VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

PRADEEP G (1BM24CS414)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

September 2025 – January 2026

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Computer Network (23CS5PCCON)" carried out by **Pradeep G (1BM24CS414),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

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

Index

Part - A

Part - B

Github Link:

https://github.com/Pradeepg06/Computer-Networks/tree/CN_lab

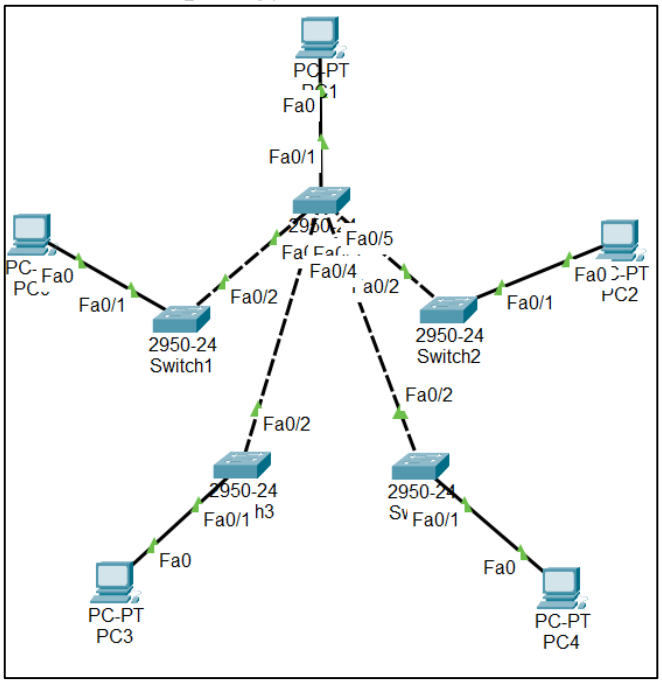
PART - A

Program 1:

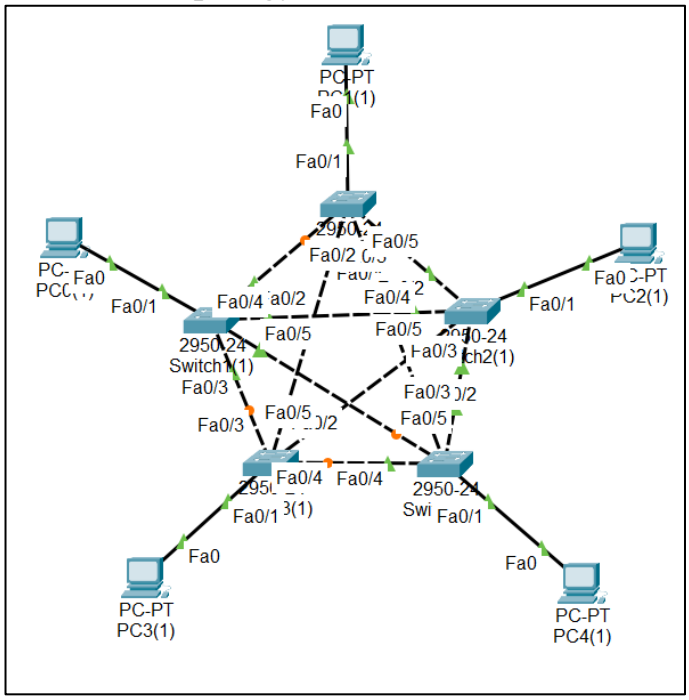
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

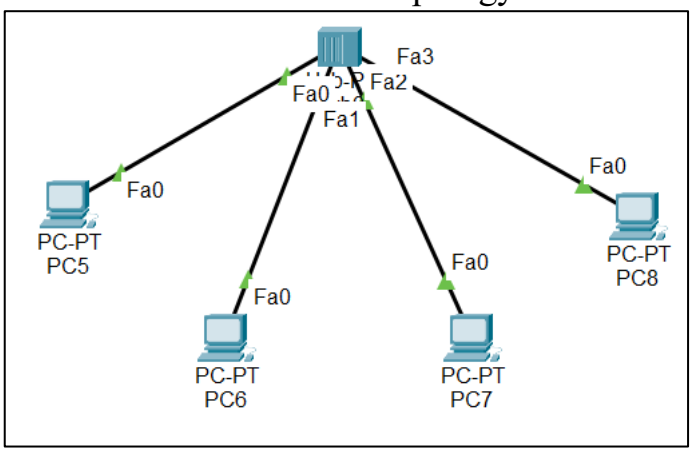
1. STAR Topology with Switch:



2. MESH Topology with Switch:



3. HUB-Based Network Topology:



Configuration:

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. using CISCO Packet Tracer.

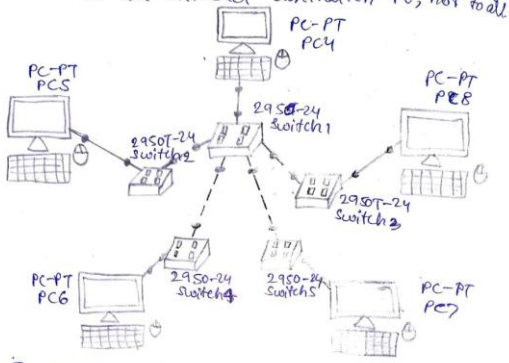
Star Topology using Switch:

① Name: Implementation of Star Topology using cisco Packet Tracer.

- Procedure:
  1. Open a new project in Packet Tracer.
  2. Drag and drop:
     - 1 central switch (Switch 0)
     - 4 Edge Switches (Switch 2, Switch 3, Switch 4, Switch 5)
     - 5 PCs (PC4, PC5, PC6, PC7, PC8)
  3. Connect each PC to its respective switch using copper Straight-Through cables.
  4. Connect all edge switches to the central switch.
  5. Assign IP addresses: [click on PC icon/device → Desktop → IP configuration → enter IP address]
     - PC4 → 192.168.1.2
     - PC5 → 192.168.1.3
     - PC6 → 192.168.1.4
     - PC7 → 192.168.1.5
     - PC8 → 192.168.1.6

     & Subnet mask: 255.255.255.0 is directly entered

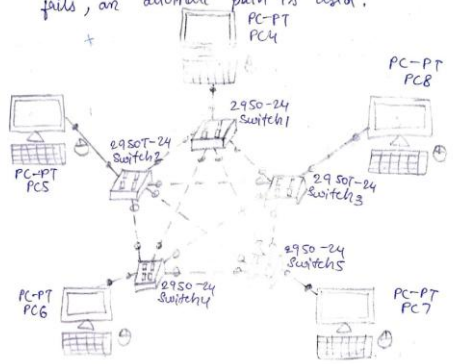
- Simulation:
  1. Switch to Simulation Mode (bottom-right corner).
  2. click the Add simple PDU (envelop icon) tool.
  3. Select source PC → then Destination PC.
  4. In the Event list, a packet creation entry will appear.
  5. Click Auto Capture/Play or Capture/Forward to watch packet flow.
  6. With Switch, the packet travels only to the intended destination PC, not to all.



② MESH Topology using Switches

- Name: Implementation of Mesh Topology using Switches in Cisco Packet Tracer.
- Procedure:
  1. Open a new project in Packet Tracer.
  2. Drag and Drop:
     - 5 switches (Switch1, Switch2, Switch3, Switch4, Switch5)
     - 5 PC's (PC4, PC5, PC6, PC7, PC8)
  3. Connect each PC to a switch.
  4. Interconnect all switches (mesh connection).
  5. Assign IP addresses:
     - PC4 → 192.168.1.2
     - PC5 → 192.168.1.3
     - PC6 → 192.168.1.4
     - PC7 → 192.168.1.5
     - PC8 → 192.168.1.6

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



③ Hub-Based Network Topology:

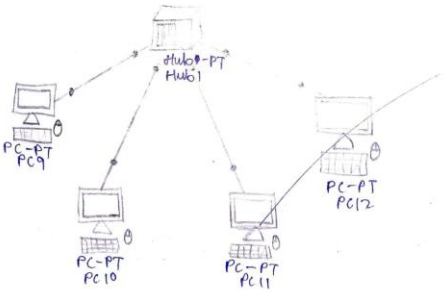
Name: Implementation of Hub-Based Network Topology in Cisco Packet Tracer.

Procedure:

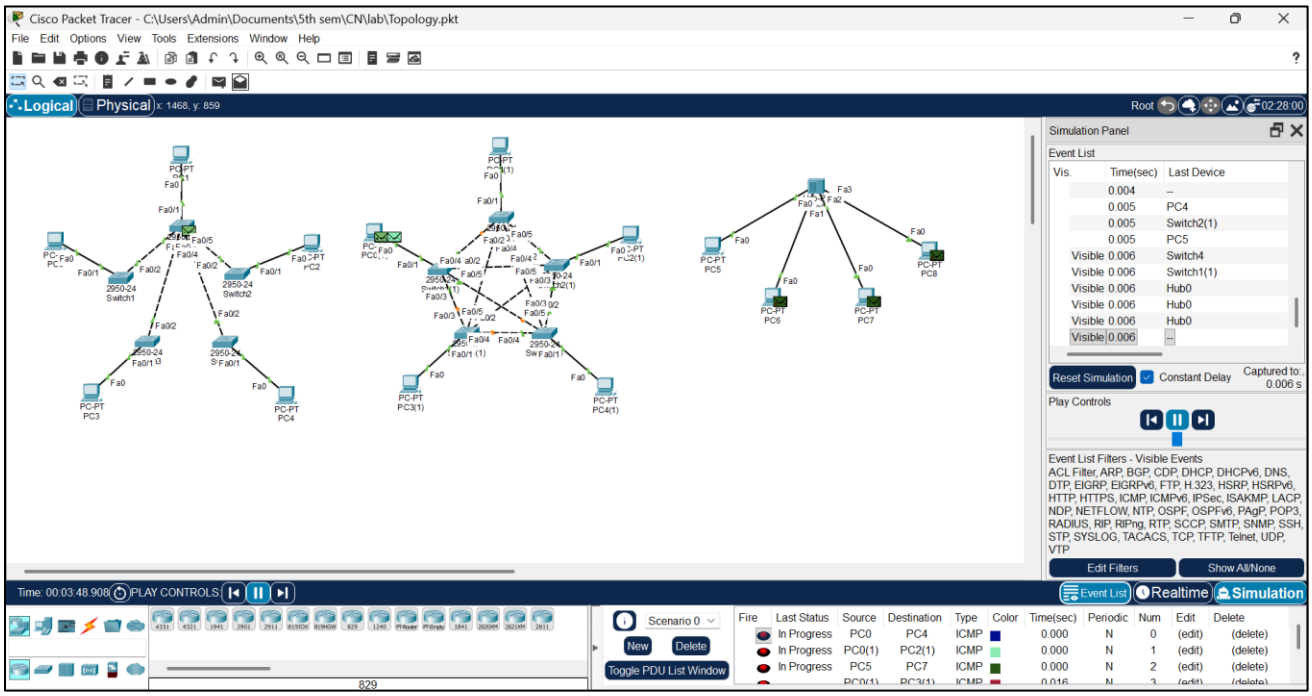
1. Open a new project in Packet Tracer.
2. Drag and drop:
   - 1 Hub (Hub1)
   - 4 PCs (PC9, PC10, PC11, PC12)
3. Connect all PCs to the hub using copper-straight-through cables.
4. Assign IP addresses:
   - PC9 → 192.168.1.2
   - PC10 → 192.168.1.3
   - PC11 → 192.168.1.4
   - PC12 → 192.168.1.5

   (Subnet Mask: 255.255.255.0)

- Simulation:
  1. Switch to Simulation Mode
  2. click the Add Simple PDU tool.
  3. choose Source PC → then Destination PC
  4. Check the Event List for packet creation.
  5. Use Auto Capture/Play or Capture/Forward to watch transmission.
  6. With a Hub, the packet is broadcast to all devices, but only the destination PC accepts it while others discard it.



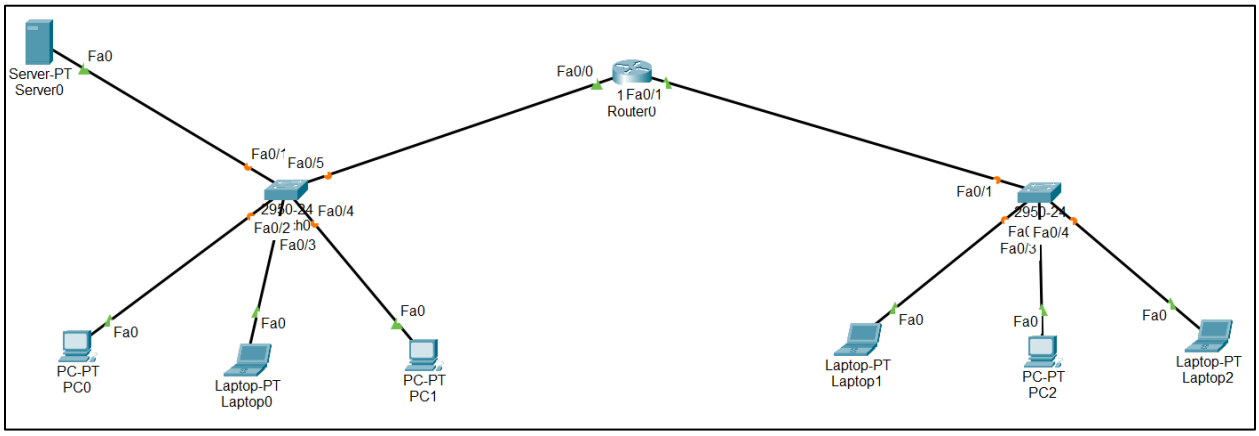
Output:


Cisco Packet Tracer - C:\Users\Admin\Documents\5th sem\CN\lab\Topology.pkt
File Edit Options View Tools Extensions Window Help
Logical Physical
Simulation Panel
Event List
Vis. Time(sec) Last Device
0.004 --
0.005 PC4
0.005 Switch2(1)
0.005 PC5
Visible 0.006 Switch4
Visible 0.006 Switch1(1)
Visible 0.006 Hub0
Visible 0.006 Hub0
Visible 0.006 Hub0
Visible 0.006 --
Reset Simulation
Constant Delay
Captured to: 0.006 s
Play Controls
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:03:48.908
PLAY CONTROLS
Event List
Realtime
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete
In Progress PC0 PC4 ICMP 0.000 N 0 (edit) (delete)
In Progress PC0(1) PC2(1) ICMP 0.000 N 1 (edit) (delete)
In Progress PC5 PC7 ICMP 0.000 N 2 (edit) (delete)

Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

9/9/25 program 2

* Configure DHCP within a LAN & outside a LAN

Procedure:

① click on Server

② Go to desktop → IP configuration → select static & give (192.168.10.2) → & give Gateway (192.168.10.1)

③ Go to Services → DHCP → Disktop IP config →
Static → 192.168.10.2
Gateway → 192.168.10.1

④ Services → DHCP → Pool Name : SwitchOne
Gateway : 192.168.10.1
Start IPadd : 192.168.10.3
Subnet mask : 255.255.255.0

Pool Name : switch two
Gate way : 192.168.20.1
Start IP add : 192.168.20.2
Subnet mask : 255.255.255.0

Add

Add

⑤ Router → CLI → no → enter.

Router # enable
conf t
① # int Fa0/0
ip address 192.168.10.1 255.255.255.0
ip helper-address 192.168.10.2
no shutdown
do write memory
exit

② # int Fa 0/1
IP address 192.168.20.1 255.255.255.0
ip helper-address 192.168.10.2
no shutdown
exit
do write memory.

