

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“JnanaSangama”, Belgaum -590014, Karnataka.



LAB RECORD

Computer Networks

Submitted by

Ayman Khan (1BM22CS062)

in partial fulfilment 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
Academic Year 2024-25 (odd)

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 **Ayman Khan (1BM22CS062)**, who is a 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.

Dr. Nandhini Vineeth Associate Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
---	--

Index

Cycle - I

Sl.No.	Date	Experiment Title	Page No.
1	01/10/2024	Create a topology involving multiple hubs and a switch connecting them to simulate with simple PDU.	1
2	08/10/2024	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	4
3	22/10/2024	Configure default route, static route to the router	7
4	29/10/2024	Configure DHCP within a LAN and outside LAN.	12
5	29/10/2024	Configure RIP routing Protocol in Routers.	15
6	12/11/2024	Configure OSPF Routing Protocol	19
7	12/11/2024	Demonstrate the TTL/ Life of a Packet	25
8	18/12/2024	Configure Web Server & DNS	27
9	18/12/2024	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	30
10	18/12/2024	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	34

11	18/12/2024	To construct a VLAN and make the PC's communicate among a VLAN	37
12	18/12/2024	To construct a WLAN and make the nodes communicate wirelessly	40

Cycle-II

Sl.No.	Date	Experiment Title	Page No.
1	03/12/2024	Write a program for error detecting code using CRC-CCITT (16-bits).	44
2	17/12/2024	Write a program for congestion control using Leaky bucket algorithm	47
3	24/12/2024	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.	49
4	24/12/2024	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.	52
5		Tool Exploration - Wireshark	55

Github Link:

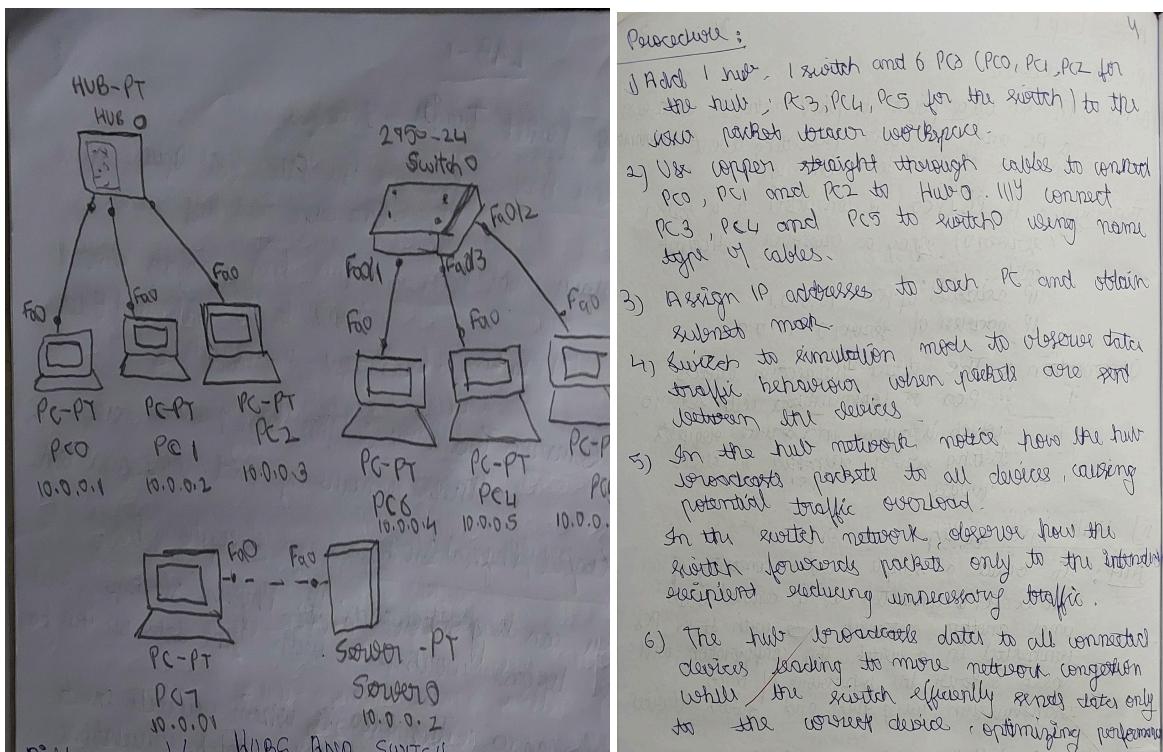
<https://github.com/AymanKhan97/CN-LAB>

Program 1

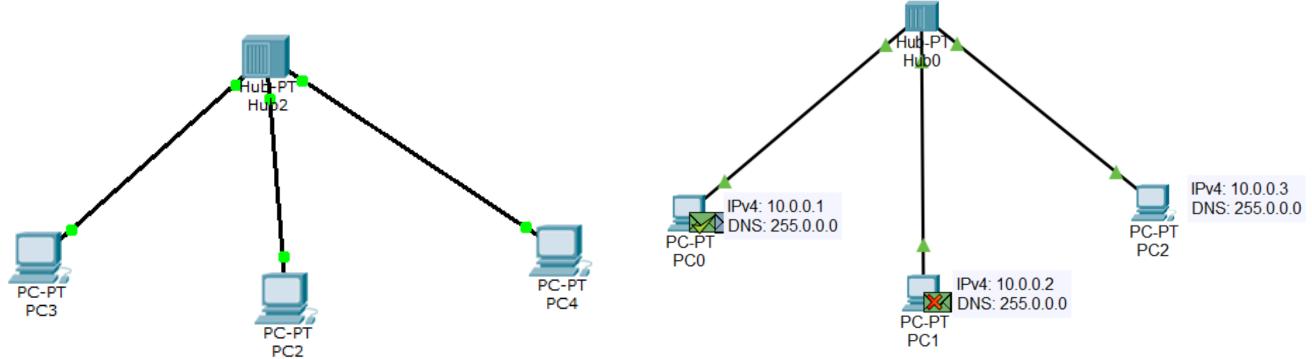
Aim of the program:

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

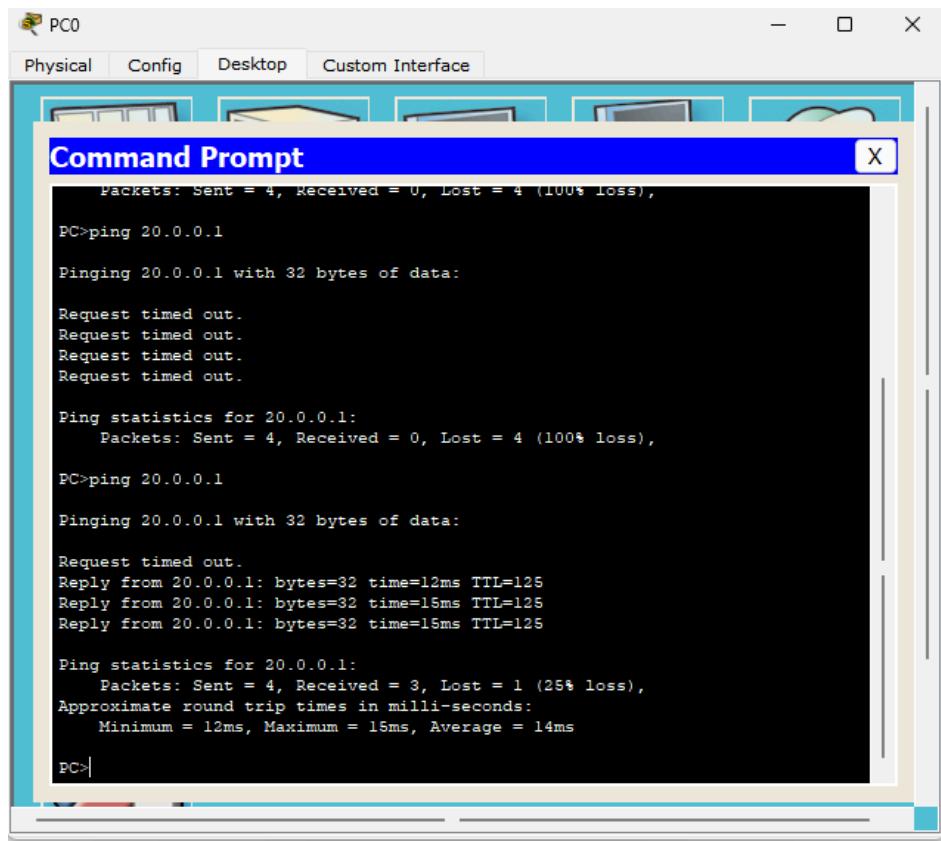
Procedure along with the topology:

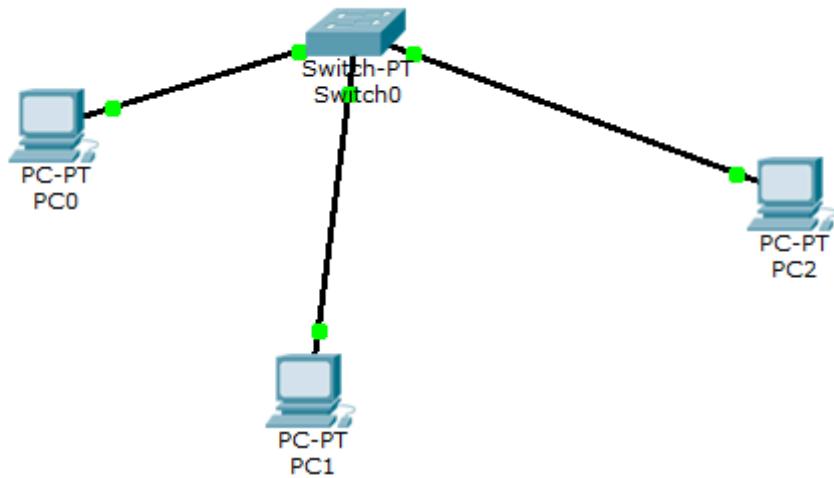


Screen shots/ output:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	





Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)

```
c:\>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 9ms, Average = 2ms
```

Observation:

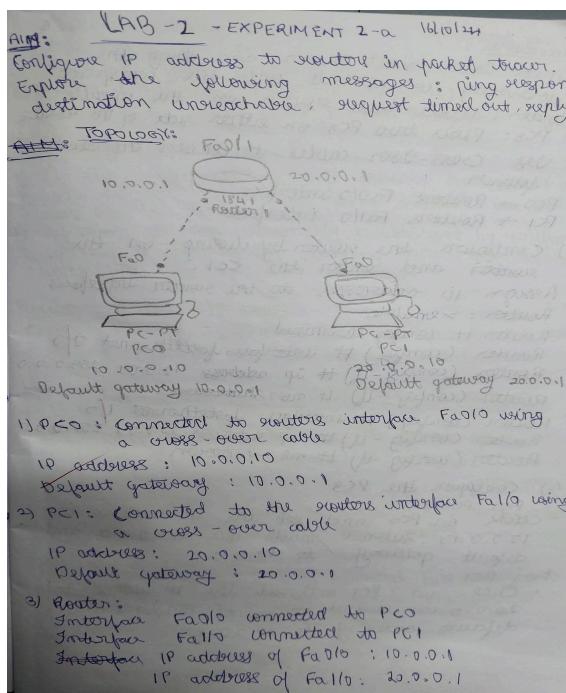
- Observation :
- 1) The hub broadcasts packets to all devices which may cause unnecessary traffic.
 - 2) The switch forwards packets only to the appropriate device by learning MAC addresses making it more efficient in reducing traffic.
- N
25/9/24*

Program 2-a

Aim of the program:

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

Procedure along with the topology:

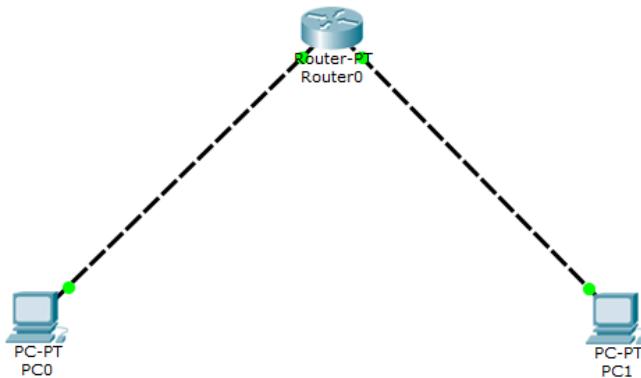


Procedure :

- Open Cisco packet tracer and drag the following components onto the workspace.
Router : Place one router in the middle.
PCs : Place two PCs on either side of the router.
- Use Cross-Over cables to connect the devices following:
PC0 → Router Fa0/0 interface
PC1 → Router Fa1/0 interface
- Configure the router by clicking on the router and enter the CLI.
Assign IP addresses to the router interfaces
Router > enable
Router # config terminal
Router (config) # interface fastethernet 0/0
Router (config-0/0) # ip address 20.0.0.1 255.0.0.0
Router (config-0/0) # no shutdown
Router (config) # interface fastethernet 1/0
Router (config-1/0) # ip address 20.0.0.1 255.0.0.0
Router (config-1/0) # no shutdown
- Configure the PCs.
For PC0 :
Click on PC0 and set the IP Address to 10.0.0.10, subnet mask to 255.0.0.0 and default gateway to 20.0.0.1.
For PC1 :
Click on PC1 and set the IP address to 20.0.0.10 subnet mask to 255.0.0.0 and default gateway to 20.0.0.1.

