VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Madhu Sarika (1BM22CS140)

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 **Madhu Sarika (1BM22CS140),** 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.

Srushti C S Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index-Cycle-I

Github Link:

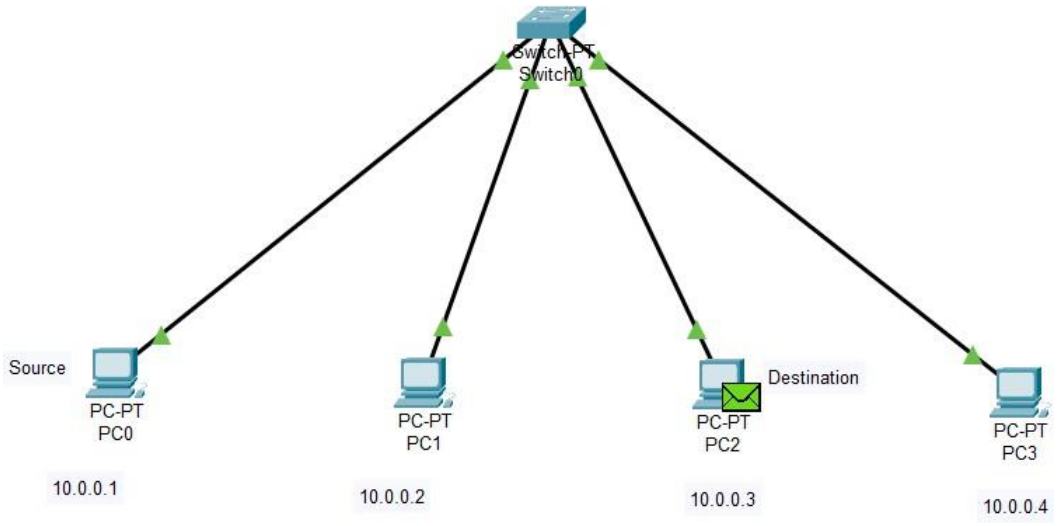
https://github.com/madhupandeyy/CN-Lab

Index-Cycle-II

EXPERIMENT - 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.

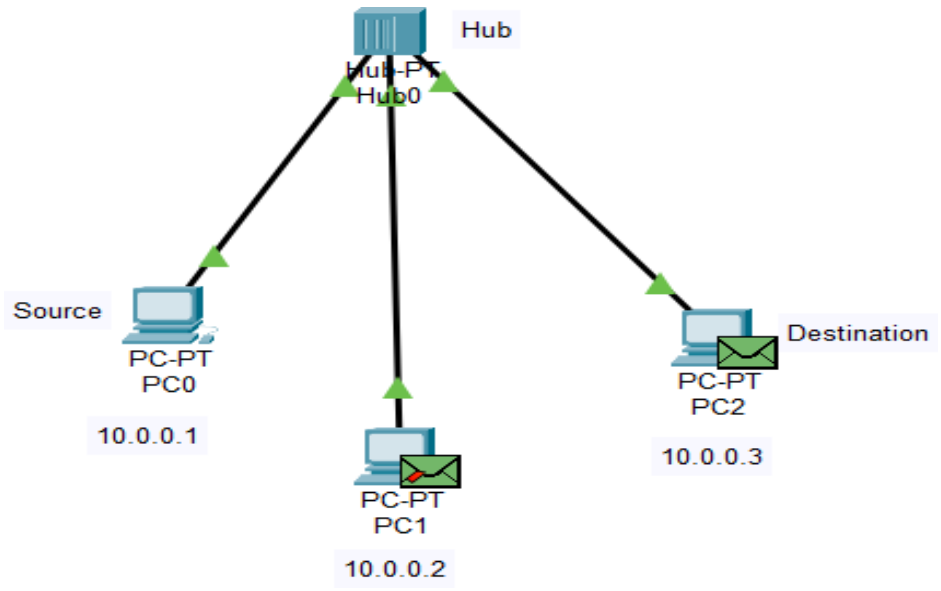
Switch:



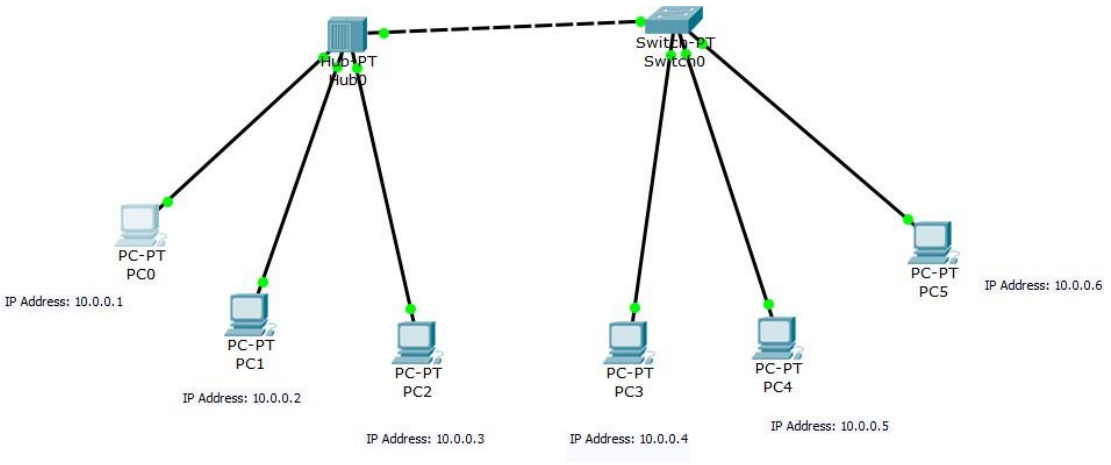
Connection: Copper Straight Through Wire

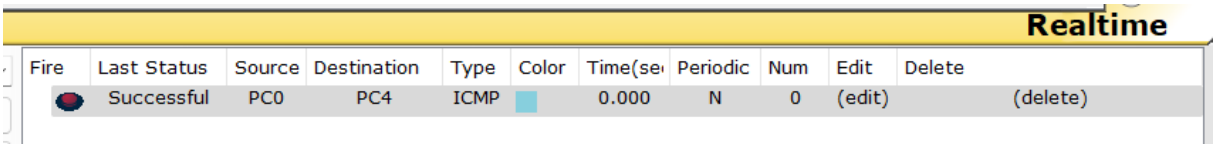
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	(delete)

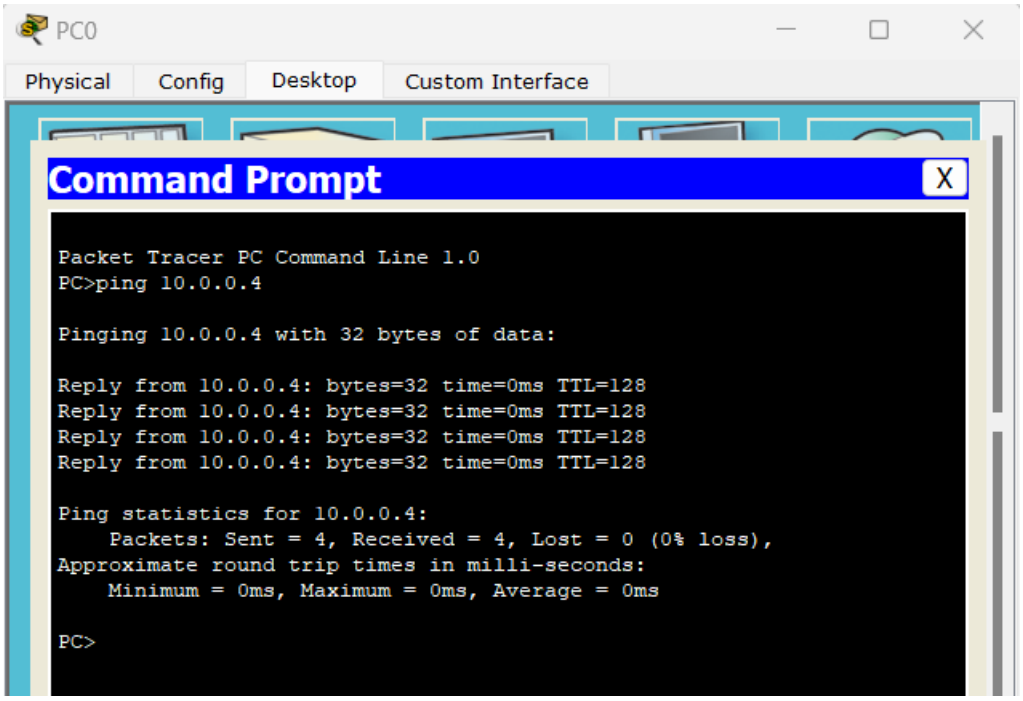
Hub:



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







No gateway was given in this, only configuration was done using IP address by click on end devices and then config and then fast ethernet and there typing IP address 10.0.0.1 and so on and then for subnet mask, just click on it.

Aim: To demonstrate the transmission of a simple PDU between 4 devices connected using a hub and a switch.

1/10/24

Topology: Star Topology

IPV4 addresses → 4 parts a.b.c.d

10.0.0.1
10.0.0.2

Hub - physical layer = no IP address

Simple PDU source → Dest (Hub)

IP address: 10.0.0.1 10.0.0.2 10.0.0.3

Connection: Copper Straight Through

~~Configuration~~:

(1) First click Realtime Mode (Shift+R) from the bottom right side. (left-most)

(2) Then go to end devices from left bottom, Select generic (three) and hub (generic).

(3) Select connection (copper straight through) cable between PC-(Generic) and Hub (generic).

Configuration:

Click on the PC and go to config and then in FastEthernet(), static, write the IP address of the form (a.b.c.d) and then click on the blank area of Subnet Mask. This will automatically generate subnet mask. Then close this.

→ Go to simple PDU on the right side, and click on the source PC first and then click on the destination PC.

→ Go to simulation mode on the right bottom side and click on the Auto capture Play to start simulation.

Observation:

(1) Hub is not an intelligent device, so it passes whatever comes to it. It is not configured.

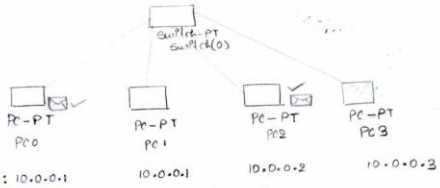
(2) When message is selected to be sent from source PC to destination PC, ~~then~~ it is also shared to other PCs connected to the hub. In return, the destination ~~if receives~~ ~~acknowledge~~ acknowledge by sending signal back to the source PC through hub.

Here PC1 which is not supposed to get the message will also receive it, which should not happen. So, we can see cross mark on the message received by PC1.

(3) At last, status of the connection is seen in event list panel at bottom left. If seen successful, means connection has been established properly.

By using switch:

Topology: Star-Topology



IP address: 10.0.0.1 10.0.0.1 10.0.0.2 10.0.0.3

Connection: Copper Straight through

Procedure:

- From the bottom left, we will bring generic PC (end devices) and a switch (generic).
- Through copper straight wire will establish connection between switch and end devices (PC) (generic).
- Go to simple PDU, and click on the source PC and then dest. PC.
- To see the simulation, click on the simulation mode on the bottom right and click Auto Capture/Play.

Configuration:

- click on the PCs (generic) and go to config and then go to FastEthernet() in interface and then in switch type the IP address and then click on the subnet mask which will be automatically generated.

Observation:

- The message is transferred from the source PC to destination PC ~~so~~ which can been in simulation mode.

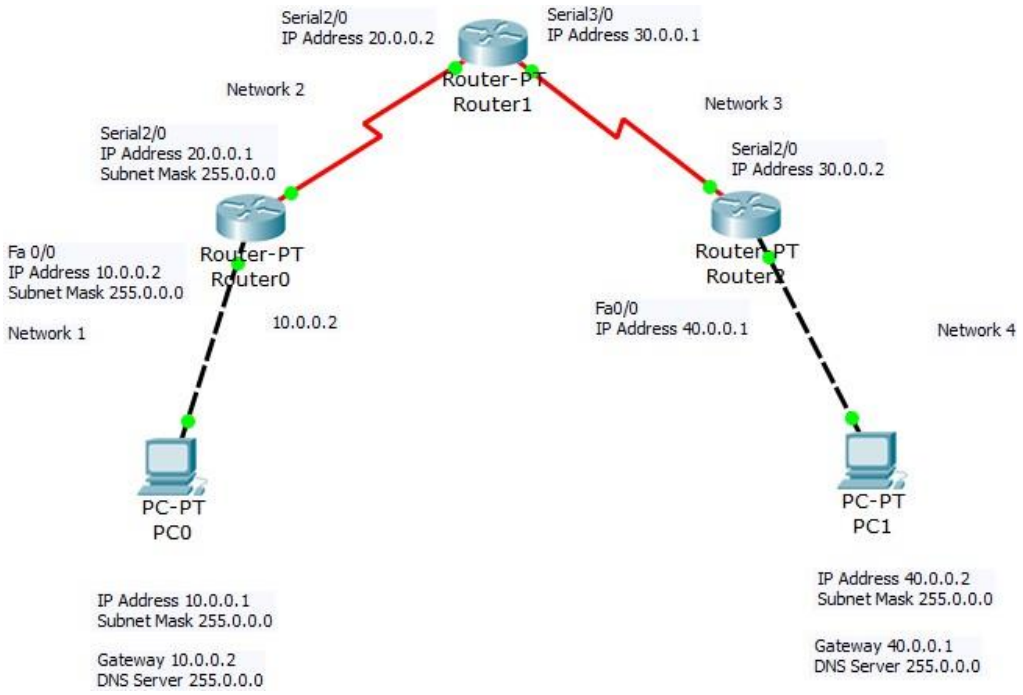
- The destination PC sends the ~~acknowledgment~~ signals back to the source PC to acknowledge that it has received.
- Due to STP-IMP protocol, a switch tries to find shortest path and keeps in the memory for further transmission.

1/10/24

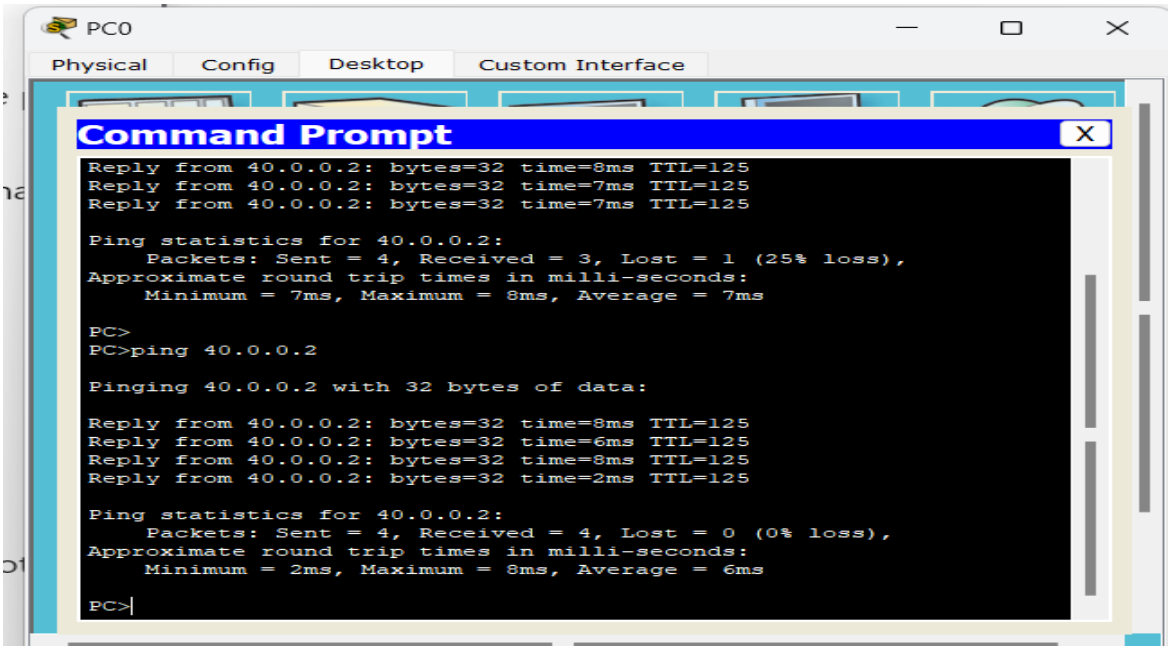
EXPERIMENT - 2

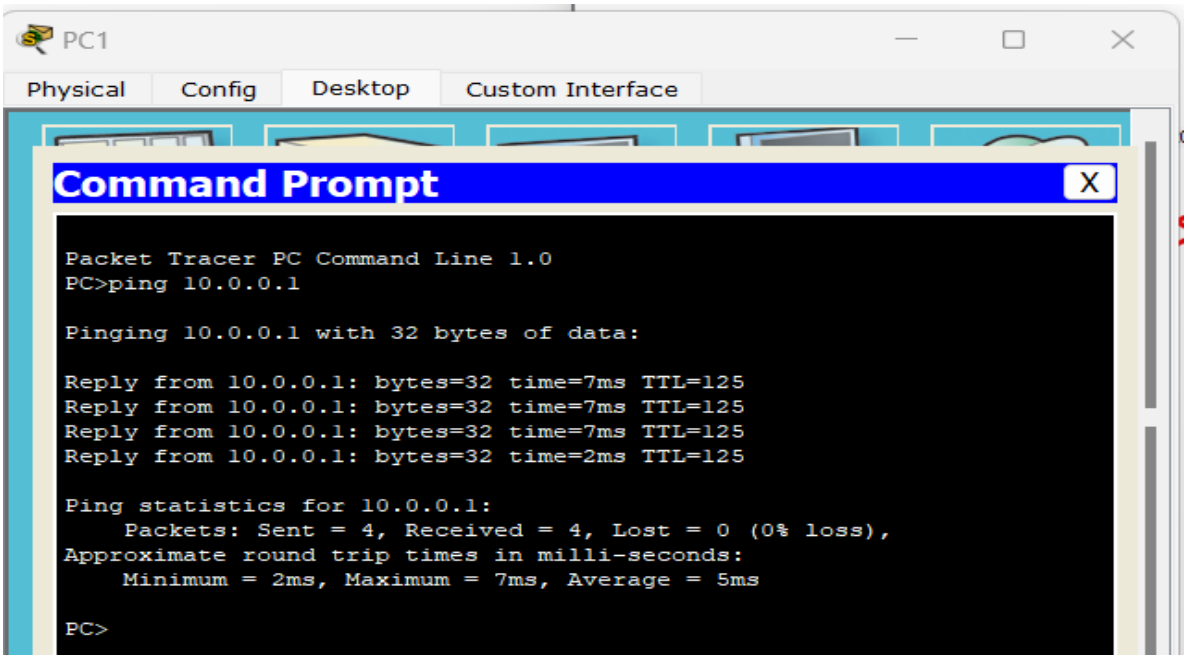
AIM: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

The **Default Gateway** for **PC0** should be the IP address of the router's interface that is directly connected to **PC0**, which in this case is **Router 0**.



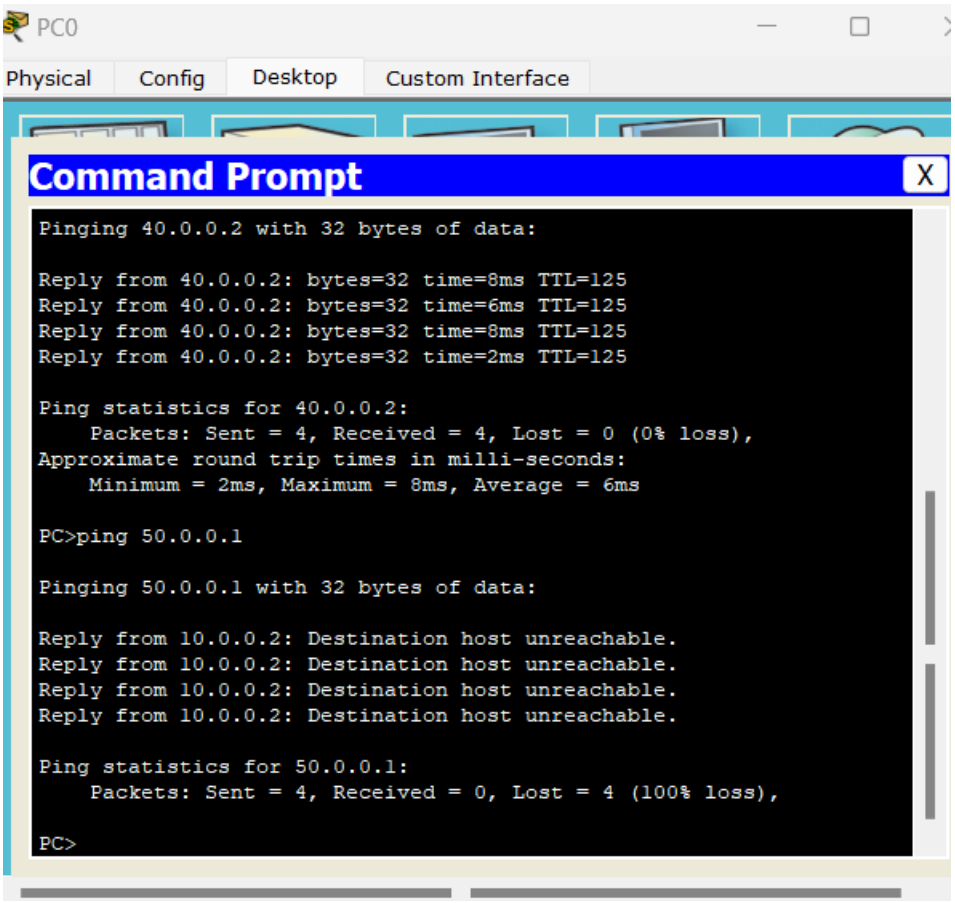
1. **Ping Response (Successful Ping)**:
   When you ping another device (like a router or PC), and the destination is reachable, you will get a **response**. This means the device is reachable and responding to your requests.