Output:

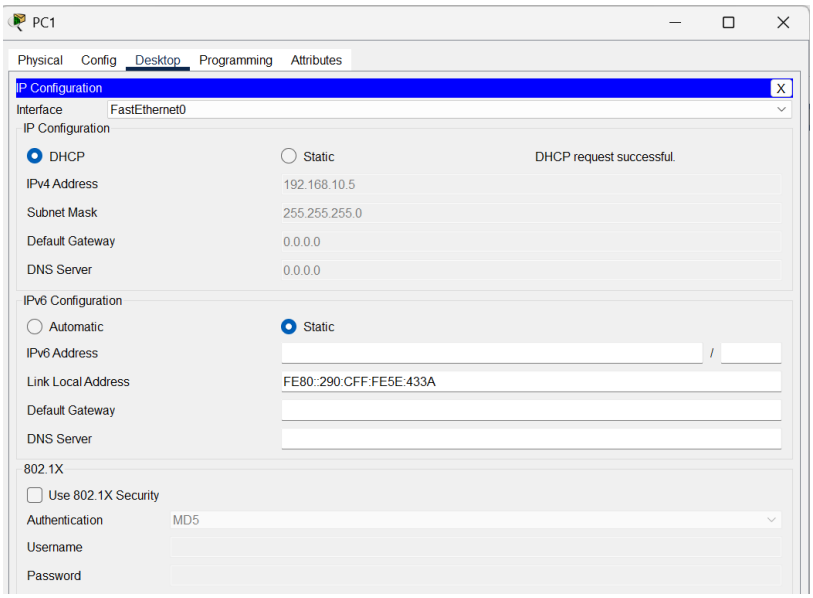


Fig 1. Ip address assigned by DHCP server within Lan (PC1)

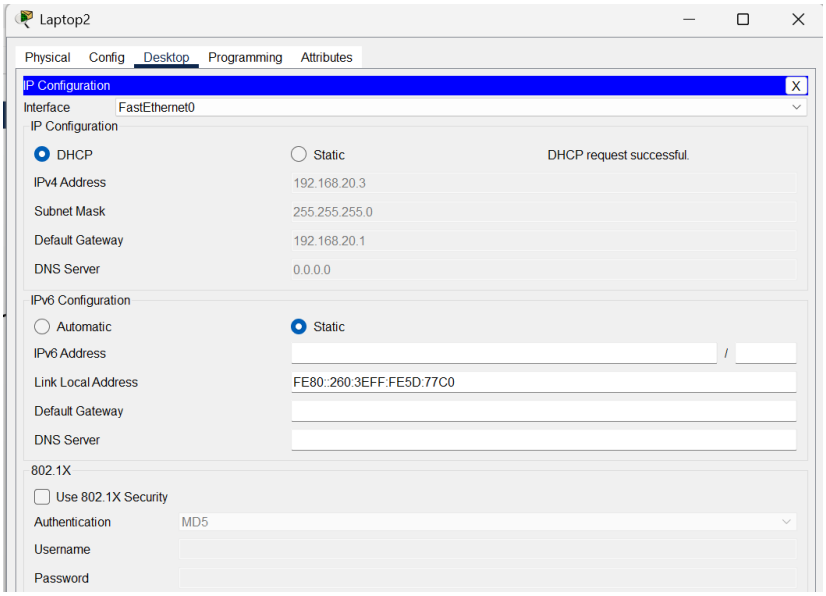
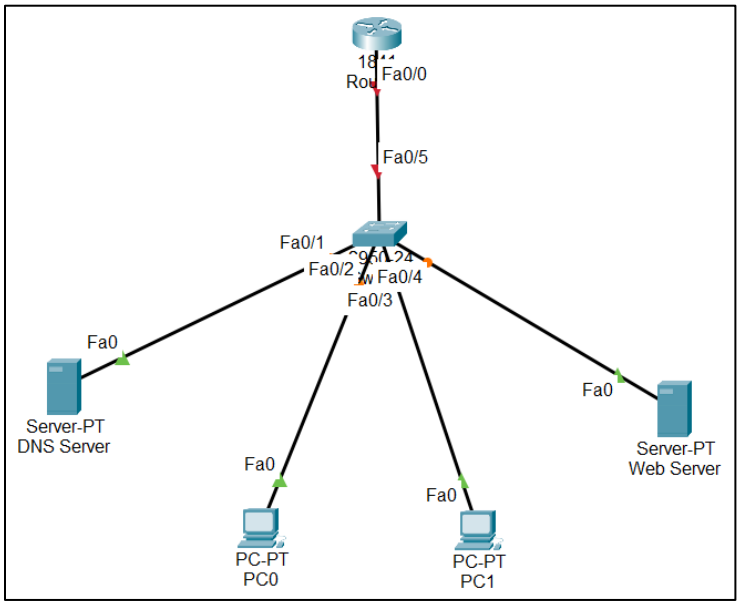


Fig 2. Ip address assigned by DHCP server outside Lan (laptop2)

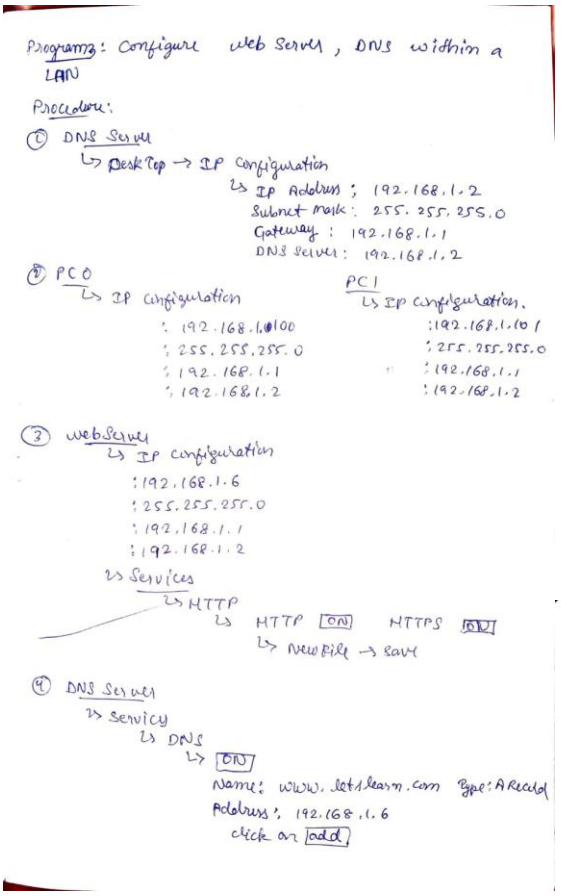
Program 3:

Aim: Configure Web Server, DNS within a LAN.

Network diagram:



Configuration:


Program: Configure web server, DNS within a LAN

Procedure:

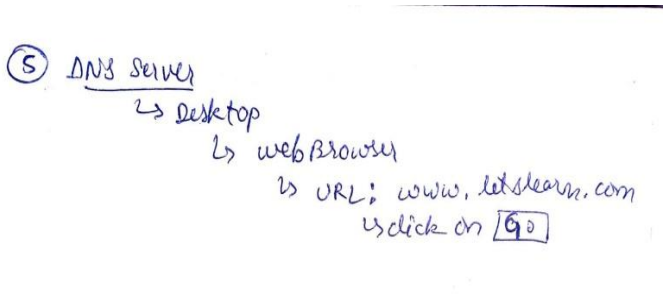
① DNS Server
↳ Desktop → IP Configuration
↳ IP Address ; 192.168.1.2
Subnet mask : 255.255.255.0
Gateway : 192.168.1.1
DNS Server : 192.168.1.2

② PC0
↳ IP configuration
: 192.168.1.100
: 255.255.255.0
: 192.168.1.1
: 192.168.1.2

PC1
↳ IP configuration
:192.168.1.101
: 255.255.255.0
: 192.168.1.1
: 192.168.1.2

③ webServer
↳ IP configuration
:192.168.1.6
: 255.255.255.0
: 192.168.1.1
: 192.168.1.2
↳ Services
↳ HTTP
↳ HTTP [ON] HTTPS [ON]
↳ New File → Save

④ DNS Server
↳ Service
↳ DNS
↳ [ON]
Name: www.letslearn.com Type: A Record
Address: 192.168.1.6
click on [add]


⑤ DNS Server
↳ Desktop
↳ webBrowser
↳ URL: www.letslearn.com
↳ click on [Go]

Output:

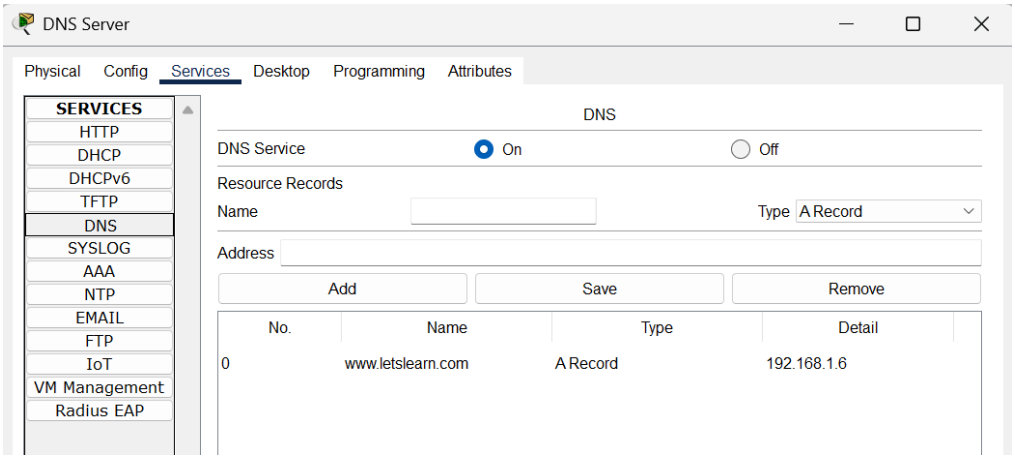


Fig 1. DNS server – DNS Services

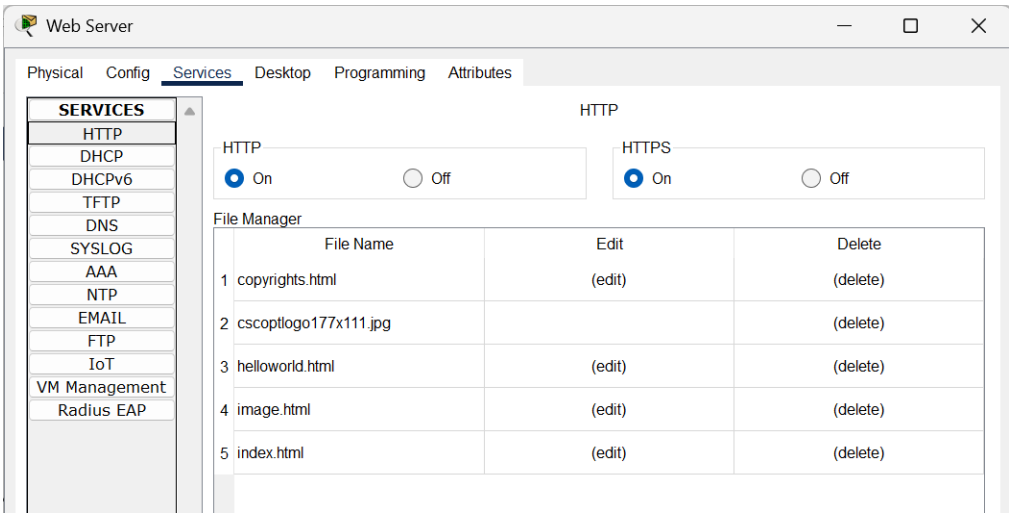


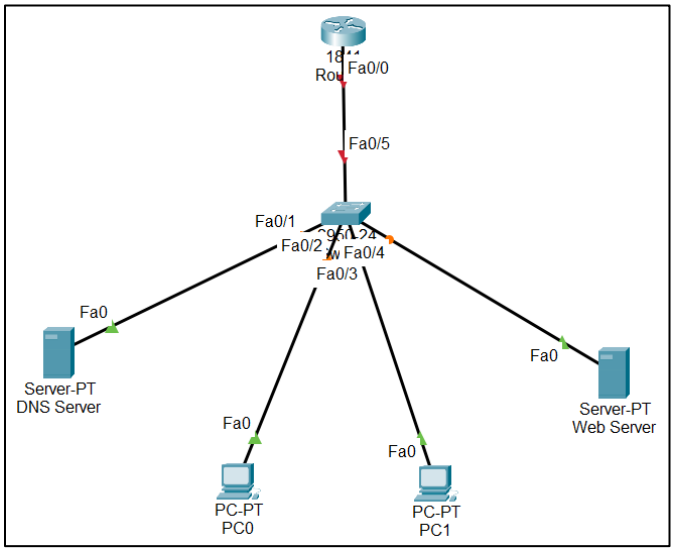
Fig 2. WEB server – HTTP Services

Fig 3. PC0 – accessing data from web browser

Program 4:

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

Network diagram:



Configuration:

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

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

Aim:

Network Connection (Diagram):

Same as prev experiment (DNS – Web Servers)

Procedure

1. Assign IP address as follow

Device	Interface	IP Address	Subnet mask	Gateway
Router0 (Fa0/0)	—	192.168.1.1	255.255.255.0	—
PC0	Fa0	192.168.1.10	255.255.255.0	192.168.1.1
PC1	Fa0	192.168.1.20	255.255.255.0	192.168.1.1
DNS Server	Fa0	192.168.1.100	255.255.255.0	192.168.1.1
Web Server	Fa0	192.168.1.200	255.255.255.0	192.168.1.1

2. Configure Router interface

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

3. Save Configuration:

Router# write

4. Configure IP & Default Gateway in PCs/Servers (Desktop → IP configuration)
5. Test connectivity using ping command from PCs.
6. Change conditions (wrong IP, wrong gateway, shut interface, power off device) to observe diff ping messages.

Observations:

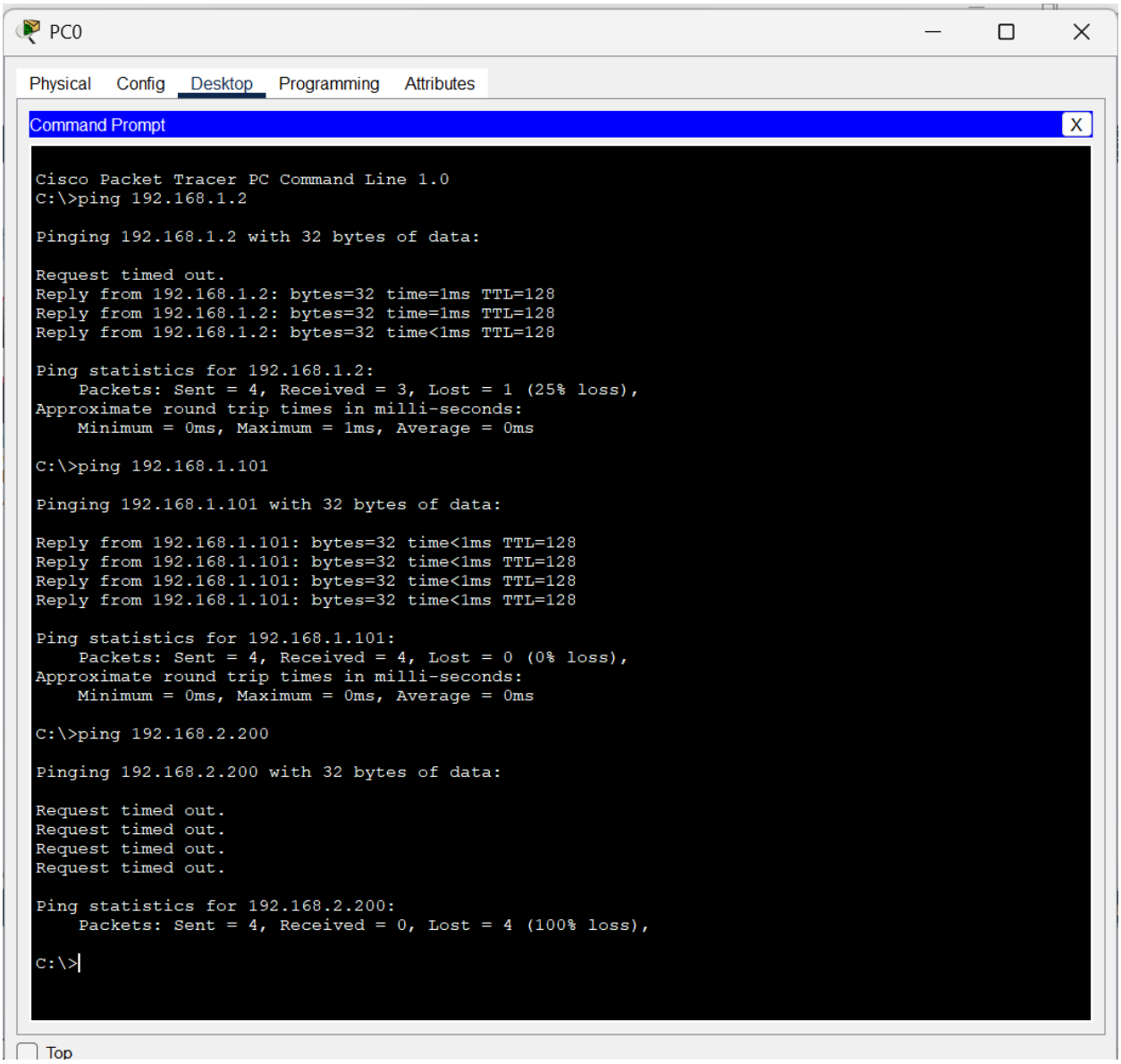
Case 1 : Ping Response
Ping Command : ping 192.168.1.20
(PC0 → PC1 in same network)
Message observed : Ping Response
Reason : ICMP Echo Request and Echo reply exchanged successfully b/w two active devices.