5) Test connectivity by opening the command prompt on PC0 and PC1.
Use the ping command to check connectivity
From PC0, ping PC1's IP (20.0.0.10)
From PC1, ping PC0's IP (10.0.0.10)

Screen shots/ output:



Router0

Physical	Config	CLI
IOS Command Line Interface		
<pre> Router>enable Router>config terminal Enter configuration commands, one per line. End with CNTL/Z. Router(config)#ip route 20.0.0.0 255.0.0.0 30.0.0.2 Router(config)# Router(config)#interface Serial2/0 Router(config-if)#exit Router(config)#exit Router# SYS-5-CONFIG_I: Configured from console by console Router#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area * - candidate default, U - per-user static route, o - ODR P - periodic downloaded static route Gateway of last resort is not set C 10.0.0.0/8 is directly connected, FastEthernet0/0 S 20.0.0.0/8 [1/0] via 30.0.0.2 C 30.0.0.0/8 is directly connected, Serial2/0 Router# Router#configure terminal Enter configuration commands, one per line. End with CNTL/Z. Router(config)# Router(config)#interface FastEthernet0/0 </pre>		
<input type="button" value="Copy"/> <input type="button" value="Paste"/>		

PC0

Physical	Config	Desktop	Custom Interface
<p>Command Prompt</p> <pre> PC>ping 30.0.0.2 Pinging 30.0.0.2 with 32 bytes of data: Reply from 30.0.0.2: bytes=32 time=13ms TTL=254 Reply from 30.0.0.2: bytes=32 time=5ms TTL=254 Reply from 30.0.0.2: bytes=32 time=10ms TTL=254 Reply from 30.0.0.2: bytes=32 time=7ms TTL=254 Ping statistics for 30.0.0.2: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum = 5ms, Maximum = 13ms, Average = 8ms PC>ping 20.0.0.1 Pinging 20.0.0.1 with 32 bytes of data: Reply from 20.0.0.1: bytes=32 time=1ms TTL=126 Reply from 20.0.0.1: bytes=32 time=7ms TTL=126 Reply from 20.0.0.1: bytes=32 time=14ms TTL=126 Reply from 20.0.0.1: bytes=32 time=7ms TTL=126 Ping statistics for 20.0.0.1: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum = 1ms, Maximum = 14ms, Average = 7ms </pre>			

Observation:

Observation :

- If the configurations and cabling are correct you will receive successful ping replies b/w the two PCs
- If there is no connectivity troubleshoot by verifying correct IP addressing, cabling type, both routers interfaces are up and running.

Router # show ip route B-B6

Codes - C - Connected, S - static I - IGRP, R - RIP, N - mobile
 0 - EIGRP, EX - EIGRP external, 0 - OSPF, 1A - OSPF intra area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2, E - EIGRP
 i - 1S-1S- L1 - 1S-1S level-1, L2 - 1S-1S level-2, ia-1S-1S inter area

* - candidate default, v - virtual link route, o - ODR
 P - periodic downloaded static route

Gateway of last resort is not set

C 10.0.0.9/8 is directly connected, Fast Ethernet 0/0
 C 20.0.0.0/8 is directly connected, Fast Ethernet 0/1

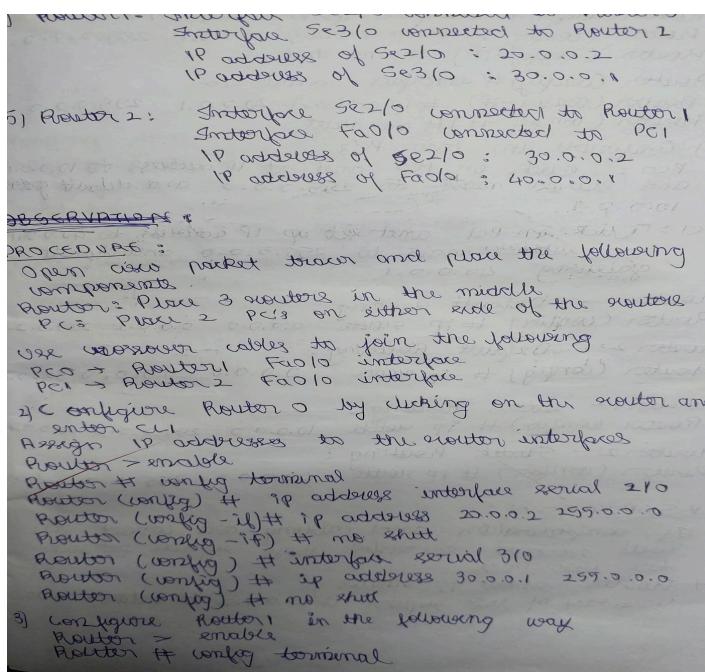
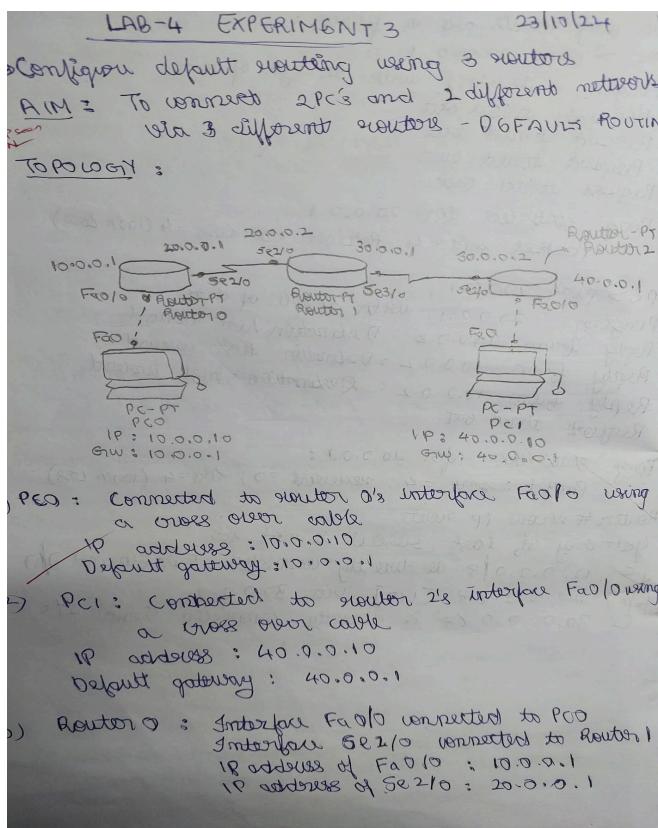
Ping 20.0.0.10
 Pinging 20.0.0.10 with 32 bytes of data.
 Reply from 20.0.0.10: bytes=32 time=0ms TTL=6
 Ping statistics for 20.0.0.10:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss).
 Approximate round trip times in milliseconds:
 Minimum = 0ms, Maximum = 3ms, Average = 0ms

Program 3

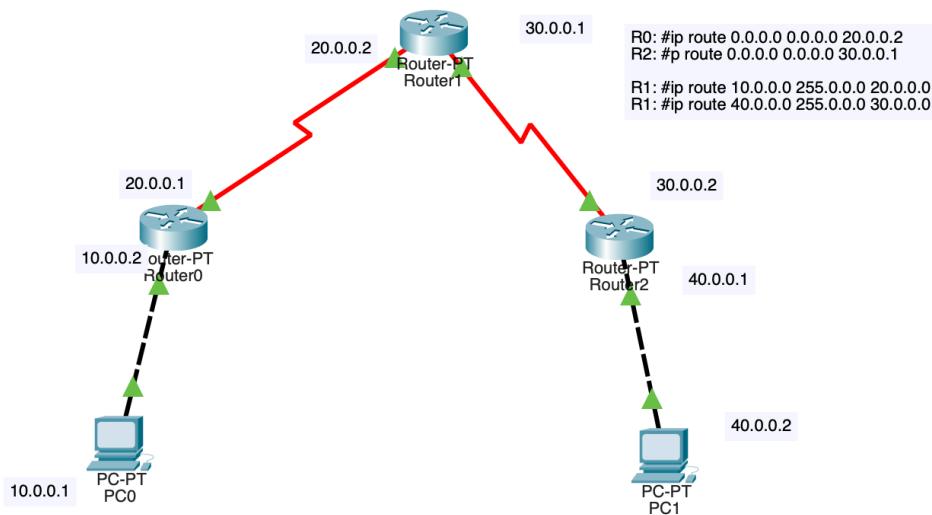
Aim of the program:

Configure default route, static route to the Router

Procedure along with the topology:



Screen shots/ output:



PC1

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

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

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=9ms TTL=125
Reply from 10.0.0.1: bytes=32 time=13ms TTL=125
Reply from 10.0.0.1: bytes=32 time=10ms TTL=125
Reply from 10.0.0.1: bytes=32 time=15ms TTL=125

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 15ms, Average = 11ms
PC>

```

PC1

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=9ms TTL=125
Reply from 10.0.0.1: bytes=32 time=13ms TTL=125
Reply from 10.0.0.1: bytes=32 time=10ms TTL=125
Reply from 10.0.0.1: bytes=32 time=15ms TTL=125

Ping statistics for 10.0.0.1:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 15ms, Average = 11ms
PC>

```

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Pinging 40.0.0.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 40.0.0.2:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>show ip route
Invalid Command.
PC>ping 40.0.0.2
Pinging 40.0.0.2 with 32 bytes of data:
Reply from 40.0.0.2: bytes=32 time=3ms TTL=125
Reply from 40.0.0.2: bytes=32 time=4ms TTL=123
Reply from 40.0.0.2: bytes=32 time=13ms TTL=123
Reply from 40.0.0.2: bytes=32 time=12ms TTL=123

Ping statistics for 40.0.0.2:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 13ms, Average = 8ms
PC>

```

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
<input checked="" type="radio"/>	Failed	PC0	PC2	ICMP	█	0.000	N	0	(edit)	(delete)
<input checked="" type="radio"/>	Successful	PC0	PC2	ICMP	█	0.000	N	1	(edit)	(delete)

New Delete Toggle PDU List Window

Router0

Physical Config CLI

IOS Command Line Interface

```

E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is not set

C 10.0.0.0/8 is directly connected, FastEthernet0/0
S 20.0.0.0/8 [1/0] via 30.0.0.2
    [1/0] via 40.0.0.2
C 30.0.0.0/8 is directly connected, Serial2/0
S 40.0.0.0/8 [1/0] via 20.0.0.2
Router#enable
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
        i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
        * - candidate default, U - per-user static route, o - ODR
        P - periodic downloaded static route

