VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Networks Laboratory (23CS5PCCON)

Submitted by

Mallikarjun M Kuri(1BM22CS144)

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 Networks (23CS5PCCON)" carried out by **Mallikarjun M Kuri(1BM22CS144),** 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 Laboratory report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

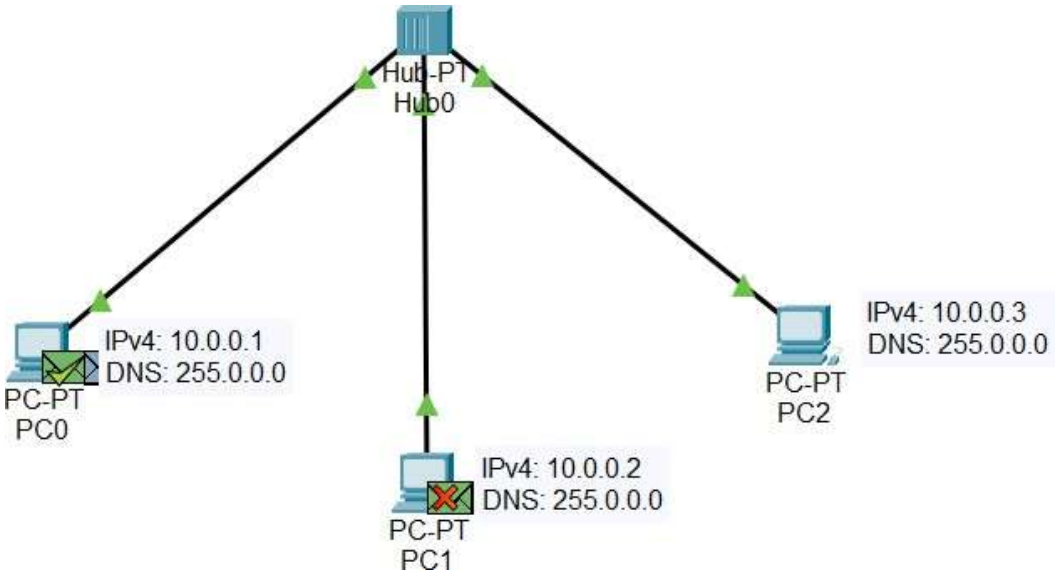
Srushti C S	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE, BMSCE	Department of CSE, BMSCE

INDEX

Github Link: https://github.com/7MalliKarjun008/CN-1BM22CS144-

LABORATORY PROGRAM – 1

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



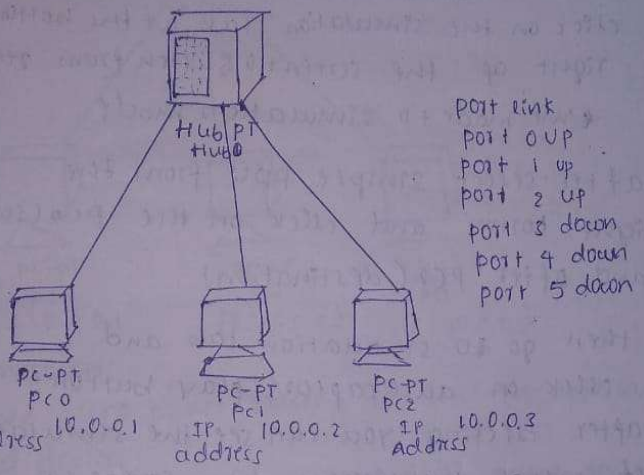
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	

```
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-1 Hub

AIM: To demonstrate the transmission of a single PDU between 2 devices connected using a hub and a switch

Observation:



port link
port 0 up
port 1 up
port 2 up
port 3 down
port 4 down
port 5 down

i) Opened cisco packet tracer

ii) selected 3 devices from the tab below the window, ~~and~~ selected 3 generic devices

iii) from connecting device, I selected Generic Hub

iv) I ~~choose~~ Tap on connection option from the bottom and I selected Automatically choosen connection and connected each device with the Hub

v) Then from the right side I selected simple PDU ~~and~~ from the Icon showed as Add simple PDU and after selecting it

Assign unique IP Addresses to each PC for example

PC0 Address: 10.0.0.1 Subnet mask: 255.0.0.0
PC1 Address: 10.0.0.2 Subnet mask: 255.0.0.0
PC2 Address: 10.0.0.3 Subnet mask: 255.0.0.0

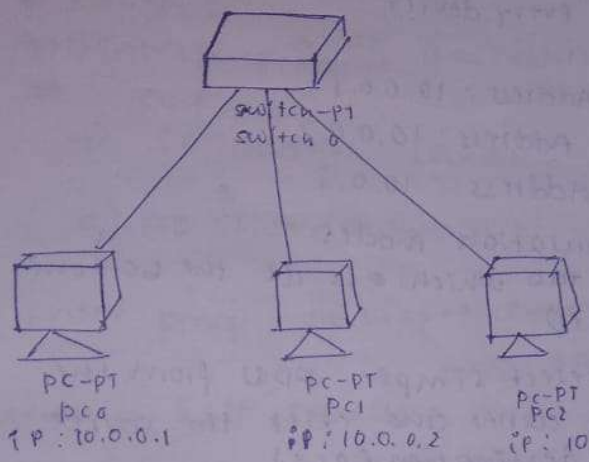
Switch to simulation mode:

click on the "simulation" tab at the bottom right of the screen to switch from real-time mode to simulation mode

* after select simple PDU from the right corner and click on the PC0 (source) and after PC2 (destination)
* then go to simulation tab and click on auto capture/play button
* after clicking you can see the simulation of packet transferring from source to destination,
* firstly the packet will goto Hub and then it will go to destination (PC2) you can see (✓) right mark on PC2 and cross (X) mark on PC1

Switch

Observation



Observation:

i) open cisko packet Tracer; launch the application

ii) Add devices:
from the bottom left corner select device and click on the blank sheet, and like this select the some more devices
(I selected 3 devices)

iii) after that click on the connecting devices and select switch from there (which is available)

connecting devices:
at the bottom left corner there is one option called connect devices; from there select connecter, choose automatic connecter or you can connect copper-through straight" cable

Assign IP address:

click on each devices to get desktop and select config tab and set ip address for every device

for example

PC 0 ip Address : 10.0.0.1

pc1 ip Address : 10.0.0.2

PC 2 ip Address : 10.0.3

* switch to simulation mode:
(click on the tab which is at the bottom right corner).

* after that select simple PDU from the bottom right corner and click the source (PC0) and destination (PC2)

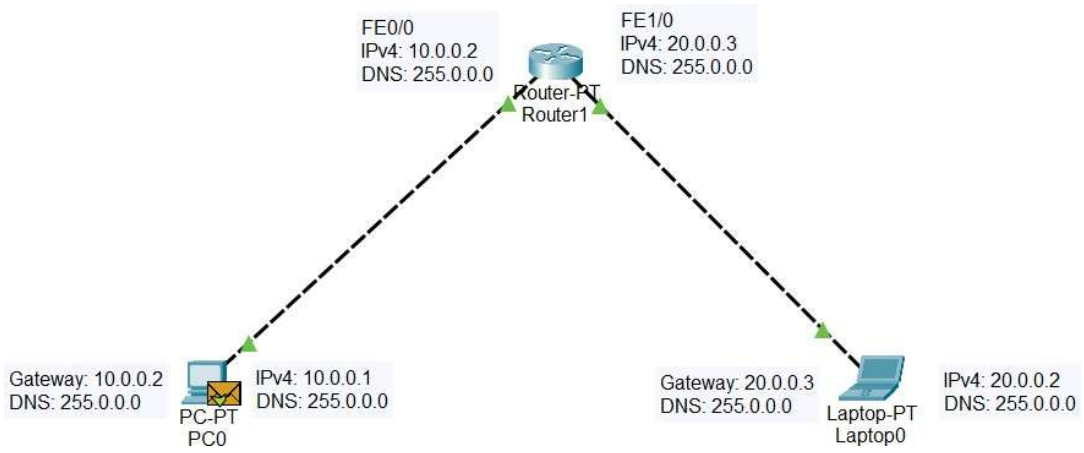
* then go to simulation mode and click on the button called Auto capture, the you can see the simulation

* firstly the packet will go to switch and through switch the packet will go to destination.
You can see this if there is (✓) mark on the destination pc and (X) mark on the remaining pc

Here you will see how the switch uses its MAC address to intelligently forward the ping (ICMP) packets to the correct destination (PC2), rather than broadcasting to all connected devices like a hub would

LABORATORY PROGRAM – 2

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



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
	In Progress	PC0	Laptop0	ICMP		0.000	N	1	(edit)	
	In Progress	PC0	Laptop0	ICMP		0.000	N	2	(edit)	

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data:

Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

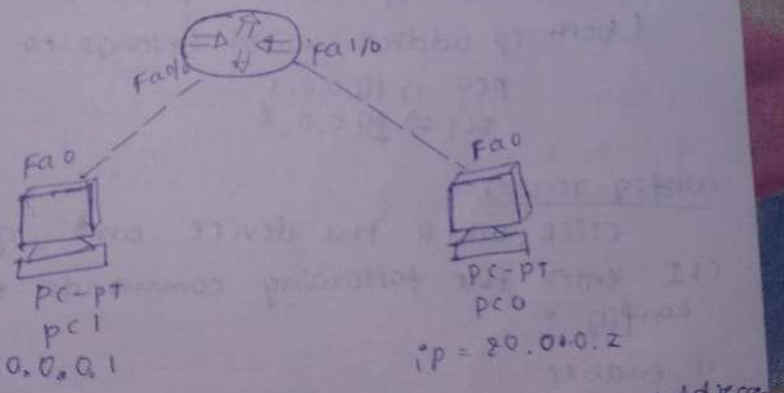
commands IN CLI

1) Enable
2) #
3) config t
4) interface name (fastethernet 0/0)
5) config if
   4) ip address 10.0.0.1 255.0.0.0
   5) no shutdown

TO know ping Desktop → command prompt
ping 20.0.0.2

Gateway ⇒ Exit Interface
Set Gateway config → gateway set 10.0.0.1
20.0.0.2
ping ip address of router

TOPOLOGY Router



Aim: To demonstrate config of ip address to the router and explore ping command

Observation:

* Open cisco packet tracer: Launch the applicatio
* Add devices: from the bottom left corner there is one showing devices select generic device from that, click on to the blank sheet, like this select another one
* Add Router: Just above the devices there is one tab called connections from that select one generic Router and place it on the blank sheet
* and select connecter ("copper-through straight" cable and connect each device with the router

Assign ip Address

click on each device and go to config tab click on the ip addressif for each device (both ip address should belongs to

pc0 => 10.0.0.1
pc1 => 10.0.0.2

config router

click on to the device and go to CLI enter the following commands to config.

1) enable
2) config t
3) interface fastethernet 0/0
4) ip address 10.0.0.2 255.0.0.0
5) no shutdown

Do the
1) ena
2) con
3) in
4) ip
5) no

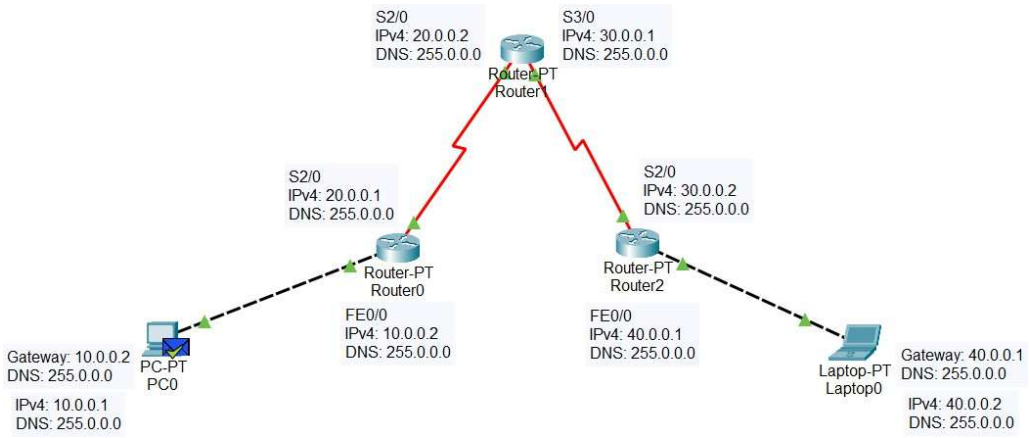
after th
TO
the ga
for de

after th
desktc
and

Slip
22/10

LABORATORY PROGRAM – 3

Configure static route to the 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
S    30.0.0.0/8 [1/0] via 20.0.0.2
S    40.0.0.0/8 [1/0] via 20.0.0.2
```

Figure 3.1: Router0

```
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
```

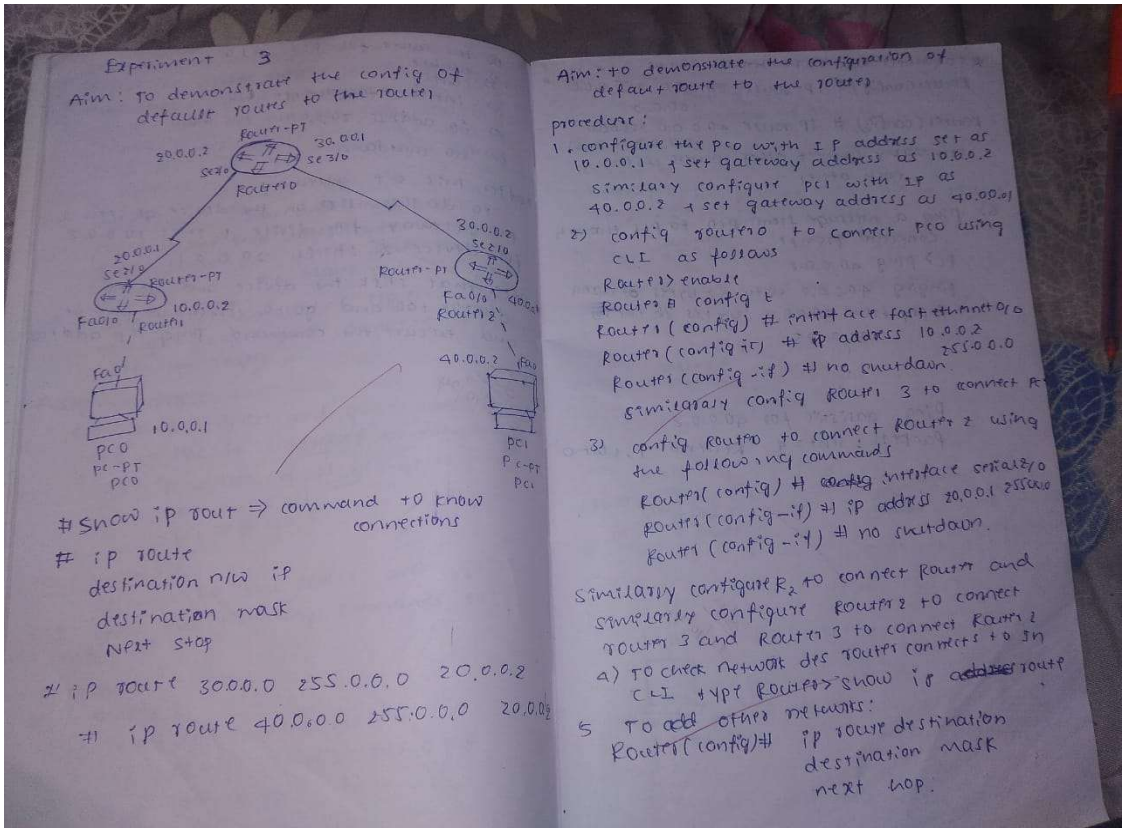
Figure 3.2: Router1

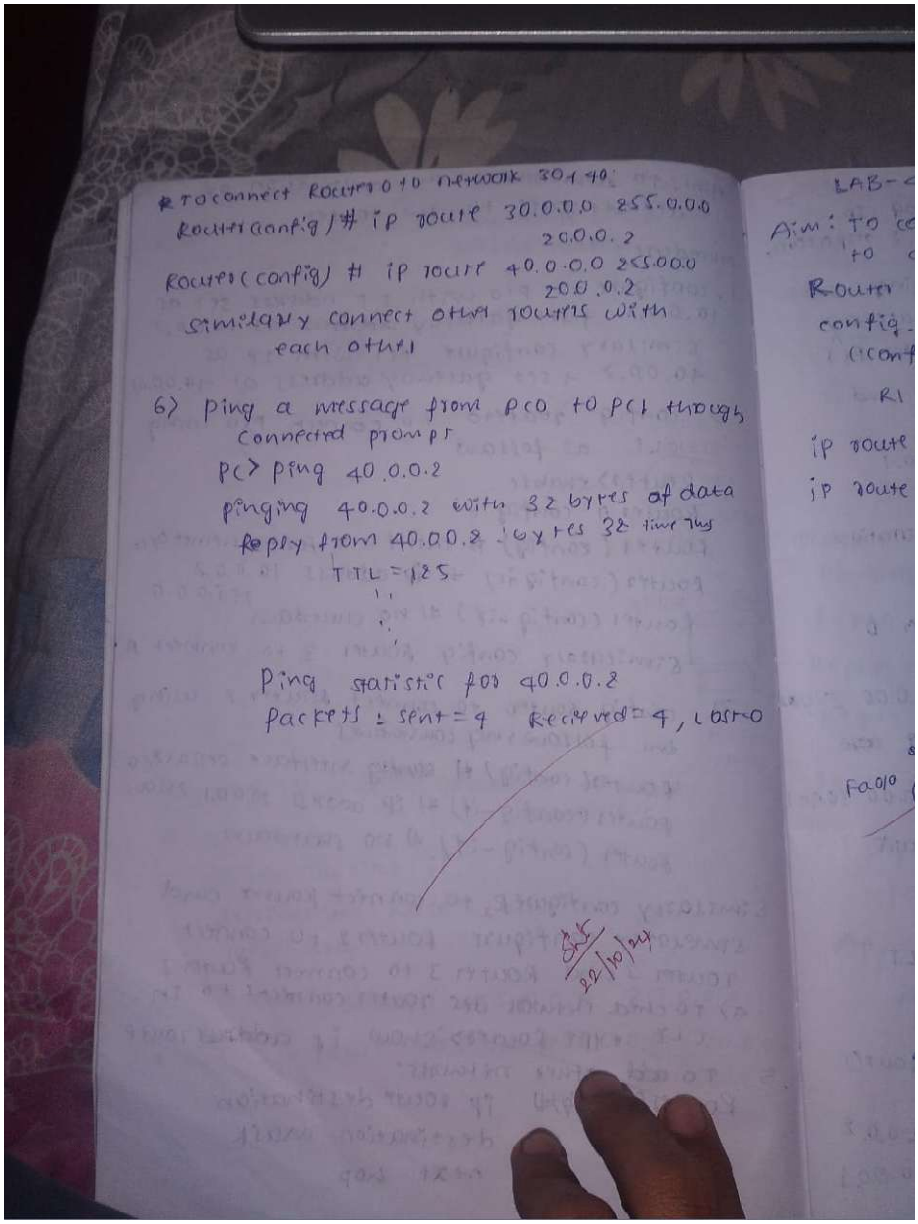
```
S    10.0.0.0/8 [1/0] via 30.0.0.1
S    20.0.0.0/8 [1/0] via 30.0.0.1
C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
```

Figure 3.3: Router3.3

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

```
Cisco Packet Tracer PC Command Line 1.0
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=36ms TTL=125
Reply from 40.0.0.2: bytes=32 time=34ms TTL=125
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125
Reply from 40.0.0.2: bytes=32 time=26ms 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 = 26ms, Maximum = 36ms, Average = 31ms
```





* To connect Router 0 to network 30.1.1.0:
Router(config)# ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router(config)# ip route 40.0.0.0 255.0.0.0 200.0.0.2
similarly connect other routers with each other

6) Ping a message from PC0 to PC1 through connected 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 7ms TTL = 125

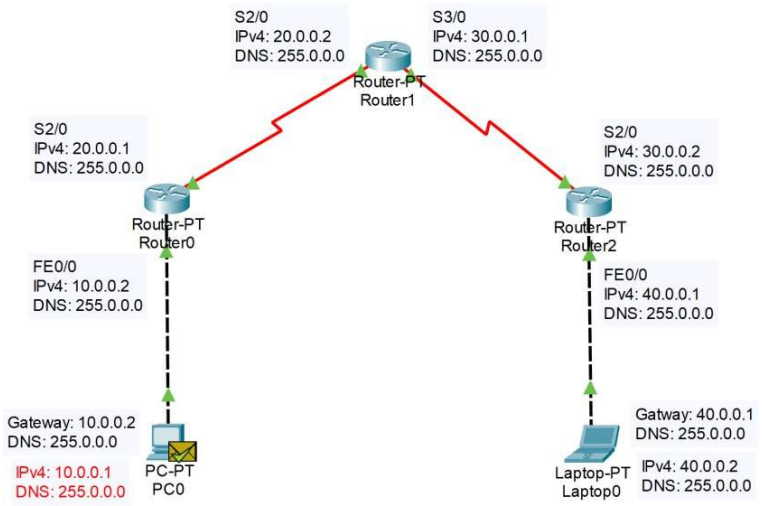
Ping statistic for 40.0.0.2
Packets : Sent = 4 Received = 4, Lost = 0

LAB-
Aim: To
Router
config
(conf
R1
ip route
ip route
Fa0/0

22/10/24

LABORATORY PROGRAM – 4(A)

Configure default route, static route to the Router.



SHOW IP ROUTE

```
Gateway of last resort is 20.0.0.2 to network 0.0.0.0

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

Figure 4.1: Router0

```
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
```

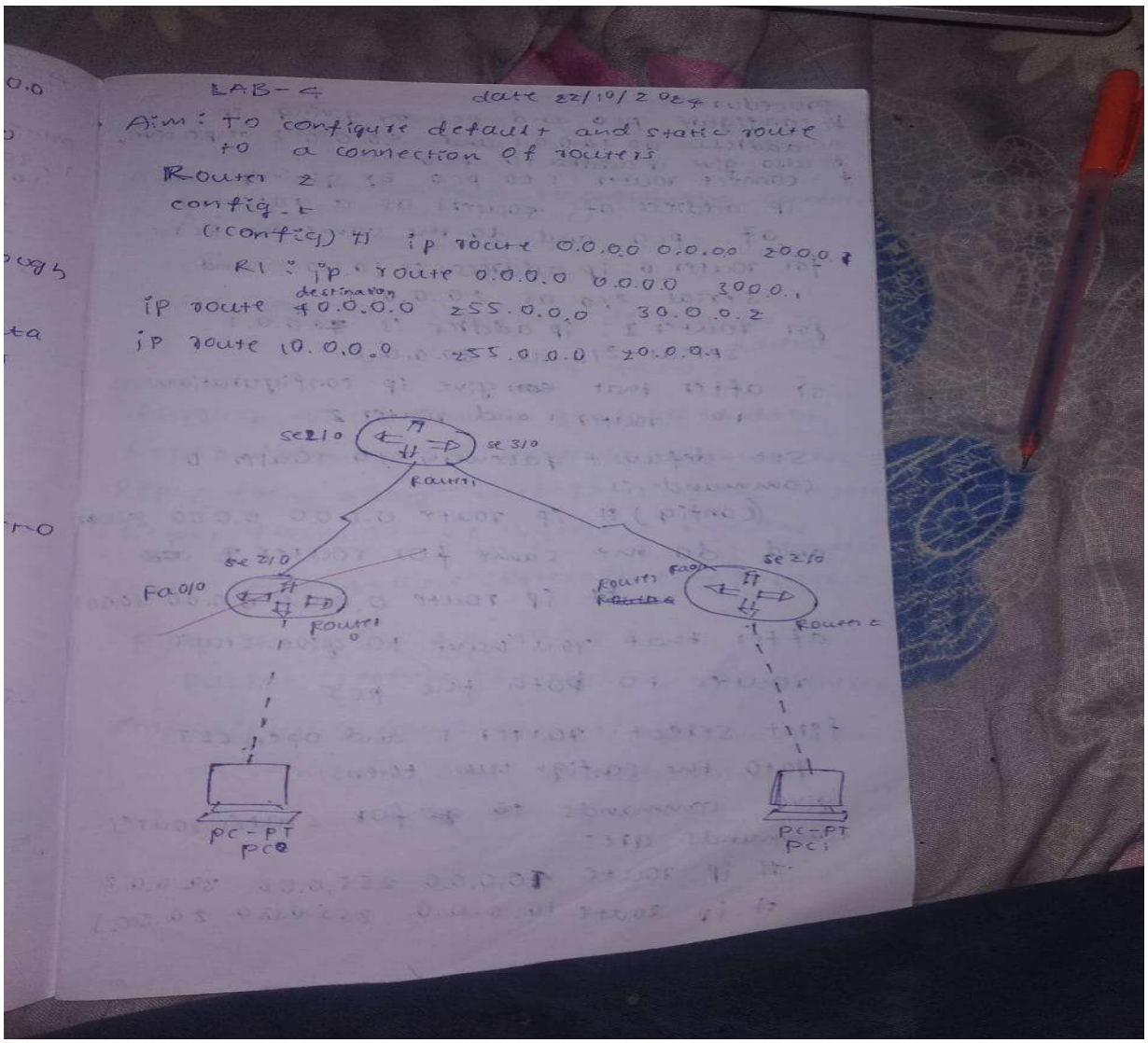
Figure4.2: Router1

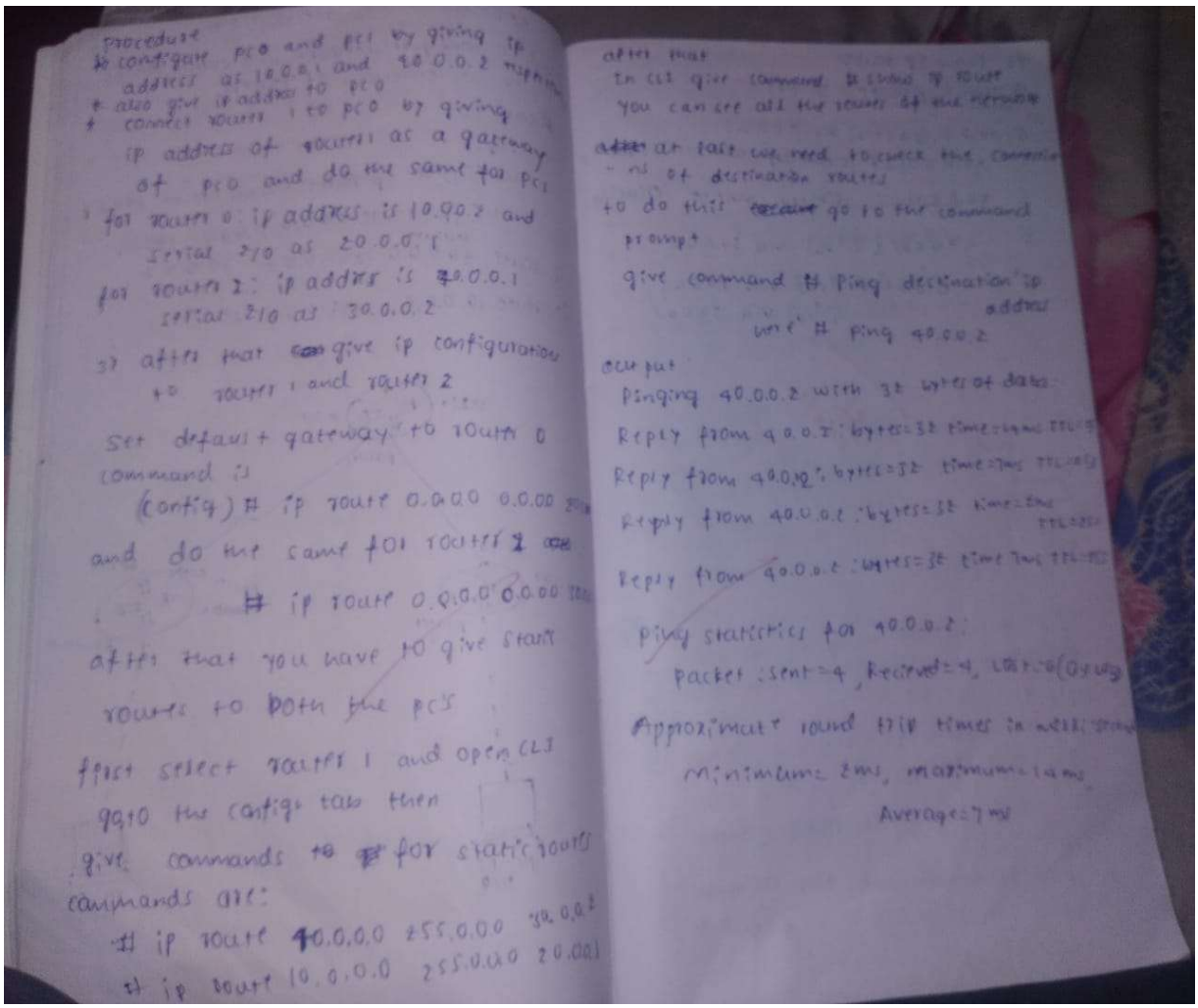
```
Gateway of last resort is 30.0.0.1 to network 0.0.0.0

C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 30.0.0.1
```

Figure 4.3: Router2

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	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=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-4
date 22/10/2024
Aim: To configure default and static route to a connection of routers
Router 2
config t
(config) ip route 0.0.0.0 0.0.0.0 20.0.0.1
R1: ip route 0.0.0.0 0.0.0.0 30.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2
destination
ip route 10.0.0.0 255.0.0.0 20.0.0.1
se2/0
se3/0
Router1
Fa0/0
Router0
Router2
PC-PT
PC0
PC1



Procedure

* configure pc0 and pc1 by giving ip address as 10.0.0.1 and 40.0.0.2 respectively
* also give ip address to pc0
* connect router 1 to pc0 by giving ip address of router1 as a gateway of pc0 and do the same for pc1

1) for router 0 ip address is 10.0.0.2 and serial 2/0 as 20.0.0.1

for router 2: ip address is 30.0.0.1 serial 2/0 as 30.0.0.2

3) after that give ip configuration to router 1 and router 2

Set default gateway to router 0 command is

(config)# ip route 0.0.0.0 0.0.0.0 [illegible]

and do the same for router 2 as

ip route 0.0.0.0 0.0.0.0 [illegible]

after that you have to give static routes to both the pcs

first select router 1 and open CLI go to the config tab then give commands for static routes commands are:

ip route 40.0.0.0 255.0.0.0 20.0.0.2

ip route 10.0.0.0 255.0.0.0 20.0.0.1

after that

In CLI give command # show ip route you can see all the routes of the network

at last we need to check the connection of destination router

to do this go to the command prompt

give command # Ping destination ip address

ex: # Ping 40.0.0.2

output

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=1ms TTL=[illegible]

Reply from 40.0.0.2: bytes=32 time=2ms TTL=[illegible]

Reply from 40.0.0.2: bytes=32 time=2ms TTL=[illegible]

Reply from 40.0.0.2: bytes=32 time=2ms TTL=[illegible]

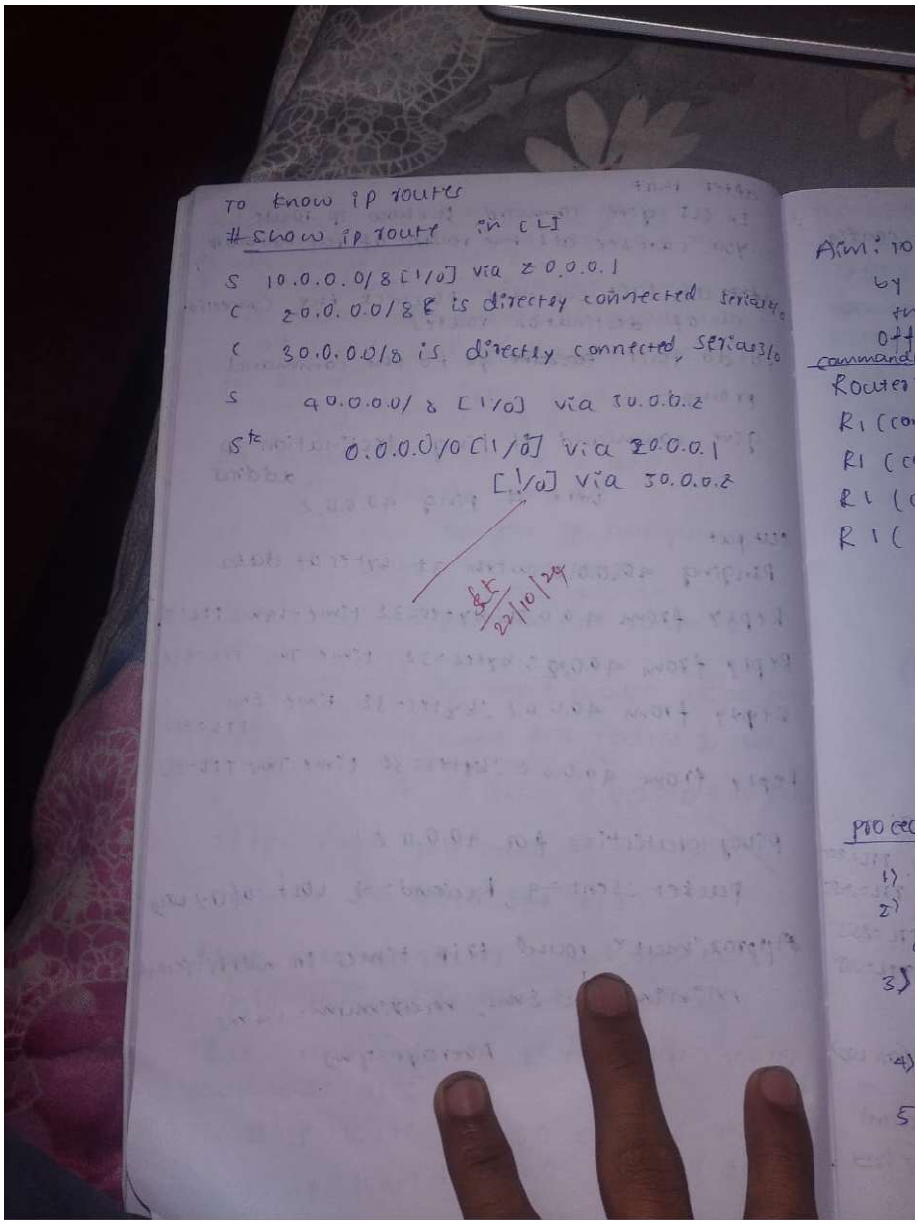
Ping statistics for 40.0.0.2:

Packet: Sent=4, Received=4, Lost=0(0% loss)

Approximate round trip times in milli-seconds:

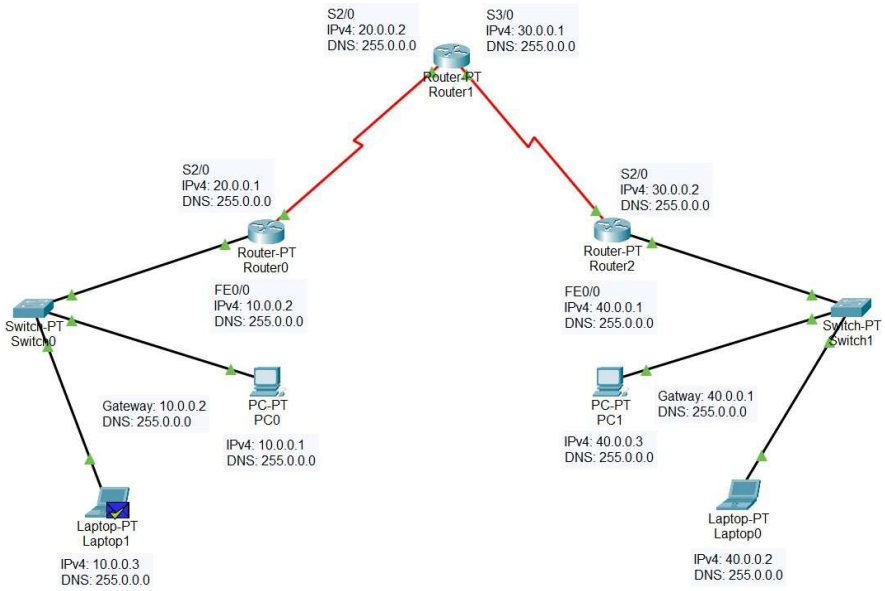
Minimum=2ms, Maximum=14ms

Average=7ms


To know ip routes
#show ip route
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected
C 30.0.0.0/8 is directly connected, Serial3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2
S* 0.0.0.0/0 [1/0] via 20.0.0.1
[1/0] via 30.0.0.2
22/10/24

LABORATORY PROGRAM – 4(B)

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



SHOW IP ROUTE

```
Gateway of last resort is 20.0.0.2 to network 0.0.0.0

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

Figure 4.1: Router0

```
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
```

Figure4.2: Router1

```
Gateway of last resort is 30.0.0.1 to network 0.0.0.0

C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 30.0.0.1
```

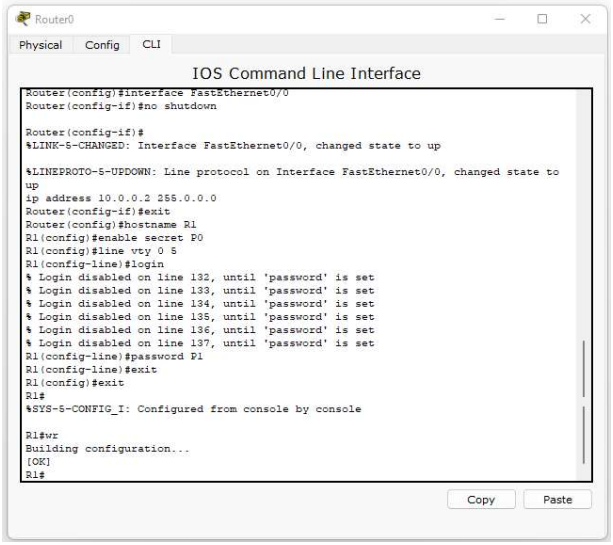
Figure 4.3: Router2

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

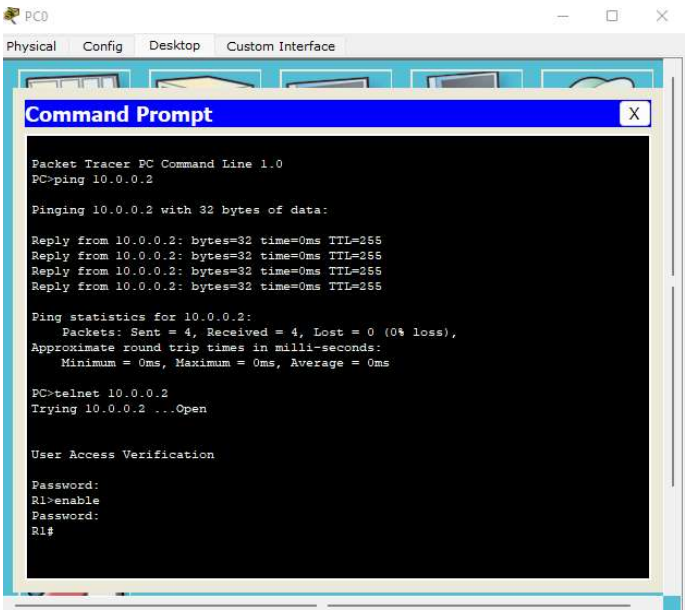
```
C:\>ping 40.0.0.3

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
```

LABORATORY 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-5

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

Commands:

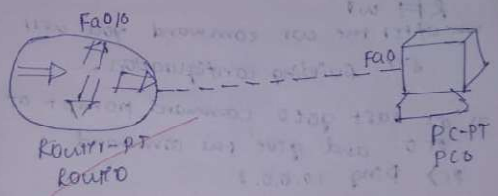
Router(config)# hostname R1

R1(config)# enable secret P0

R1(config-line)# login

R1(config-line)# password P1

R1(config-line)# exit



Procedure:

1) select one PC from the devices
2) select one router from the connecting devices
3) config PC with ip address
PC0 10.0.0.1 255.0.0.0
4) config router with ip addres 10.0.0.2 255.0.0.0
5) enable the connection and give the gateway to router and make sure the connection is successful.

6) after that go to CLI of router and give the below commands in config tab

R1(config-if)# hostname R1
R1 (config)# enable secret P0
R1 (config-line) vty 0 5
R1 (config-line)# login
R1 (config-line)# password P1
R1 (config-line)# exit
R1 (config)# exit
R1 #
R1# wr
after the wr command you will see Building configuration...

7) at last goto command prompt of PC0 and give the command
PC> ping 10.0.0.2

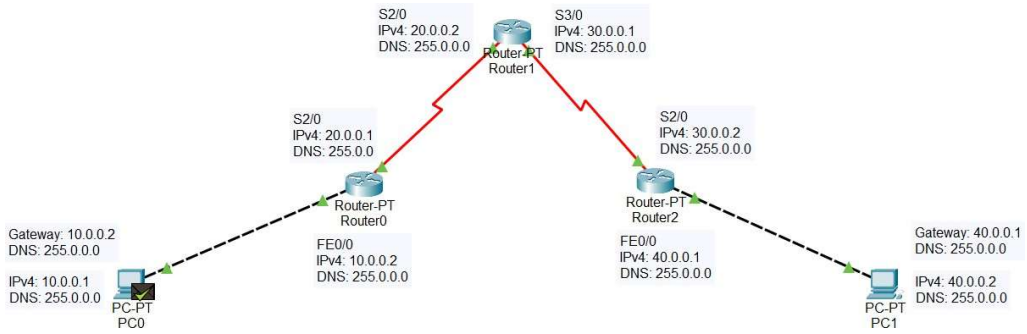
pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=0ms TTL=
Reply from 10.0.0.2: bytes=32 time=0ms TTL=
Reply from 10.0.0.2: bytes=32 time=4ms TTL=
Reply from 10.0.0.2: bytes=32 time=4ms TTL=

ping statistics for 10.0.0.2
packets: sent =4, Received = 4, Lost =0 (0%
Approximate round trip time in milli-seconds
minimum = 0ms, Maximum = 4ms, Average =

8) then you execute the command
PC> telnet 10.0.0.2
Trying 10.0.0.2 ... Open
User Access Verification
Password: P1
R1>
[Connection to 10.0.0.2 closed by foreig

LABORATORY PROGRAM – 6

Demonstrate the TTL/ Life of a Packet.



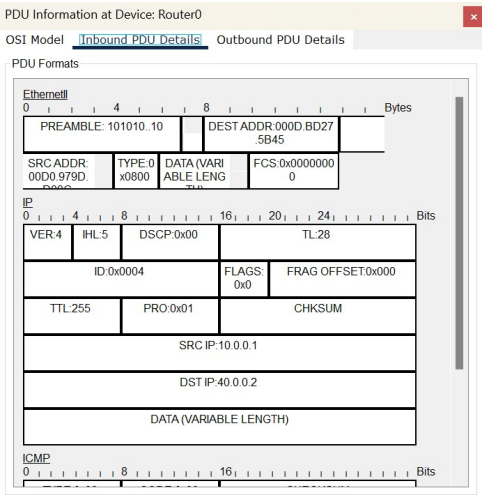


Figure 6.1: Inbound PDU, Router0

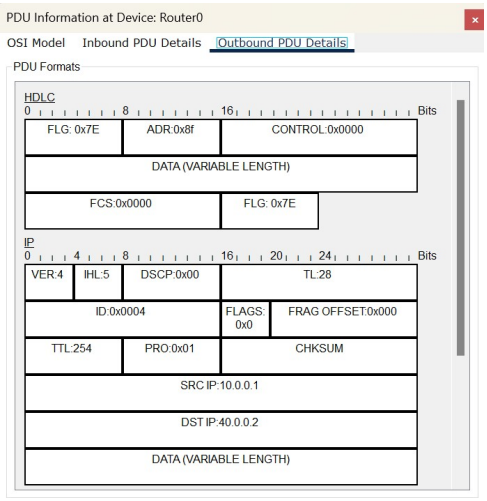


Figure 6.2: Outbound PDU, Router0

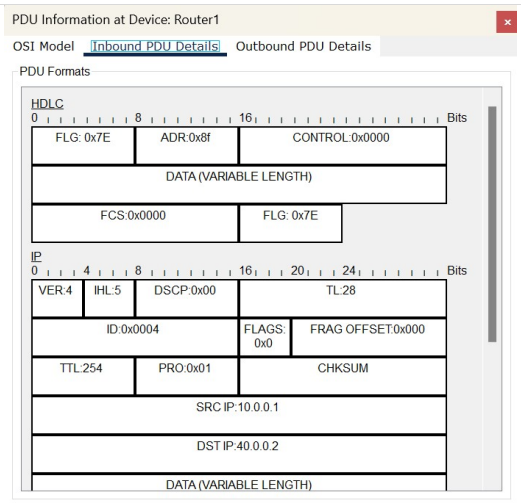


Figure 6.3: Inbound PDU, Router1

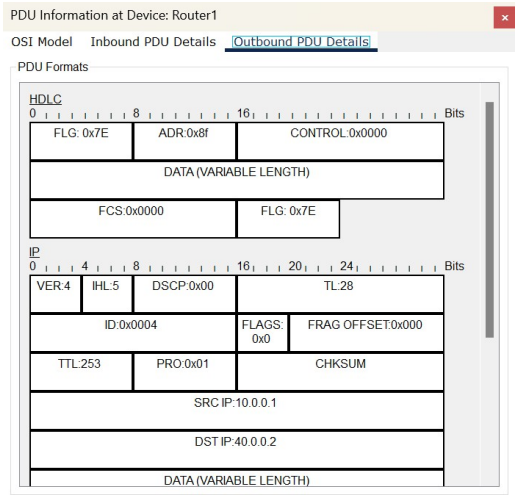


Figure 6.4: Outbound PDU, Router1

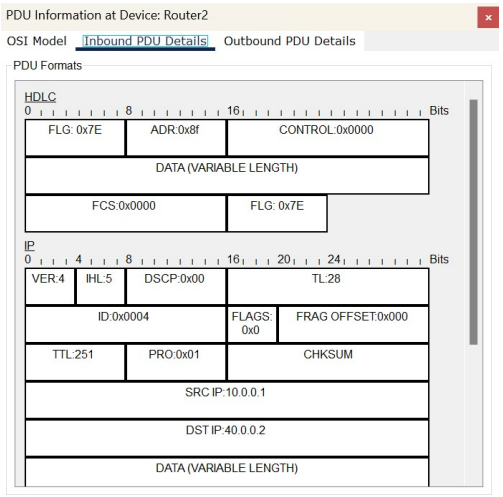


Figure 6.5: Inbound PDU, Router2

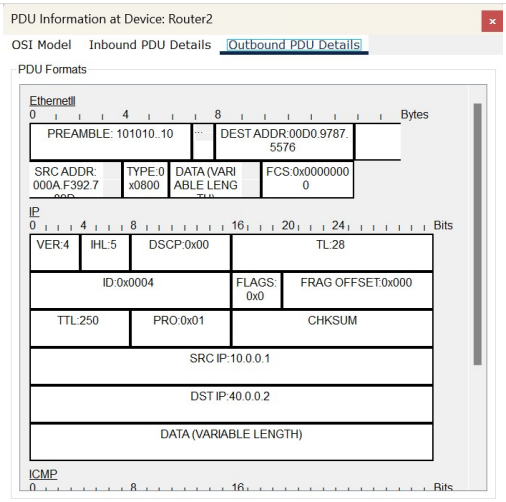


Figure 6.6: Outbound PDU, Router2

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
```

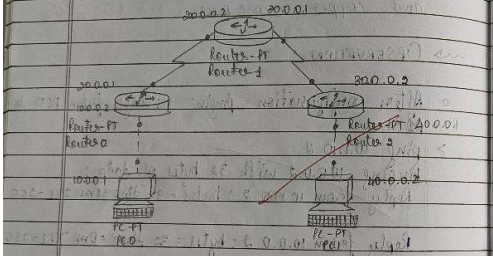
29.10.24

DATE: PAGE:

LABORATORY PROGRAM - 6

AIM OF THE EXPERIMENT : Demonstrate the TTL / Life of a Packet

TOPOLOGY :



CONNECTIONS : Copper Cross-Over (from end devices to routers)
Serial DTE (from Router to Router)

TOPOLOGY CREATION :

Construct a network as mentioned in diagram.

CONFIGURATION :

- Config all routers and end devices accordingly as done in previous experiment with default and static routing.

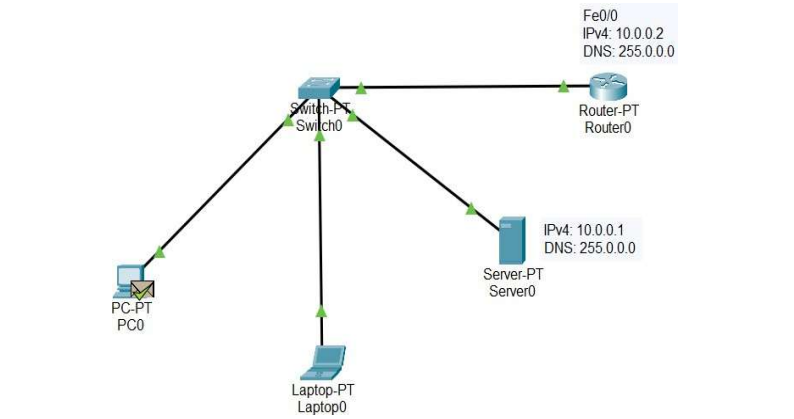
DATE: PAGE:

OBSERVATION :

- After configuration and pinging message from 10.0.0.1 to 20.0.0.2 using Simple PDU.
- In simulation mode, on clicking PDU at Initial stage, at Inbound PDU details the TTL : 255 and after 1 hop, the Outbound PDU details on TTL : 254. Thus, at each hop, the TTL is decreased by one.
- At destination, in PC1, the TTL is 250, means there is a decreament of TTL at each hop.
- The message is transmitted successfully
- And if the TTL is less than requised hop, the packet is discharded, and again a new message has to be resent.

LABORATORY PROGRAM – 7(A)

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



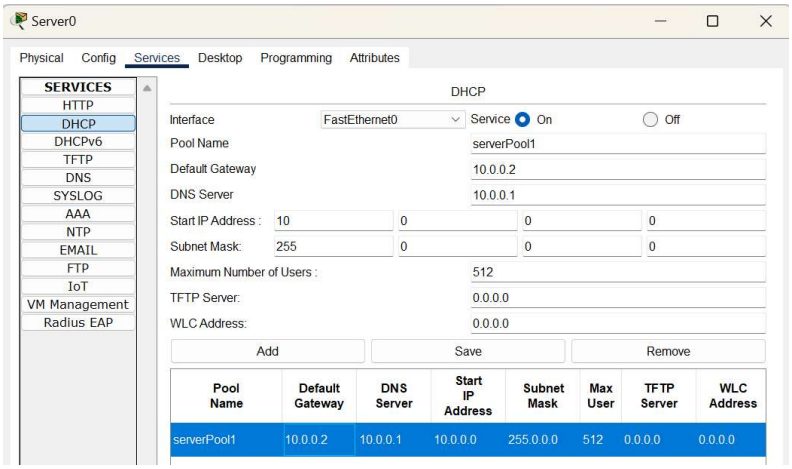


Figure 7.1: DHCP Service, Server0

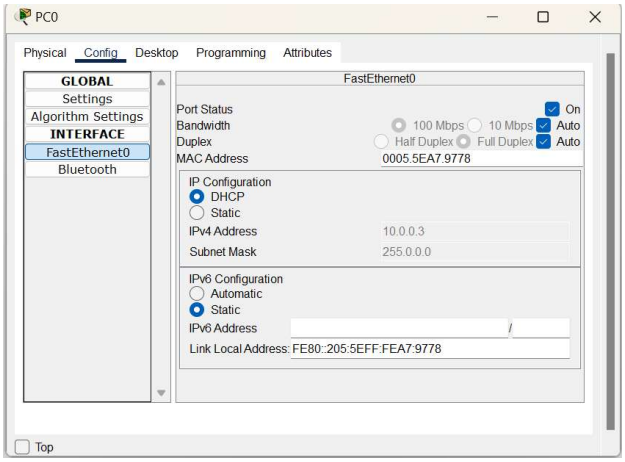


Figure 7.2: DHCP Service, PC0

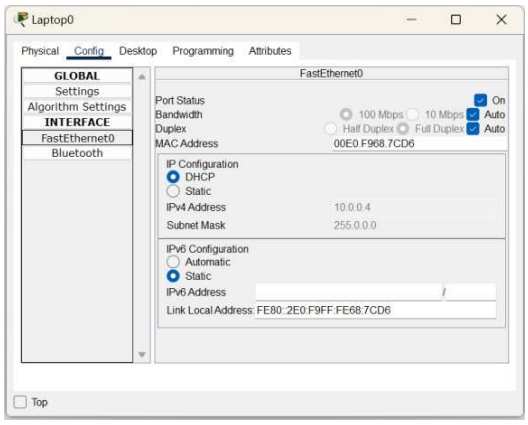
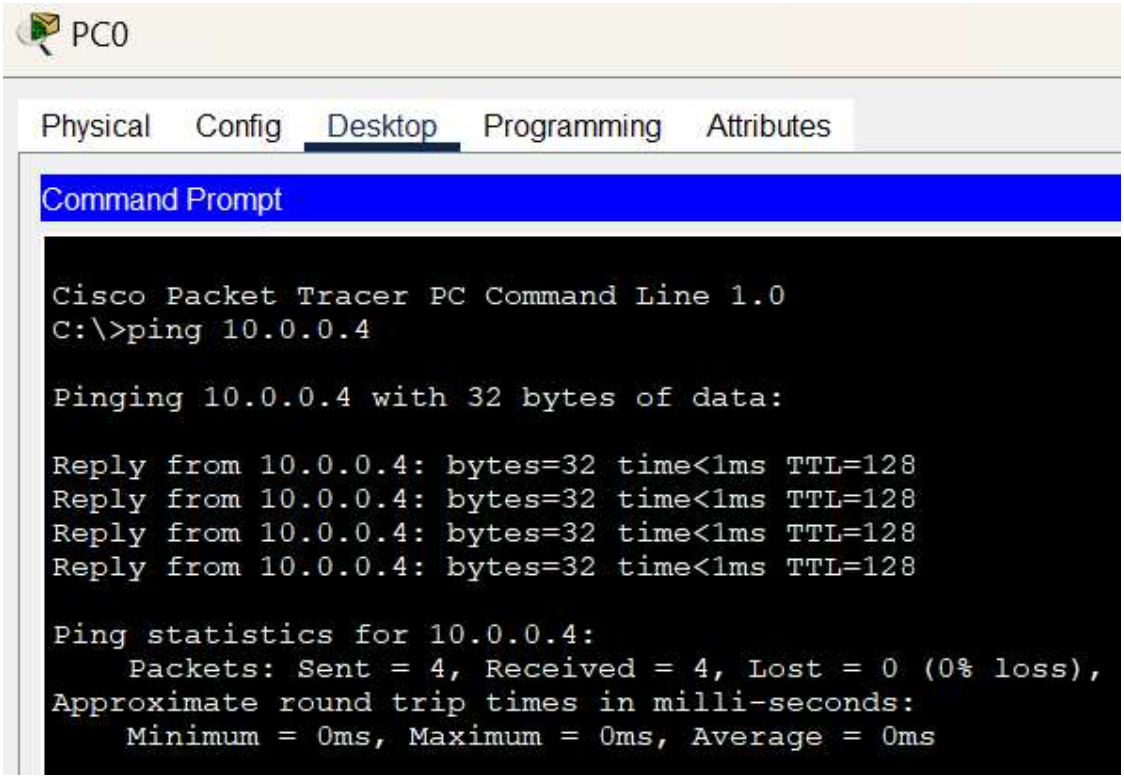


Figure 7.3: DHCP Service, Laptop0

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



LAB - 6

AIM: To configure ip addresses of the host using DHCP server present within the line

B) AIM: To configure ip addresses of the host using DHCP server present in the diff. -rent Line

DHCP => Dynamic host configuration protocol

=> assign IP address to server manually 10.0.0.1

-> configure router, assign IP address to router 10.0.0.2

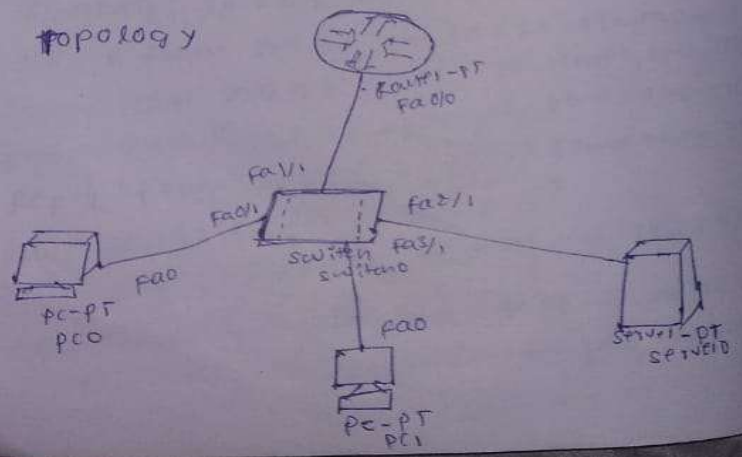
-> Set gateway for server (routers interface addr)

-> configure DHCP protocol
server -> service -> DHCP -> turn on serv.

poolname : serverpool
Default : 10.0.0.2
DNS : 10.0.0.1

-> Dynamically configure the end devices by selecting DHCP in configuration tab of devices

Topology



check ping ⇒ command prompt

Ping 10.0.0.4

pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4 : bytes = 32 time=0ms TTL=128

Reply from 10.0.0.4 : bytes = 32 time=0ms TTL=128

Reply from 10.0.0.4 : bytes = 32 time=0ms TTL=128

Reply from 10.0.0.4 : bytes = 32 time = 2ms TTL=128

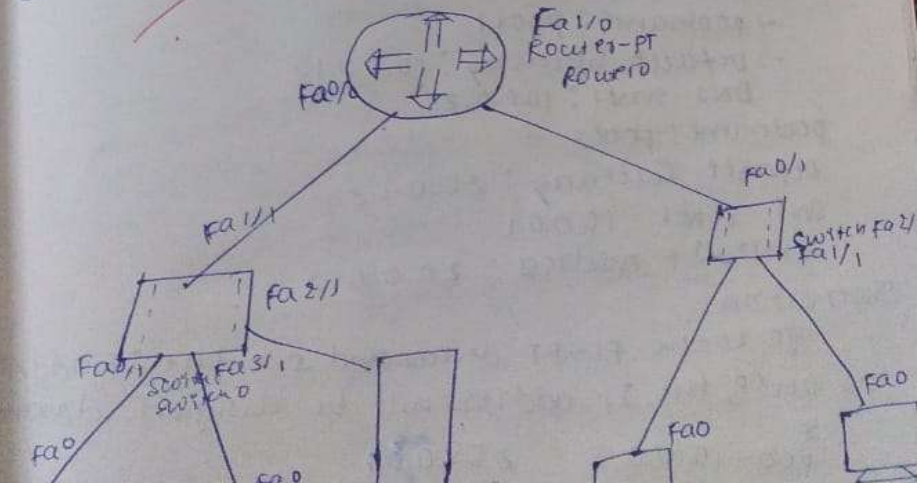
ping statistics for 10.0.0.4

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

Approximate round trip times in milli-seconds:

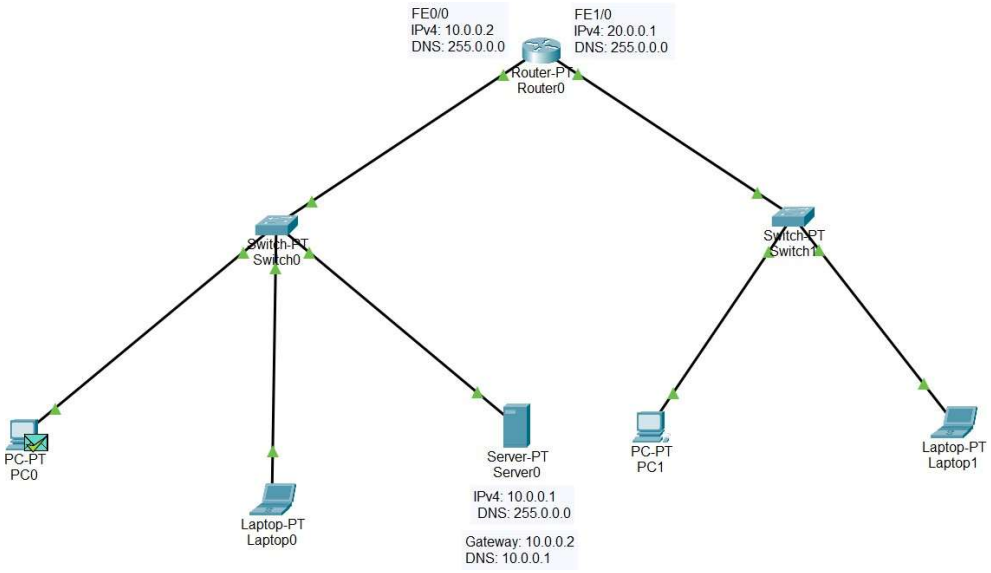
minimum = 0ms, maximum = 2ms, Avrige = 0ms

B)



LABORATORY PROGRAM – 7(B)

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



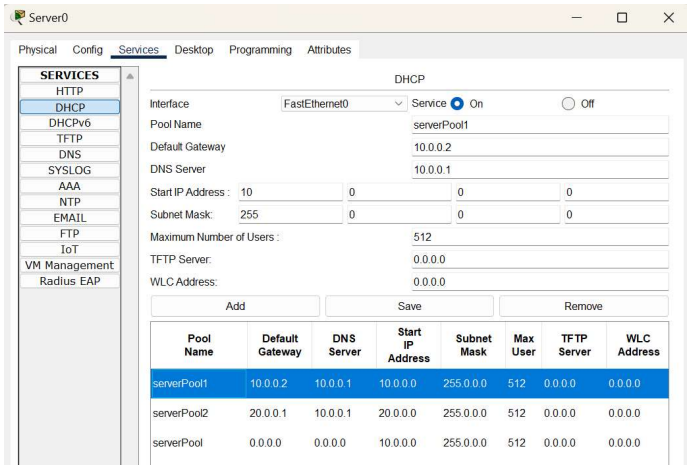
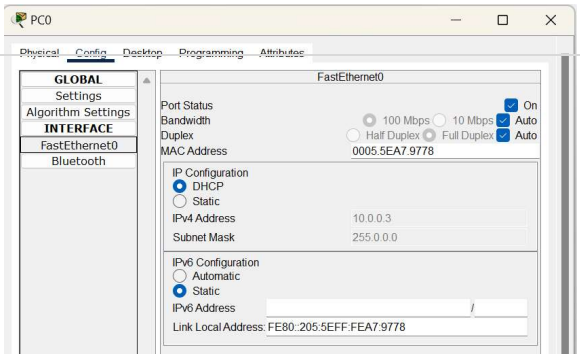


Figure 7.1.1: DHCP Service, Server0

Figure7.2.2: DHCP Service, PC0



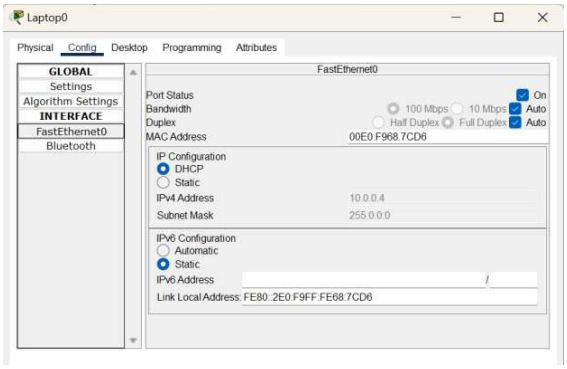


Figure 7.2.3: DHCP Service, Laptop0

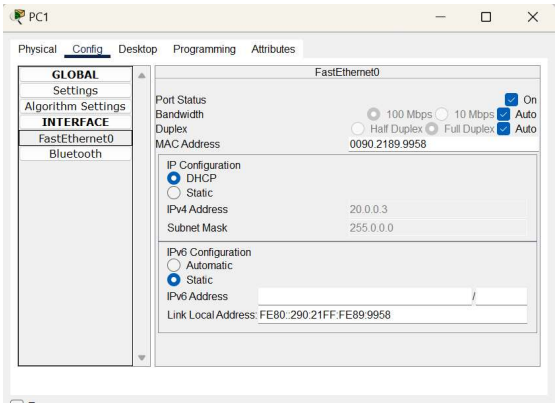


Figure 7.2.4: DHCP Service, PC1

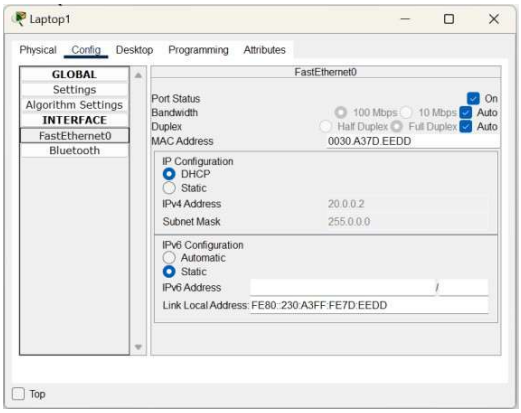
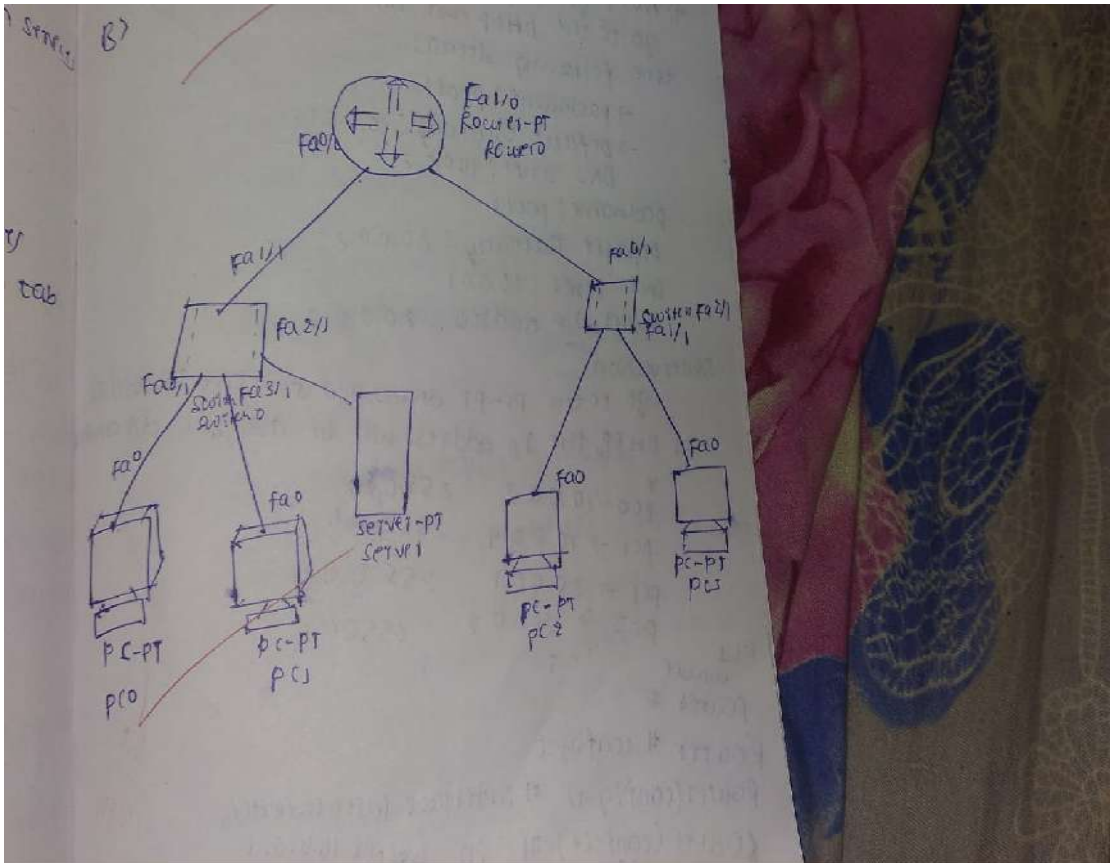


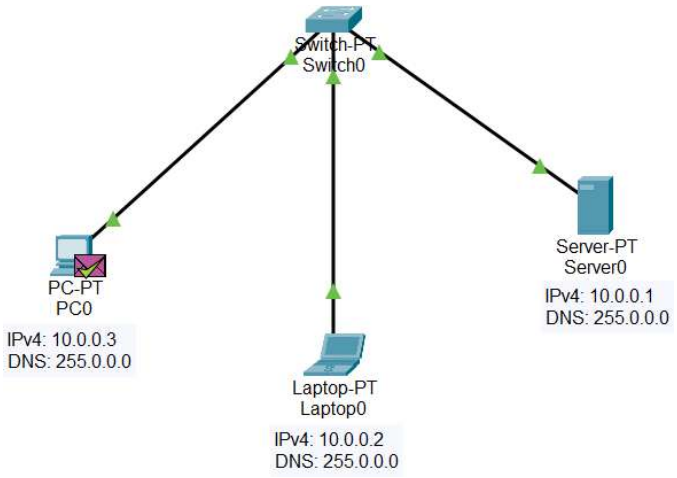
Figure 7.2.5: DHCP Service, Laptop1

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)	


Fa1/0
Router-PT
Router0
Fa0/0
Fa1/1
Fa0/1
Switch=Fa2/1
Fa1/1
Fa2/1
Fa0/1
Switch=Fa3/1
Switch=Fa0
Fa3/1
Fa0
Fa0
Fa0
Fa0
Fa0
server-PT
Server1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3

LABORATORY PROGRAM – 8

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



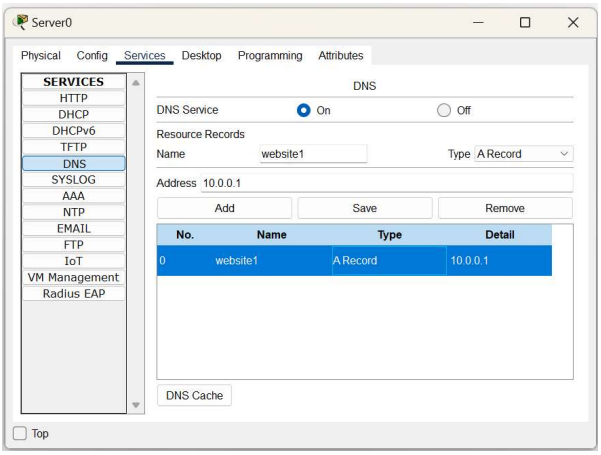


Figure 8.1: DNS Service, Server0

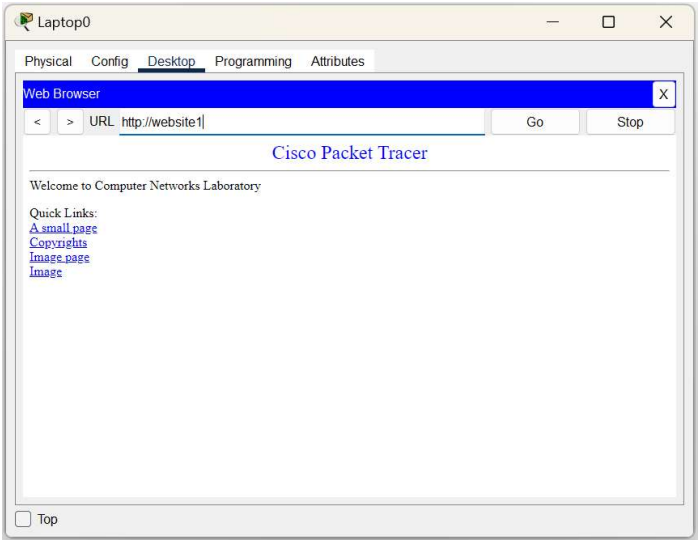
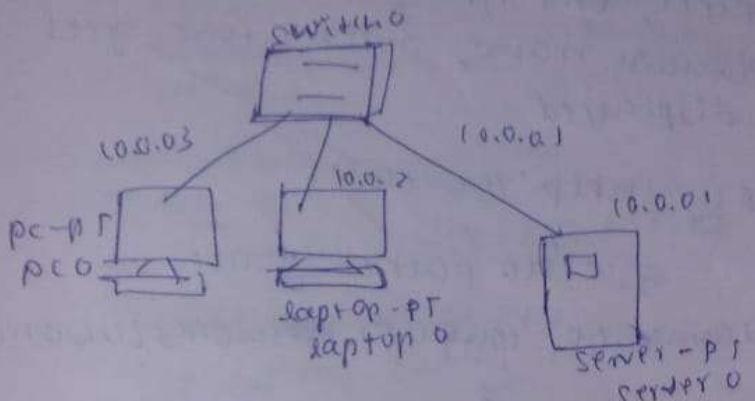


Figure 8.2: DNS Service, Laptop0

exit

LAB Program-8

AIM: TO configure DNS server, to demonstrate the mapping of IP address and domain names



configuration

connection: copper straight through

At server 0:

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 server

Name: website

Address: 10.0.0.1

In HTTP:

- edit index.html, as per your requirement

In PC0: click on DHCP, the IP address get configured automatically

- In laptop: on clicking DHCP, the IP address get configured automatically

Thus all the configuration are done

Destination:

- using any end device, click on "web browser" in desktop menu
- Then, on typing website, the domain name, index.html gets displayed

URL: http://website

cisco packet tracer

welcome to computer networks laboratory

Quick Links

A small page

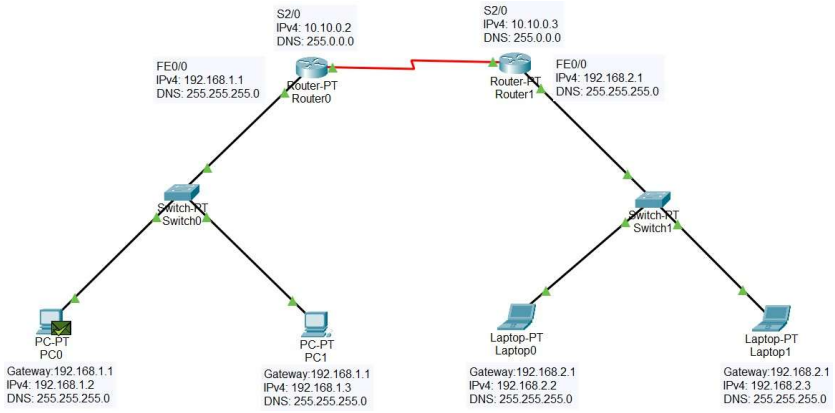
copy rights

Image page

Image

LABORATORY PROGRAM – 9

To Configure RIP routing protocol in Routers.



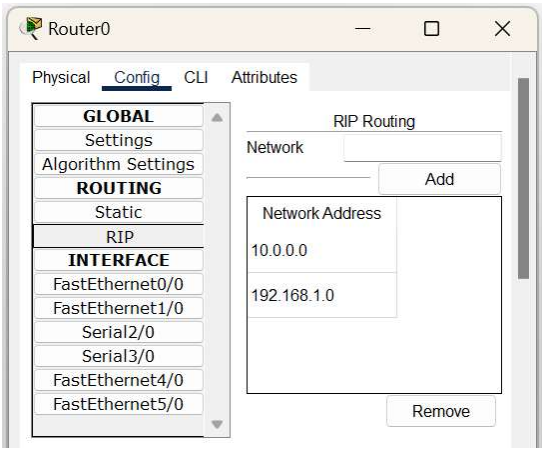


Figure 9.1: RIP, Router0

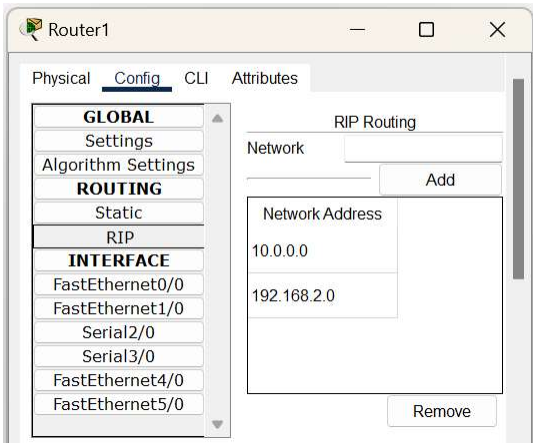
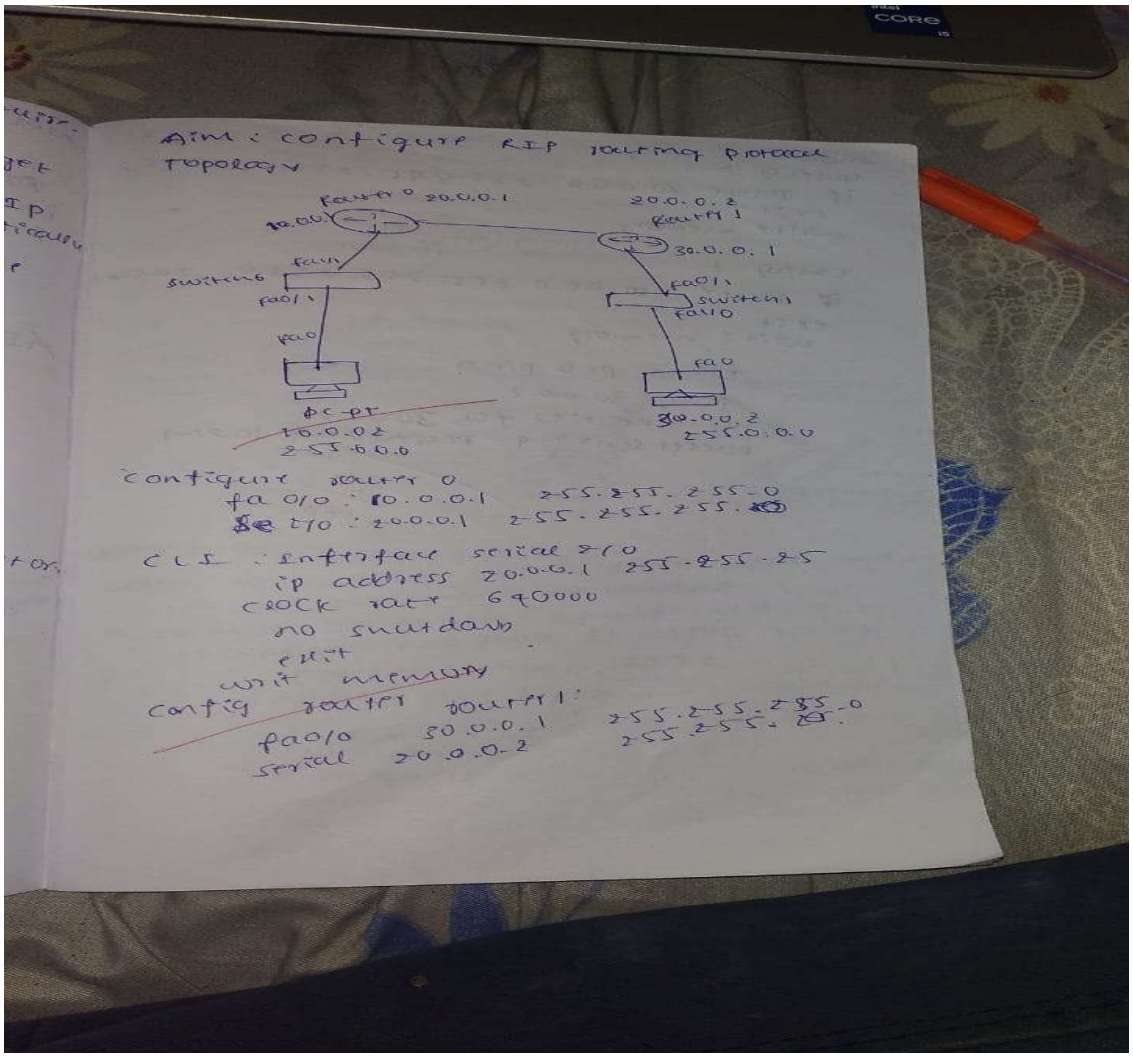
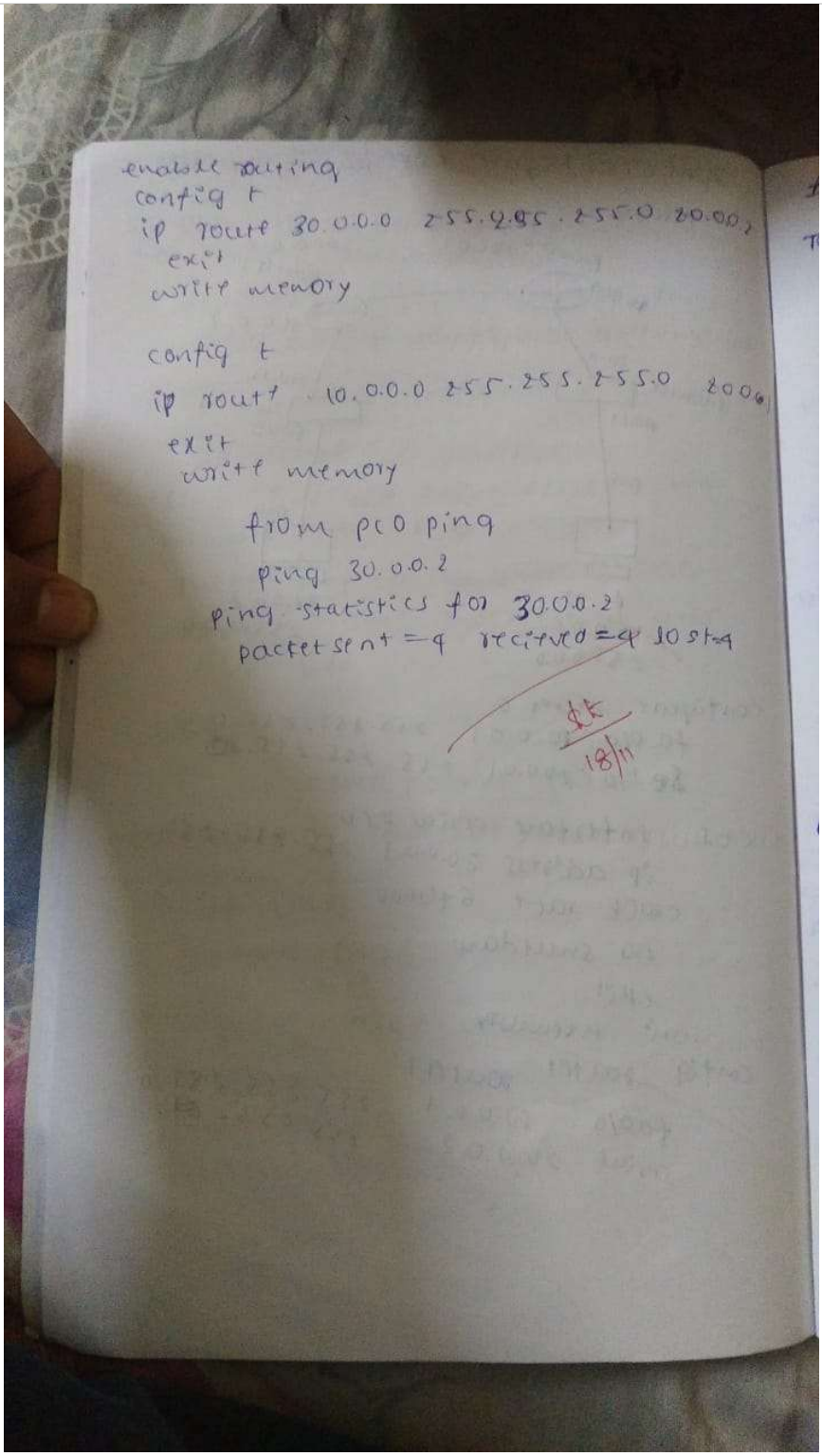


Figure 9.2: RIP, Router

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
```





enable routing
config t
ip route 30.0.0.0 255.255.255.0 20.0.0.2
exit
write memory

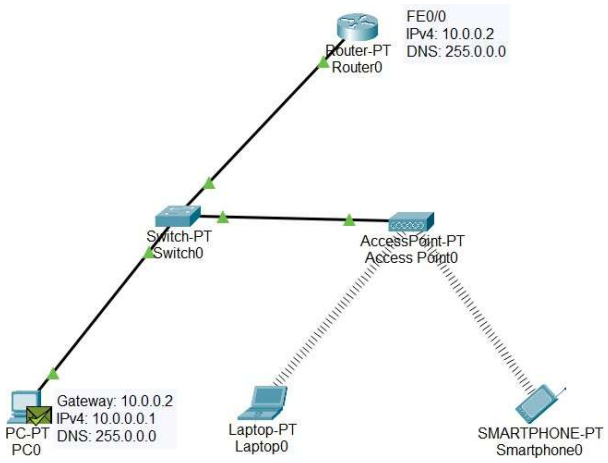
config t
ip route 10.0.0.0 255.255.255.0 20.0.0.1
exit
write memory

from pc0 ping
ping 30.0.0.2
Ping statistics for 30.0.0.2
packet sent = 4 received = 4 lost = 0

18/11

LABORATORY PROGRAM – 10

To demonstrate communication between two devices using a wireless LAN.