Case 2 : Reply
Ping Command : ping 192.168.1.100
(PC0 → DNS Server)
Message observed : Reply from 192.168.1.100
Reason : Destination device is active, reachable & properly configured.

Case 3 : Destination Unreachable
Ping Command : ping 192.168.1.200
(towards webserver)
First Remove Gateway on PC0
Message observed : Destination Host Unreachable
Reason : Router cannot be reached due to missing/incorrect gateway, so no route exists.

Case 4 : Request timed out
Ping Command : ping 192.168.1.1500
(non-existent device)
Message observed : Request timed out
Reason : No reply received since IP does not exist/device is off.

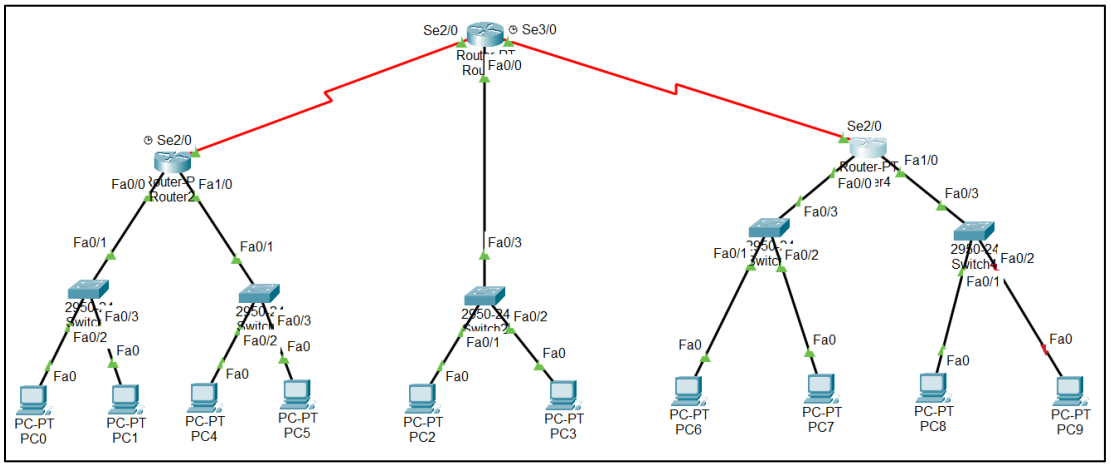
Output:


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2
Pinging 192.168.1.2 with 32 bytes of data:
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

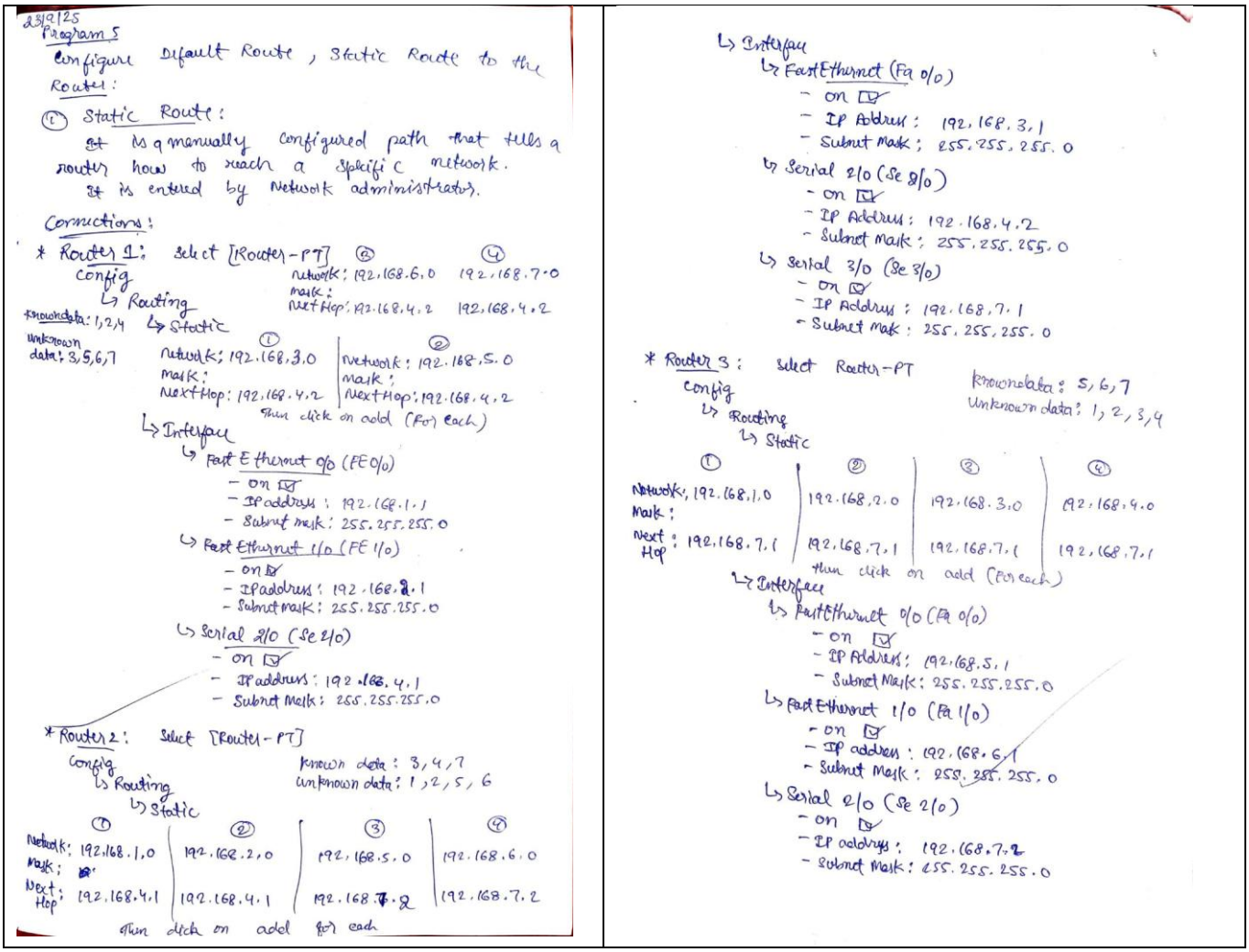
Program 5:

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

Network diagram:



Configuration:



* PC 0 (select PC-PT)
  ↳ Desktop
    ↳ IP configuration
      ↳ IP address : 192.168.1.2
        Subnet mask : 255.255.255.0
        Default Gateway : 192.168.1.1

* PC 1 (select PC-PT)
  ↳ Desktop
    ↳ IP configuration
      ↳ IP address : 192.168.1.3
        Subnet mask : 255.255.255.0
        Default Gateway : 192.168.1.1

* PC2 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.2.2
        Subnet mask : 255.255.255.0
        Default Gateway : 192.168.2.1

* PC3 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.2.3
        Subnet Mask : 255.255.255.0
        Default Gateway : 192.168.2.1

* PC4 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.3.2
        Subnet Mask : 255.255.255.0
        Default Gateway : 192.168.3.1

* PC5 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.3.3
        Subnet mask : 255.255.255.0
        Default Gateway : 192.168.3.1

* PC6 (select PC-PT)
  ↳ ~~Conf~~ Desktop
    ↳ IP configuration
      ↳ IP address : 192.168.5.2
        Subnet mask : 255.255.255.0
        Default Gateway : 192.168.5.1

* PC7 (select PC-PT)
  ↳ Desktop
    ↳ IP configuration
      ↳ IP address : 192.168.5.3
        Subnet Mask : 255.255.255.0
        Default Gateway : 192.168.5.1

* PC8 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.6.2
        Subnet Mask : 255.255.255.0
        Default Gateway : 192.168.6.1

* PC9 (select PC-PT)
  ↳ Desktop
    ↳ IP Configuration
      ↳ IP address : 192.168.6.3
        Subnet Mask : 255.255.255.0
        Default Gateway : 192.168.6.1

Output:

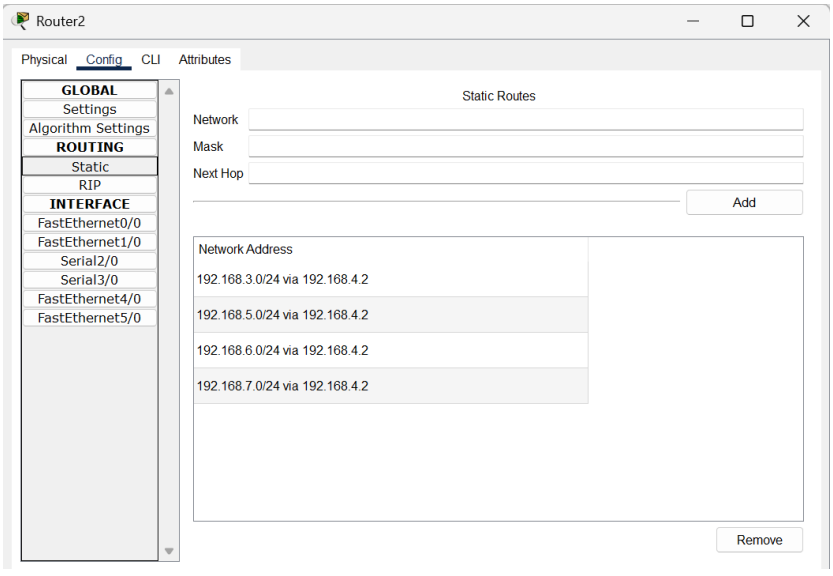


Fig 1. Router 2 – Static routing

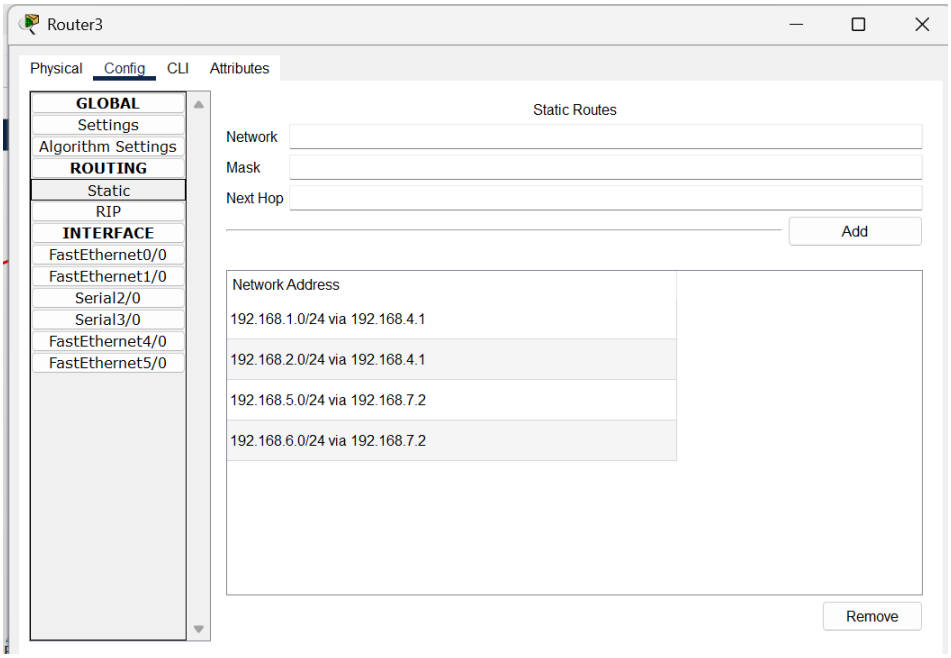


Fig 2. Router 3 – Static routing

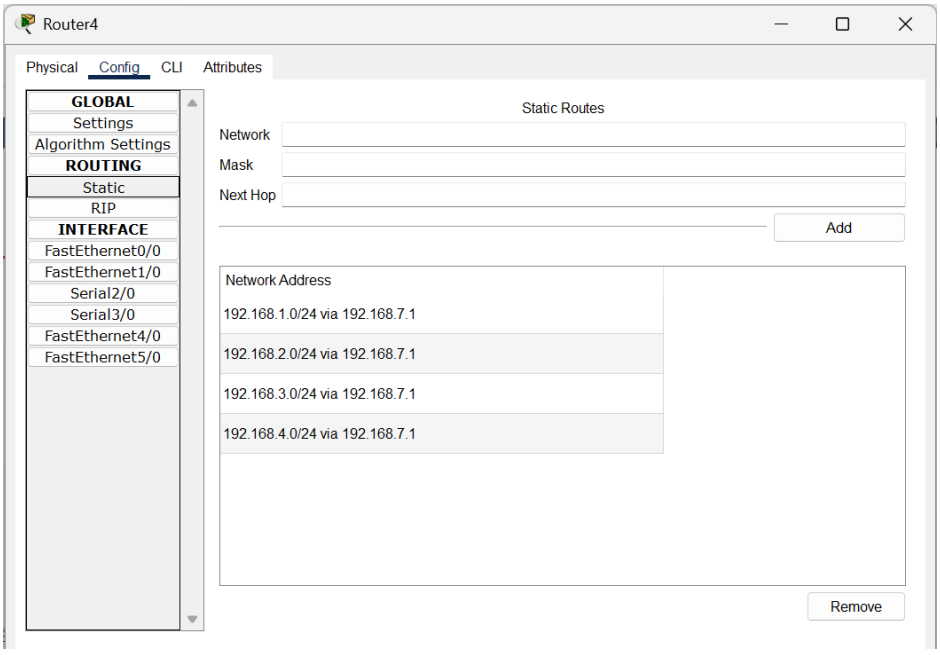
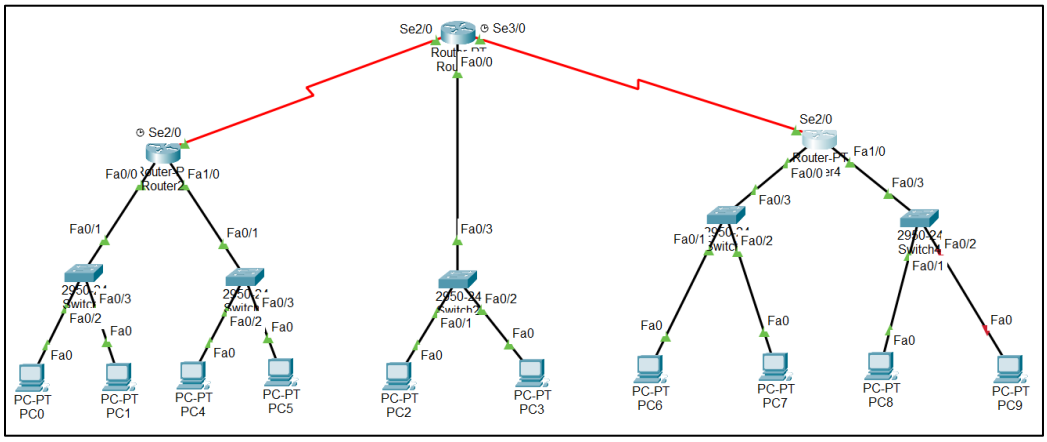


Fig 3. Router 4 – Static routing

Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:

(2) Dynamic Route :-

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

Connections :

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

* Router 1 : (select Router-PT)
  - ↳ Config
    - ↳ Routing
      - ↳ RIP Routing
        - ↳ Network : 192.168.1.0
          192.168.2.0
          192.168.4.0
          then click on add [For each]

* Router 2 :
  - ↳ Config
    - ↳ Routing
      - ↳ RIP Routing
        - ↳ Network : 192.168.3.0
          192.168.4.0
          192.168.7.0
          then click on add [For each]

* Router 3 :
  - ↳ Config
    - ↳ Routing
      - ↳ RIP Routing
        - ↳ Network : 192.168.5.0
          192.168.6.0
          192.168.7.0
          then click on add [For each]

