VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Jeevan A (1BM22CS119)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

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 **Jeevan A (1BM22CS119),** 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.

Rajeshwari Madli Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Github Link:
Jeevan-017/CN-LAB-1BM22CS119

LAB PROGRAM – 1

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

01/10/2024

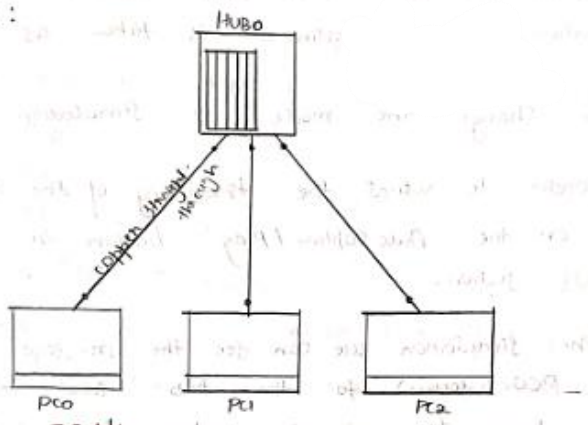
Computer Networks Lab

LAB - 1

AIM :

To demonstrate the transmission of a simple PDU between two devices connected using a hub and a switch

TOPOLOGY :



STEPS :-

Topology formation

1. Click on the End Devices, and select three generic PC's - PC0, PC1, PC2 and place it in the workspace
2. Select a Generic Hub and place it in the workspace
3. Connect the devices to the hub with copper ~~cross-over~~ straight-through connection and turn on the devices.
4. This forms a topology containing a central hub and three generic devices.
5. Configure the IP Address of each device

- For each device set the following IP Address

PC0 - 10.0.0.1
PC1 - 10.0.0.2
PC2 - 10.0.0.3

Simple PDU Transmission :-

6. To send a simple message from source to destination ; select a simple PDU option and first click on the source , source here is considered as PC1 and then click on the destination , here destination is taken as PC2 .

7. Then change the mode to simulation mode

8. In order to visual the travelling of the message click on the Auto Capture / Play button in Play controls section.

9. In the simulation we can see the message passing from PC0 (source) to the hub and then from hub to all the other devices in the topology (∵ Hub does not have intelligence to identify the destination system , so it just forwards to all the devices in the topology).

10. Here the message will be received by PC2 only and rejected by PC1

PDU transmission between two devices connected using a switch

A. Topology Formation :-

1. Select three devices, generic PC's and a switch and place them on the workspace
2. Connect the devices to the switch using copper straight-through connection.

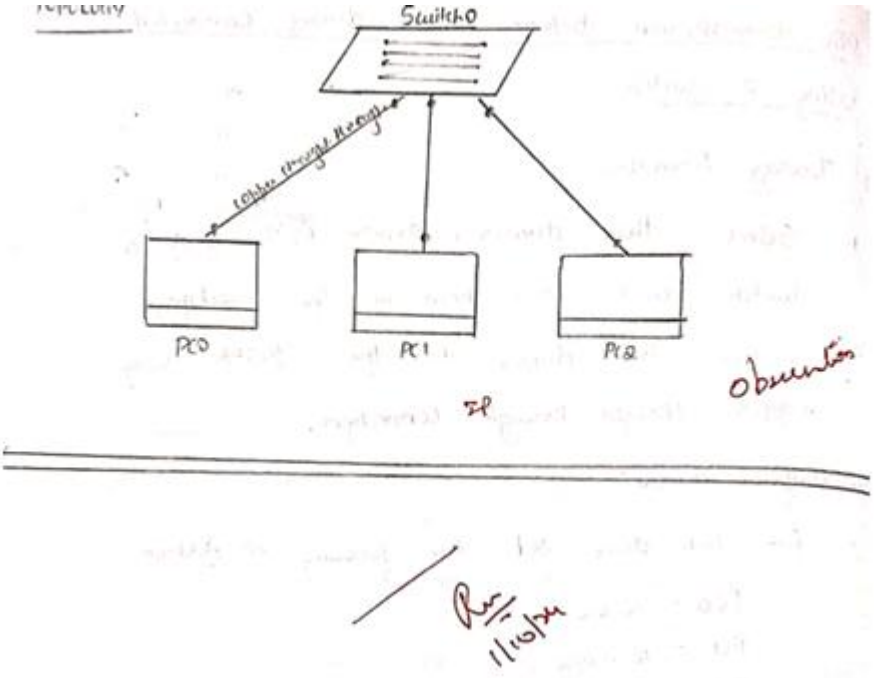
B. Configure devices :-

1. For each device set the following IP Addresses

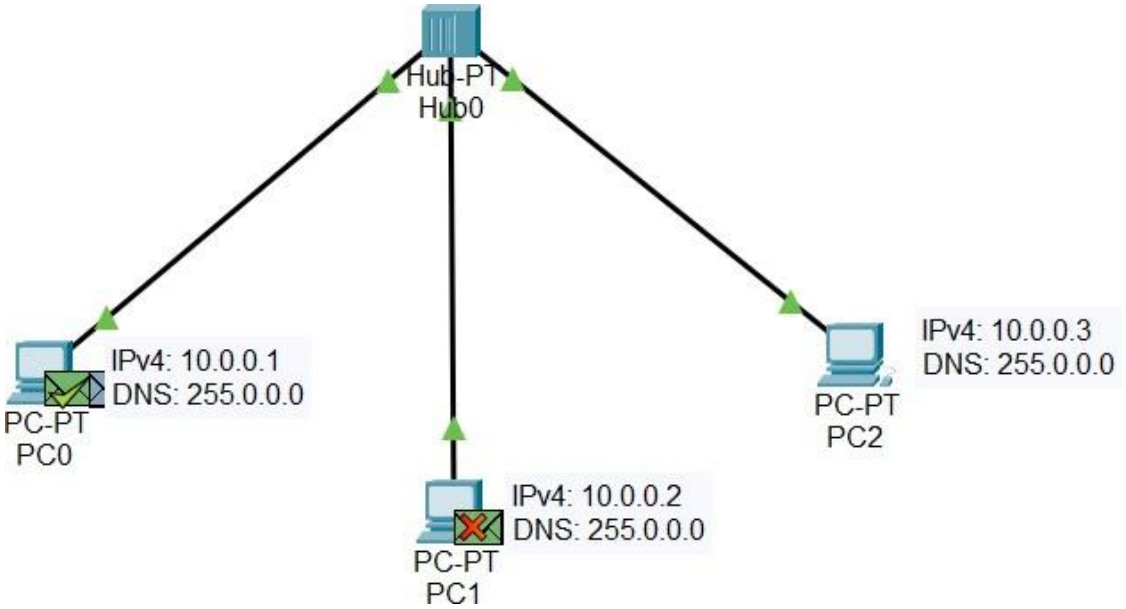
   PC0 - 10.0.0.1
   PC1 - 10.0.0.2
   PC2 - 10.0.0.3

C. Simple PDU Transmission :-

1. Select simple PDU option and first click on the source, here source is considered as PC0,
   and then click on the destination, here dest is considered as PC2.
2. Change the mode to simulation mode
3. Visualize the simulation using Auto Capture / Play in Play controls section
4. In the simulation we can see that the message is passed from source (PC0) to switch and then from switch it passes the message only to the destination system (PC2)


Switch-0
PC0
PC1
PC2
IP
observation

OUTPUT:




```
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
```

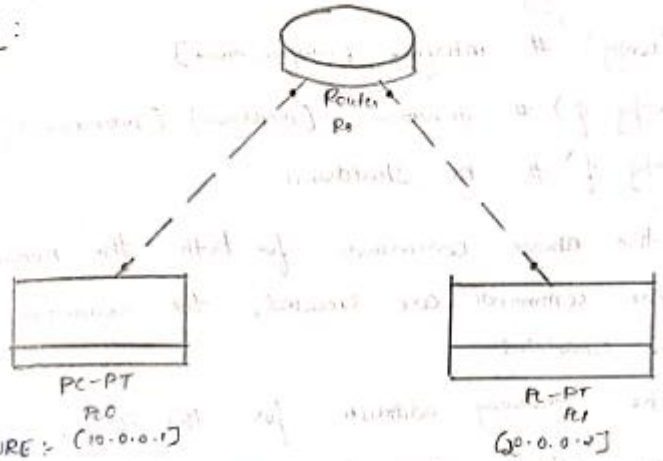
LAB PROGRAM – 2

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

LAB - 02

AIM: Configure IP address to routers and explore ping responses.

TOPOLOGY:



PROCEDURE:

1. Select two generic devices and a Generic router, place them into the workspace
2. Connect the PC's to the router using Copper cross-over connection line.
3. This created a topology

Configuring devices and routers:-

1. For PC0 and PC1, go to config section and set the following IP Address

   PC0 :- 10.0.0.1

   PC1 :- 20.0.0.2

   Set Different Network Id's for both the devices to represent two networks.