Gateway of last resort is not set

C 10.0.0.0/8 is directly connected, FastEthernet0/0
S 20.0.0.0/8 [1/0] via 30.0.0.2
    [1/0] via 40.0.0.2
C 30.0.0.0/8 is directly connected, Serial2/0
S 40.0.0.0/8 [1/0] via 20.0.0.2
Router#

```

Copy Paste

Observation:

QBS68XATIS N :

- 1) Interconnection and cabling are correct, you will receive successful ping response between the two PCs.
- 2) Router : show ip routes
Gateway of last resort is 20.0.0.2 to network 0.0.0.0

C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 20.0.0.0/8 is directly connected, Serial2/0
S 0.0.0.0/0 [1/0] via 20.0.0.2

The ping results are as follows:

PC > Ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Reply from 40.0.0.10 bytes=32 time=6ms TTL=125
Reply from 40.0.0.10 bytes=32 time=6ms TTL=125
Reply from 40.0.0.10 bytes=32 time=10ms TTL=125
Reply from 40.0.0.10 bytes=32 time=9ms TTL=125

Ping statistics for 40.0.0.10
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milliseconds:
Minimum = 6ms, Maximum = 10ms, Average = 7ms

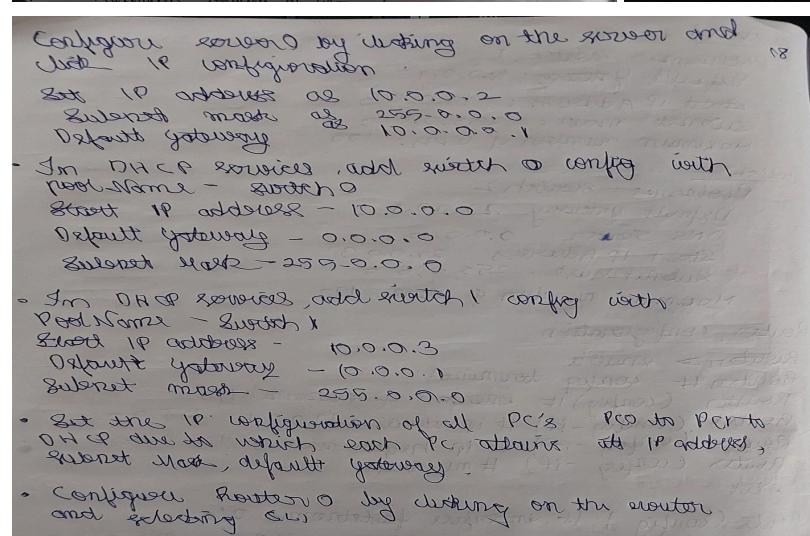
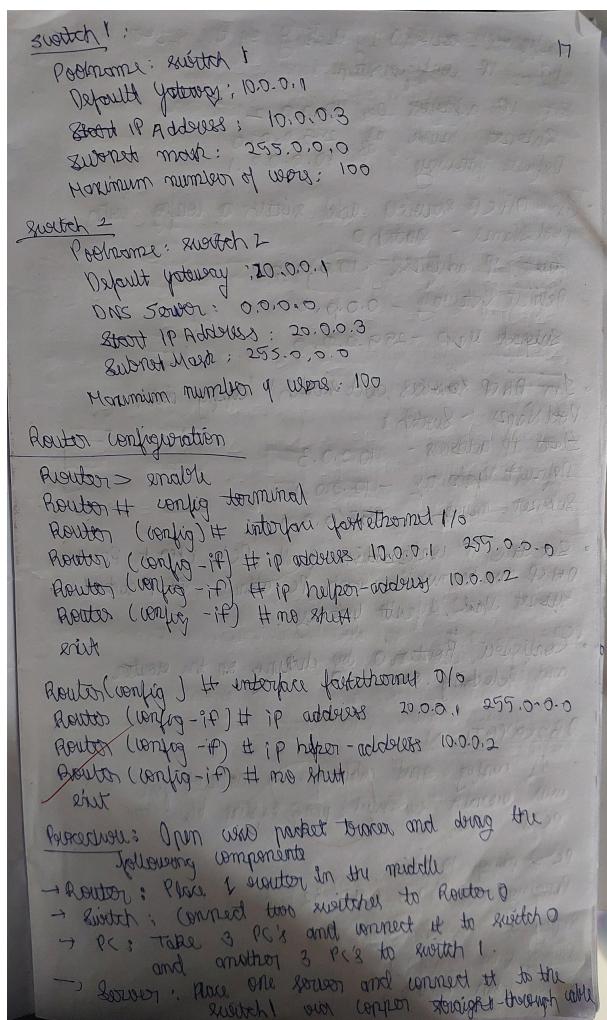
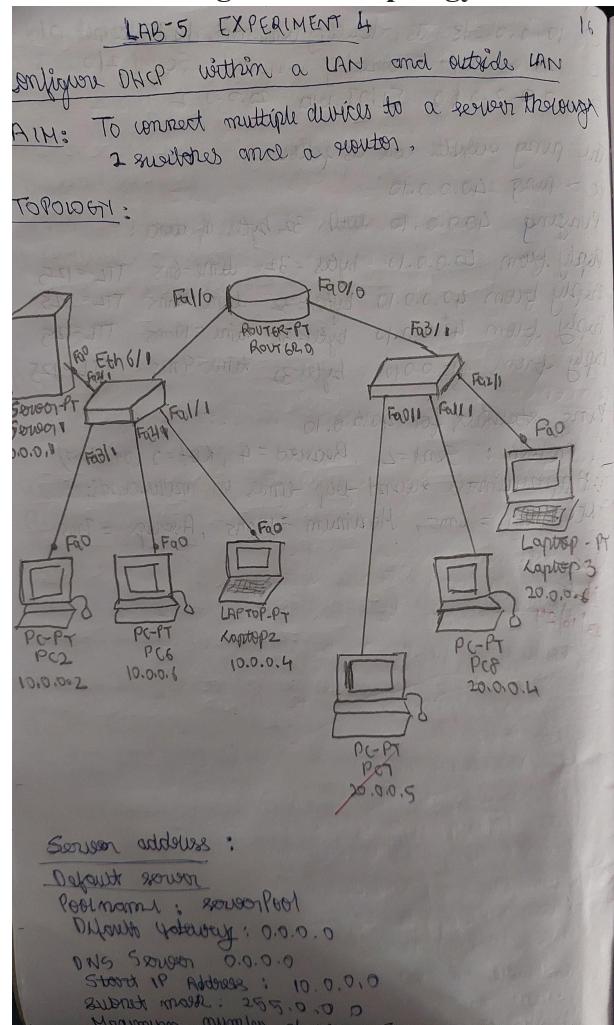
N
23/10/24

Program 4

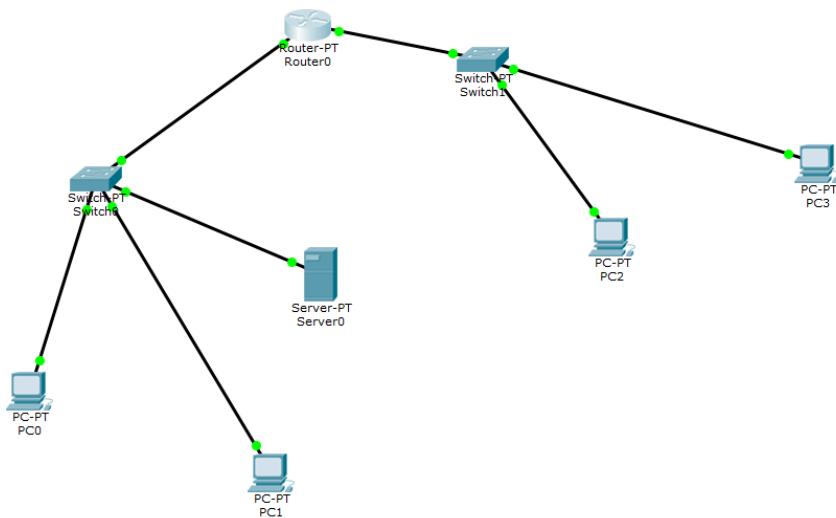
Aim of the program:

Configure IP address of the host using DHCP server within and outside a LAN.

Procedure along with the topology:



Screen shots/ output:



Server0

Physical Config Services Desktop Programming Attributes

SERVICES

- HTTP
- DHCP**
- DHCPv6
- TFTP
- DNS
- SYSLOG
- AAA
- NTP
- EMAIL
- FTP
- IoT
- VM Management
- Radius EAP

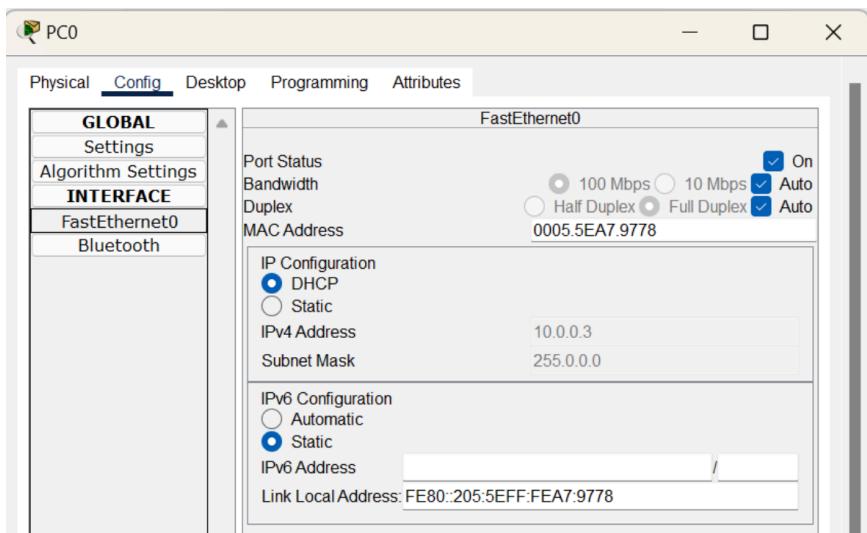
DHCP

Interface	FastEthernet0	<input checked="" type="radio"/> On	<input type="radio"/> Off
Pool Name	serverPool1		
Default Gateway	10.0.0.2		
DNS Server	10.0.0.1		
Start IP Address :	10	0	0
Subnet Mask:	255	0	0
Maximum Number of Users :	512		
TFTP Server:	0.0.0.0		
WLC Address:	0.0.0.0		

Add Save Remove

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max User	TFTP Server	WLC Address
serverPool1	10.0.0.2	10.0.0.1	10.0.0.0	255.0.0.0	512	0.0.0.0	0.0.0.0
serverPool2	20.0.0.1	10.0.0.1	20.0.0.0	255.0.0.0	512	0.0.0.0	0.0.0.0
serverPool	0.0.0.0	0.0.0.0	10.0.0.0	255.0.0.0	512	0.0.0.0	0.0.0.0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
●	Successful	PC1	Laptop1	ICMP		0.004	N	1	(edit)	



Observation:

OBSERVATION

- If config and cabling are correct, you will receive successful ping replies b/w two PC's

PC > ping 10.0.0.3
 Pinging 10.0.0.3 with 32 bytes of data

Reply from 10.0.0.3 : bytes = 32 time = 1ms TTL = 128
 Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL = 128
 Reply from 10.0.0.3 : bytes = 32 time = 0ms TTL = 128

Reply from 10.0.0.3 : bytes = 32 time = 2ms TTL = 120

ping statistics for 10.0.0.3:

Packets : Sent = 4, Received = 4, Loss = 0 (0% loss)

Approximate round trip times in milliseconds:

Minimum = 0ms, Maximum = 2ms, Average = 0ms

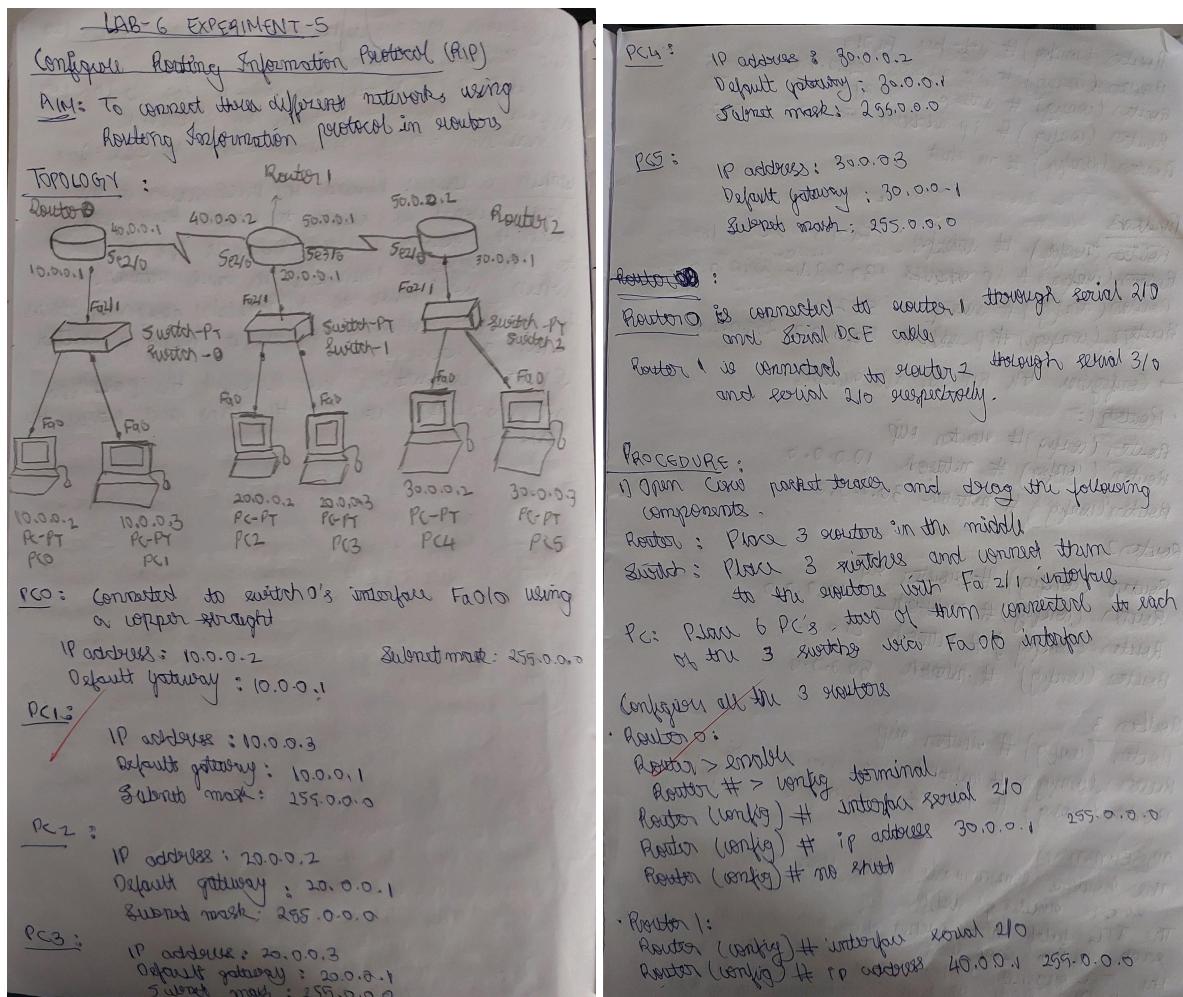
Within a LAN : Placing the DHCP server in the same subnet as clients to ensure broadcasts reach the server directly. Dynamic IPs are given to the systems connected in same network. When we have to dynamically assign IP address to another network we do it using a router and a server.

If the connections are successful the IP addresses are assigned within the LAN and outside the LAN

Program 5

Aim of the program:
Configure RIP routing Protocol in Routers

Procedure along with the topology:

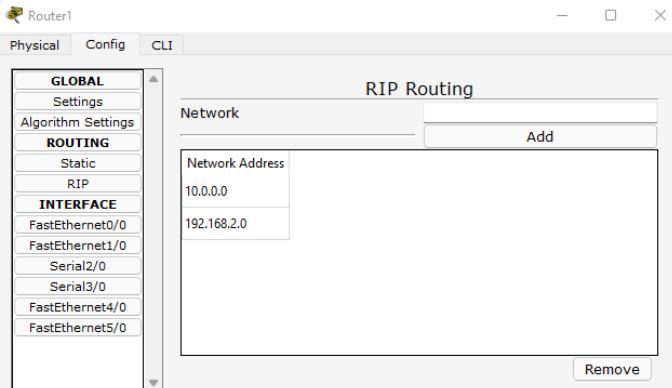
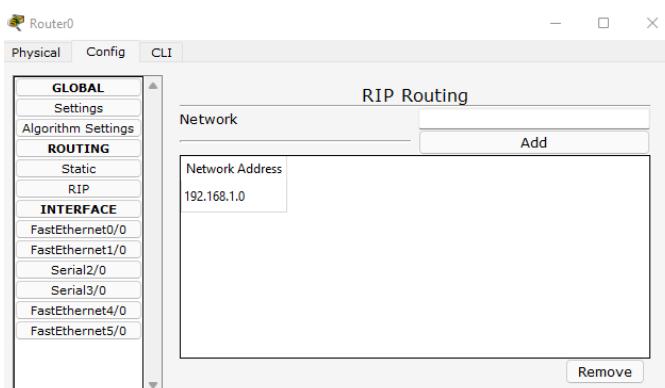
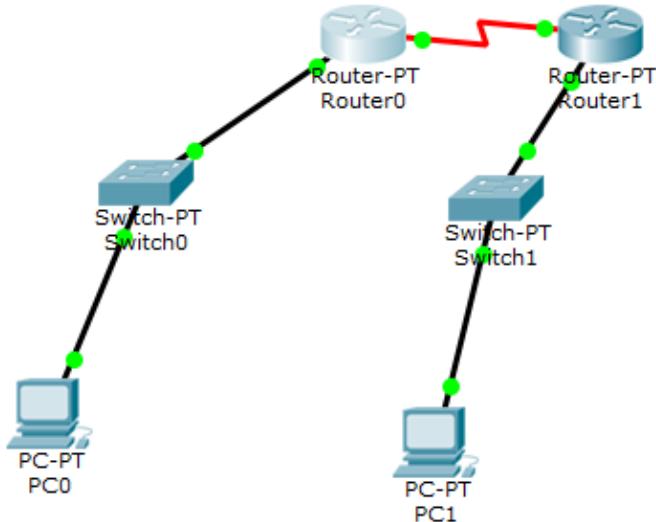


```

Router (config)# interface Fast1/0
Router (config)# ip address 20.0.0.1 255.0.0.0
Router (config)# interface Se3/0
Router (config)# ip address 50.0.0.1 255.0.0.0
Router (config)# no shut
Router (config)#
Router2
Router (config)# interface Fast1/0
Router (config)# ip address 30.0.0.1 255.0.0.0
Router (config)# interface Se2/0
Router (config)# ip address 50.0.0.2 255.0.0.0
Router (config)#
→ Configuration the RIP for all routers
Router1:
Router (config)# router rip
Router (config)# network 10.0.0.0
Router (config)# network 30.0.0.0
Router (config)#
Router2:
Router (config)# router rip
Router (config)# network 40.0.0.0
Router (config)# network 20.0.0.0
Router (config)#
Router3:
Router (config)# router rip
Router (config)# network 30.0.0.0
Router (config)#

```

Screen shots/ output:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Failed	PC0	PC1	ICMP	Green	0.000	N	0	(edit)	(delete)
●	Successful	Router0	Router1	ICMP	Blue	0.000	N	1	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	Brown	0.000	N	2	(edit)	(delete)

Router0

Physical Config CLI

IOS Command Line Interface

