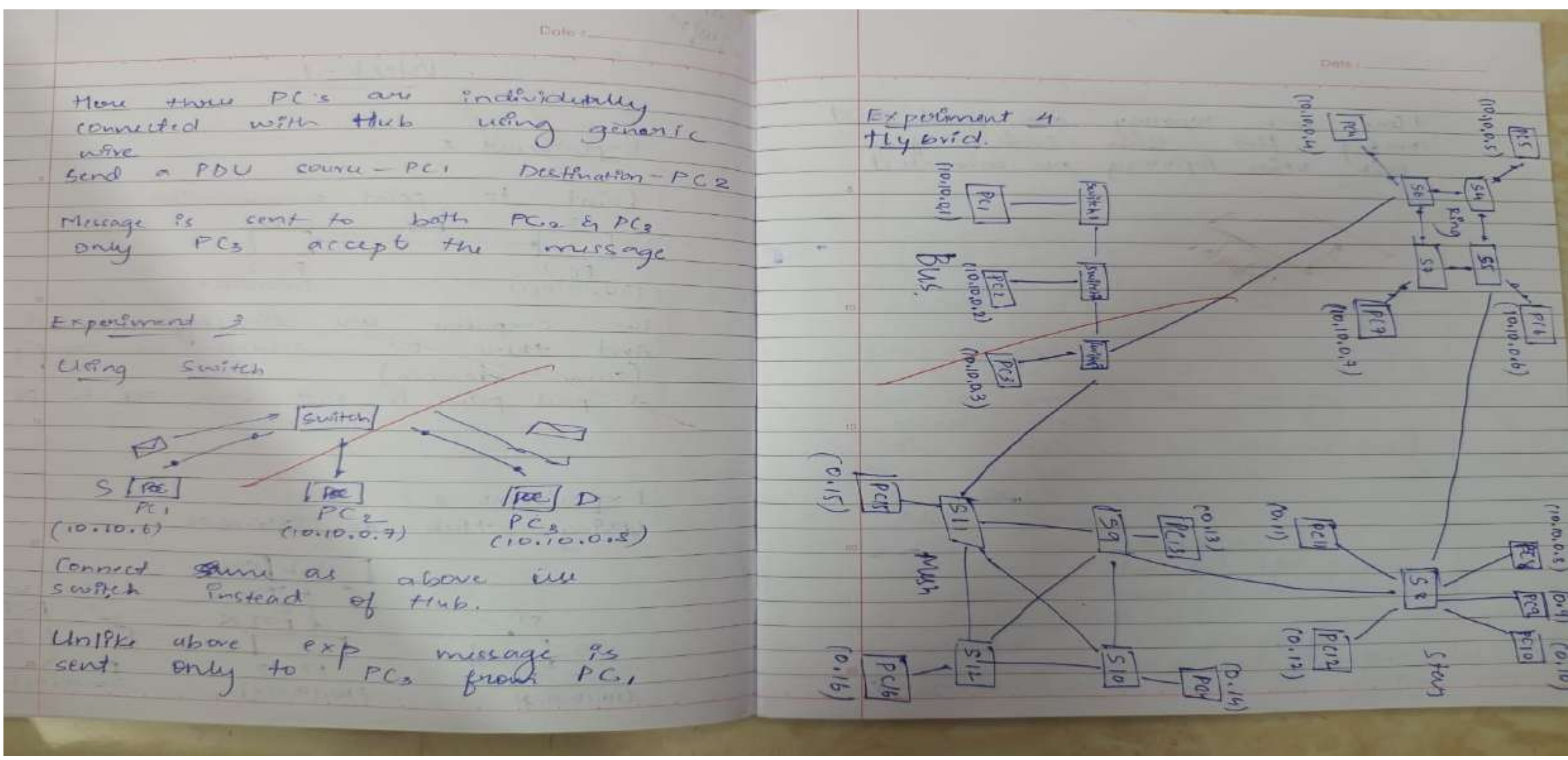
Week-1

Experiment 1:

Point to point communication

Two computers are directly connected And their PC address are config (same domain) A ~~fixed~~ pdu is sent from PC1 to PC2

Experiment 2:
Using Hub as [illegible]



Here three PC's are individually connected with Hub using generic wire
Send a PDU source - PC1 Destination - PC2

Message is sent to both PC2 & PC3 only PC3 accept the message

Experiment 3

Using Switch

Connect same as above use switch instead of Hub.

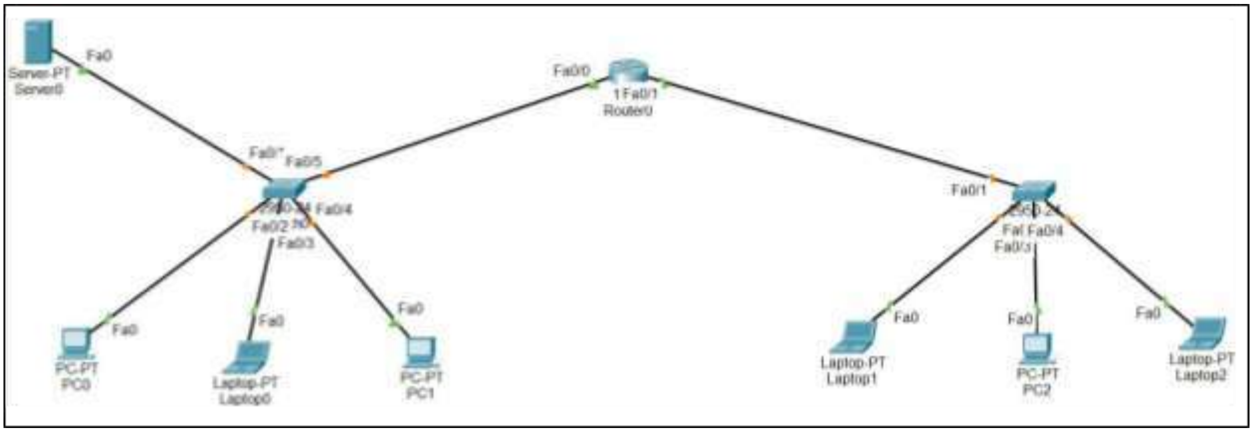
Unlike above exp message is sent only to PC3 from PC1

Experiment 4
Hybrid.

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

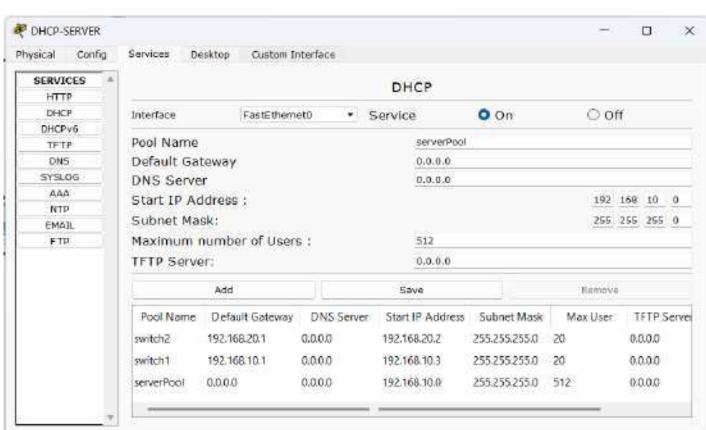
   Gateway: 192.168.10.1

   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**

i. Router>enable

ii. Router#configure terminal

(Within Lan)

iii. Router(config)# int fa0/0

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

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

vi. Router(config-if)# no shutdown

vii. Router(config-if)# exit

(Outside Lan)

viii. Router(config)# int fa0/1

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

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

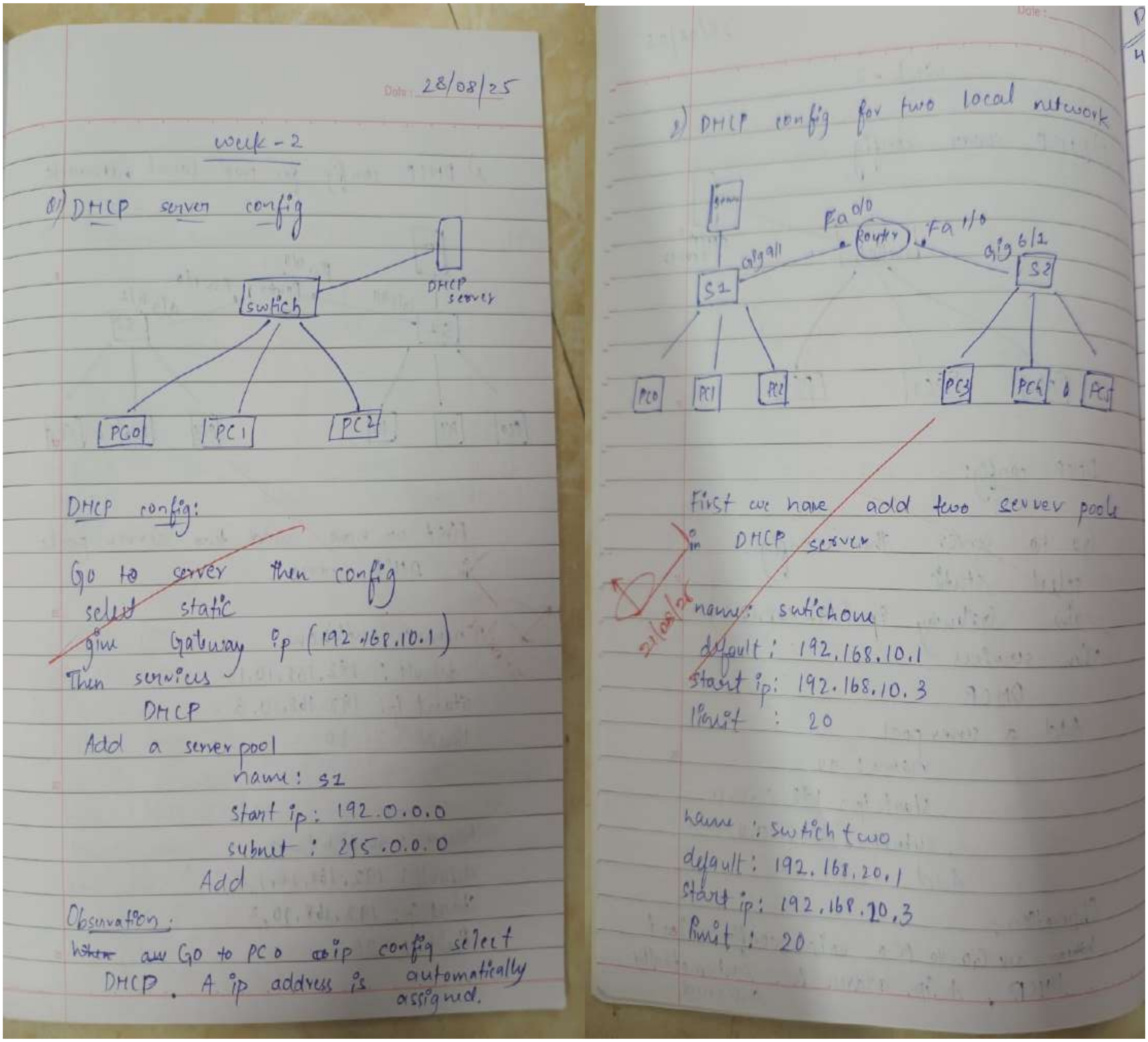
xi. Router(config-if)# no shutdown

xii. Router(config-if)# exit

xiii. Router(config)# exit

xiv. Router# write memory

Observation:



Date: 28/08/25

week - 2

Q1) DHCP server config

switch

DHCP server

PC0

PC1

PC2

DHCP config:

Go to server then config

select static

give Gateway ip (192.168.10.1)

Then services

DHCP

Add a server pool

name: s1

start ip: 192.0.0.0

subnet: 255.0.0.0

Add

Observation:

~~When~~ we Go to PC0 ip config select DHCP. A ip address is automatically assigned.

2) DHCP config for two local network

server

Fa 0/0

Router

Fa 1/0

gig 0/1

gig 0/1

S1

S2

PC0

PC1

PC2

PC3

PC4

PC5

First we have add two server pools in DHCP server

21/08/25

name: switchone

default: 192.168.10.1

start ip: 192.168.10.3

limit: 20

name: switch two

default: 192.168.20.1

start ip: 192.168.20.3

limit: 20

Output:

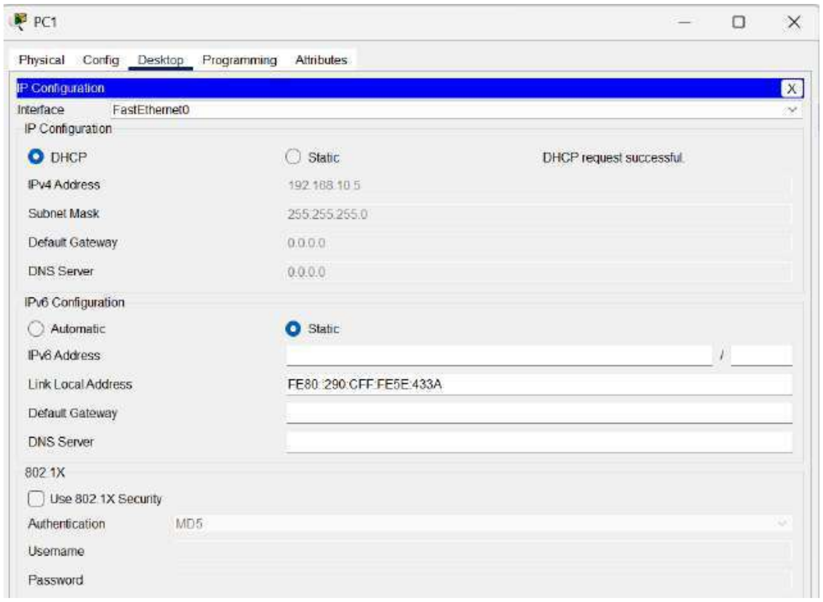


Fig 1. Ip address assigned by DHCP server within Lan (PC1)

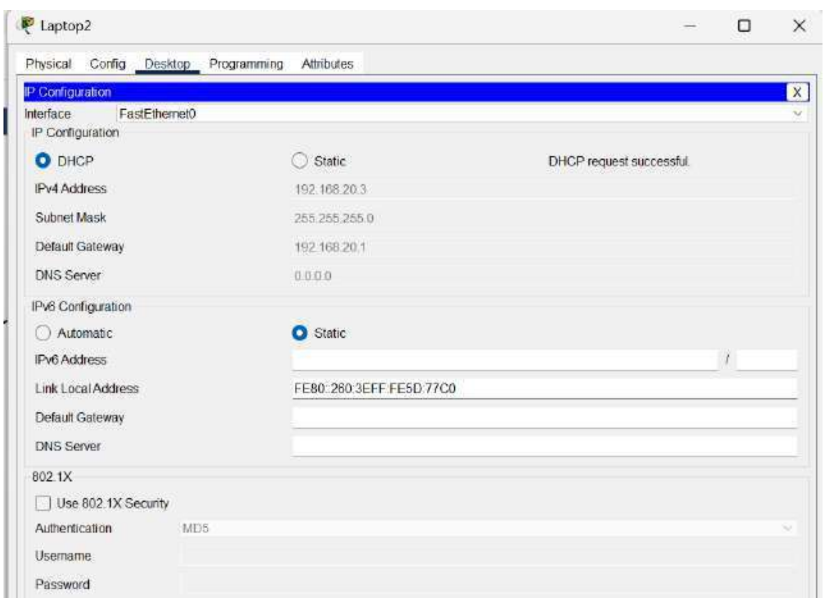
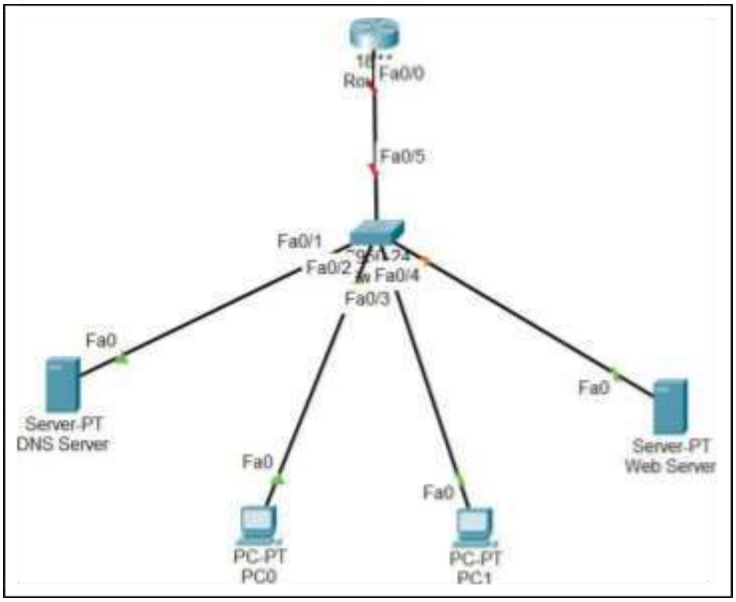


Fig 2. Ip address assigned by DHCP server outside Lan (laptop2)

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.