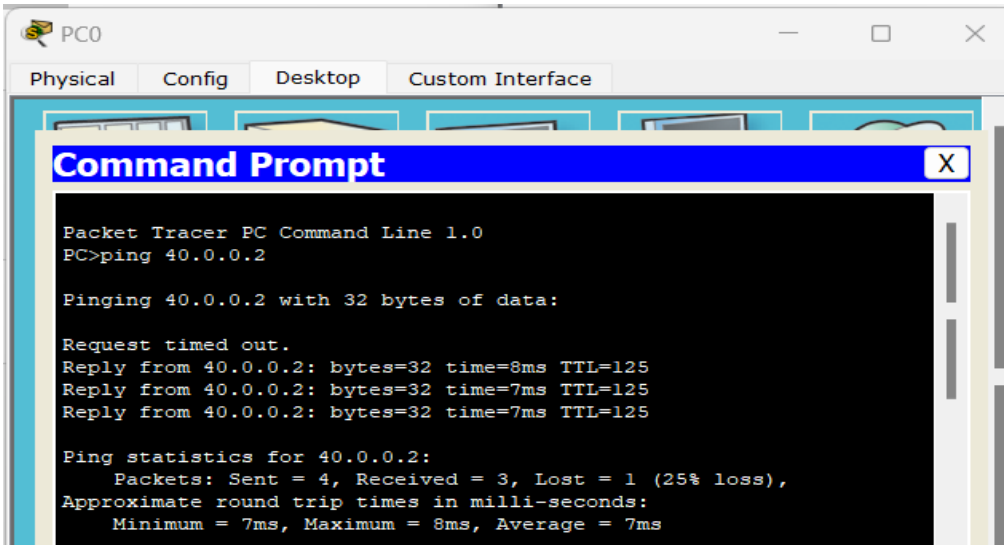
2. **Destination Unreachable**:

If you ping a device and it cannot be reached, you will get a **"Destination Unreachable"** message. This could happen due to a variety of reasons, such as incorrect IP addressing, routing issues, or firewall blocking.



3. **Request Timed Out**:

If the ping request is sent, but no response is received within the timeout period, you will get a **"Request Timed Out"** message. This usually happens due to network congestion, incorrect routing, or the destination device not responding in time.



AIM: To demonstrate configuration of IP addresses to the routers and explain ping command

Commandline commands for router

* exit

① enable ②(#)

③ config t (config)# → however we need dot

④ Interface name → Interface fastethernet 0/0 (config-if)

FaO/0 Fa1/0 (Interfaces)

⑤ ip address

10.0.0.2 (space) 255.0.0.0

~~20.0.0.3~~

Generic comp: end devices

10.0.0.1

20.0.0.2

⑥ no shutdown

⑦ exit from the first interface

⑧ interface fastethernet 1/0

⑨ ip address 20.0.0.3 255.0.0.0

⑩ no shutdown (both green)

Now, check for message passing b/w two end devices

dest ip address = 20.0.0.2

* ping 20.0.0.2 ⇒ request timed out

Also configure gateway for router.

(exit from the internet which gate i should go)

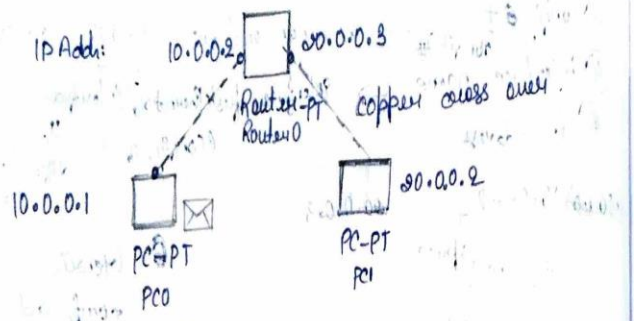
- default gateway is to be configured
- default gateway for pc1 ⇒ (10.0.0.2) gateway

pc2 ⇒ 20.0.0.3

- Now ping again

ping 20.0.0.2.

Topology:



Configuration steps:

(1) Go to router, CLI and then type commands.

(2) Give IP address to both end devices

(3) Give IP address configuration for the router and gateway for the ~~Router~~ end devices

(4) for gateway, PC0: 10.0.0.2

PC1: 20.0.0.3

(5) ~~Simple PDU is not working here~~, that's why we will give ping command in command prompt of PC(0) (sender).

(desktop)

ping 20.0.0.2 ← destination ip address } if we do this without

Observation:

Reasons of Ping : ping is just checks up or down.

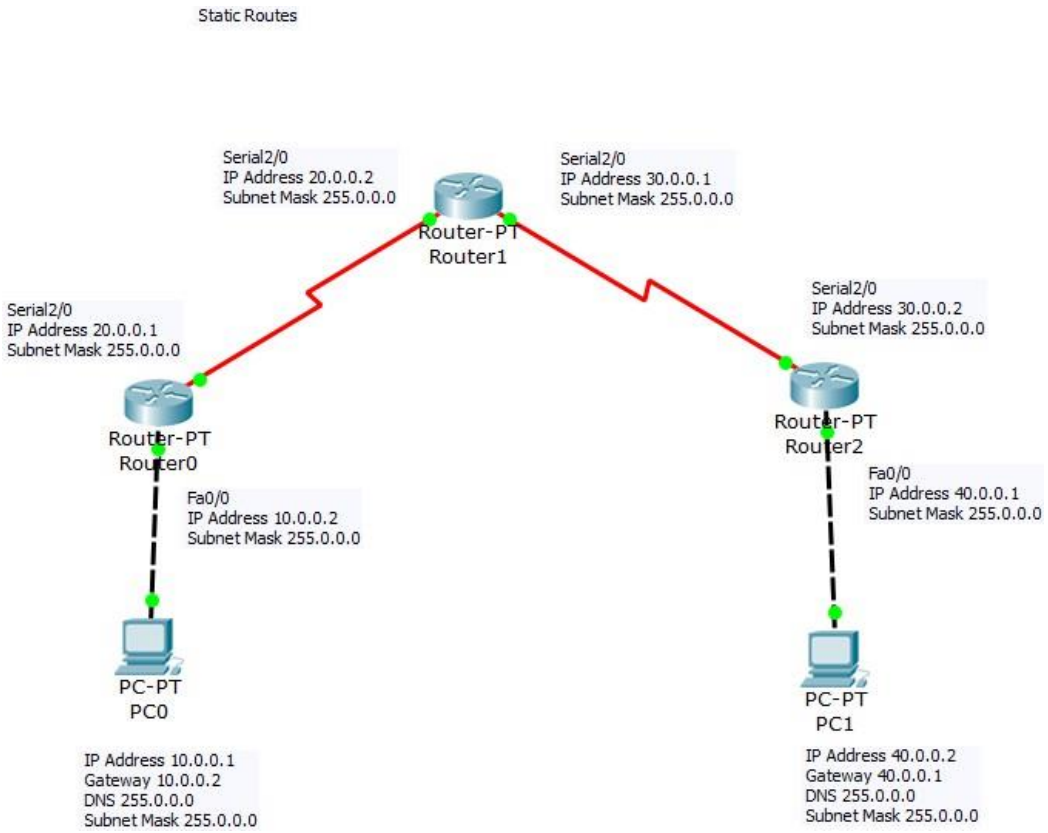
Message is shared in from source pc to destination PC.

EXPERIMENT - 3

AIM: Configure static route to the Router.

Default and Static Routes

- **Default Route:** A route used to forward packets to a destination not explicitly listed in the routing table. Represented as 0.0.0.0/0 in IPv4 or ::/0 in IPv6.
- **Static Route:** A manually configured route that defines a specific path to a destination network.



Static Route:

Router 0:

Equivalent IOS Commands

```
Router(config-if)#exit
Router(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router(config)#ip route 40.0.0.0 255.0.0.0 20.0.0.2
```

Router 1:

Only Static Route:

Equivalent IOS Commands

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

Router(config-if)#exit
Router(config)#ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router(config)#
```

Router 2:

Equivalent IOS Commands

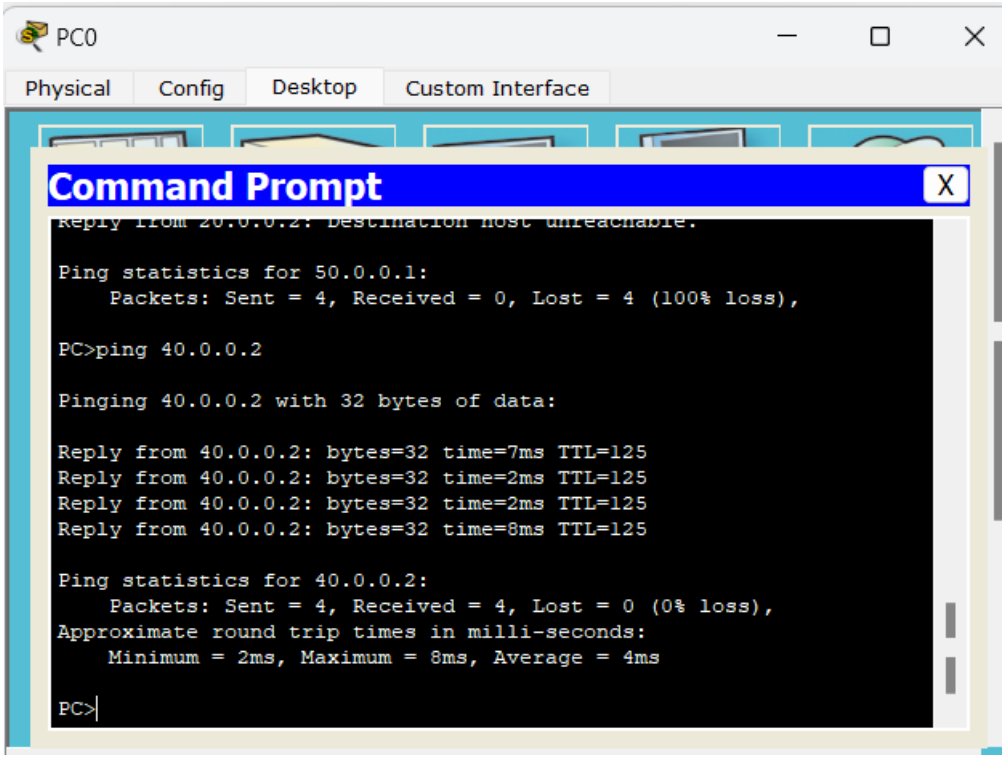
```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed
state to up

Router(config-if)#exit
Router(config)#ip route 20.0.0.0 255.0.0.0 30.0.0.1
Router(config)#ip route 10.0.0.0 255.0.0.0 30.0.0.1
```

Verify Connectivity

Use the following commands to test and verify connectivity:

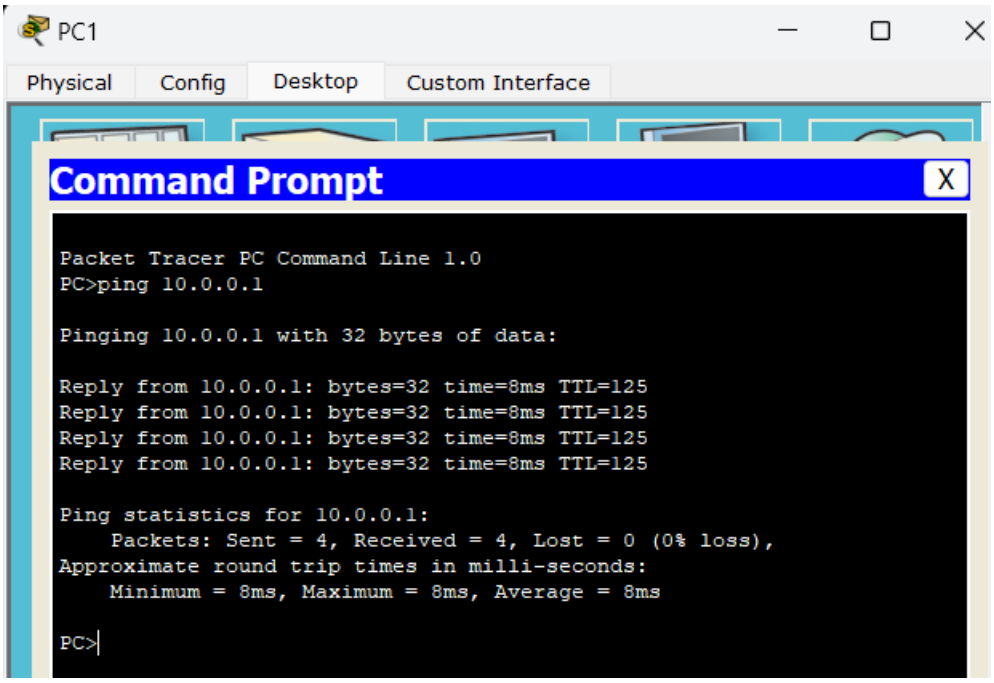
- **From PC0 to PC1:** Open PC0's Command Prompt:

ping 40.0.0.2

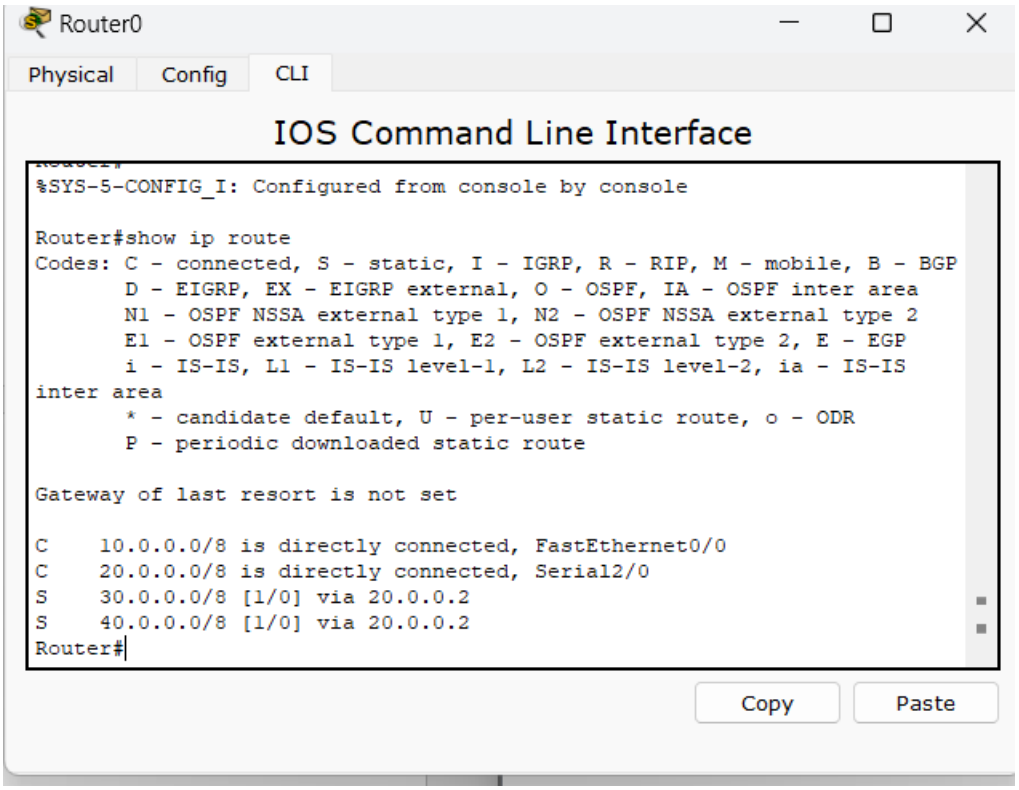


- **From PC1 to PC0:** Open PC1's Command Prompt:

ping 10.0.0.1



show ip route



Router1

Physical | Config | CLI

IOS Command Line Interface

```
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
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

S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
Router#
```

Copy | Paste

Router2

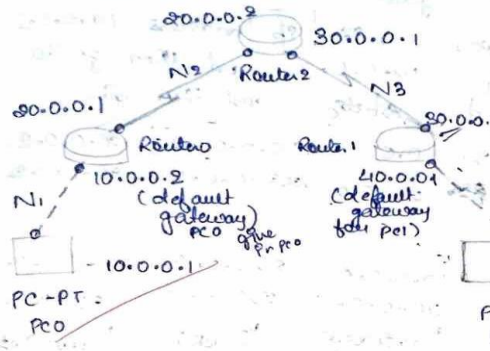
Physical | Config | CLI

IOS Command Line Interface

```
Router#
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#no ip route 0.0.0.0 0.0.0.0 30.0.0.1
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