```

Router>enable
Router>configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#network 40.0.0.0
Router(config-router)#exit
Router(config)#exit
Router#
*SYS-5-CONFIG_I: Configured from console by console

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
      * - candidate default, U - per-user static route, o - ODR
      P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
R    20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:12, Serial2/0
R    30.0.0.0/8 [120/2] via 40.0.0.2, 00:00:12, Serial2/0
C    40.0.0.0/8 is directly connected, Serial2/0
R    50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:12, Serial2/0
Router#

```

Copy Paste

PC0

Physical Config Desktop Custom Interface

Command Prompt

```

Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 30.0.0.2: bytes=32 time=9ms TTL=125
Reply from 30.0.0.2: bytes=32 time=8ms TTL=125
Reply from 30.0.0.2: bytes=32 time=10ms TTL=125

Ping statistics for 30.0.0.2:
  Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 10ms, Average = 9ms
PC>

```

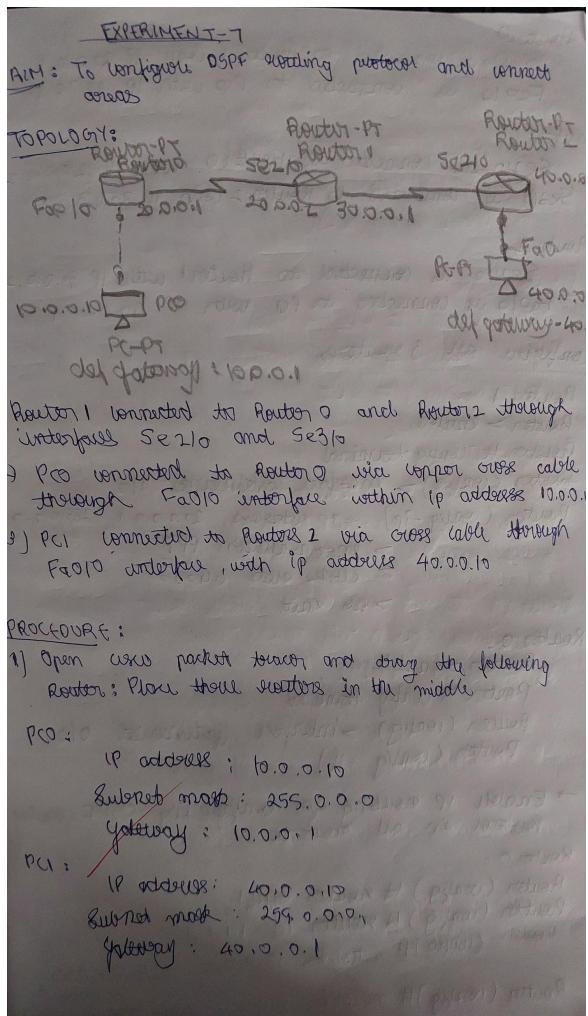
Observation:

The TTL field in a packet decrements by 1 at each router hop to prevent infinite loops. If the TTL reaches 0, the router discards the packet and sends an ICMP ("Time Exceeded") message back to the sender.

Program 6

Configure OSPF routing protocol

Procedure along with the topology



Router 0:
S2/0 is connected to Router 1 with IP 20.0.0.1
Fa0/0 is connected to PC0 with IP 10.0.0.1

Router 1:
S2/0 is connected to Router 0 with IP 20.0.0.1
S3/0 is connected to Router 2 with IP 30.0.0.1

Router 2:
S2/0 is connected to Router 1 with IP 30.0.0.1
Fa0/0 is connected to PC1 with IP 40.0.0.1

Configuration all 3 routers

Router 1:
Router > enable
Router # config terminal
Router (config) > interface fastethernet 0/0
Router (config-ip) > ip address 20.0.0.1 255.0.0.0
Router (config-ip) > encapsulation PPP
Router (config-ip) > clock rate 64000
Router (config-ip) > no shutdown

Router 0:
Router > enable
Router # config terminal
Router (config) > interface fastethernet 0/0
Router (config-if) > no shutdown

→ Enable IP routing for configuring OSPF area 0 in all routers

Router 0:
Router (config) # router ospf 1
Router (config) # router-id 1.1.1.1
Router (config) # network 10.0.0.0 0.255.255.255 area 0
Router (config) # network 20.0.0.0 0.255.255.255 area 0

```

Router (config) # exit
Router 1:
Router (config) # router ospf 1
Router (config) # router-id 2.2.2.2
Router (config) # network 20.0.0.0 0.255.255.255 00
Router (config) # network 20.0.0.0 0.255.255.255 00
Router (config) # exit

Router 2:
Router (config) # router ospf 1
Router (config) # router-id 3.3.3.3
Router (config) # network 30.0.0.0 0.255.255.255 00
Router (config) # network 40.0.0.0 0.255.255.255 00
Router (config) # exit

→ Configuration loopback address to routers:-
R0 (config) # interface loopback 0
R0 (config) # ip address 172.16.1.252 255.255.255.0
R0 (config) # no shutdown
R1 (config) # interface loopback 0
R1 (config) # ip address 172.16.1.253 255.255.255.0
R1 (config) # no shutdown
R2 (config) # interface loopback 0
R2 (config) # ip address 172.16.1.254 255.255.255.0
R2 (config) # no shutdown

```

Collects vertical link between R0, R1, R2

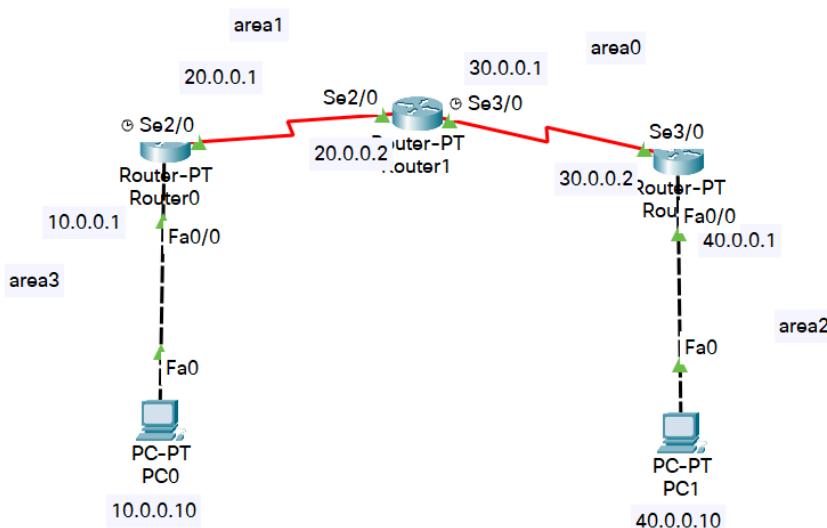
Router 0

R0 (config) # router ospf 1
R0 (config) # area 0 virtual-link 2.2.2.2
R0 (config) # exit

Router 1

R1 (config) # router ospf 1
R1 (config) # area 0 virtual-link 1.1.1.1
R1 (config) # exit

Screen shots/ output



Router0

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Fa0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed
state to up

Router(config-if)#exit
Router(config)#interface Se2/0
Router(config-if)#ip address 20.0.0.1 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#clock rate 64000
Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial2/0, changed state to down
Router(config-if)#exit
Router(config)#+
```

Router1

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Se2/0
Router(config-if)#ip address 20.0.0.2 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to
up

Router(config-if)#exit
Router(config)#interface Se3/0
Router(config-if)#ip address 30.0.0.1 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#clock rate 64000
Router(config-if)#no shutdown
^
% Invalid input detected at '^' marker.

Router(config-if)#no shutdown
```

Router2

```

Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Fa0/0
Router(config-if)#ip address 40.0.0.1 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed
state to up
%IP-4-DUPADDR: Duplicate address 40.0.0.1 on FastEthernet0/0, sourced by
000D.BDDA.0123

Router(config-if)#exit
Router(config)#interface Se3/0
Router(config-if)#ip address 30.0.0.2 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface Serial3/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to
up

```

OSPF Routing Protocol

Router0

```

Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#router-id 1.1.1.1
Router(config-router)#network 10.0.0.0 0.255.255.255 area 3
Router(config-router)#network 20.0.0.0 0.255.255.255 area 1
Router(config-router)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#sho
00:27:19: *OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on Serial2/0 from LOADING to FULL, Loading Done
w ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
      20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C          20.0.0.0/8 is directly connected, Serial2/0
C          20.0.0.2/32 is directly connected, Serial2/0
O IA 30.0.0.0/8 [110/128] via 20.0.0.2, 00:00:02, Serial2/0
O IA 40.0.0.0/8 [110/129] via 20.0.0.2, 00:00:02, Serial2/0

```

Router1

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#router-id 2.2.2.2
Router(config-router)#network 20.0.0.0 0.255.255.255 area 1
Router(config-router)#network 30.0.0.0 0.255.255.255 area 0
Router(config-router)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

00:26:21: %OSPF-5-ADJCHG: Process 1, Nbr 3.3.3.3 on Serial3/0 from LOADING to FULL, Loading Done
00:27:18: %OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on Serial2/0 from LOADING to FULL, Loading Done

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
      * - candidate default, U - per-user static route, o - ODR
      P - periodic downloaded static route

Gateway of last resort is not set

      20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        20.0.0.0/8 is directly connected, Serial2/0
C        20.0.0.1/32 is directly connected, Serial2/0
      30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        30.0.0.0/8 is directly connected, Serial3/0
C        30.0.0.2/32 is directly connected, Serial3/0
O IA 40.0.0.0/8 [110/65] via 30.0.0.2, 00:02:00, Serial3/0
```

Router2

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#router-id 3.3.3.3
Router(config-router)#network 40.0.0.0 0.255.255.255 area 2
Router(config-router)#network 30.0.0.0 0.255.255.255 area 0
Router(config-router)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#
00:26:19: %OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on Serial3/0 from LOADING to FULL, Loading Done

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
      * - candidate default, U - per-user static route, o - ODR
      P - periodic downloaded static route

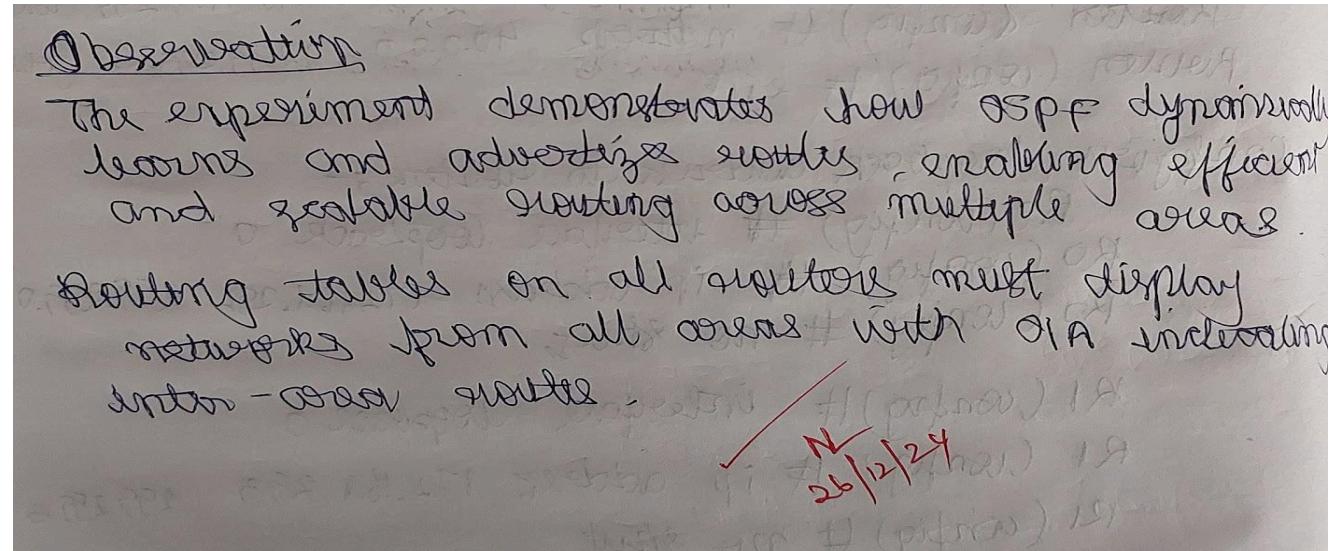
Gateway of last resort is not set

O IA 20.0.0.0/8 [110/128] via 30.0.0.1, 00:02:45, Serial3/0
      30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        30.0.0.0/8 is directly connected, Serial3/0
C        30.0.0.1/32 is directly connected, Serial3/0
C        40.0.0.0/8 is directly connected, FastEthernet0/0
```

Pinging

```
C:\>ping 40.0.0.10  
Pinging 40.0.0.10 with 32 bytes of data:  
Reply from 40.0.0.10: bytes=32 time=24ms TTL=125  
Reply from 40.0.0.10: bytes=32 time=18ms TTL=125  
Reply from 40.0.0.10: bytes=32 time=18ms TTL=125  
Reply from 40.0.0.10: bytes=32 time=20ms TTL=125  
  