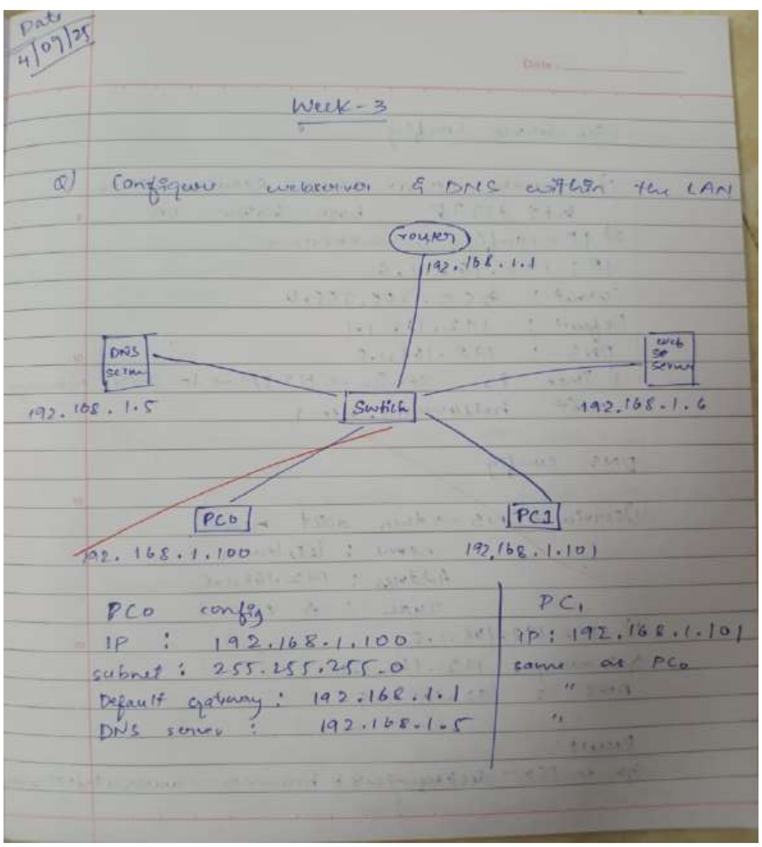
4. Configure Web Server

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:

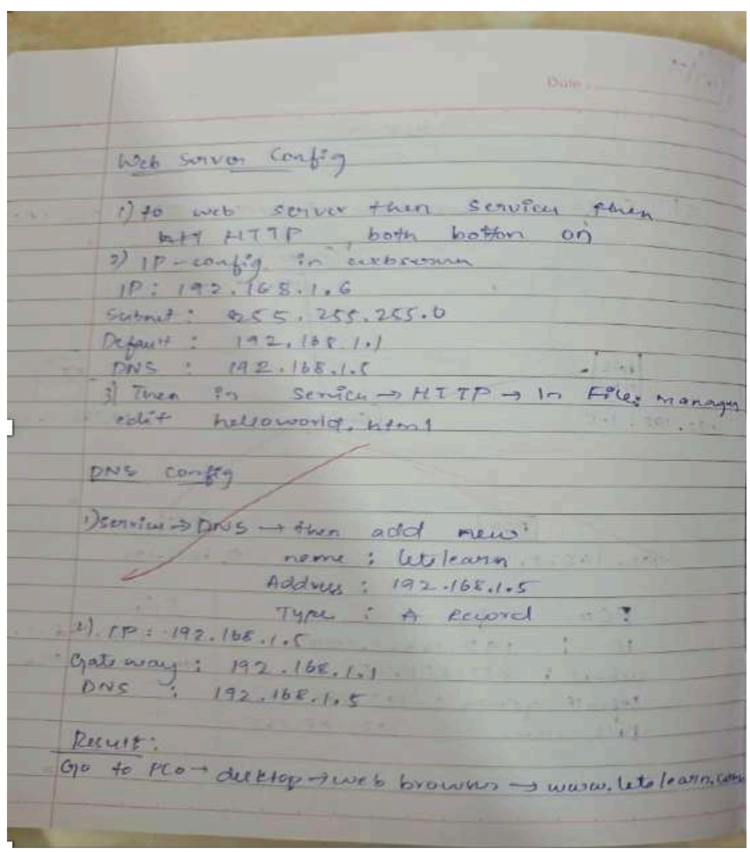

Date 4/09/25

Week-3

Q) Configure webserver & DNS within the LAN

PC0 config
IP : 192.168.1.100
subnet : 255.255.255.0
Default gateway : 192.168.1.1
DNS server : 192.168.1.5

PC1
IP : 192.168.1.101
same as PC0
"
"


Web Server Config

1) to web server then services then HTTP, both button on
2) IP-config in desktop
IP : 192.168.1.6
Subnet : 255.255.255.0
Default : 192.168.1.1
DNS : 192.168.1.5
3) Then in Service → HTTP → In File manager edit helloworld.html

DNS config

1) Service → DNS → then add new:
name : letslearn
Address : 192.168.1.5
Type : A Record
2) IP : 192.168.1.5
Gateway : 192.168.1.1
DNS : 192.168.1.5

Result:
Go to PC0 → desktop → web browser → www.letslearn.com

Output:

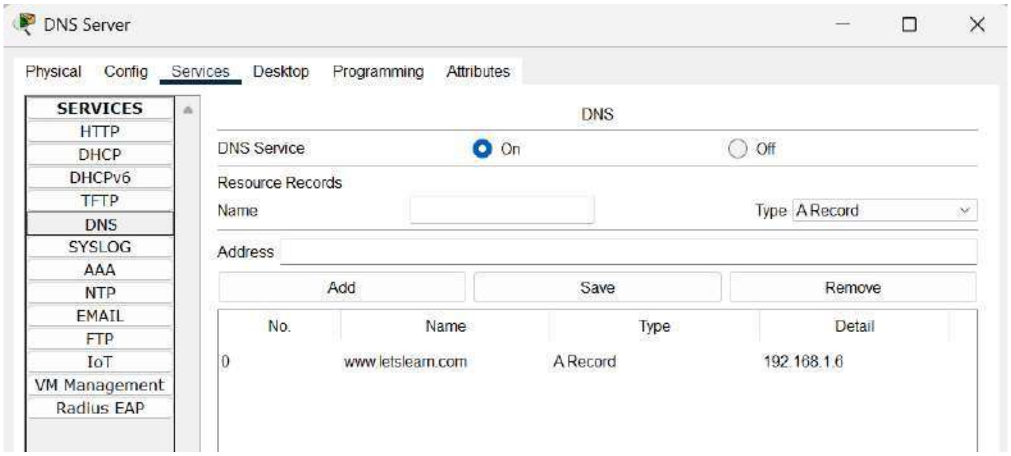


Fig 1. DNS server – DNS Services

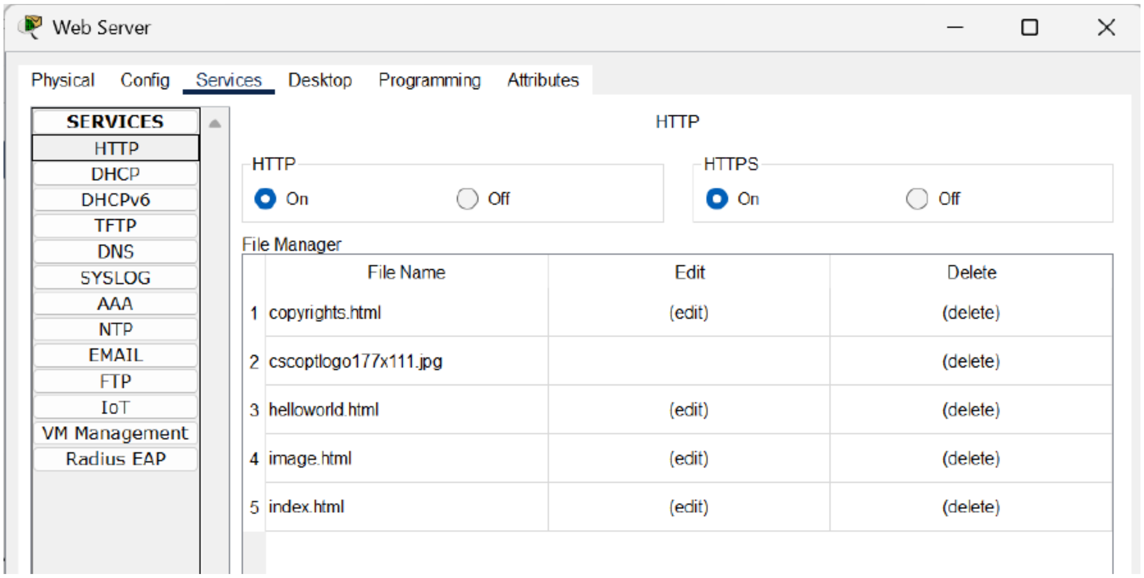


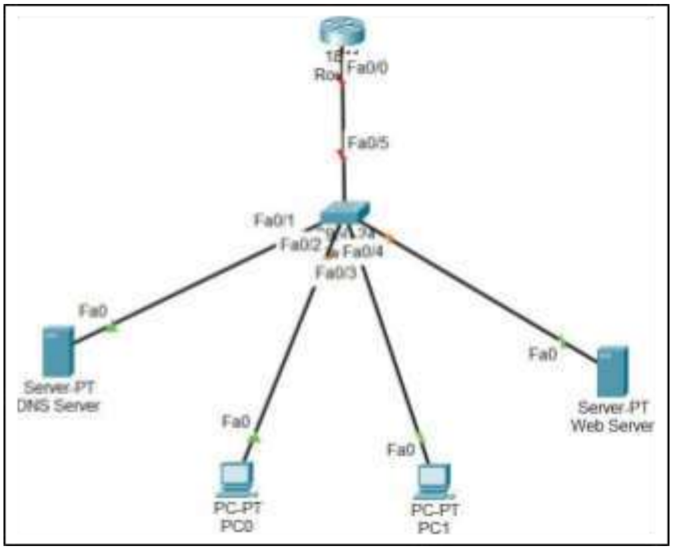
Fig 2. WEB server – HTTP Services

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.

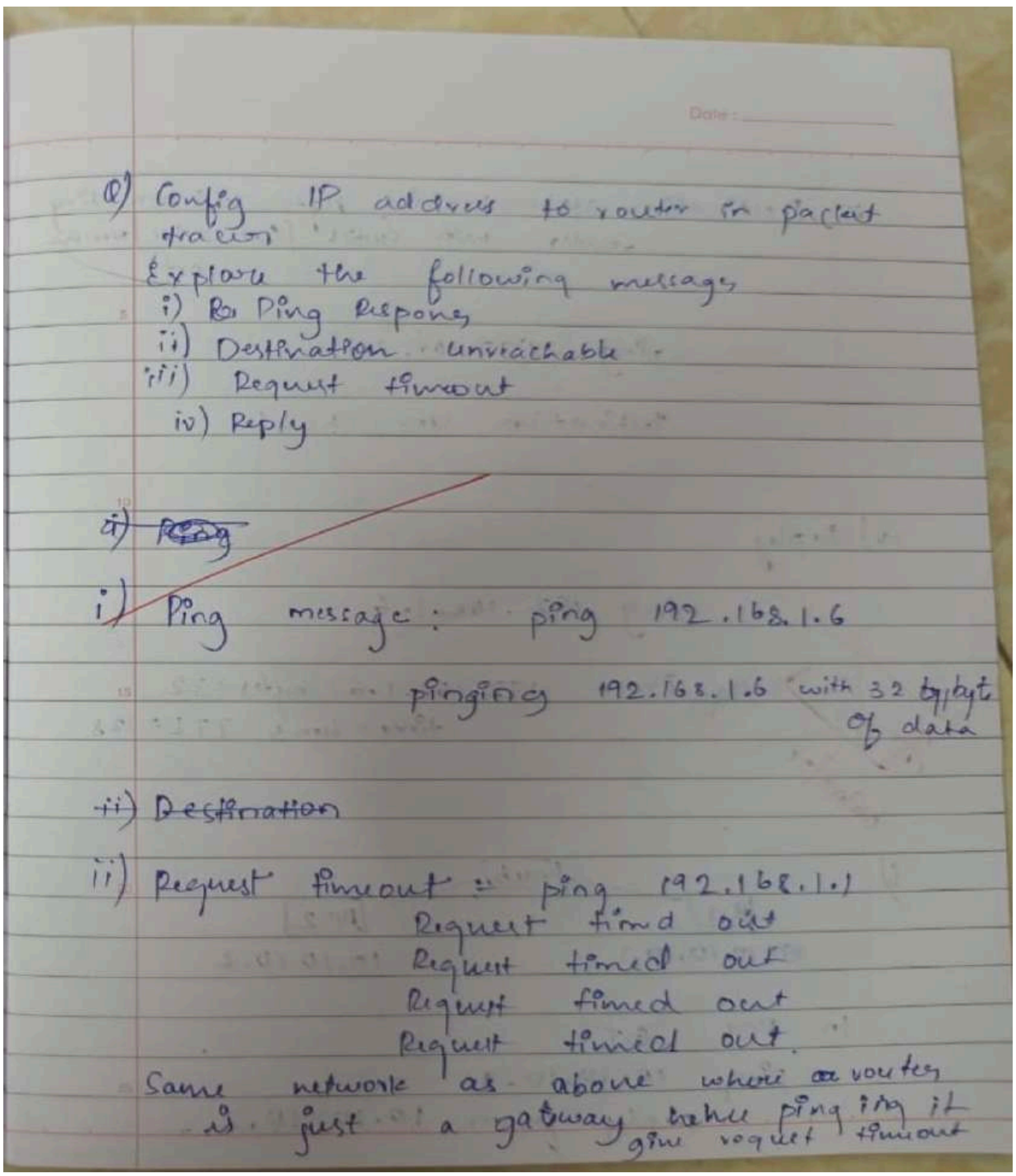
   Example commands:

   ping 192.168.1.2

   ping 192.168.1.101

   ping 192.168.2.200

Observation:



Q) Config IP address to router in packet tracer

Explore the following message

i) ~~Rs~~ Ping Respons

ii) Destination unreachable

iii) Request timeout

iv) Reply

~~i) Ping~~

i) Ping message : ping 192.168.1.6

pinging 192.168.1.6 with 32 byte of data

~~ii) Destination~~

ii) Request timeout : ping 192.168.1.1

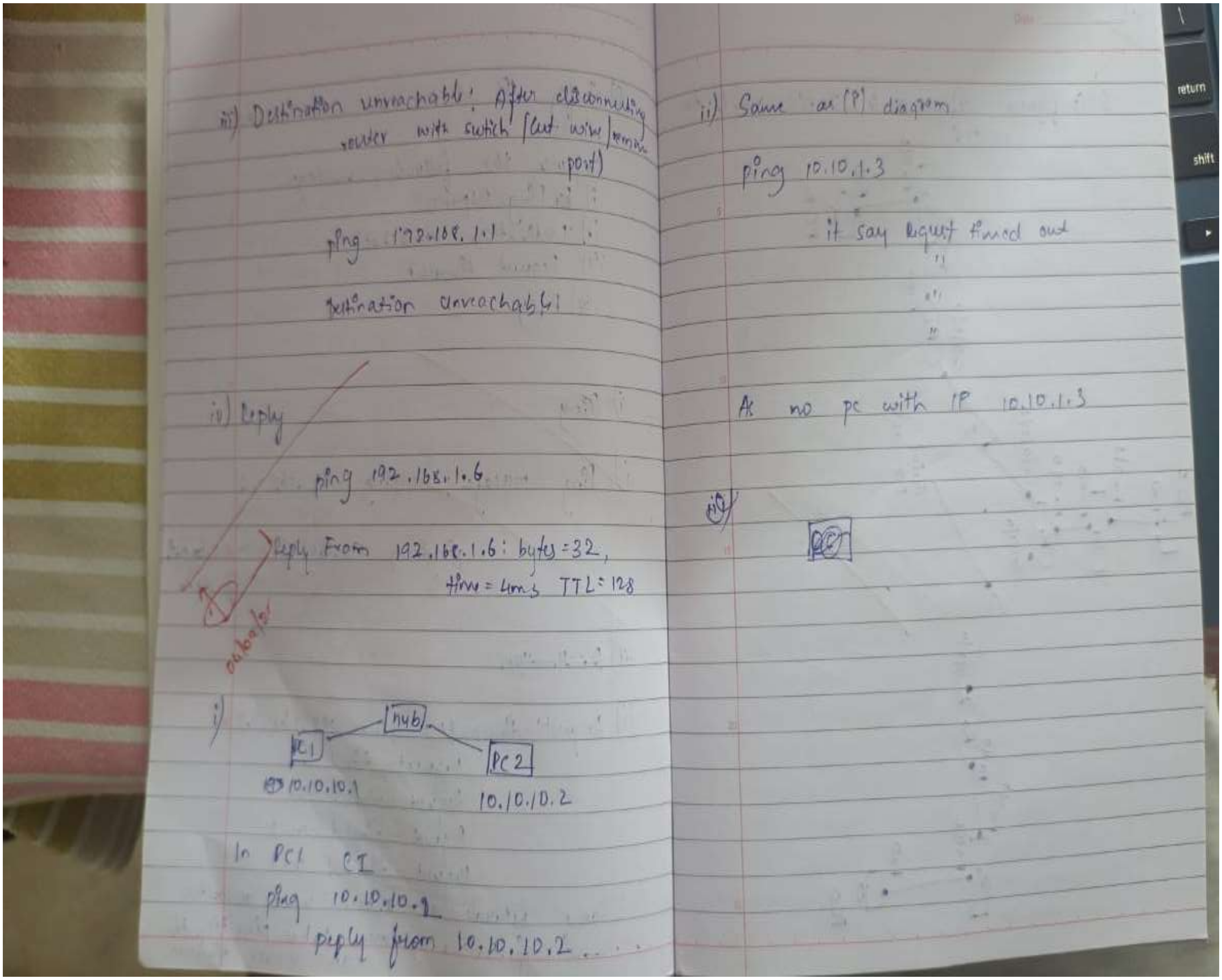
Request timed out

Request timed out

Request timed out

Request timed out.