2. To configure the router, click on router and go to CLI section and execute the following commands:-

Router > enable

Router # config t

Router (config) # interface [interface_name]

Router (config-if) # ip address [ipaddress] [subnetmask]

Router (config-if) # no shutdown

3. Follow the above commands for both the networks

4. Once the commands are executed, the connection will be established

5. Set the Gateway addresses for the PC's.

6. We can explore ping command by Executing the following command in command prompt of PC device

PC> ping 20.0.0.2

Observations:-

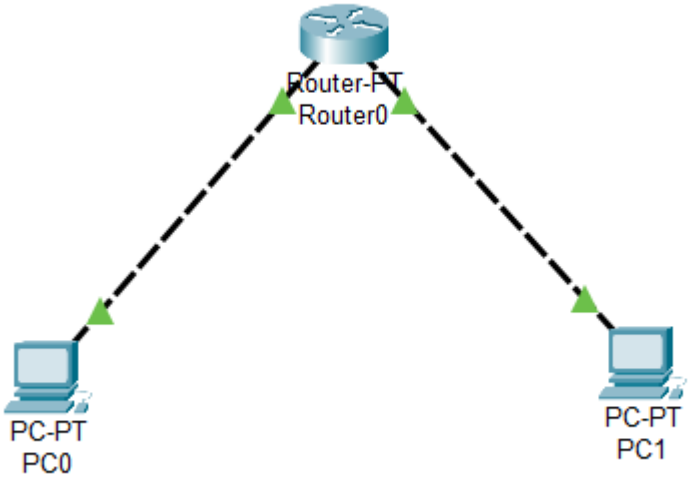
The execution of ping command resulted in the following response

Reply from 20.0.0.2 : bytes = 32 time = 0ms TTL=127

Reply from 20.0.0.2 : bytes = 32 time = 0ms TTL=128

Reply from 20.0.0.2 : bytes = 32 time = 0ms TTL=129

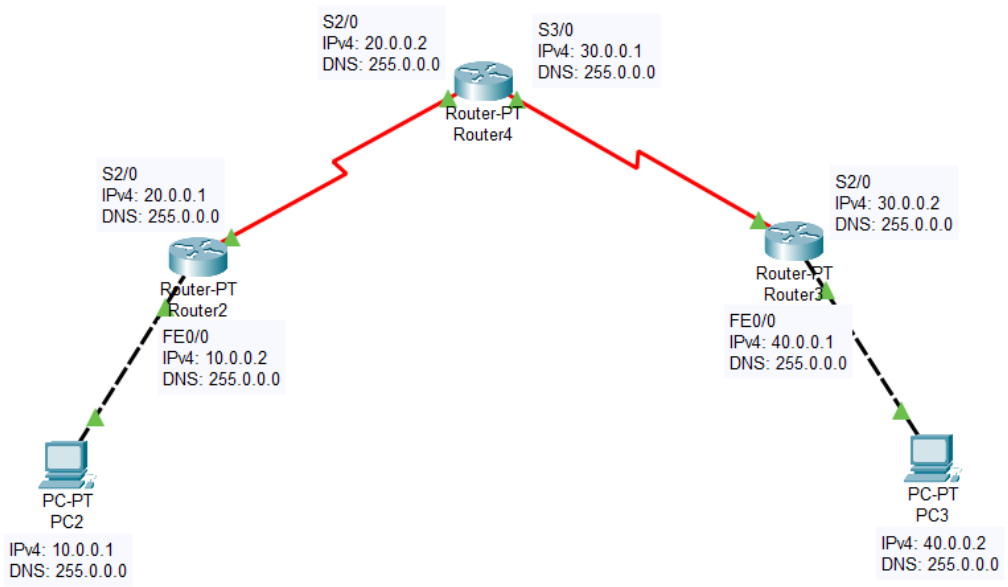
OUTPUT:




```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127
Reply from 20.0.0.2: bytes=32 time=1ms TTL=127
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Reply from 20.0.0.2: bytes=32 time<1ms TTL=127
Reply from 20.0.0.2: bytes=32 time=1ms TTL=127
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127
Reply from 20.0.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

LAB PROGRAM – 3

AIM: Configure static route to the Router.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC2	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC3	PC2	ICMP		64.457	N	1	(edit)	(delete)

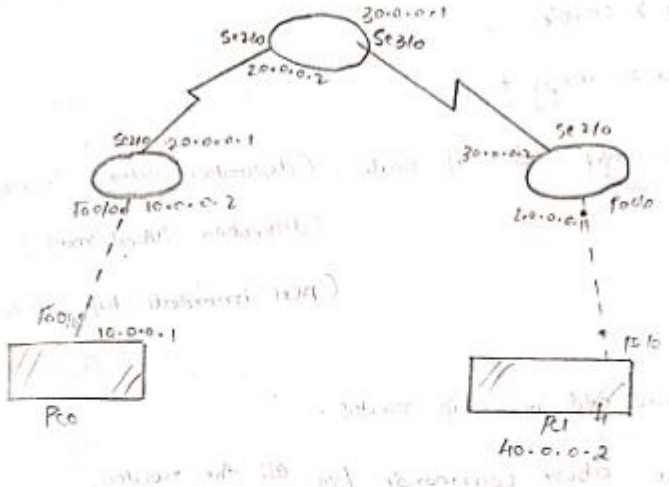
```
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=44ms TTL=125
Reply from 40.0.0.2: bytes=32 time=24ms TTL=125
Reply from 40.0.0.2: bytes=32 time=19ms TTL=125
Reply from 40.0.0.2: bytes=32 time=2ms 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 = 44ms, Average = 22ms
```

LAB – 03

15/10/24

AIM: To demonstrate the configuration of default routes to the router

TOPOLOGY:



PROCEDURE:-

1. Select devices and routers and place them on the workspace and connect them to form a network.
2. Set IP address for the device

To configure router :-

Run the below command :-

Router# show ip route

C 10.0.0.0 /8 is directly connected, FastEthernet0/0

C 20.0.0.0 /8 is directly connected, Serial2/0

=) This will show the list of networks to which the

Router is connected.

- In order to configure the routes to other networks, the commands are:-

In Router 0 CLI,

- Router > enable
- Router # config t
- Router (config) # ip route (destination network IP address) (destination subnet mask) (next immediate hop ip address)

c) This will add new ip routes.

Run the above commands for all the routers.

IP add:-

Router R0 - 10.0.0.1

Router 0 - 10.0.0.2
20.0.0.1

Router 1 - 20.0.0.2
30.0.0.1

Router 2 - 30.0.0.2
40.0.0.1

Server R1 - 40.0.0.2

> In PC-PT - R0 > Desktop > Command Prompt > ping
> ping 40.0.0.2 (Destination IP address)

OUTPUT:-

> Pinging 40.0.0.2 with 32 bytes of data

Request Timed out

Reply from 40.0.0.2 : byte = 32 time = 7ms TTL = 125

Reply from 40.0.0.2 : byte = 32 time = 5ms TTL = 125

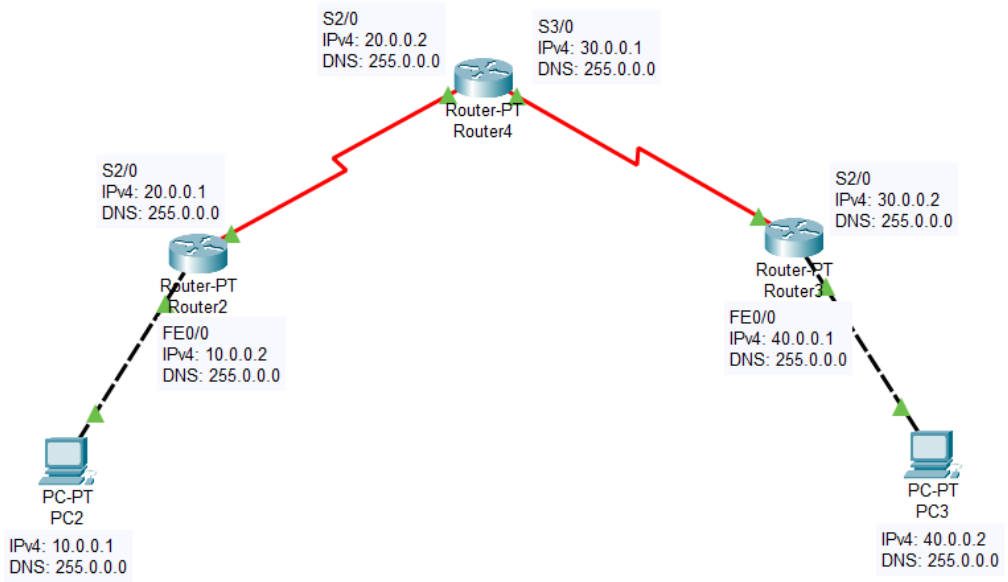
Ping statistics for 40.0.0.2

Packets sent = 4, Received = 3, Lost = 1 (25% loss)

15/10/24

LAB PROGRAM – 4(A)

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

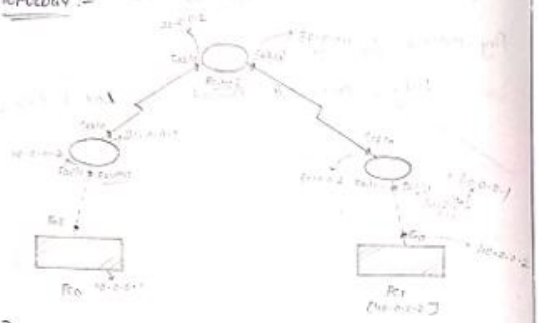