Output:

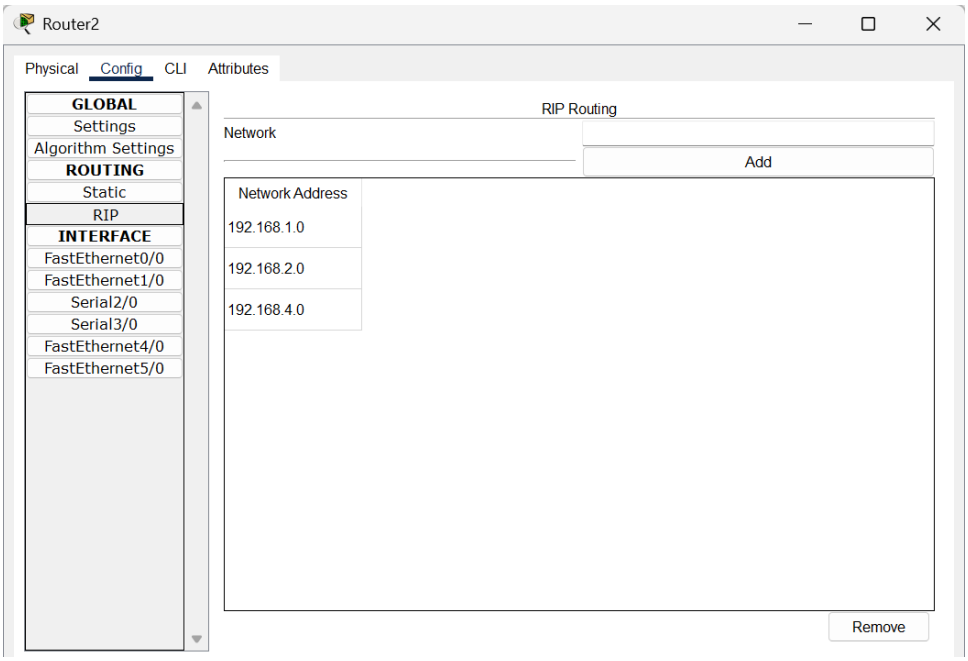


Fig 1. Router 2 – RIP routing

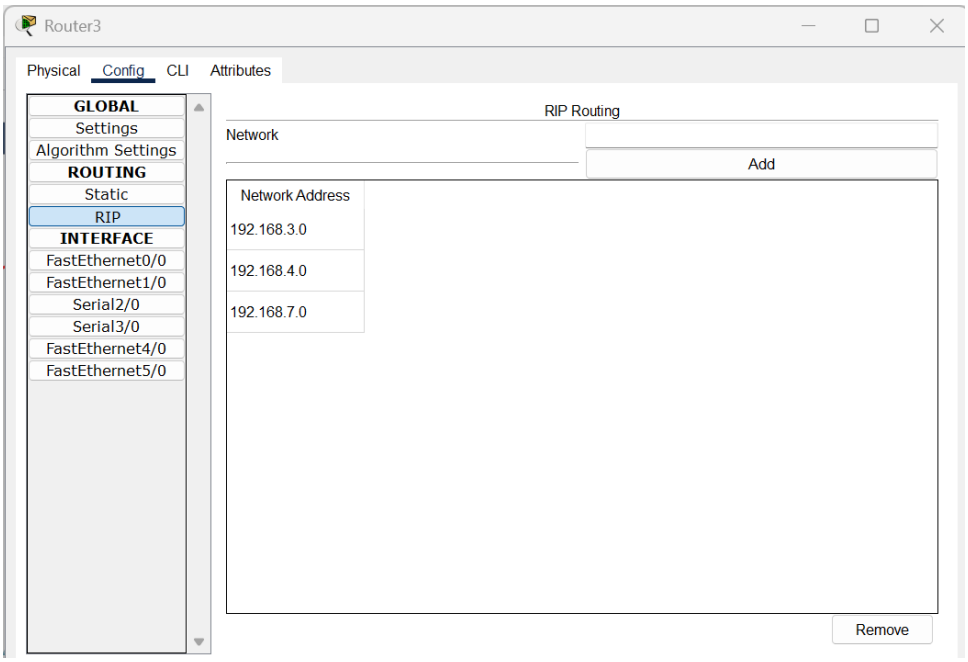


Fig 2. Router 3 – RIP routing

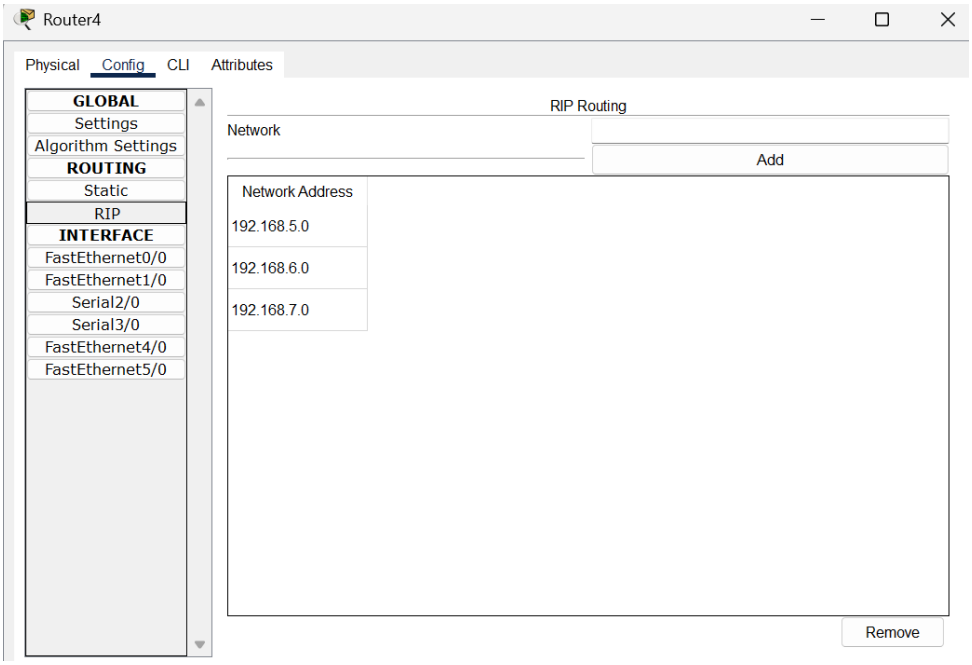
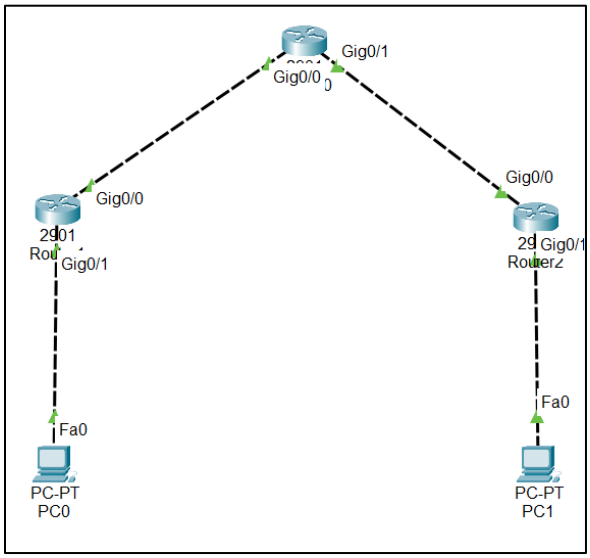


Fig 3. Router 4 – RIP routing

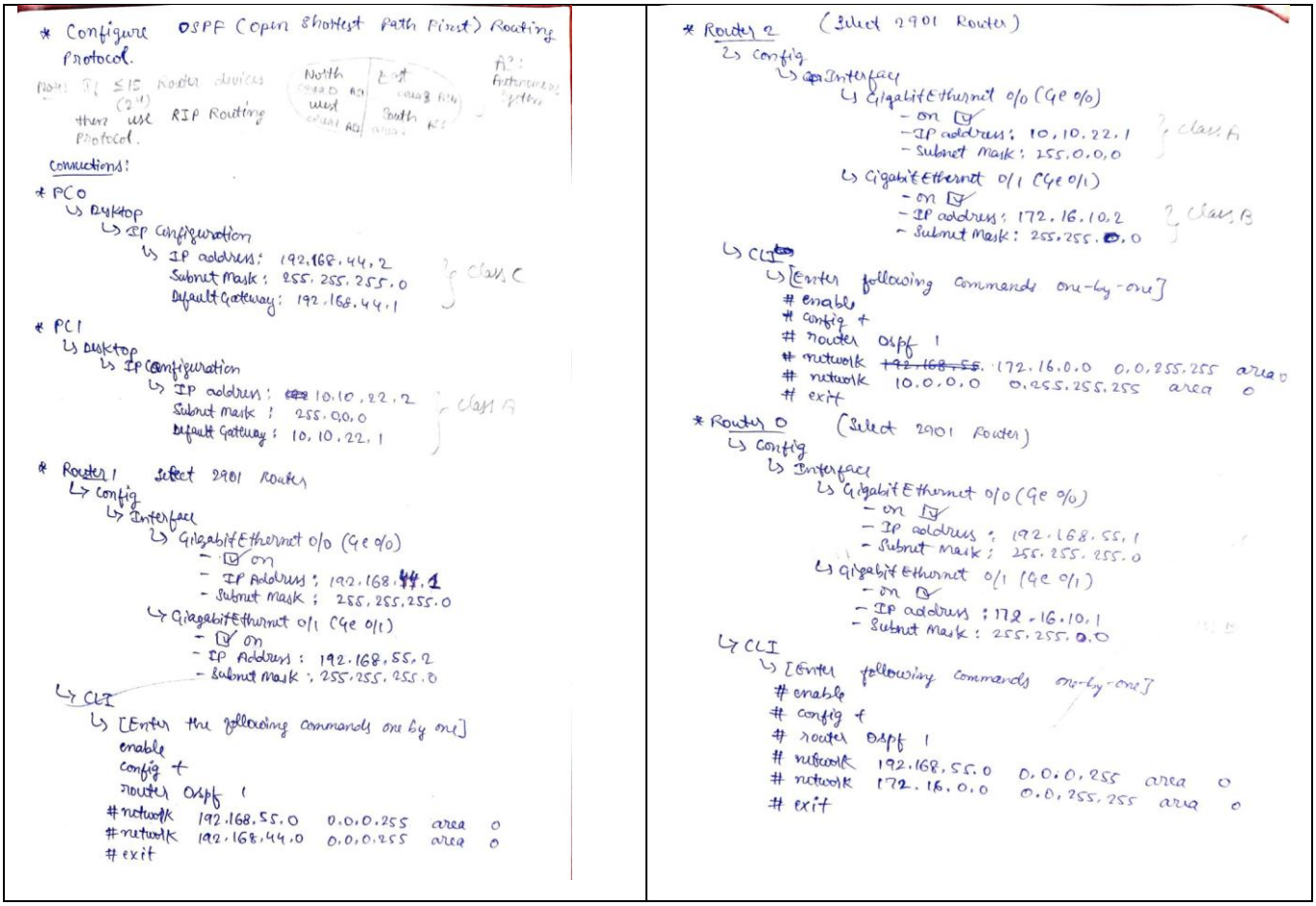
Program 7:

Aim: Configure OSPF routing protocol.

Network diagram:



Configuration:



* Configure OSPF (Open Shortest Path First) Routing Protocol.

Note: If ≤ 15 Router devices (2nd) then use RIP Routing Protocol.

Connections:

* PC0
  - Desktop
    - IP Configuration
      - IP address: 192.168.44.2
      - Subnet Mask: 255.255.255.0
      - Default Gateway: 192.168.44.1

} Class C

* PC1
  - Desktop
    - IP Configuration
      - IP address: 10.10.22.2
      - Subnet Mask: 255.0.0.0
      - Default Gateway: 10.10.22.1

} Class A

* Router 1 (Select 2901 Router)
  - Config
    - Interface
      - GigabitEthernet 0/0 (Ge 0/0)
        - on
        - IP Address: 192.168.44.1
        - Subnet Mask: 255.255.255.0
      - GigabitEthernet 0/1 (Ge 0/1)
        - on
        - IP Address: 192.168.55.2
        - Subnet Mask: 255.255.255.0
  - CLI
    - [Enter the following commands one by one]

```
enable
config t
router ospf 1
#network 192.168.55.0 0.0.0.255 area 0
#network 192.168.44.0 0.0.0.255 area 0
#exit
```

* Router 2 (Select 2901 Router)
  - Config
    - Interface
      - GigabitEthernet 0/0 (Ge 0/0)
        - on
        - IP address: 10.10.22.1
        - Subnet Mask: 255.0.0.0

} Class A

      - GigabitEthernet 0/1 (Ge 0/1)
        - on
        - IP address: 172.16.10.2
        - Subnet Mask: 255.255.0.0

} Class B

  - CLI
    - [Enter following commands one-by-one]

```
# enable
# config t
# router ospf 1
# network ~~192.168.55.~~ 172.16.0.0 0.0.255.255 area 0
# network 10.0.0.0 0.255.255.255 area 0
# exit
```

* Router 0 (Select 2901 Router)
  - Config
    - Interface
      - GigabitEthernet 0/0 (Ge 0/0)
        - on
        - IP address: 192.168.55.1
        - Subnet Mask: 255.255.255.0
      - GigabitEthernet 0/1 (Ge 0/1)
        - on
        - IP address: 172.16.10.1
        - Subnet Mask: 255.255.0.0
  - CLI
    - [Enter following commands one-by-one]

```
# enable
# config t
# router ospf 1
# network 192.168.55.0 0.0.0.255 area 0
# network 172.16.0.0 0.0.255.255 area 0
# exit
```

Output:

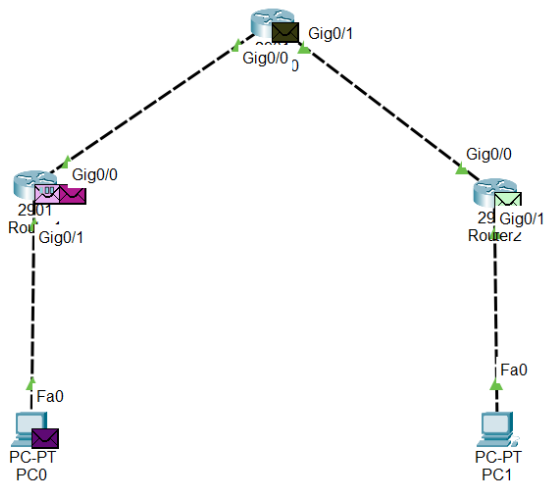


Fig 1. Sending PDU message from PC0 to PC1

PDU List Window

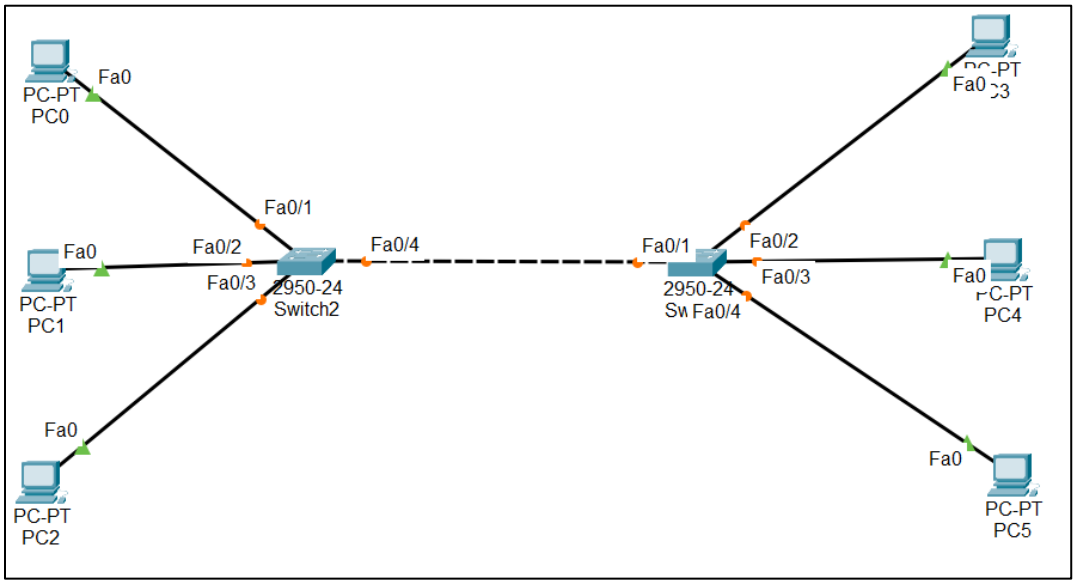
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Router2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	Router0	ICMP		0.000	N	2	(edit)	(delete)
	Successful	Router0	PC1	ICMP		0.000	N	3	(edit)	(delete)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

Fig 2. Checking PDU messages

Program 8:

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

Network diagram:



Configuration:

14/10/25

Aim: To construct a VLAN (Virtual LAN) & make the PCs communicate among a VLAN.