Same network as above where router is just a gateway hence pinging it give request timeout



iii) Destination unreachable: After disconnecting router with switch (cut wire/remove port)

ping 192.168.1.1

Destination unreachable

iv) Reply

ping 192.168.1.6

Reply From 192.168.1.6: bytes=32, time=4ms TTL=128

i)

hub

PC1 10.10.10.1

PC2 10.10.10.2

In PC1 CI

ping 10.10.10.2

reply from 10.10.10.2 ...

ii) Same as (i) diagram

ping 10.10.1.3

- it say request timed out

"

"

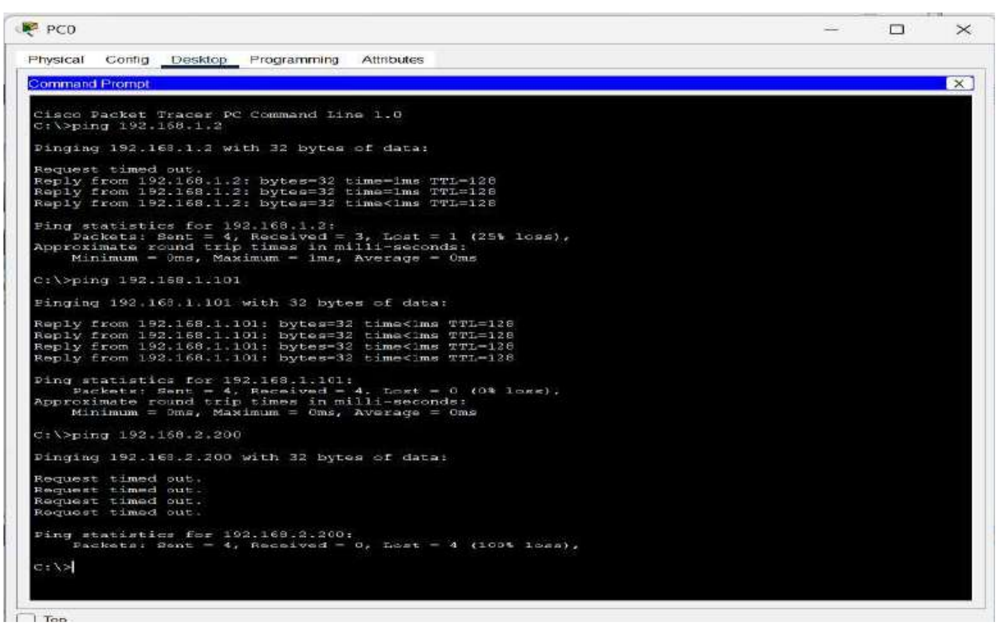
"

As no pc with IP 10.10.1.3

iii)

PC

Output:



PC0

Physical Config Desktop Programming Attributes

Command Prompt

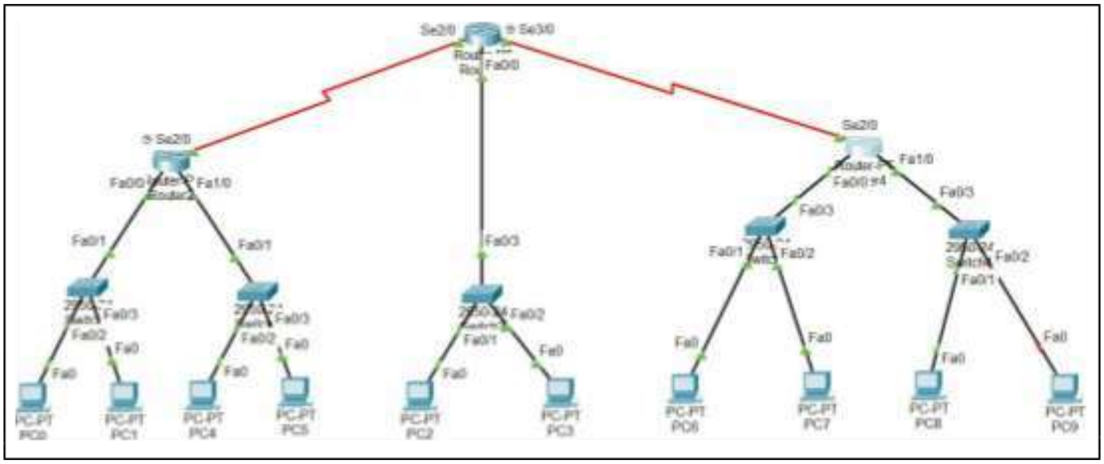
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

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

<u>Program 5:</u>

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

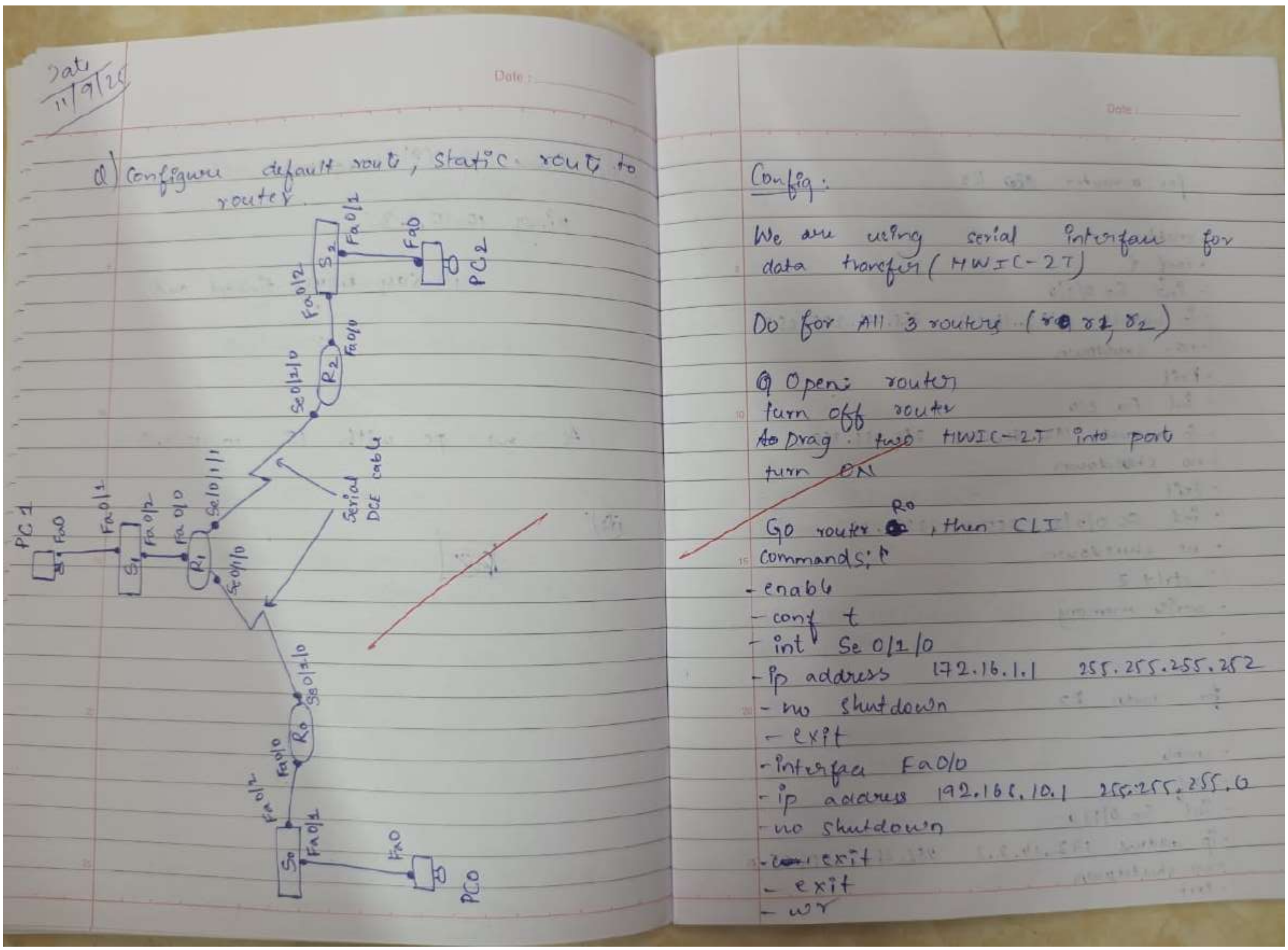
- 192.168.2.0 /24 → via 192.168.7.1
- 192.168.3.0 /24 → via 192.168.7.1
- 192.168.4.0 /24 → via 192.168.7.1

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



for a router R1

- enable
- conf t
- int Se 0/1/0
- ip address 172.16.1.2 255.255.255.252
- no shutdown
- Exit
- int Fa 0/0
- ip address 192.168.20.1 255.255.255.0
- no shutdown
- Exit
- int Se 0/1/2 255.255.255.252
- no shutdown
- ctrl + Z
- write memory

for router R2

- enable
- conf t
- int Se 0/1/0
- ip address 172.16.2.2 255.255.255.252
- no shutdown
- Exit
- int Fa0/0
- ip address 192.168.30.1 255.255.255.0
- no shutdown
- Exit
- ctrl + Z
- wr

PC configure

PC0
IP: 192.168.10.10
Gatway: 192.168.10.1

PC1:
IP: 192.168.20.10
Gatway: 192.168.20.1

PC2
IP: 192.168.30.10
Gatway: 192.168.30.1

Rout Configuration:

R0:
- enable
- conf t
- ip rout 192.168.20.0 255.255.255.0 172.16.1.2
- ip rout 172.16.2.0 255.255.255.252 172.16.1.2
- ip rout 192.168.30.0 255.255.255.0 172.16.1.2
- ctrl + Z
- wr

R1:
- enable
- conf t
- ip rout 192.168.10.0 255.255.255.0 172.16.1.1
- ip rout 192.168.30.0 255.255.255.0 172.16.2.2
- ctrl Z
- wr

R3:
- conf t
- ip rout 0.0.0.0 0.0.0.0 Se 0/1/0
- ctrl + Z
- wr

[This don to make R3 as default

To check
Router - CLI
#show ip rout

If all successful, it is right

ping 192.168.20.1 from PC0
ping 192.168.30.1 from PC0
Ping 192.168.20.1 from PC2
} should Reply successfull

11/09/25

Output:

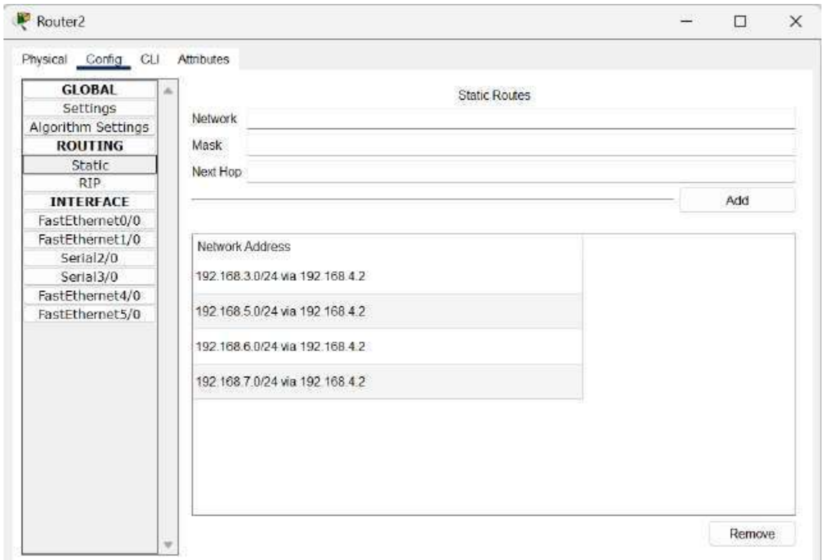


Fig 1. Router 2 – Static routing

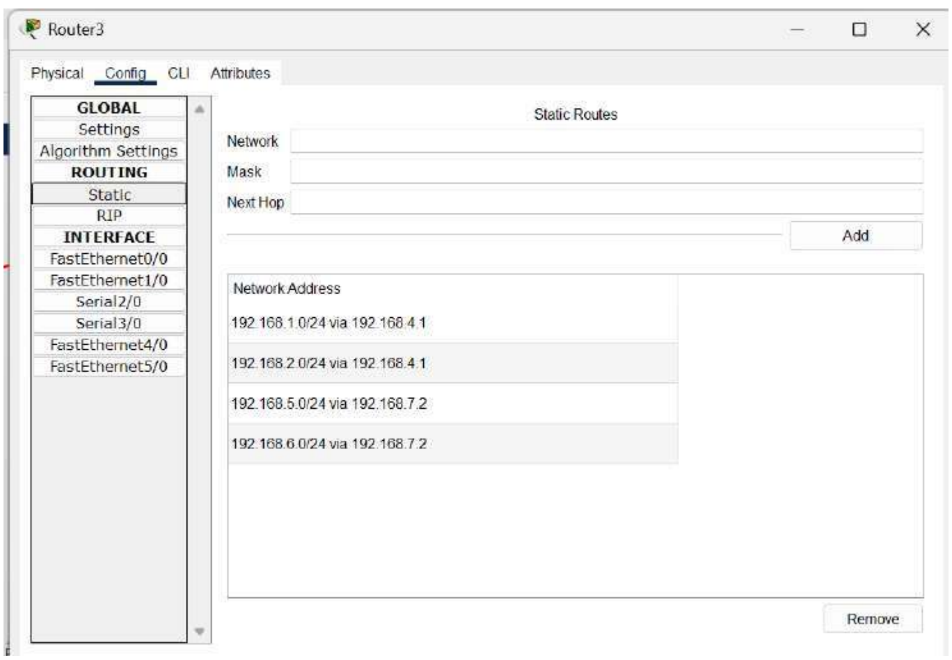


Fig 2. Router 3 – Static routing

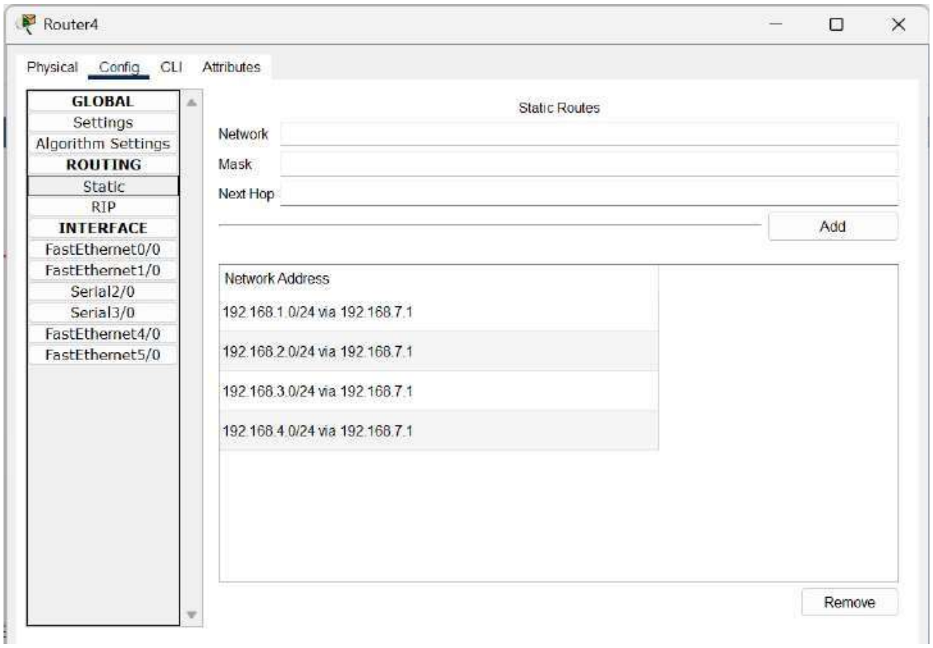
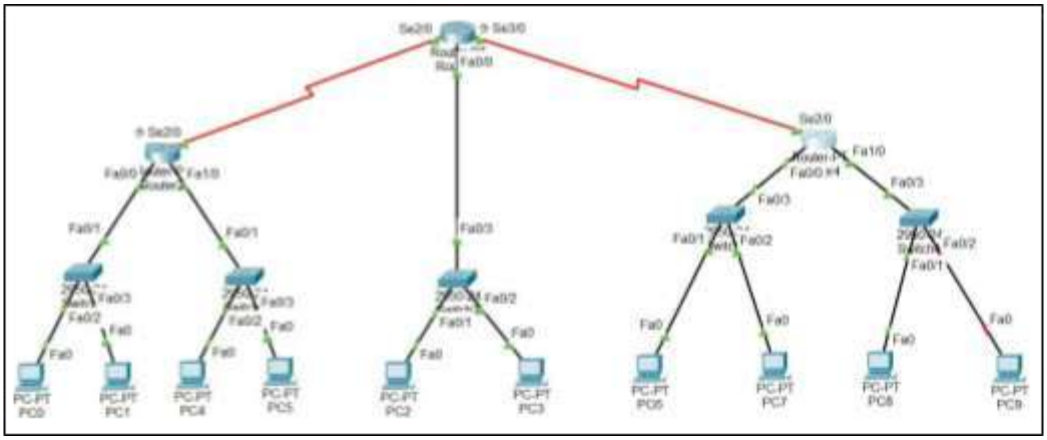


Fig 3. Router 4 – Static routing

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

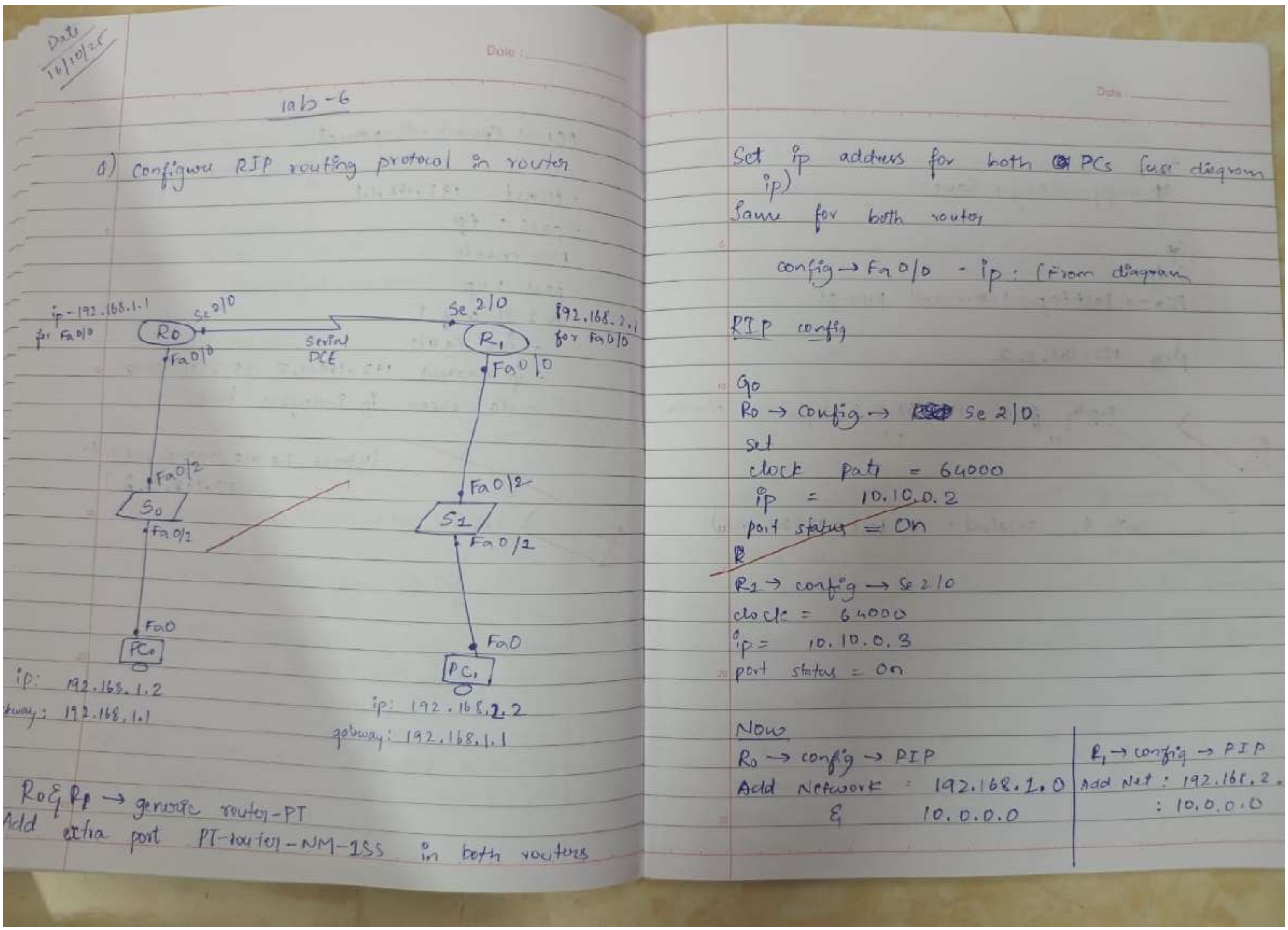
1. On any router → CLI
2. Use:
3. show ip route

→ 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



Date :

then
st. config → setting → Save

Output:
PC0 → Desktop → command prompt

ping 192.168.2.2

Reply from 192.168.2.2 : bytes = 32 time...
" "
" "
" "

sent = 4, received = 4, lost = 0 (0% loss)

Output:

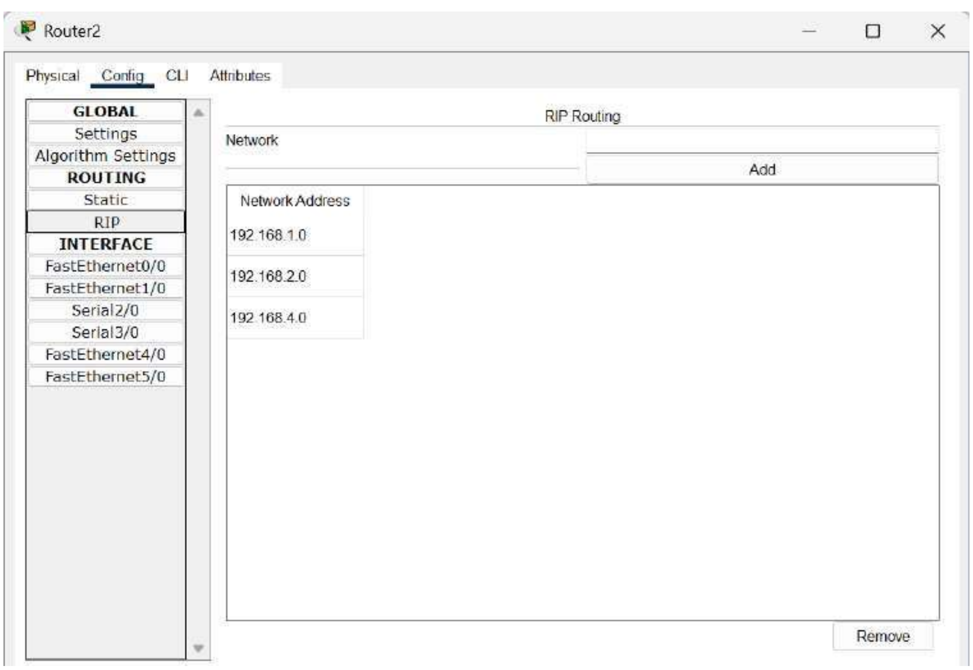


Fig 1. Router 2 – RIP routing

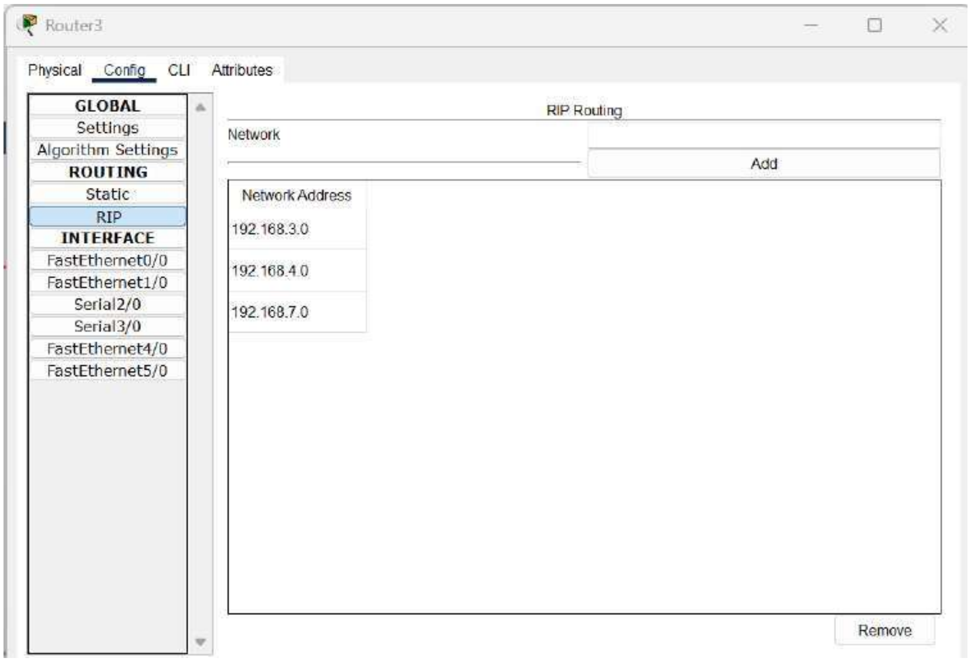


Fig 2. Router 3 – RIP routing

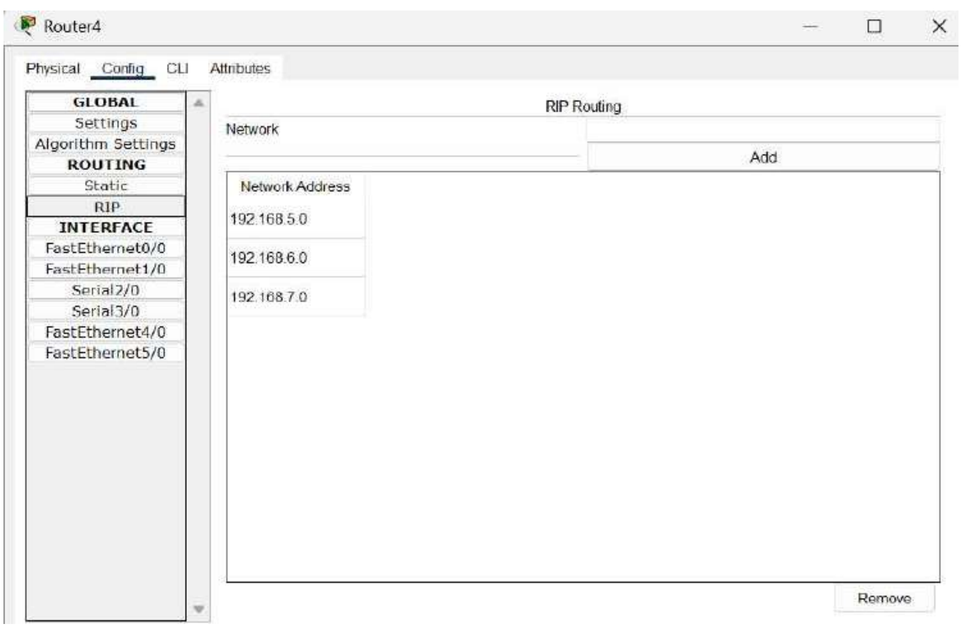
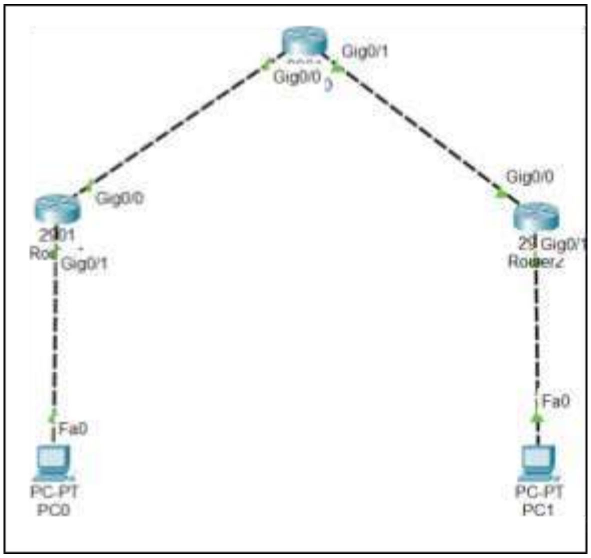


Fig 3. Router 4 – RIP routing

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)

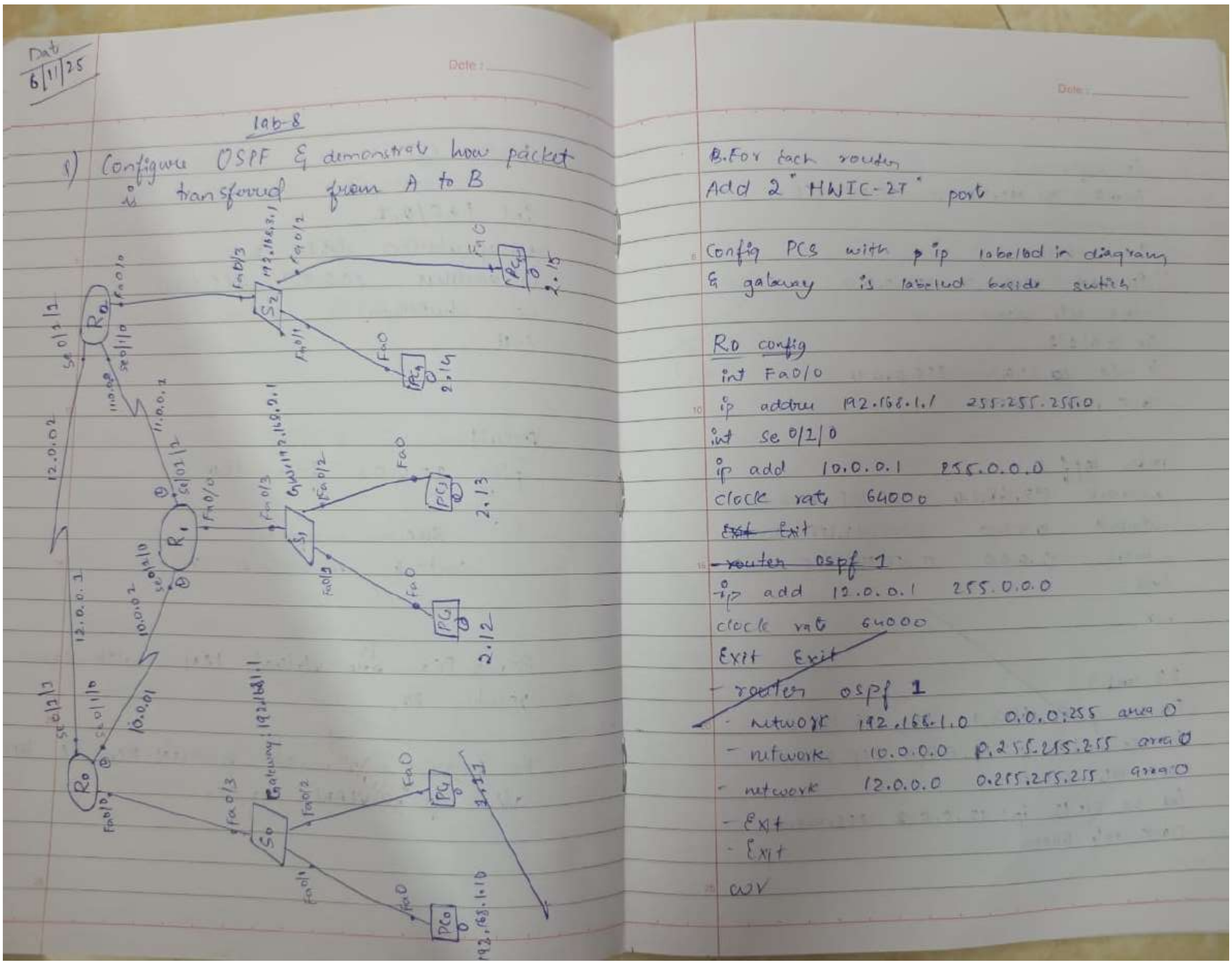
Router 2

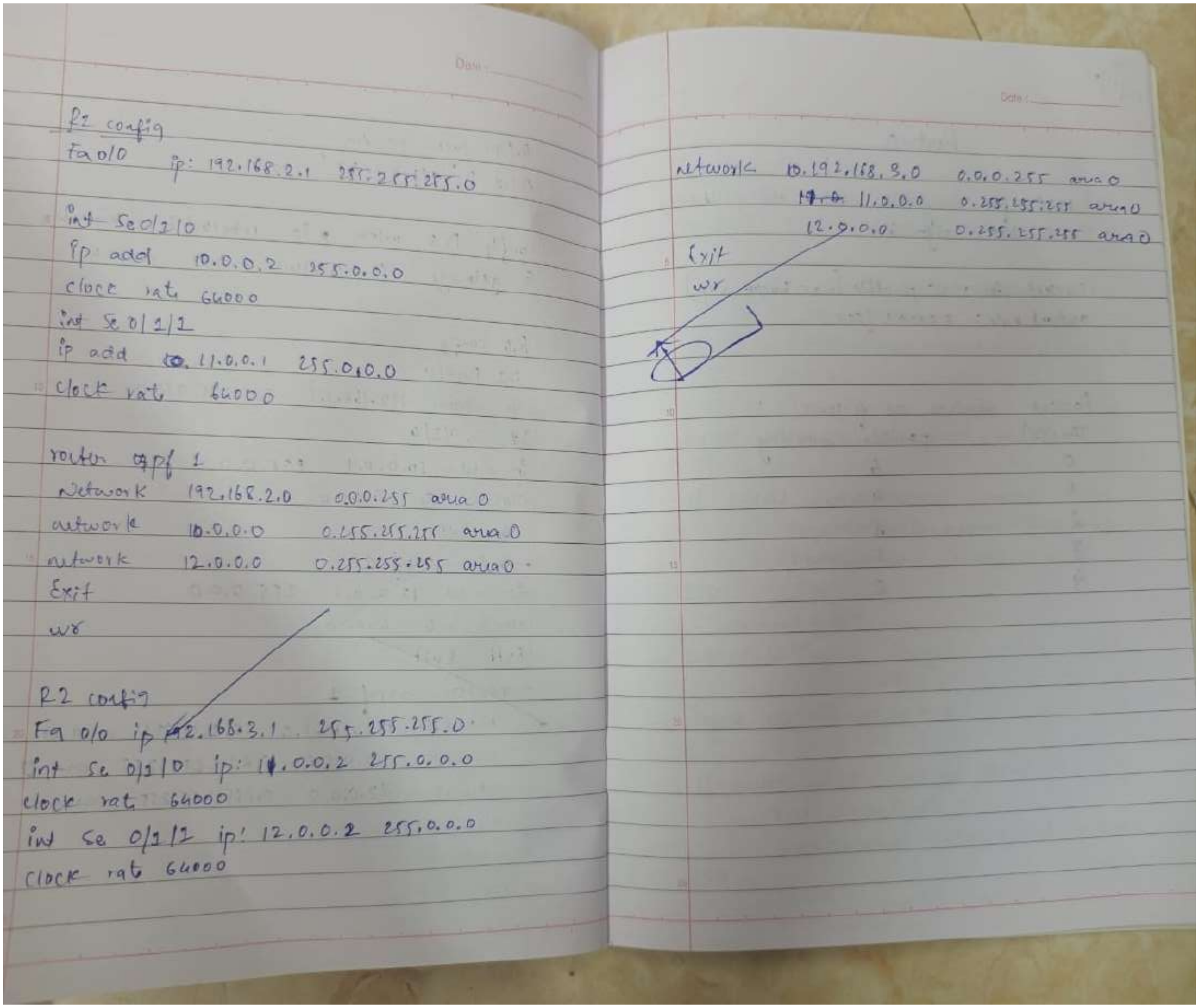
1. Process ID = 1
2. Add networks:
   - 192.168.2.0 /24
   - 20.0.0.0 /30 (link to center router)

3. Test Connectivity

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

3. Observation:




R2 config
Fa0/0 ip: 192.168.2.1 255.255.255.0

int Se0/1/0
ip add 10.0.0.2 255.0.0.0
clock rate 64000
int Se 0/1/1
ip add 11.0.0.1 255.0.0.0
clock rate 64000

router ospf 1
network 192.168.2.0 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
wr

R2 config
Fa 0/0 ip 192.168.3.1 255.255.255.0
int Se 0/1/0 ip: 11.0.0.2 255.0.0.0
clock rat 64000
int Se 0/1/1 ip: 12.0.0.2 255.0.0.0
clock rat 64000

network 192.168.3.0 0.0.0.255 area 0
11.0.0.0 0.255.255.255 area 0
12.0.0.0 0.255.255.255 area 0
Exit
wr

Output:

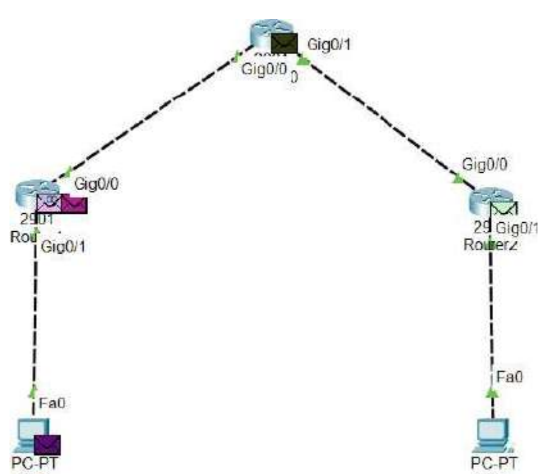
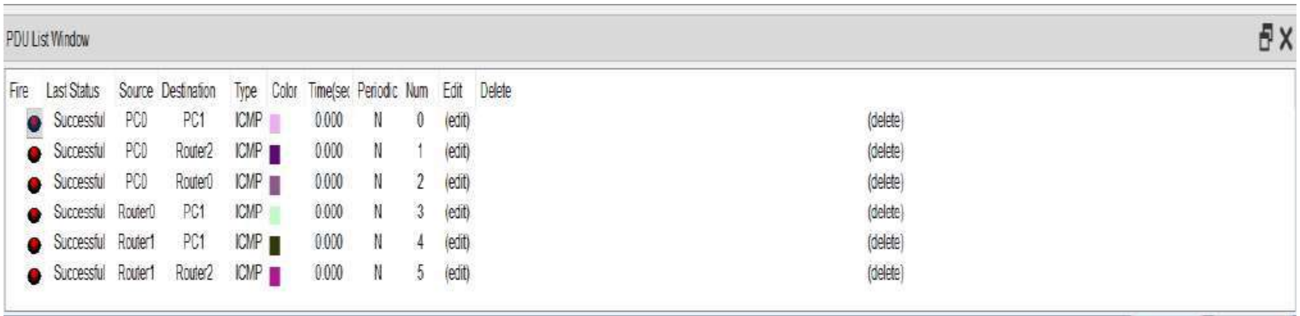


Fig 1. Sending PDU message from PC0 to PC1


PDU List Window

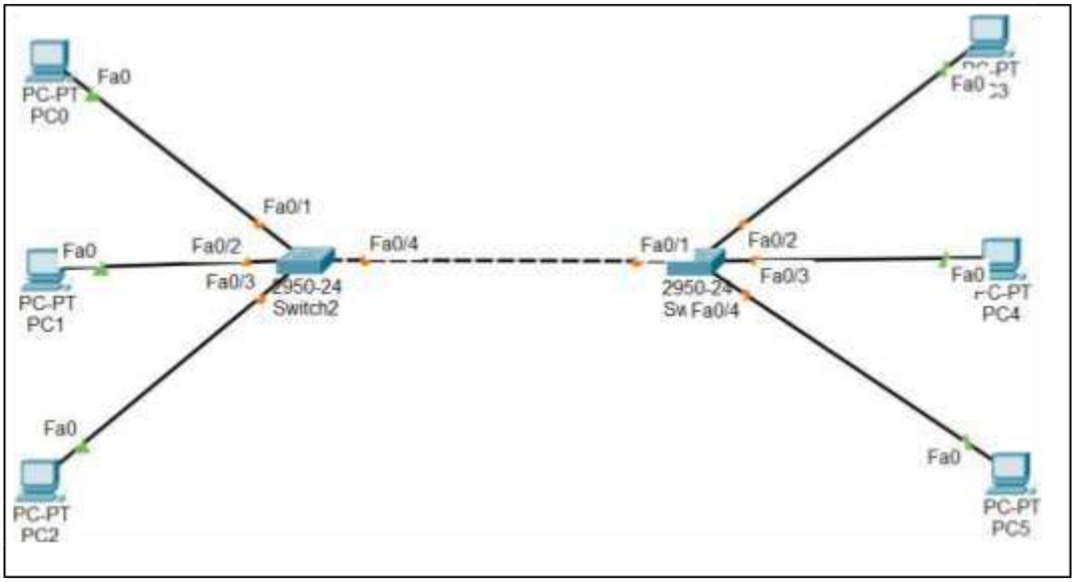
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Router2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	Router0	ICMP		0.000	N	2	(edit)	(delete)
	Successful	Router0	PC1	ICMP		0.000	N	3	(edit)	(delete)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

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.

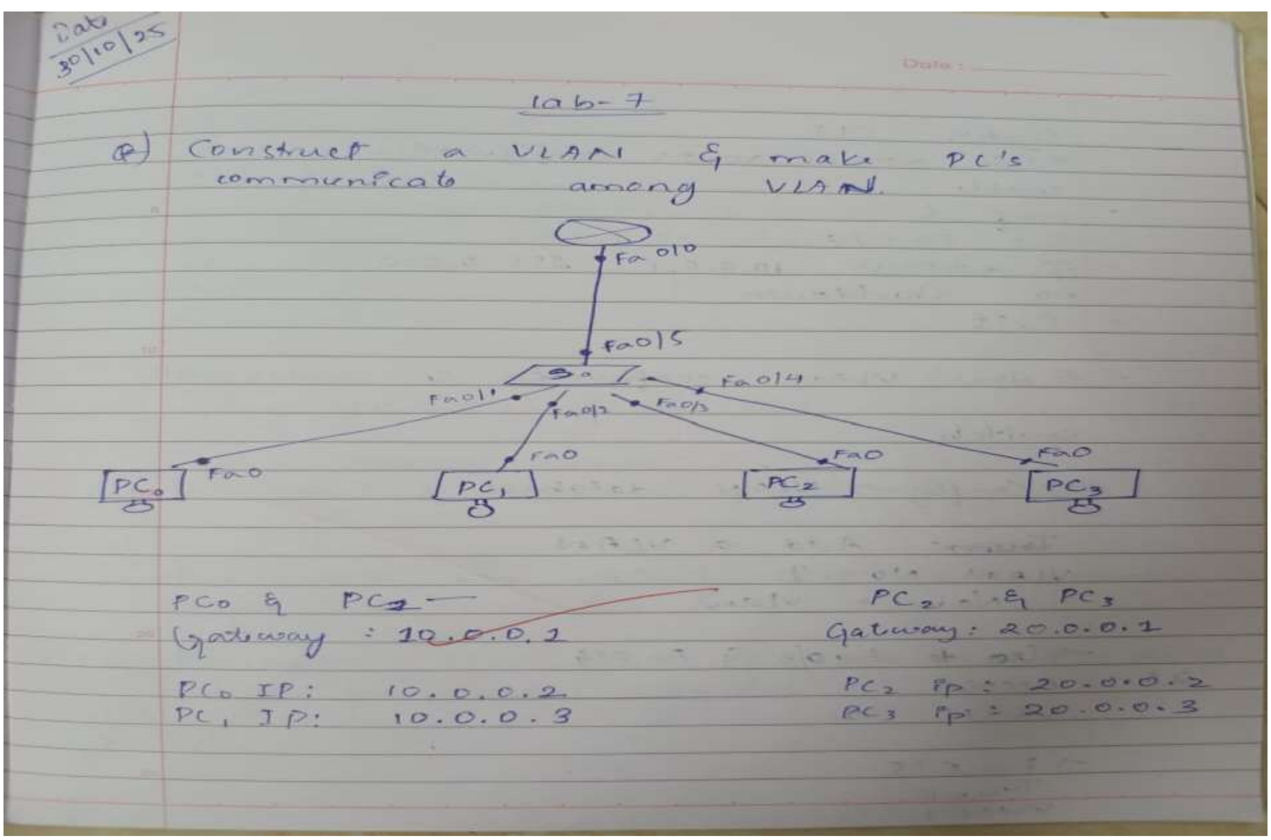
4. Assign IPs to PCs

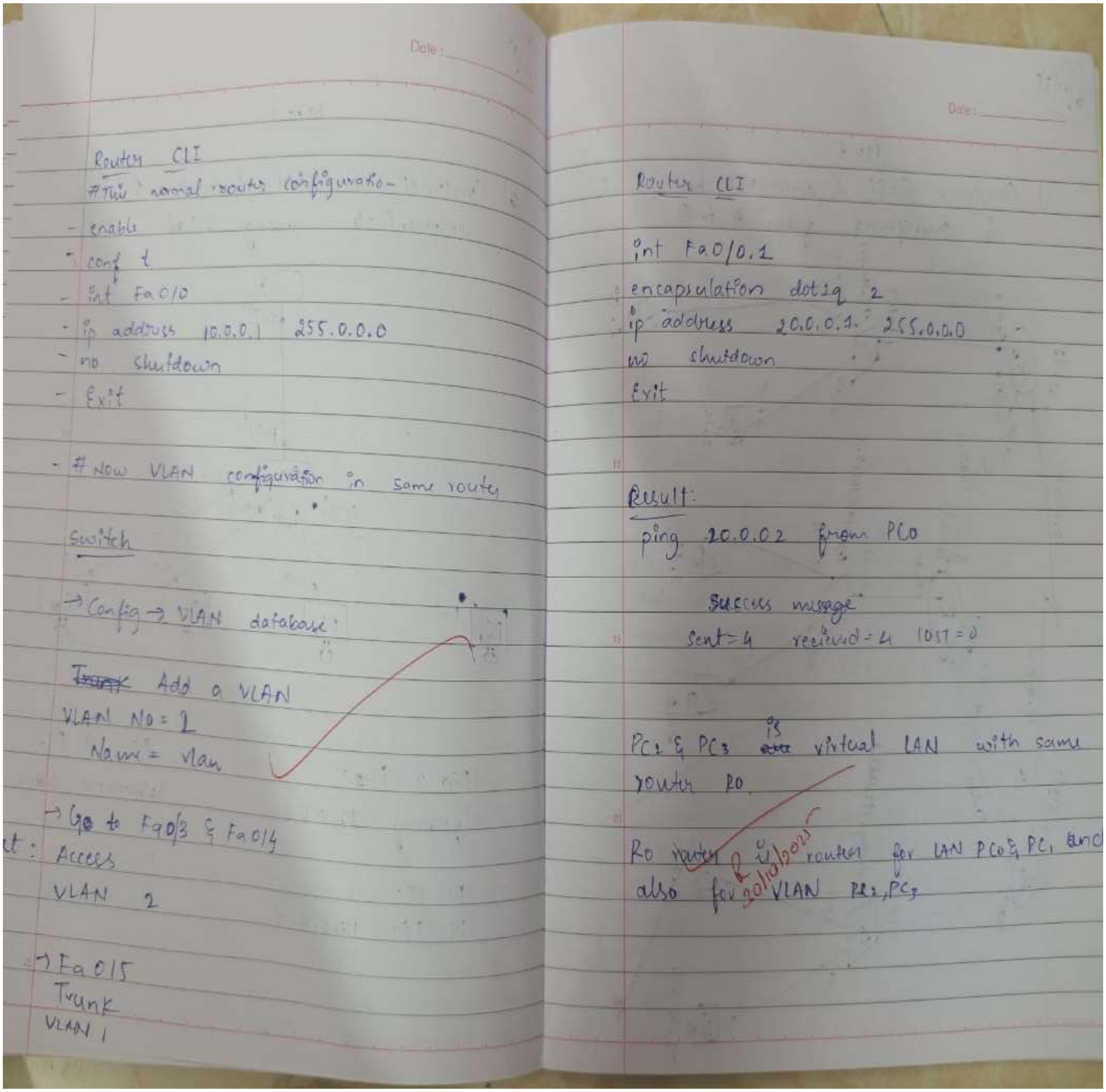
1. On each PC → Desktop → IP Configuration
2. Assign IPs in VLAN-specific networks (example):
   - VLAN 10 → 192.168.10.x
   - VLAN 20 → 192.168.20.x

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:





Router CLI
This normal router configuration
- enable
- conf t
- int Fa0/0
- ip address 10.0.0.1 255.0.0.0
- no shutdown
- Exit

- # Now VLAN configuration in same router

Switch

→ Config → VLAN database

~~Trunk~~ Add a VLAN
VLAN No = 2
Name = vlan

→ Go to Fa0/3 & Fa0/4
t: Access
VLAN 2

→ Fa0/5
Trunk
VLAN 1

Router CLI

int Fa0/0.1
encapsulation dot1q 2
ip address 20.0.0.1 255.0.0.0
no shutdown
Exit

Result:
ping 20.0.0.2 from PC0

Success message
Sent = 4 received = 4 lost = 0

PC2 & PC3 is virtual LAN with same router R0.

R0 ~~router~~ is router for LAN PC0 & PC1 and also for VLAN PC2, PC3

R 20/10/2025

Output:

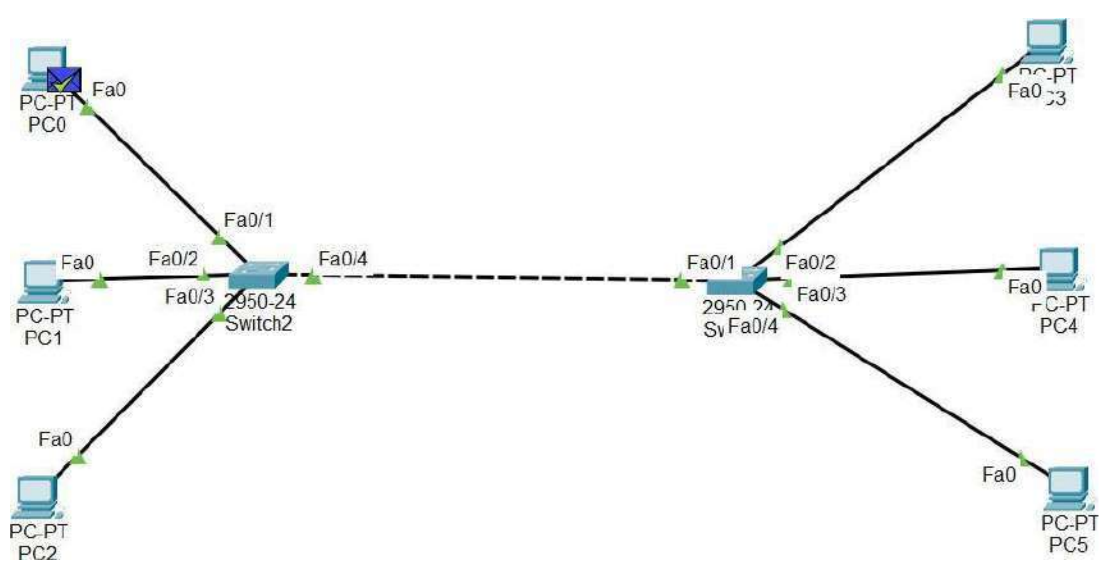


Fig 1. Sending PDU message from PC0 to PC5

PDU List Window

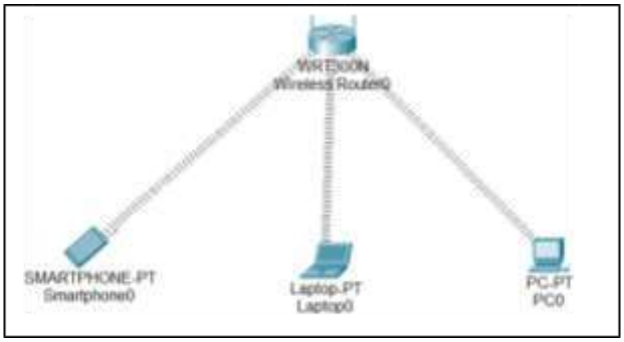
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC5	ICMP		0.000	N	2	(edit)	(delete)
	Successful	PC1	PC3	ICMP		0.000	N	3	(edit)	(delete)
	Successful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

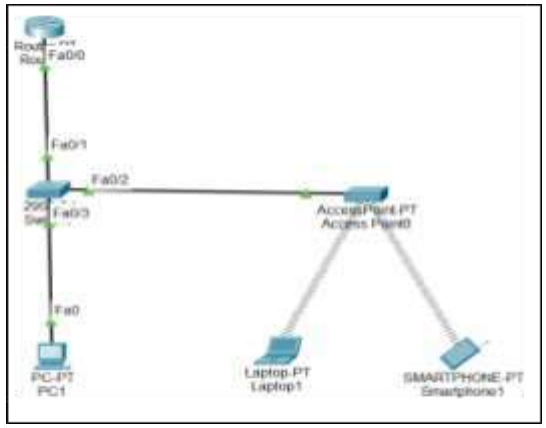
Program 9:

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)

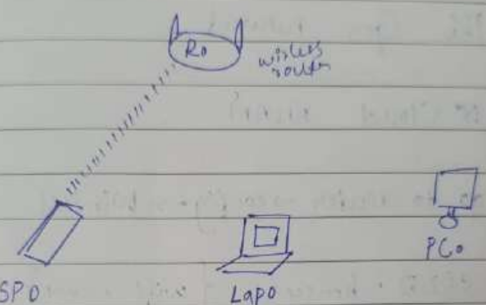
1. Use DHCP (automatic) or manually assign from the same network.

5. Test Wireless Communication

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

Observation:

Q) To construct a WLAN & make the nodes communicate wirelessly.