S    10.0.0.0/8 [1/0] via 30.0.0.1
S    20.0.0.0/8 [1/0] via 30.0.0.1
C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
Router#
```

Copy | Paste

Experiment 3: To demonstrate the configuration of default routes to the routers



* Serial cable b/w two routers IP address
PC0 - 10.0.0.1
PC1 - 10.0.0.2
* 4 networks (atleast 2)

=# show ip route * Serial 2/0

#ip route
destination n/w ip
destination mask
Next hop
} for reaching to unknown networks

for router 0
#ip route 30.0.0.0 255.0.0.0 20.0.0.2
#ip route 40.0.0.0 255.0.0.0 20.0.0.2

Configuration steps: —

(1) Take two generic pc and three routers.
(2) Configure PC0 with ip address 10.0.0.1 and gateway 10.0.0.2 and then go to routers and config and then give
ip route 30.0.0.0 255.0.0.0 20.0.0.2
ip route 40.0.0.0 255.0.0.0 20.0.0.2

for router 1:
ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2
(network (mask) next hop)

for router 2:
ip route 20.0.0.0 255.0.0.0 30.0.0.1
ip route 10.0.0.0 255.0.0.0 30.0.0.1

(3) for PC2 set ip address as 40.0.0.2 and gateway as 40.0.0.1 and also DNS server 255.0.0.0

3) Go to command prompt of PC0 and type
ping 40.0.0.2

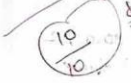
observations:
- Message was delivered from PC0 to PC1.
- Using the ping command, to reach devices connected to the network.

Output:

ping 40.0.0.2

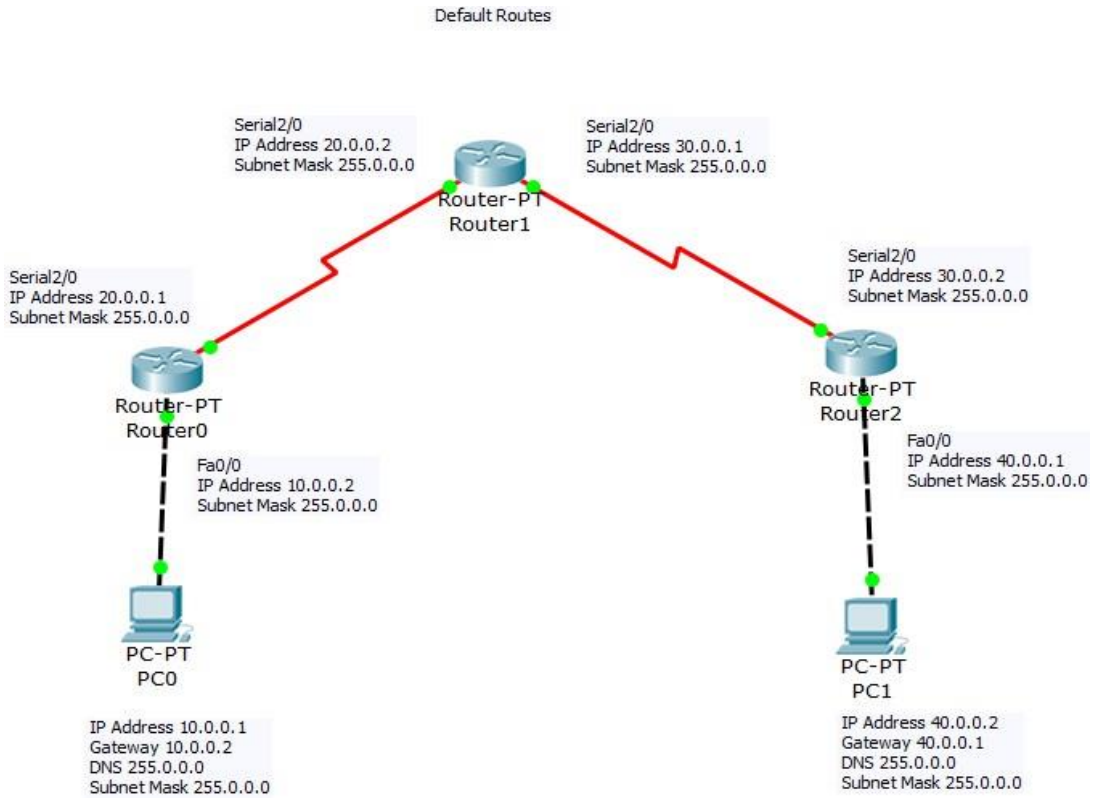
Pinging 40.0.0.2 with 32 bytes of data

Reply from 40.0.0.2 bytes = 32 time = 9ms TTL = 125
Reply from 40.0.0.2 bytes = 32 time = 8ms TTL = 125
Reply from 40.0.0.2 bytes = 32 time = 9ms TTL = 125
Reply from 40.0.0.2 bytes = 32 time = 7ms TTL = 125



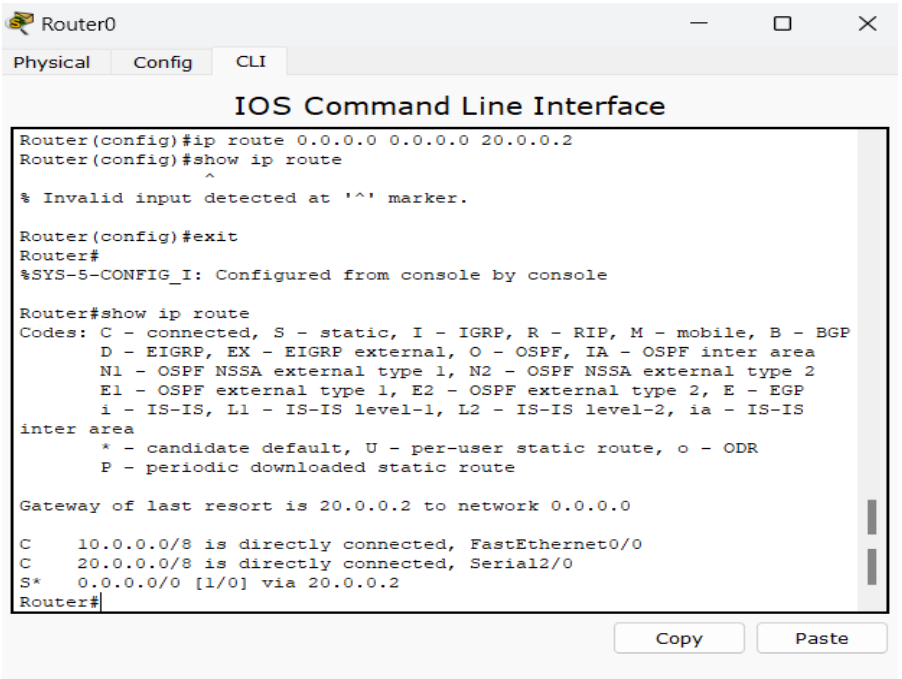
EXPERIMENT – 4 A

AIM: Configure default route to the Router.

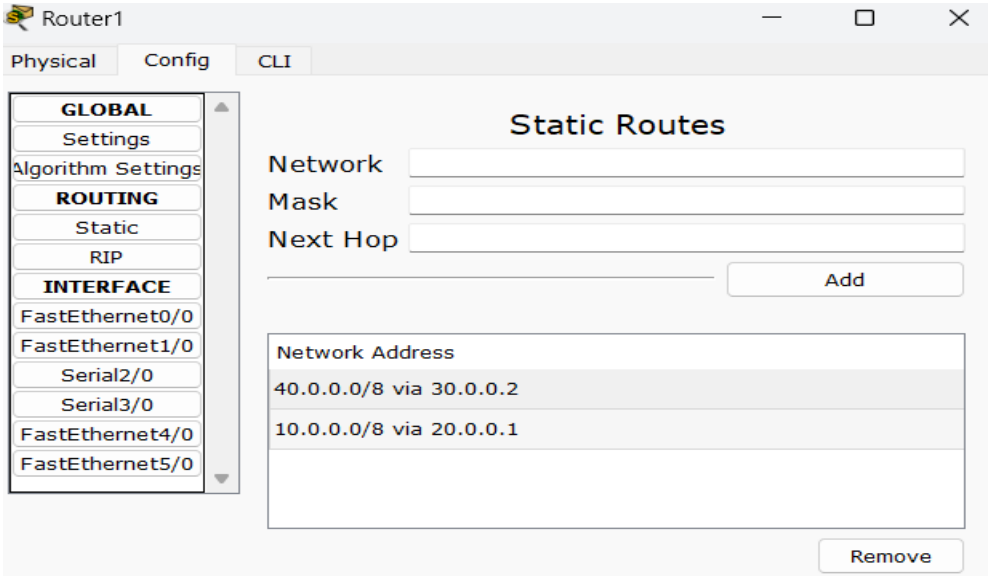


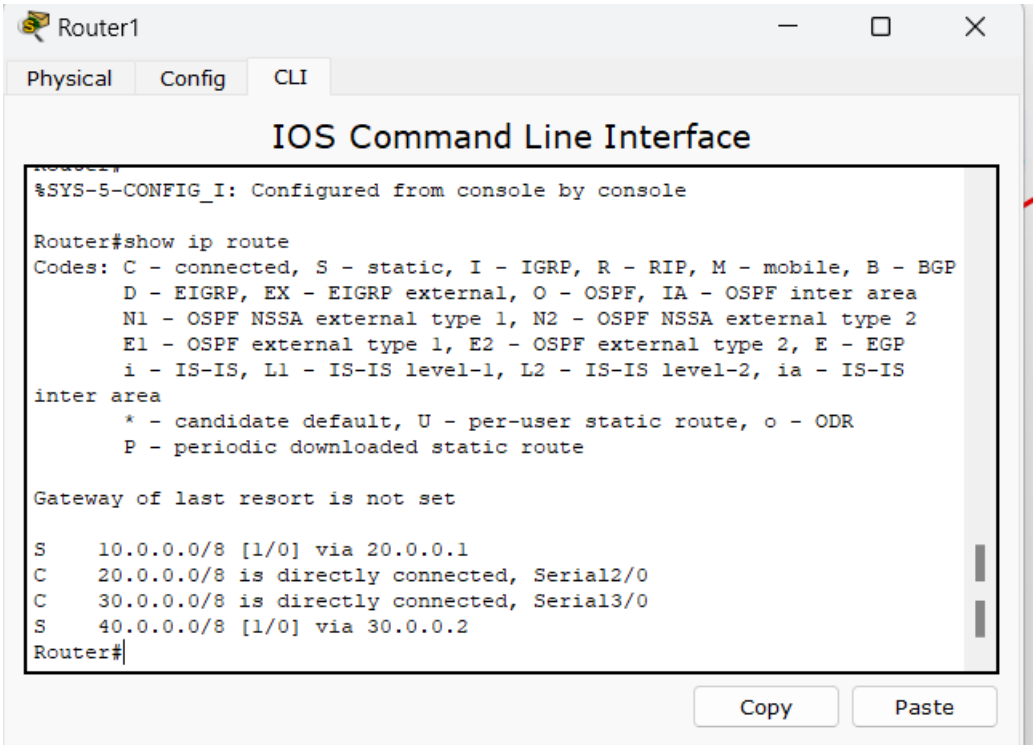
show ip route

Router 0:

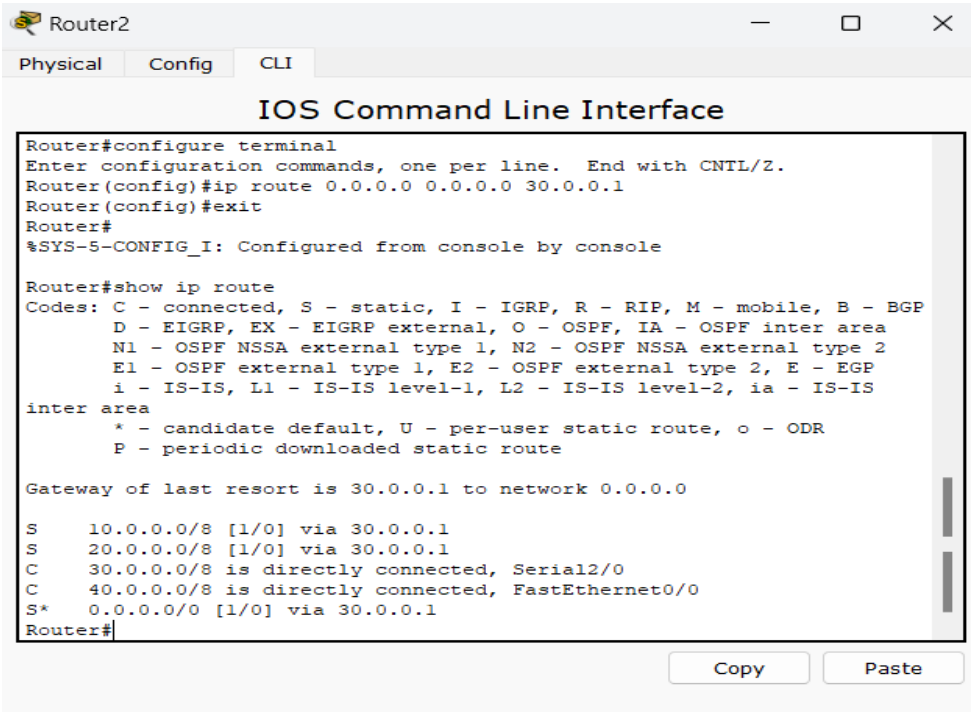


Router 1:


Router1
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Static Routes
Network
Mask
Next Hop
Add
Network Address
40.0.0.0/8 via 30.0.0.2
10.0.0.0/8 via 20.0.0.1
Remove


Router1
Physical
Config
CLI
IOS Command Line Interface
%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
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.0/8 is directly connected, Serial3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2
Router#
Copy
Paste

Router 2:

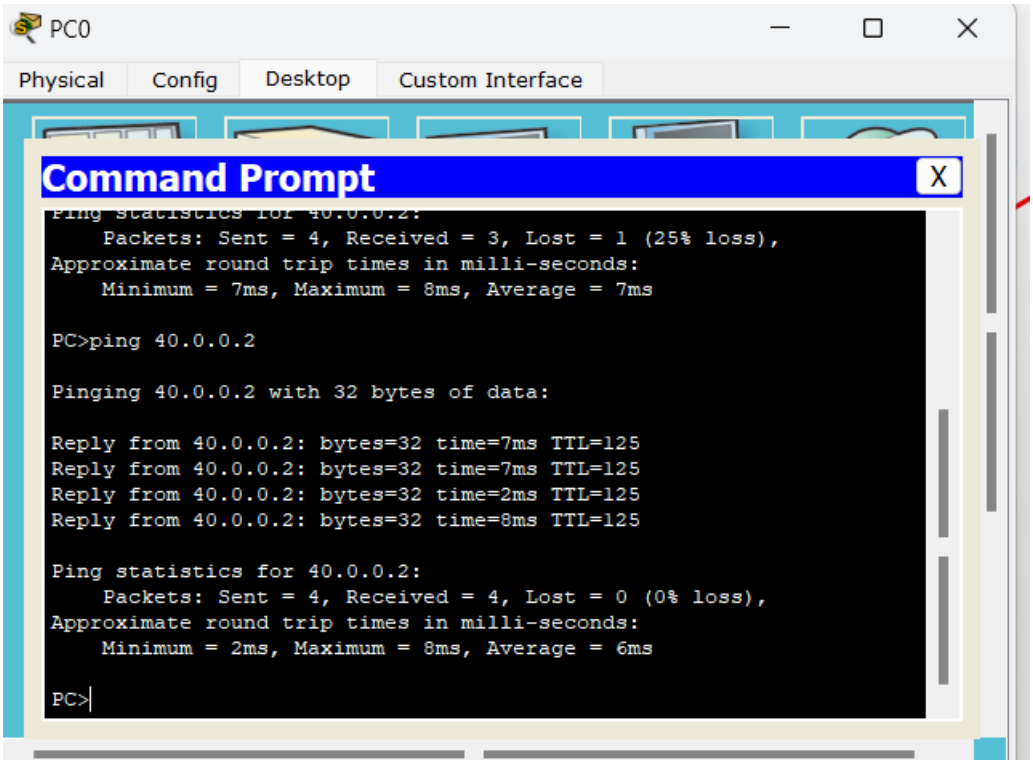


Verify Connectivity

Use the following commands to test and verify connectivity:

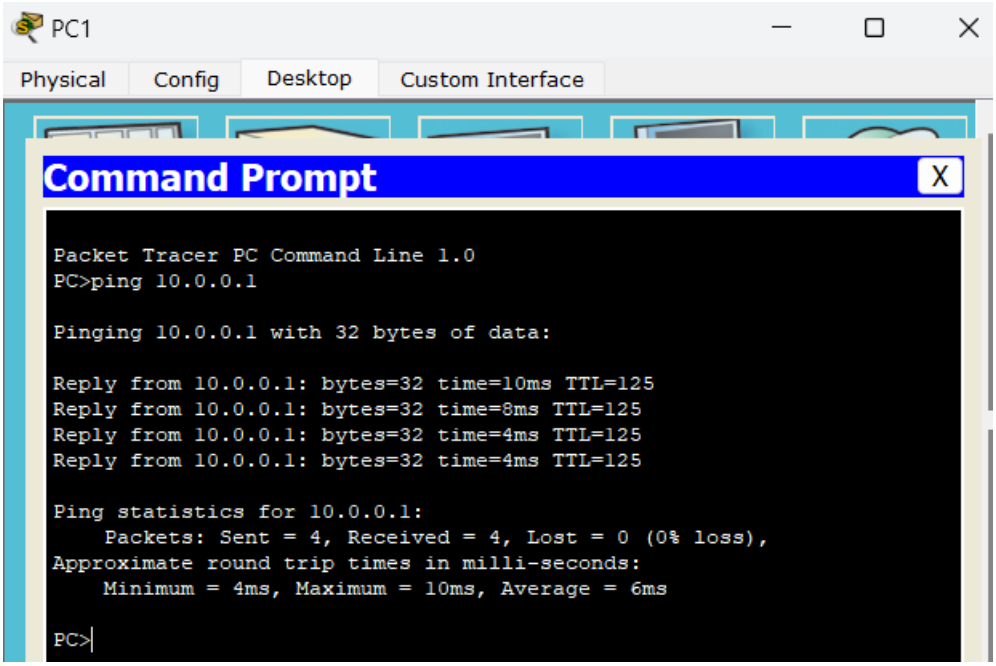
- **From PC0 to PC1:** Open PC0's Command Prompt:

ping 40.0.0.2



- **From PC1 to PC0:** Open PC1's Command Prompt:

ping 10.0.0.1


PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
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=10ms TTL=125
Reply from 10.0.0.1: bytes=32 time=8ms TTL=125
Reply from 10.0.0.1: bytes=32 time=4ms TTL=125
Reply from 10.0.0.1: bytes=32 time=4ms 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 = 4ms, Maximum = 10ms, Average = 6ms
PC>

Exp-4

AIM – To config default and static routers through a connection of routers

After configuration, follow these:

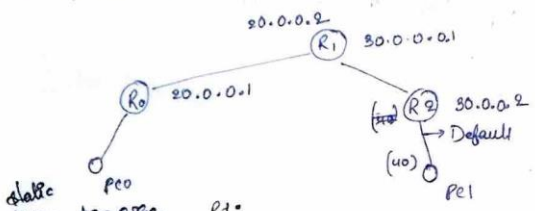
R0

config. t

R0 config # ip route 0.0.0.0 0.0.0.0 20.0.0.2, default

for static R1 is default route for R0

R2 # ip route 0.0.0.0 0.0.0.0 30.0.0.1



static

R1:

ip route 40.0.0.0 255.0.0.0 30.0.0.2

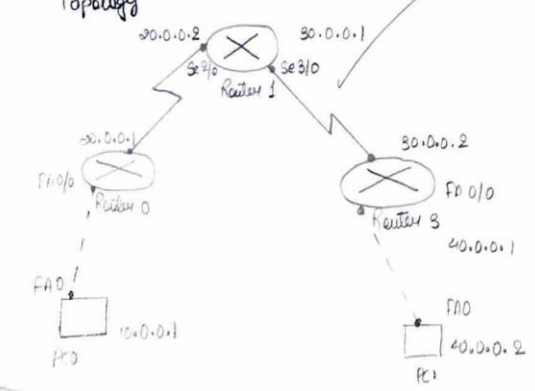
source network

dest hop

ip route 180.0.0.0 255.0.0.0 20.0.0.1

Configuration:

Topology



- Configure the routers, pcs using the above commands.
- Once the connection is set up, send the message.
- Go to the command prompt of PC0 and type ping 40.0.0.2

Observation

- By setting up with proper ip address and gateway and configuring the router, we can see the connection is proper

Output:

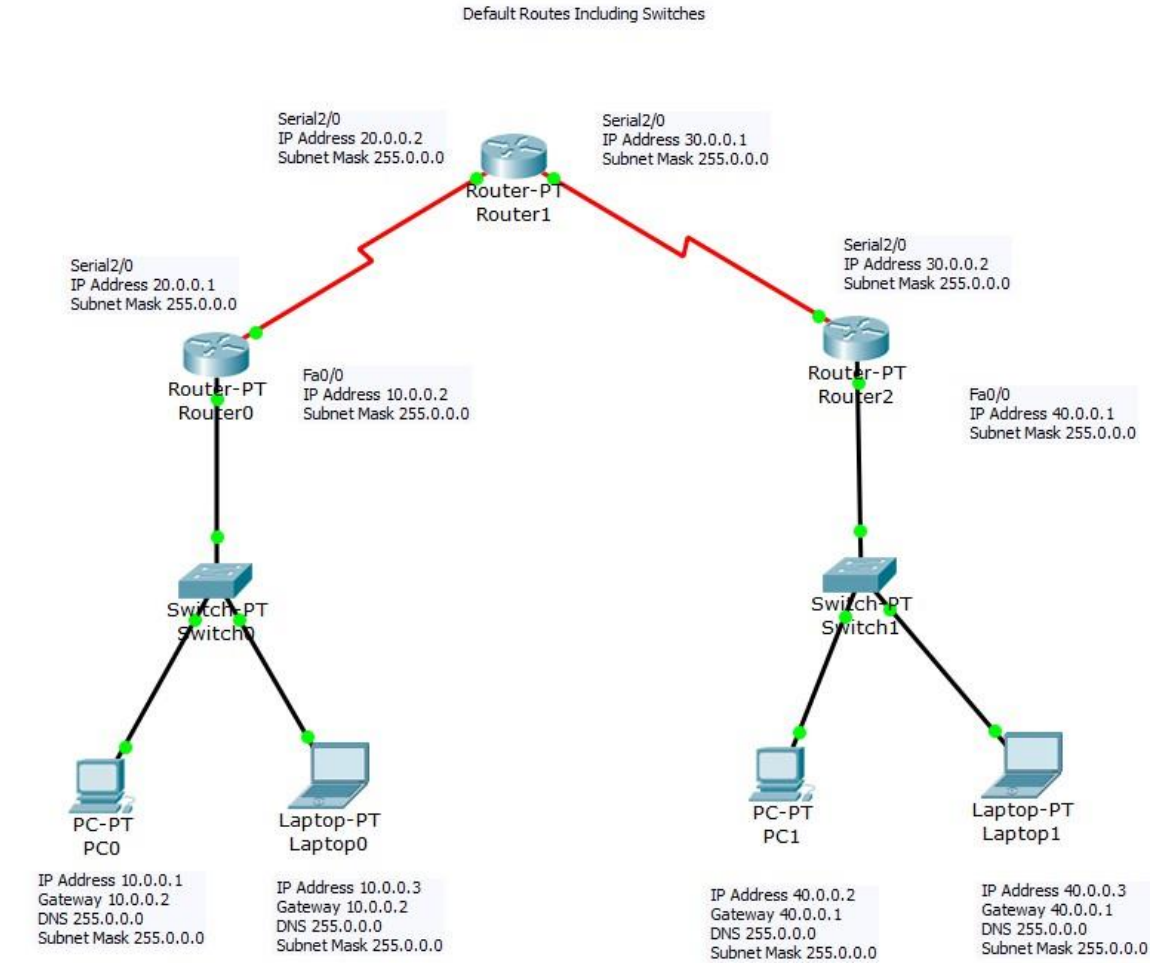
ping 40.0.0.2

pinging 40.0.0.2 with 32 bytes of data

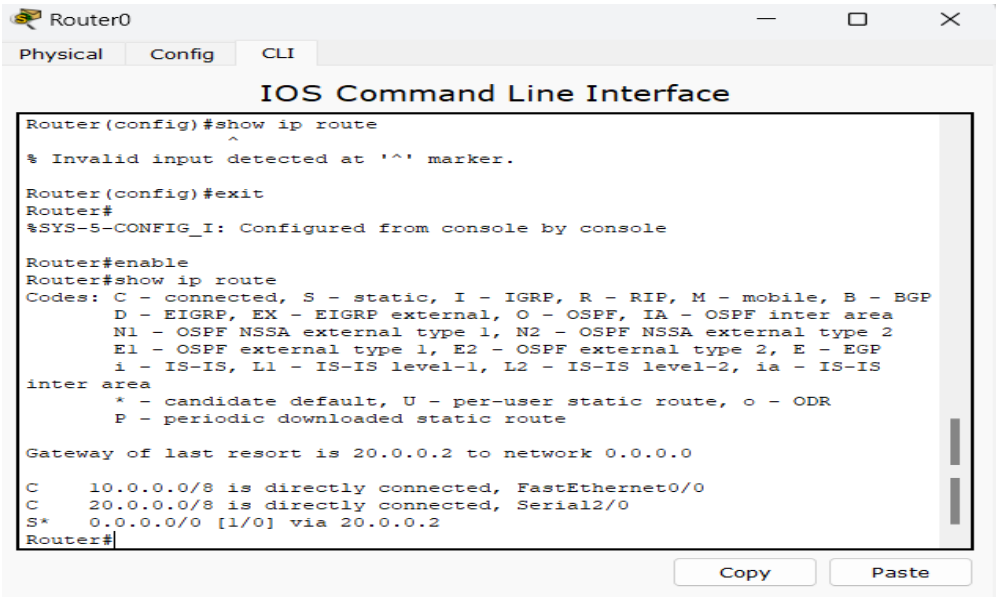
Reply from 40.0.0.2 bytes = 32 time = 9ms TTL=125
Reply from 40.0.0.2 bytes = 32 time = 11ms TTL=123
Reply from 40.0.0.2 bytes = 32 time = 4ms TTL=123
Reply from 40.0.0.2 bytes = 32 time = 32ms TTL=123

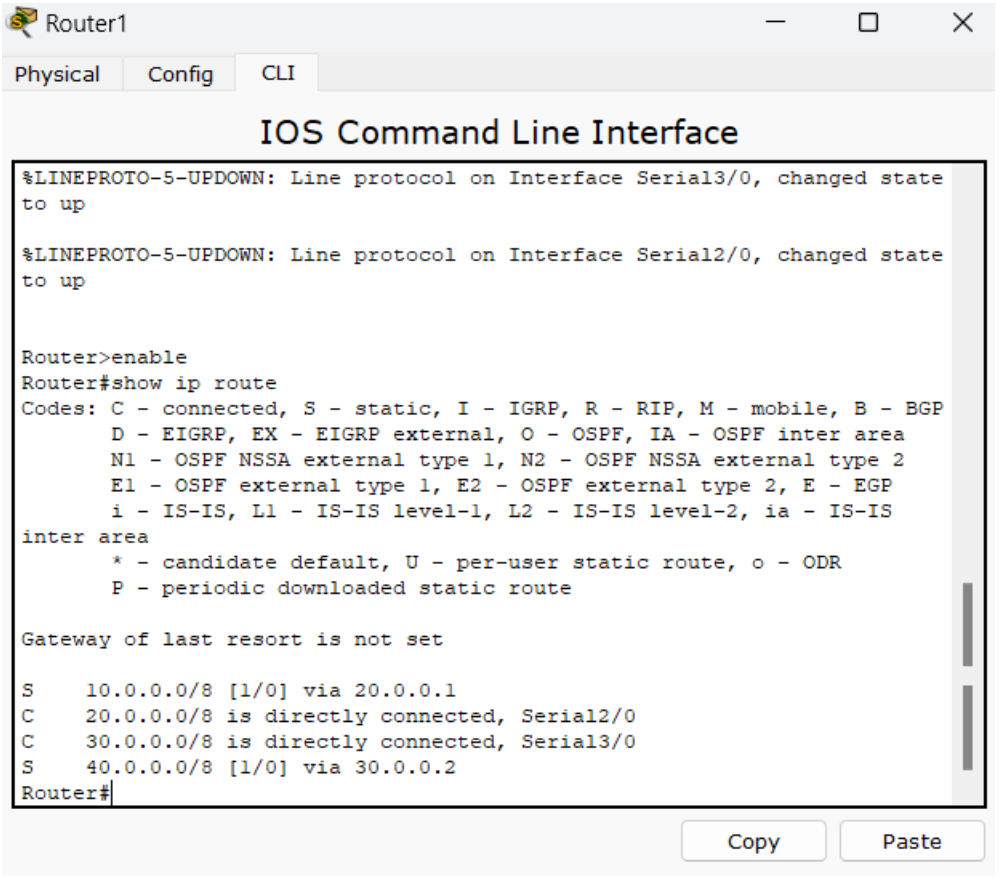
EXPERIMENT – 4 B

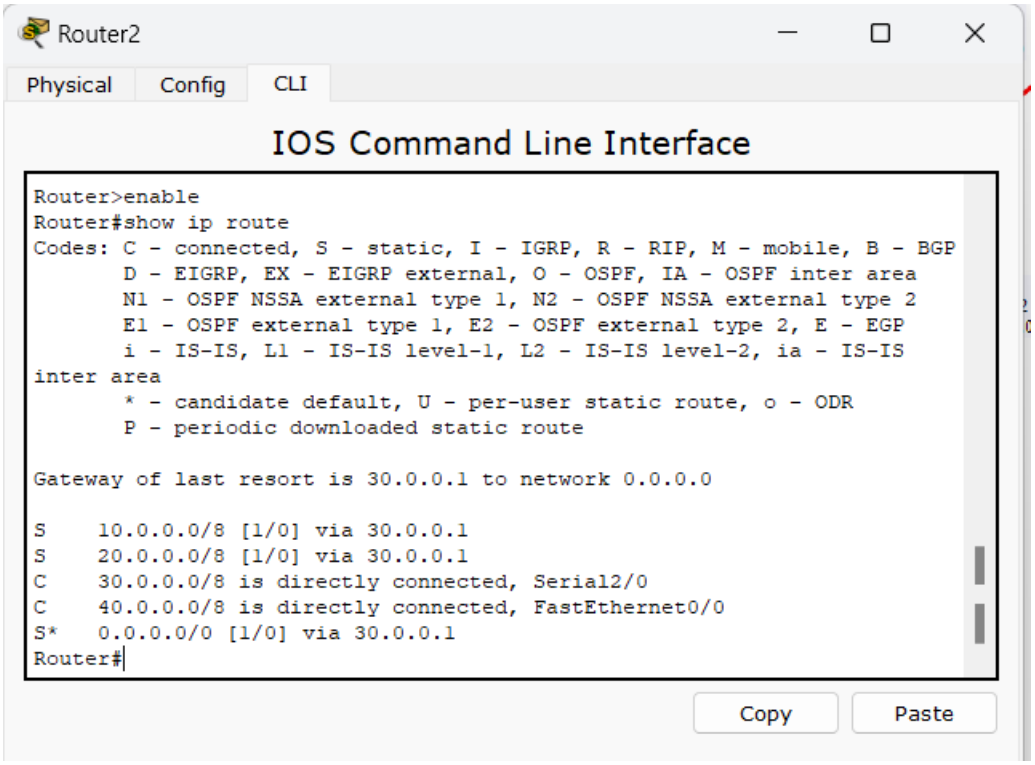
AIM: Configure default route to the Router.



show ip route





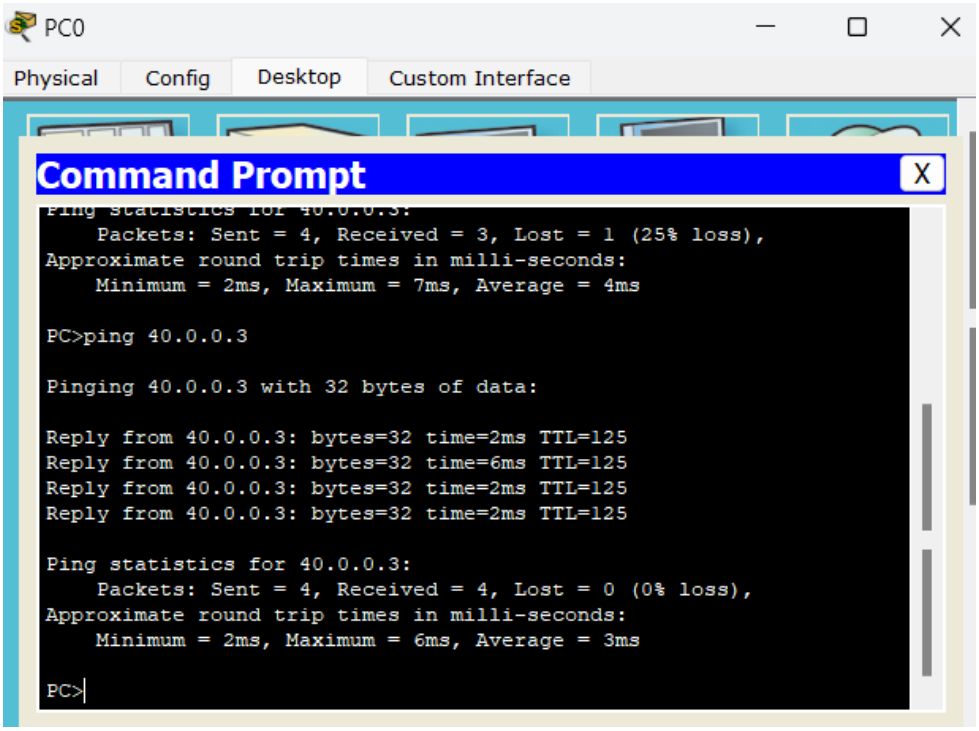


Verify Connectivity

Use the following commands to test and verify connectivity:

- **From PC0 to Laptop1:** Open PC0's Command Prompt:

ping 40.0.0.3


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 7ms, Average = 4ms
PC>ping 40.0.0.3
Pinging 40.0.0.3 with 32 bytes of data:
Reply from 40.0.0.3: bytes=32 time=2ms TTL=125
Reply from 40.0.0.3: bytes=32 time=6ms TTL=125
Reply from 40.0.0.3: bytes=32 time=2ms TTL=125
Reply from 40.0.0.3: bytes=32 time=2ms TTL=125
Ping statistics for 40.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 6ms, Average = 3ms
PC>

EXPERIMENT – 5

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

1. Understanding TELNET

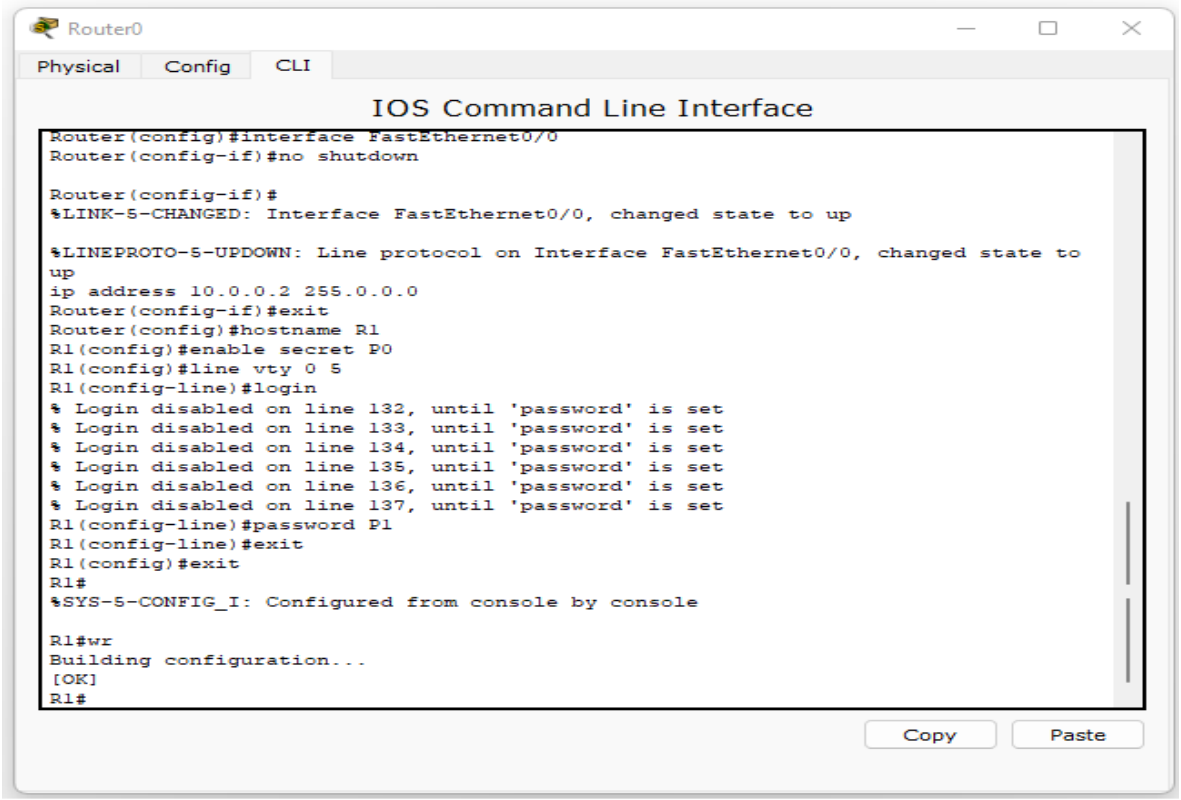
- TELNET is a network protocol used to provide bidirectional interactive communication over a TCP/IP network.
- It allows users to remotely access and manage network devices like routers, switches, and servers using command-line interfaces.

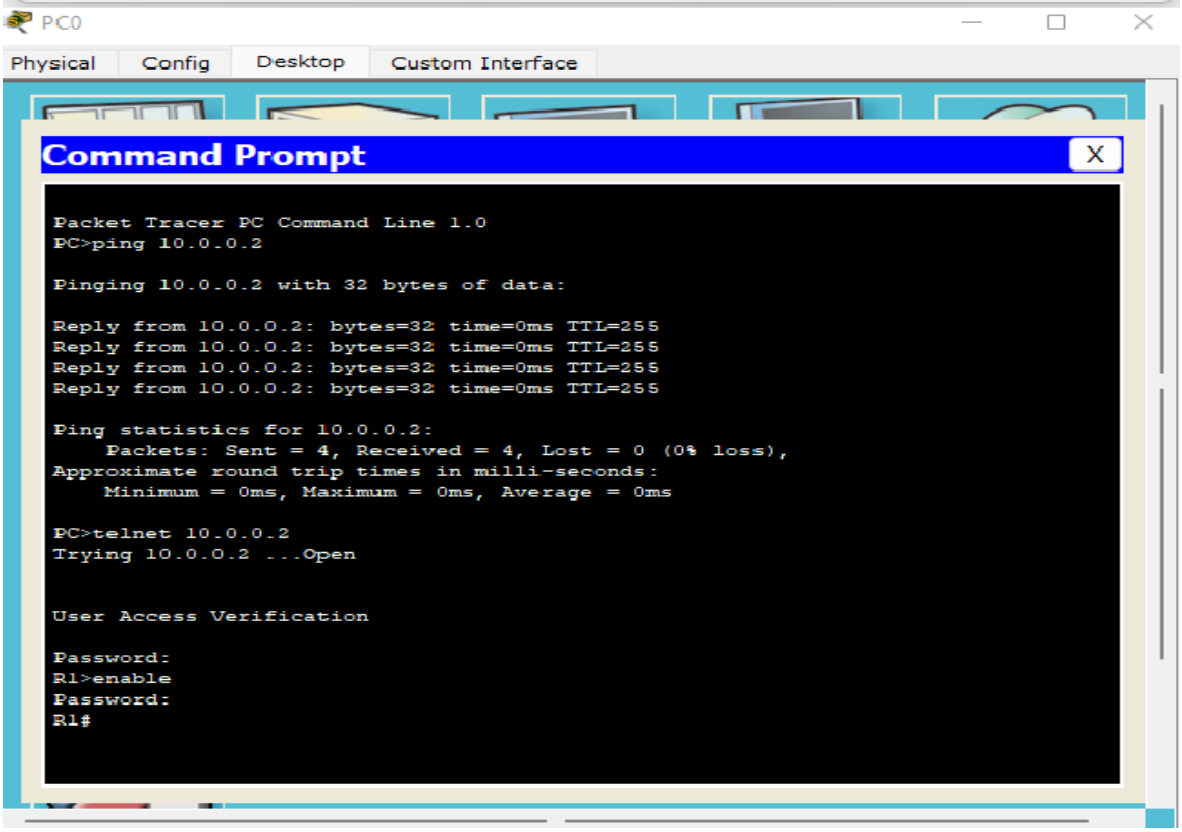
---

2. Pre-requisites

- **Router Configuration:**
  - TELNET service must be enabled on the router.
  - An IP address should be assigned to the router interface that is accessible from the PC in the IT office.
  - The router must have a username and password set up for authentication.

Topology:


Router0
Physical
Config
CLI
IOS Command Line Interface
Router(config)#interface FastEthernet0/0
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 address 10.0.0.2 255.0.0.0
Router(config-if)#exit
Router(config)#hostname R1
R1(config)#enable secret P0
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 P1
R1(config-line)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
R1#wr
Building configuration...
[OK]
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=0ms 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 = 0ms, Average = 0ms
PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#

Exp-5

AIM: To understand the operation of TELNET by accessing the router ~~to understan~~ placed in the server room from a PC in it office.

TELNET

- Tele - far
- a protocol that allows you to access the router remotely.
- configure a router remotely not directly.

Procedure:

1) Draw one generic PC and a router and establish a connection
2) Set ip address and gateway and subnet mask of PC0.

   ip address = 10.0.0.1
   Subnet = 255.0.0.0
   Gateway = 10.0.0.2
3) Configure the router

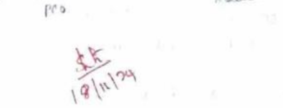
```
        # interface FastEthernet 0/0
        # no shutdown
        ip address 10.0.0.2 255.0.0.0
        # exit
Router(config)#hostname R1
R1(config) #enable secret P0 ] secret key
R1(config)#line vty 0 5        (virtual terminal)
R1(config-line)#login { Pd       0,1,2,3,4,5
     Login disabled on line 132, until password is set
R1(config-line)# password P1
#exit
#exit
R1#
R1#-wr     [OK]
```

4) Now go to command prompt of PC0 and type ping 10.0.0.2.
5) Then type telnet 10.0.0.2 and first type the Login password (P1) and type enable and type the secret key.

Output:

```
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 = 0ms TTL = 255
Reply from 10.0.0.2 bytes = 32 time = 0ms TTL = 255
Reply from 10.0.0.2 bytes = 32

PC > telnet 10.0.0.2
  Trying 10.0.0.2 ... open
User Access Verification
Password:
R1> enable
Password:
R1#
```

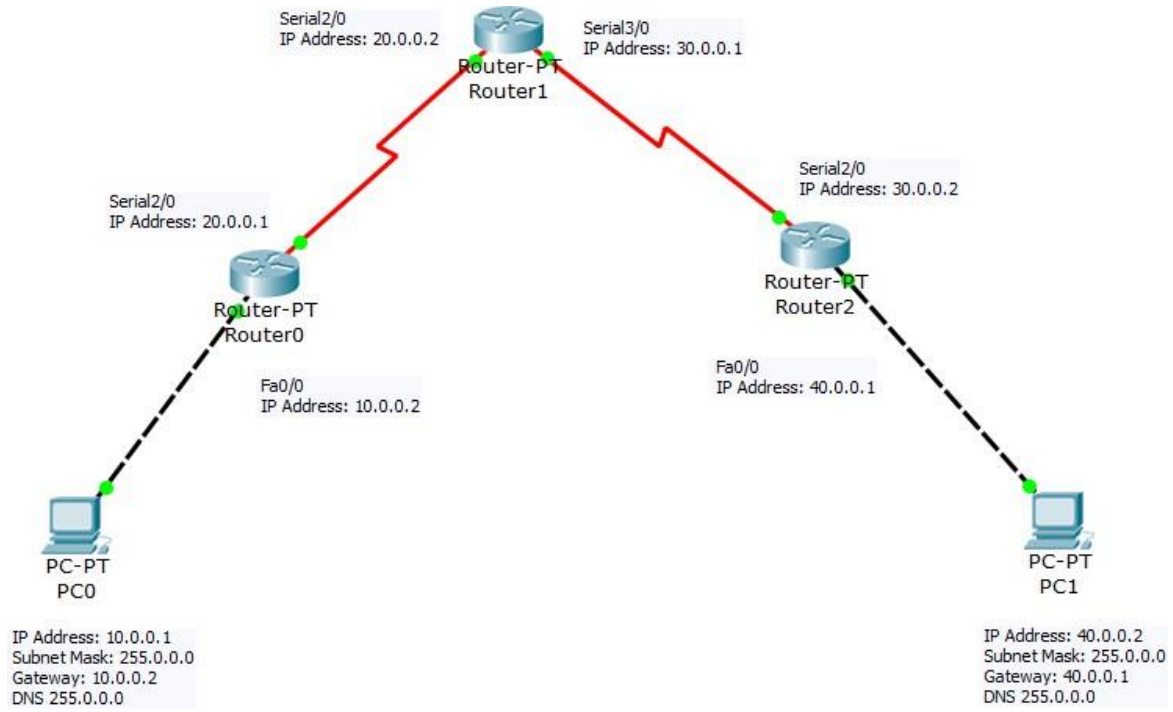
Topology



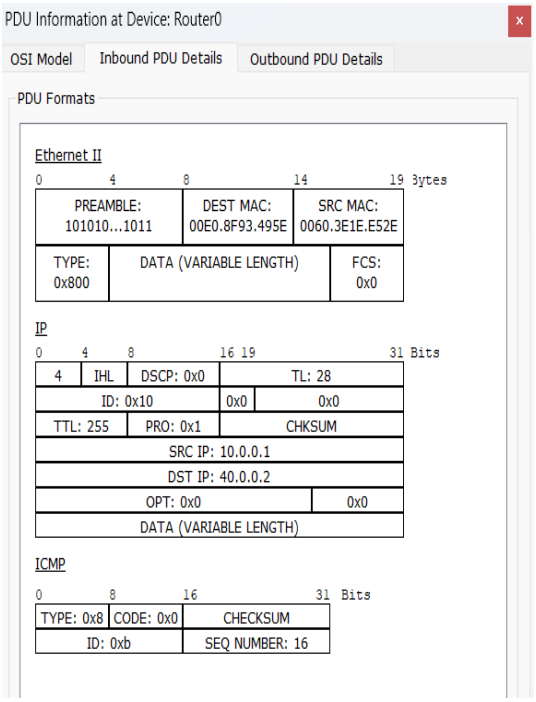
18/11/24

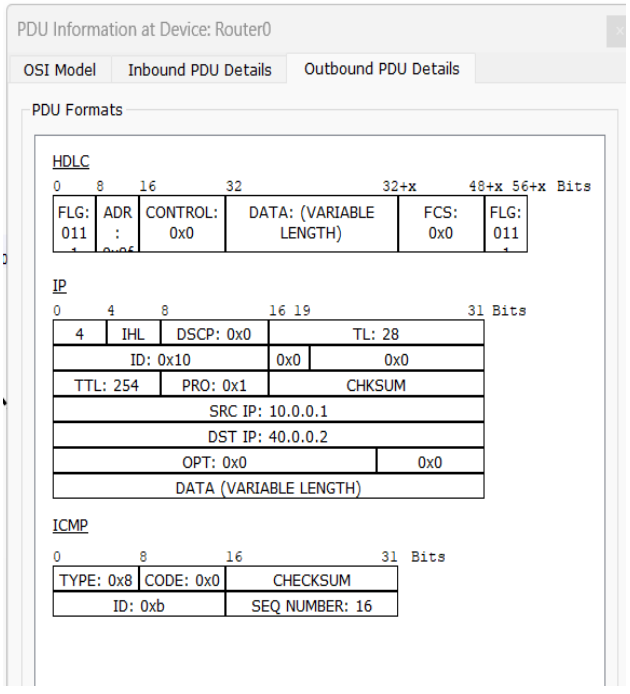
EXPERIMENT – 6

AIM: Demonstrate the TTL/ Life of a Packet.



Inbound and Outbound PDU Details:





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+x Bits
FLG: 011	ADR:	CONTROL: 0x0	DATA: (VARIABLE LENGTH)	FCS: 0x0	FLG: 011

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x11		0x0	0x0	
TTL: 254	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: 0xc		SEQ NUMBER: 17	

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+x Bits
FLG: 011	ADR:	CONTROL: 0x0	DATA: (VARIABLE LENGTH)	FCS: 0x0	FLG: 011

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x11		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: 0xc		SEQ NUMBER: 17	

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+x Bits
FLG: 011	ADR:	CONTROL: 0x0	DATA: (VARIABLE LENGTH)	FCS: 0x0	FLG: 011

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x11		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: 0xc		SEQ NUMBER: 17	

PDU Information at Device: Router2

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

Ethernet II

0	4	8	14	19 Bytes
PREAMBLE: 101010...1011		DEST MAC: 000B.BE20.4BD4	SRC MAC: 0050.0F28.BBEB	
TYPE: 0x800	DATA (VARIABLE LENGTH)			FCS: 0x0

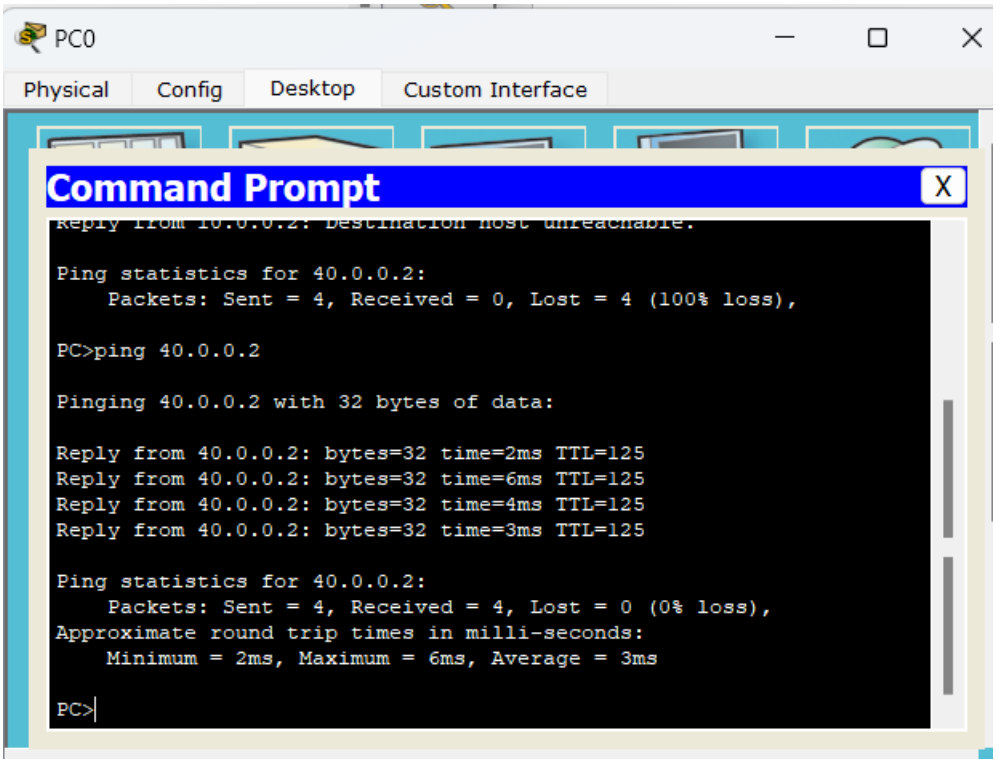
IP

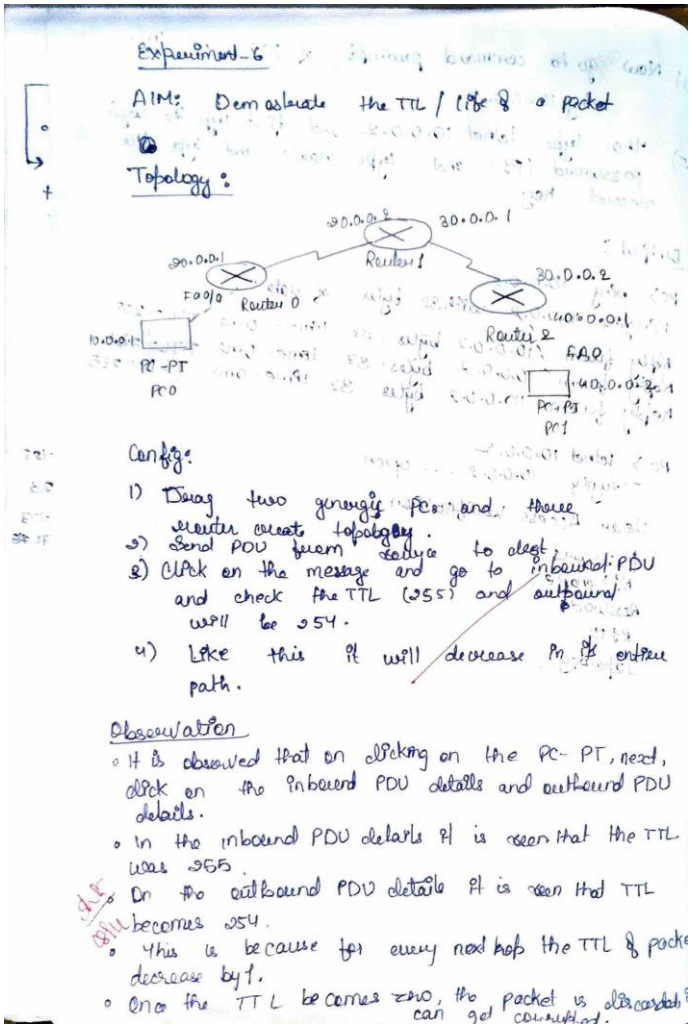
0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x11		0x0	0x0	
TTL: 252	PRO: 0x1	CHKSUM		
SRC IP: 10.0.0.1				
DST IP: 40.0.0.2				
OPT: 0x0			0x0	
DATA (VARIABLE LENGTH)				

ICMP

0	8	16	31 Bits
TYPE: 0x8	CODE: 0x0	CHECKSUM	
ID: 0xc		SEQ NUMBER: 17	

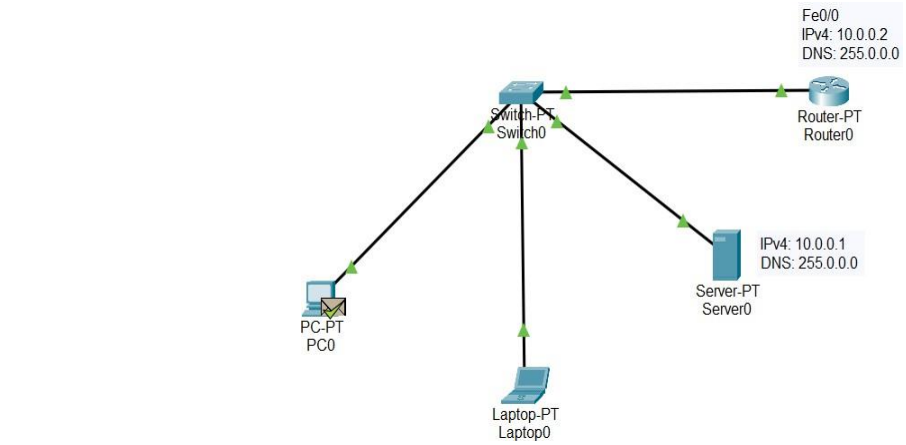
Fire	Last Status	Source	Destination	Type	Color	Time(se	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Reply from 10.0.0.2: Destination host unreachable.
Ping statistics for 40.0.0.2:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
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=2ms TTL=125
Reply from 40.0.0.2: bytes=32 time=6ms TTL=125
Reply from 40.0.0.2: bytes=32 time=4ms TTL=125
Reply from 40.0.0.2: bytes=32 time=3ms TTL=125
Ping statistics for 40.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 6ms, Average = 3ms
PC>


Experiment-6
AIM: Demonstrate the TTL / Life of a packet
Topology:
30.0.0.1
Router1
30.0.0.2
Router 0
Router 2
FA0/0
FA0
PC-PT
PC0
PC-PT
PC1
Config:
1) Drag two generic PC and three router create topology.
2) Send PDU from source to dest.
3) Click on the message and go to inbound PDU and check the TTL (255) and outbound will be 254.
4) Like this it will decrease in the entire path.
Observation
• It is observed that on clicking on the PC-PT, next, click on the inbound PDU details and outbound PDU details.
• In the inbound PDU details it is seen that the TTL was 255.
• In the outbound PDU details it is seen that TTL becomes 254.
• This is because for every next hop the TTL of packe decrease by 1.
• Once the TTL becomes zero, the packet is discarded can get

EXPERIMENT – 7 A

AIM: To Configure IP addresses of the host using DHCP server within a LAN.



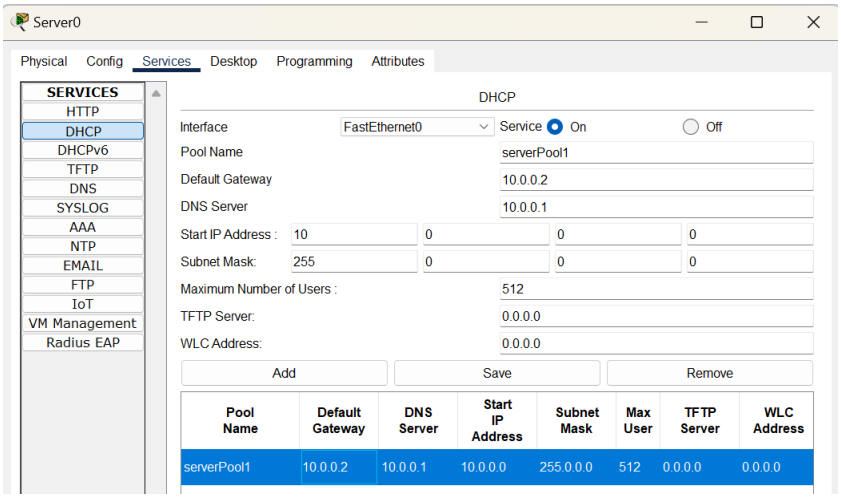


Figure 7.1: DHCP Service, Server0

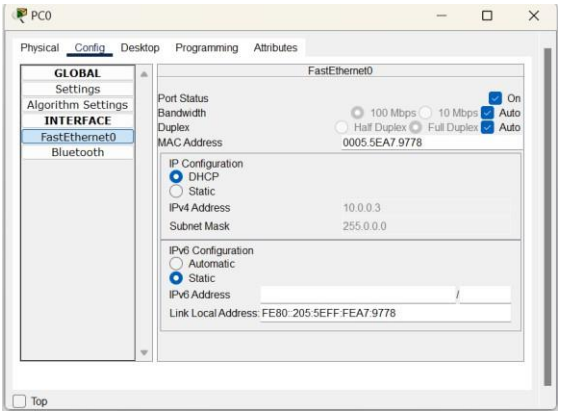


Figure 7.2: DHCP Service, PC0

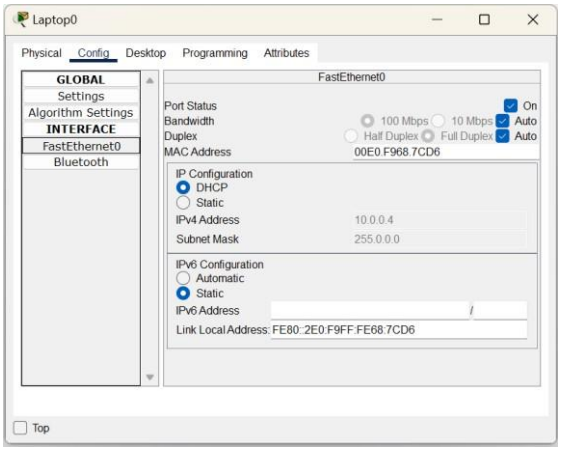
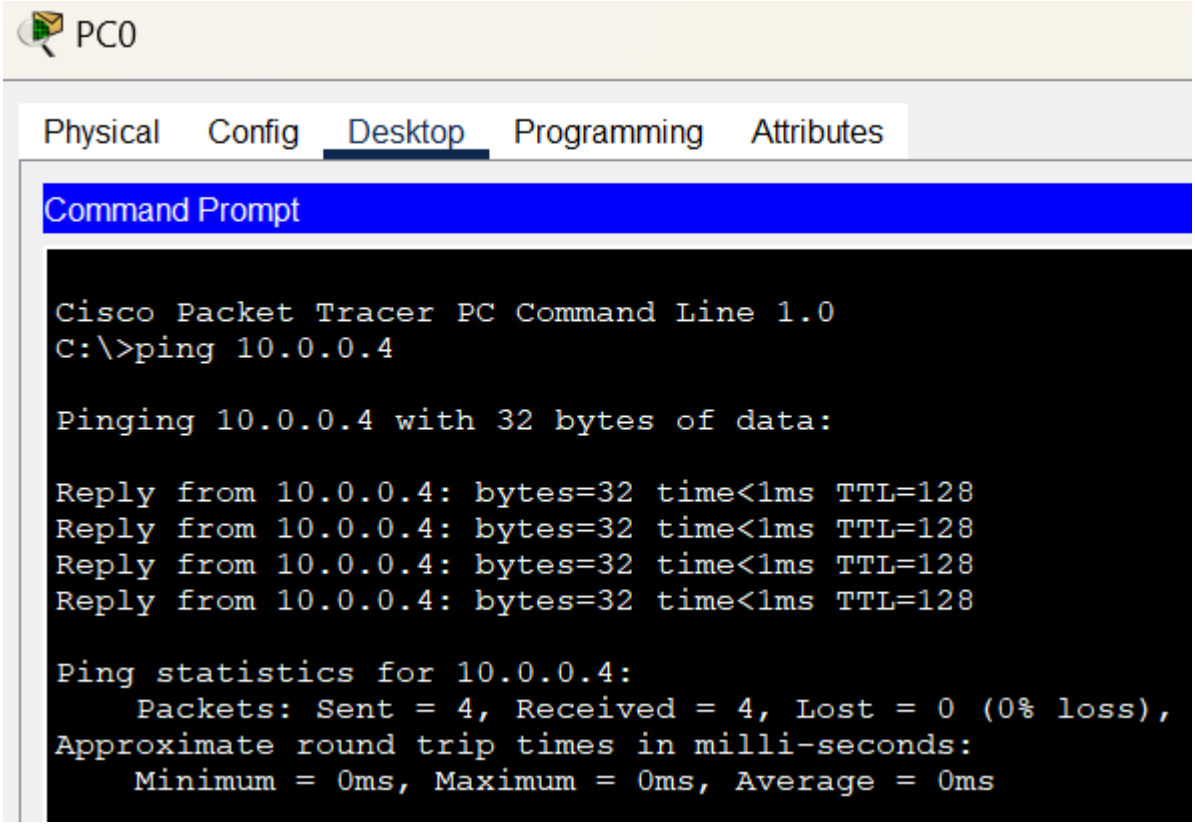


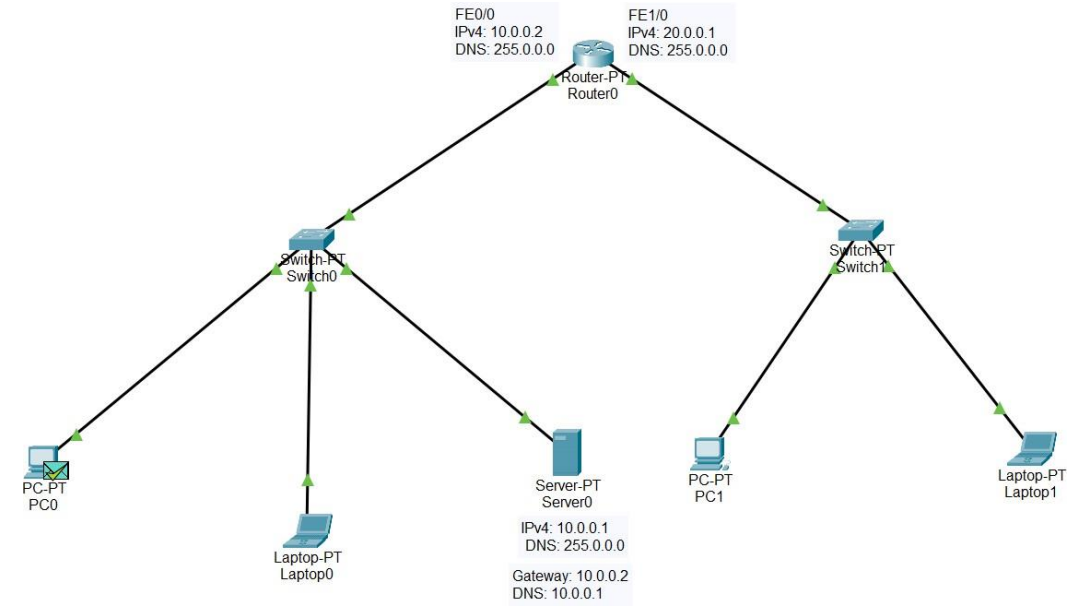
Figure 7.3: DHCP Service, Laptop0

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



EXPERIMENT – 7 B

To Configure IP addresses of the host using DHCP server outside a LAN.



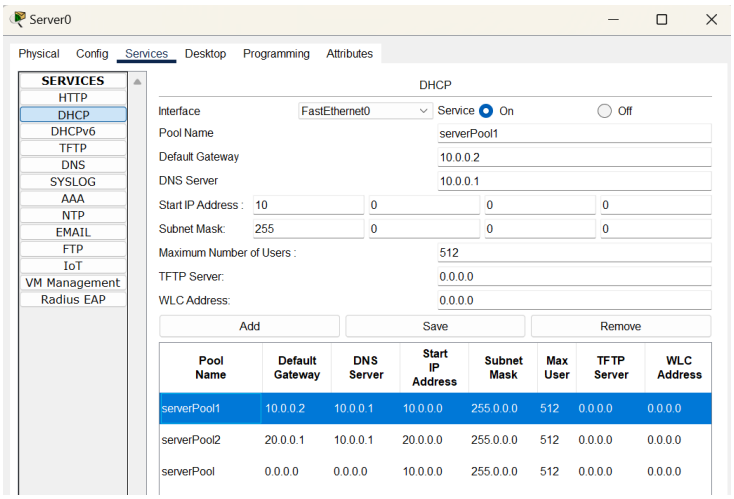
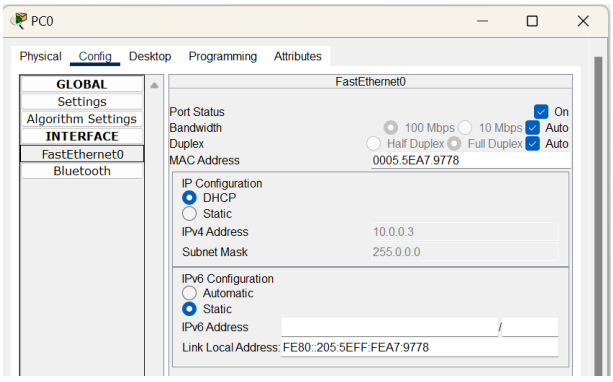


Figure 7.1.1: DHCP Service, Server0



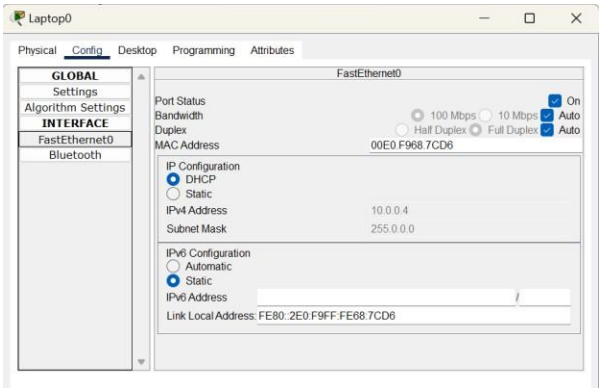


Figure7.2.2: DHCP Service, PC0

Figure 7.2.3: DHCP Service, Laptop0

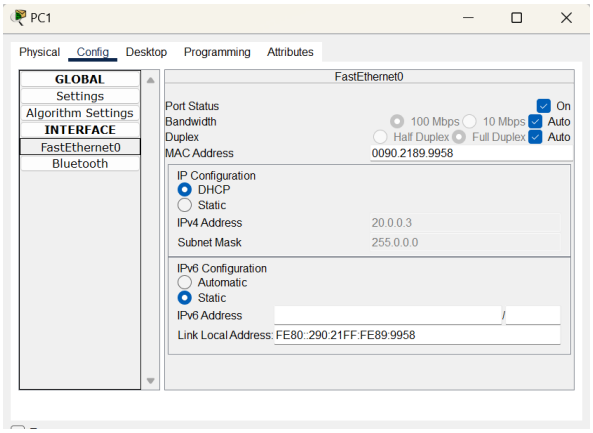


Figure 7.2.4: DHCP Service, PC1

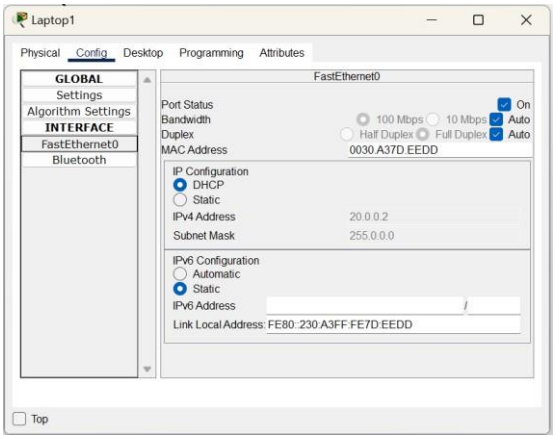


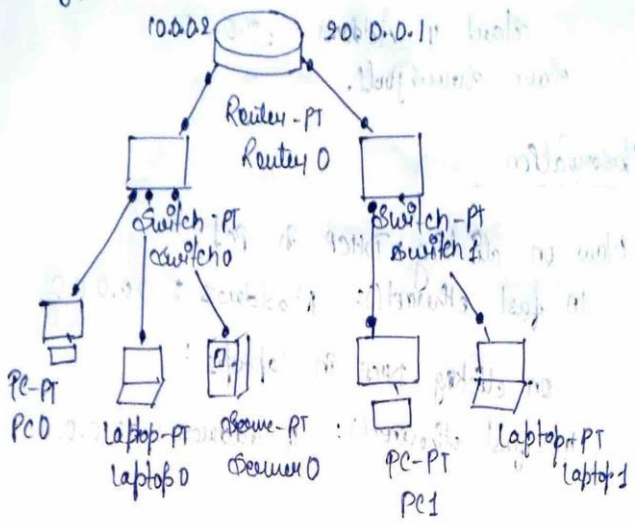
Figure 7.2.5: DHCP Service, Laptop1

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)	

12.11.24

AIM: To configure DHCP server present outside the LAN.

Topology:



Configuration:

- As our topology, connections are made through copper straight through. And for the main network, the configuration are done as for previous experiment.
- And for remote network the:

Config R0 in the fast ethernet 0/0
IP address: 20.0.0.1

And > IP helper 10.0.0.1
in fast ethernet 0/0
> IP helper 20.0.0.1

Config Server0. At services, using DHCP protocol:
Service is ON

→ PoolName: ServerPool2
Default gateway: 20.0.0.1
DNS - 10.0.0.1
Start IP: 20.0.0.0
Save server pools.

PoolName: ServerPool1
Default Gateway: 10.0.0.2
DNS server: 10.0.0.1
Start IP Address: 10.0.0.0
Save ServerPools.

Observation

Now on clicking DHCP in PC1:
In fast ethernet0: IP address: 20.0.0.2

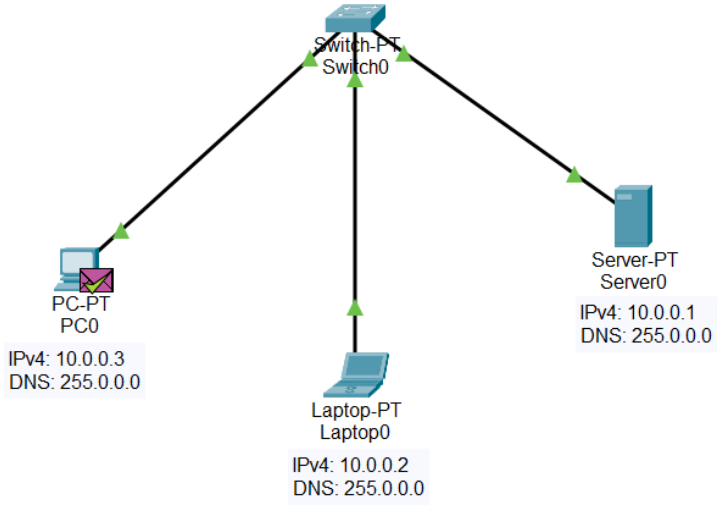
on clicking DHCP in laptop1:
In fast ethernet0: IP address: 20.0.0.3

Output

```
Ping 20.0.0.2
pinging 20.0.0.2 with 32 bytes of data.
Reply from 20.0.0.2 bytes = 32 time TTL 128
Ping statistics for 20.0.0.2
```

EXPERIMENT – 8

To Configure DNS server to demonstrate the mapping of IP addresses and Domain names.



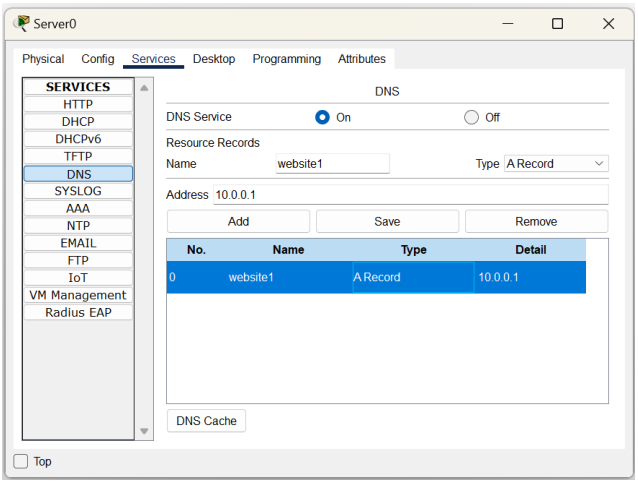


Figure 8.1: DNS Service, Server0

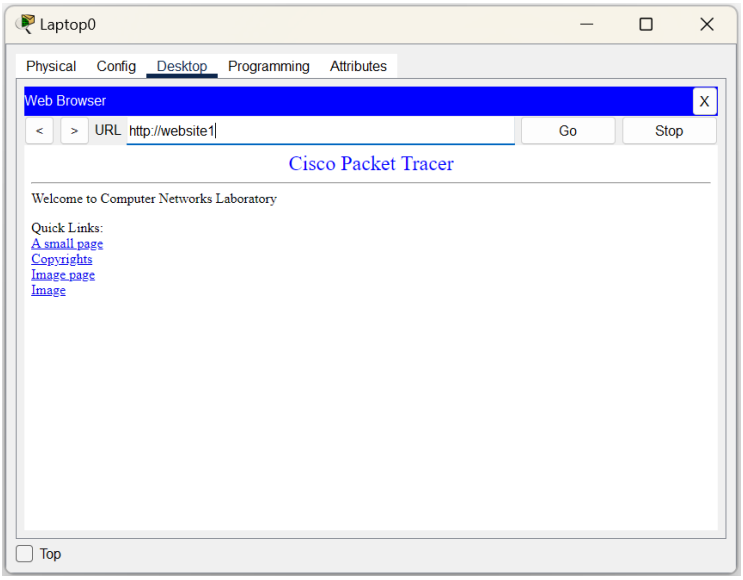
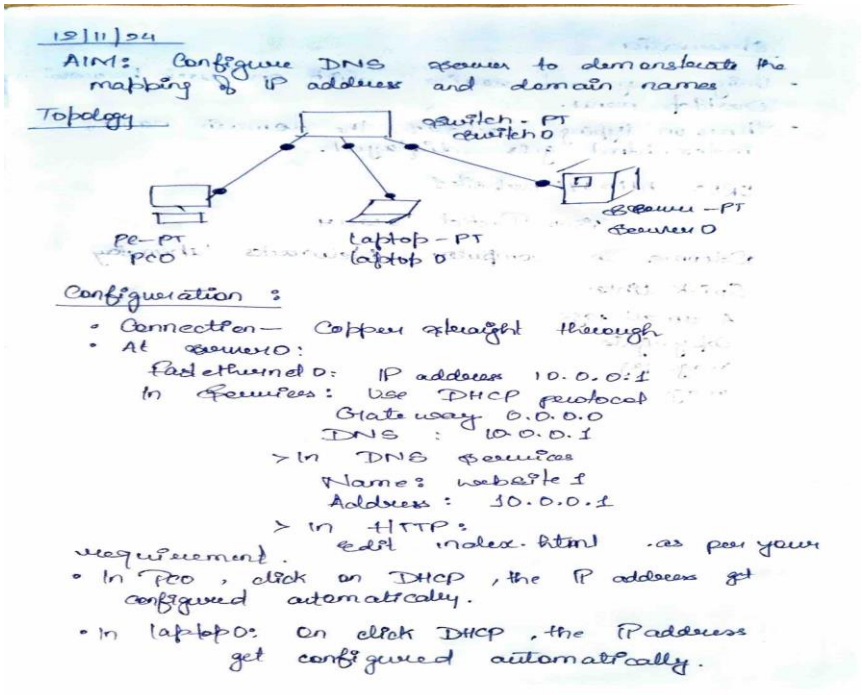
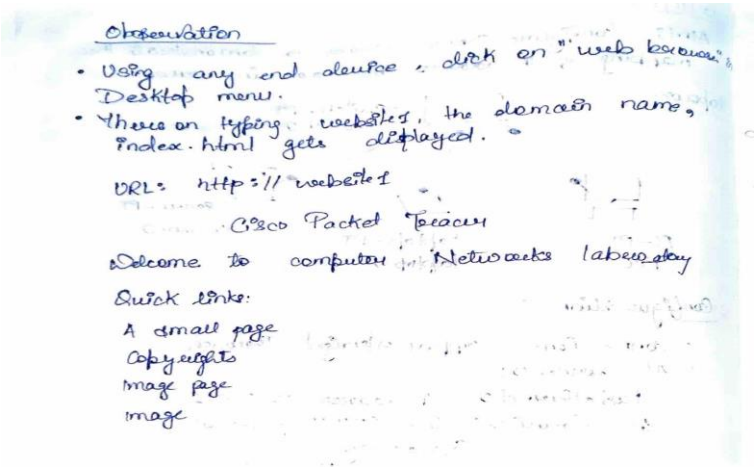


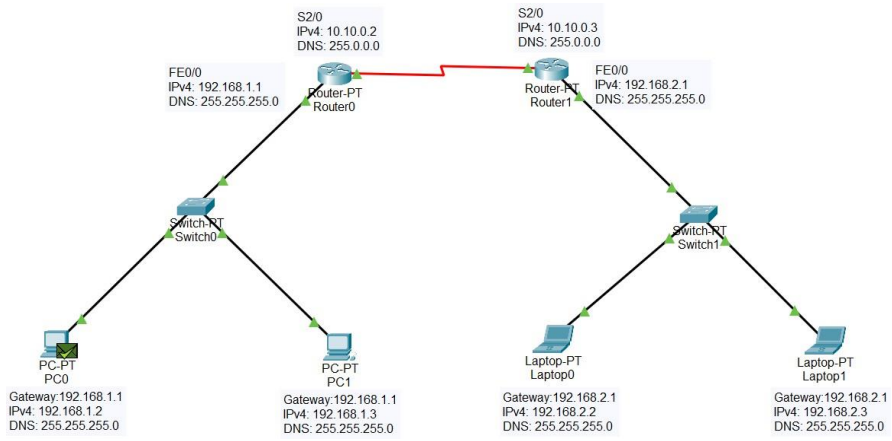
Figure 8.2: DNS Service, Laptop0





EXPERIMENT – 9

To Configure RIP routing protocol in Routers.



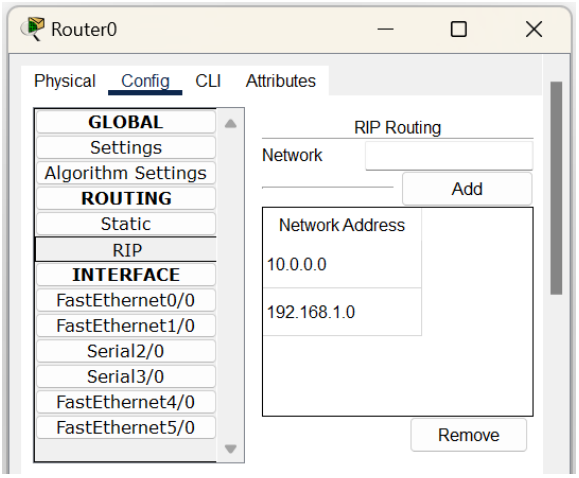


Figure 9.1: RIP, Router0

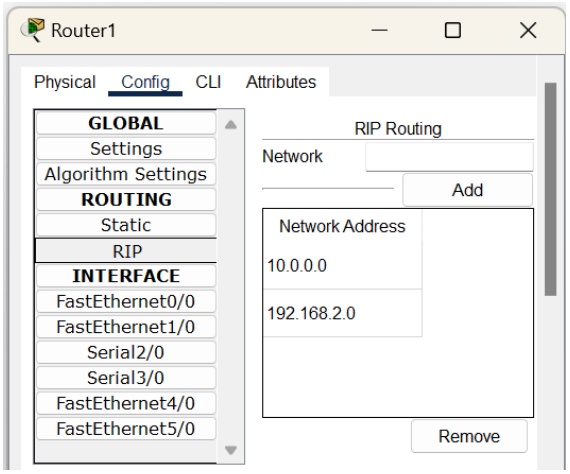


Figure 9.2: RIP, Router

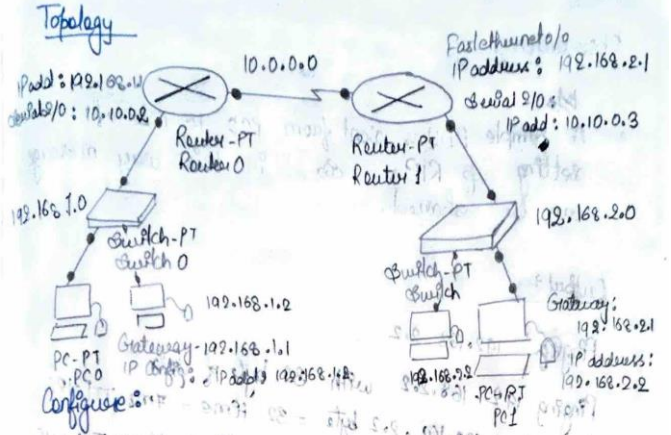
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop1	ICMP		0.000	N	0	(edit)	

```
C:\>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Reply from 192.168.2.3: bytes=32 time=18ms TTL=126
Reply from 192.168.2.3: bytes=32 time=14ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 18ms, Average = 8ms
```

19/11/2024

AIM: Configure RIP routing protocol in Routers

Topology



Configure:

1. Take two generic PCs and for Pc0 put gateway as 192.168.1.1 and then go to Desktop IP Config and in static put ip address as 192.168.1.2 with default gateway as 192.168.1.1. Similarly do for PC1 with IP address as 192.168.2.2 and default gateway as 192.168.2.1. Give subnet mask as 255.255.255.0
2. Put two generic switches and two routers and open one of the routers and then give IP address as 192.168.1.1 with subnet mask as 255.255.255.0 with port status on in fast ethernet 0/0. Then go to Serial 2/0 and put clock rate 64000 with IP address as 10.10.0.2 and subnet mask as 255.0.0.0. Similarly do for router 1 in fastethernet 0/0 with ip address as 192.168.2.1 with subnet mask 255.255.255.0 and port status on and then go to CLI and type commands:

```
#exit
#router rip
#network 192.168.2.0
#network 10.0.0.0
#exit
```

3. Connect the routers, switches and Pcs with connection.

Observation:

- A simple PDU is sent from PC0 to PC1 after setting up RIP. So, in this way message can be shared.

Output:

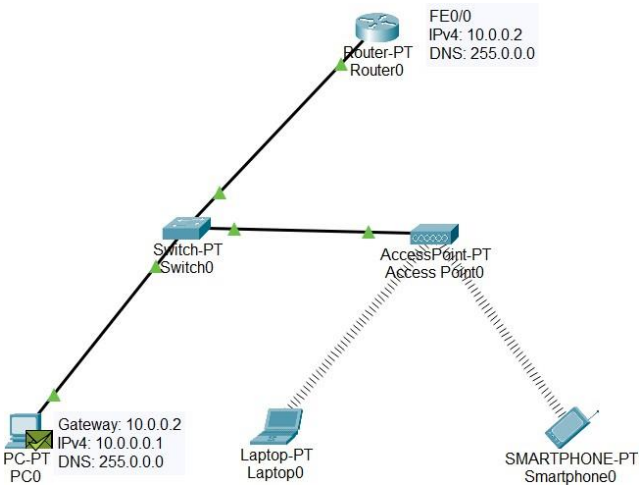
Pinging 192.168.2.2

Pinging 192.168.2.2 with 32 byte of data
Reply from 192.168.2.2 byte = 32 time = 7ms TTL=126
Reply from 192.168.2.2 byte = 32 time = 7ms TTL=126
Reply from 192.168.2.2 byte = 32 time = 6ms TTL=126
Reply from 192.168.2.2 byte = 32 time = 6ms TTL=126

19/11/24

EXPERIMENT – 10

To demonstrate communication between two devices using a wireless LAN.



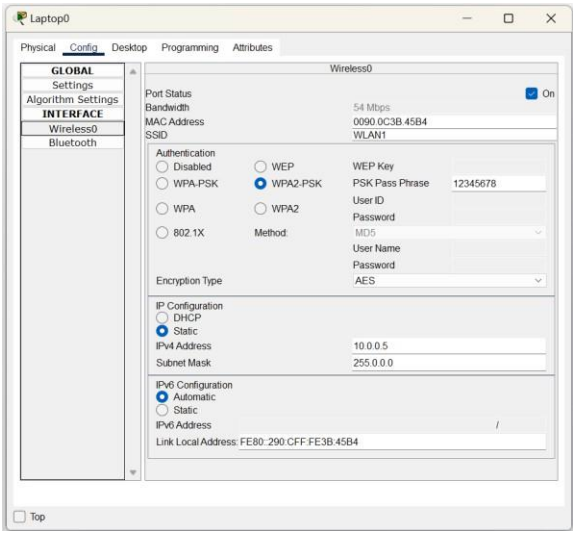


Figure 10.1: Laptop0, Wireless0

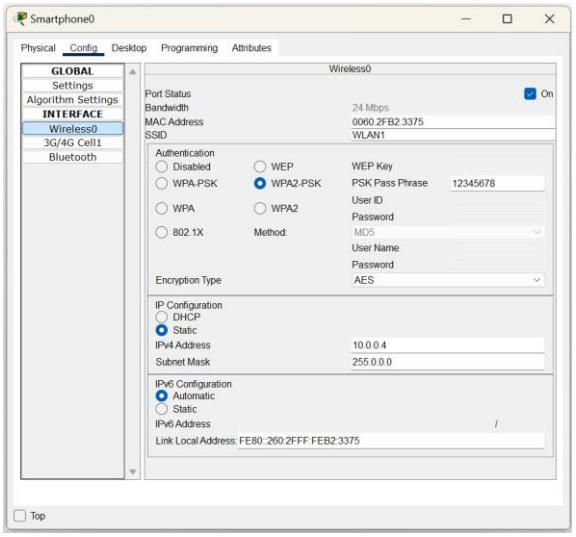


Figure 10.2: Smartphone0, Wireless0

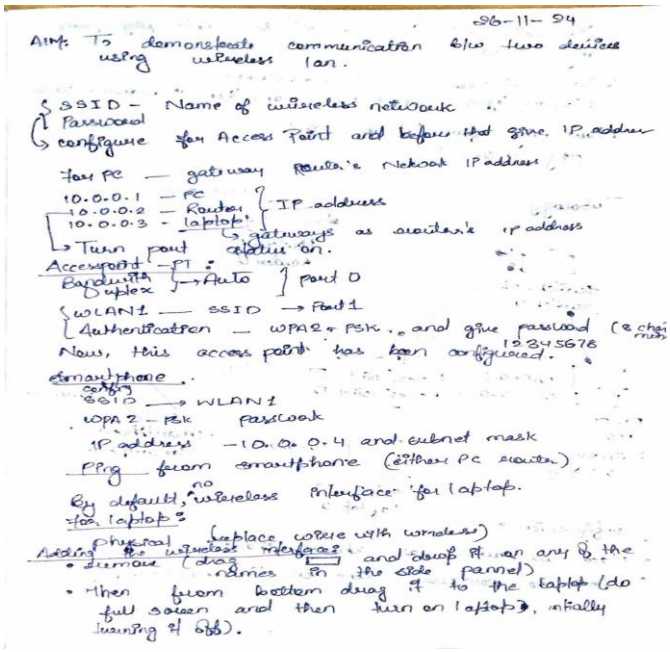
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

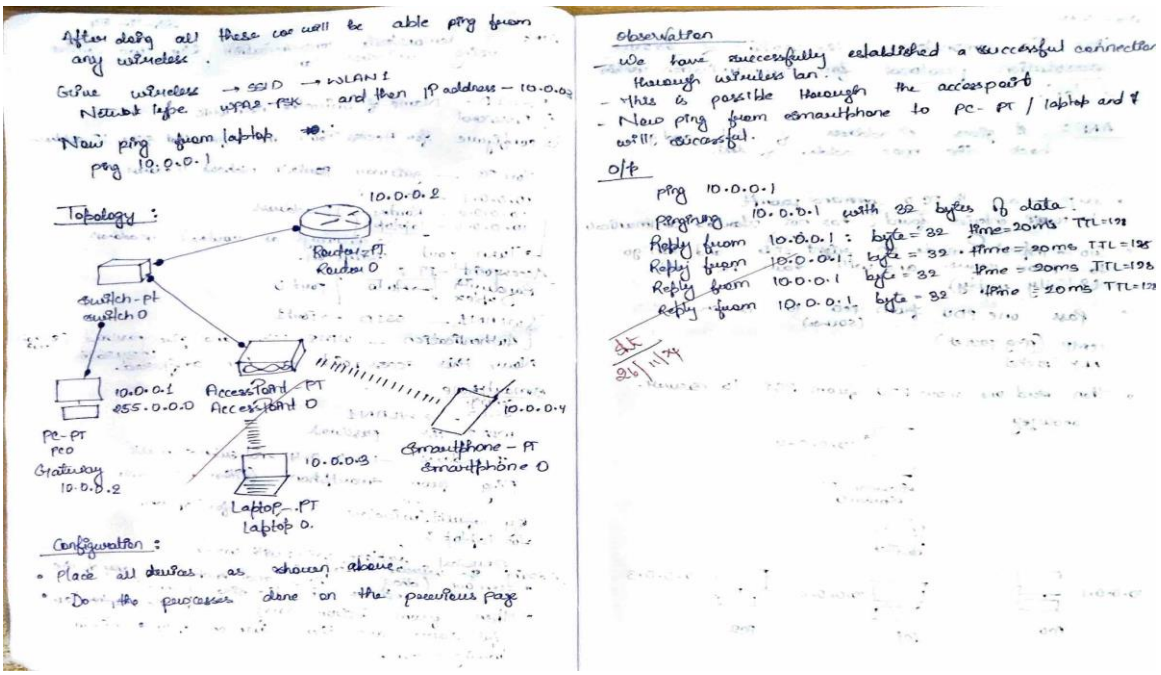
PC0

Physical | Config | Desktop | Programming | Attributes

Command Prompt

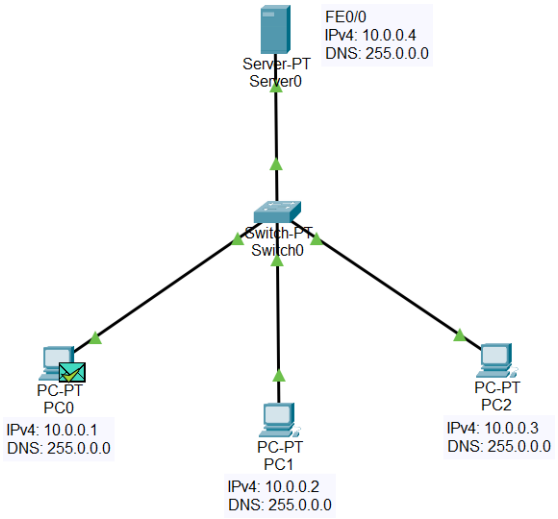
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data:

Reply from 10.0.0.5: bytes=32 time=8ms TTL=128
Reply from 10.0.0.5: bytes=32 time=28ms TTL=128
Reply from 10.0.0.5: bytes=32 time=30ms TTL=128
Reply from 10.0.0.5: bytes=32 time=36ms TTL=128

Ping statistics for 10.0.0.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 36ms, Average = 25ms
```





EXPERIMENT – 11

To demonstrate the working of Address Resolution Protocol (ARP) within a LAN for communication.



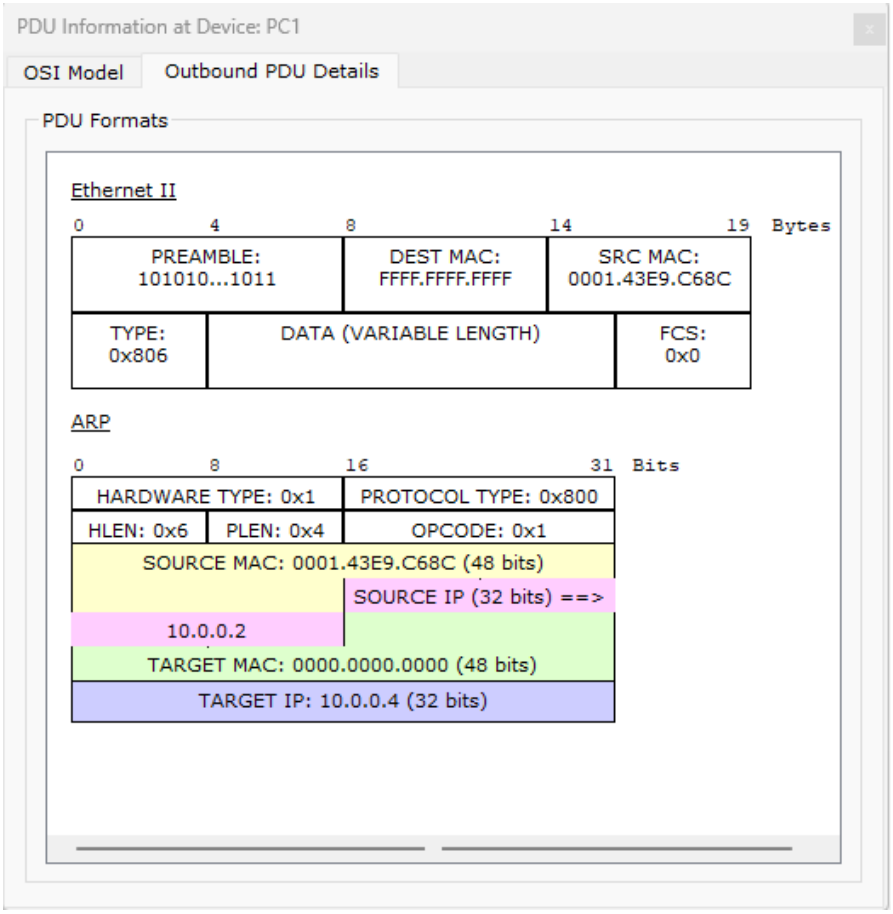


Figure 11.1: Inbound ARP, PC1

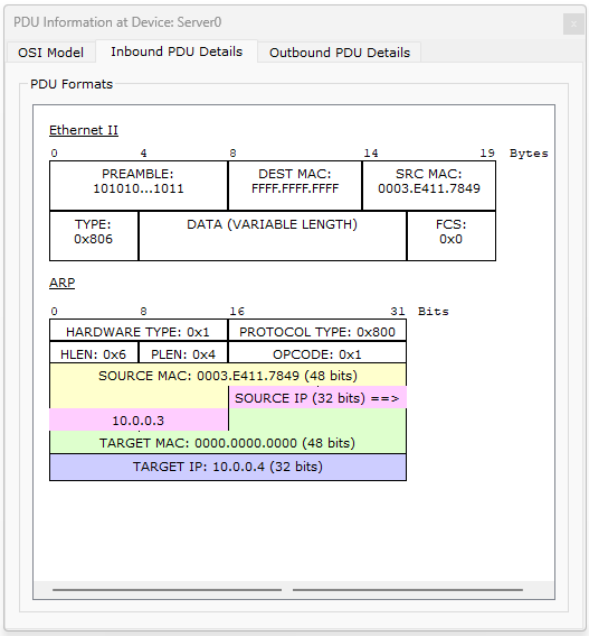


Figure 11.2: Inbound ARP, Server0

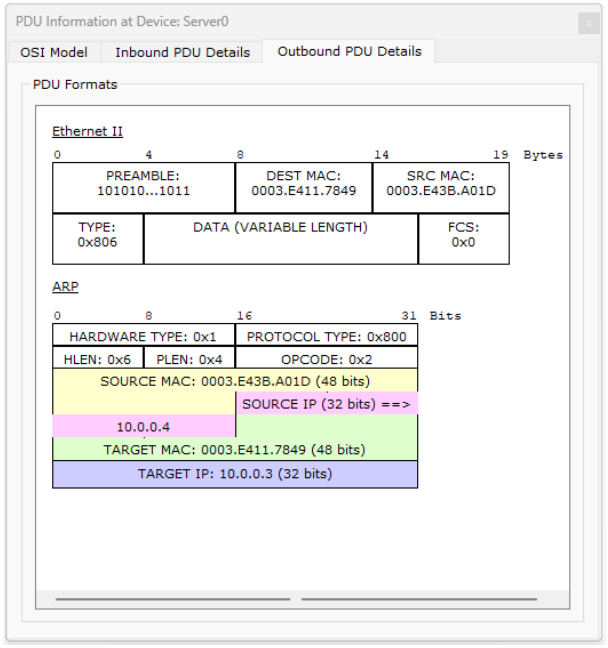


Figure 11.3: Outbound ARP, Server0

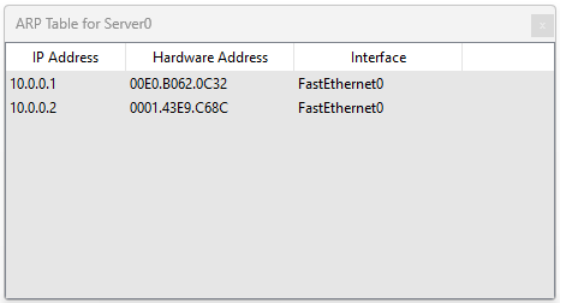


Figure 11.4: ARP Table, Server0

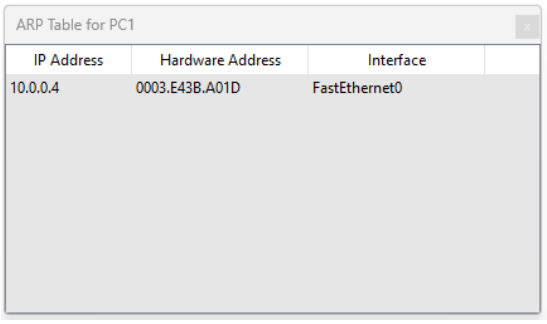
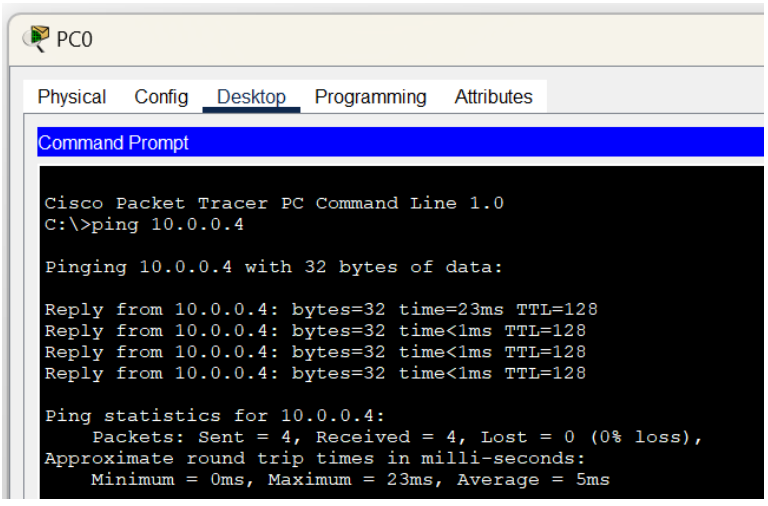


Figure 11.5: ARP Table, PC1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Server0	ICMP		0.000	N	0	(edit)	



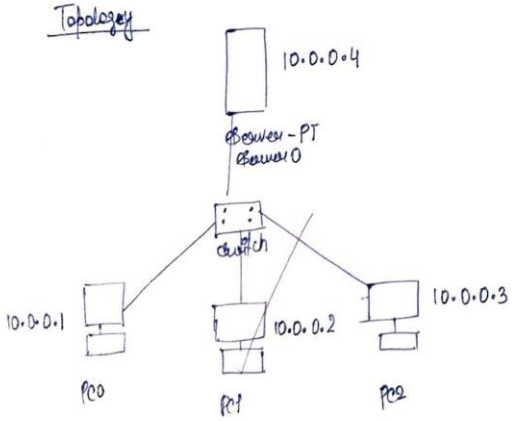
26-11-24

AIM:- To demonstrate the working of Address resolution protocol for communication within a lan.

ARP: It gives IP address of dest and gets back the mac addr. of dest.

- arp -a in PC o command prompt
  - to check ARP entries found (if has not established communication)
- Go to inspect mode on right side and then go to PC or server and will have ARP table (initially empty)
- Pass one PDU from PC0 to server (same)
  - ICMP (ping packet)
  - ARP (packet)
- Then send one more PDU from PC1 to server.

Topology



Configuration:

- Configure as shown method.
- Send a simple PDU from PC0 to PC1 and then go to simulation mode Auto capture/play and then send one more PDU from PC1 to Server.

Observation:

When sending first PDU entries will be created in ARP table of PC0 and Server 0 and then again next PDU will be sent and this will add entries in the PC1 and Server 0.

Initially when it doesn't know, it is broadcasted and when acknowledgement comes, it only goes to the PC1. and we can check the PDU information when are we are sending and when it comes back. when it comes, in its inbounds PDU details we can see the mac address.

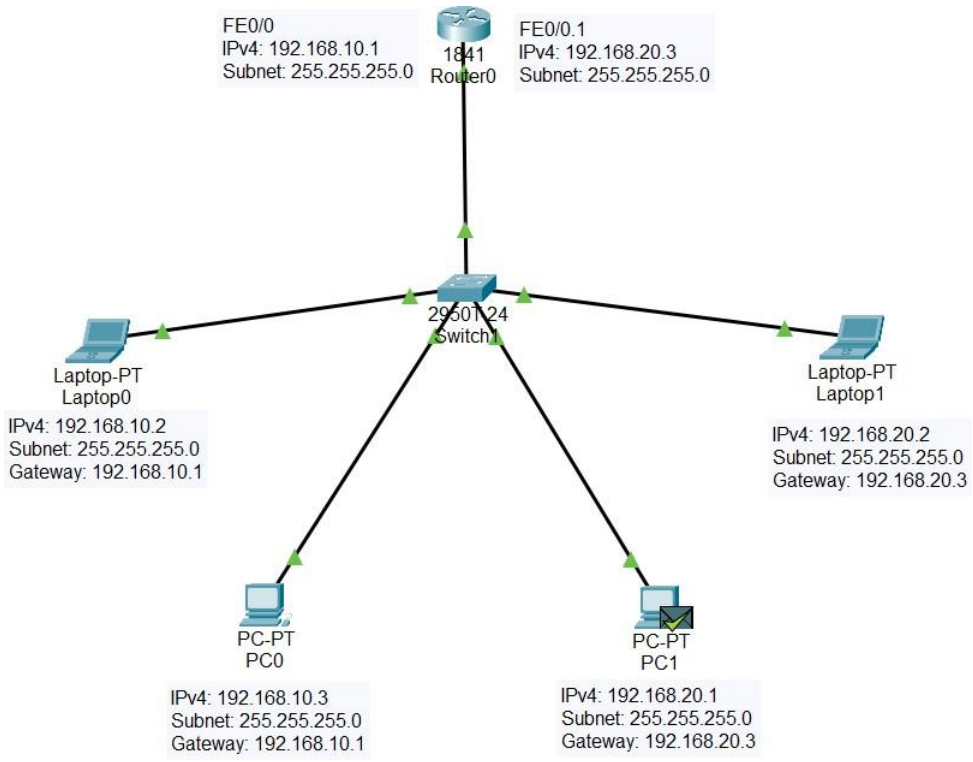
Inbound PDU details

Source MAC: 0090.2BA3.1417 (48 bits)
Source IP (32 bits) ==> 10.0.0.4
Target MAC: 0001.9740.1215 (48 bits)
Target IP: 10.0.0.2 (32 bits)

26/11/24

EXPERIMENT – 12

To create a VLAN on top of the physical LAN and enable communication between physical LAN and virtual LAN.



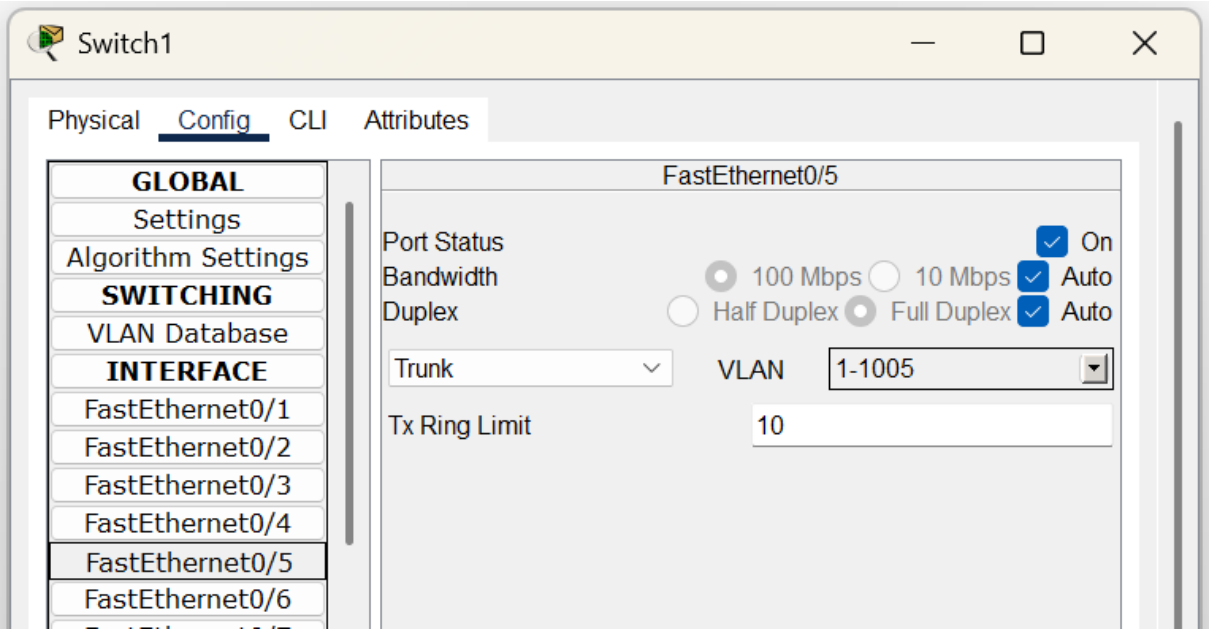


Figure 12.1: FE0/5 Switchport Trunk

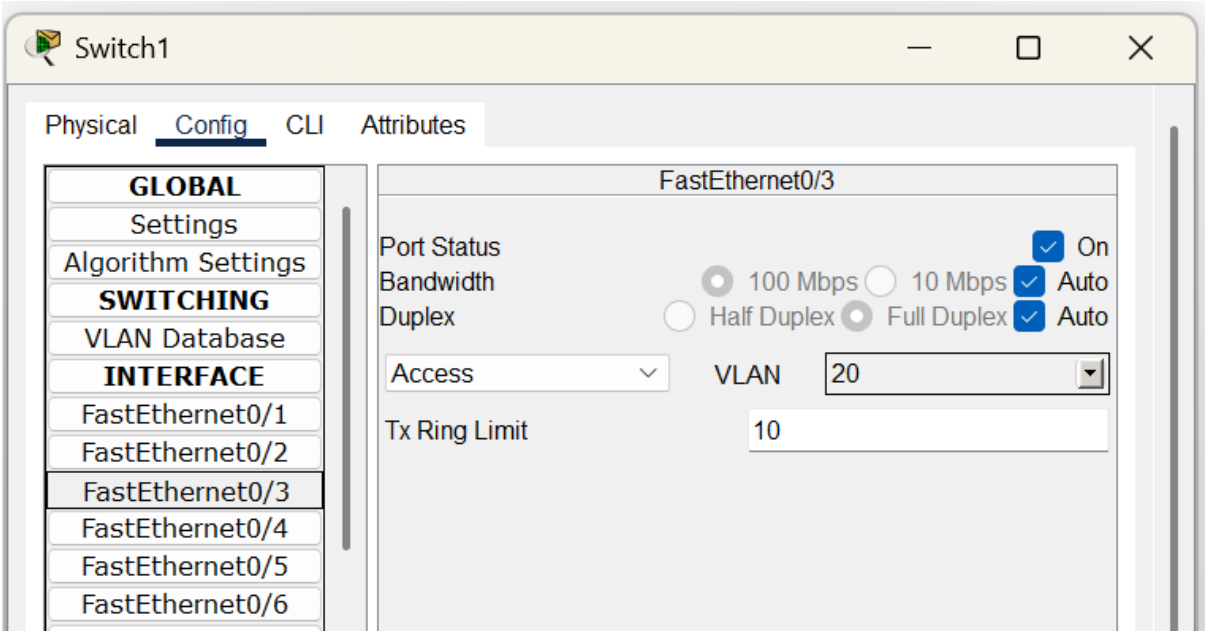


Figure 12.2: FE0/3 Switchport Access

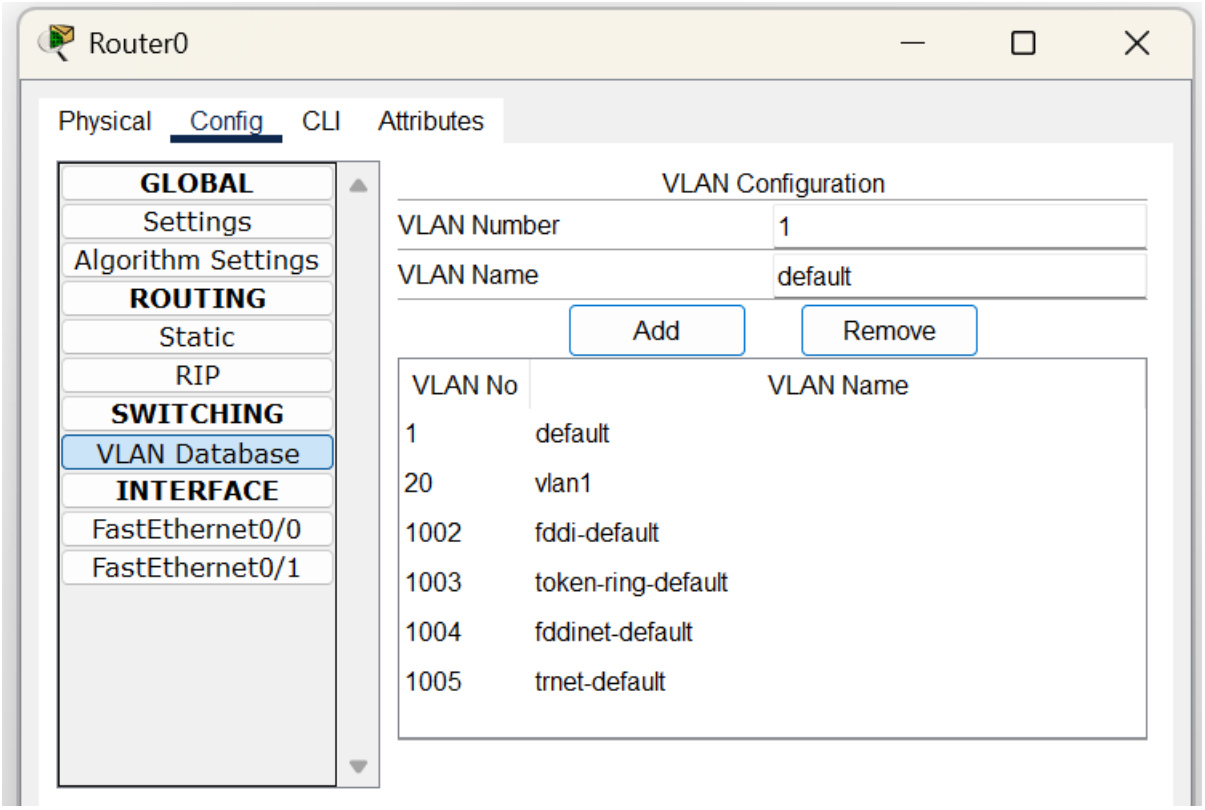


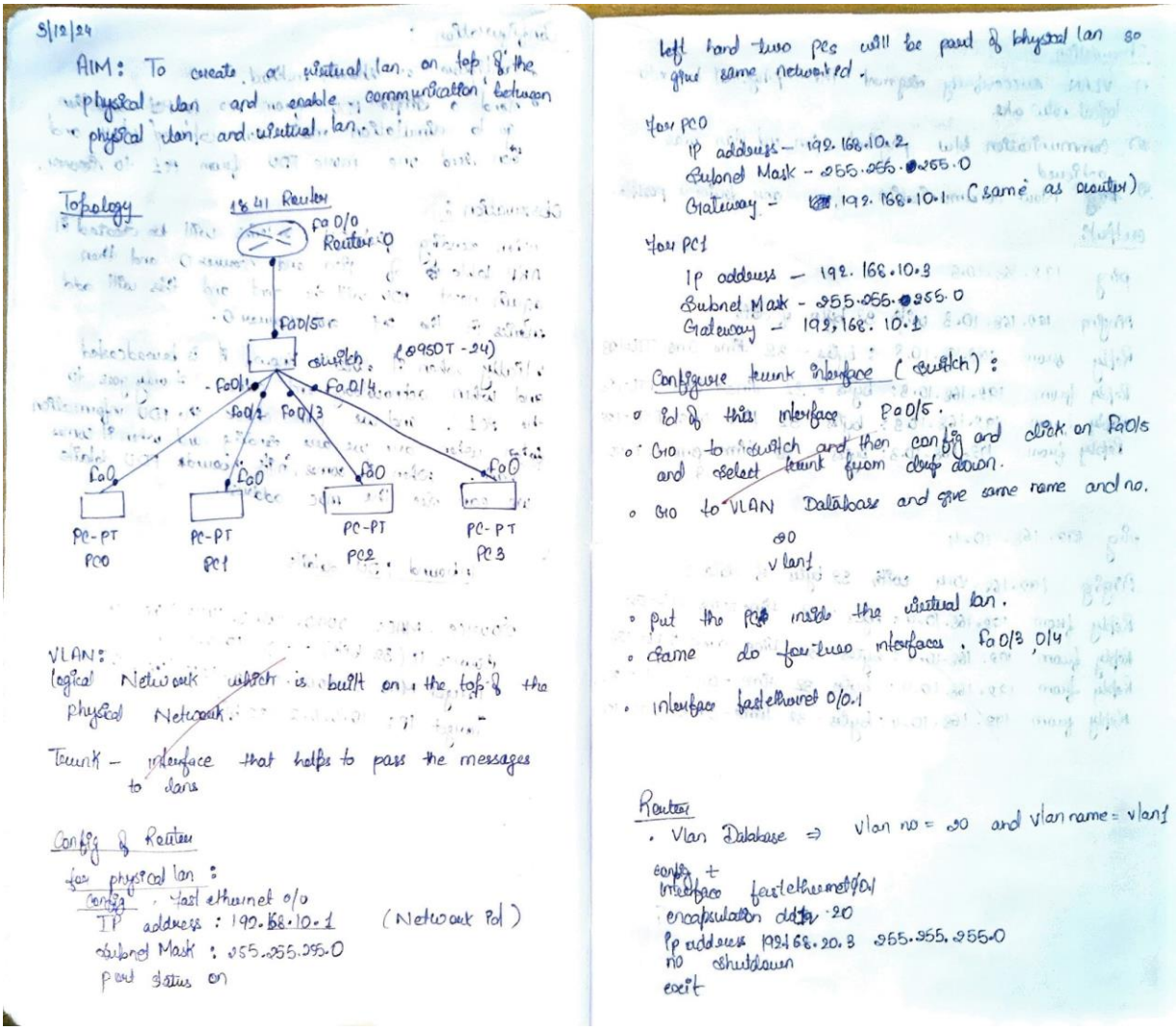
Figure 12.3: Router0 VLAN Database

```
Router(config)#interface FastEthernet0/0.1
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown
```

Figure 2: Router0, FE0/0.1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	Router0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 192.168.20.3

Pinging 192.168.20.3 with 32 bytes of data:

Reply from 192.168.20.3: bytes=32 time=2ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.20.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms
```



Observation:

1) VLAN successfully segment the physical lan into logical networks
2) Communication b/w physical lan and vlan was achieved.
3) ~~Ping~~ New communication from any device is possible.

Output:

ping 192.168.10.3

Pinging 192.168.10.3 with 32 bytes of data:
Reply from 192.168.10.3 : bytes = 32 time=0ms TTL=128
Reply from 192.168.10.3: bytes = 32 time=0ms TTL=128
Reply from 192.168.10.3: bytes = 32 time=0ms TTL=128
Reply from 192.168.10.3: bytes = 32 time=0ms TTL=128

ping 192.168.10.4

Pinging 192.168.10.4 with 32 bytes of data:
Reply from 192.168.10.4 : bytes = 32 time 5ms TTL=128
Reply from 192.168.10.4 : bytes = 32 time=0ms TTL=128
Reply from 192.168.10.4 : bytes = 32 time=0ms TTL=128
Reply from 192.168.10.4 : bytes = 32 time=0ms TTL=128

AIM: CRC (8) Error checking using any prog. lang.

Sender side
polynomial - $x^8+x^3+x^1+1$ (divisor)
dataword (i/p) - append $(8-1)$ 0 x^{8-1}
Binary division → obtain rem and append to dataword (codeword)

Recieve side
Read codeword and check error.

3/12/24

EXPERIMENT – 13

Write a program for error detecting code using CRC-CCITT (8-bits).

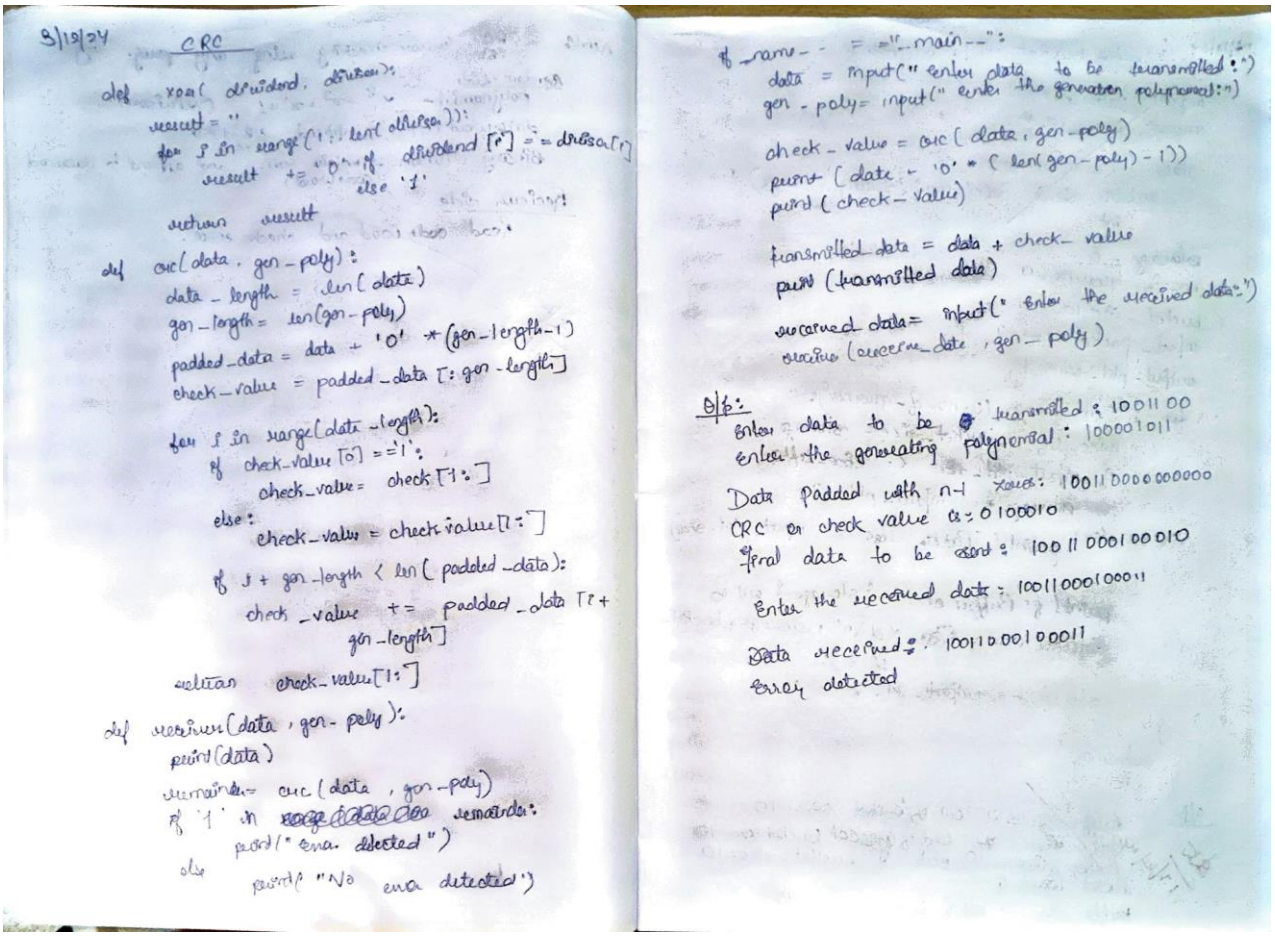
Code

```
def xor(dividend, divisor):
    """Perform XOR operation between dividend and divisor."""
    result = ''
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result


def crc(data, gen_poly):
    """Compute the CRC check value using CRC-CCITT (8-bit)."""
    data_length = len(data)
    gen_length = len(gen_poly)

    # Append n-1 zeros to the data
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            # XOR operation if the first bit is 1
            check_value = xor(check_value, gen_poly)
        else:
            # Retain original check value if first bit is 0
            check_value = check_value[1:]
        # Shift left and add the next data bit
        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]

    return check_value[1:]  # Remove the leading bit


def receiver(data, gen_poly):
    """Simulate the receiver side to check for errors."""
    print("\n-----------------------------")
    print("Data received:", data)

    # Perform CRC computation on received data
    remainder = crc(data, gen_poly)

    # Check if the remainder is all zeros
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")


if __name__ == "__main__":
    # Input data and generator polynomial
    data = input("Enter data to be transmitted: ")
```

```
    gen_poly = input("Enter the Generating
polynomial: ")


    # Compute CRC check value
    check_value = crc(data, gen_poly)
    print("\n----------------------------
--")
    print("Data padded with n-1 zeros:",
data + '0' * (len(gen_poly) - 1))
    print("CRC or Check value is:",
check_value)


    # Append check value to data for
transmission
    transmitted_data = data + check_value
    print("Final data to be sent:",
transmitted_data)
    print("----------------------------
\n")


    received_data = input("Enter the      #
Simulate the receiver side
received data: ")
    receiver(received_data, gen_poly)
```

Output

```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------

Enter the received data: 10011000100011

------------------------------
Data received: 10011000100011
Error detected
```



EXPERIMENT – 14

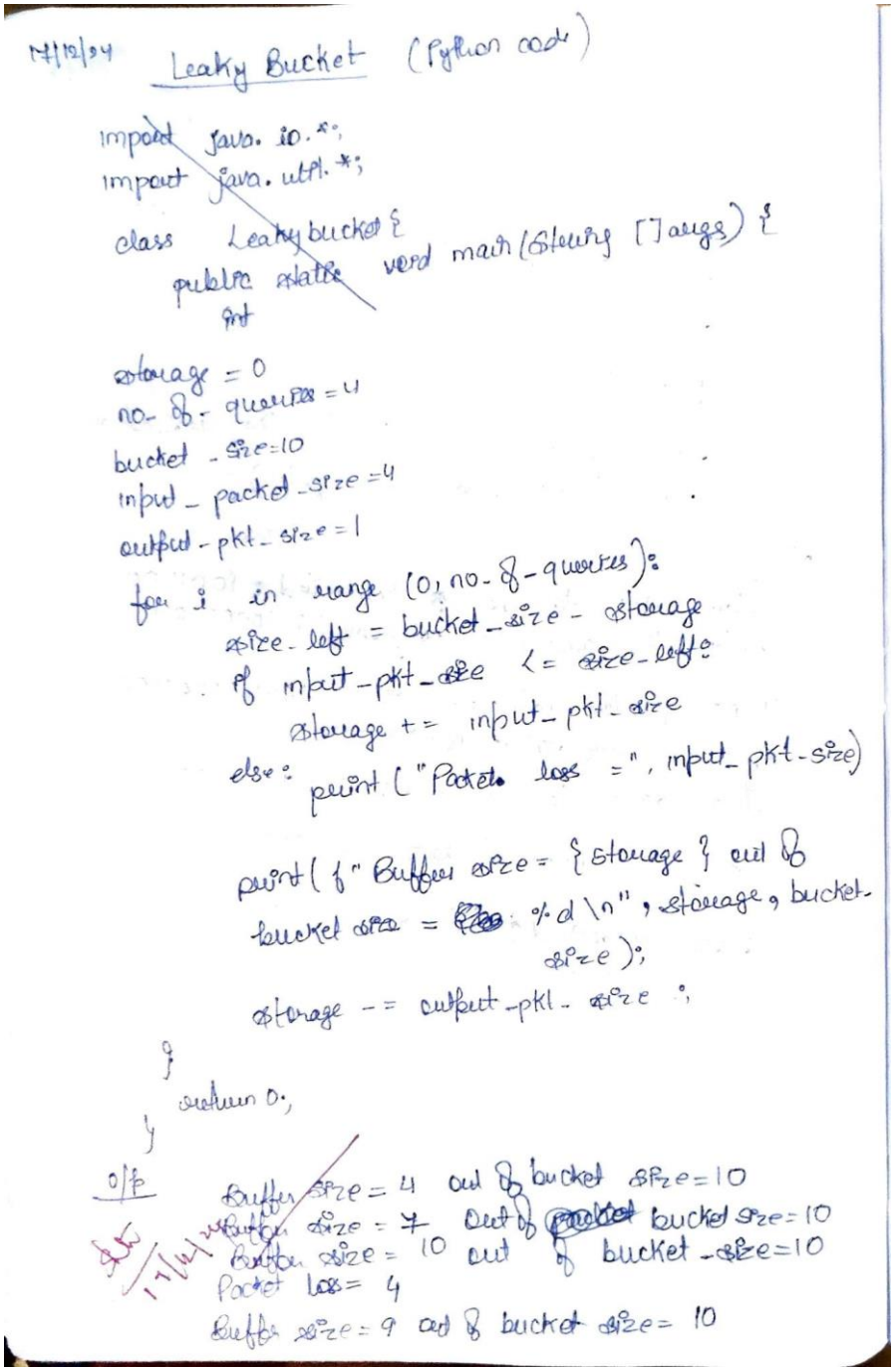
Write a program for congestion control using Leaky bucket algorithm.

Code

```
# Getting user inputs
storage = int(input("Enter initial packets in the bucket: "))
no_of_queries = int(input("Enter total no. of times bucket content is checked: "))
bucket_size = int(input("Enter total no. of packets that can be accommodated in the bucket: "))
input_pkt_size = int(input("Enter no. of packets that enters the bucket at a time: "))
output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time: "))


for i in range(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
```



17/12/24 Leaky Bucket (Python code)

```
import java.io.*;
import java.util.*;
class LeakyBucket {
    public static void main(String[] args) {
        int storage = 0
        no_of_queries = 4
        bucket_size = 10
        input_packet_size = 4
        output_pkt_size = 1
        for i in range(0, no_of_queries):
            size_left = bucket_size - storage
            if input_pkt_size <= size_left:
                storage += input_pkt_size
            else:
                print("Packet loss =", input_pkt_size)
            print(f"Buffer size = {storage} out of
            bucket size = %d \n", storage, bucket_
                                    size);
            storage -= output_pkt_size;
        }
        return 0;
    }
```

O/P

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

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.

Code: Client.py

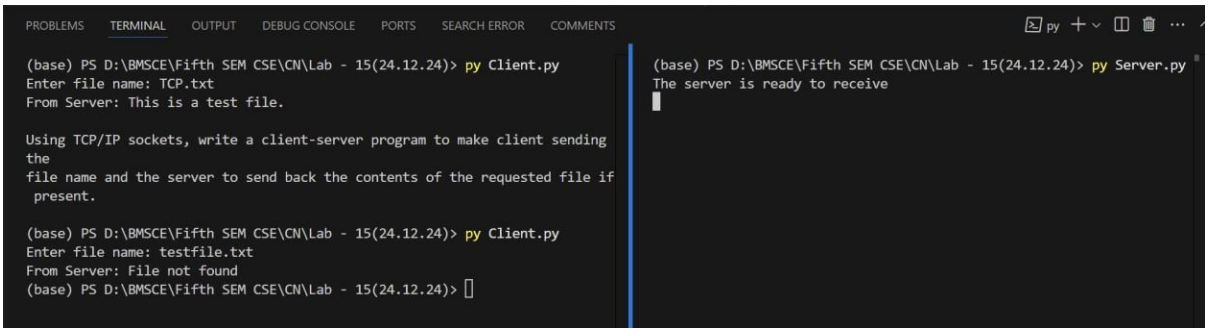
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))  # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

Code: Server.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number to listen on

# Create TCP socket
```

```
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))  # Bind socket to the address and port
serverSocket.listen(1)  # Listen for 1 connection
print("The server is ready to receive")

while True:
    # Accept a connection
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Try opening the file
    try:
        file = open(sentence, "r")  # Open file in read mode
        fileContents = file.read(1024)  # Read file content (up to 1024 bytes)
        connectionSocket.send(fileContents.encode())  # Send file contents to client
        file.close()
    except FileNotFoundError:
        # Send error message if file not found
        connectionSocket.send("File not found".encode())

    # Close the connection
    connectionSocket.close()
```

Output



24/10/24 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.

client.py

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

server.py

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

Output

> py server.py
The server is ready to receive

> py client.py
Enter file name: TCP.txt
From Server: This is a test file.

> py client.py
Enter file name: testfile.txt
From Server: File not found

EXPERIMENT – 16

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: ClientUDP.py

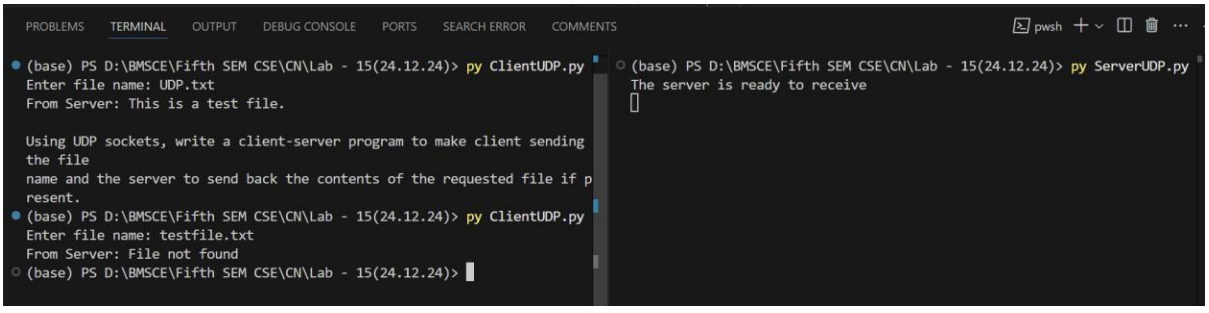
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create UDP socket
clientSocket = socket(AF_INET, SOCK_DGRAM)

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send the file name to the server using UDP
clientSocket.sendto(sentence.encode("utf-8"), (serverName, serverPort))

# Receive file contents from the server
fileContents, serverAddress = clientSocket.recvfrom(2048)

# Print the file contents received from the server
print("From Server:", fileContents.decode())

# Close the UDP socket
clientSocket.close()
```

Code: ServerUDP.py

```
from socket import *
serverPort = 12000  # Port number to listen on

# Create UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
```

```
serverSocket.bind(("127.0.0.1", serverPort))  # Bind the socket to the server address and port

print("The server is ready to receive")

while True:
    # Receive file name from the client
    sentence, clientAddress = serverSocket.recvfrom(2048)

    # Try opening the file
    try:
        file = open(sentence.decode(), "r")  # Open file in read mode
        fileContents = file.read(2048)  # Read file content (up to 2048 bytes)
        serverSocket.sendto(fileContents.encode("utf-8"), clientAddress)  # Send file contents to client
        file.close()
    except FileNotFoundError:
        # Send error message if file not found
        serverSocket.sendto("File not found".encode("utf-8"), clientAddress)
```

Output



Experiment - 16

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.

* 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(bytes(sentence, 'utf-8'), (
        serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom
                                        (2048)
print("from Server:", fileContents)
clientSocket.close()
```

* Server UDP.py

```
from socket import *;
serverPort = 12000
serverSocket = socket(AF_NET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    file = open(sentence, "r")
    l = file.read(2048)
    serverSocket.sendto(bytes(l, "utf8"),clientAddress)
    print("Sent back to client", l)
file.close()
```

Output

```
> py serverUDP.py
The server is ready to receive
```

```
> py clientUDP.py
enter file name: UDP.txt
from Server: This is a test file.
```

```
> py clientUDP.py
enter file name: testfile.txt
from server: file not found
```