Ping statistics for 40.0.0.10:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
    Approximate round trip times in milli-seconds:  
        Minimum = 18ms, Maximum = 24ms, Average = 20ms
```

Observation

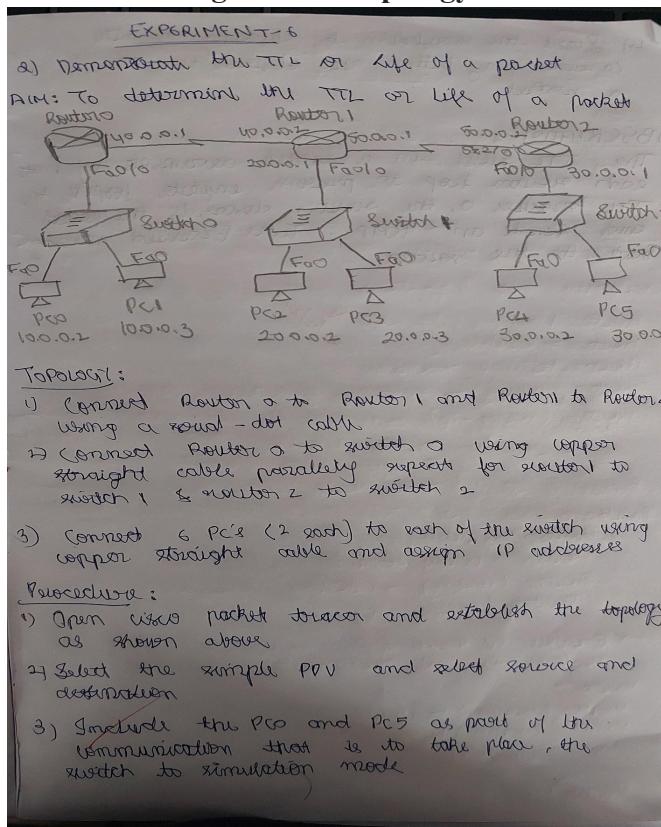


Program 7

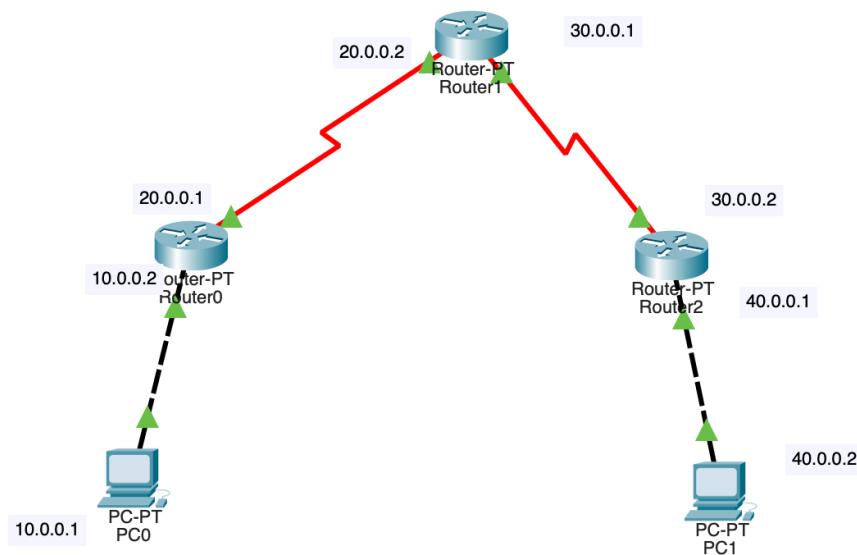
Aim of the program:

Demonstrate the TTL/ Life of a Packet

Procedure along with the topology:



Screen shots/ output:



PDU Information at Device: Router0

OSI Model Inbound PDU Details Outbound PDU Details

PDU Formats

Ethernet II					
0	4	8	14	19	Bytes
PREAMBLE: 101010...1011		DEST MAC: 0001.C780.D8C7		SRC MAC: 00E0.F711.E727	
TYPE: 0x800		DATA (VARIABLE LENGTH)		FCS: 0x0	

IP					
0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 28		
ID: 0x4		0x0	0x0		
TTL: 255	PRO: 0x1	CHKSUM			
SRC IP: 10.0.0.1					
DST IP: 40.0.0.2					
OPT: 0x0		0x0			
DATA (VARIABLE LENGTH)					

ICMP					
0	8	16	31 Bits		
TYPE: 0x8	CODE: 0x0	CHECKSUM			
ID: 0x5	SEQ NUMBER: 4				

PDU Information at Device: Router1

OSI Model Inbound PDU Details Outbound PDU Details

PDU Formats

HDLC					
0	8	16	32	32+x	48+x 56+ Bits
FLG: 0111	ADR: 0x8f	CONTROL: 0x0	DATA: (VARIABLE LENGTH)		FCS: 0x0
1110					FLG: 0111

IP					
0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 28		
ID: 0x6		0x0	0x0		
TTL: 253	PRO: 0x1	CHKSUM			
SRC IP: 10.0.0.1					
DST IP: 40.0.0.2					
OPT: 0x0		0x0			
DATA (VARIABLE LENGTH)					

ICMP					
0	8	16	31 Bits		
TYPE: 0x8	CODE: 0x0	CHECKSUM			
ID: 0x7	SEQ NUMBER: 6				

PDU Information at Device: Router2

OSI Model Inbound PDU Details Outbound PDU Details

PDU Formats

HDLC					
0	8	16	32	32+x	48+x 56+ Bits
FLG: 0111	ADR: 0x8f	CONTROL: 0x0	DATA: (VARIABLE LENGTH)		FCS: 0x0
1110					FLG: 0111

IP					
0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 28		
ID: 0x4		0x0	0x0		
TTL: 124	PRO: 0x1	CHKSUM			
SRC IP: 40.0.0.2					
DST IP: 10.0.0.1					
OPT: 0x0		0x0			
DATA (VARIABLE LENGTH)					

ICMP					
0	8	16	31 Bits		
TYPE: 0x0	CODE: 0x0	CHECKSUM			
ID: 0x7	SEQ NUMBER: 6				

Observation:

QUESTION: The TTL field in a packet decrements by 1 at each router hop to prevent infinite loops. If TTL reaches 0, the router discards the packet and sends an ICMP - "Time Exceeded" message back to the sender.

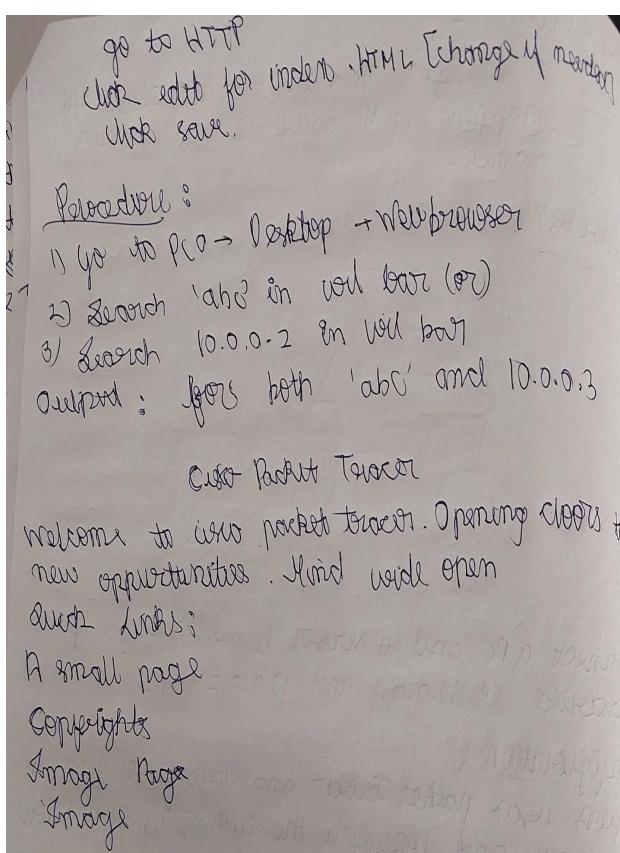
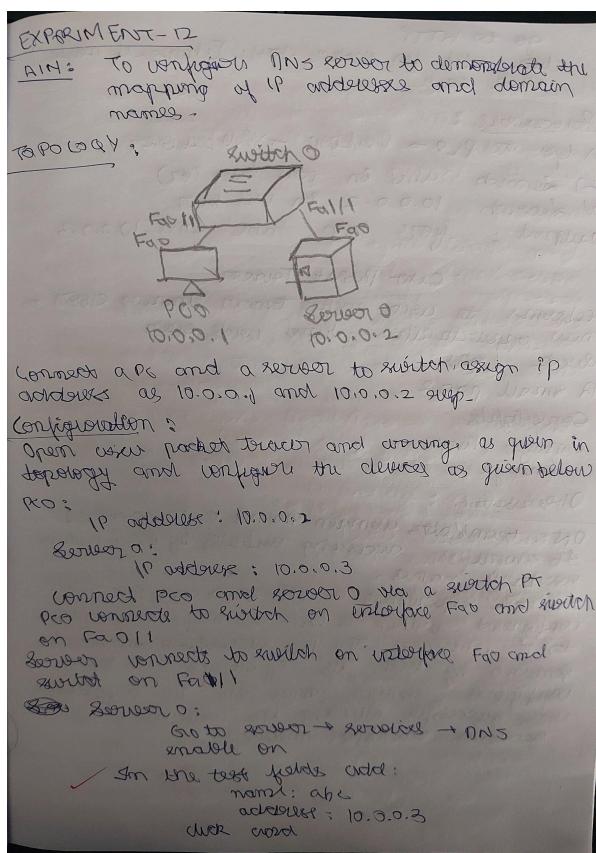
✓
25/12/24

Program 8

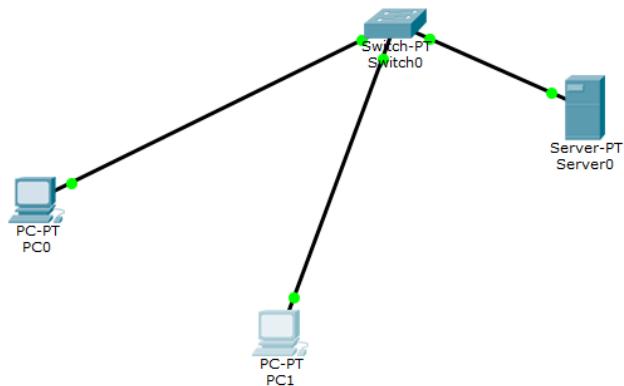
Aim of the program:

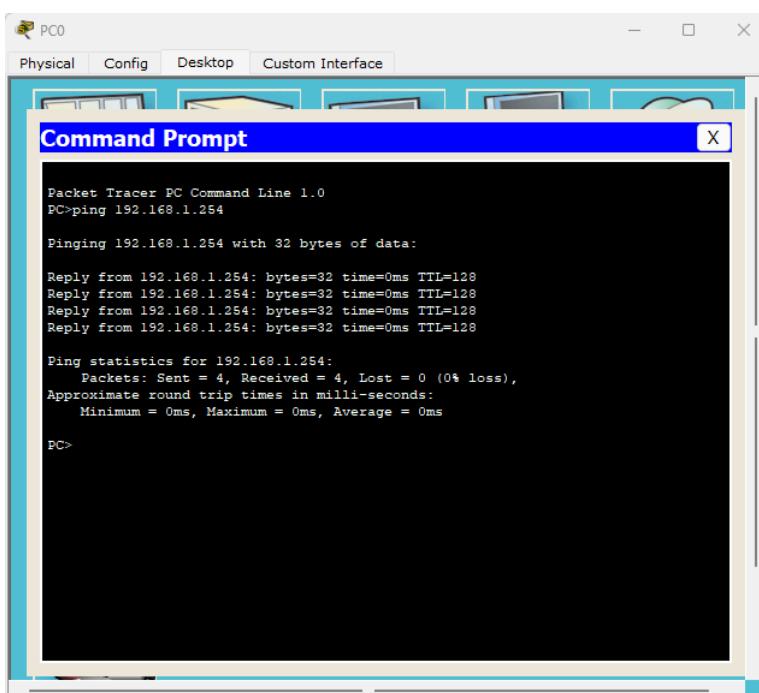
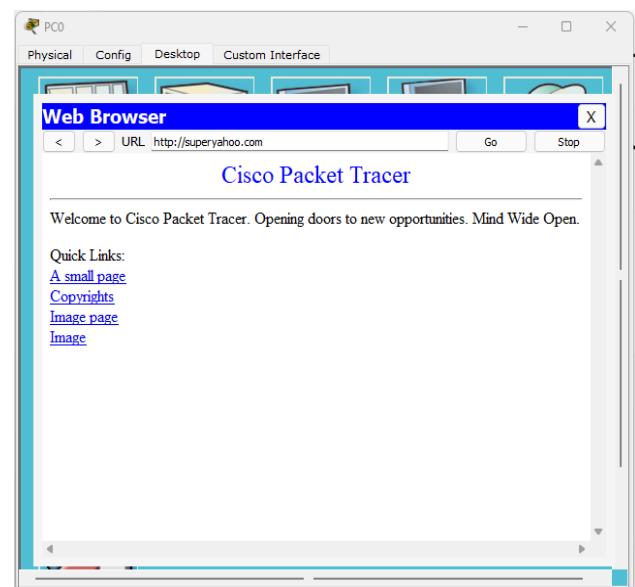
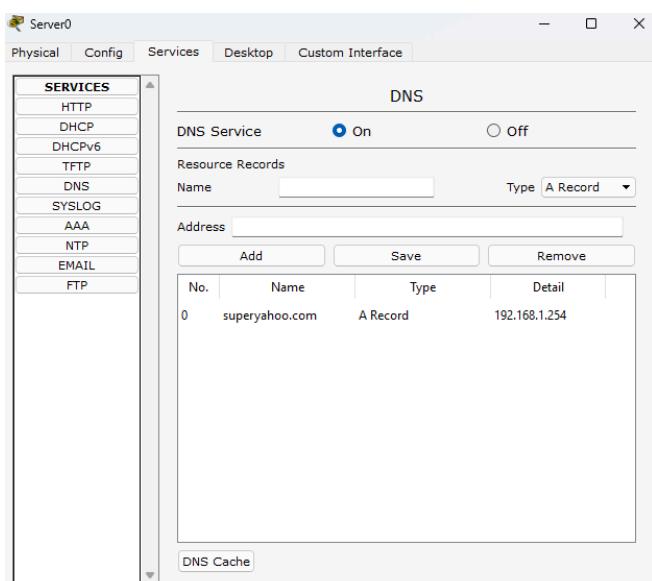
Configure Web Server, DNS within a LAN.

Procedure along with the topology:



Screen shots/ output:





Observation:

Observation :

DNS translates domain names to IP addresses to simplify accessing websites by using human-readable names. In this experiment a web server and DNS were configured within a LAN to map domain names to IP addresses. The PCs successfully accessed the servers by both its IP address and the configured domain name 'abc'. The configuration was successful allowing the web page to be accessed via both methods.

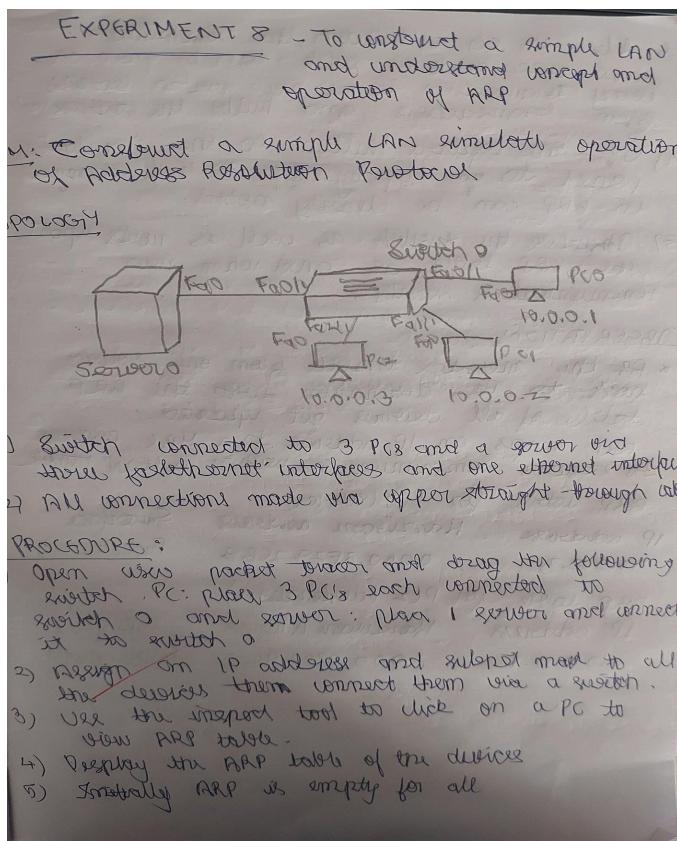
✓
20/11/24

Program 9

Aim of the program:

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

Procedure along with the topology:



- 6) A LAN in case of injection, the command = show mac address table can be given on every transmission to see how switch learns from broadcast and build the address table
- 7) Use the capture button in the simulation panel to go step by step so that changes in ARP can be clearly noted.
- 8) Observe the switch as well as nodes until the ARP table is built when new communication starts.

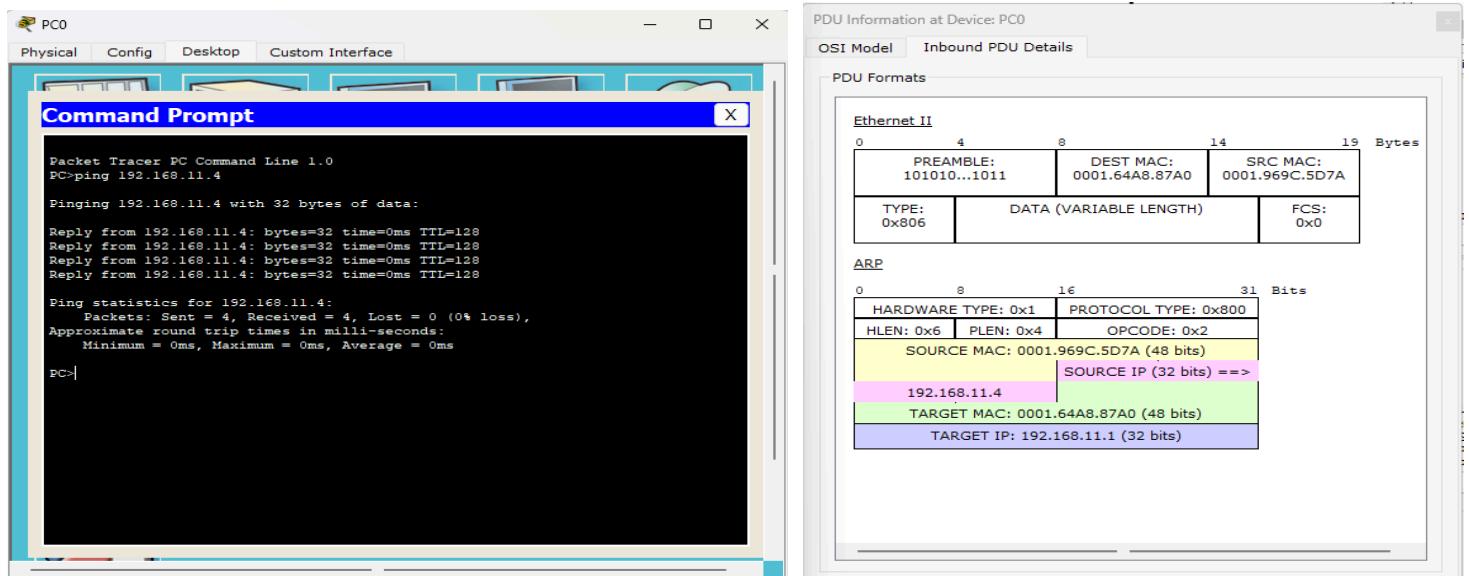
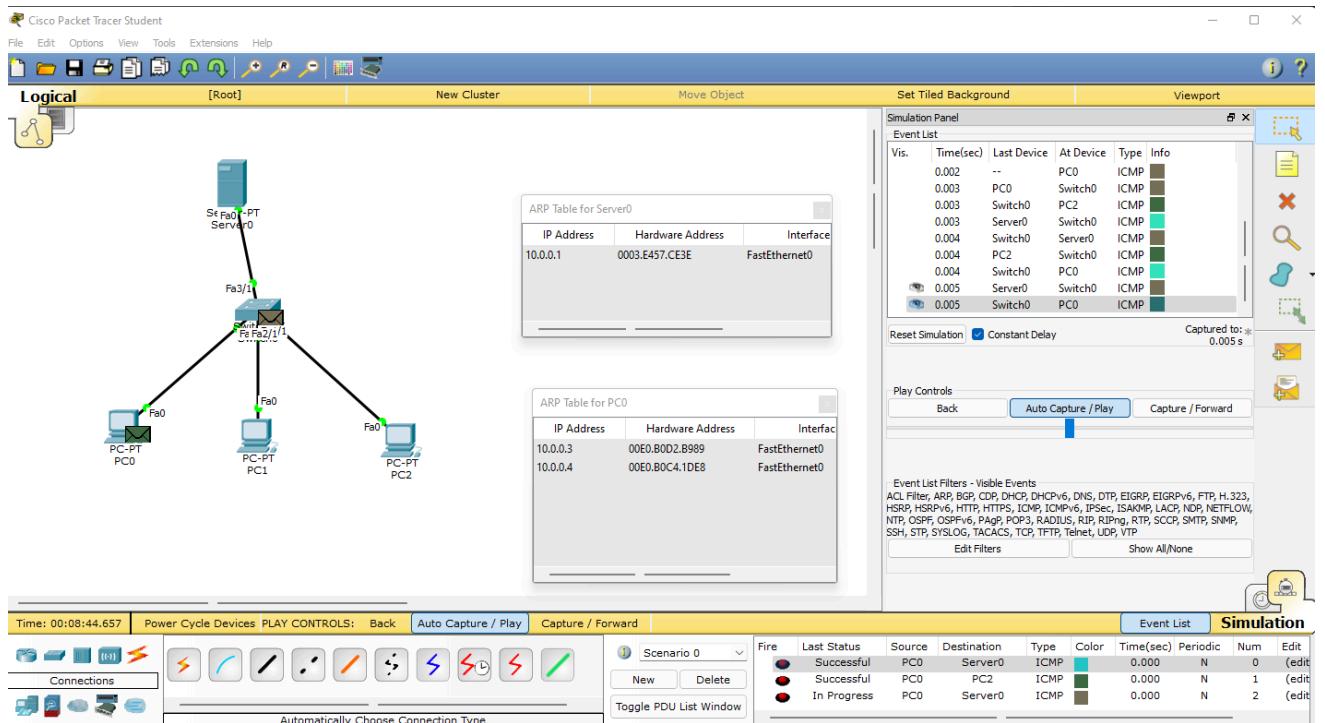
Screen shots/ output:

Screenshot 1: Cisco Packet Tracer Simulation View

Vlan	DeviceID	Last Device	Air Dickey	Type	Info
0.002	Switch0	PC1	ARP		
0.003	Switch0	PC0	ARP		
0.004	Switch0	PC1	ARP		
0.005	PC0	Switch0	ICMP		
0.006	PC0	Server0	ICMP		
0.007	PC0	Switch0	ICMP		
0.008	PC0	Server0	ICMP		

Screenshot 2: Cisco Packet Tracer Realtime View

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
Successful	PC0	Server0	ICMP	Cyan	0.000	N	0	(edit)	
Successful	PC0	PC2	ICMP	Green	0.000	N	1	(edit)	
In Progress	PC0	Server0	ICMP	Dark Green	0.000	N	2	(edit)	



Observation:

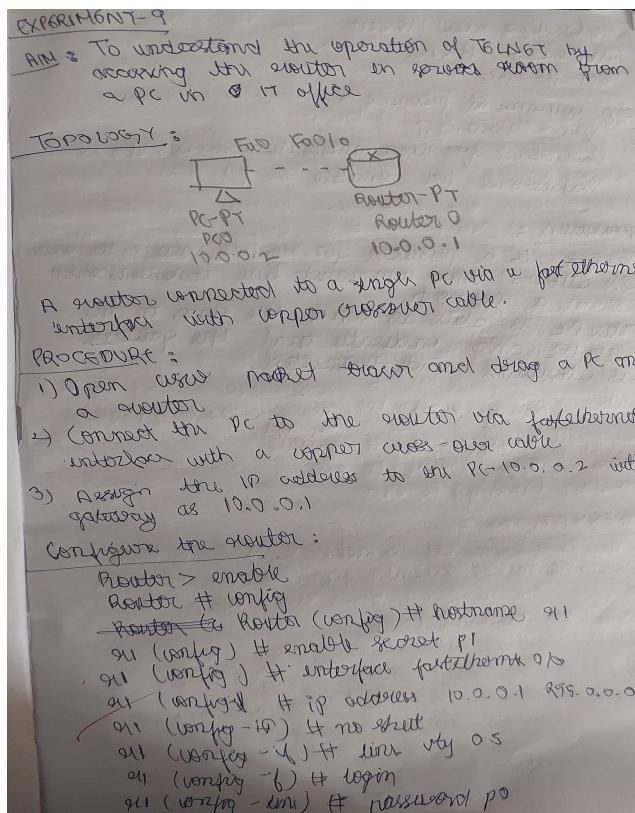
<u>OBSERVATION :</u>			
* As the message travels from one source host to the destination host the ARP tables of all devices get updated.			
ARP maps an IP address to a MAC address			
To enable communication within a local network			
ARP table for PC1 (source)			
IP address	Hardware Address	Interface	
10.0.0.3	0060.2F29.26B8	Fast Ethernet	
ARP table for PC2 (destination)			
IP address	Hardware Address	Interface	
10.0.0.1	0000.0302.960B	Fast Ethernet	
			26/12/21

Program 10

Aim of the program:

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

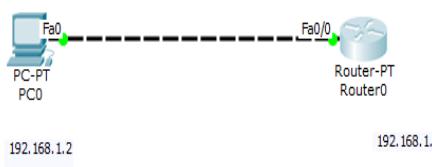
Procedure along with the topology:



R1 (config-line) # exit
 R1 (config) # exit
 R1 # wr

In command prompt :
 rung 192.0.0.1
 Password for user authentication is P0
 Password for enable is p1

Screen shots/ output:



Router0 Physical Config CLI

IOS Command Line Interface

```
Router(config)#ip address 192.168.1.1 255.255.255.0
Router(config)#exit
Router(config)#hostname R1
R1(config)#enable secret rp
R1(config)#int g0/0/0
R1(config)#line vty 0 5
R1(config-line)#login
# Login disabled on line 132, until 'password' is set
# Login disabled on line 133, until 'password' is set
# Login disabled on line 134, until 'password' is set
# Login disabled on line 135, until 'password' is set
# Login disabled on line 136, until 'password' is set
# Login disabled on line 137, until 'password' is set
R1(config-line)#password tp
R1(config-line)#exit
R1(config)#
R1(config)#exit
R1#
SYS-5-CONFIG_I: Configured from console by console

R1#wr
Building configuration...
[OK]
R1#
R1#
```

Copy Paste

PC0 Physical Config Desktop Custom Interface

Command Prompt X

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=2ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms

PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open

User Access Verification

Password:
R1>enable
R1>enable
Password:
R1#
```

Observation:

Observation :

Telnet is a protocol for remote access to server. It allows command line communication over a network. The PC is able to send the data to the gateway and indicates that it is available and connected.

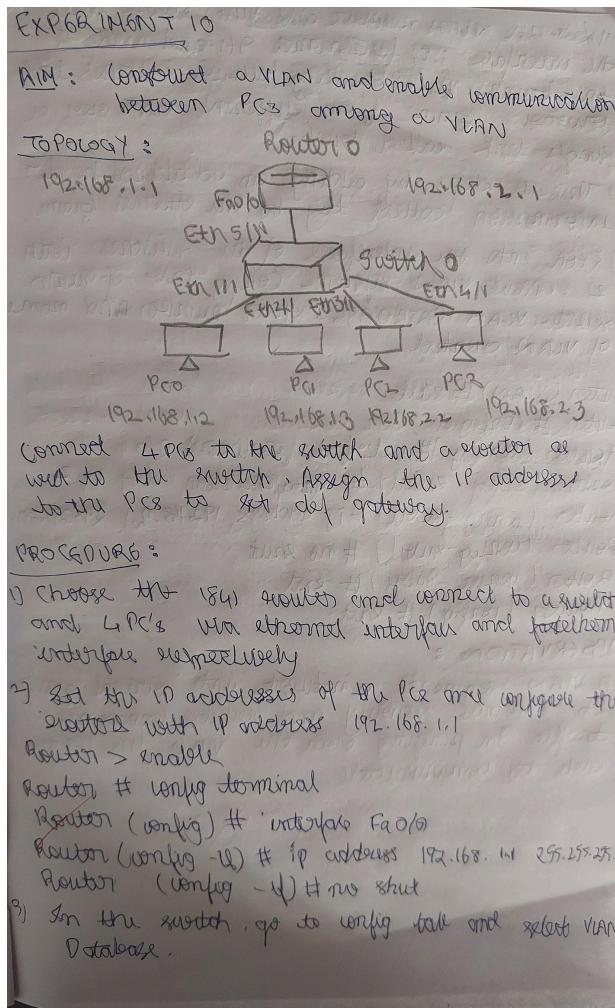
N
25/12/21

Program 11

Aim of the program:

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

Procedure along with the topology:



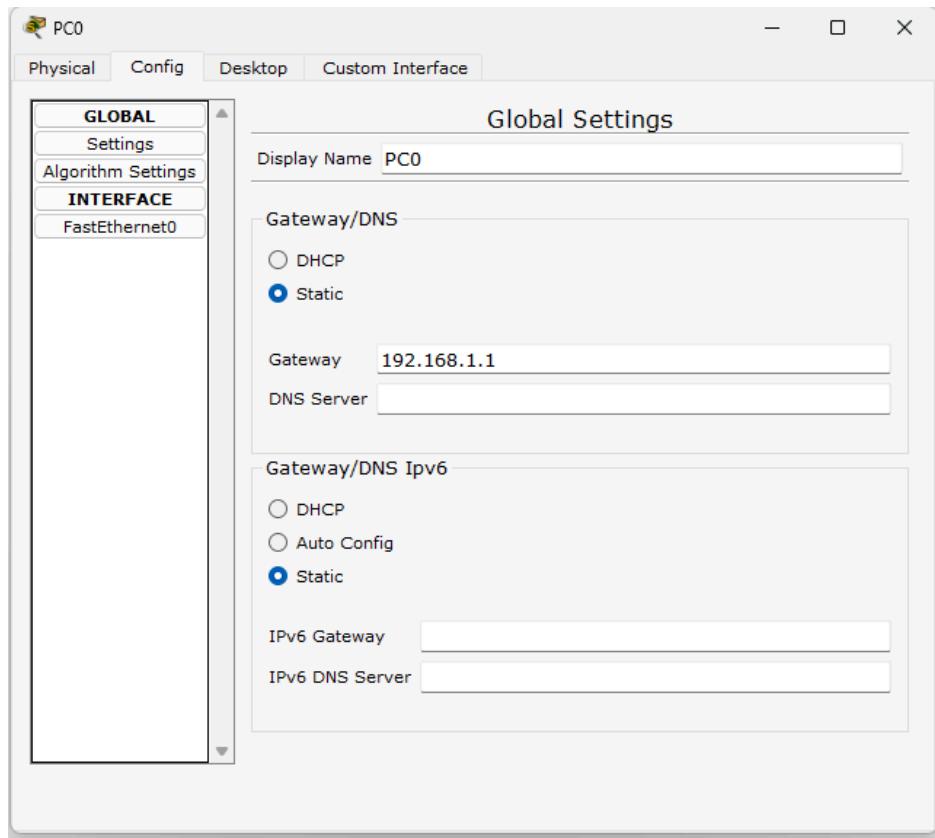
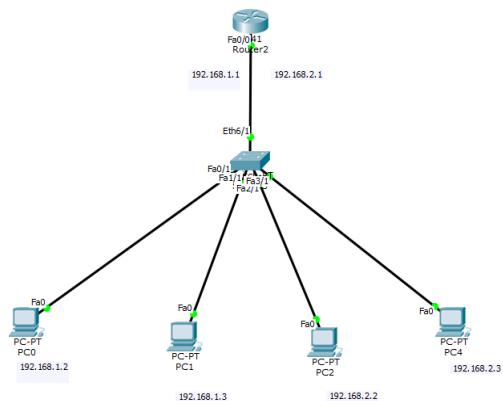
Set the VLAN number and VLAN name over all interface i.e., fastethernet 9/1 and make it the trunk. VLAN trunking allows switches to forward frame from different VLAN over a single link called trunk

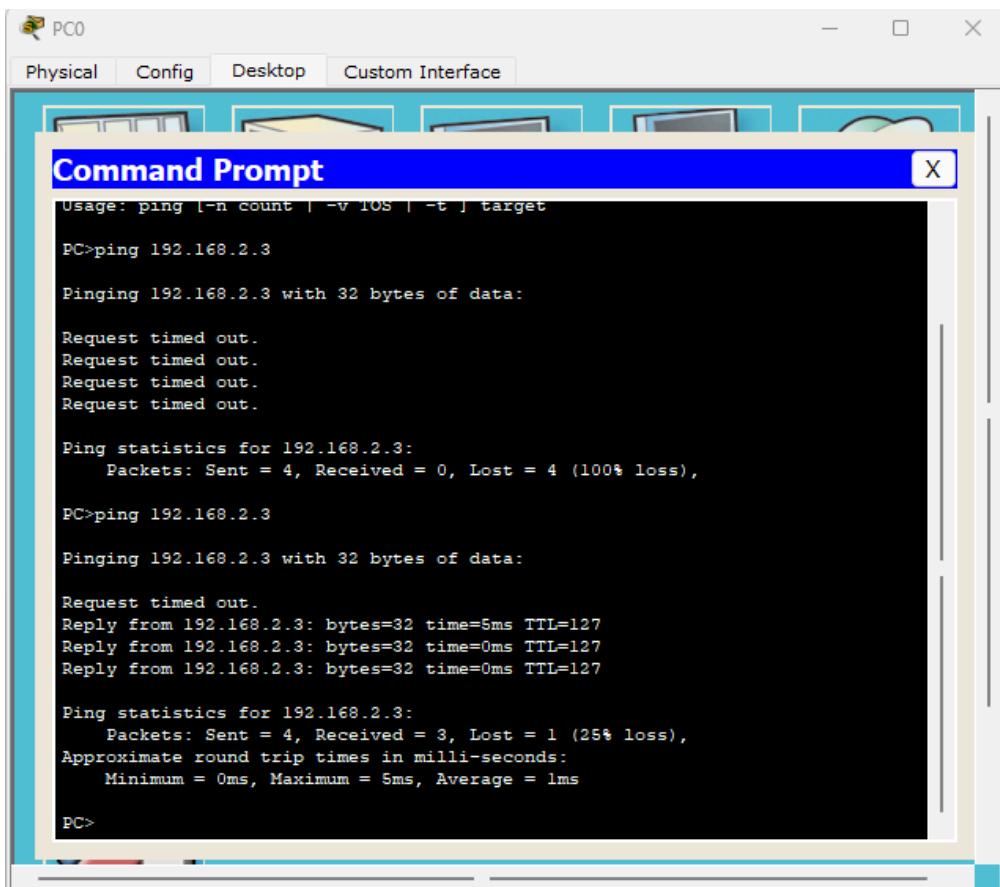
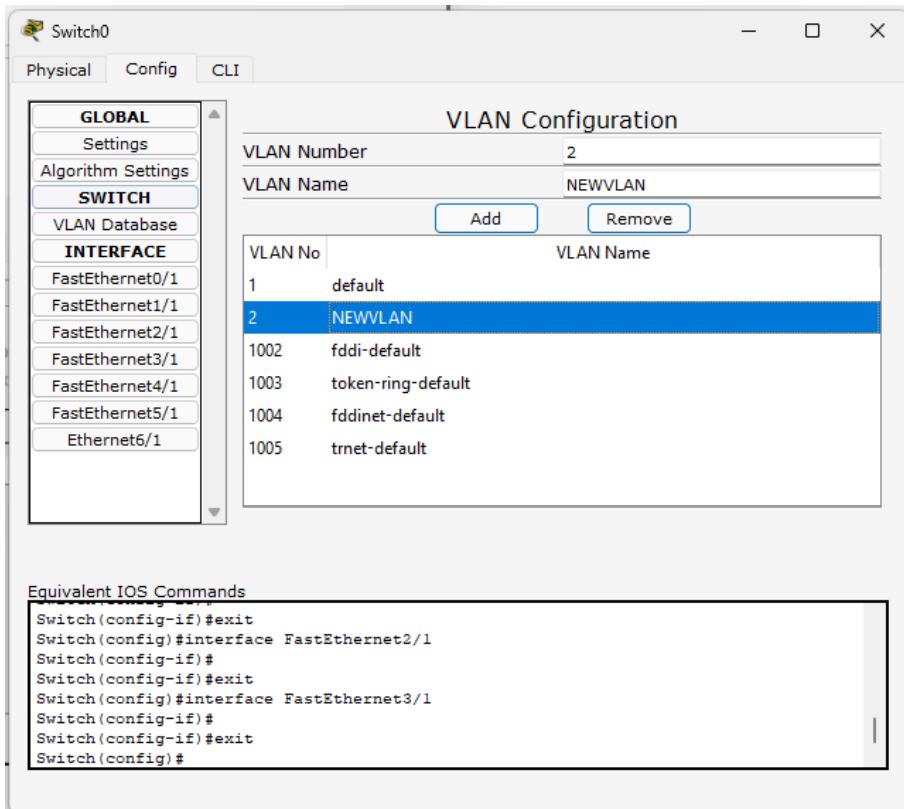
- 1) This is done by adding an additional header information called tag to the ethernet frame
- 2) Look into the interface of the switches and 2 new VLAN systems Config tab of switch

```
Select VLAN DATABASE - enter number and no of VLAN created -
```

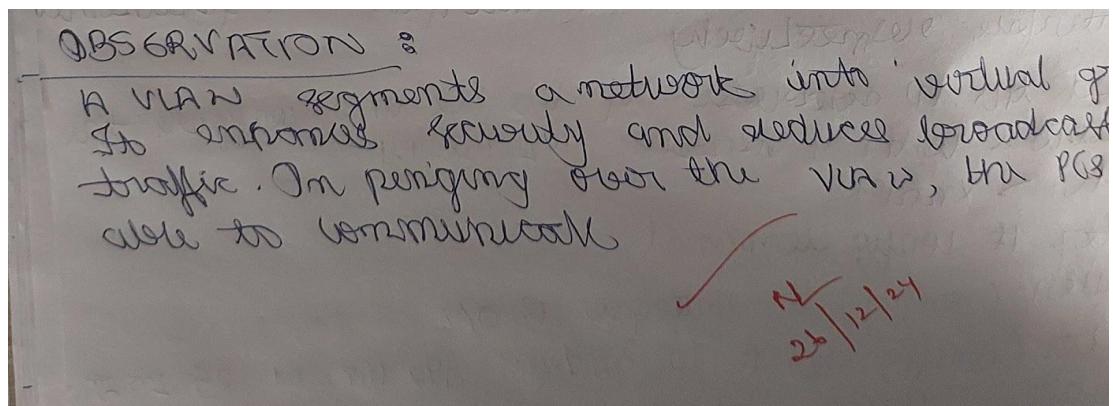
```
Router (VLAN) # exit
Router # config
Router (config) # interface fastethernet 9/1
Router (config-if) # encapsulation dot1q 2
Router (config-if) # ip address 192.168.2.1 255.255.255.0
Router (config-if) # no shutdown
Router (config-if) # exit
Router (config) # exit
Router (config) # exit
```

Screen shots/ output:





Observation:

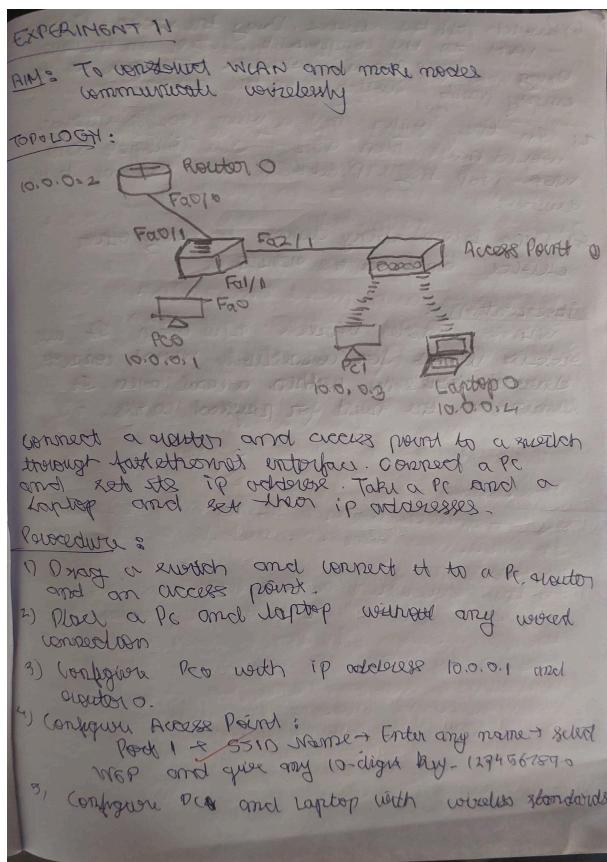


Program 12

Aim of the program:

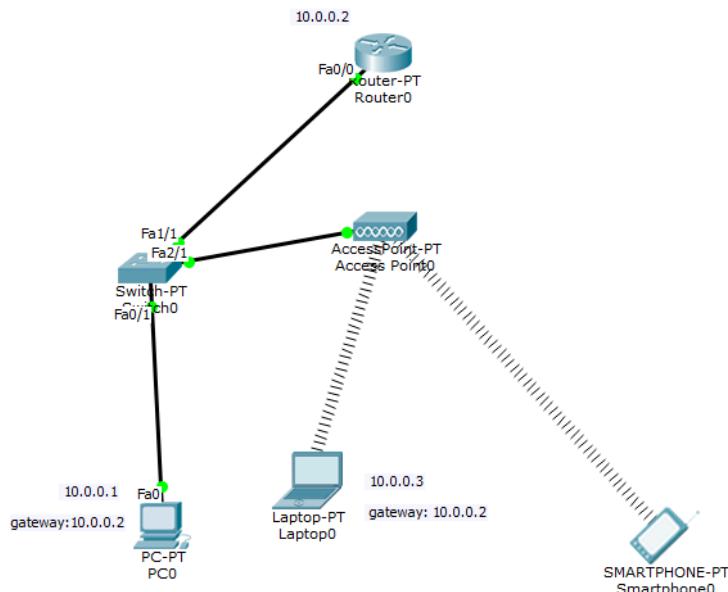
To construct a WLAN and make the nodes communicate wirelessly

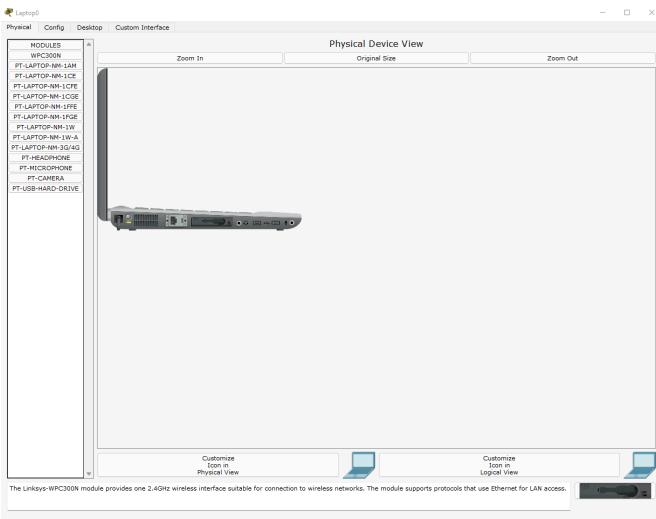
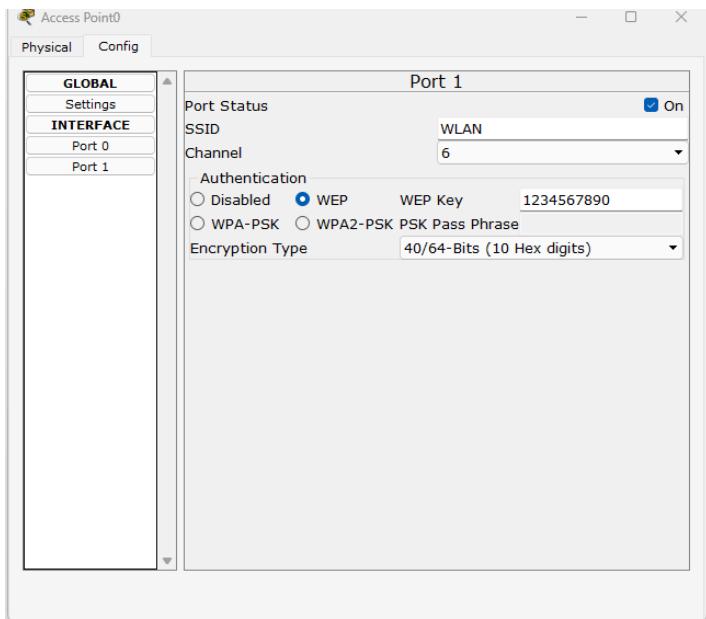
Procedure along with the topology:

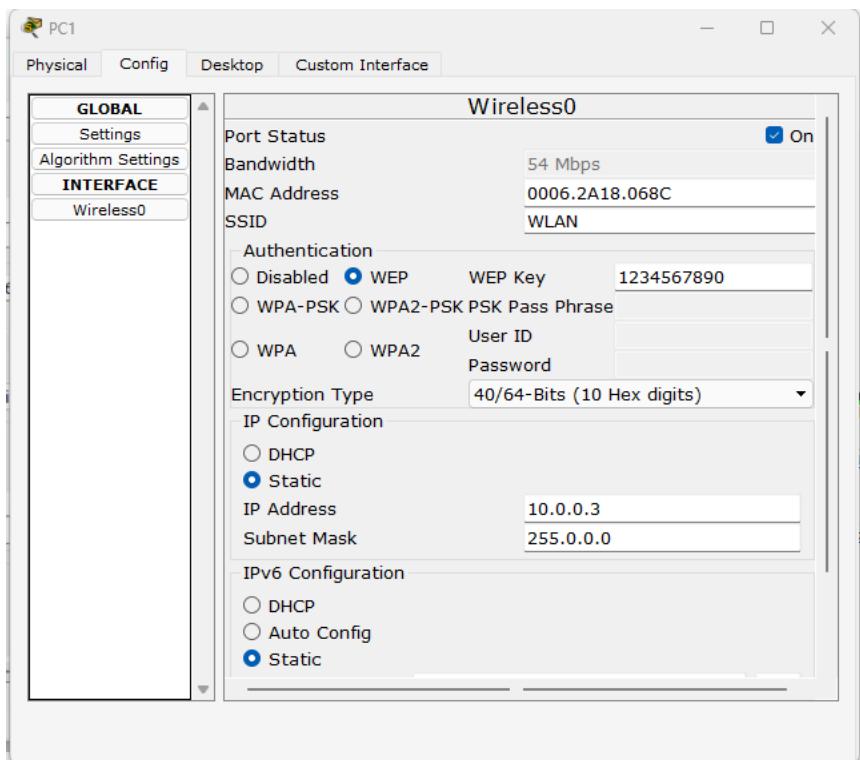


- 6) Switch off the device. Drag the existing PT-HOST - VAP to the components listed in the UI. Drag WMP 300 N wireless interface to the empty port switch on the device.
- 7) In this empty box, a new wireless interface would have been added. Now configure SSID, WEP, WEP key, IP address and gateway to the device.
- 8) Ping from every device to every other device and see the results

Screen shots/ output :







The screenshot shows the Command Prompt window on PC0. The user has run ping commands to test connectivity between PC1 and PC0.

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=20ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=6ms TTL=128
Reply from 10.0.0.3: bytes=32 time=8ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 6ms, Maximum = 20ms, Average = 10ms

PC>ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4: bytes=32 time=18ms TTL=128
Reply from 10.0.0.4: bytes=32 time=10ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=11ms TTL=128

Ping statistics for 10.0.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 7ms, Maximum = 18ms, Average = 11ms
```

Observation:

Observation :

WLAN enables wireless n/w b/wmm. It uses radio waves for connectivity. WLAN connects devices wirelessly within a local area. It eliminates the need for physical cables.

21/12/29

CYCLE-2

Program 1

Aim of the program:

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

Observation Book:

The handwritten notes describe a pseudocode implementation of a CRC-CCITT (16-bits) error detection code. It includes functions for polynomial division (mod2 div), polynomial multiplication (mult), and polynomial addition (xor). The pseudocode uses bit manipulation and loops to process data blocks. A worked example shows the calculation of a remainder for a given data and key, followed by the decoding of the received data to check for errors.

```

EXPERIMENT - 13
1) Write a program for error detecting code using
   CRC- CCITT (16-bits)

def xor(a, b):
    result = []
    for i in range(0, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)

def mod2_div(dividend, divisor):
    pick = len(divisor) - 1
    temp = dividend[0:pick]
    while pick < len(dividend):
        if temp[0] == '1':
            temp = xor('0' * pick, temp)
        else:
            temp = xor('0' * pick, temp)
    temp = xor('0' * pick, temp)
    dividend = temp + dividend[pick:]
    return temp

def encode(data, key):
    l = len(key)
    appended_data = data + '0' * (l - key - 1)
    remainder = mod2_div(appended_data, key)
    return appended_data + remainder

```

Output:

remainder = 11
 encoded_data (data + remainder) = 100100100010010011
 remainder after decoding = 000
 no error detected in received data

Code:

```

#include<stdio.h>
#include<string.h>
#define N strlen(gen_poly)

char data[28], check_value[28], gen_poly[10];
int data_length,i,j;

void XOR(){
    for(j = 1;j < N; j++)
        check_value[j] = ((check_value[j] == gen_poly[j])?'0':'1');
}

void receiver(){

    printf("Enter the received data: ");
    scanf("%s", data);
}

```

```

printf("\n-----\n");
printf("Data received: %s", data);

crc();

for(i=0;(i<N-1) && (check_value[i]!='1');i++)
{
    if(i<N-1)
        printf("\nError detected\n\n");
    else
        printf("\nNo error detected\n\n");
}

void crc(){
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
        if(check_value[0]=='1')
            XOR();
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
}

int main()
{
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\nEnter the Generating polynomial: ");
    scanf("%s",gen_poly);
    data_length=strlen(data);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n-----");
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n-----");
    crc();
    printf("\nCRC or Check value is : %s",check_value);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n-----");
    printf("\n Final data to be sent : %s",data);
    printf("\n-----\n");
    receiver();
    return 0;
}

```

OUTPUT:

```

Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011
-----
Data padded with n-1 zeros: 1001100000000000
CRC or Check value is: 0100010

```

Program 2

Aim of the program:

Write a program for congestion control using Leaky bucket algorithm.

Observation Book:

EXPERIMENT - 14
Leaky Bucket Program : Write a program for congestion control using leaky bucket algorithm.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NDF_PACKETS 5
int main (void)
{
    int sum = (random() % 10) / a
    return sum == 0 ? 1 : m
}

#include <stdlib.h>
long rand (void);
int main ()
{
    int packet_sz[NDF_PACKETS], i, clk, b_size, op, p_sz, p_time;
    p_sz = p_time = op = 0;
    for (i=0; i<NDF_PACKETS; i++)
        packet_sz[i] = random() % 10;
    for (i=0; i<NDF_PACKETS; i++)
        printf ("Enter the output rate:");
    scanf ("%d", &rate);
    printf ("Enter the Bucket Size:");
    scanf ("%d", &b_size);
    for (i=0; i<NDF_PACKETS; i++)
    {
        if ((packet_sz[i] + p_sz - sum) > b_size)
            if (packet_sz[i] > b_size)
                printf ("In Incoming packet size (%d bytes)\n"
                       "is greater than bucket capacity\n"
                       "(%d bytes)- PACKET REJECT", packet_sz[i], b_size);
            else
                printf ("In Bucket capacity exceeded -PACKETS\n"
                       "REJECT");
        else
            p_sz -= sum + packet_sz[i];
        printf ("In Incoming Packet size (%d", packet_sz[i]
    }
}

```

printf ("Initial scheduling to Transmitt: Y-%d, R-%d, m-%d", p_time, op, sum);
 p_time = random() * p;
 printf ("Time left for transmission: %d units,\n"
 "P-time);
 for (clk=0; clk < p_time; clk += 10)
 while (p_sz - sum > 0)
 {
 sleep (1);
 if (p_sz == sum)
 {
 if (p_sz < 0.5 * sum)
 op = p_sz - sum, p_sz -= sum;
 else
 op = 0, p_sz -= sum;
 printf ("In Packets of size %d Transmitter: %d", op);
 printf ("--Bytes Remaining To transmit: %d", sum);
 }
 else
 printf ("In No packets to transmit (%d)", p_sz);
 }
}

Code:

```
# initial packets in the bucket
storage = 0

# total no. of times bucket content is checked
no_of_queries = 4

# total no. of packets that can
# be accommodated in the bucket
bucket_size = 10

# no. of packets that enters the bucket at a time
input_pkt_size = 4

# no. of packets that exits the bucket at a time
output_pkt_size = 1
for i in range(0, no_of_queries): # space left

    size_left = bucket_size - storage
    if input_pkt_size <= size_left:
        # update storage
        storage += input_pkt_size
    else:
        print("Packet loss = ", input_pkt_size)

    print(f'Buffer size= {storage} out of bucket size = {bucket_size}')

# as packets are sent out into the network, the size of the storage decreases
storage -= output_pkt_size
```

OUTPUT:

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

Program 3

Aim of the program:

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

Observation Book:

EXPERIMENT-15

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

Programme ClientTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Input file name:")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("From Server:", filecontents)
clientSocket.close()
```

Server TCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while True:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("Sent contents of " + sentence)
    file.close()
    connectionSocket.close()
```

Code:

```
ClientTCP.py
from socket import *
```

```
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\nEnter file name: ")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("\nFrom Server:\n")
print(filecontents)
clientSocket.close()
```

```
ServerTCP.py
from socket import *
```

```
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
```

```
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")

    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()
```

OUTPUT:

The screenshot shows a code editor interface with the title bar "CN_LAB". The left sidebar displays a file tree with "CN LAB" expanded, showing files "ClientTCP.py", "ServerTCP.py", "VLAN.pkt", and "WLAN.pkt". The main editor area contains the "ServerTCP.py" script:

```
from socket import *
serverName= '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    print ("\nSent contents of" + sentence)
    file.close()
    connectionSocket.close()
```

The bottom terminal window shows the output of the Python script:

```
PS C:\Users\dell\Desktop\CN_LAB & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ServerTCP.py
The server is ready to receive
Sent contents ofServerTCP.py
The server is ready to receive
```

The screenshot shows a code editor interface with the title bar "CN_LAB". The left sidebar displays a file tree with "CN LAB" expanded, showing files "ClientTCP.py", "ServerTCP.png", "ServerTCP.py", "VLAN.pkt", and "WLAN.pkt". The main editor area contains the "ClientTCP.py" script:

```
from socket import *
serverName= '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    print ("\nSent contents of" + sentence)
    file.close()
    connectionSocket.close()
```

The bottom terminal window shows the output of the Python script:

```
PS C:\Users\dell\Desktop\CN_LAB & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ClientTCP.py
Enter file name: ServerTCP.py
From Server:
from socket import *
serverName='127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
```

Program 4

Aim of the program:

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

Observation Book:

EXPERIMENT-16

Using UDP sockets with 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.

clientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name: ")
clientSocket.sendto(sentence.encode("utf-8"), (serverName, serverPort))
filecontents, serverAddress = clientSocket.recvfrom(2048)
print("In reply from server: \n")
print(filecontents.decode("utf-8"))
for i in filecontents:
    print(str(i), end=" ")
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    fileContent = file.read(2048)
    serverSocket.sendto(fileContent.encode("utf-8"), clientAddress)
    print("In sent content of, end=1")
    print(sentence)
    file.close()
```

Code:

```
ClientUDP.py
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))
filecontents, serverAddress = clientSocket.recvfrom(2048)

print("\nReply from Server:\n")
print(filecontents.decode("utf-8"))
clientSocket.close()

ServerUDP.py
from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

print("The server is ready to receive")

while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")

    file = open(sentence, "r")
    con = file.read(2048)
    serverSocket.sendto(bytes(con, "utf-8"), clientAddress)

    print("\nSent contents of ", end=' ')
    print(sentence)

    file.close()
```

OUTPUT:

The image shows two terminal windows side-by-side, both titled "CN_LAB".

Terminal 1 (Left): This window contains the code for ClientUDP.py. It starts with a print statement: "The server is ready to receive". Then it enters a loop where it receives data from port 2048, decodes it to utf-8, and writes it to a file named "sentence". Finally, it prints the contents of the file.

```
from socket import *
serverPort = 12800
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ('\nSent contents of ', end ='')
    print (sentence)
    # for i in sentence:
    #     # print (str(i), end = " ")
    file.close()
```

Terminal 2 (Right): This window contains the code for ServerUDP.py. It starts with a print statement: "The server is ready to receive". Then it enters a loop where it receives data from port 2048, decodes it to utf-8, and prints the contents. It also prints a message indicating that the server is ready to receive.

```
from socket import *
serverPort = 12800
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ('\nSent contents of ', end ='')
    print (sentence)
    # for i in sentence:
    #     # print (str(i), end = " ")
    file.close()
```

Took Exploration - Wireshark

EXPERIMENT-17 - WIRESHARK

Wireshark is a powerful and widely used network protocol analyzer. It allows you to capture and inspect data packets travelling over a network in real-time making it a crucial tool for studying computer networks, troubleshooting network issues and understanding protocols.

Key features :

- 1) **Packet capture :** Captures live network traffic from various interfaces (ex: ethernet, wi-fi)
- 2) **Protocol Analyzers :** Support hundreds of protocols (ex: TCP, UDP, HTTP, FTP)
- 3) **Filtering :** Offers powerful filters to isolate specific packets or traffic types.
- 4) **Visualization :** Displays packet details with hierarchical layers (ethernet, IP, TCP/UDP)

Use cases of Wireshark

- 1) **Network Troubleshooting :**
 - * Diagnosing slow network speeds
 - * Identifying bottlenecks and misconfigurations
- 2) **Security analysis :**
 - * Detecting malicious traffic or intrusions
- 3) **Protocol Study :**
 - * Understanding packet structures and flow.

Common filters :

- * http : show only http traffic
- * udp : show only UDP traffic
- * tcp port = 80 : show traffic on TCP port 80