Smartphone is b always on ~~wifi mode~~ wireless mode

When we place wireless router & smartphone, it is SP automatically connects to R0 wirelessly

Laptop & PC are in ethernet mode by default

Go Laptop → physical -

Turn off
drag out ethernet port
drag in wifi port
Turn ON

PC
same as laptop

Once we do this LAP & PC are also wirelessly connected to R0

This Open network

* ~~N~~ Closed WLAN

go to router → config → Wireless

SSID : bmsce {wifi name}
Authentication : WPA2-PSK
Pass phrase : bms12345

Once we do this all three devices get disconnected

To connect (For all three)
device → config → Wireless
SSID - bmsce
pass - bms12345

~~Send message~~ can be
Message can be sent from any device to other

Output:

1. Do Physical Connections In:

- Laptop
- PC

Fig 1.1 Step1: Turn off light / Power off laptop

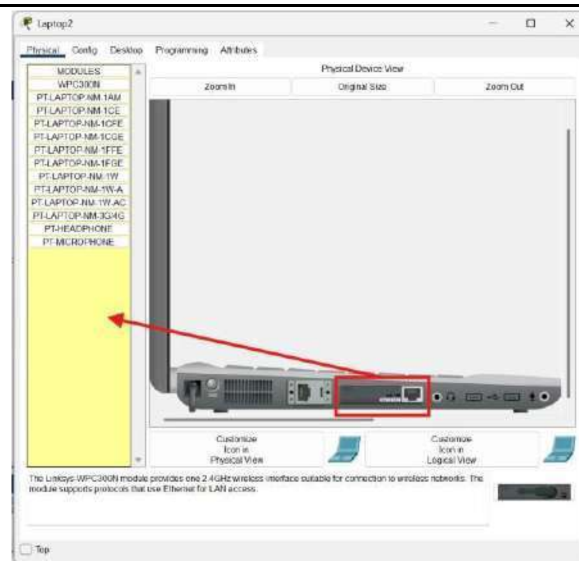


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

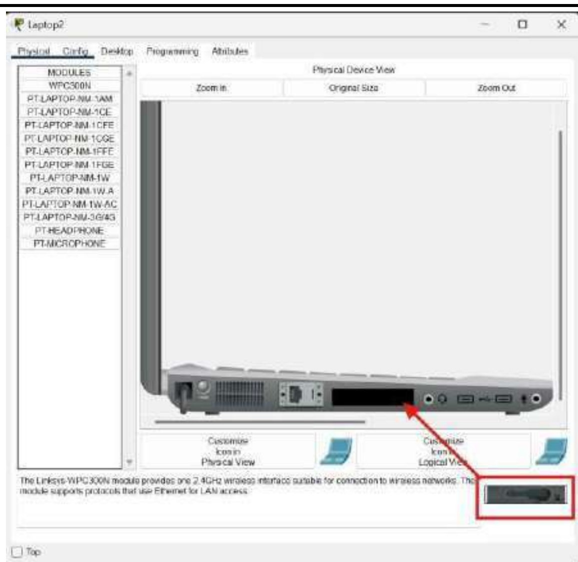


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

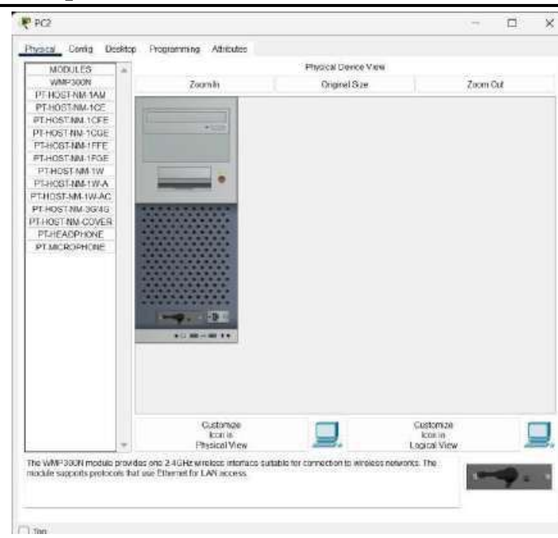


Fig 2. PC physical connection (combined 3 steps)

2. Do Wireless Connection in:

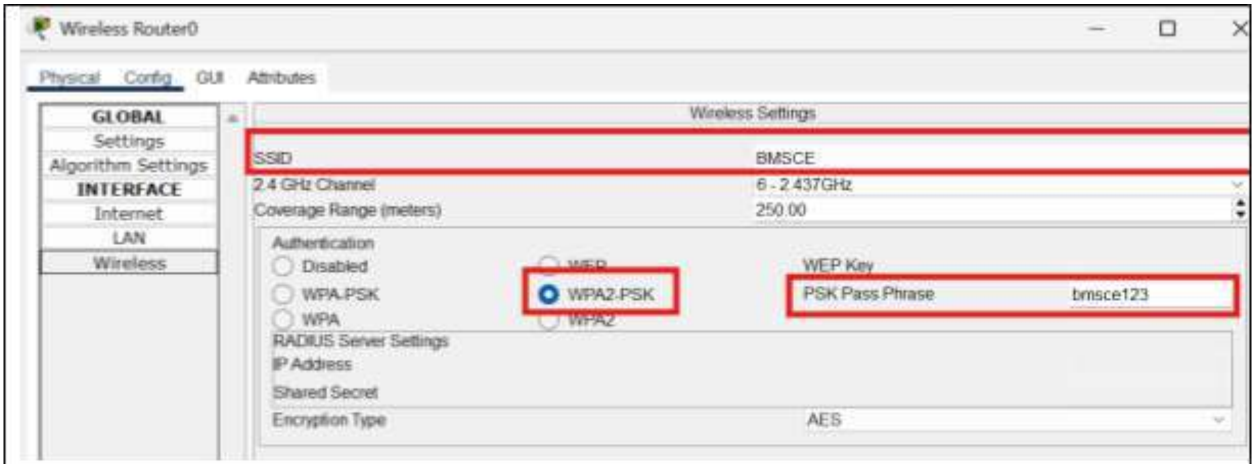


Fig 1. Config at Device Wireless Router0

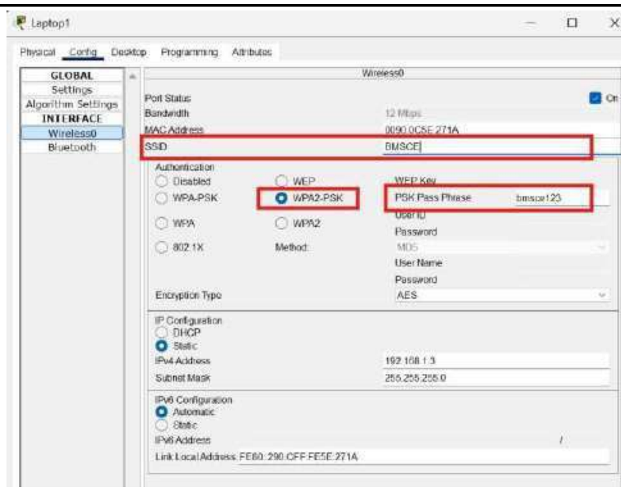


Fig 2. Config at Device Laptop0

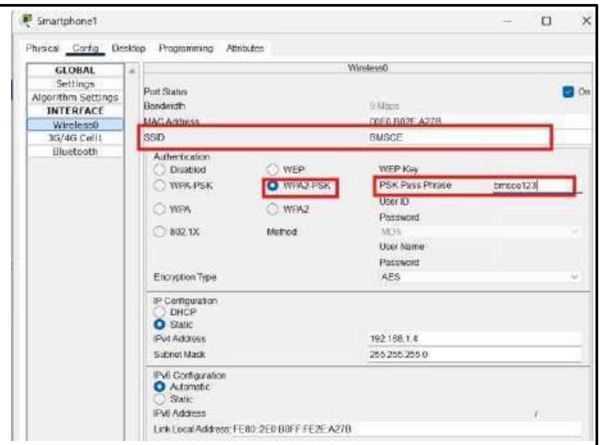


Fig 3. Config at Device Smartphone0

PDU List Window

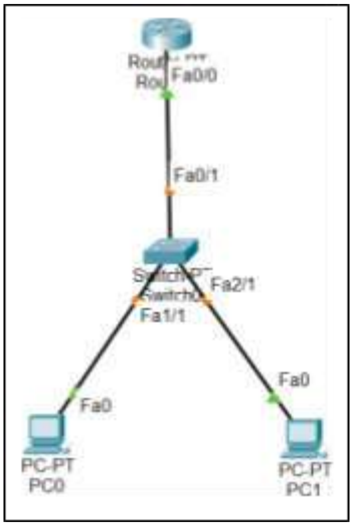
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Failed	Smar...	Laptop0	ICMP		0.000	N	0	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	1	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	2	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	3	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	4	(edit)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

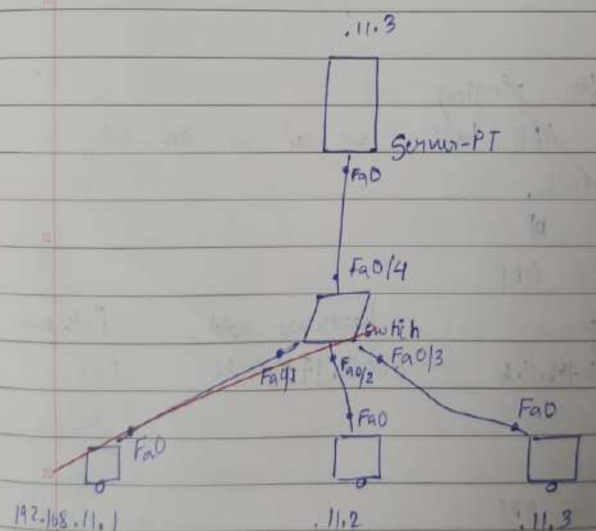
6. Observe TTL Decrement

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

Observation:

Q) Construct simple LAN & understand the concept and operation of ARP (address resolution protocol)

- ARP is used to map IP address to a MAC address
- ARP is used to get data link layer address MAC address with the help of ip address



Configure IP address for all three PCs & server.

select inspect
open ARP table for PC0 & server

- initially both tables are empty

go to PC0 → Command
ping 192.168.11.4 (server ip)

success message

After pinging
In ARP table we can see one entry for each.

O/P
PC0 ARP

ip	hardware add	interface
192.168.11.4	0002.1770.4C3C	Fa0

Server ARP

ip	hardware	Interface
192.168.11.1	0030.A328.6D50	Fa0

Conclusion: Once PDU sent first time MAC address of destination fetched along response PDU.

Output:

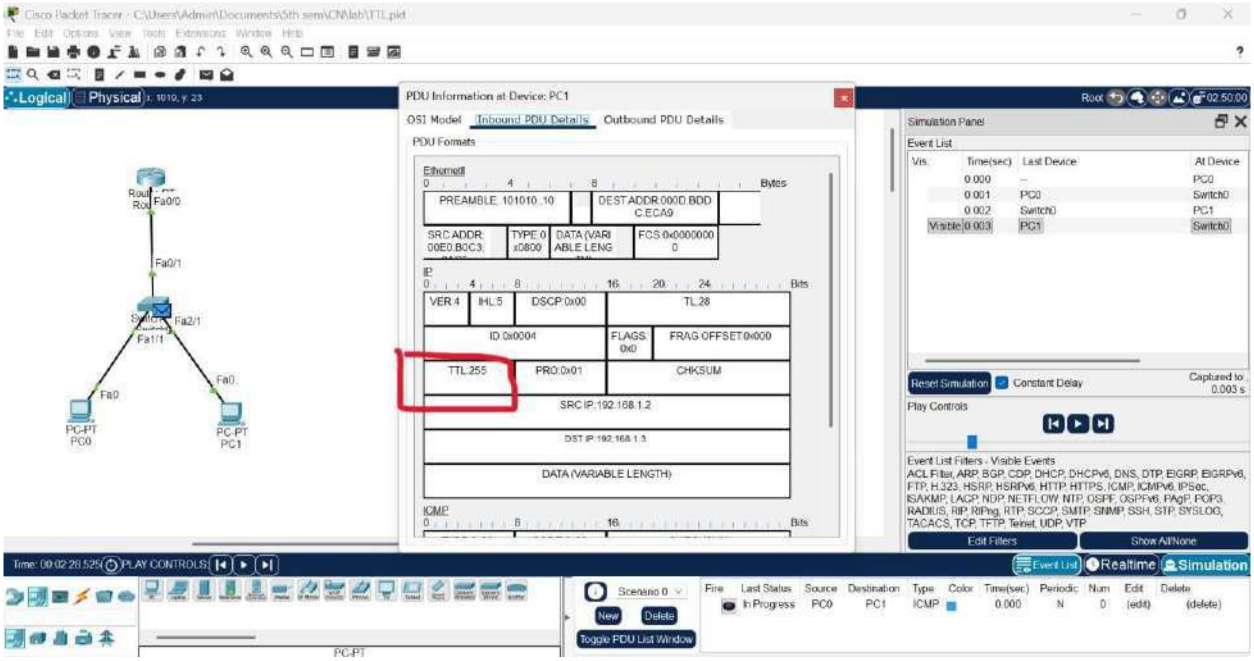


Fig 1. Inbound PDU Details at Device PC1

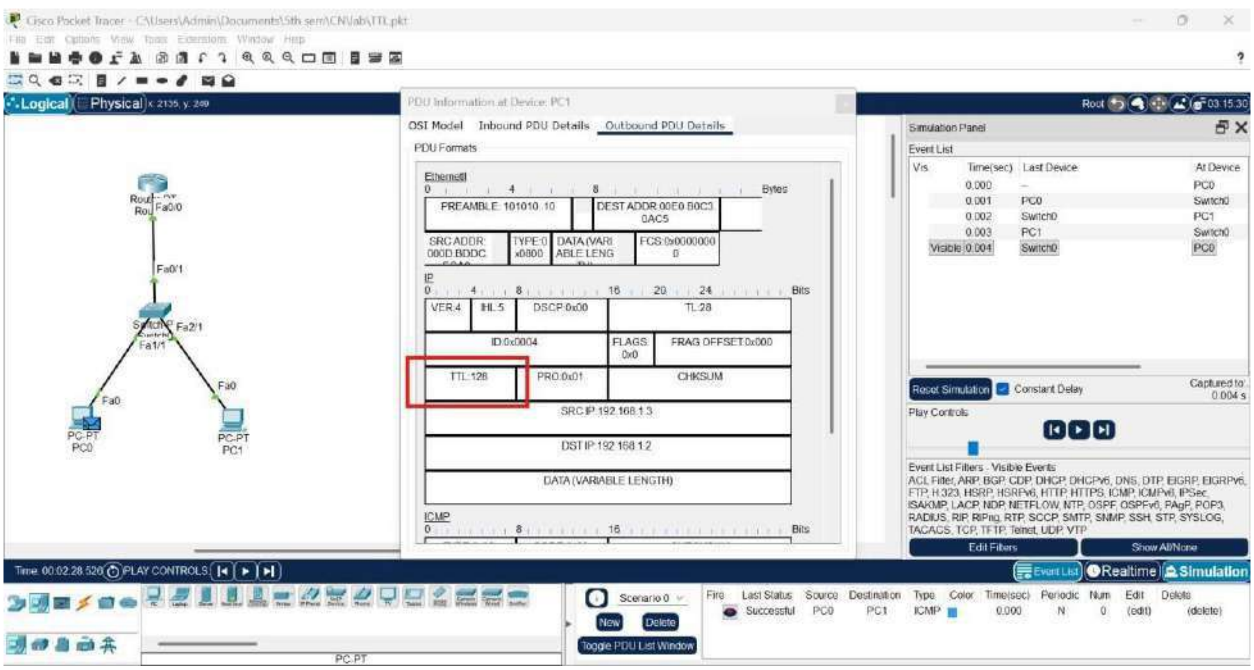
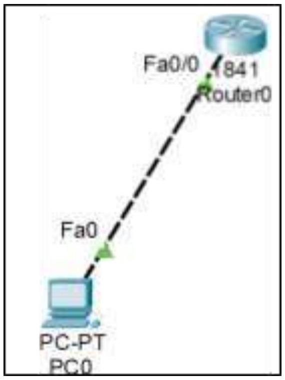


Fig 1. Outbound PDU Details at Device PC1

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

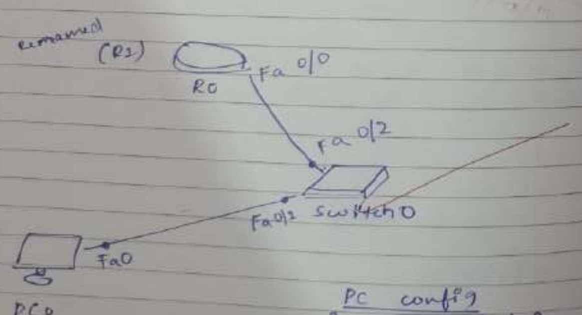
Observation:

Date
9/10/2025

LAB-5

i) Configure telnet to access server/router remotely

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



PC config
ip : 192.168.1.2
default gateway : 192.168.1.1

router 0 config:
go to CLI
- enable
- conf t
- hostname R1 (Renaming it to R1)
- enable secret rp (rp is variable name)
- int Fa 0/0 (interface on router side)
- ip address 192.168.1.1 255.255.255.0
(router ip) config

- no shutdown

(virtual environment with range 0 5)
- line vty 0 5
- login
- password tp (pass set as tp)
- exit
- exit
- wr

- show ip interface brief — command to show all interface state

Now we s for R1 router Fa 0/1 is idle we set ip for it from PC0

PC0 → command prompt
- ping 192.168.1.1 (router ip)
(successful message)

- telnet 192.168.1.1
- password : tp
R1> enable
Pass : rp
R1# conf t
- int Fa0/1
- ip address 192.168.1.4 255.255.255.0
- do show ip interface brief

(you should see Fa 0/1 assigned ip 192.168.1.4)

Add a extra PC1 connected to switch
PC1 config
ip 192.168.1.5
gateway 192.168.1.1

PC1 → Command prompt

- telnet 192.168.1.1
- pass : tp
R1> enable
pass : rp
R1# conf t
- int Fa0/1
- ip address 192.168.1.8 255.255.255.0
- do show ip interface brief