```
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=34ms TTL=125
Reply from 40.0.0.2: bytes=32 time=33ms TTL=125
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125
Reply from 40.0.0.2: bytes=32 time=33ms 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 = 30ms, Maximum = 34ms, Average = 32ms
```

LAB - 04

AIM :- To configure default and static routes to a connection of routers.

TOPOLOGY :-



PROCEDURE :-

1. Create a new file and build the Topology as above and connect the devices and routers
2. Set the ip addresses

PC0 - 10.0.0.1

Router 0 - 10.0.0.2
20.0.0.1

Router 1 - 20.0.0.2
30.0.0.1

Router 2 - 30.0.0.2
40.0.0.1

PC1 - 40.0.0.2

→ Now configure default routes to Router 0 and Router 2

1. Go to Router 0 > CLI
2. Run the below commands :

Router> enable
Router # config t
Router (config) # ip route 0.0.0.0 0.0.0.0 (next hop IP address)

→ Then configure static route to Router 1

1. Go to Router > CLI
2. Run the below commands :-

Router (config)# ip route (destination network ip) 255.0.0.0 [next hop]

Now for Router 0 & Router 2 :-

R0 # ip route 0.0.0.0 0.0.0.0 20.0.0.2

R2 # ip route 0.0.0.0 0.0.0.0 30.0.0.1

For Router 1 :-

R1 # ip route 40.0.0.0 255.0.0.0 30.0.0.2

R1 # ip route 10.0.0.0 255.0.0.0 20.0.0.1

OBSERVATION

To ensure that the routing was successful, go to

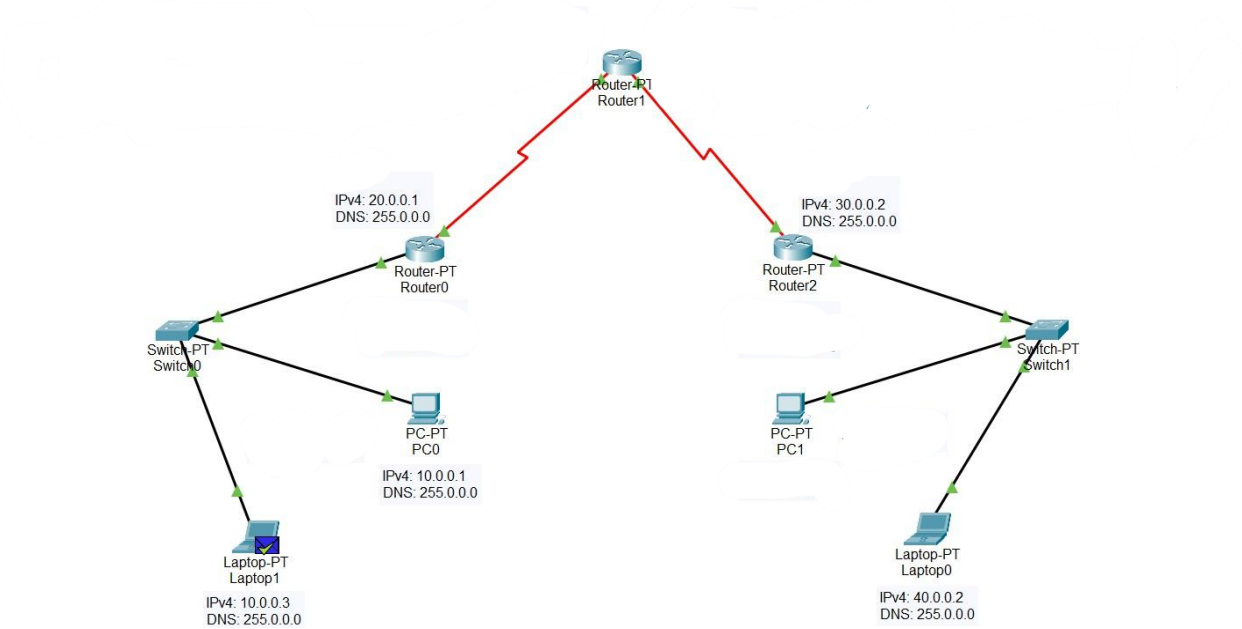
PC0 > Command Prompt

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 = 9ms TTL = 1
Reply from 40.0.0.2 : bytes = 32 time = 15ms TTL = 1
Ping statistics for 40.0.0.2 :

LAB PROGRAM – 4(B)

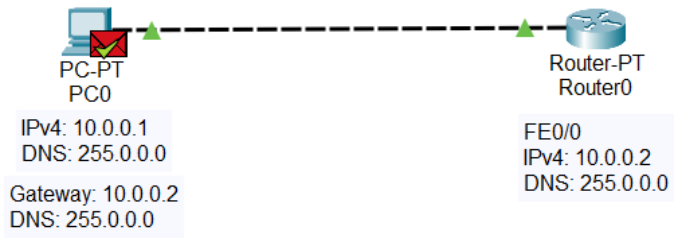
AIM: Configure default route, static route to the Router, inclusive switches.



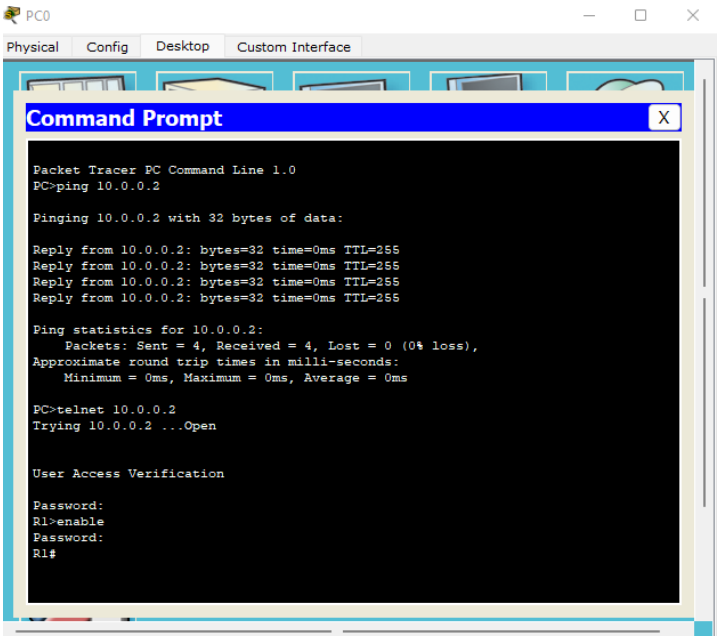

```
C:\>ping 40.0.0.3

Pinging 40.0.0.3 with 32 bytes of data:

Reply from 40.0.0.3: bytes=32 time=35ms TTL=125
Reply from 40.0.0.3: bytes=32 time=37ms TTL=125
Reply from 40.0.0.3: bytes=32 time=24ms TTL=125
Reply from 40.0.0.3: bytes=32 time=38ms 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 = 24ms, Maximum = 38ms, Average = 33ms
```

LAB PROGRAM – 5

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



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Router0	ICMP		0.000	N	0	(edit)	



LAB-05

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

Topology:-

Procedure:-

1. Pick a generic PC and a router and place them on the workplace.
2. Configure PC0 IP address : 10.0.0.1
   and Router0 IP address : 10.0.0.2
3. In the CLI of Router0 type the following commands :-

```
Router (config-if)# hostname R1
R1 (config)# enable secret P0
R1 (config)# line vty 0 5
R1 (config-line)# login
R1 (config-line)# password P1      (pw not for login)
R1 (config-line)# exit
R1 #
```

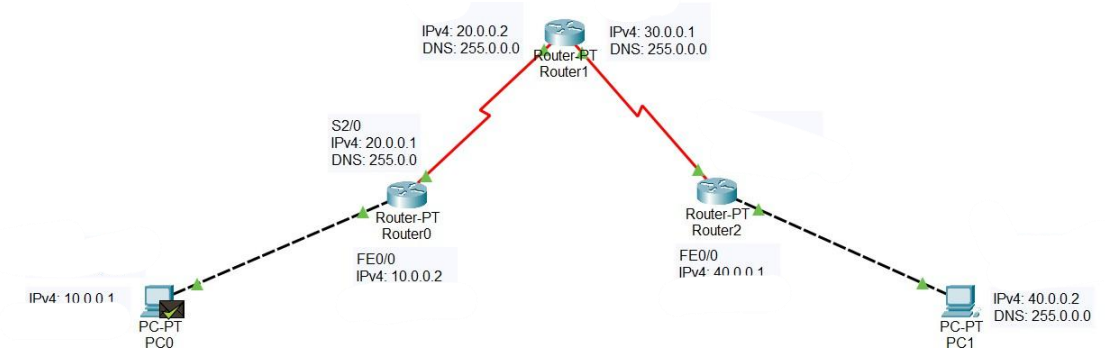
Observation:-

```
PC > telnet 10.0.0.2
Trying 10.0.0.2 ..... Open

User Access Verification
Password : P1
R1 > enable
Password : P0
R1#
```

8/12/24

LAB PROGRAM – 6

Demonstrate the TTL/ Life of a Packet.



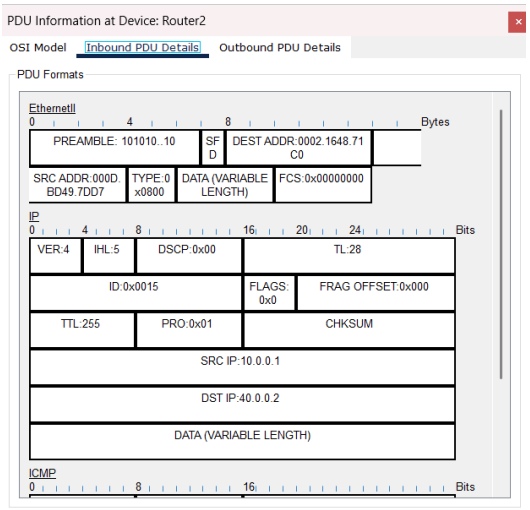


Figure 6.1: Inbound PDU, Router2

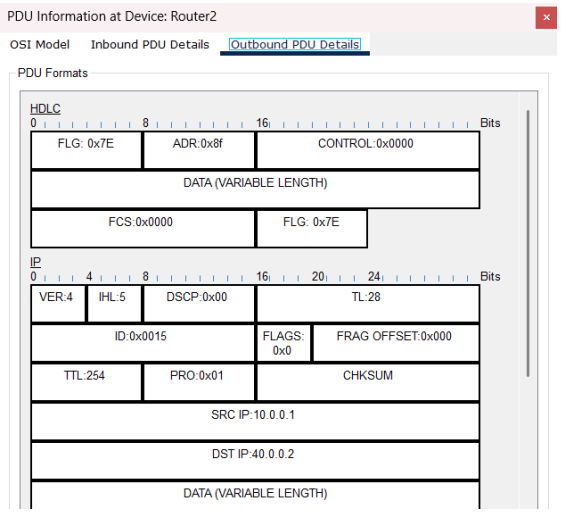


Figure 6.2: Outbound PDU, Router2

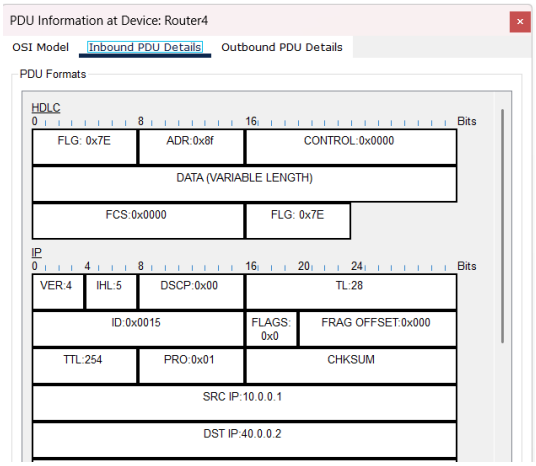


Figure 6.3: Inbound PDU, Router4

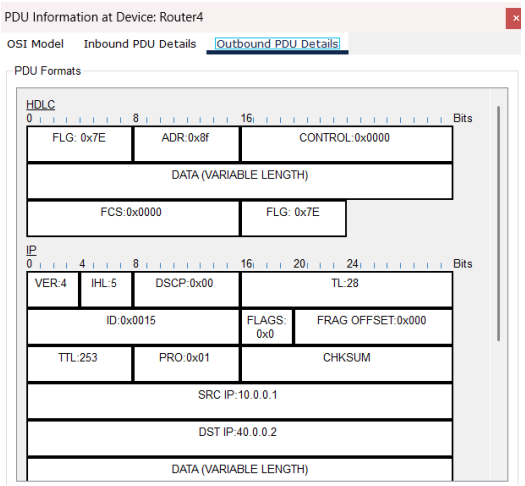


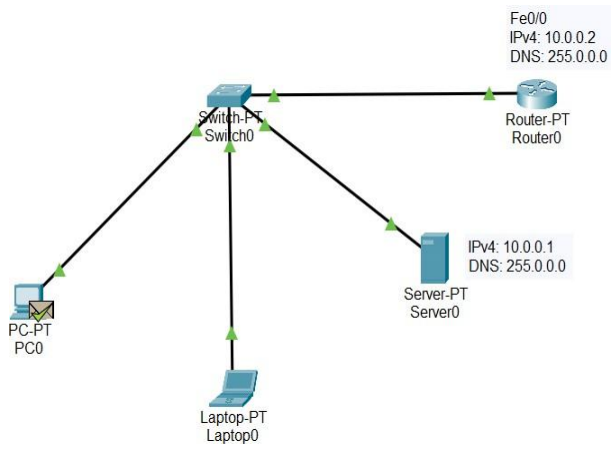
Figure 6.4: Outbound PDU, Rout

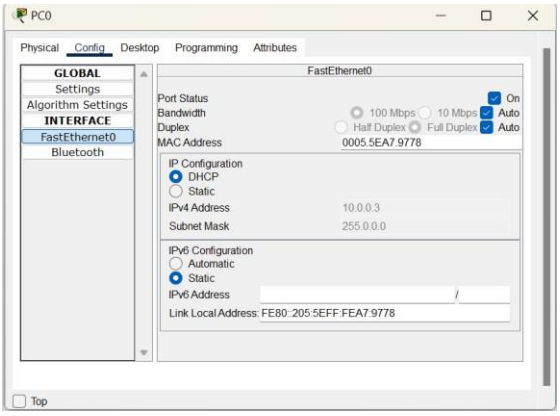
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	

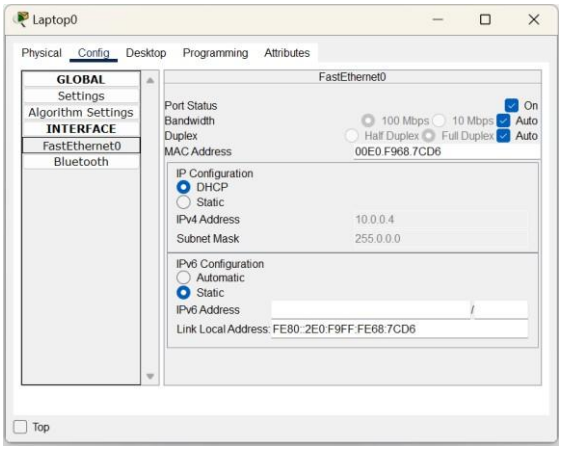
```
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=72ms TTL=123
Reply from 40.0.0.2: bytes=32 time=53ms TTL=123
Reply from 40.0.0.2: bytes=32 time=55ms TTL=123
Reply from 40.0.0.2: bytes=32 time=69ms 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 = 53ms, Maximum = 72ms, Average = 62ms
```

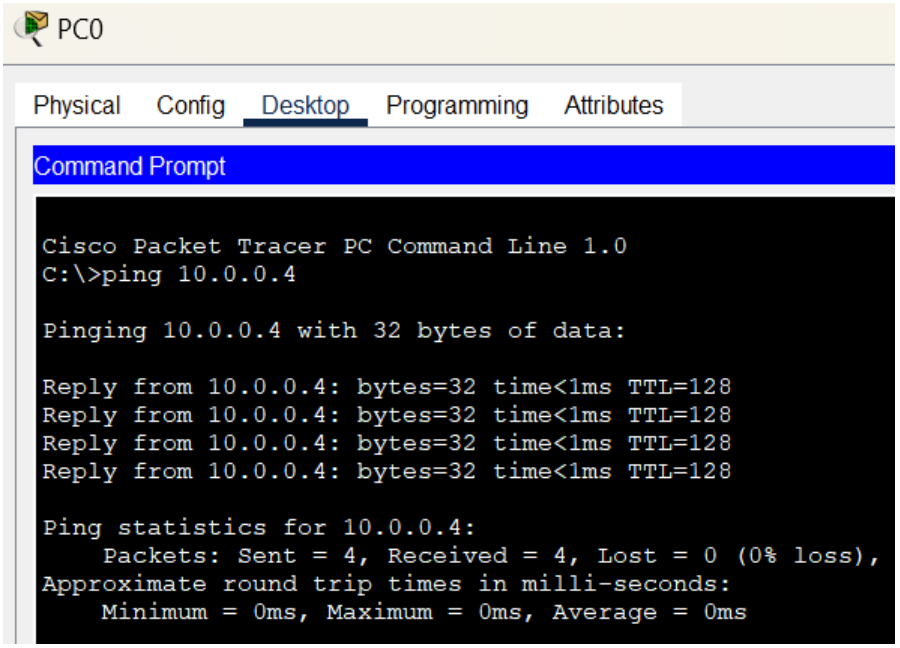
LAB PROGRAM – 7(A)

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









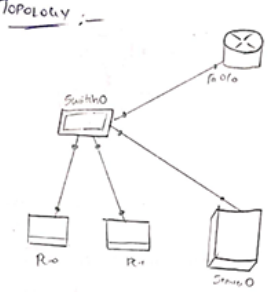
LAB-06

AIM:-

A. To configure IP Addresses of the host using DHCP server present within the LAN

B. To configure IP Addresses of the host present in the different LAN.

TOPOLOGY :-



PROCEDURE :-

1. Select 2 Generic PC's, 1 Generic Server and a Switch, router and place it on the workplace
2. Connect the network as shown in the topology
3. In the services of the server, enable DHCP, add a new servernpool with the gateway (10.0.0.1) and DNS server
4. Here Server0 is considered as the DHCP server
5. Shift the PC's from static to DHCP mode.

Observation

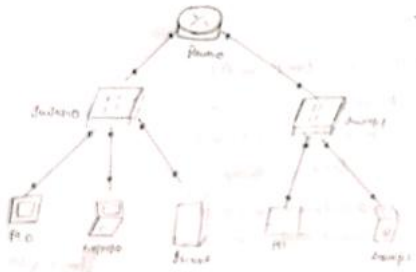
> ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data

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 = 0ms TTL = 128

> DHCP server (Server0) was able to dynamically assign the PC's in the Green LAN from the pool of IP's in the server pool.

PART-B

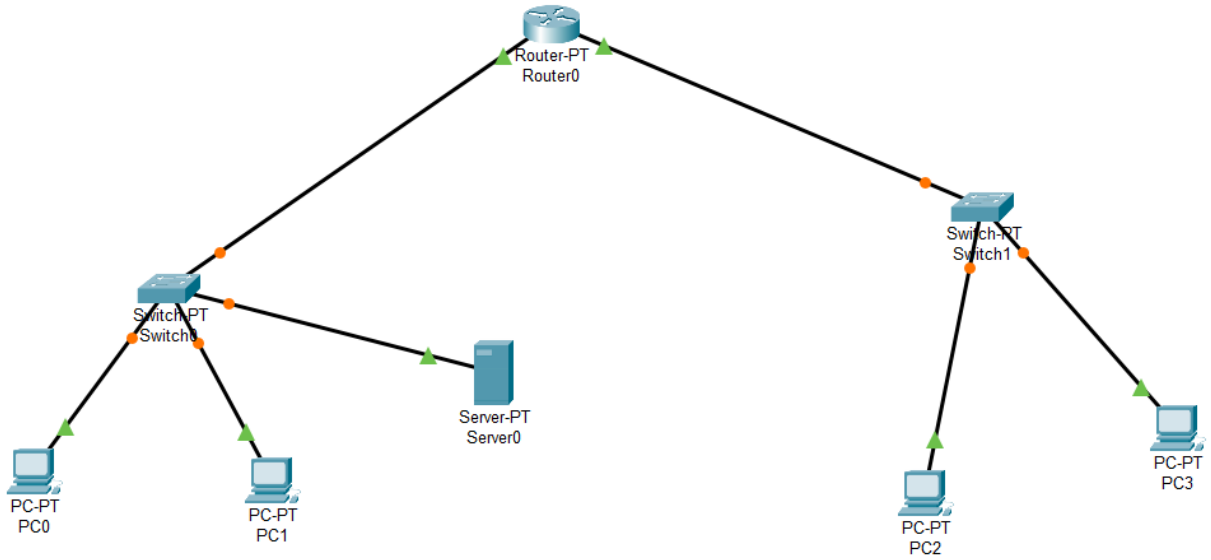


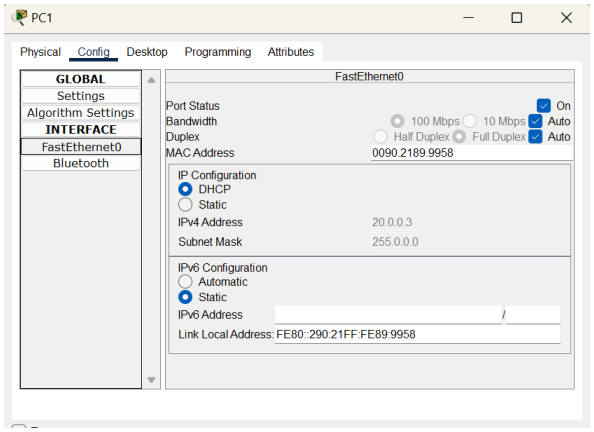
Observation

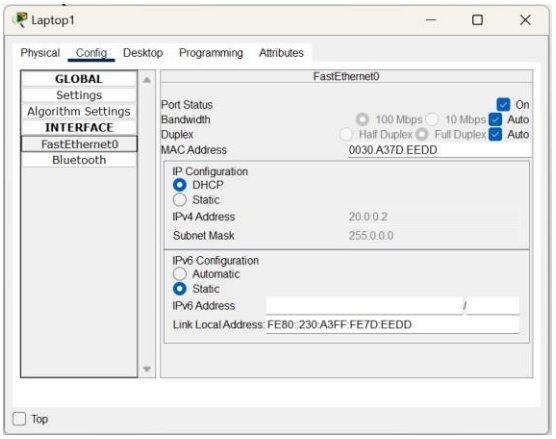
- Connections are made as per the topology shown
- For the internet network,
  - > IP helper 10.0.0.1
    in FastEthernet port
  - > IP helper 20.0.0.1

LAB PROGRAM – 7(B)

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







Config server0 : At services , using DHCP protocol :
Service is 'On'

→ Pool Name : ServerPool2
Default Gateway : 20.0.0.1
DNS Server : 10.0.0.1
Start IP Address : 20.0.0.0
Subnet Mask : 255.0.0.0.

→ Save the ServerPool2

→ Pool Name : ServerPool1
Default Gateway : 20.0.0.2
DNS Server : 10.0.0.1
Start IP Address : 10.0.0.0

→ Save the ServerPool1

⇒ Observations :-

- Now clicking on DHCP in PC1 :
  IP Address : 20.0.0.2
  Subnet Mask : 255.0.0.0
  On clicking DHCP in laptop1 :
  IP Address : 20.0.0.3
  Subnet Mask : 255.0.0.0

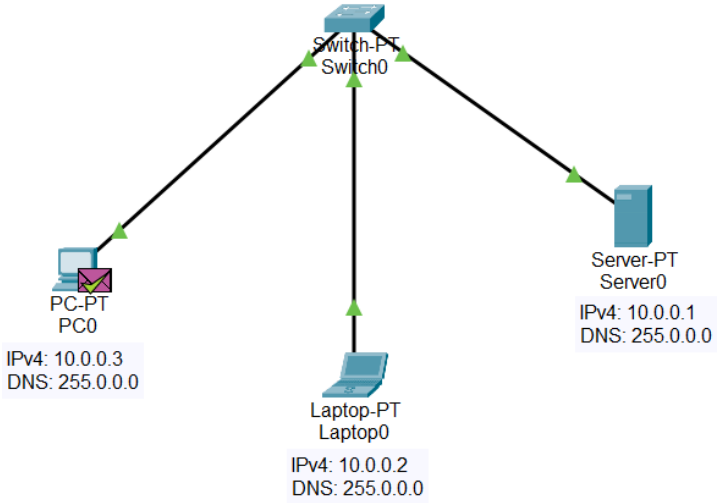
- In PC0 > ping 20.0.0.2

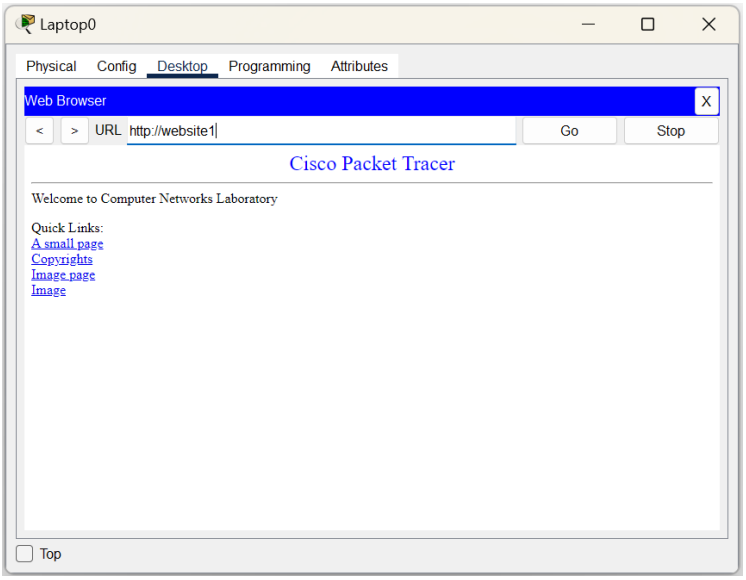
  Pinging 20.0.0.2 with 32 bytes of data.
  Reply from 20.0.0.2 : bytes = 32 time<1ms TTL = 125
  ⋮
  Ping statistics for 20.0.0.2
  Packets sent = 4, Received = 4 Lost = 0 (0% Loss)

3/12/24

LAB PROGRAM – 8

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

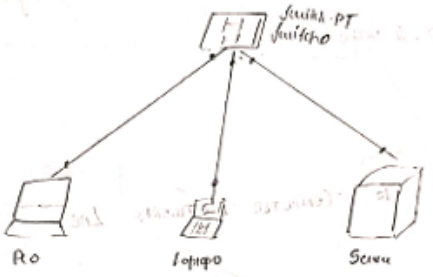




LAB - 07

AIM:- To configure DNS server to demonstrate the mapping of IP Address and domain Name

TOPOLOGY:-



PROCEDURE:-

- Connections are made according to the topology above
- At Server0 :

  In FastEthernet 0 : IP address : 10.0.0.1

  In Services > Use DHCP Protocol

  Default Gateway : 0.0.0.0

  DNS Server : 10.0.0.1

  > In DNS services :

  Name : Website

  Address : 10.0.0.1

  > In HTTP :- Edit index.html as per the requirements
- In PC0 : click on DHCP, the IP address get configured automatically
- In Laptop0 : On clicking DHCP, the address gets configured
- Config are done.

Observation:-

- Using any end device, click on "web browser" in desktop menu.
- On typing website, the domain name, index.html get displayed

  URL : http://website

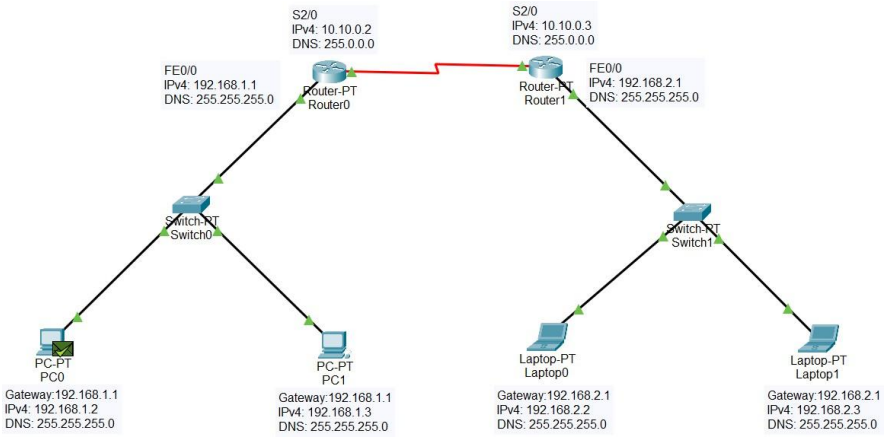
  Sample Output:-

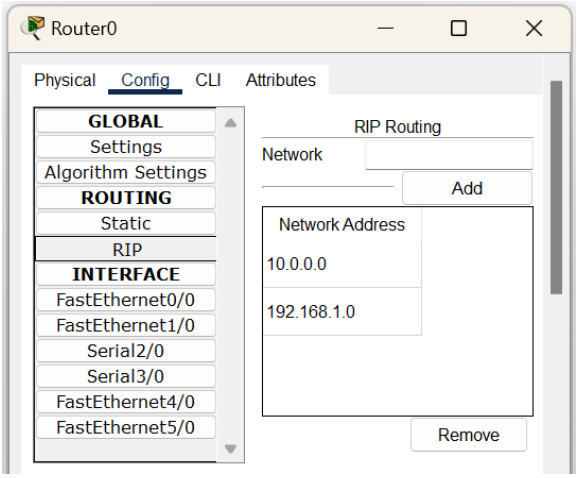
  WELCOME TO COMPUTER NETWORKS LAB

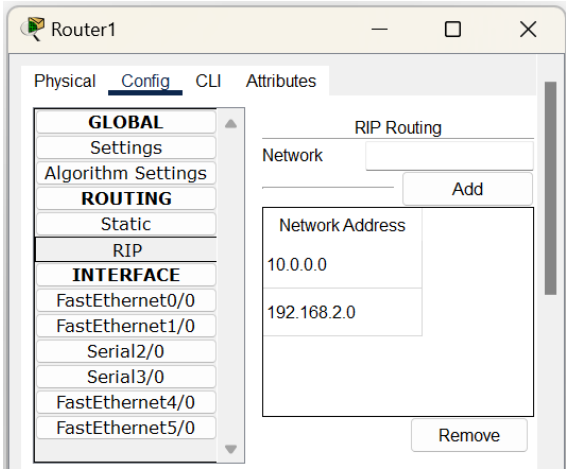
3/12/24

LAB PROGRAM – 9

To Configure RIP routing protocol in Routers.





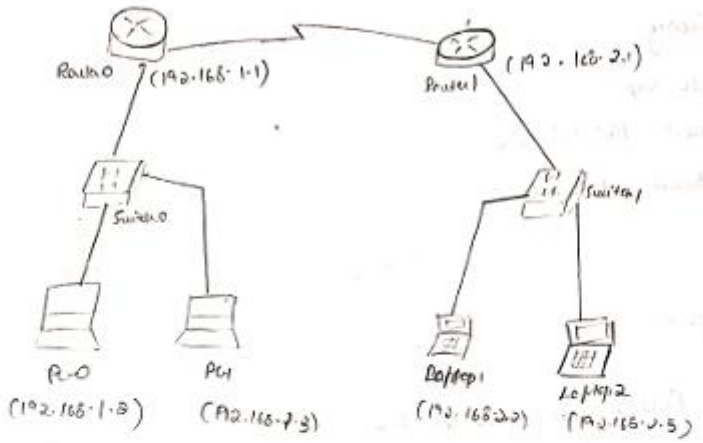



```
C:\>ping 192.168.2.3

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
```

LABORATORY - 08

AIM :- To configure RIP Routing protocol in routers.

TOPOLOGY :-



PROCEDURE :-

- Connections are made as per the topology above
- Config end devices with the following IP address

PC0 → 192.168.1.2 (IP)

192.168.1.1 (Gateway)

Laptop0 → 192.168.2.2 (IP)

192.168.2.1 (Gateway)

Router 0 :

In Fa0/0 : IP address : 192.168.1.1

In Se2/0 : IP address : 10.10.0.2

Clock Rate : 64000

=) RIP Routing :-

Router (Config) # router rip

Router (config-router) # network 192.168.1.0

Router (config-router) # network 10.0.0.0

Router (config-route) # end

- Router1 :-

In Fa0/0 : IP Address : 192.168.2.1

In Se 2/0 : IP Address : 10.10.0.3

Clock Rate : 2000000

RIP Routing :

router rip

network 192.168.2.0

network 10.0.0.0

end

Observation :-

In PC0 > ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data :

Reply from 192.168.2.2 : bytes = 32 Time = 7ms TTL = 126

:

Ping Statistics :-

Packets : Sent = 4, Recieved = 4, Lost = 0 (0% Loss)

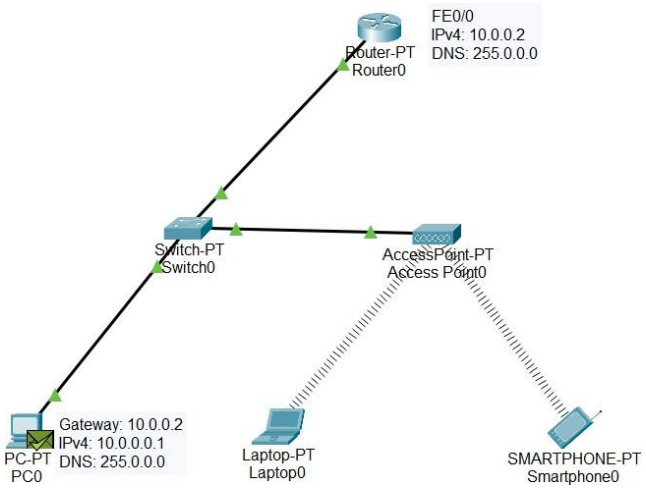
Approx round trip time in ms :

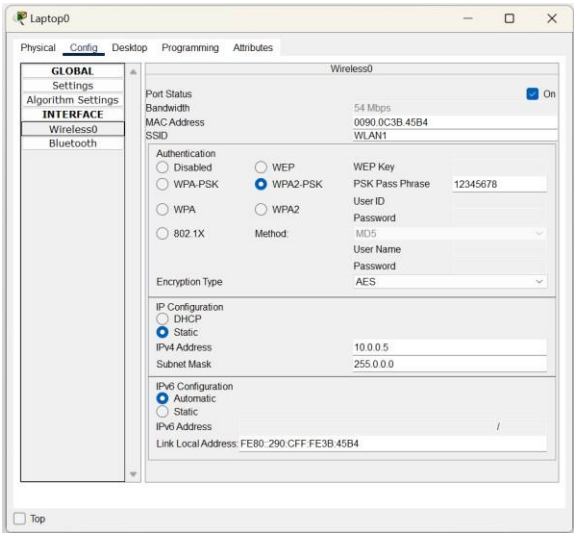
Min : 6ms , Max = 7ms , Avg = 6ms

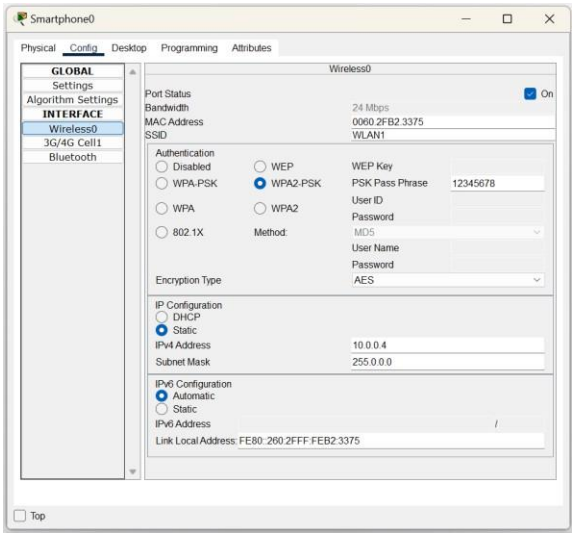
3/12/24

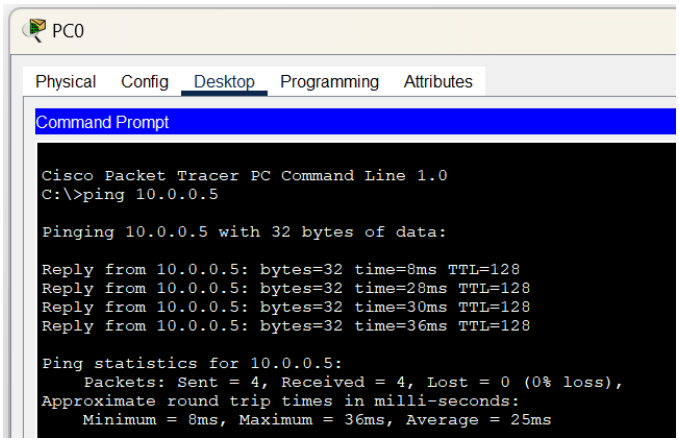
LAB PROGRAM – 10

To demonstrate communication between two devices using a wireless LAN.





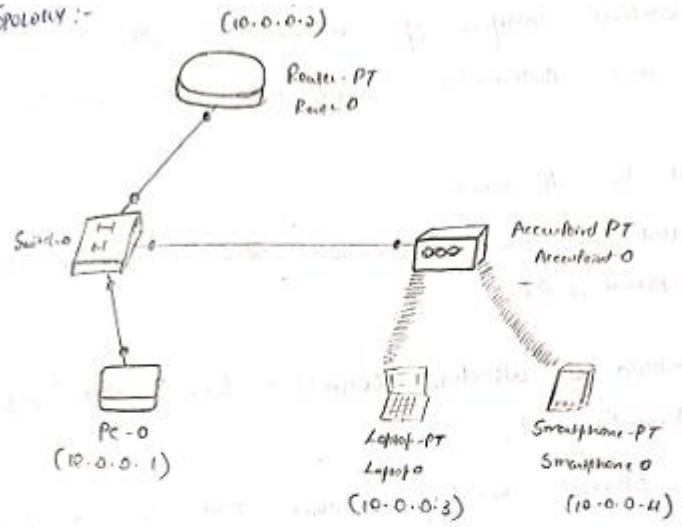




LABORATORY - 09

AIM:- To demonstrate the communication between two devices using wireless LAN.

TOPOLOGY :-



PROCEDURE :-

1. Select a Generic Access Point-PT Wireless device, a smartphone and a laptop and place it in the workspace.
2. Connect a generic device to a switch, Wireless device and Router
3. Set IP 10.0.0.1 to PC0
   10.0.0.2 → Router0
   10.0.0.3 → Laptop0
   10.0.0.4 → Smartphone0
4. Go to config mode of AccessPoint-PT, ↳ Under Port 0, set Bandwidth and Duplex to 'Auto' mode

5. Now under Port 1, set SSID to 'WLAN1'
and Authentication to WPA2-PSK
Set the Password (8-digit)

6. Set the Gateway addresses of Laptop and Smartphone

7. Under Wireless0 interface of Smartphone, set SSID to WLAN1 and Authentication WPA2-PSK and enter the password.
Then set the IP address

8. This establishes wireless connection B/W smartphone and the AccessPoint-PT

9. To establish a wireless connection between the laptop and Access Point-PT,
Go to physical view of the device and set 2.4GB Radio to the laptop.
and follow step 7 for laptop.

Observation :-

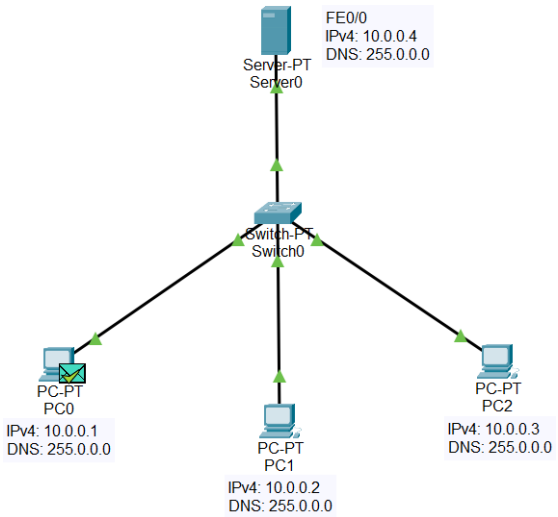
- We are able to establish a wireless connection B/W the devices like smartphone and laptop and the AccessPoint-PT

Go to Command prompt to laptop and execute the following commands
↳ ping 10.0.0.1

Pinging 10.0.0.1 with 32 byte of data
Reply from 10.0.0.1 : bytes = 32 time = 0ms TTL = 126
Reply from 10.0.0.1 : bytes = 32 time = 0ms TTL = 125
Reply from 10.0.0.1 : bytes = 32 time = 0ms TTL = 124
Reply from 10.0.0.1 : bytes = 32 time = 0ms TTL = 123

LABORATORY PROGRAM – 11

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



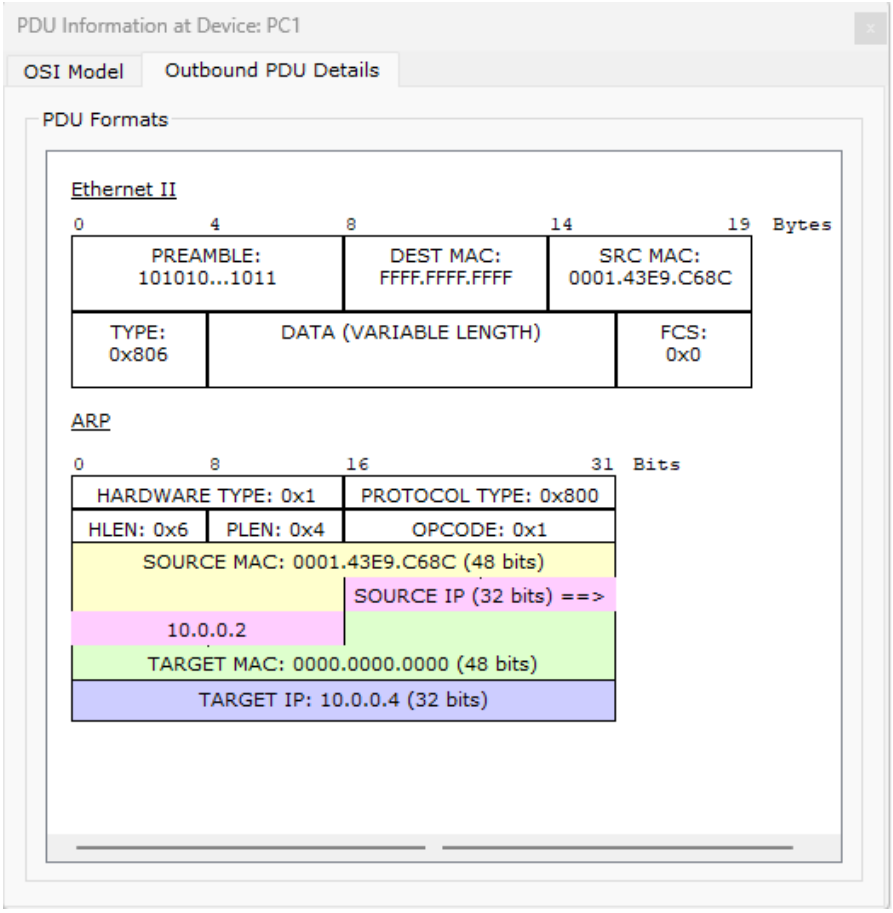


Figure 11.1: Inbound ARP, PC1

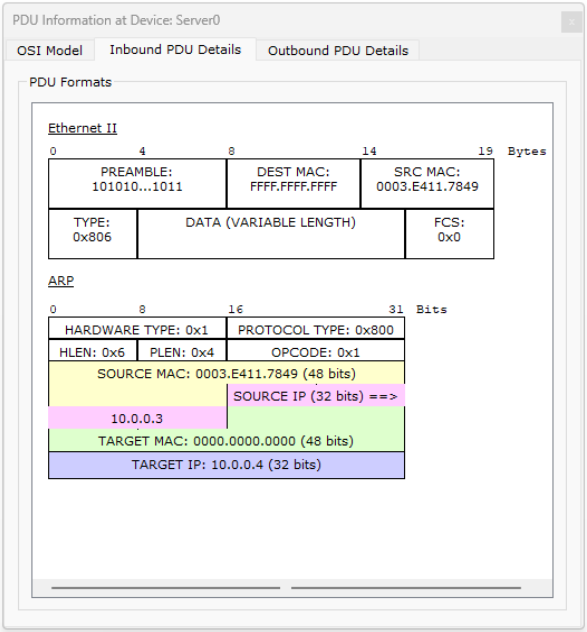


Figure 11.2: Inbound ARP, Server0

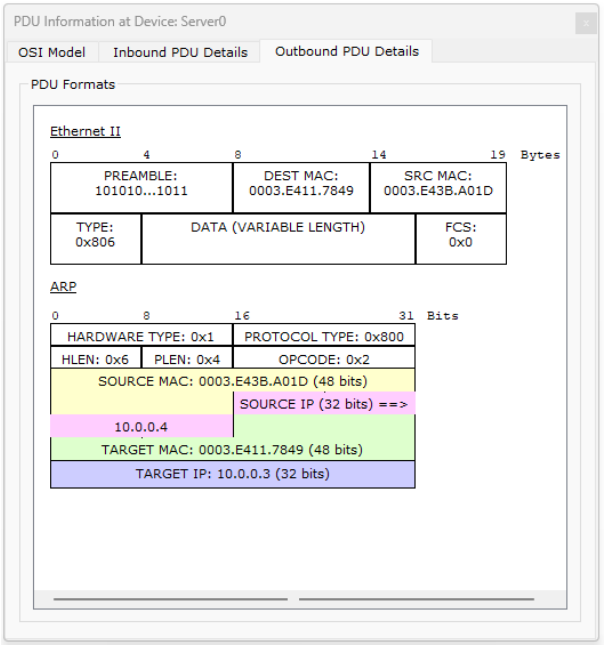


Figure 11.3: Outbound ARP, Server0

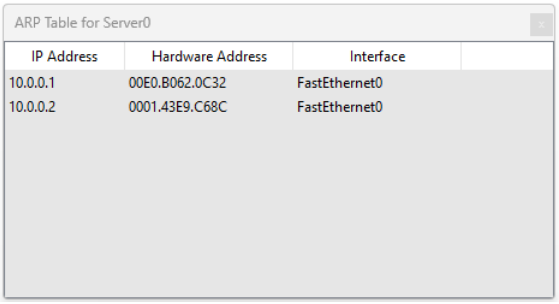


Figure 11.4: ARP Table, Server0

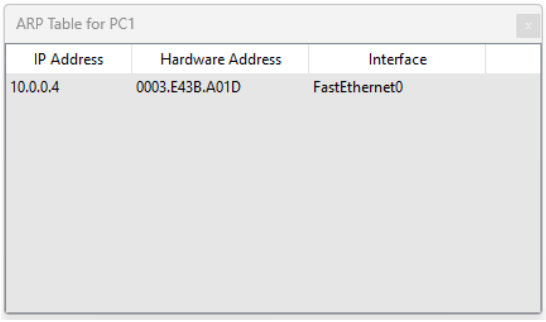
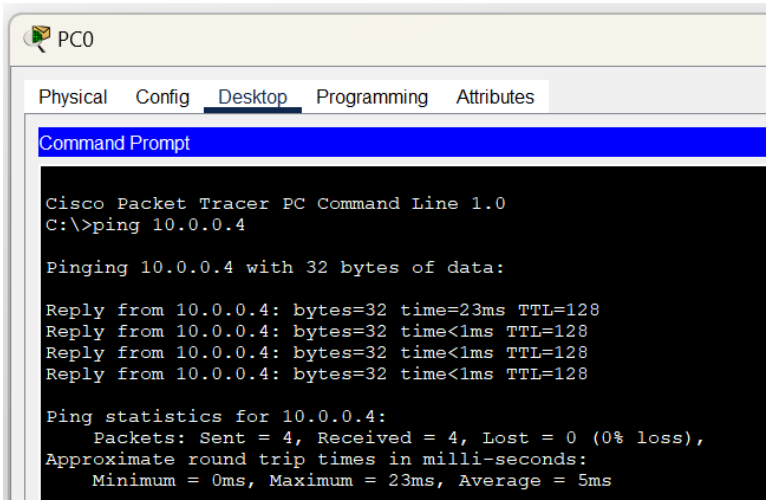


Figure 11.5: ARP Table, PC1

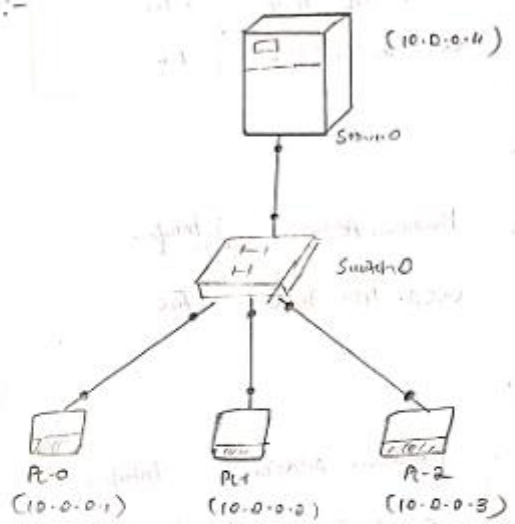
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Server0	ICMP		0.000	N	0	(edit)	



LABORATORY- 10

AIM :- To demonstrate the working of address resolution protocol for communication within a LAN

TOPOLOGY :-



PROCEDURE :-

1. Select 3 generic devices, a server and a switch
2. Set the IP address of the devices as

   PC0 - 10.0.0.1

   PC1 - 10.0.0.2

   PC2 - 10.0.0.3

   Server0 - 10.0.0.4
3. Go to and select the Inspect tool, click on PC0 and then select the ARP Table
4. Select a PDU and send a PDU from PC0 to the server
5. Now the ARP table will be populated with the entries.

ARP Table for Server 0

IP Address	Hardware Address	Interface
10.0.0.1	0060.3EDC.3375	Fa0
10.0.0.2	0009.7CE.431E	Fa0
10.0.0.3	0001.4278.C72C	Fa0

ARP Table for PC0

IP Address	Hardware Address	Interface
10.0.0.4	000A.4111.9B00	Fa0

ARP Table for PC1

IP Address	Hardware Address	Interface
10.0.0.4	000A.4111.9B00	Fa0

Inbound PDU Details

ARP

Initially, SOURCE MAC was 0000.0000.0000

~~Table~~,

SOURCE MAC : 000A.4111.9B00 (48 bits)

TARGET MAC : 0001.4278.C72C (48 bits)

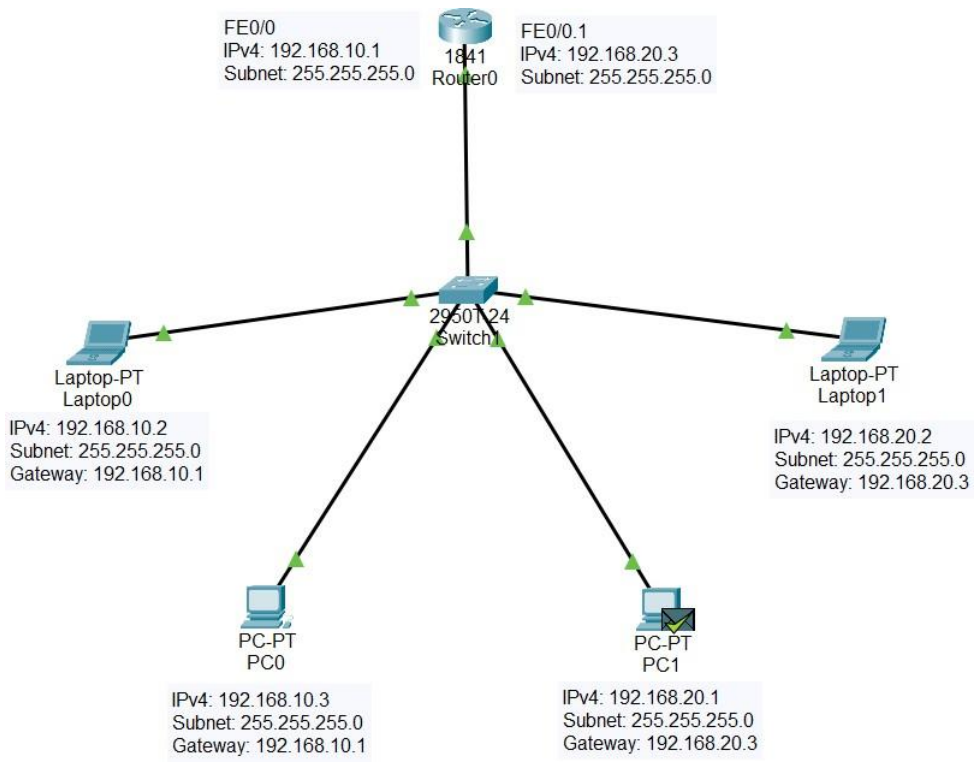
TARGET IP : 10.0.0.3 (32 bits)

SOURCE IP : 10.0.0.4 (32 bits)

03/12/24

LAB PROGRAM – 12

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



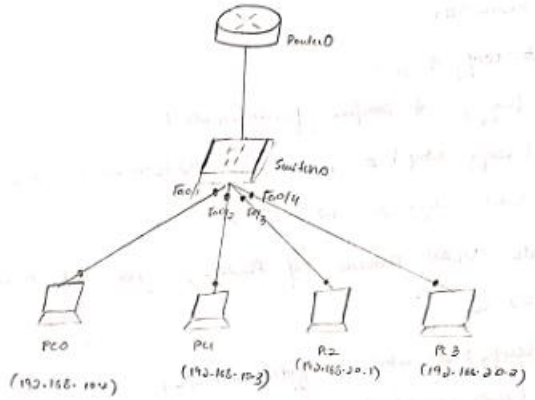

```
Router(config)#interface FastEthernet0/0.1
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown
```

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
```

AIM :- To create a virtual LAN on top of the physical LAN and enable communication between physical LAN and virtual LAN

TOPOLOGY :-



Procedure :-

- Connections are made according to the topology above
- Set IP address for the physical network as

  PC0 → 192.168.10.2
  PC1 → 192.168.10.3 } → gateway → 192.168.10.1
  Router0 → 192.168.10.1

- Go to switch → config
  Select VLAN Database
  Add a new VLAN configuration with VLAN Number (20) and VLAN Name (VLAN1)
- Under Fa0/5 of switch, select 'Trunk' Mode

- Go to Fa0/3 and select the VLAN as 20 VLAN1 (newly added VLAN)
  Similarly do for Fa0/4
- Now set IP addresses for Virtual LAN devices
  PC2 → 192.168.20.1
  PC3 → 192.168.20.2
- Under router CLI
  Router# config t
  Router (config) # interface fastethernet 0/0.1
  Router (config-subif) # ip address 192.168.20.3 255.255.255.
  Router (config-subif) # no shutdown.
- Now under VLAN Database of Router0, Add the new VLAN
- In Router0 CLI
  Router(config) # interface fastethernet 0/0.1
  Router (config-subif) # encapsulation dot1q 20 → VLAN

Observations :-

We are able to ping successfully from Physical VLAN to Virtual LAN and vice versa

PC0 > ping 192.168.20.2

Pinging 192.168.20.2 with 32 bytes of data :

Reply from 192.168.20.2 : bytes = 32 time = 0ms TTL =

⋮

Ping Statistics :

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

Approximate Round-trip times in milli-seconds : Average = 0ms

LAB PROGRAM – 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")


    # Simulate the receiver side

    received_data = input("Enter the 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
```

Implement CRC Error detecting code

Main implementation ⇒

```
def xor (dividend, 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) :
    dataLen = len(data)
    gen Len = len (gen_poly)

    padded_data = data + '0' * (genLen -1)
    check_value = padded_data [: genLen]

    for i in range(dataLen) :
        if check_value [0] == '1' :
            check_value = xor (checkvalue, gen_poly)
        else:
            check_value = check_value [1:]

        if i+ gen_length < len (padded_data) :
            check_value += check_value[-1] padded_data [i+genLen]
    return check_value [1:]

def reciever (data, gen_Poly) :
    print ("Data recieved :", data)
    remainder = crc (data, gen_Poly)
```

```
    if '1' in remainder :
        print (" Error detected ")
    else
        print (" No error detected")

if __name__ == "main" :
    data = input (" Enter data to be transmitted :")
    gen_poly = input (" Enter the generating polynomial ")
    Check Value = CRC (data, gen_poly)
    print ( " Data padded with n-1 zeros :", data +
                                    '0' * (len (gen_poly)-1))
    print (" CRC or check value :", checkvalue)
    print (" Data sent : ", (data + checkValue))
    receiver (sent)
    receivedData = input (" Enter the received data ")
    receiver ( receivedData, gen_poly)
```

⇒ OUTPUT:-

Enter data to be transmitted : 1001001

Enter the Generating Polynomial : 10000101

CRC or check value : 0000101

Final data sent : 10010010000101

Enter the data received : 10011100010100

Error detected.

17/10/24

LAB PROGRAM – 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
```

LABORATORY - 13

17/10/24

⇒ Write a program for congestion control implementation using leaky bucket algorithm

Python Implementation :-

```
storage = int (input (" Enter the initial packets in the bucket "))
No_of_queries = int (input (
```

C implementation :-

```
# include <stdio.h>
int main()
{
    int incoming, outgoing, buck_size, n, store = 0;
    printf (" Enter bucket size, outgoing rate and no. of inp ");
    scanf (" %d %d %d ", &buck_size, &outgoing, &n);
    while (n!=0)
    {
        printf (" Enter the incoming packet size ");
        scanf (" %d ", &incoming);
        printf (" Incoming packet size %d\n", incoming)
        if ( incoming <= (buck_size - store))
        {
            store += incoming;
            printf (" Bucket buffer size %d out of %d in store, buck_size);
        }
        else {
            printf (" Dropped %d no of packets in ",
                                incoming - (buck_size - store));
            store = buck_size;
            store = store - outgoing;
            printf (" After outgoing %d bytes left out of %d in buffer ", store, buck_size);
            n--;
        }
    }
```

Output :-

```
Enter bucket size, outgoing rate, and no. of inputs : 10 3 3
Enter the incoming packet size : 5
Bucket buffer size 5 out of 10
After outgoing 2 bytes left out of 10 in buffer
Enter the incoming packet size : 6
Dropped 1 no of packet
Bucket buffer size 5 out of 10
After outgoing 2 bytes left out of 10 in buffer
```

17/10

LAB PROGRAM – 15(A)

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

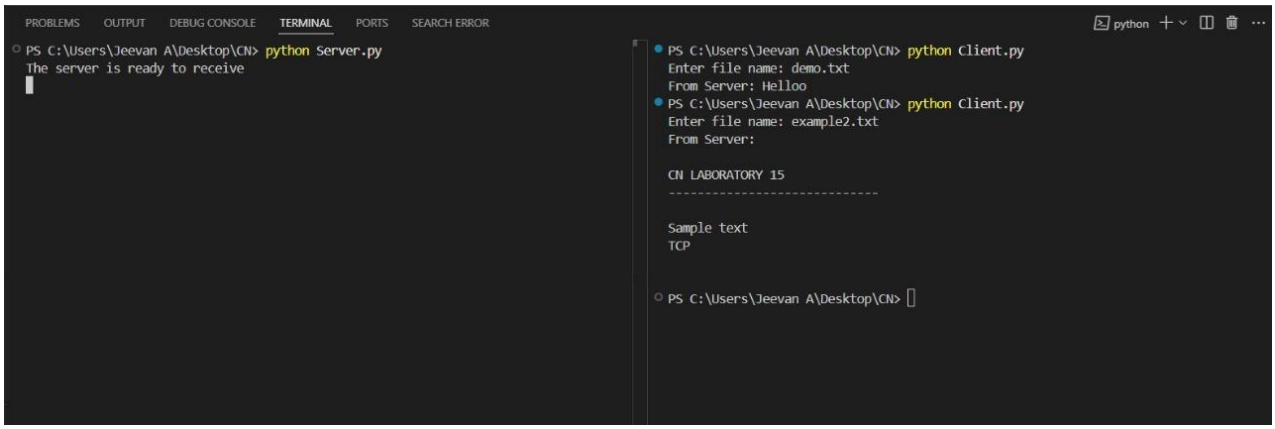
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# 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 (TCP):



LAB PROGRAM – 15(B)

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 (UDP):