Connections:

* PC0: (select PC-PT)
  - Desktop
    - IP Configuration
      - IP address : 192.168.1.2
        Subnet mask : 255.255.255.0

* PC1:
  - Desktop
    - IP configuration
      - IP address : 192.168.1.3
        Subnet mask : 255.255.255.0

* PC2:
  - Desktop
    - IP configuration
      - IP address : 192.168.1.4
        Subnet mask : 255.255.255.0

* PC3:
  - Desktop
    - IP configuration
      - IP address : 192.168.1.5
        Subnet mask : 255.255.255.0

* PC4:
  - Desktop
    - IP configuration
      - IP address : 192.168.1.6
        Subnet mask : 255.255.255.0

* PC5:
  - Desktop
    - IP configuration
      - IP address : 192.168.1.7
        Subnet mask : 255.255.255.255

VLAN10		Vlan 20		Vlan 30	
fa0/1	PC0	fa0/2	PC1	fa0/3	PC2
fa0/2	PC3	fa0/3	PC4	fa 0/4	PC5

* Switch0:
  - CLI (Command Line Interface)
    - [type this commands one by one]

```
enable
config t
int fa 0/1
switchport access vlan 10
int fa 0/2
switchport access vlan 20
int fa 0/3
switchport access vlan 30
int fa 0/4
switchport mode trunk
exit
```

* Switch1;
  - CLI

```
enable
config t
int fa 0/2
switchport access vlan 10
int fa 0/3
switchport access vlan 20
int fa 0/4
switchport access vlan 30
int fa 0/1
switchport mode trunk
exit
```

PC-PT

Output:

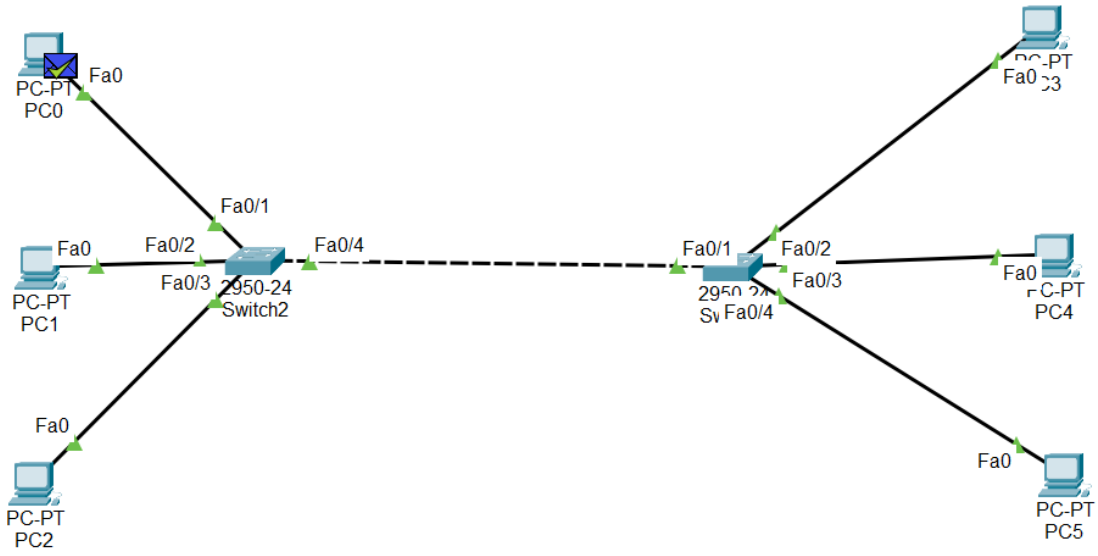


Fig 1. Sending PDU message from PC0 to PC5

PDU List Window

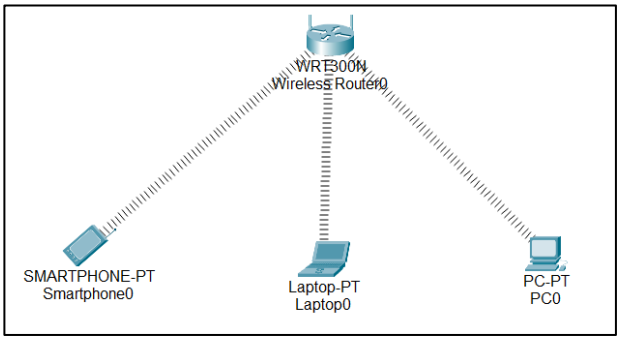
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC5	ICMP		0.000	N	2	(edit)	(delete)
	Successful	PC1	PC3	ICMP		0.000	N	3	(edit)	(delete)
	Successful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

Fig 2. Checking PDU messages

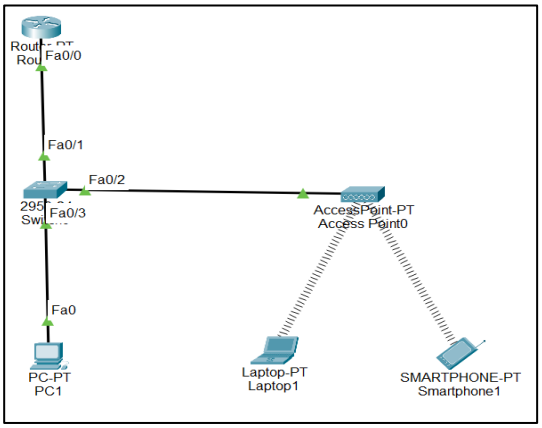
Program 9:

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

Network diagram:

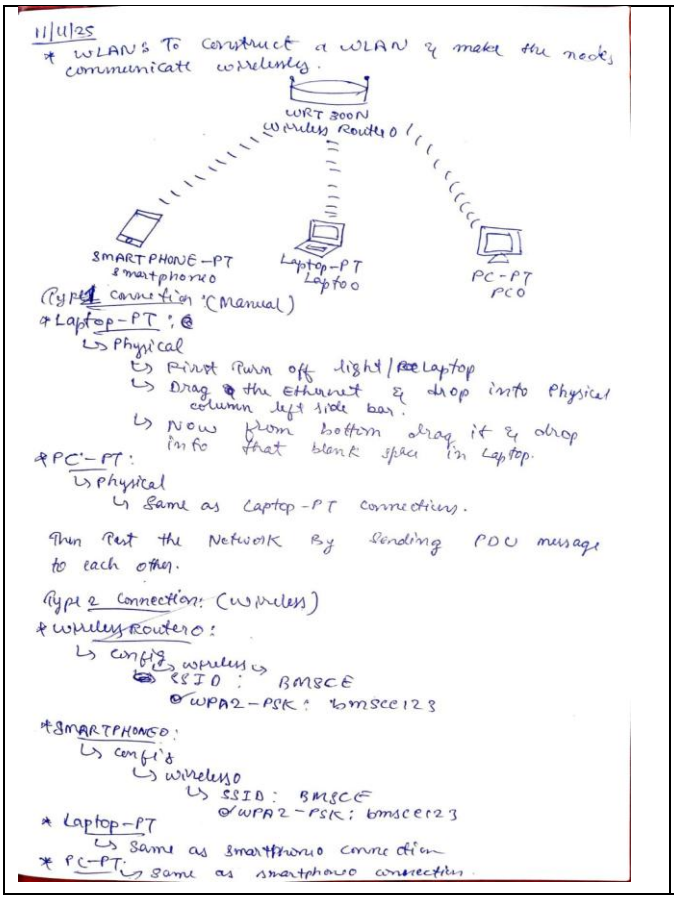


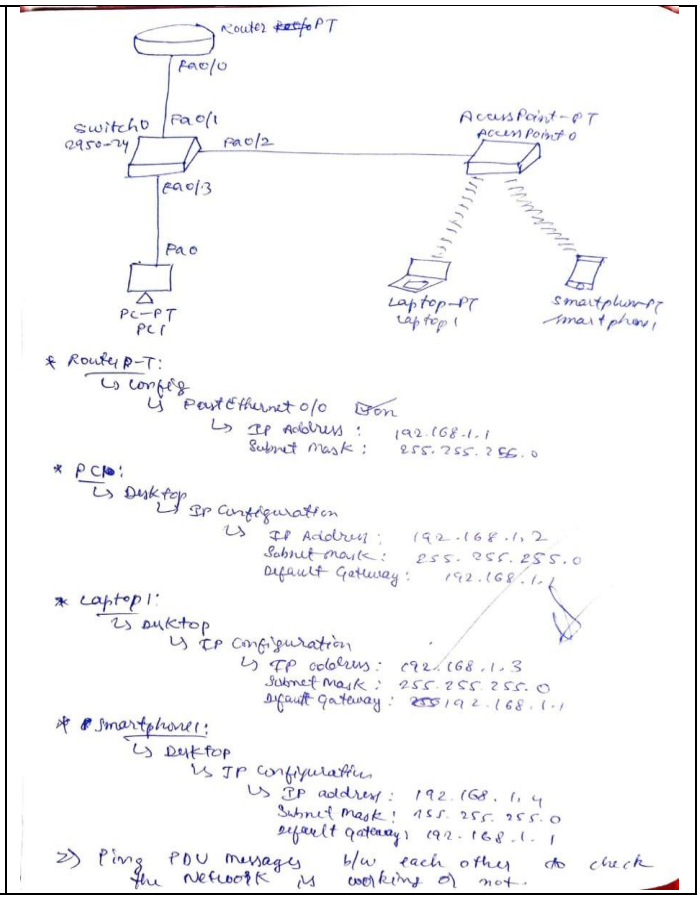
Configuration 1



Configuration 2

Configuration:





Output:

Do Physical Connections In:

- Laptop
- PC

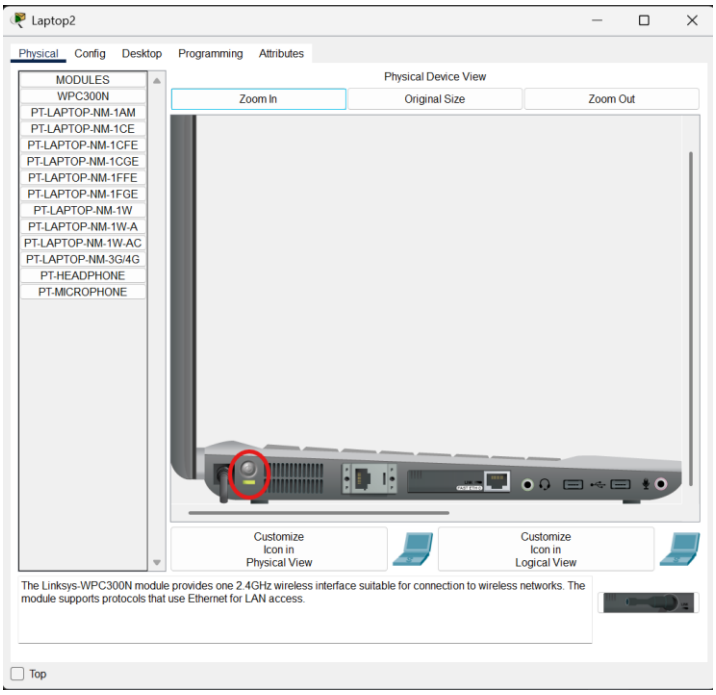


Fig 1.1 Step1: Turn off light / Power off laptop

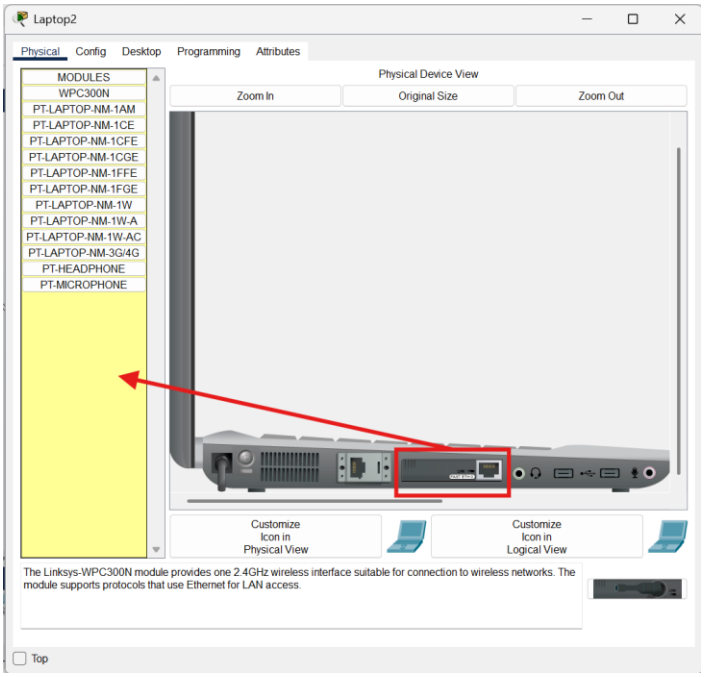


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

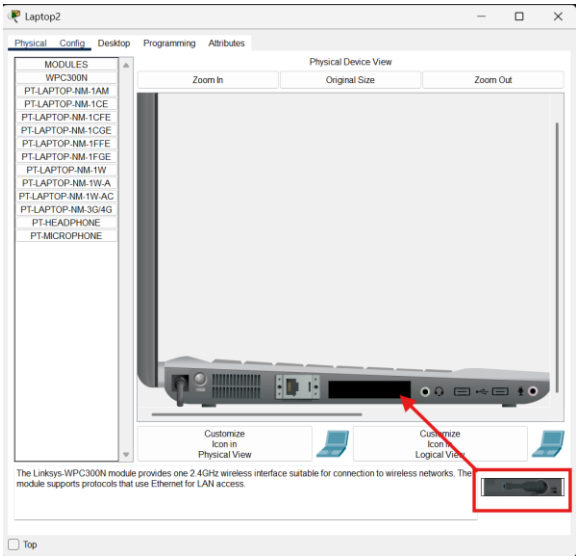


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

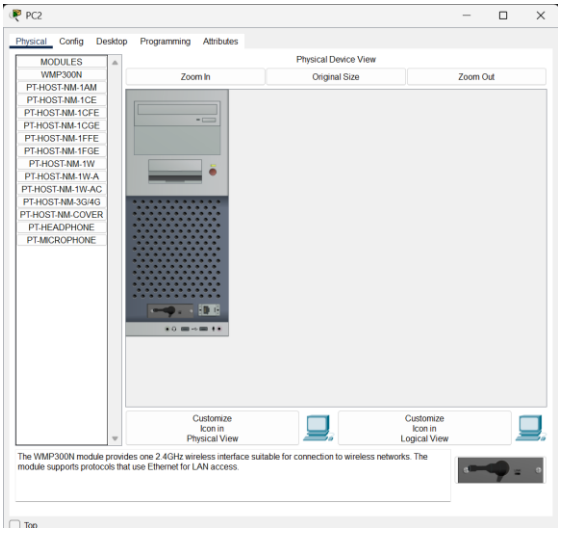


Fig 2. PC physical connection (combined 3 steps)

PDU List Window

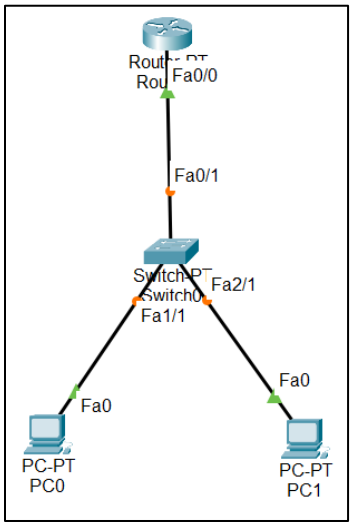
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit
	Failed	Smar...	Laptop0	ICMP		0.000	N	0	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	1	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	2	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	3	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	4	(edit)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