(you'll see Fa 0/1 reassigned to 192.168.1.8)

Output:

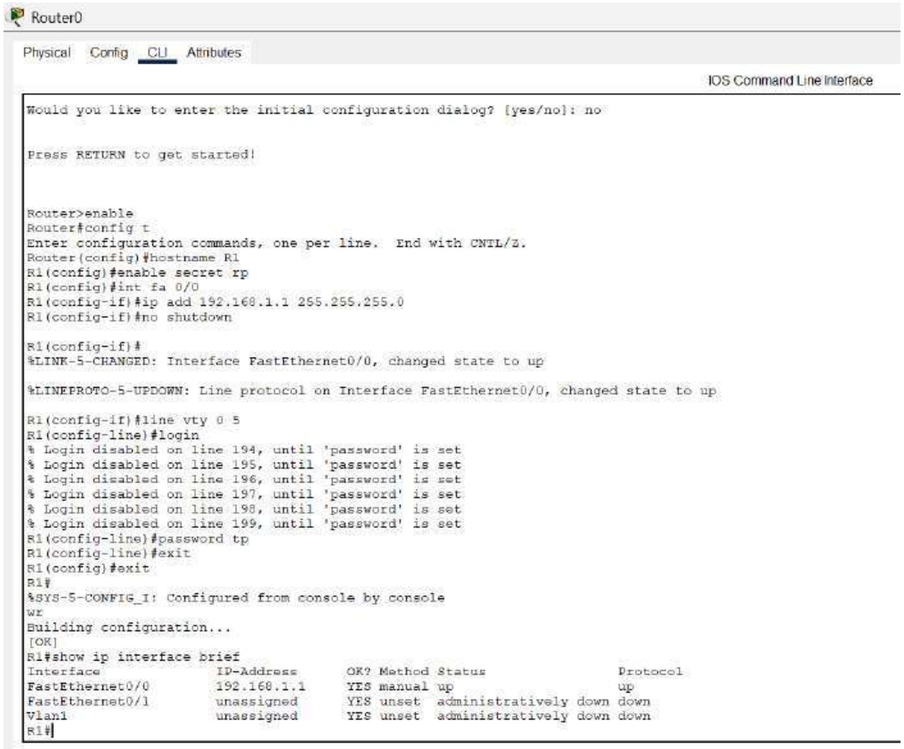


Fig 1. Router0 – CLI commands

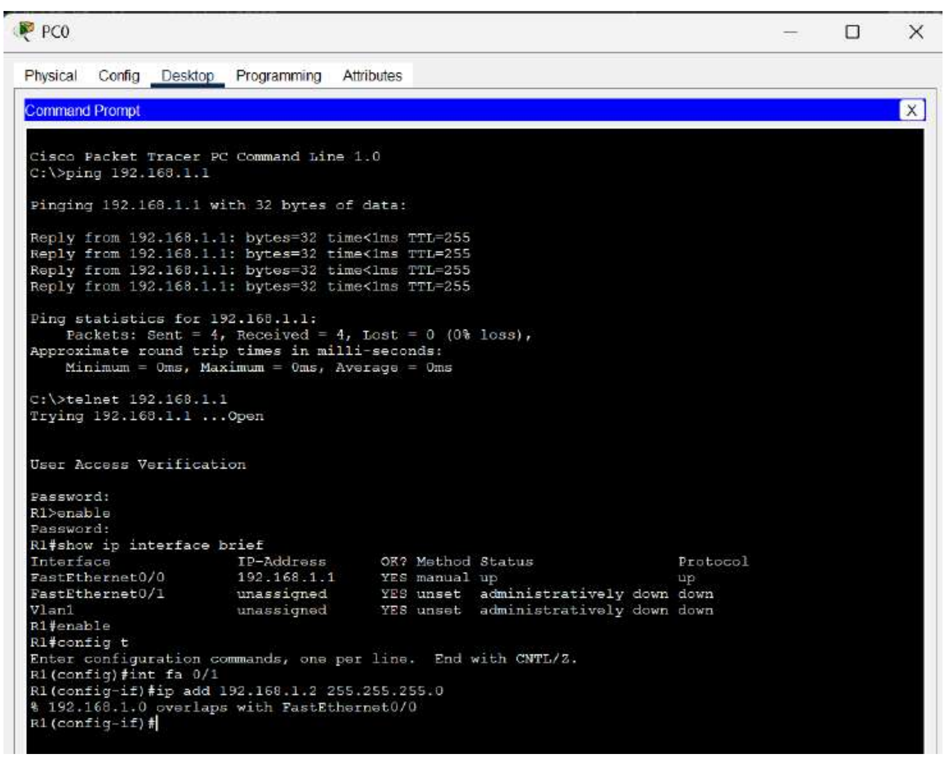


Fig2. PC command line prompt

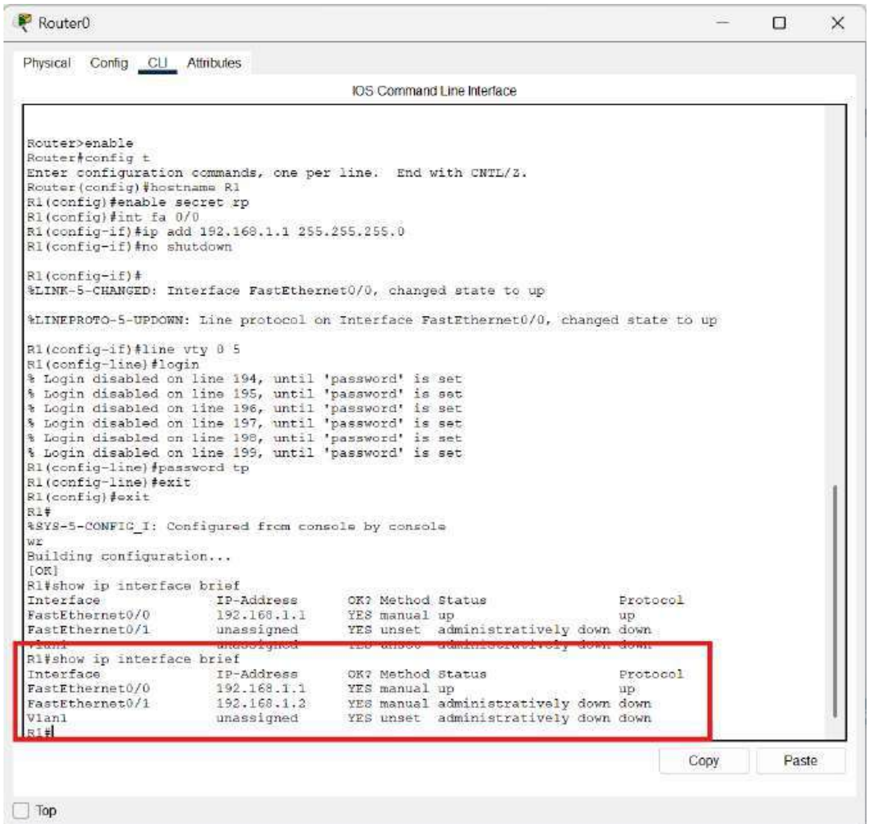


Fig 3. Updated the changes into Router0

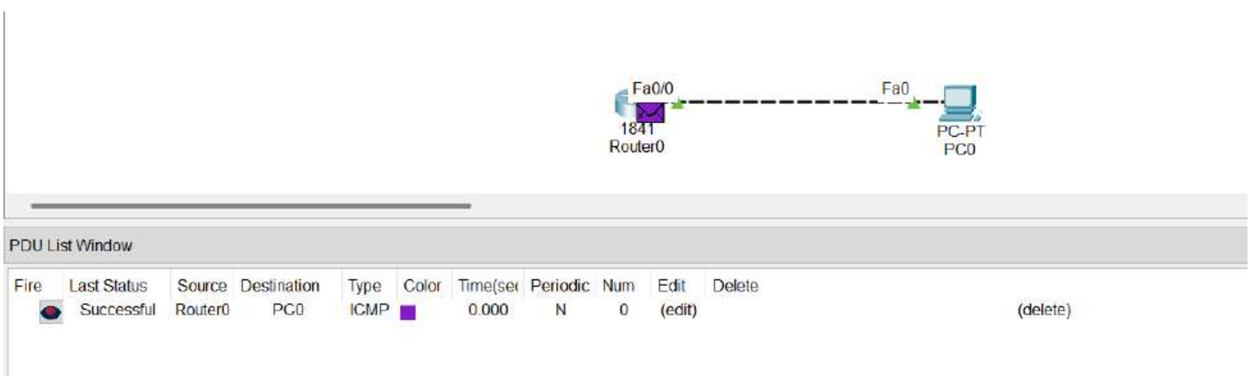
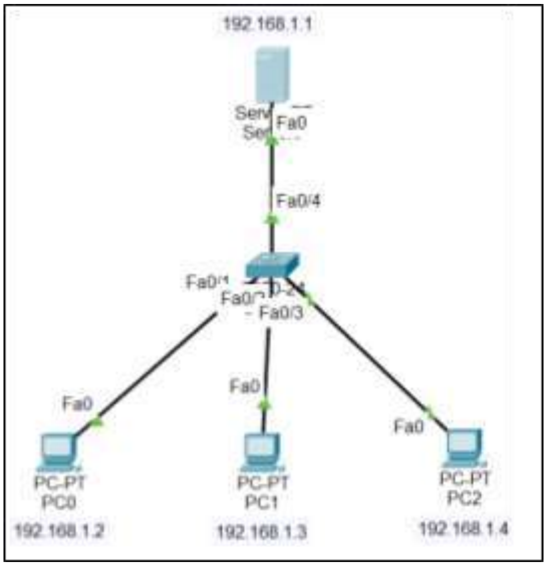


Fig 4. PDU message Successful

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

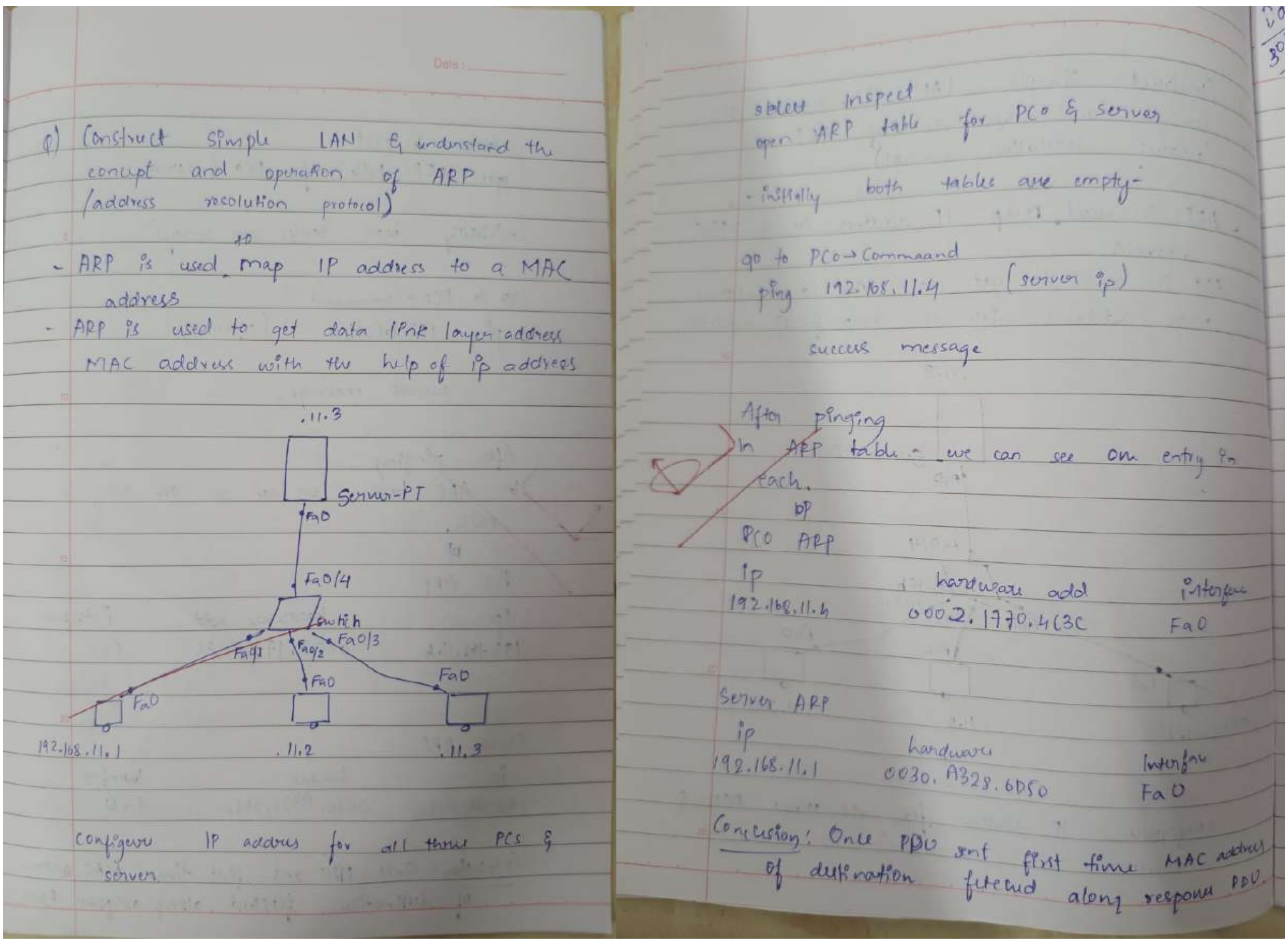
3. PC0 sends an ARP request → switch → server.
4. Server replies with its MAC address.

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:



Date :

Q) Construct simple LAN & understand the concept and operation of ARP (address resolution protocol)

- ARP is used to map IP address to a MAC address
- ARP is used to get data link layer address MAC address with the help of ip address

configure IP address for all three PCs & server.

select inspect
open ARP table for PC0 & server

- initially both tables are empty.

go to PC0 → Command
ping 192.168.11.4 (server ip)

success message

After pinging
In ARP table we can see one entry in each.

op

PC0 ARP

ip	hardware add	interface
192.168.11.4	0002.1770.4C3C	Fa0

Server ARP

ip	hardware	Interface
192.168.11.1	0030.A328.6D50	Fa0

Conclusion: Once PDU sent first time MAC address of destination fetched along response PDU.

Output:

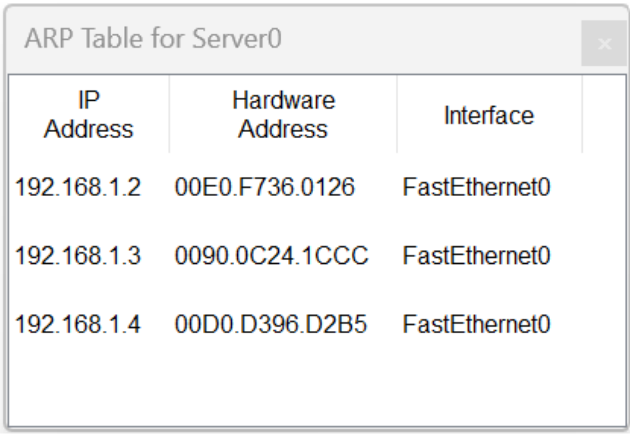


Fig 1.1 ARP table at Server0

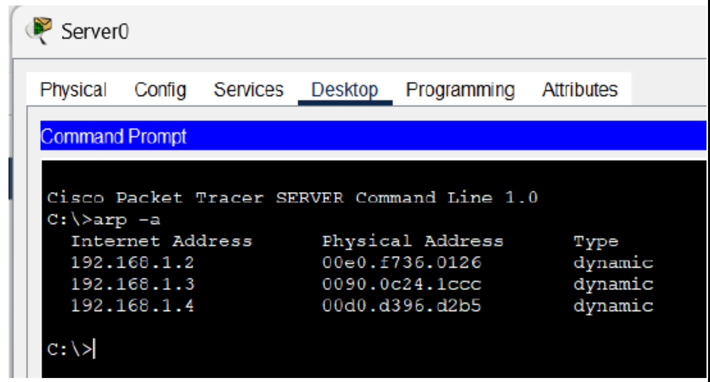


Fig 1.2 Command Prompt at Server0

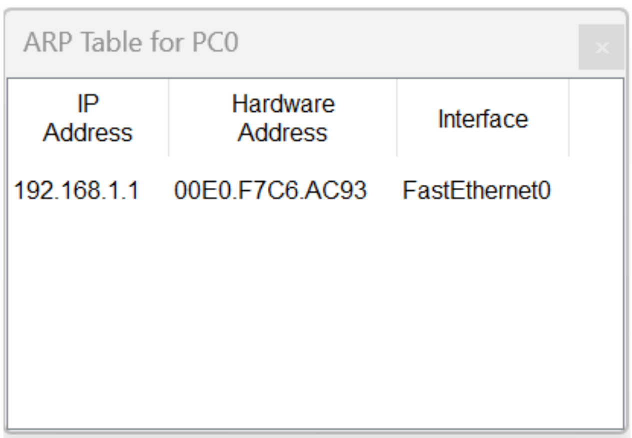


Fig 2.1 ARP table at PC0

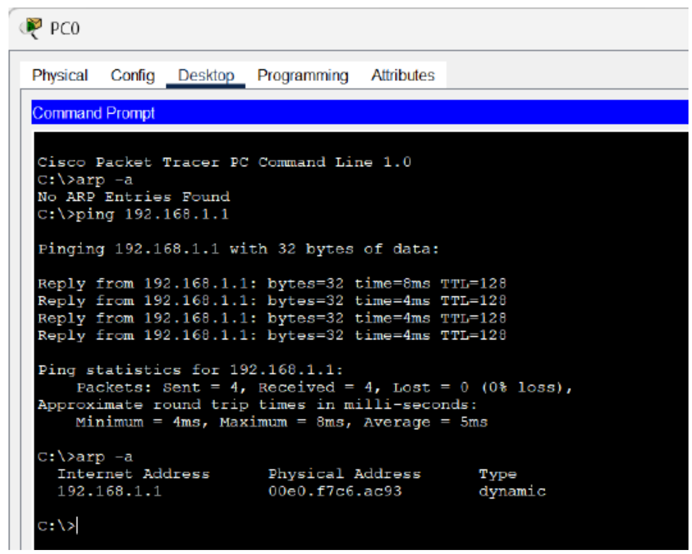


Fig 2.2 Command Prompt at PC0

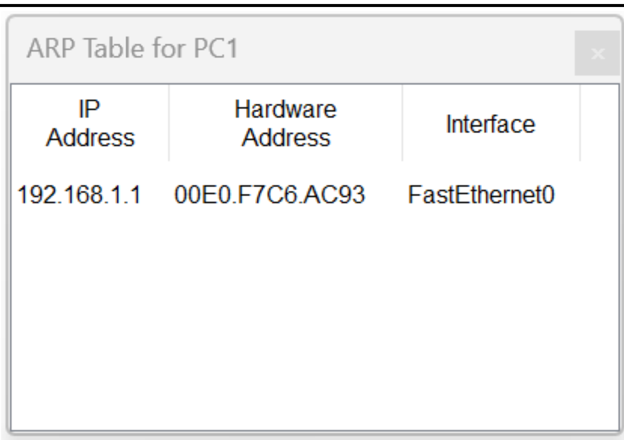


Fig 3.1 ARP table at PC1

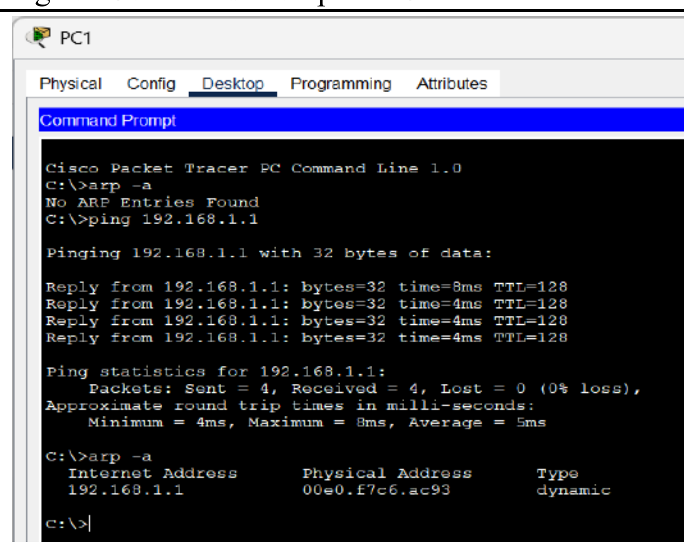


Fig 3.2 Command Prompt at PC1

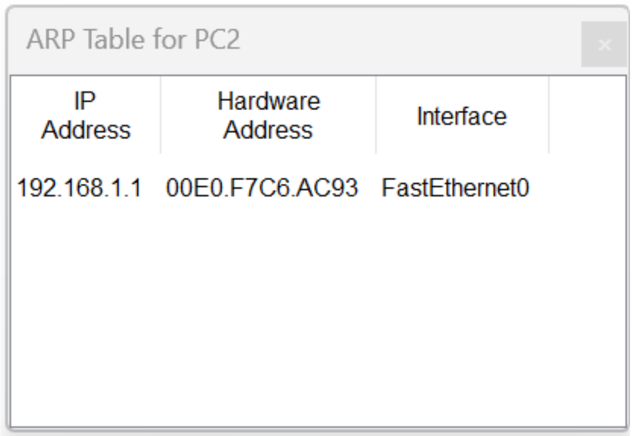


Fig 4.1 ARP table at PC2

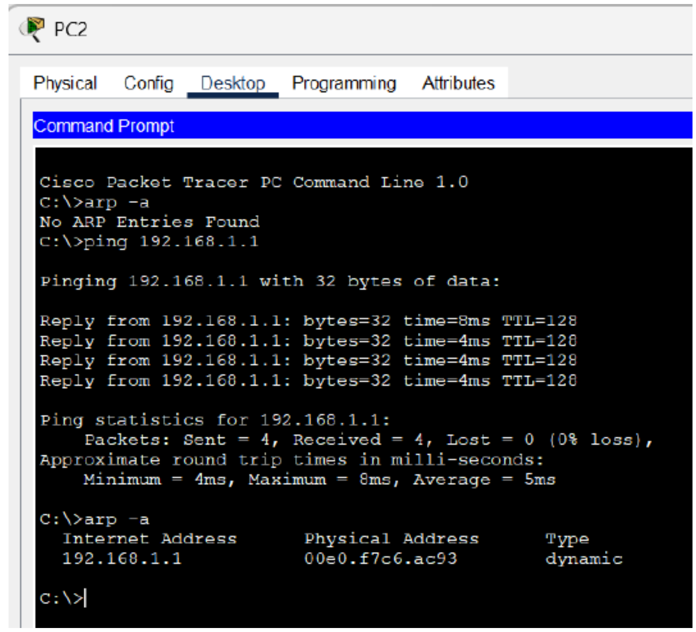


Fig 4.2 Command Prompt at PC2

PART - B

Program 1:

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

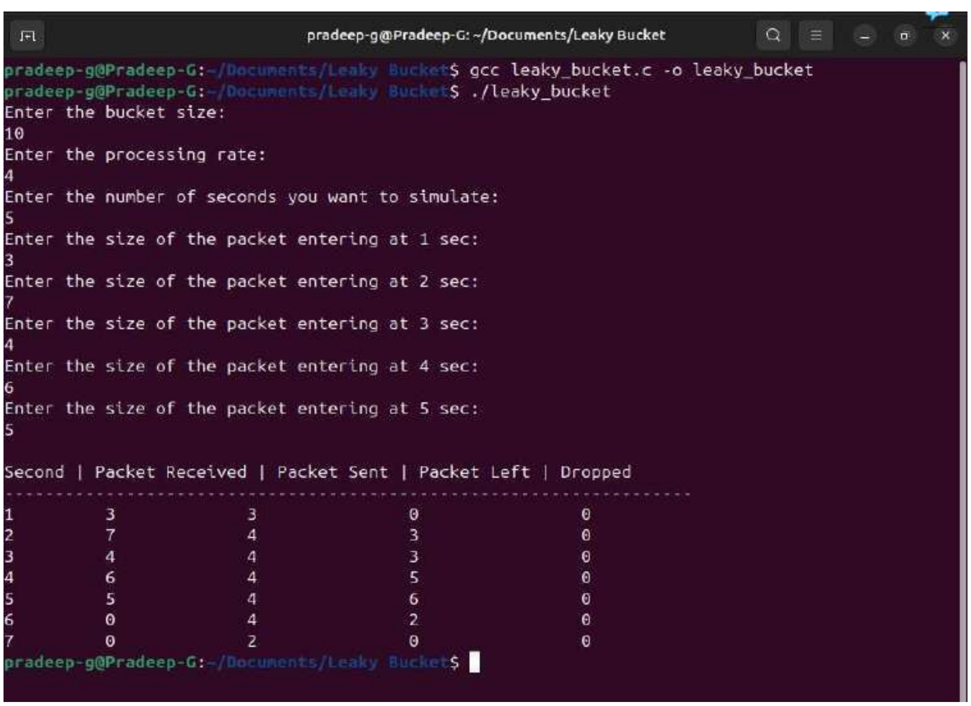
Code:

```
#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:

Date 13/11/25

Part-B

a) WAP for congestion control using leaky bucket algorithm

Bucket size : 10 packets (max bucket cap)
output rate : 1 packet / sec

Packet arrive as follows:

Time (s)	packets arriving
0	6
1	4
2	8
3	1
4	0

Algorithm :

input : bucket_cap, leak_rate, packet arrivals []
output: Table of time-wise processing

For each packets in packet_arrival so
    time = time + 1
    total_arrived = total_arrived + packets

bucket_content = bucket_content + packets

If bucket_content > bucket_capacity then
    dropped = bucket_content - bucket_capacity
    bucket_content = bucket_capacity
Else
    dropped = 0
ENDIF

before_leak = bucket_content

If bucket_content ≥ leak_rate THEN
    leaked = leak_rate
    bucket_content = bucket_content - leak_rate
Else
    leaked = bucket_content
    bucket_content = 0

total_dropped += dropped
total_leaked += leaked

Output :-

Enter the leak rate (packets/second) : 3
Enter the bucket capacity : 10
Enter number of seconds to simulate : 5
Enter number of packets arriving each second:
Sec 1 : 5
2 : 7
3 : 4
4 : 6
5 : 2

time	packet arriving	packet in Bucket (BL)	leaked	Bucket after leak	packet dropped
1	5	5	3	2	0
2	7	9	3	6	0
3	4	10	3	7	1
4	6	13	3	7	3
5	2	9	3	6	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:

```python
# 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()
```

```python
# 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:

Date :

Q) AIM: client - server program using TCP/IP sockets. to make client sending the filename and the server to send back the contents of the requested file if present.

client program

```
import java.io.*;
import java.net.*;
class ~~tcp~~ client {
    public static void main(String arg[]) throws Exception {
        Socket csock = new socket(host: "127.0.0.1",
                                  port: 4000)
        system.out.println("connection successful");
        system.out.println("Enter file name:");
        BufferedReader br = new BufferedReader(new
                     InputStreamReader(system.in));

        OutputStream ostream = client Scocket. get OutputStream();
        PrintWriter pwrite = new PrintWriter(ostream, true);
        pwrite.println(fname);
```

```
while( (str= socketRead.readLine())!= null){
        system.out.println(str);
  }
  socketRead.close();
  pwrite.close();
  clientSocket.close();
 }
}
```

Server Program

```
import java.io.*;
import java.net.*;

class server {
    public static void main(String args[])throws
                                    Exception {
        ServerSocket sersock= new ServerSocket(4000);
        sout ("server ready for connection...");

        Socket sock = sersock.accept();
        sout ("Connection established.");
        sout ("Waiting for file name...");
InputStream istream= sock.getInputStream();
BufferedReader br= new BufferedReader(
                new InputStreamReader(istream));
String fname= br.readLine();
OutputStream ostream= sock.getOutputStream();
PrintWriter pwrite = new PrintWriter(ostream, true);

String str;
try {
    BufferedReader contentRead = new BufferedReader
                        (new FileReader(fname));

    while((str= contentRead.readLine())!= null){
            pwrite . println(str);
    }
    sout ("File content sent successfully.");
    contentRead.close();
}catch (FileNotFoundException e){
        pwrite.println("File not found at server");
        sout ("File not found:"+fname);
  }
  br.close();
  pwrite.close();
  sock . close();
  sersock.close();
  sout ("Server closed.");
}}
```

```
cli 1 :-
javac server.java
java server

SERVER READY FOR CONNECTION
CONNECTION IS SUCCESSFUL AND IS
WAITING FOR CHATTING

cli 2 :
java client .

Enter the file name : text.txt
Hello 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:

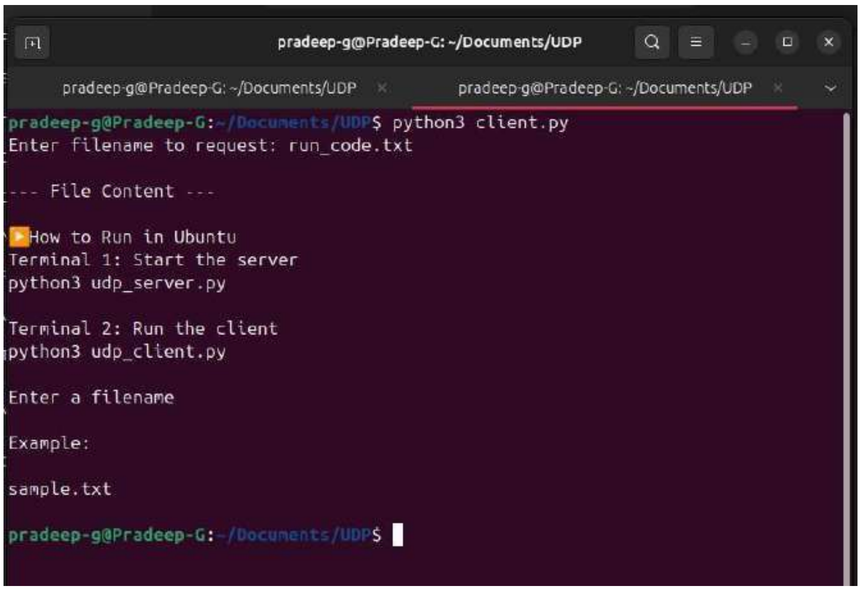
```
# udp_client.py

import socket


# Step 1: Create UDP socket

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


server_address = ('localhost',
8081)


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


# Step 2: Send filename to
server

client_socket.sendto(filename.en
code(), server_address)


# Step 3: Receive response

data, addr =
client_socket.recvfrom(4096)


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

print(data.decode())


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

```
# 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"Fil
e not found on server.", addr)
```

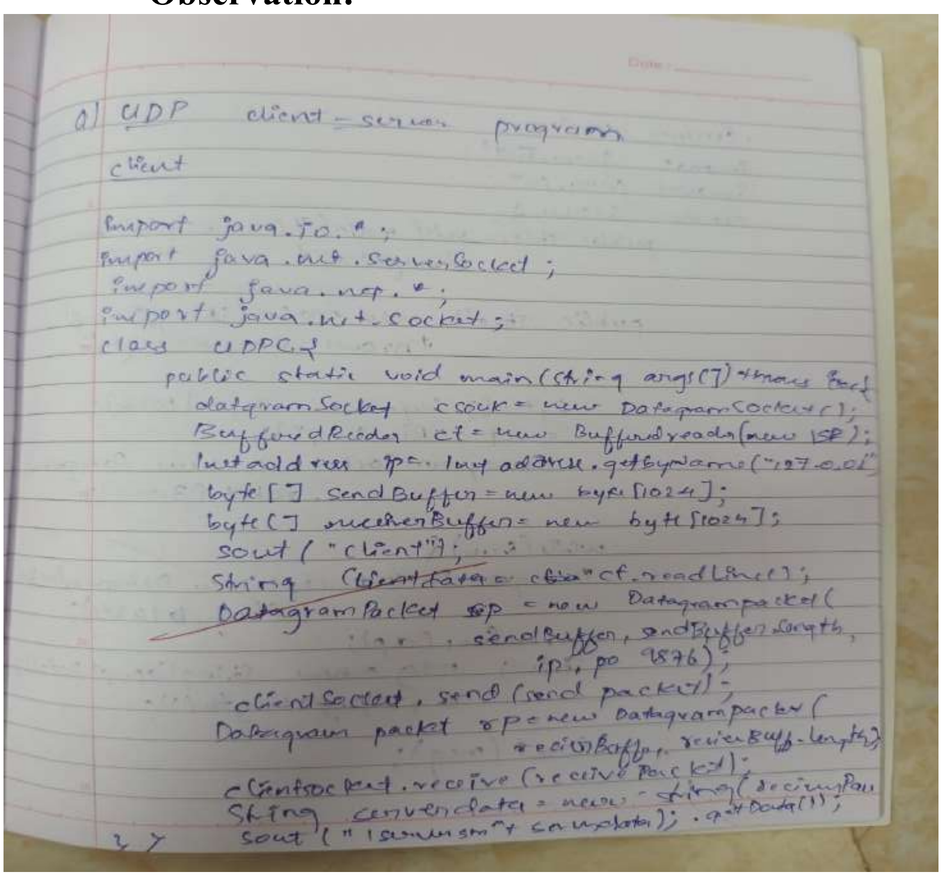
Output:

Server side Terminal:

Client side Terminal:



Observation:



a) UDP client – server program

client

```
import java.io.*;
import java.net.ServerSocket;
import java.net.*;
import java.net.Socket;
class UDPC {
  public static void main(String args[]) throws Exc{
    DatagramSocket csock = new DatagramSocket();
    BufferedReader cf = new BufferedReader(new ISR);
    InetAddress ip = InetAddress.getByName("127.0.0.1"
    byte[] sendBuffer = new byte[1024];
    byte[] receiverBuffer = new byte[1024];
    sout("client");
    String clientData = cf.readLine();
    DatagramPacket sp = new DatagramPacket(
        sendBuffer, sendBuffer.length,
        ip, po 9876);
    clientSocket.send(send packet);
    Datagram packet rp = new Datagrampacket(
        receiveBuffer, receiveBuff.length);
    clientSocket.receive(receivePacket);
    String serverdata = new String(receivePac
        .getData());
    sout("server" + serverdata);
} }
```

Server / Receiver

```
import java.net.*;
import java.io.*;
~~class Server {~~
    ~~public static void main(String args[]) throws
            Exception {~~

    public static void main(String[] args)
                throws Exception {

        byte[] buf = new byte[1024];
        sout ("Receiver");
        DatagramSocket ds = new DatagramS
                    ocket (3000);

     while (true) {
      DatagramPacket dp = new DatagramPacket
                    (buf, 1024);
      ds.receive (dp);
      String msg = new String(dp.getData(),
               0, dp.getLength());

      sout (msg);
   }
}
```

Output:

Cli 1

java sender

Sender
Enter message:
Hi
Hello

Cli 2

java receiver

Receiver

Hi
Hello

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:\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:

Total packets produced = 24
Total packets dropped = 4

4 WAP for error detecting code using CRC-CCITT (16 bits)

CRC_CCITT(data, polynomial = 0x1021, initial_value = 0xFFFF)

crc ← initial_value

For each byte in data do
crc ← crc XOR (byte << 8)

for i ← 1 to 8 do
if (crc AND 0x8000) ≠ 0 then
crc ← (crc << 1) XOR polynomial
Else
crc ← (crc << 1) ~~XOR~~ .

End if

crc ← crc AND 0xFFFF
End for
End for

return crc
End algorithm

Output:

Input = "HELLO"
Final CRC = 0x49D6