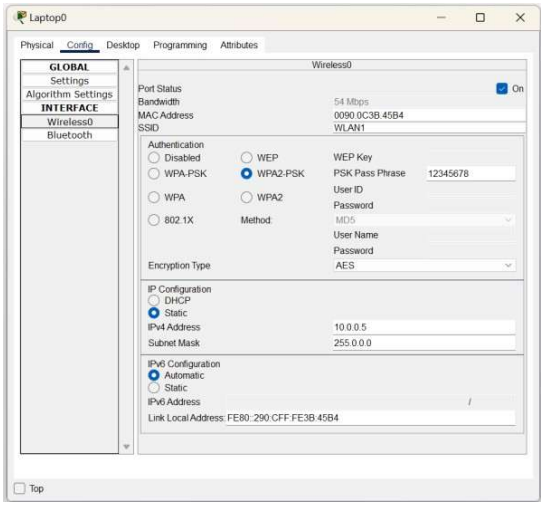


Figure 10.1: Laptop0, Wireless0

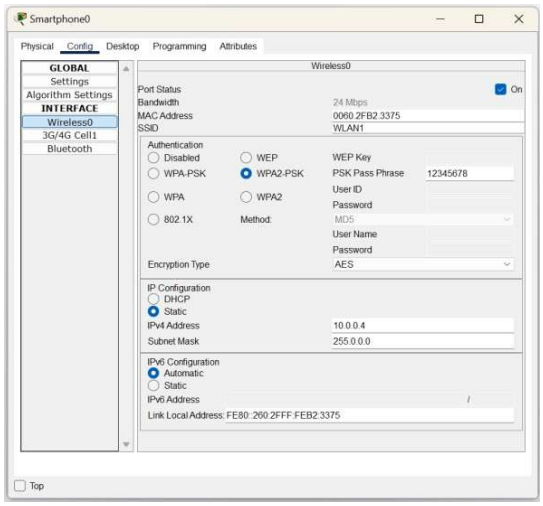
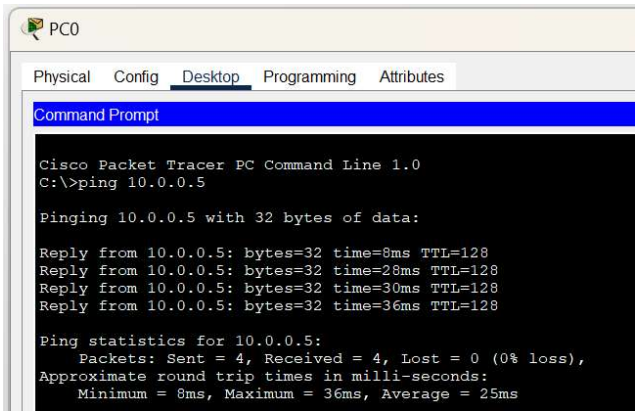
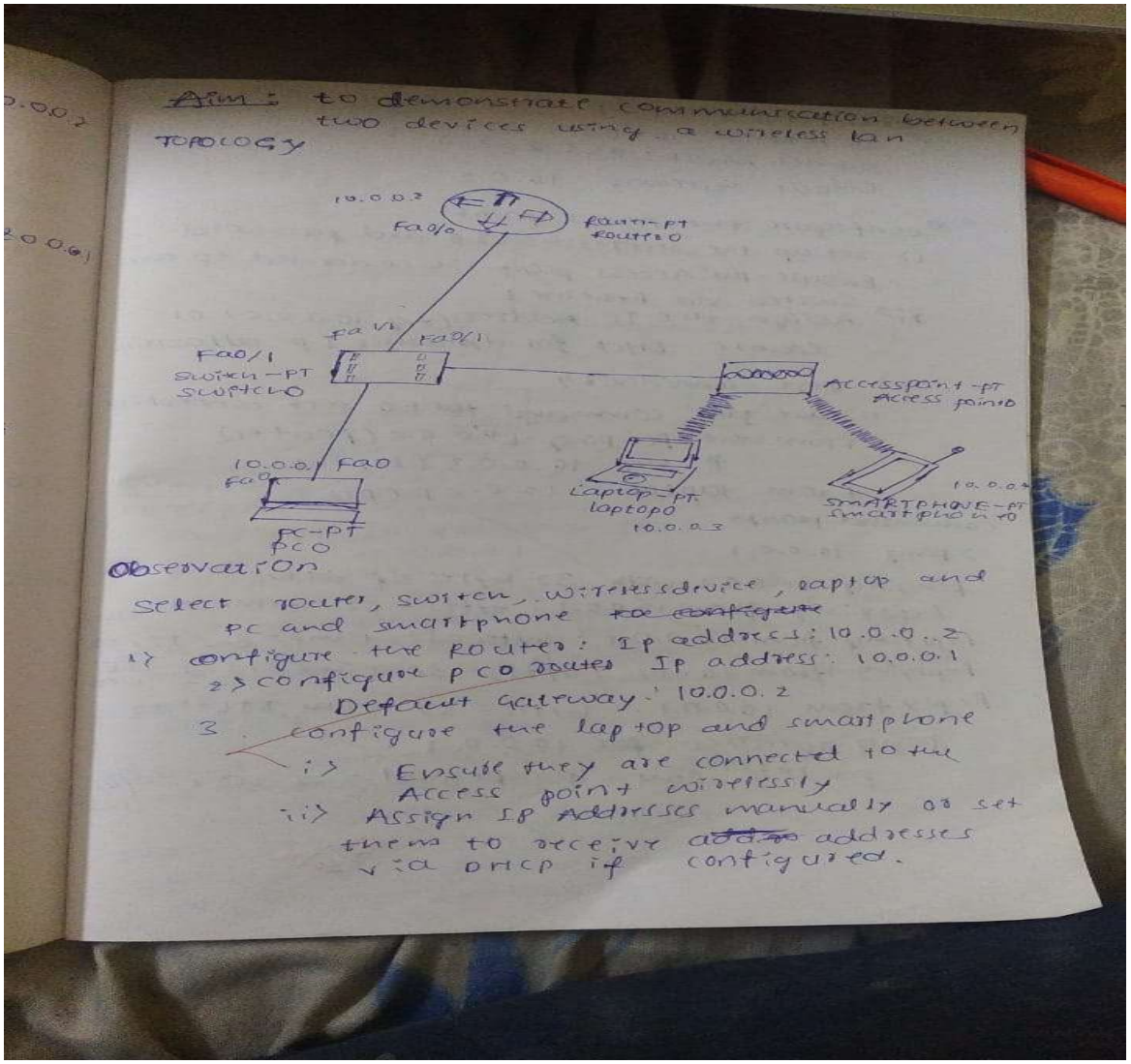


Figure 10.2: Smartphone0, Wireless0

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





Example for laptop:

Ip Address : 10.0.0.3

Subnet mask : 255.255.255.0

Default gateway : 10.0.2

4) Configure the Access point

i> set up the wireless SSID and password

- Ensure the Access point is connected to the Switch via Ethernet

ii> Assign the IP Address(e.g 10.0.0.5) or enable DHCP for dynamic IP allocation

5) Test connectivity

i) use ping command to test connectivity

From the pc: ping 10.0.0.2 (Router)

ping 10.0.0.3 (laptop)

from laptop : 10.0.0.1 (pc)

command prompt:

> ping 10.0.0.1

pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1 : bytes = 32 time=2ms

Reply from 10.0.0.1 : bytes = 32 time=8ms

Reply from 10.0.0.1 : bytes = 32 time=13ms

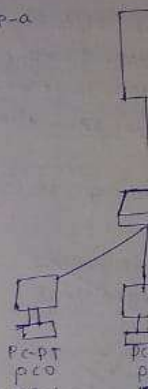
Reply from 10.0.0.1 : bytes = 32 time, TTL=

ping statistics for 10.0.0.1 :

packets: sent = 4, received = 4, lost = 0

Aim: To demonstrate resolution protocol within a LAN

ARP-a



Observation:

configure pc's

cointigure ip to

pc0 => 10.0

pc1 => 10.0

pc2 => 10.

To check ARP

in command p

you can see

and also conf

10.0.0.4

+ select simple p

destination,

the result

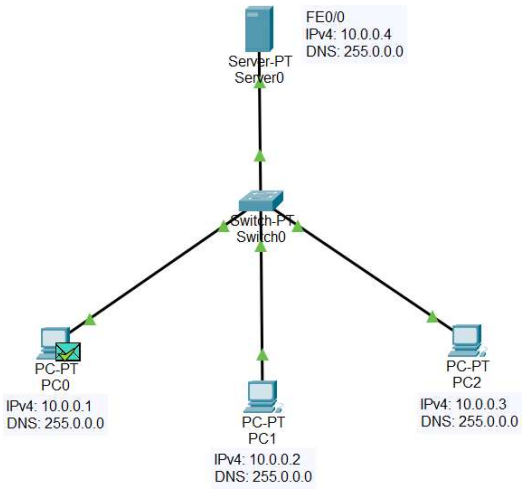
then click on

devices then

you can see

LABORATORY PROGRAM – 11

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



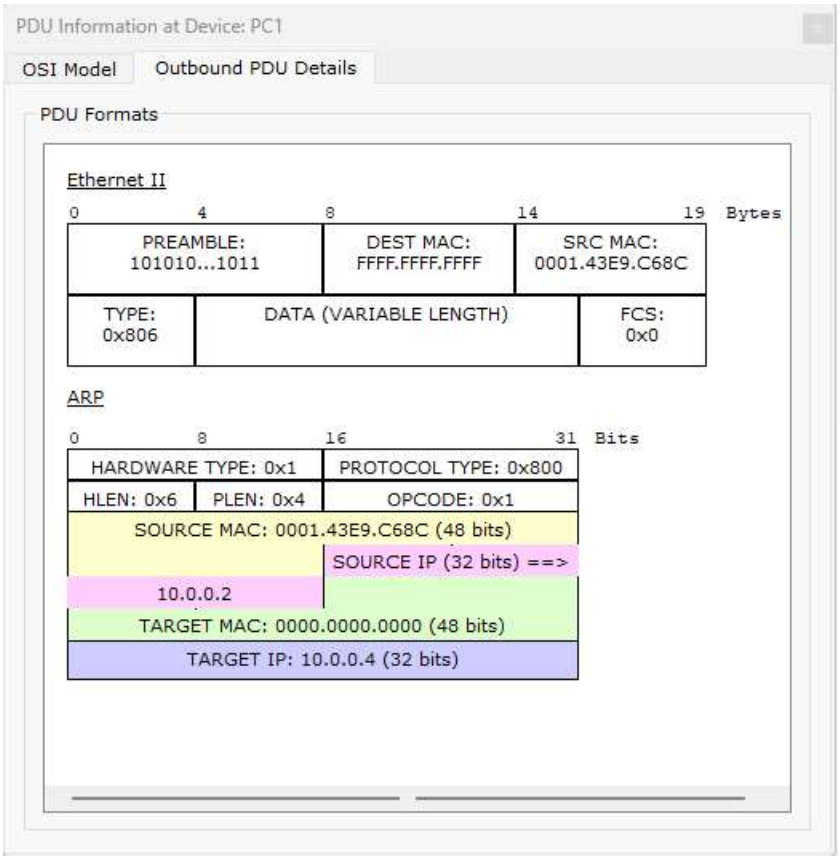


Figure 11.1: Inbound ARP, PC1

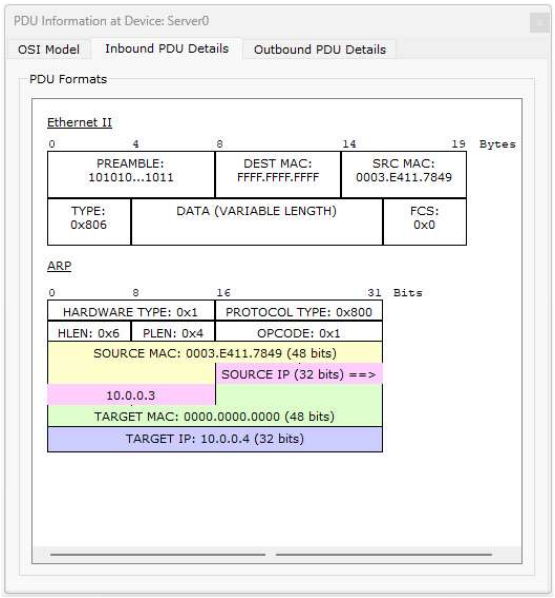


Figure 11.2: Inbound ARP, Server0

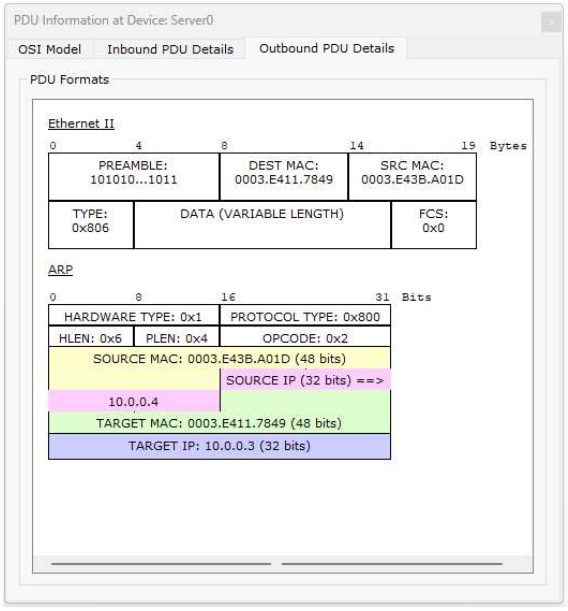


Figure 11.3: Outbound ARP, Server0

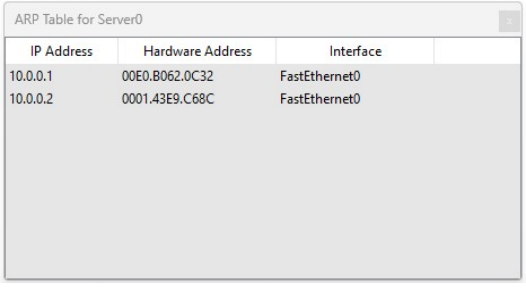


Figure 11.4: ARP Table, Server0

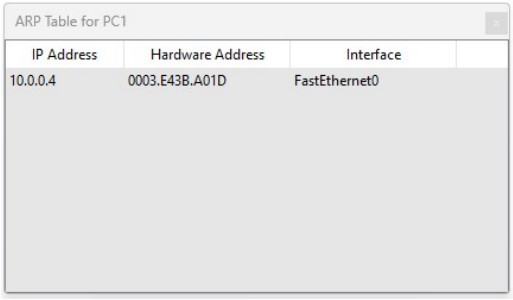
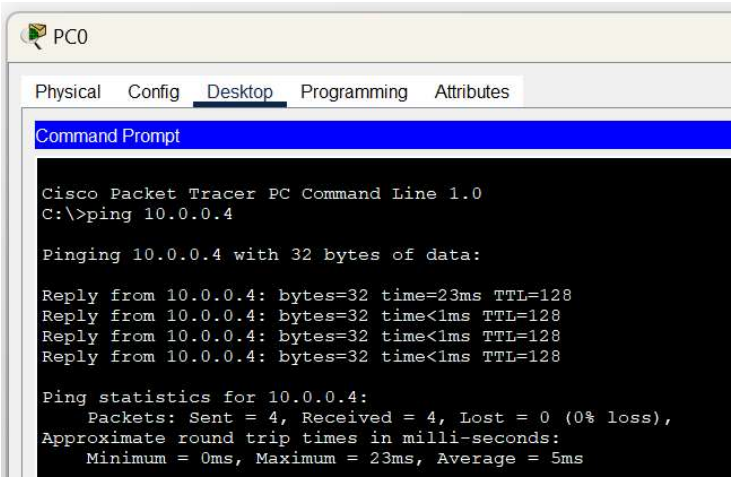


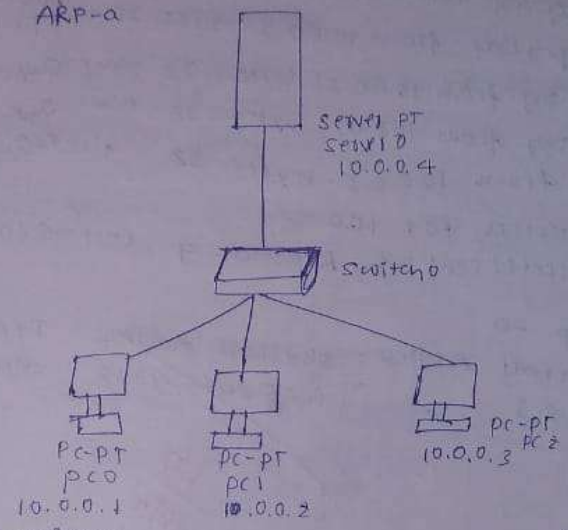
Figure 11.5: ARP Table, PC1

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



Aim: To demonstrate the working of Address resolution protocol for communication within a LAN

ARP-a



Observation:

configure PC's address are shown in figure

configure ip for each pc

PC0 ⇒ 10.0.0.1

PC1 ⇒ 10.0.0.2

PC2 ⇒ 10.0.0.3 Server0 ⇒ 10.0.0.4

To check ARP table execute command arp -a in command prompt after pinging you can see the arp table.

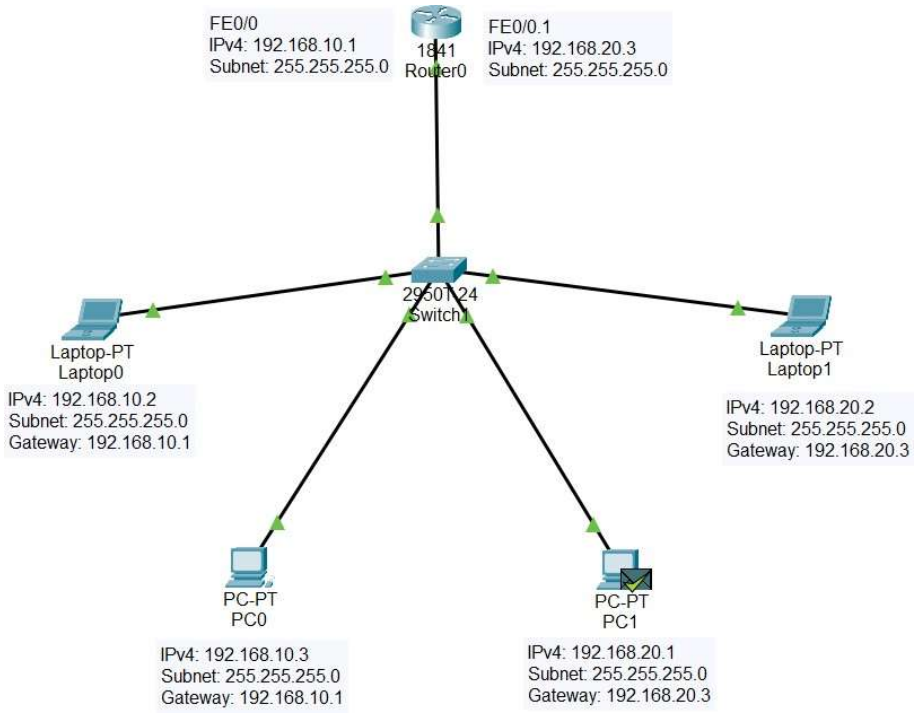
and also configure server with ip address 10.0.0.4

* select simple PDU and choose source and destination, you can also capture to see the results.

then click on the ☑ mark on the source devices then goto inbound device and you can see ARP table

LABORATORY PROGRAM – 12

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



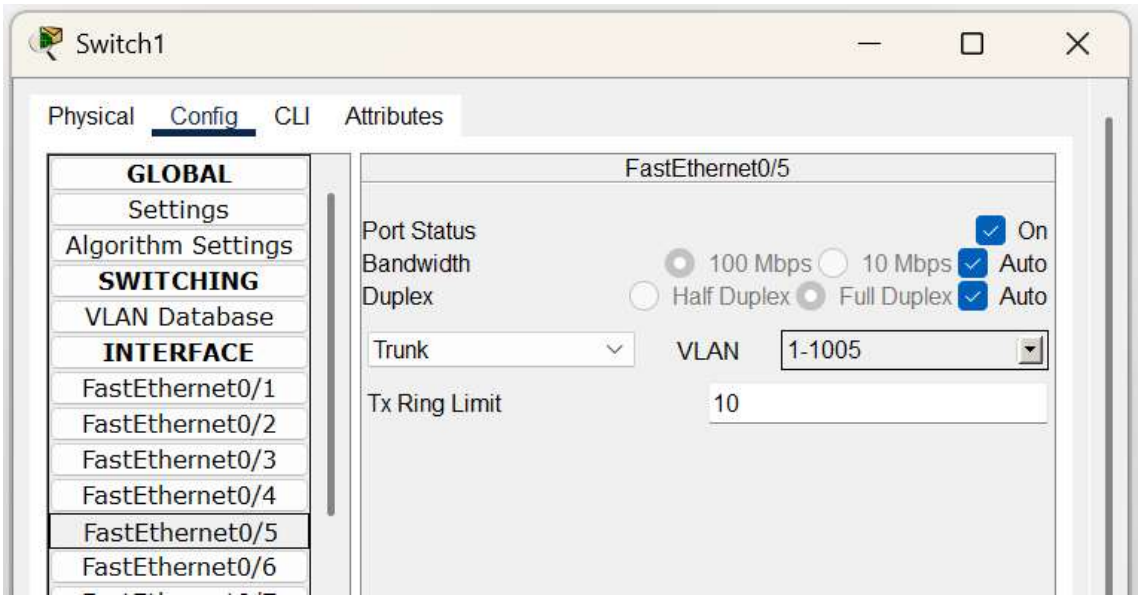


Figure 12.1: FE0/5 Switchport Trunk

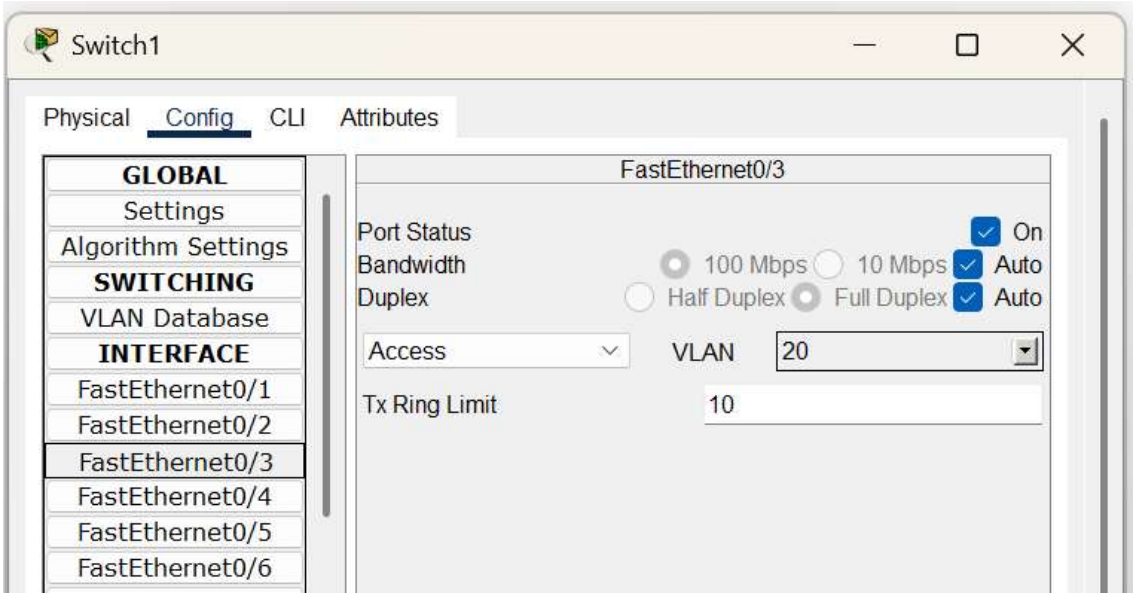


Figure 12.2: FE0/3 Switchport Access

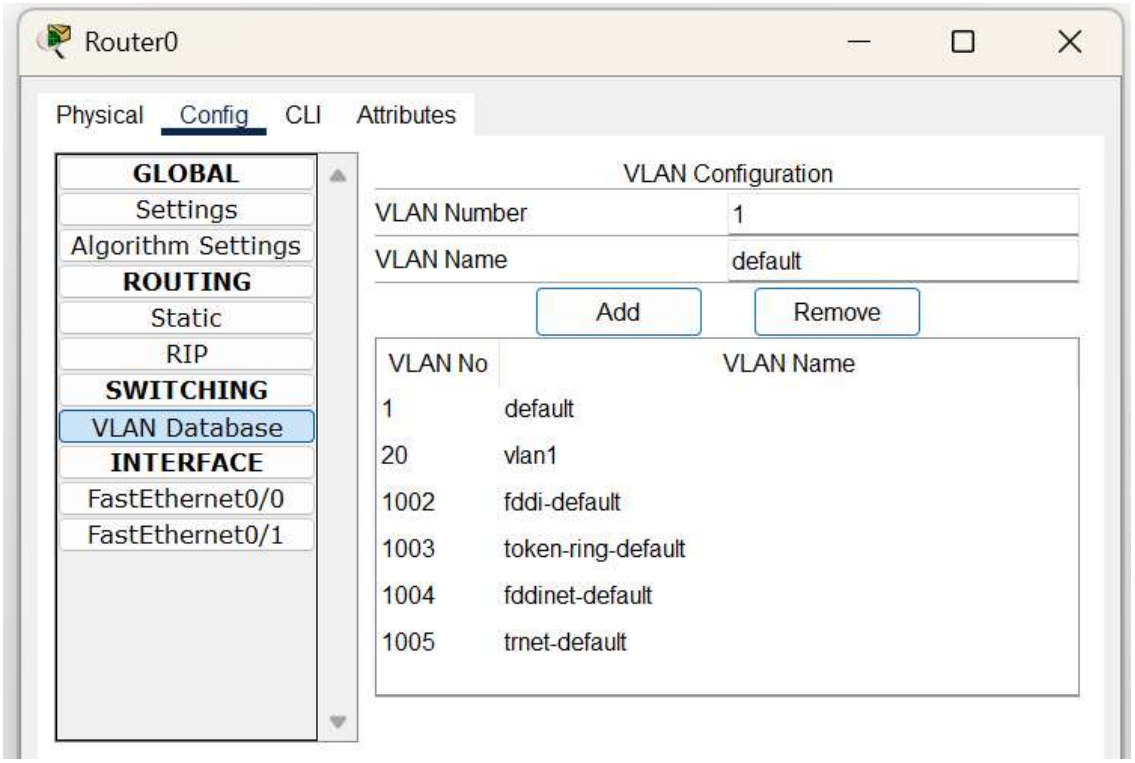


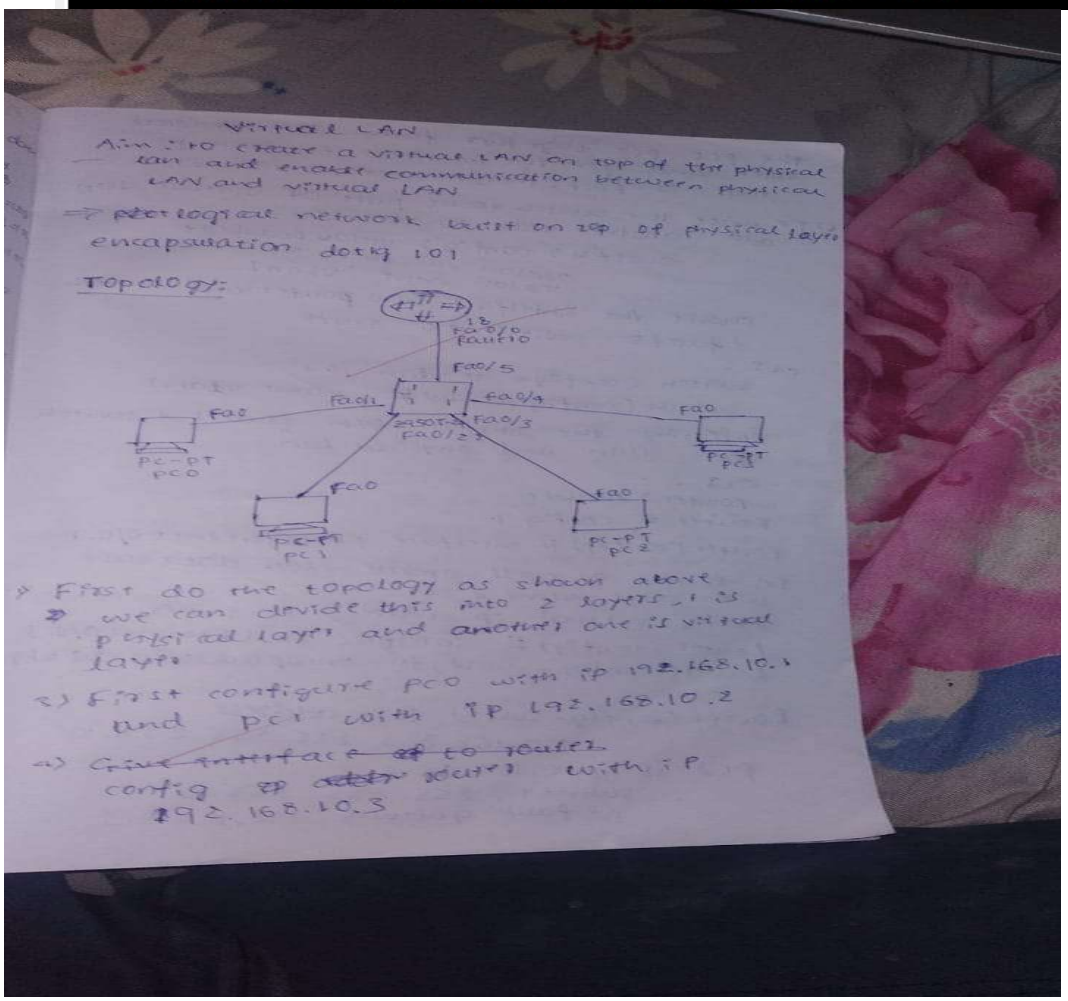
Figure 12.3: Router0 VLAN Database

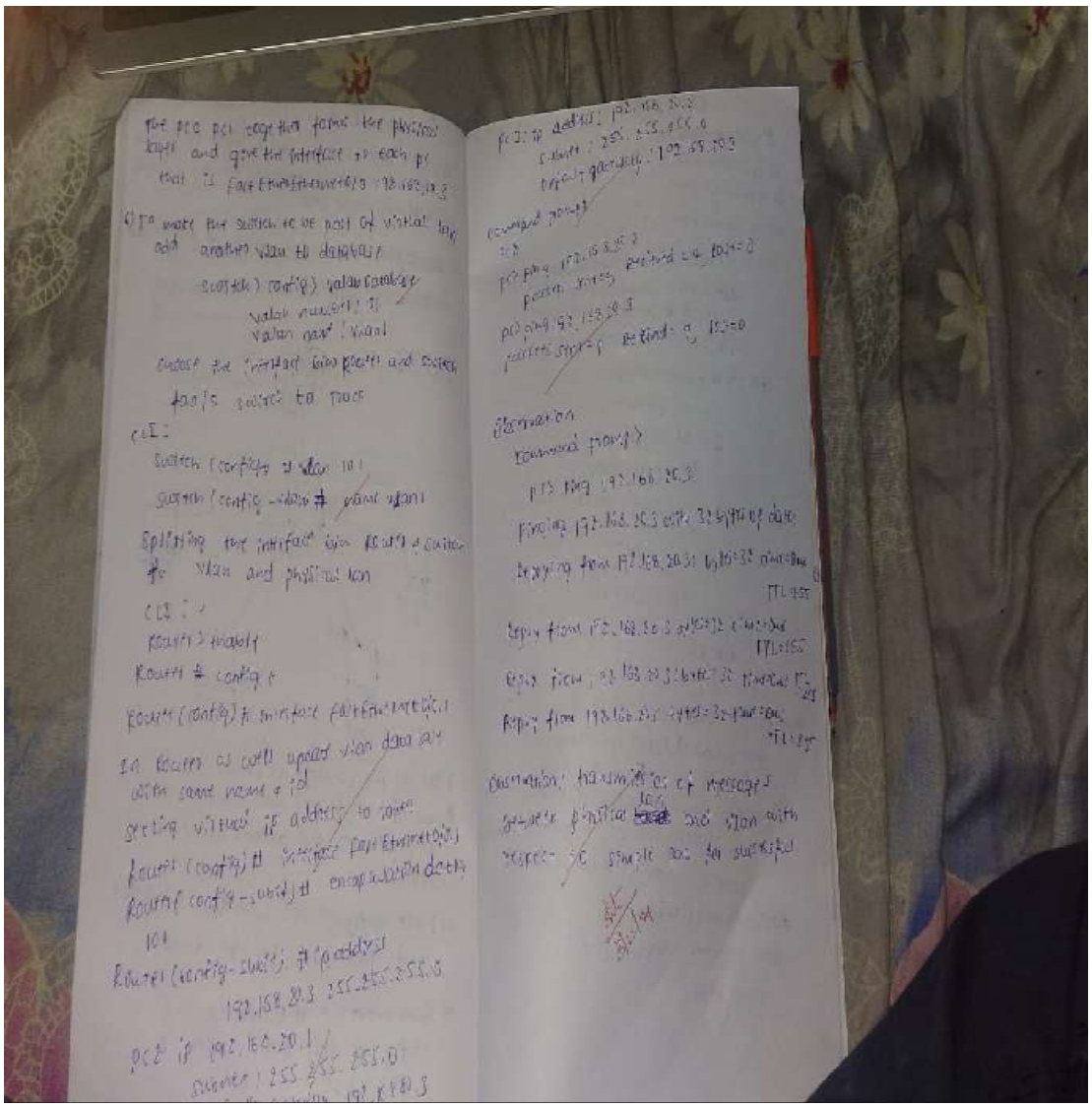
```
Router(config)#interface FastEthernet0/0.1
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown
```

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
```





LABORATORY PROGRAM – 13

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

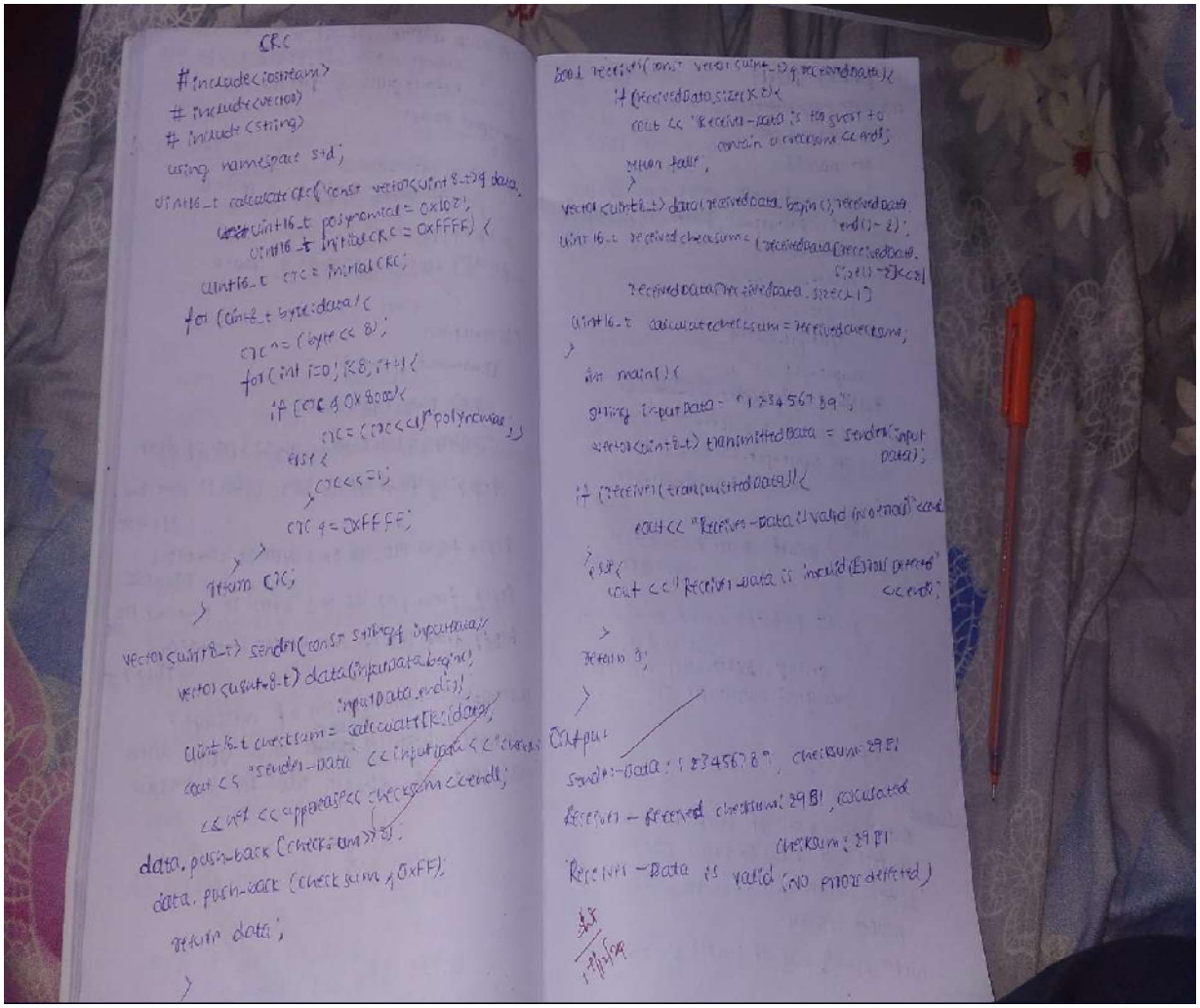
Code

```
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
```



CRC

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
uint16_t calculateCRC(const vector<uint8_t>& data,
    uint16_t polynomial = 0x1021,
    uint16_t initialCRC = 0xFFFF) {
uint16_t crc = initialCRC;
for (uint8_t byte:data){
    crc ^= (byte << 8);
    for(int i=0; i<8; i++){
        if (crc & 0x8000){
            crc = (crc<<1)^polynomial;
        }else{
            crc<<=1;
        }
        crc &= 0xFFFF;
    }
}
return crc;
}
vector<uint8_t> sender(const string& inputData){
vector<uint8_t> data(inputData.begin(),
                     inputData.end());
uint16_t checksum = calculateCRC(data);
cout << "sender-data: " << inputData << "\n"
<< hex << uppercase << checksum << endl;
data.push_back((checksum>>8));
data.push_back(checksum & 0xFF);
return data;
}
```

```
bool receiver(const vector<uint8_t>& receivedData){
    if (receivedData.size()<2){
        cout << "Received-data is too short to
                contain a checksum" << endl;
        return false;
    }
vector<uint8_t> data(receivedData.begin(), receivedData.
                     end()-2);
uint16_t receivedChecksum = (receivedData[receivedData.
                            size()-2]<<8)|
    receivedData[receivedData.size()-1];
uint16_t calculatedChecksum = receivedChecksum;
}
int main(){
string inputData = "123456789";
vector<uint8_t> transmittedData = sender(inputData);
if (receiver(transmittedData)){
    cout << "Received-data is valid (no error)" << endl;
}else{
    cout << "Received-data is invalid (Error detected)"
         << endl;
}
return 0;
}
```

Output

sender-data: 123456789, checksum: 29B1

Receiver - Received checksum: 29B1, calculated checksum: 29B1

Received-Data is valid (no error detected)

17/9/24

LABORATORY PROGRAM – 14

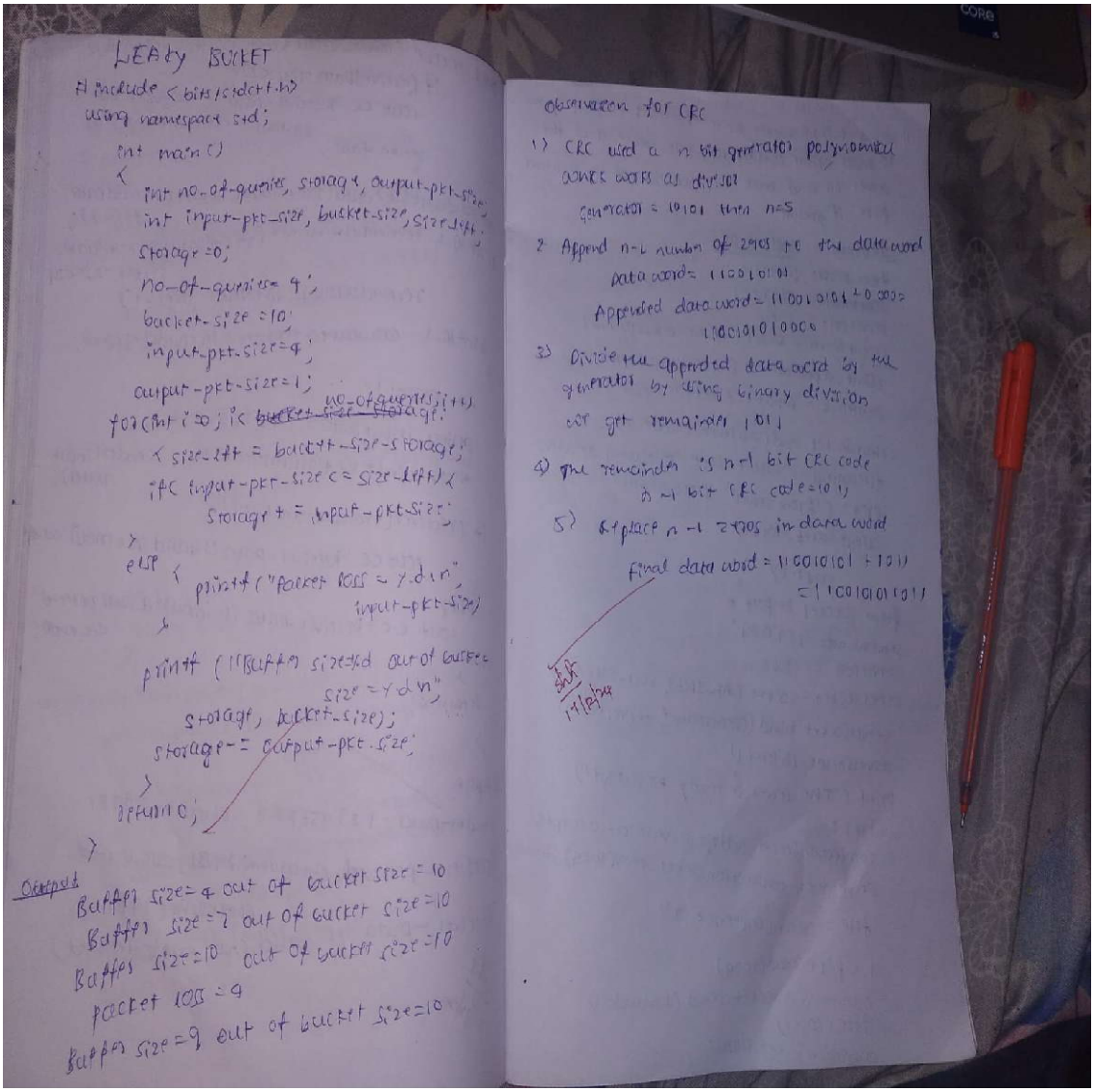
Write a program for congestion control using Leaky bucket algorithm.

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
```



LEAky BUCKET

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int no_of_queries, storage, output_pkt_size;
    int input_pkt_size, bucket_size, size_left;
    storage=0;
    no_of_queries= 9;
    bucket_size =10;
    input_pkt_size=4;
    output_pkt_size=1;
for(int i=0; i< no_of_queries;i++)
    {
    size_left = bucket_size-storage;
    if( input_pkt_size <= size_left){
        storage += input_pkt_size;
    }
    else {
        printf("packet loss = %d\n",
                input_pkt_size)
    }
    printf("Buffer size=%d out of bucket
            size = %d\n",
        storage, bucket_size);
    storage -= output_pkt_size;
    }
return 0;
}
```

Output

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

observation for CRC

1) CRC used a n bit generator polynomial which works as divisor
   generator = 10101 then n=5
2) Append n-1 number of zeros to the data word
   data word = 11001001
   Appended data word = 11001001 + 0000 =
   110010010000
3) Divide the appended data word by the generator by using binary division
   or get remainder 101
4) The remainder is n-1 bit CRC code
   i.e 4 bit CRC code = 101
5) Replace n-1 zeros in data word
   Final data word = 11001001 + 101
   = 110010010101

skip
replace

LABORATORY 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

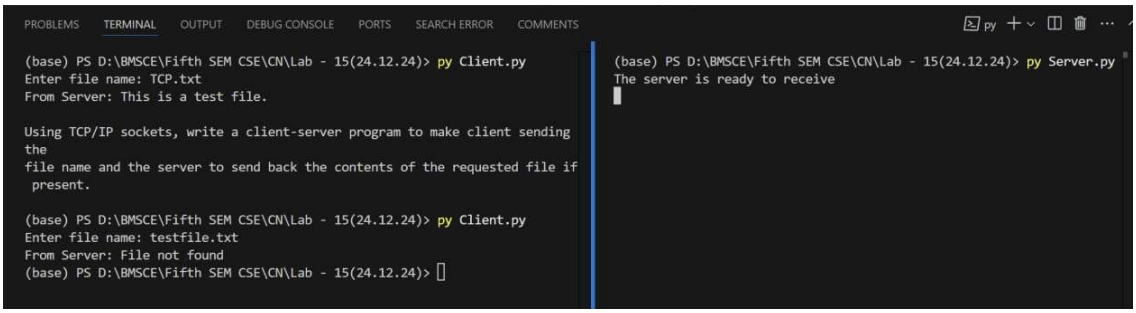
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# 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()
```

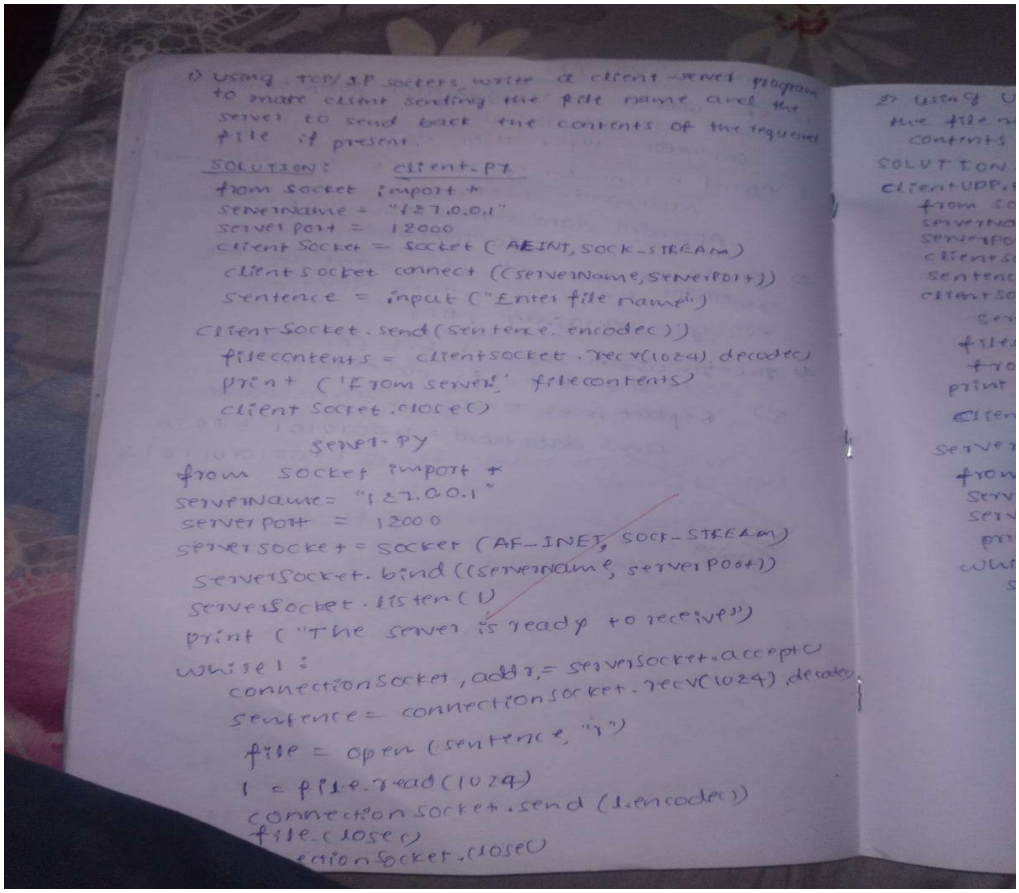
Code: Server.py

```
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





LABORATORY 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()
```

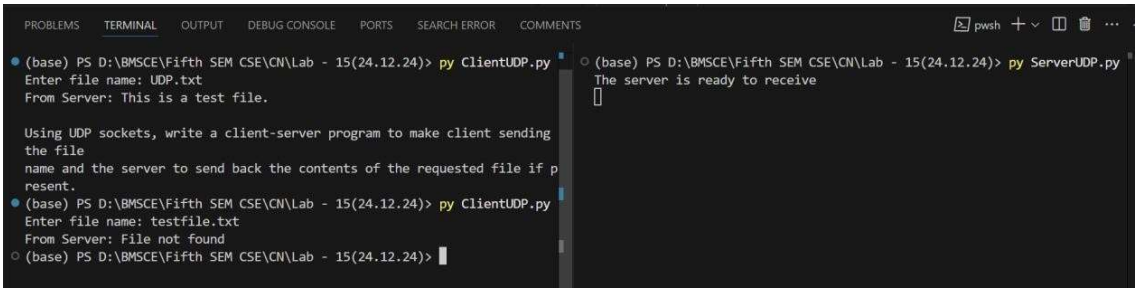
Code: ServerUDP.py

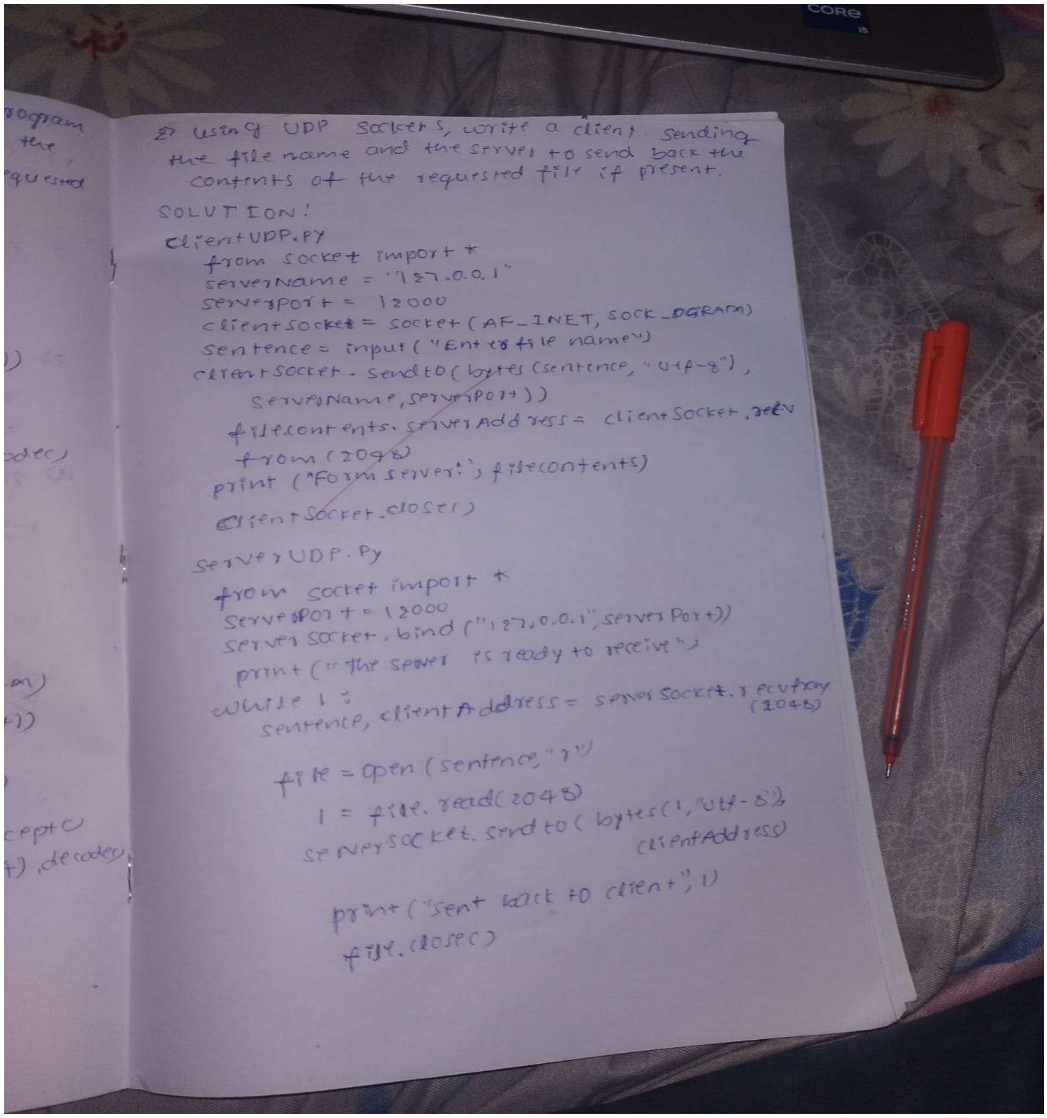
```
from socket import *
```

```
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





2) Using UDP sockets, write a client sending the file name and the server to send back the contents of the requested file if present.

SOLUTION:

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.recv
    from(2048)
print("From server:", filecontents)
clientSocket.close()
```

ServerUDP.Py

```
from socket import *
serverPort = 12000
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, "utf-8"),
                        clientAddress)
    print("sent back to client", l)
    file.close()
```