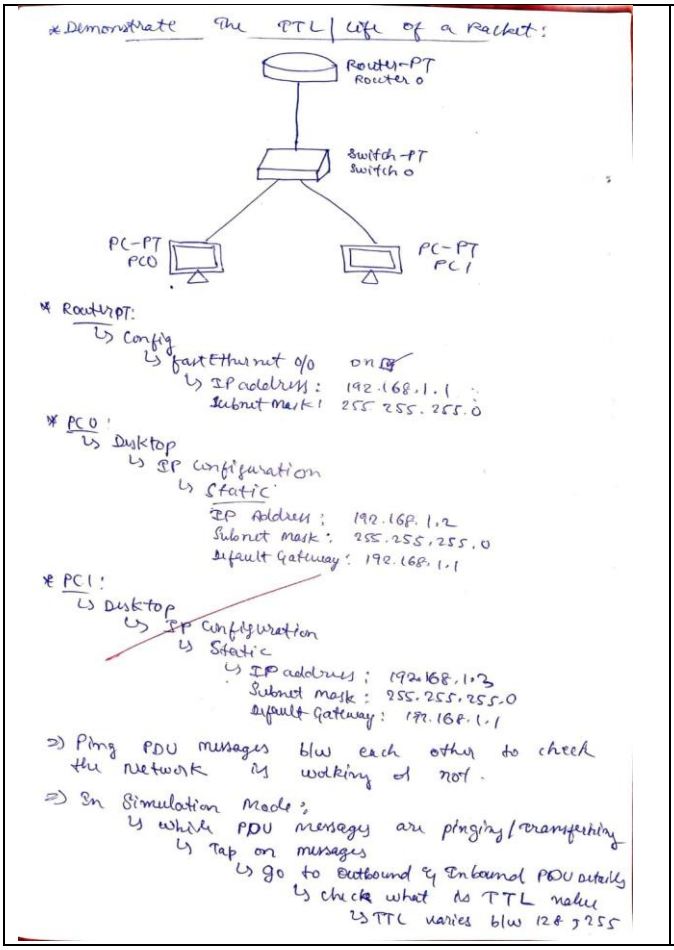
Program 10:

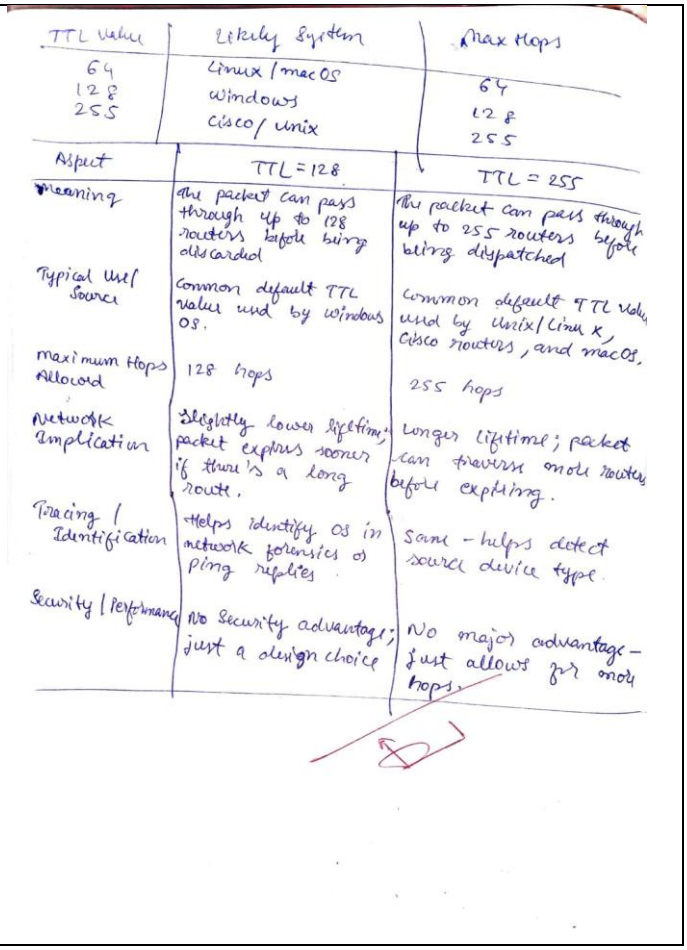
Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:





Output:

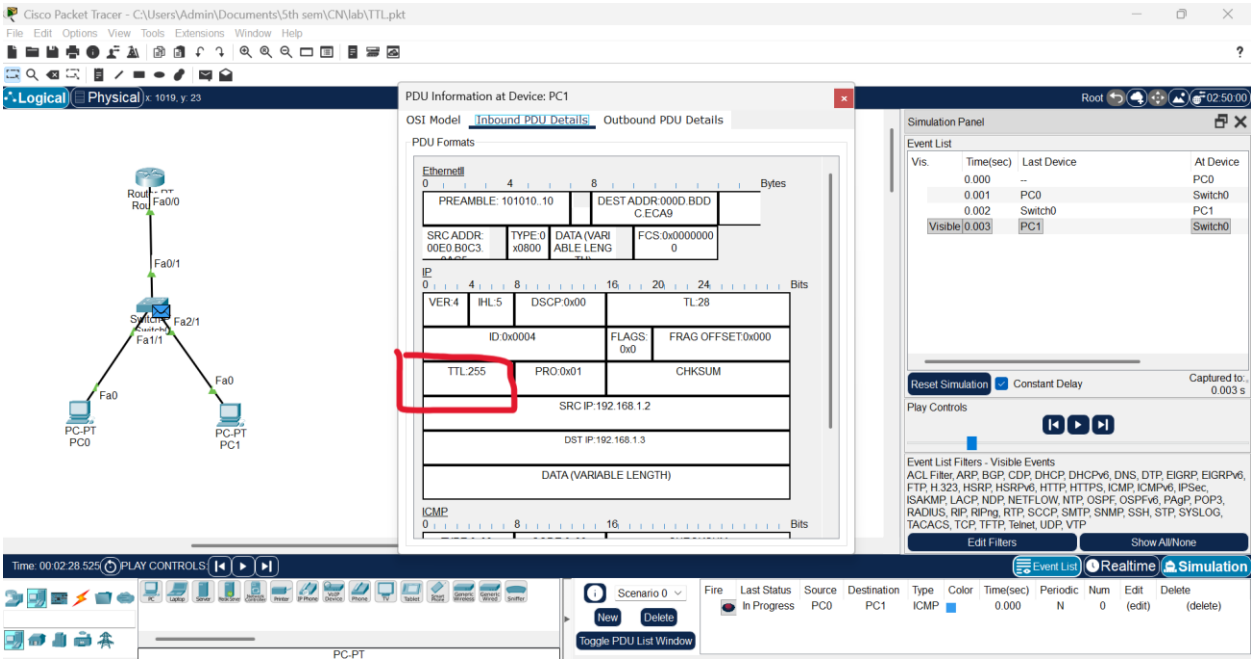


Fig 1. Inbound PDU Details at Device PC1

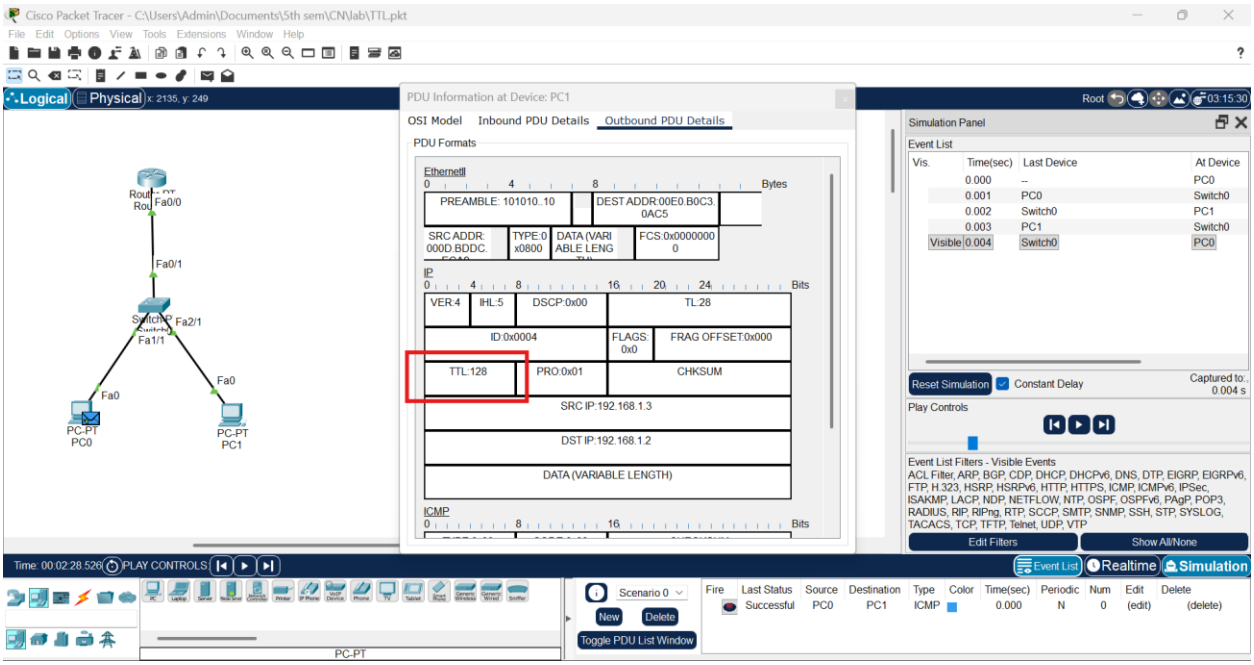
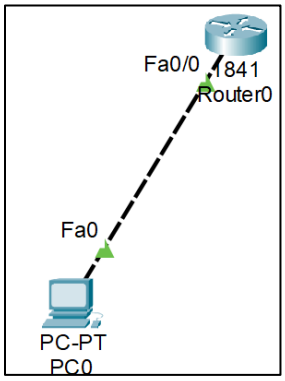


Fig 1. Outbound PDU Details at Device PC1

Program 11:

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

Network diagram:



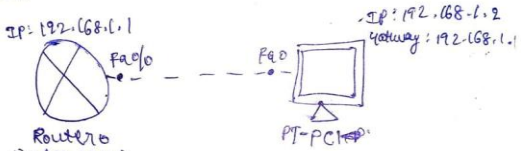
Configuration:

18/4/25

* Construct a topology to demonstrate concept of Telnet.

TELNET:

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



Configuration

* Router0:

↳ Go to CLI
↳ [yes/No]: No
↳ enter
↳ enter
↳ enable
↳ config t
↳ hostname R1
↳ enable secret cp
↳ int Fa 0/0
↳ ip add 192.168.1.1 255.255.255.0
↳ no shutdown
↳ enter
↳ enter
↳ line vty 0 5
↳ login
↳ password tp
↳ exit
↳ exit
↳ wr
↳ show ip interface brief.

*PC:

PC > ping 192.168.1.1
PC > telnet 192.168.1.1
Trying 192.168.1.1 open
User Access verification
Password: tp
R1 > enable
Password: np
R1# show ip interface brief
R1# en
R1# config t
R1(config)# int Fa 0/1
R1(config)# ip add 192.168.1.2 255.255.255.0
% 192.168.1.0 overlaps with FastEthernet0/0

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

Output:

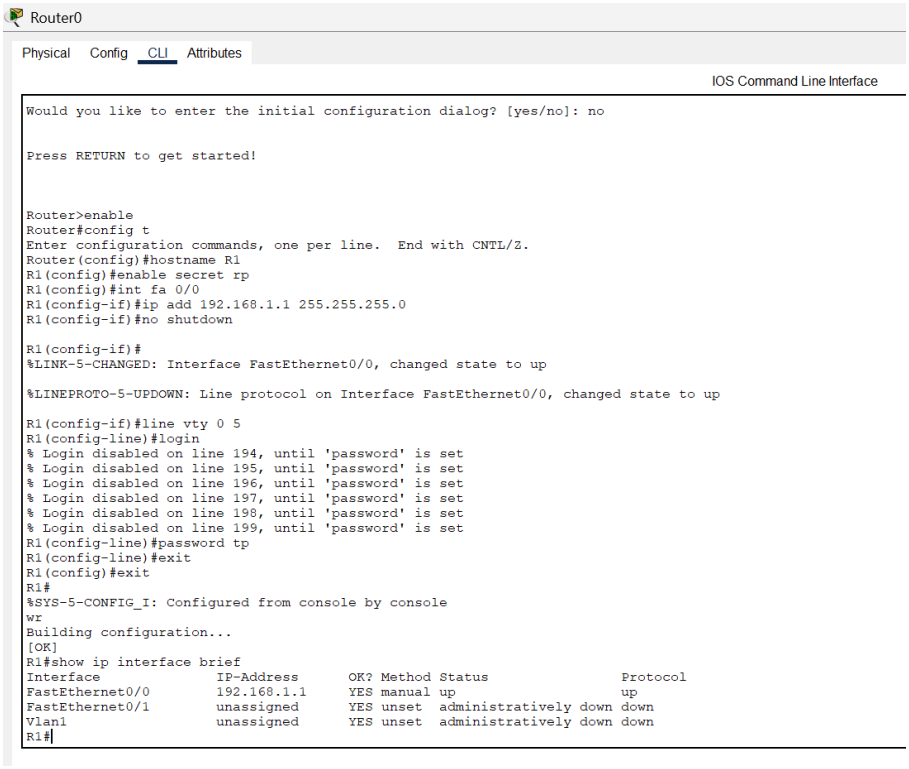


Fig 1. Router0 – CLI commands

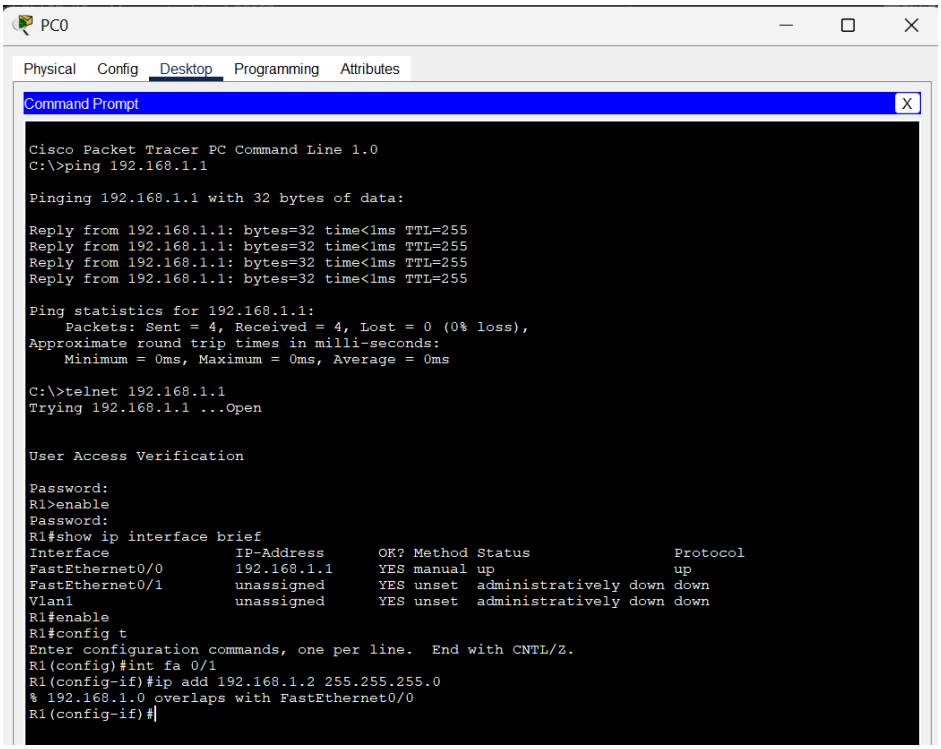


Fig2. PC command line prompt

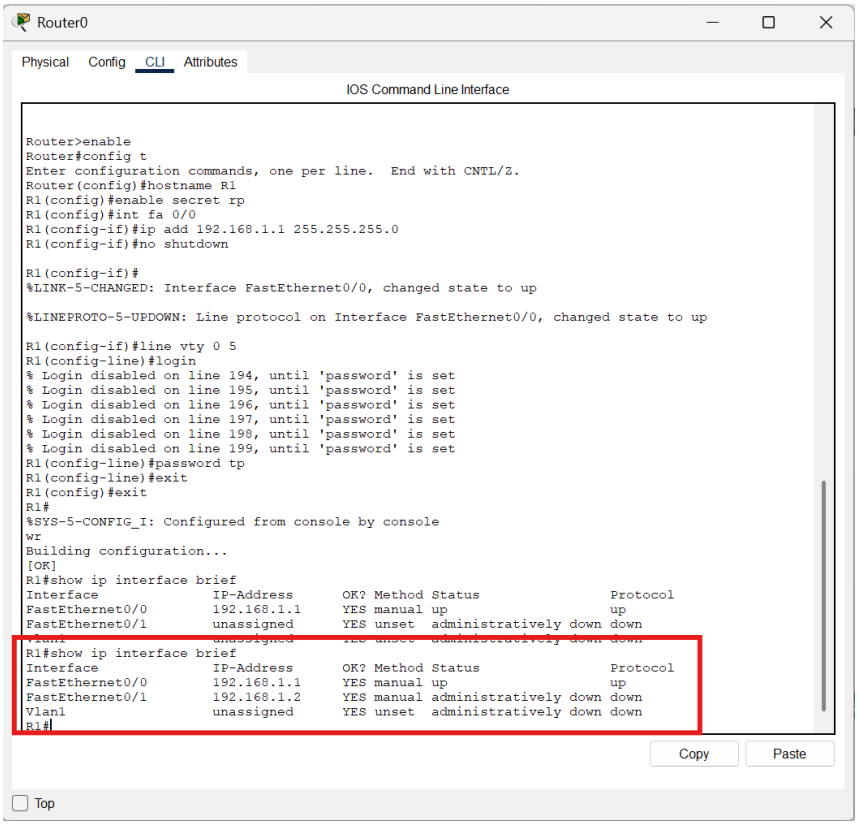


Fig 3. Updated the changes into Router0

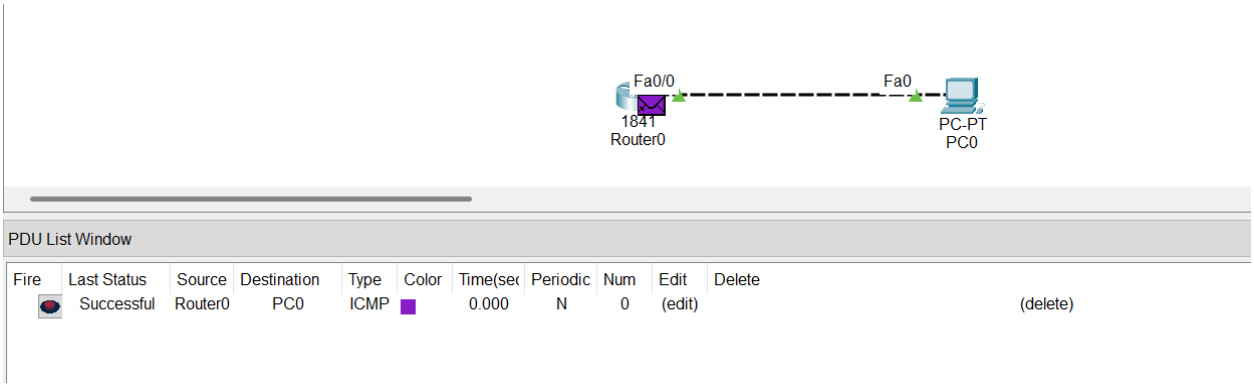
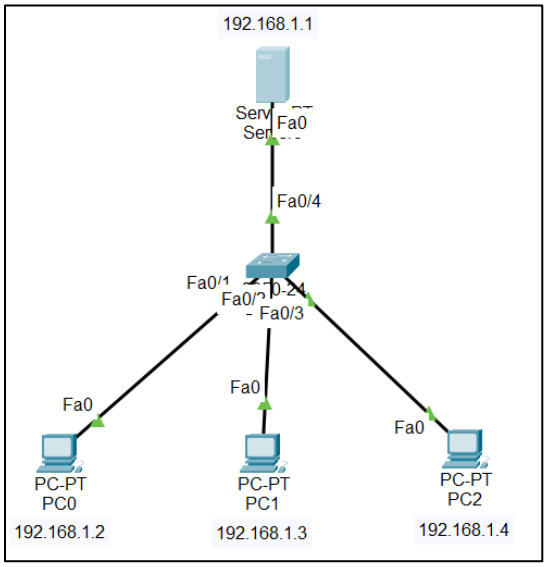


Fig 4. PDU message Successful

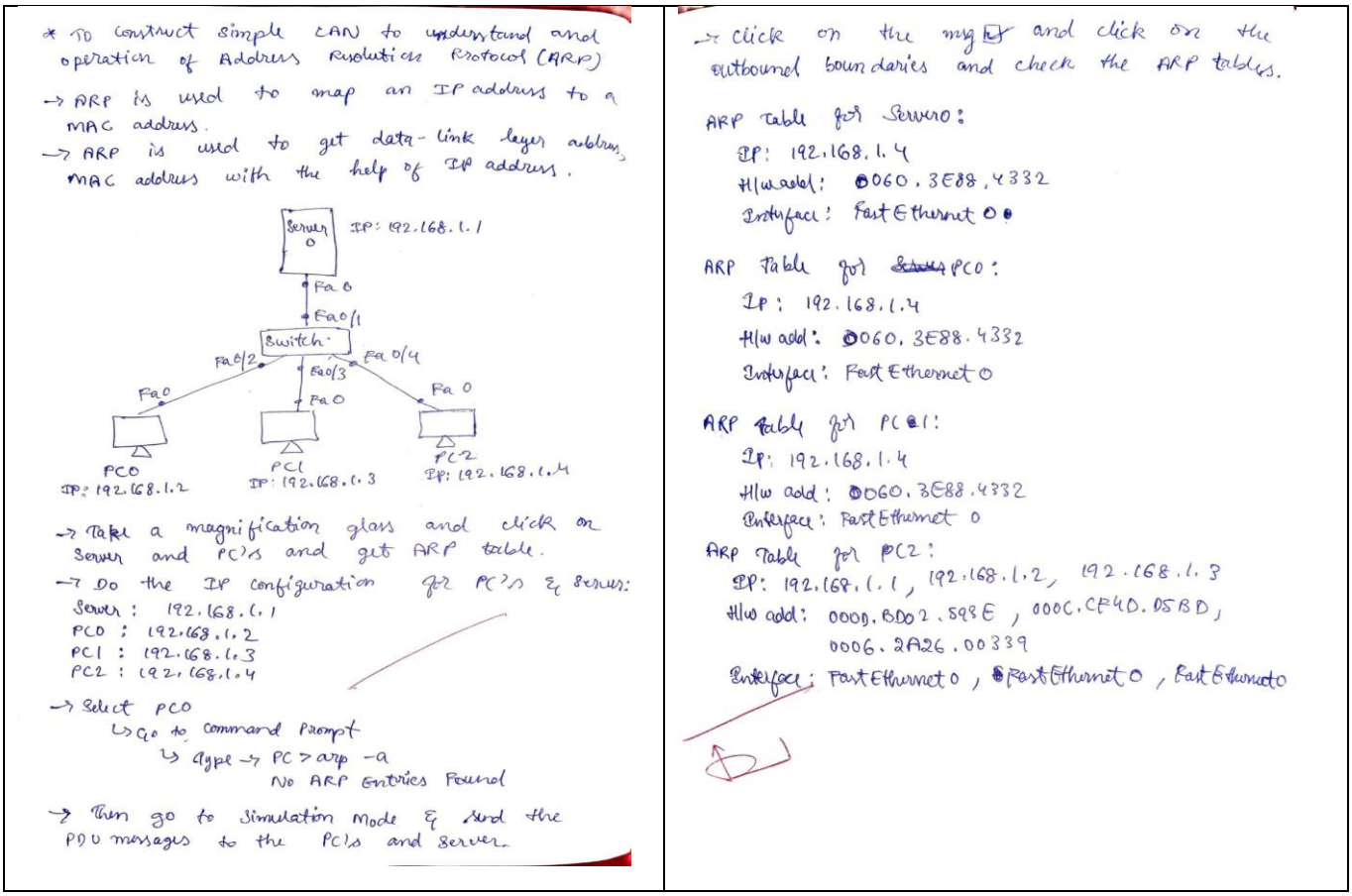
Program 12:

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

Network diagram:



Configuration:



* To construct simple LAN to understand and operation of Address Resolution Protocol (ARP)

→ ARP is used to map an IP address to a MAC address.

→ ARP is used to get data-link layer address, MAC address with the help of IP address.

Server 0 IP: 192.168.1.1
Fa 0
Fa0/1
Switch
Fa0/2
Fa0/3
Fa0/4
Fa 0
Fa 0
Fa 0
PC0 IP: 192.168.1.2
PC1 IP: 192.168.1.3
PC2 IP: 192.168.1.4

→ Take a magnification glass and click on Server and PC's and get ARP table.

→ Do the IP configuration for PC's & Server:
Server : 192.168.1.1
PC0 : 192.168.1.2
PC1 : 192.168.1.3
PC2 : 192.168.1.4

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

→ Then go to Simulation Mode & send the PDU masages to the PC's and Server.

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

ARP Table for Server0:
IP: 192.168.1.4
Hlw addl: 0060.3E88.4332
Interface: FastEthernet 0

ARP Table for ~~Server~~ PC0:
IP: 192.168.1.4
Hlw add: 0060.3E88.4332
Interface: FastEthernet 0

ARP Table for PC1:
IP: 192.168.1.4
Hlw add: 0060.3E88.4332
Interface: FastEthernet 0

ARP Table for PC2:
IP: 192.168.1.1, 192.168.1.2, 192.168.1.3
Hlw add: 0009.BD02.598E, 000C.CF4D.05BD, 0006.2A26.0033
Interface: FastEthernet 0, FastEthernet 0, FastEthernet 0

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

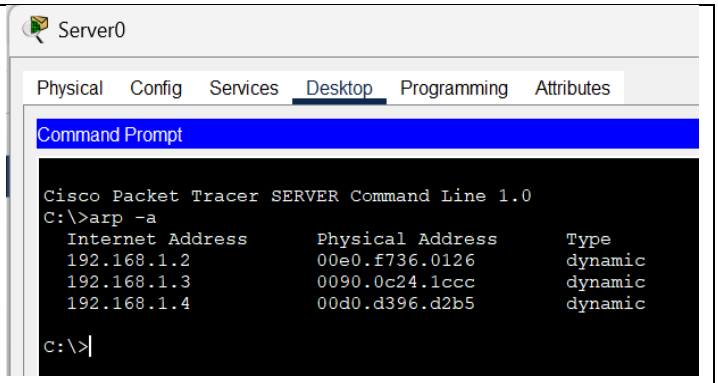


Fig 1.2 Command Prompt at Server0

ARP Table for PC0

IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 2.1 ARP table at PC0

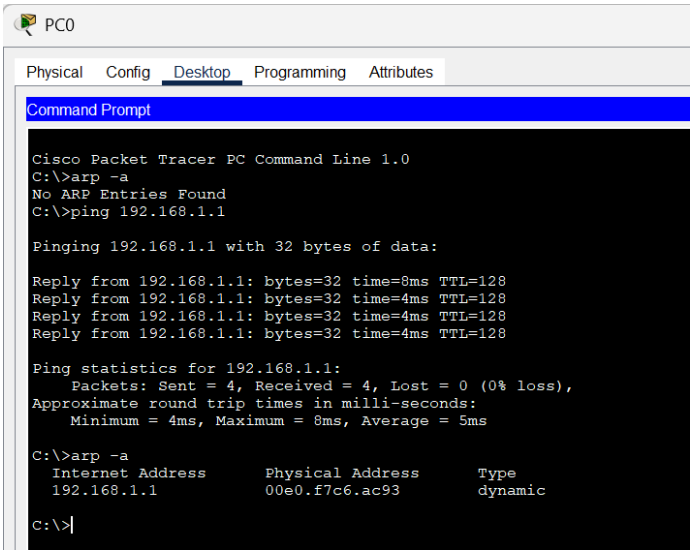


Fig 2.2 Command Prompt at PC0

ARP Table for PC1

IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 3.1 ARP table at PC1

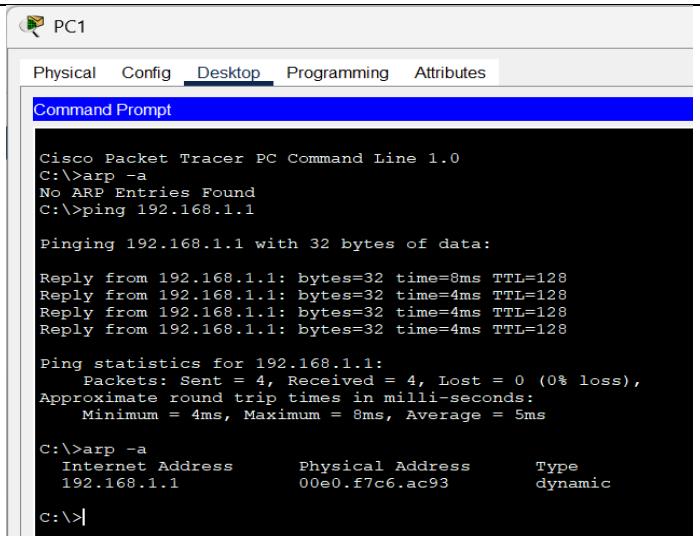


Fig 3.2 Command Prompt at PC1

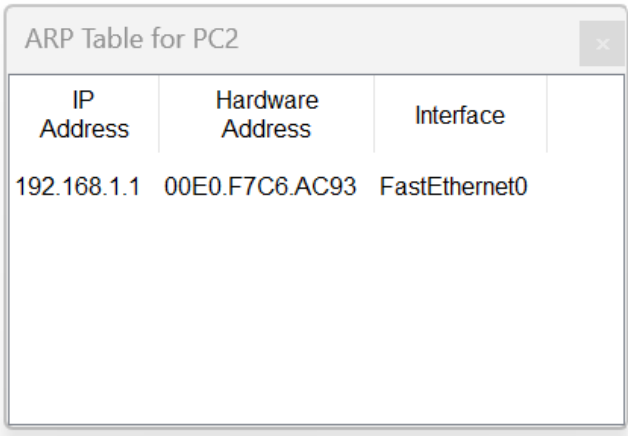


Fig 4.1 ARP table at PC2

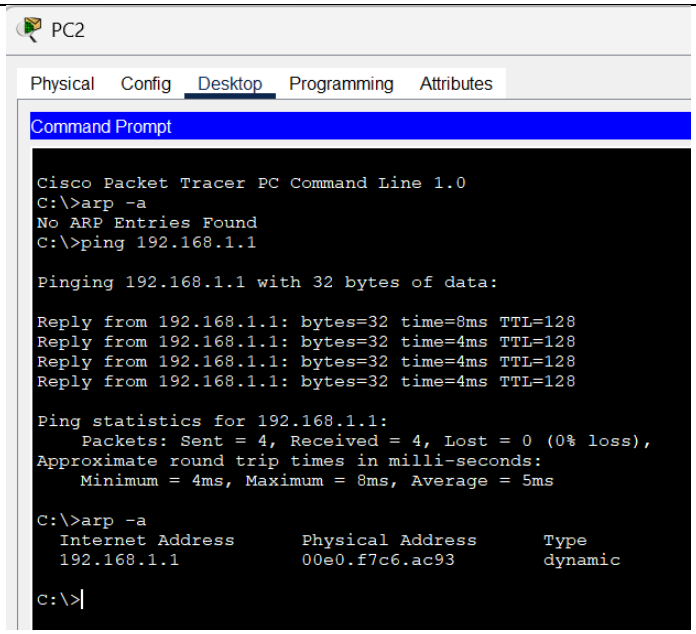


Fig 4.2 Command Prompt at PC2

PART - B

Program 1:

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

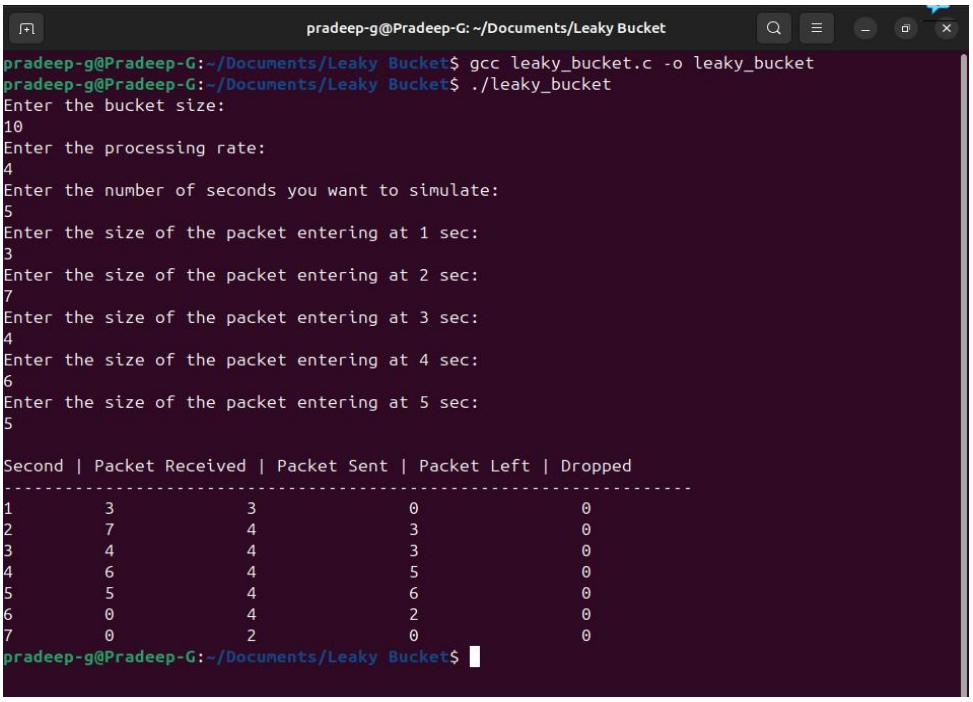
Code:

```
#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);
```

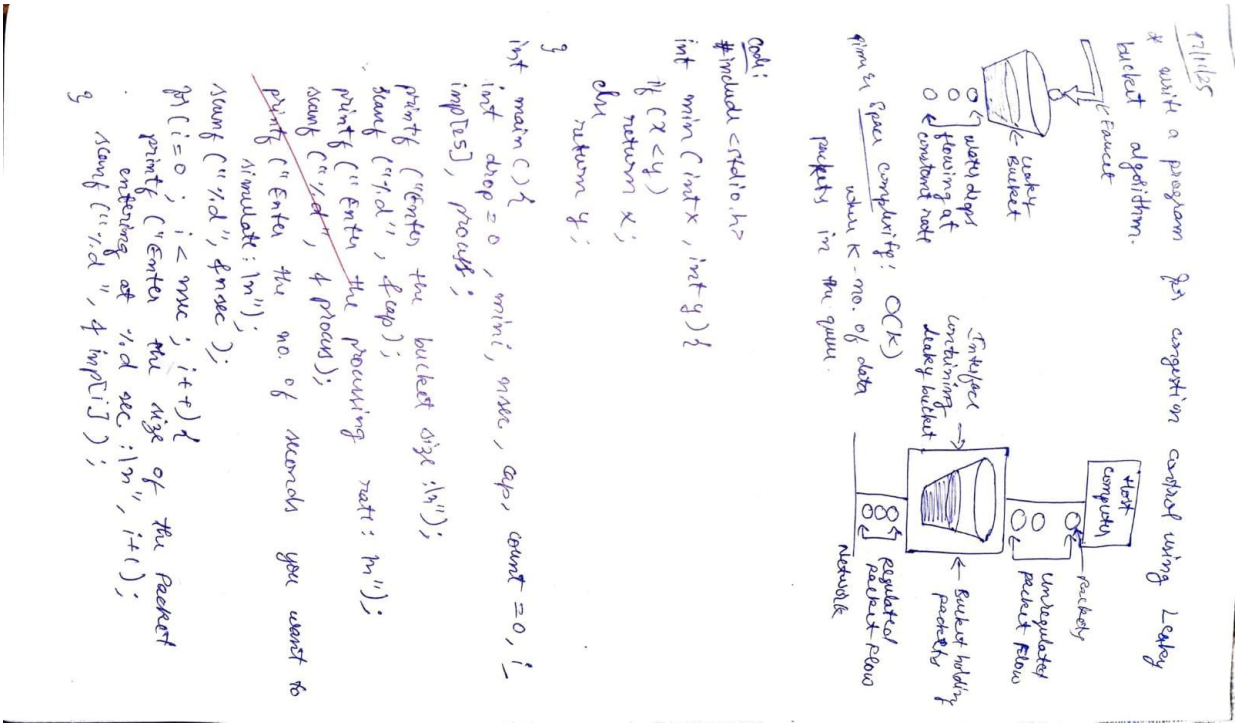
```
        scanf("%d", &inp[i]);
    }

    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:



Observation:




```
#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 no. 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:

```
Enter the bucket size:
10
Enter the processing rate:
4
Enter the no. of seconds you want to simulate:
5
Enter the size of the packet entering at 1sec: 3
Enter the size of the packet entering at 2sec: 7
Enter the size of the packet entering at 3sec: 4
Enter the size of the packet entering at 4sec: 6
Enter the size of the packet entering at 5sec: 5
```

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

Program 2:

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

Code:

```
# tcp_client.py


import socket


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


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


# Step 3: Send filename
filename = input("Enter filename to
request: ")


client_socket.send(filename.encode())


# Step 4: Receive file contents
data =
client_socket.recv(4096).decode()


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


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

```
# tcp_server.py

import socket

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

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

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

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

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

try:
    # Step 6: Open and read file
    with open(filename, 'r') as f:
        data = f.read()

    conn.send(data.encode())  # Send
file contents

except FileNotFoundError:
    conn.send(b"File not found on
server.")

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

Output:

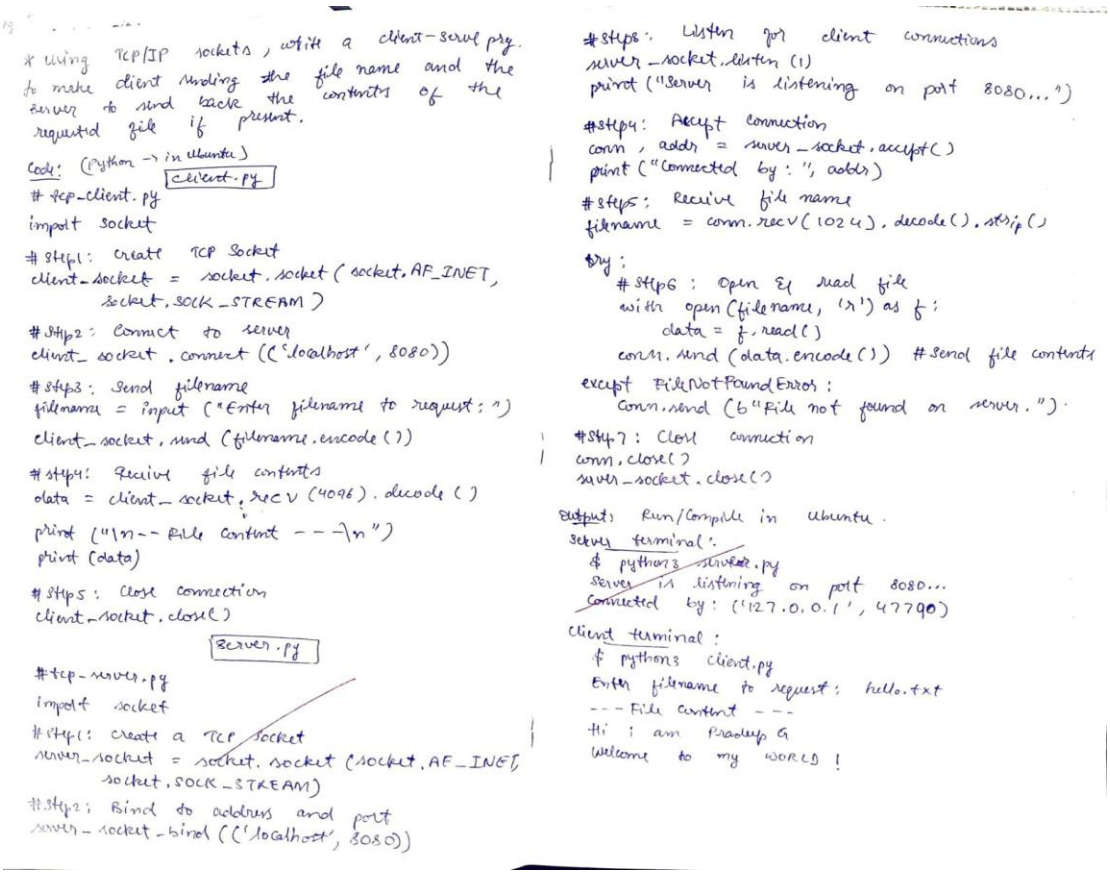
Server side Terminal:

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



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

Code: (Python → in ubuntu)

client.py

```
# tcp_client.py
import socket
# Step1: create TCP socket
client_socket = socket.socket(socket.AF_INET,
        socket.SOCK_STREAM)
# Step2: Connect to server
client_socket.connect(('localhost', 8080))
# Step3: Send filename
filename = input("Enter filename to request: ")
client_socket.send(filename.encode())
# Step4: Receive file contents
data = client_socket.recv(4096).decode()
print("\n--- File Content ---\n")
print(data)
# Step5: Close connection
client_socket.close()
```

server.py

```
#tcp_server.py
import socket
# Step1: create a TCP socket
server_socket = socket.socket(socket.AF_INET,
        socket.SOCK_STREAM)
# Step2: Bind to address and port
server_socket.bind(('localhost', 8080))
# Step3: Listen for client connections
server_socket.listen(1)
print("Server is listening on port 8080...")
# Step4: Accept connection
conn, addr = server_socket.accept()
print("Connected by:", addr)
# Step5: Receive file name
filename = conn.recv(1024).decode().strip()
try:
    # Step6: Open & read file
    with open(filename, 'r') as f:
        data = f.read()
    conn.send(data.encode()) # Send file contents
except FileNotFoundError:
    conn.send(b"File not found on server.")
# Step7: Close connection
conn.close()
server_socket.close()
```

Output: Run/Compile in Ubuntu.

Server terminal:

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

Client terminal:

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

Program 3:

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

Code:

```
# udp_client.py


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:

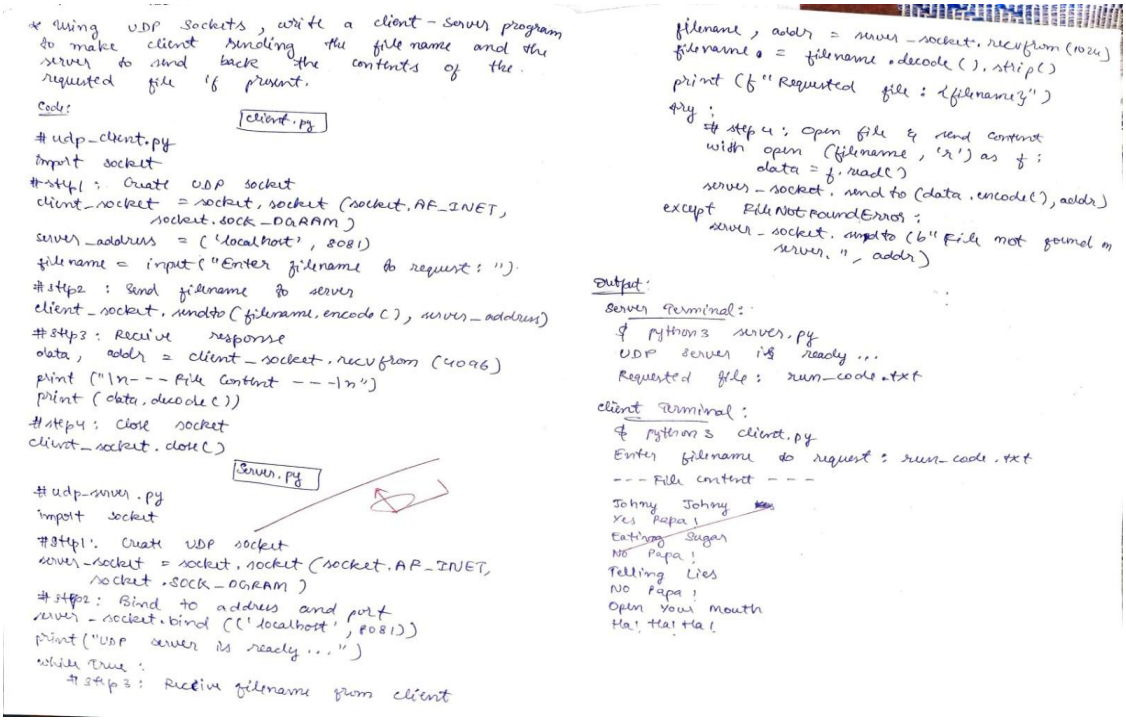
Server side Terminal:

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



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:

client.py

```
# 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()
```

server.py

```
# 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 & send content
        with open(filename, 'r') as f:
            data = f.read()
        server_socket.sendto(data.encode(), addr)
    except FileNotFoundError:
        server_socket.sendto("File not found on
                server.", addr)
```

Output:

Server Terminal:

```
$ python3 server.py
UDP server is ready...
Requested file: run_code.txt
```

Client Terminal:

```
$ python3 client.py
Enter filename to request: run_code.txt
--- File Content ---
Johny Johny
Yes Papa!
Eating Sugar
No Papa!
Telling Lies
No Papa!
Open your Mouth
Ha! Ha! Ha!
```

Program 4:

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

Code:

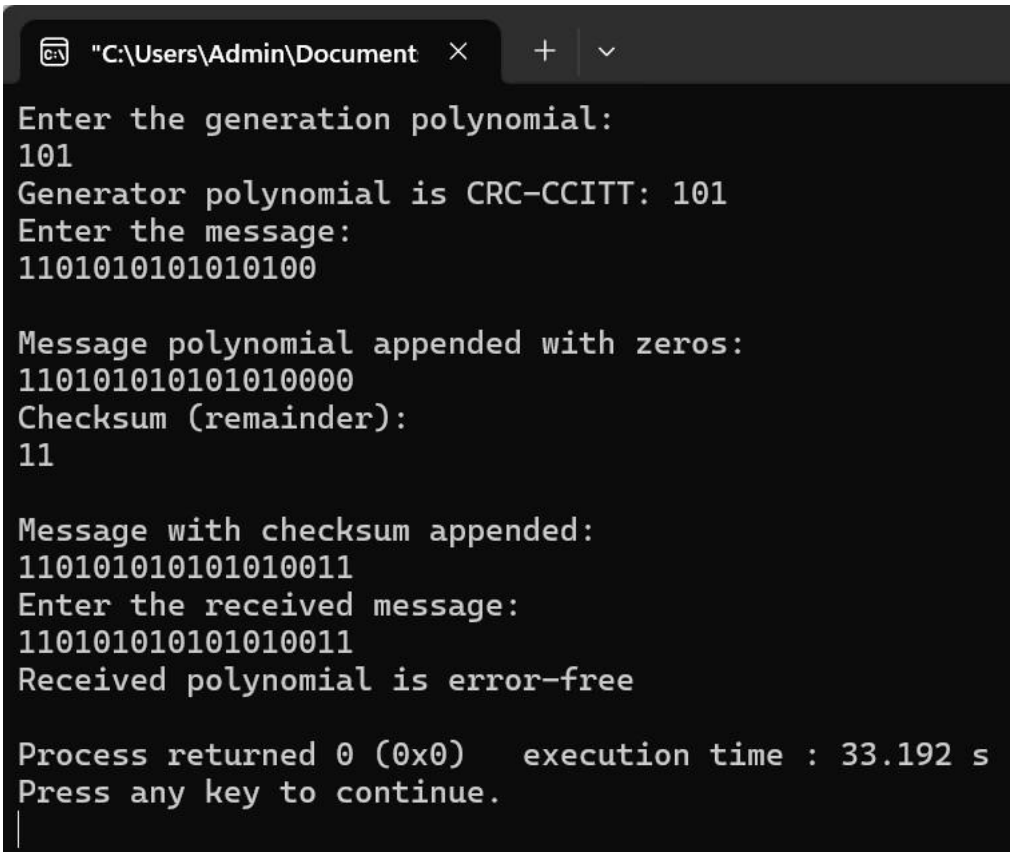
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

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:



Observation: