VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

SYED FARHAN (1BM23CS424)

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 **SYED FARHAN (1BM23CS424),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

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

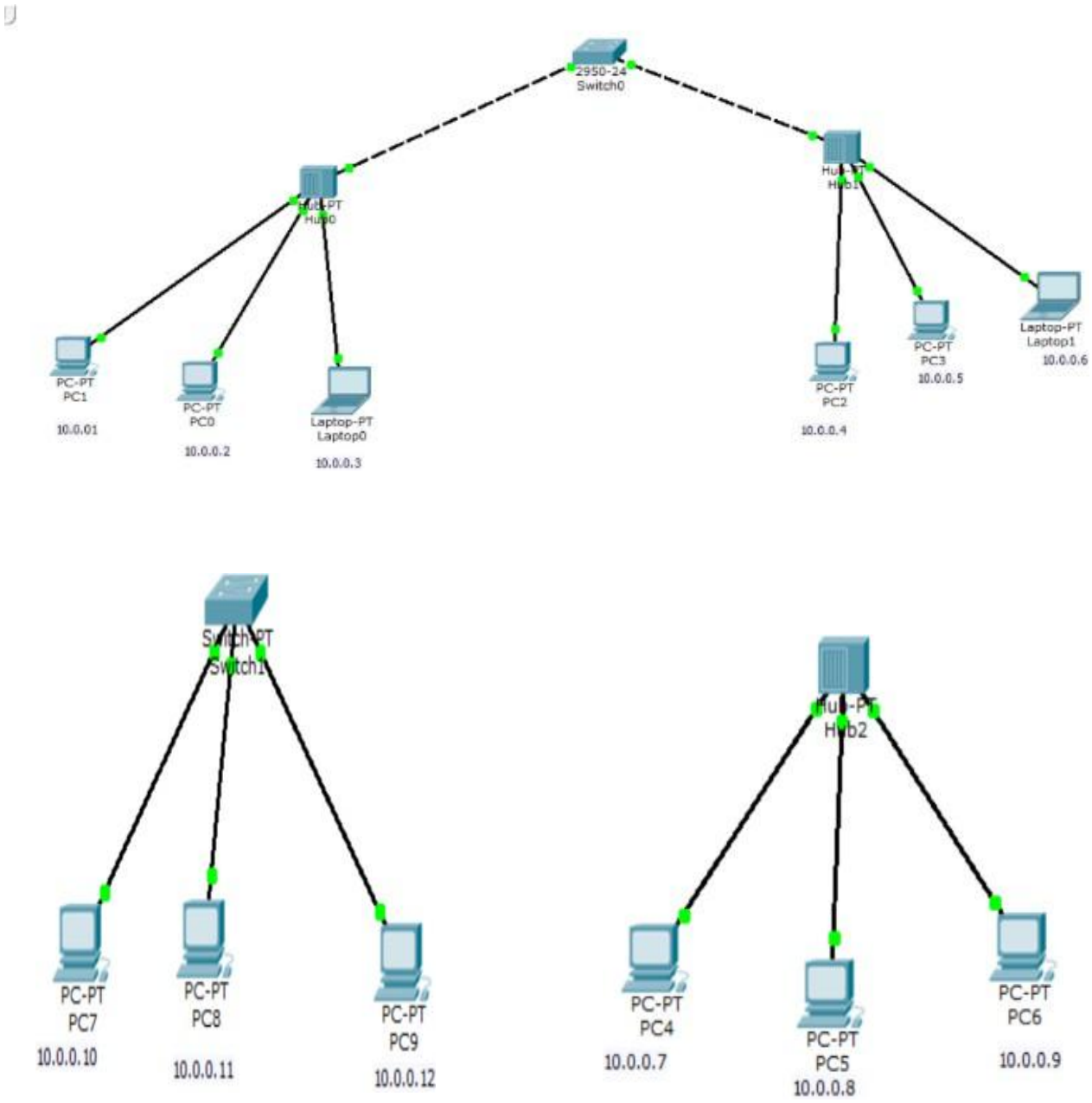
Index

Github Link : https://github.com/Syed-Farhan-bmsce/CN.git

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 messages.

Topology:



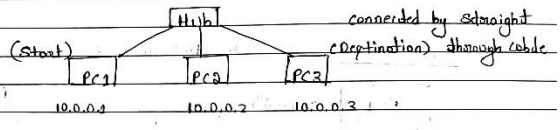
Procedure and Observations:

Hub / Switch.

Aim: Simulate transmission of simple PDU using hub and switch as connecting devices using a Cisco Packet Tracer.

Devices used:
Hub, Switch, and End Devices.

Topology 1:
Hub and 3 End Devices.

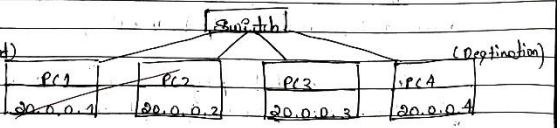


Procedure and Observations.

i. Connect end devices PC1, PC2, and PC3 to the hub through straight cable.
ii. Assign IP Address to each of the end devices.
iii. Select a simple PDU, select PC1 as start node and PC3 as destination.

During simulation, the message will be received by PC3 by PC1 and acknowledges the same.

Topology 2: Switch and End Devices.

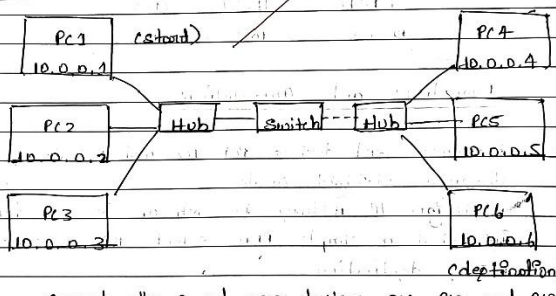


Connect 4 end devices PC1, PC2, PC3, PC4 to the switch with mentioned IP Address.

Select simple PDC, PC1 as start and PC4 has destination and Simulate

Connection to be made through straight through cables. The message will be sent from PC1 to PC4 and in return the acknowledgement will be sent from PC4 to PC1

Topology 3: Switch, Hub and End devices



Connect the 3 end user devices PC1, PC2 and PC3 with mentioned IP addresses to a hub and further connected to a switch.

The connection blw the hub and switch is through a cross over cable.

Then connect switch to another hub with 3 end user devices with mentioned IP address

The successful ping message is confirming the connectivity blw source & destination

classmate
Date
Page 03

Difference between Hub / Switch and End device.

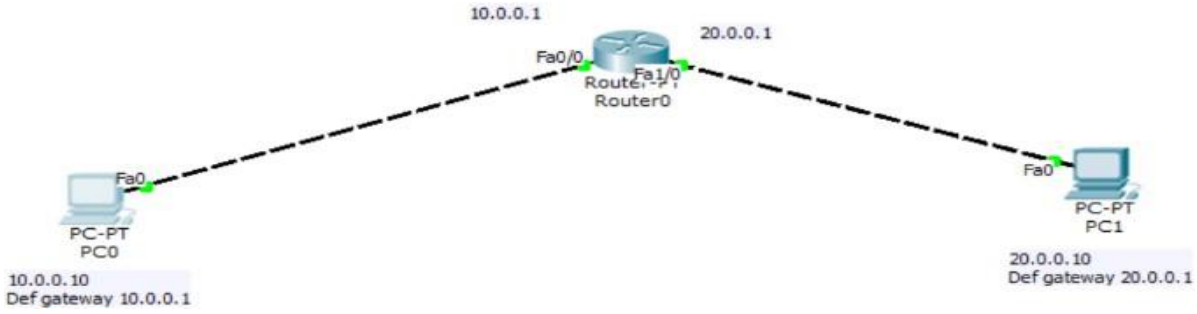
Component	Hub	Switch	End Device
Function	connects multiple devices broadcasts data to all	connects devices & forwards data selectively.	Source or destination of data in a network.
OSI Layer	Layer 1 physical layer.	Layer 2 Data link layer.	varies, Application, Network.
Data Handling	Broadcasts to all connected devices.	sends data to specific devices using MAC Address.	sends or receives data.
Efficiency	lows layers network Congestion	High reduces congestion	Depends on the device, functionality
Intelligence	No data filtering or decision-making	Filters & forwards data intelligently	No forwarding capability
Use case	Small or Simple network	Modern, larger or more complex	user device, (PC's, phones)

Sub
3/10/24

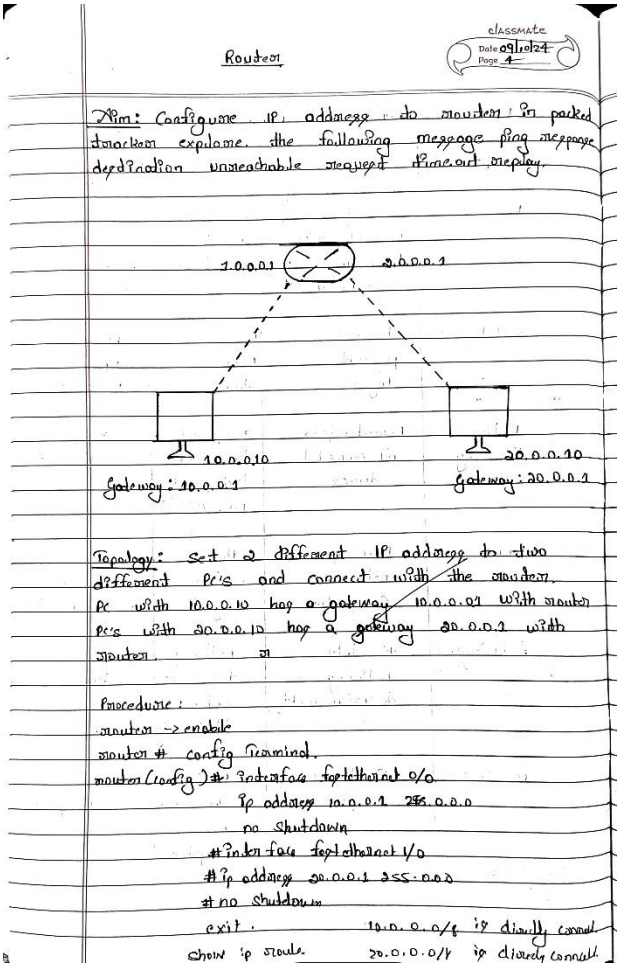
Program 2 :

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

Topology:



Procedure and Observations:


Router

Aim: Configure IP address to routers in packet tracker explore the following message ping response destination unreachable request timeout replay.

10.0.0.1 20.0.0.1

10.0.0.10 20.0.0.10

Gateway: 10.0.0.1 Gateway: 20.0.0.1

Topology: Set 2 different IP address to two different PC's and connect with the router. PC with 10.0.0.10 hop a gateway 10.0.0.1 with router PC's with 20.0.0.10 hop a gateway 20.0.0.1 with router.

Procedure:

router -> enable

router # config terminal.

router(config)# interface fastethernet 0/0

ip address 10.0.0.1 255.0.0.0

no shutdown

#interface fastethernet 1/0

#ip address 20.0.0.1 255.0.0.0

#no shutdown

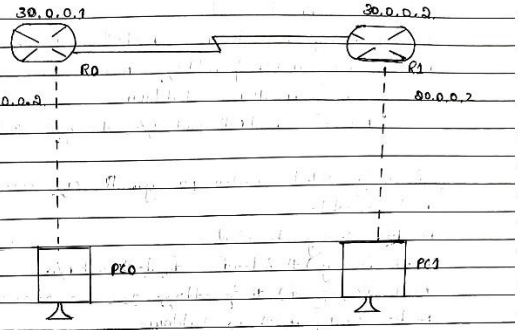
exit.

show ip route. 10.0.0.0/8 is directly connected

20.0.0.0/8 is directly connected

Router.

Aim: Simulate using 2 Router and 2 end devices.



Procedure:

* Select a generic router R1
* Connect an end device PC1 to router R1 through parallel connection fastethernet 0/0
* Configure PC1 with Ip address 10.0.0.1 and gateway 10.0.0
* Similarly select another generic router R2 and connect an end device PC2 fastethernet 1/0.
* Configure PC2 with Ip address 20.0.0.1 and gateway 20.0.0.2.

Now Select router R1 go to CLI and execute the the following

```
Router>enable
Router# config terminal
Router(config)# interface fastethernet 0/0
Router(config-if)# ip address 10.0.0.2 255.0.0.0
Router(config-if)# no shutdown
```

"Interface fastEthernet 0/0 changed state to up"

Similarly Select router R2 go to CLI execute the same

```
Router>enable
Router# config terminal
Router(config)# interface fastethernet 1/0
Router(config-if)# ip address 20.0.0.2 255.0.0.0
Router(config-if)# no shutdown
```

"Interface fastethernet 1/0 changed state to up"

Hence the connection b/w Routers & end devices is established.

Now Connect router R1 with router R2 using Serial cable (Serially Connected)

To Setup connection b/w routers again.

→ Select router R1 and go to CLI

```
Router(config)# interface serial 2/0
Router(config-if)# ip address 30.0.0.1 255.0.0.0
Router(config-if)# no shutdown
```

→ Select router R2 and go to CLI

```
Router(config)# interface serial 3/0
Router(config-if)# ip address 30.0.0.2 255.0.0.0
Router(config-if)# no Shutdown
```

"Interface serial 2/0 changed state to up"

3/0.

Observations:

-> After setting up the mentioned topology Now try to ping PC2 with PC1.

Open command prompt from PC1 type Ping 20.0.0.1

-> Destination host unreachable.

Packets sent:4 received:0 lost=4 Loss=100%.

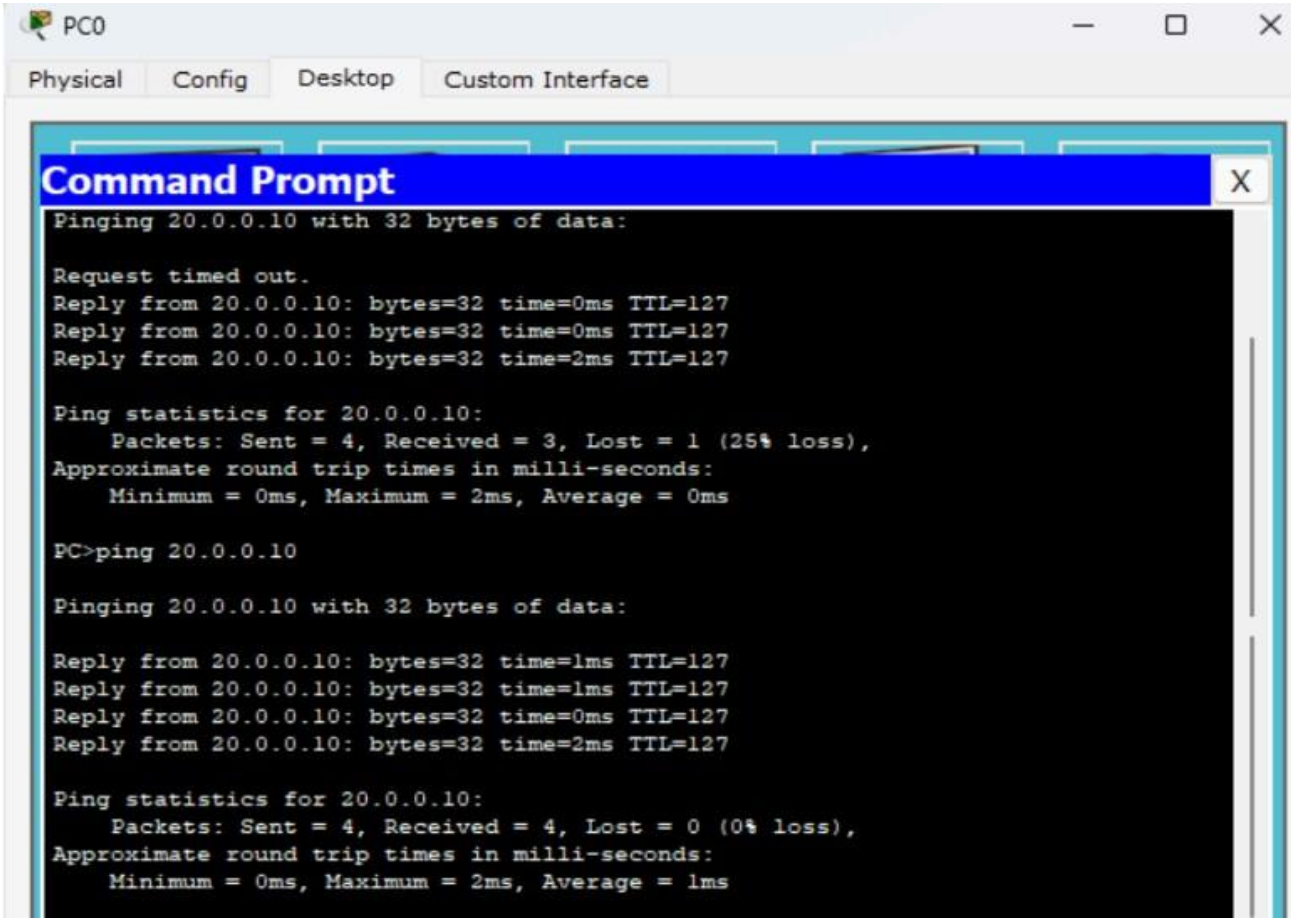
It is also observed that the end system PC1 was only pinged with router R1 only.

ping 20.0.0.1 -> Successful.

Packets sent:4 received:4 lost:0 0%.

Hence although the routers were connected serially. the end devices were unreachable to ping other devices.

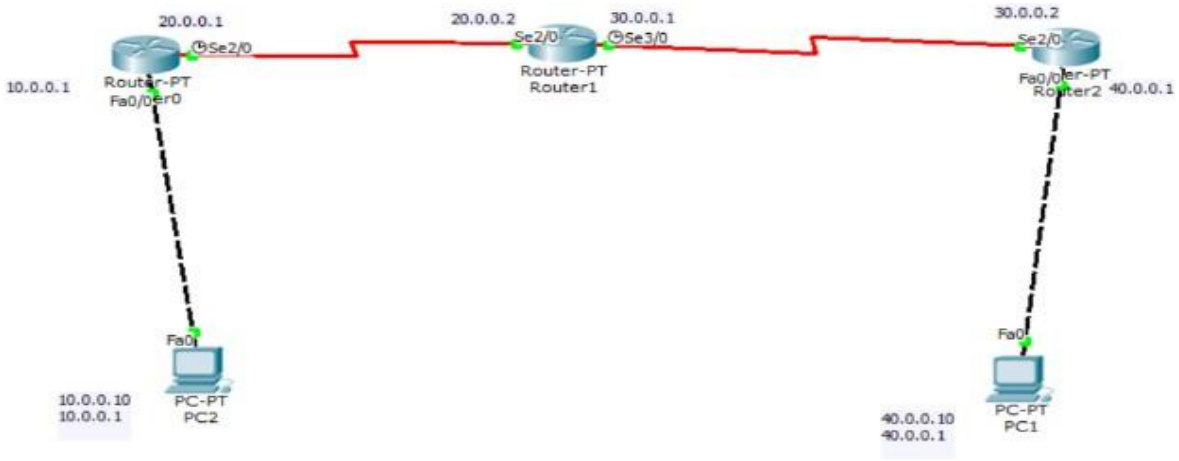
25/10/24

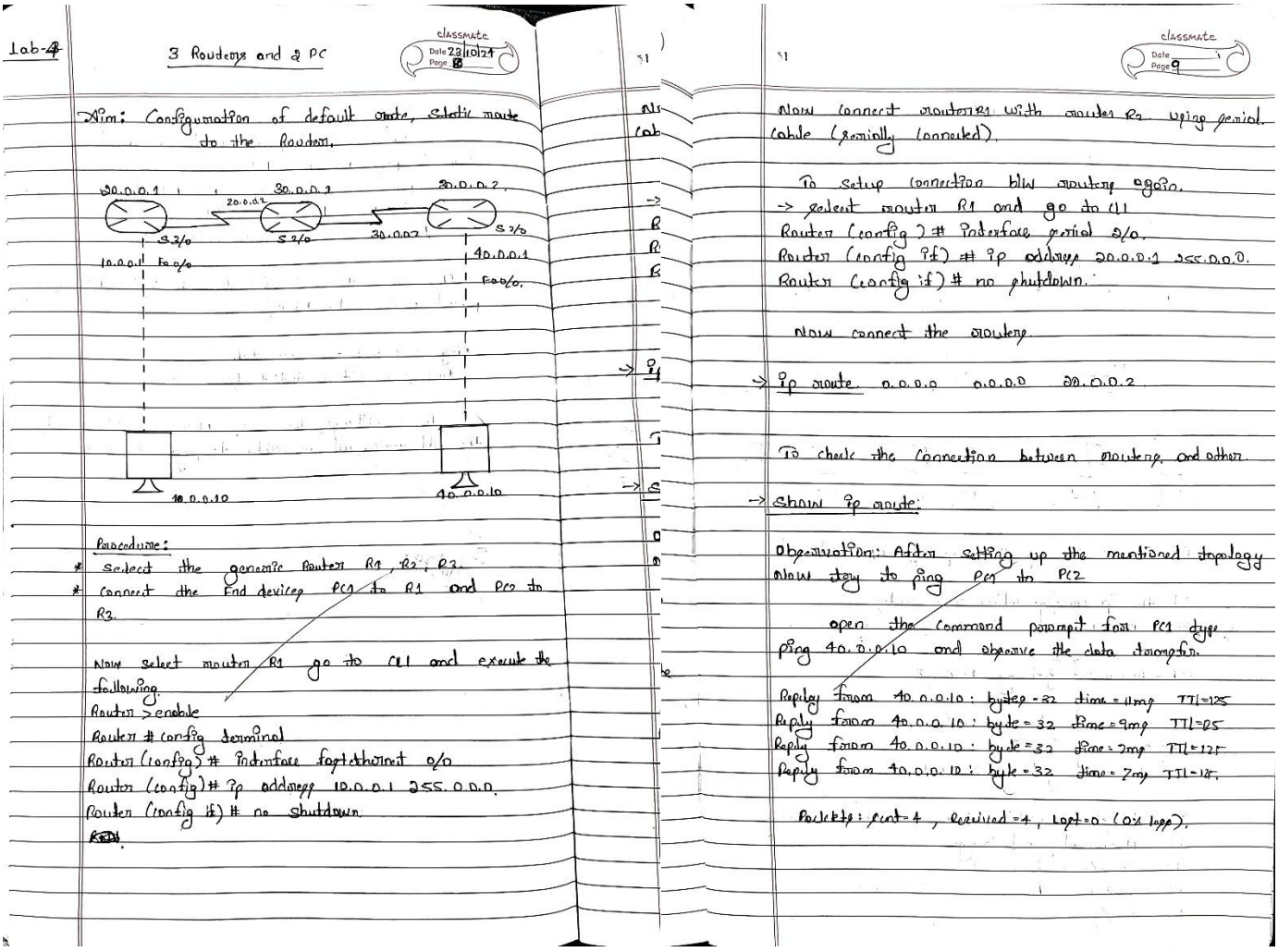

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Pinging 20.0.0.10 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=2ms TTL=127
Ping statistics for 20.0.0.10:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>ping 20.0.0.10
Pinging 20.0.0.10 with 32 bytes of data:
Reply from 20.0.0.10: bytes=32 time=1ms TTL=127
Reply from 20.0.0.10: bytes=32 time=1ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=2ms TTL=127
Ping statistics for 20.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 1ms

Program 3:

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

Topology, Procedure and Observations:




Lab-4

3 Routers and 2 PC

Aim: Configuration of default route, static route to the Routers.

Procedure:

* Select the generic Routers R1, R2, R3.
* Connect the End devices PC1 to R1 and PC2 to R3.

Now select router R1 go to CLI and execute the following.

Router >enable
Router # config terminal
Router (config)# interface fastethernet 0/0
Router (config)# ip address 10.0.0.1 255.0.0.0
Router (config if) # no shutdown

Now connect routers R1 with router R2 using serial cable (serially connected).

To setup connection b/w routers again.

-> select router R1 and go to CLI
Router (config)# interface serial 2/0.
Router (config if) # ip address 20.0.0.1 255.0.0.0.
Router (config if) # no shutdown.

Now connect the routers.

-> ip route 0.0.0.0 0.0.0.0 20.0.0.2

To check the connection between routers and other.

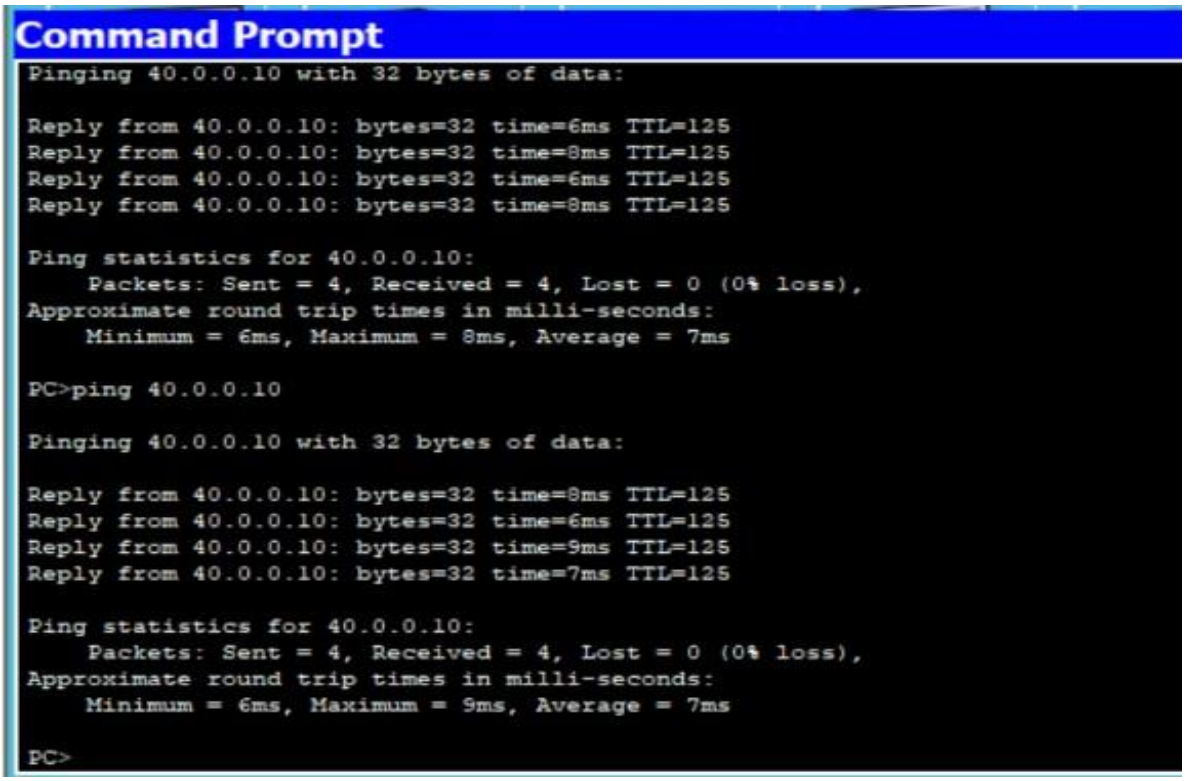
-> show ip route:

Observation: After setting up the mentioned topology now try to ping PC1 to PC2

open the command prompt from PC1 type ping 40.0.0.10 and observe the data transfer.

Reply from 40.0.0.10 : bytes=32 time=11ms TTL=125
Reply from 40.0.0.10 : byte=32 time=9ms TTL=125
Reply from 40.0.0.10 : byte=32 time=7ms TTL=125
Reply from 40.0.0.10 : byte=32 time=7ms TTL=125.

Packets: sent=4, received=4, lost=0 (0% loss).

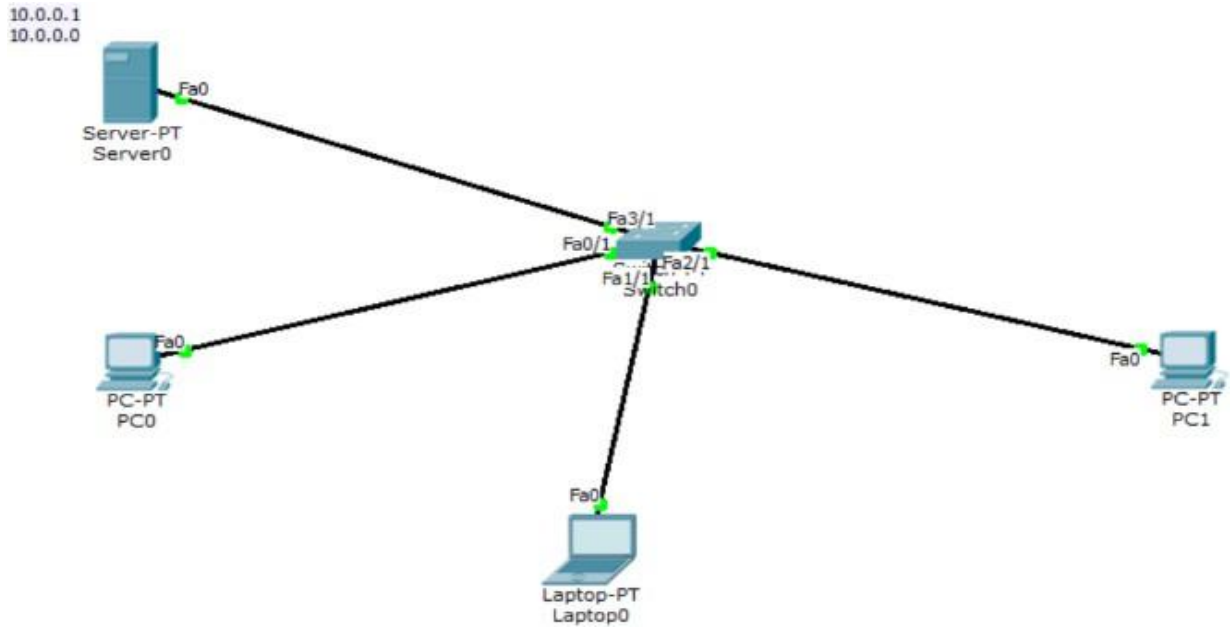

Command Prompt
Pinging 40.0.0.10 with 32 bytes of data:
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Ping statistics for 40.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 8ms, Average = 7ms
PC>ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=9ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Ping statistics for 40.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 9ms, Average = 7ms
PC>

Program 4:

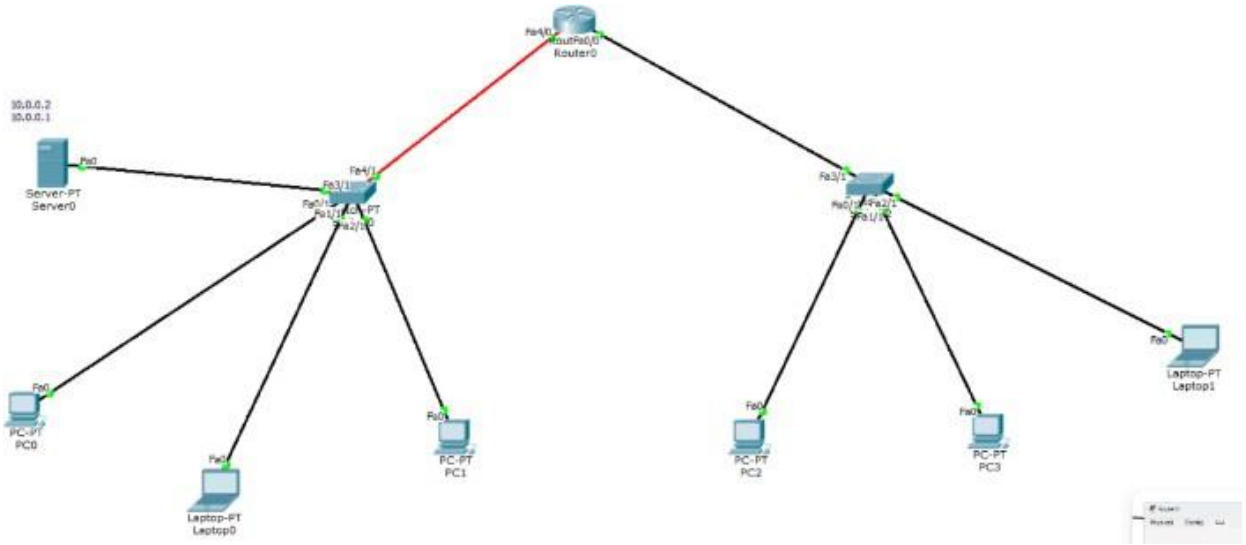
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

Within LAN



Outside LAN



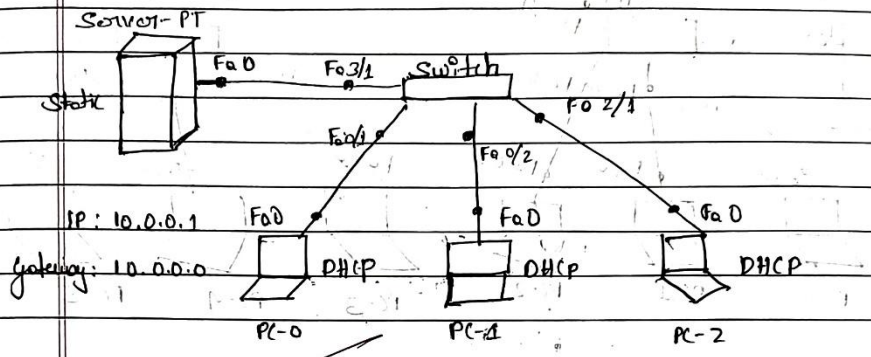
Procedure and Observation:

Lab-5

5) Design a DHCP within LAN and outside LAN Dynamic Host configuration Protocol.

Devices used: One switch, one server, 3 end devices.

Topology: (within LAN)

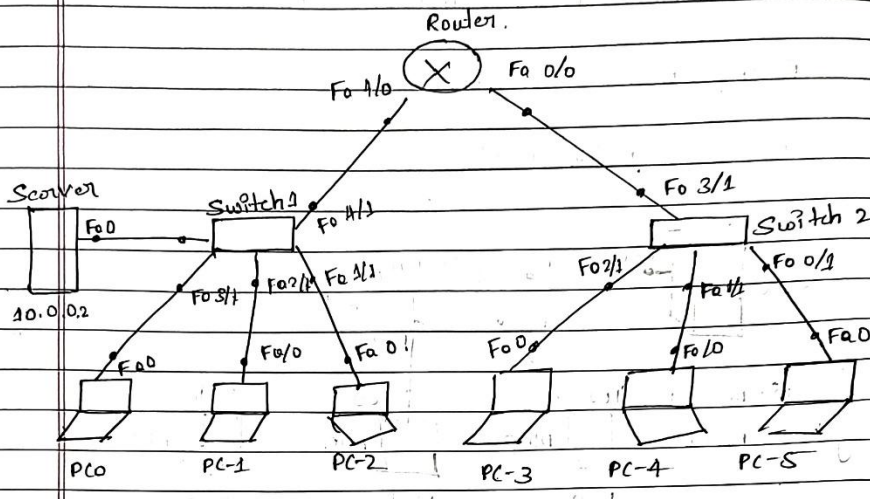


Procedure

Set up the topology as mentioned.
Goto server IP configuration (Desktop)
Select IP addresses 10.0.0.1, 255.0.0.0, 10.0.0.0
Then to set up DHCP, goto config, services select DHCP.
Make DHCP to all three PC's
Dynamic IP addresses will be assigned.

Now ping PC-0 with PC-1 and PC-2
PC-0 had 10.0.0.2 so goto command prompt
-> ping was successful ping 10.0.0.3.

Outside LAN:



Set up the topology as mentioned

Go to server - PT, Desktop, IP configuration change default gateway to 10.0.0.1

Now goto Router CLI

Router > enable
Router # config terminal
Router (config) # interface fastethernet 4/0
Router (config if)# ip address 10.0.0.1 255.0.0.0
Router (config-if)# ip helper-address 10.0.0.2
Router (config-if)# no shut

classmate
Date
Page 15

1.

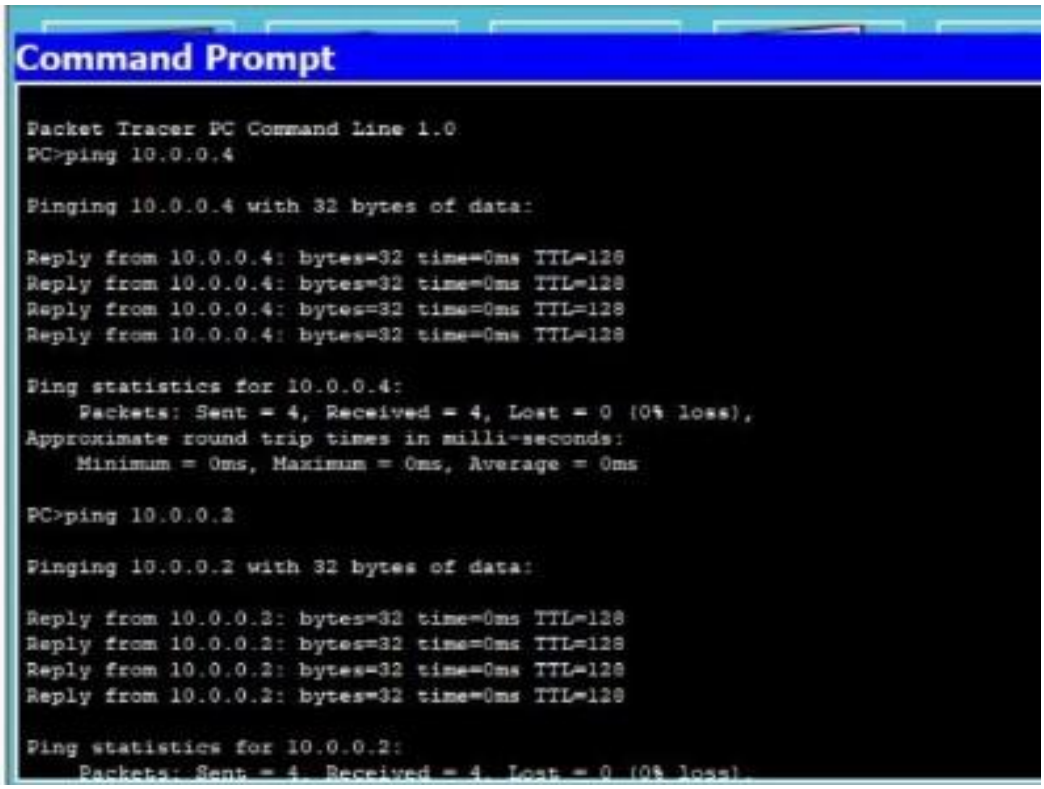
Both networks have established connection with switch & switch 2 with router

We can notice Ip address of

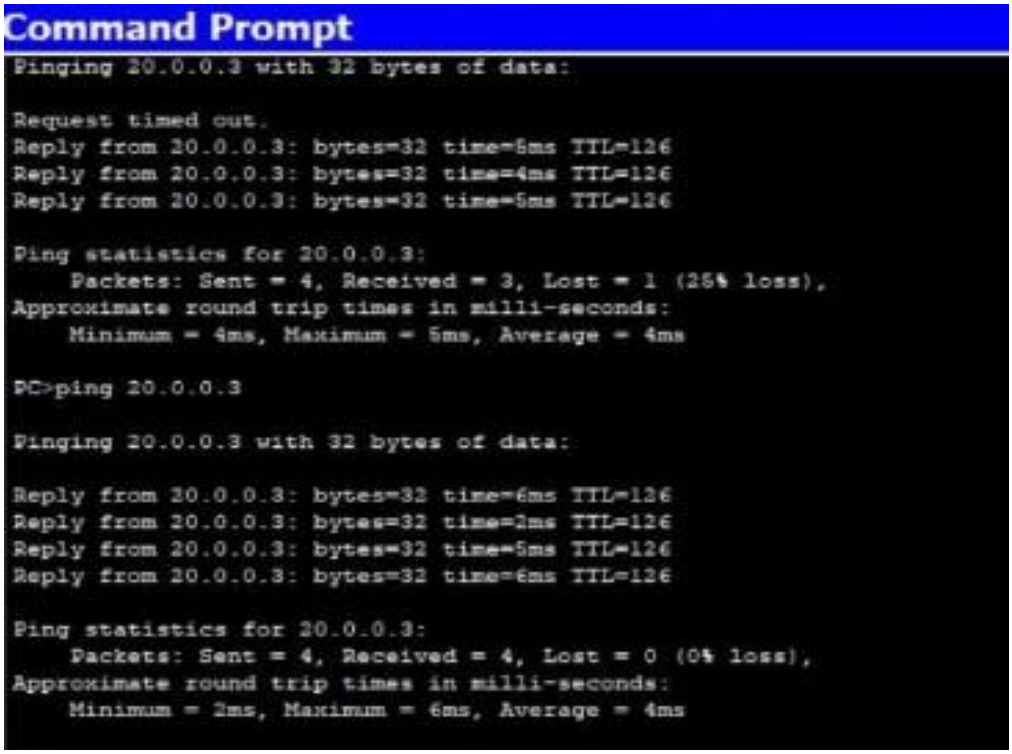
PC0 - 10.0.0.5	PC3 - 20.0.0.4
PC1 - 10.0.0.6	PC4 - 20.0.0.3
PC2 - 10.0.0.3	PC5 - 20.0.0.5

Now, try to ping PC0 with PC5
Goto Desktop of PC0, command prompt

ping was successful.



Within LAN

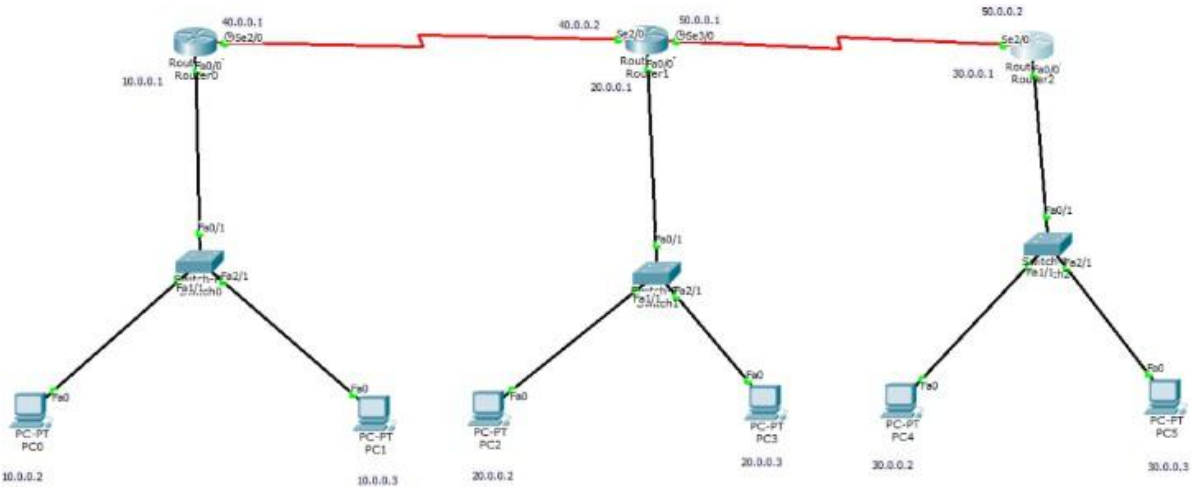


Outside LAN

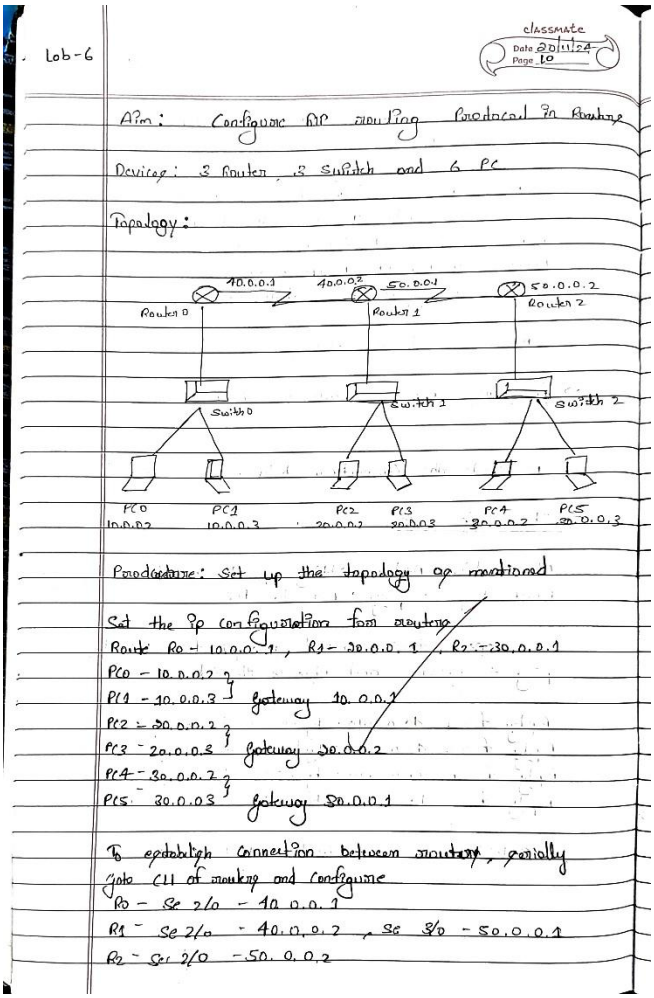
Program 5:

Aim: Configure RIP routing Protocol in Routers.

Topology:



Procedure and Observation:



To establish RIP in routers, goto CLI of Router 0.

Router (config)# router rip
Router (config-router)# network 10.0.0.0
network 40.0.0.0

Router 1
Router (config)# router rip
Router (config-router)# network 20.0.0.0
network 40.0.0.0
network 50.0.0.0

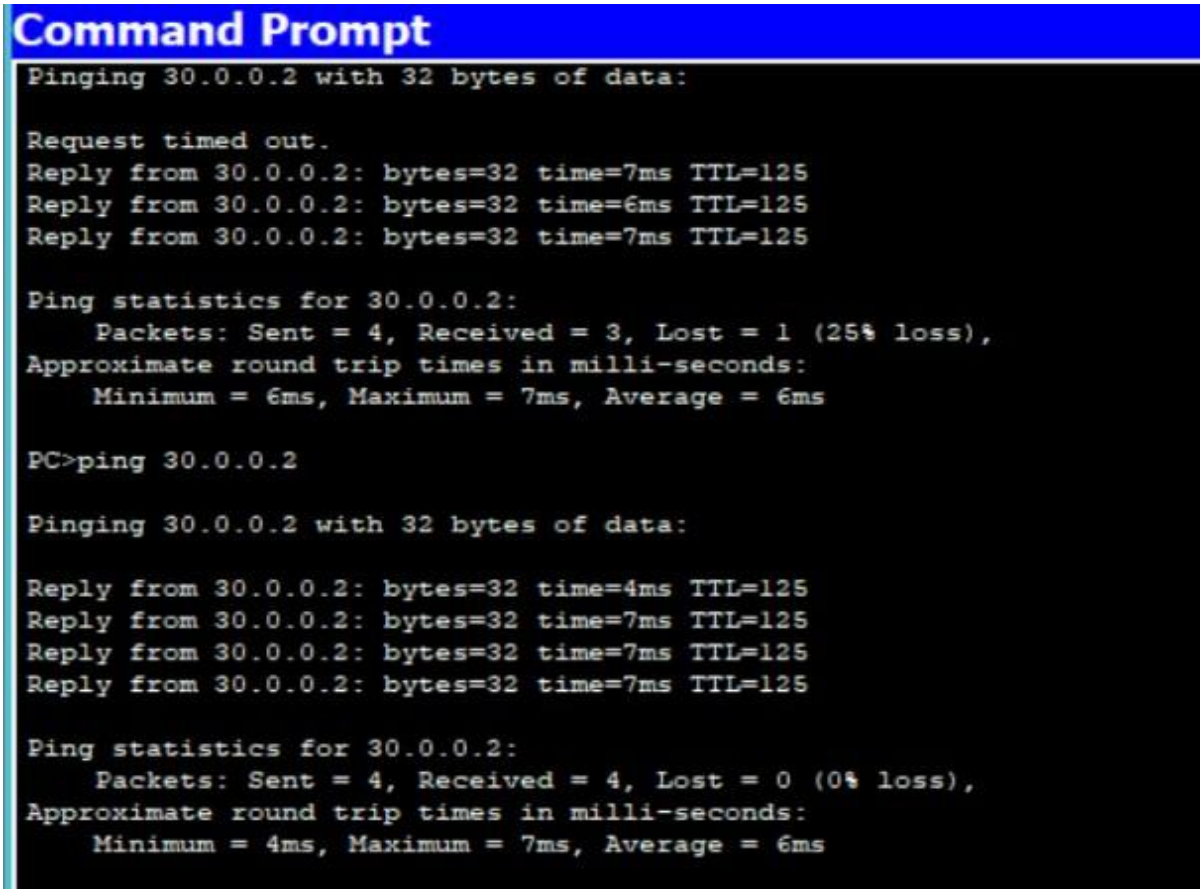
Router 2
Router (config)# router rip
Router (config-router)# network 30.0.0.0
network 50.0.0.0

To overcome the rip connection, check show ip route in router CLI denoting R.

Now in PC0 goto command prompt and

PC > ping 30.0.0.2
pinging 30.0.0.2 with 32 bytes of data:
Reply from 30.0.0.2 : bytes = 32 time=7ms, TTL = 125
bytes = 32 time = 7ms, TTL = 125
bytes = 32 time = 9ms, TTL = 125
Reply from 30.0.0.2 : bytes = 32 time = 7ms, TTL = 125

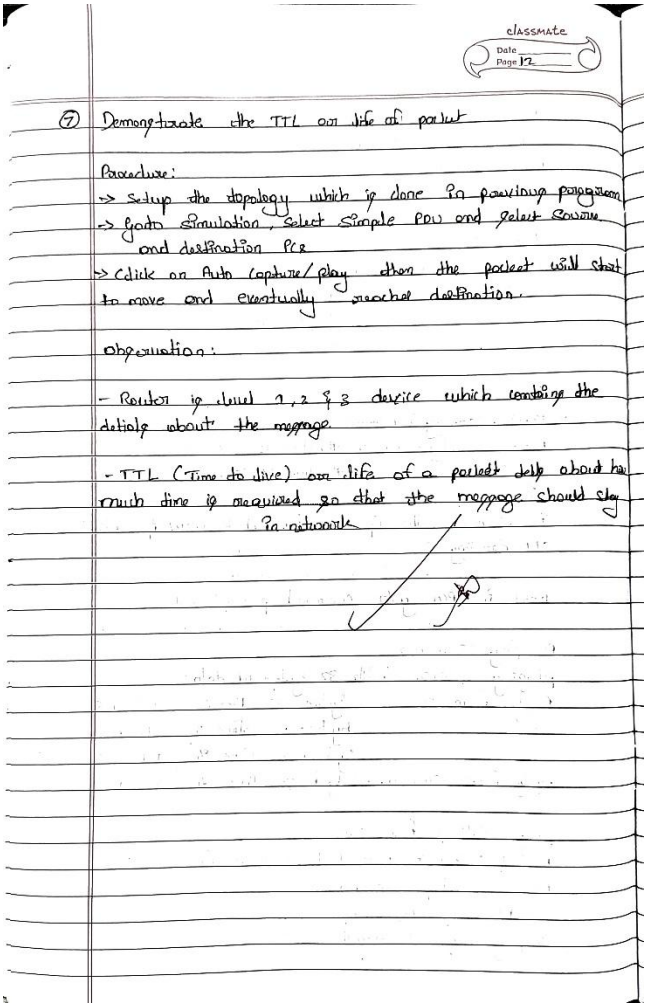
ping statistics for 30.0.0.2
Packets : Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip time in ms.
Min = 7ms, max = 9ms, Avg = 7ms.
∴ ping was successfull.


Command Prompt
Pinging 30.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=6ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 7ms, Average = 6ms
PC>ping 30.0.0.2
Pinging 30.0.0.2 with 32 bytes of data:
Reply from 30.0.0.2: bytes=32 time=4ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Reply from 30.0.0.2: bytes=32 time=7ms TTL=125
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 7ms, Average = 6ms

Program 6:

Aim: Demonstrate the TTL/ Life of a Packet.

Procedure and Observation:



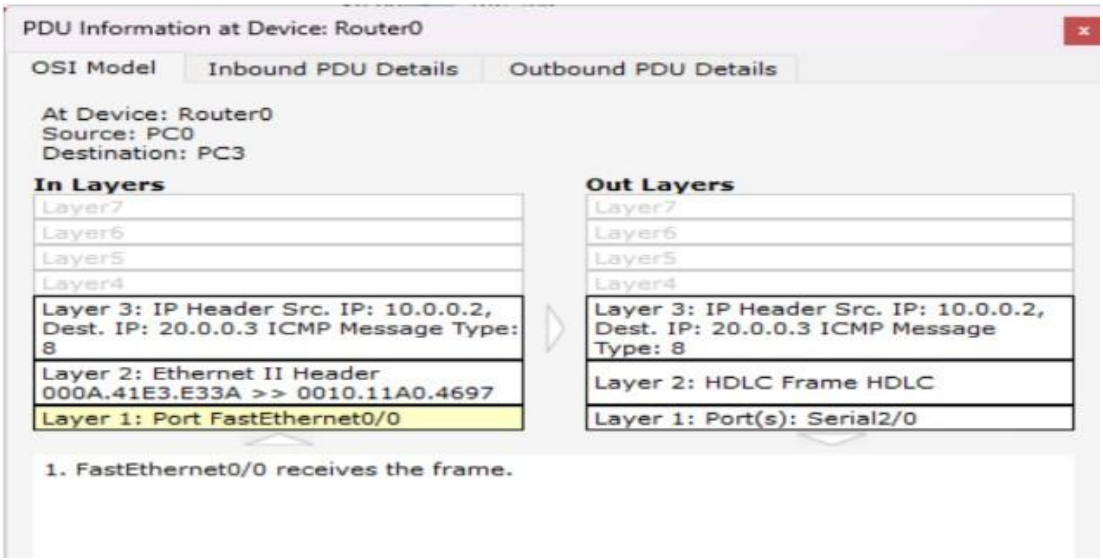
⑦ Demonstrate the TTL on life of packet

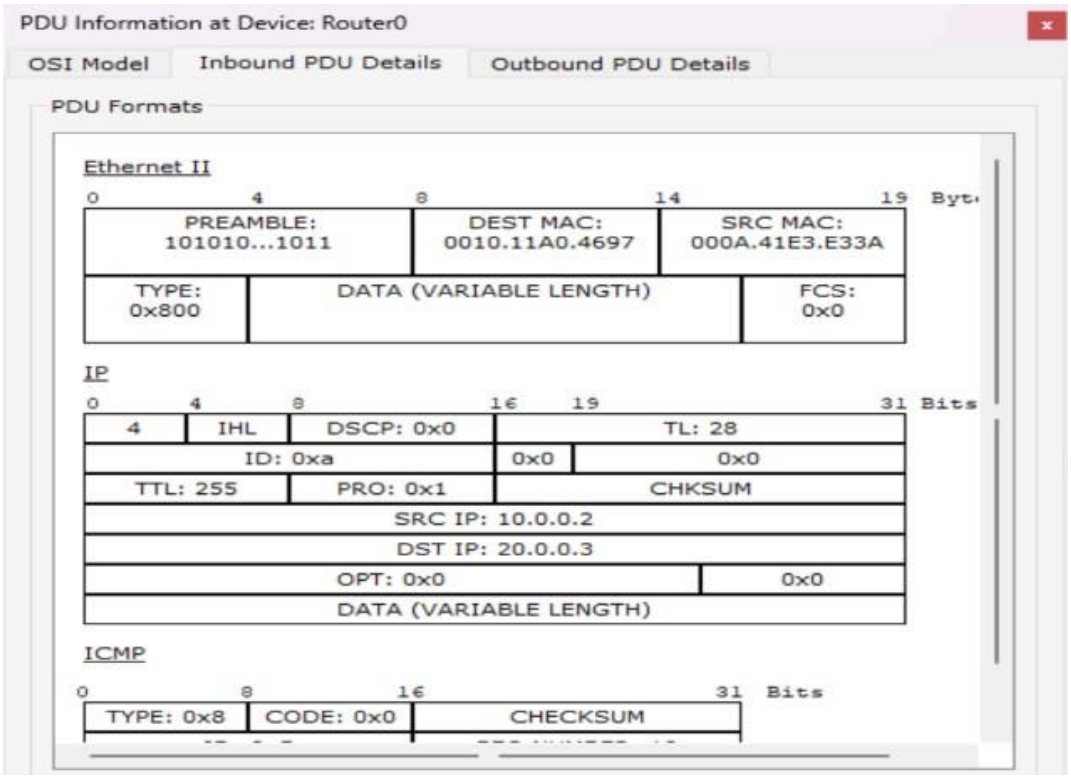
Procedure:

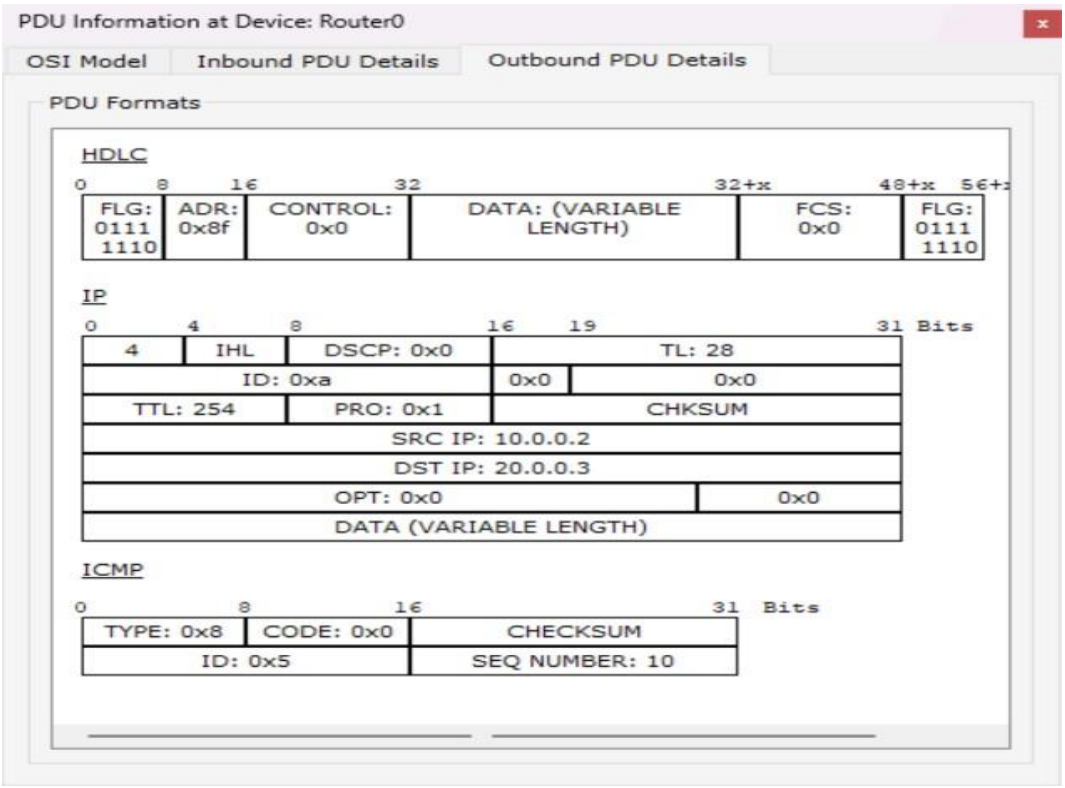
=> Setup the topology which is done in previous program
=> Goto simulation, Select simple PDU and select source and destination PCs
=> Click on Auto capture/play then the packet will start to move and eventually reached destination.

observation:

- Router is level 1, 2 & 3 device which contains the details about the message
- TTL (Time to live) or life of a packet tells about how much time is required so that the message should stay in network



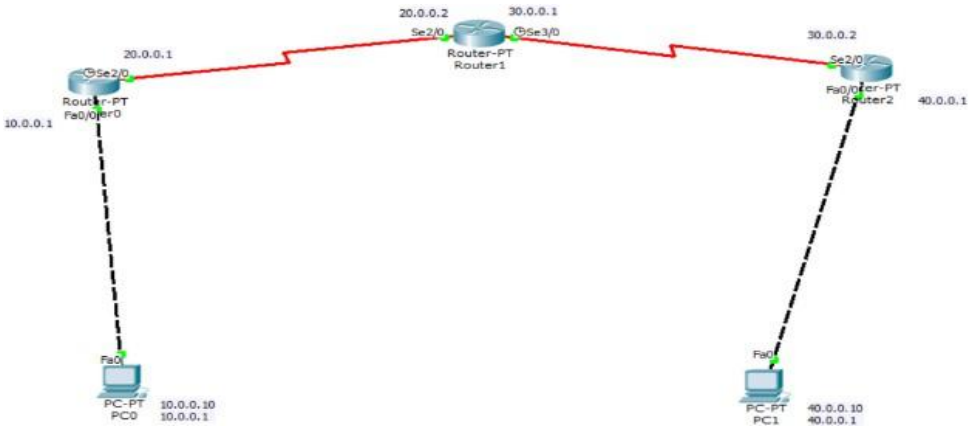

PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Byt
PREAMBLE: 101010...1011
DEST MAC: 0010.11A0.4697
SRC MAC: 000A.41E3.E33A
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
0x0
TTL: 255
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM


PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
HDLC
0 8 16 32 32+x 48+x
FLG: 0111 1110
ADR: 0x8f
CONTROL: 0x0
DATA: (VARIABLE LENGTH)
FCS: 0x0
FLG: 0111 1110
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
0x0
TTL: 254
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 10

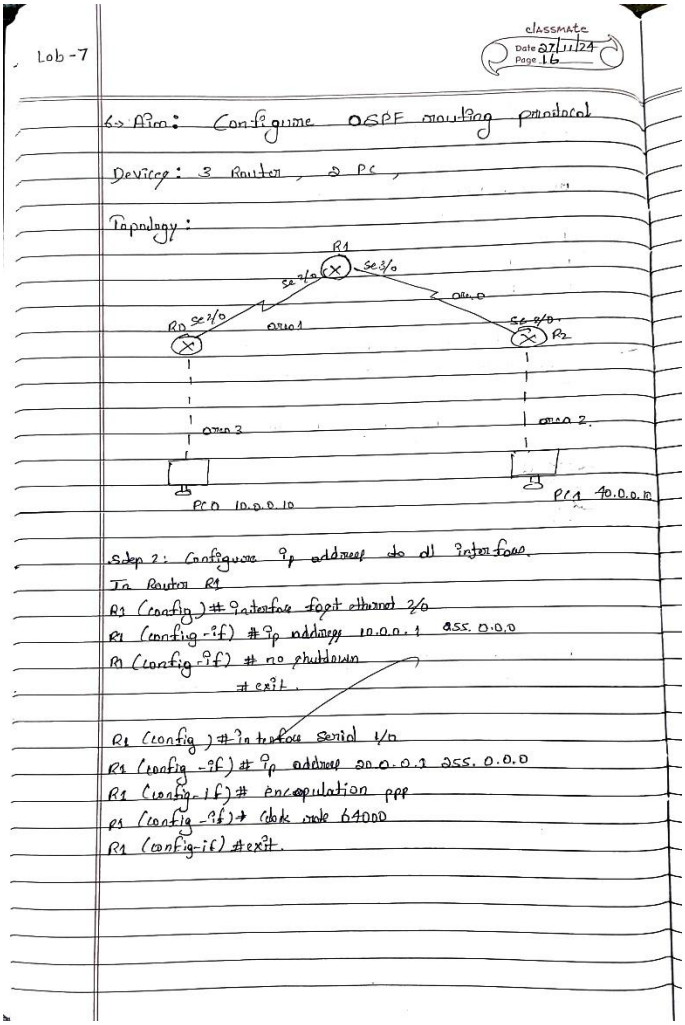
Program 7:

Aim: Configure OSPF routing protocol**.**

Topology:



Procedure and Observation:



In Router R2

R2 (config)# interface serial 1/0
R2 (config-if)# ip address 20.0.0.2 255.0.0.0
encapsulation ppp
no shutdown
exit

R2 (config)# interface serial 1/1
R2 (config-if)# ip address 30.0.0.1 255.0.0.0
encapsulation ppp
clock rate 64000
no shutdown
exit

In Router R3

R3 (config)# interface serial 1/0
R3 (config-if)# ip address 30.0.0.2 255.0.0.0
R3 (config-if)# encapsulation ppp
no shutdown
exit

R3 (config)# interface fastethernet 2/0
R3 (config-if)# ip address 40.0.0.1 255.0.0.0
no shutdown
exit

Step 3: Now, Enable ip routing by configuring OSPF routing protocol in all routers.

In Router R1

R1 (config)# router OSPF 1
R1 (config-router)# router-id 1.1.1.1
R1 (config-router)# network 10.0.0.0 0.255.255.255 area 3
R1 (config-router)# network 20.0.0.0 0.255.255.255 area 1
exit

In Router R2
R2 (config)# router OSPF 1
R2 (config-router)# router-id 2.2.2.2
network 20.0.0.0 0.255.255.255 area 1
network 30.0.0.0 0.255.255.255 area 0.
exit.

In Router R3
R3 (config)# router OSPF 1
R3 (config-router)# router-id 3.3.3.3.
network 30.0.0.0 0.255.255.255 area 0.
network 40.0.0.0 0.255.255.255 area 2

You have to configure router id when we configure ospf. It is used to identify the router.

Step 4: Now check routing table of R1.
Router # show ip route.

C. 10.0.0.0/8 is directly connected FastEthernet 2/0
C. 20.0.0.0/8 is directly connected Serial 1/0
O IA 40.0.0.0/8 [110/129] via 20.0.0.2 00:04:23 Serial 1/0
O IA 30.0.0.0/8 [110/128] via 20.0.0.2 00:07:29. Serial 1/0

Here, R2 knows Area 0. Network 20.0.0.0 connected to R2 from R1 learns network 1/0 through this network.

R3 (config)# router OSPF 1 ; here 1 is process ID

There must be one interface up to keep OSPF protocol up. So it's better to configure loopback address to router. If it is a virtual interface never goes down once we config.

R1 (config)# interface loopback 0
R1 (config-if)# ip address 172.16.1252 255.255.0.0
no shutdown

R2 (config)# interface loopback 0
R2 (config-if)# ip address 172.16.1.253 255.255.0.0
no shutdown

R3 (config)# interface loopback 0
R3 (config-if)# ip address 172.16.1.254 255.255.0.0
no shutdown

Step 5: Now check Routing table of R3.

R3 # show ip route
O IA 20.0.0.0/8 [110/128] via 30.0.0.1, 00:15:38 ser 1/0
C 40.0.0.0/8 is directly connected, FastEthernet 2/0
C 30.0.0.0/8 is directly connected, Serial 1/0.

Here, R3 doesn't know about the area 3, so we have to create virtual link b/w R1 and R2.

Step 6: Create virual link b/w R1, R2 by this we create a virtual link to connect area 3 to area 0.

In Router R1,
R1 (config)# router OSPF 1
R1 (config)# area 1 virtual link 2.2.2.2

In Router R2
R2 (config)# router OSPF 1
R2 (config-router)# area 1 virtual link 1.1.1.1
exit

classmate
Date
Page 20

Step 7: R2 and R3 get updated about Area 2, Now, check routing table of R3.

R3 # show ip route

O IA 20.0.0.0/8 [110/72] via 30.0.0.1, 00:01:51 Serial 1/0
C 40.0.0.0/8 is directly connected, FastEthernet 2/0
C IA 10.0.0.0/8 [110/129] via 30.0.0.1, 00:01:51 Serial 1/0
C R0.0.0.0/8 is directly connected, Serial 1/0.

Step 8: Check connectivity b/w host 10.0.0.10 to 40.0.0.10.

#ping 40.0.0.10

Now, if we get the reply without drops then the connection is established

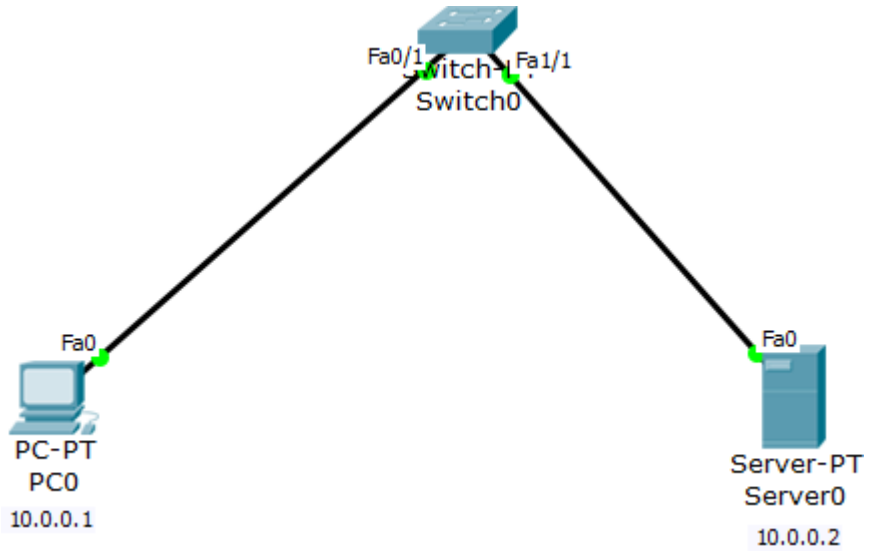
30/12/24

```
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125

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

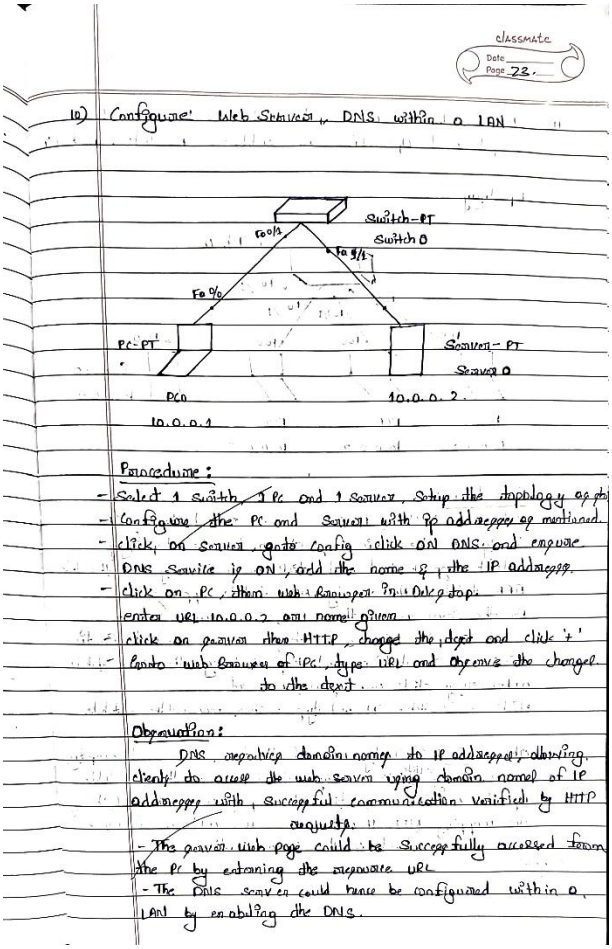
Program 8:

Aim: Configure Web Server, DNS within a LAN.

Topology:



Procedure and Observations:

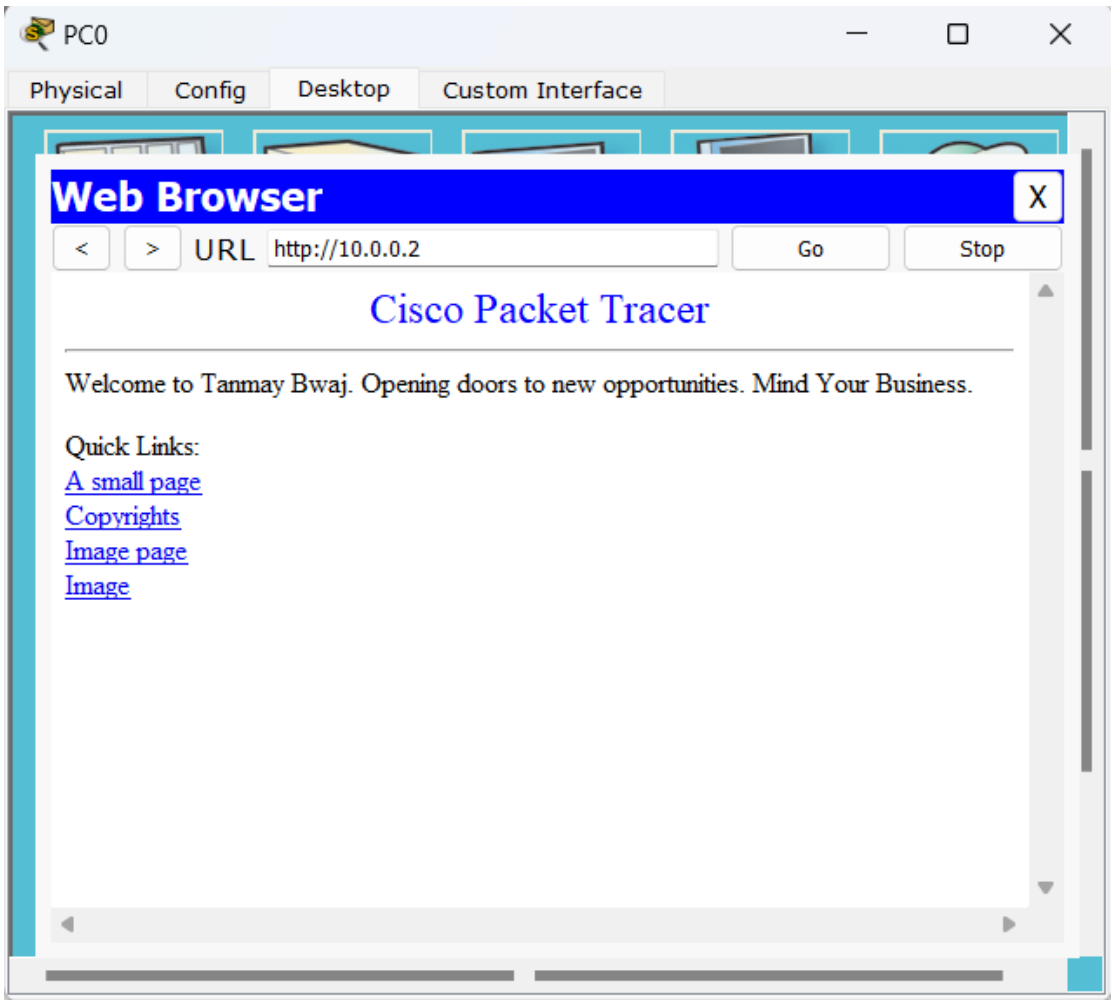


10) Configure Web Server, DNS within a LAN

Switch-PT
Switch 0
Fa0/1
Fa 0/2
Fa 0/3
Fa 0/4
PC-PT
PC0
10.0.0.1
Server-PT
Server 0
10.0.0.2

Procedure:
- Select 1 switch, 1 PC and 1 server, setup the topology as given
- Configure the PC and Server with IP addresses as mentioned.
- Click on server, goto config, click on DNS and ensure DNS service is ON, add the name & the IP addresses.
- Click on PC, then web browser in Desktop.
enter URL 10.0.0.2 and name given.
- Click on server then HTTP, change the index and click '+'
- Goto web browser of PC, type URL and observe the changes to the index.

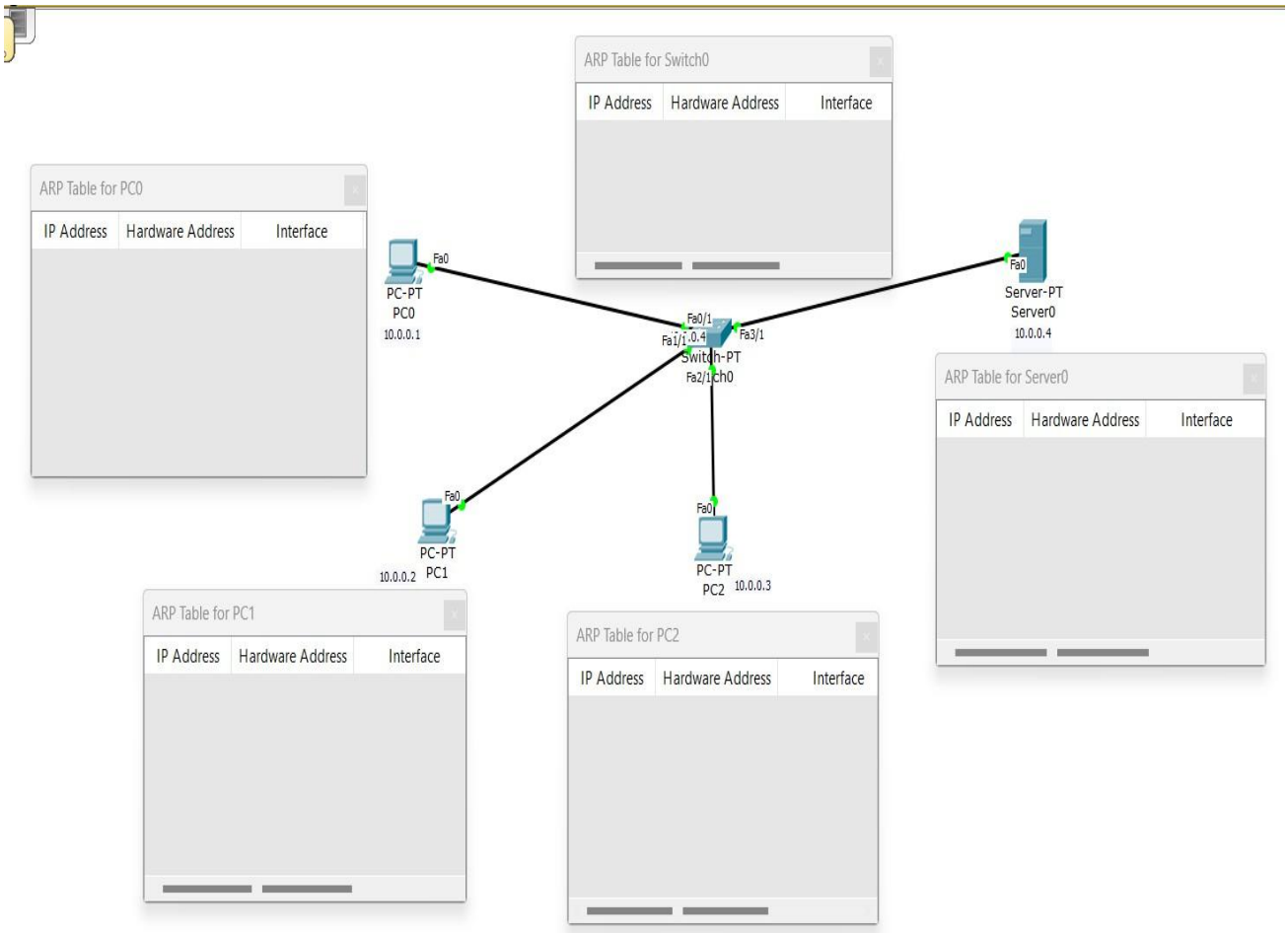
Observation:
DNS resolving domain names to IP addresses, allowing clients to access the web servers using domain names of IP addresses with successful communication verified by HTTP requests.
- The server web page could be successfully accessed from the PC by entering the respective URL
- The DNS service could hence be configured within a LAN by enabling the DNS.


PC0
Physical
Config
Desktop
Custom Interface
Web Browser
X
<
>
URL
http://10.0.0.2
Go
Stop
Cisco Packet Tracer
Welcome to Tanmay Bwaj. Opening doors to new opportunities. Mind Your Business.
Quick Links:
A small page
Copyrights
Image page
Image

Program 9:

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

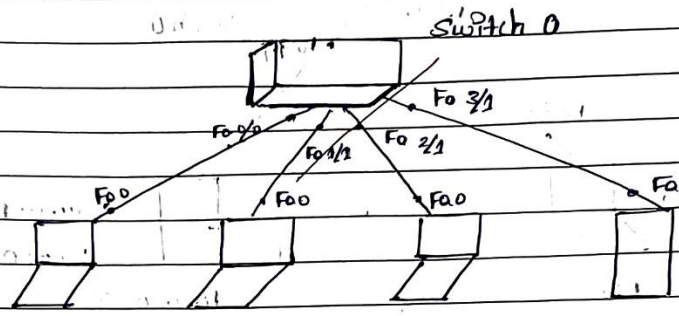
Topology:



Procedure and Observations:

1) To Construct a Simple LAN and Understand the concept and Operation of address Resolution Protocol (ARP)

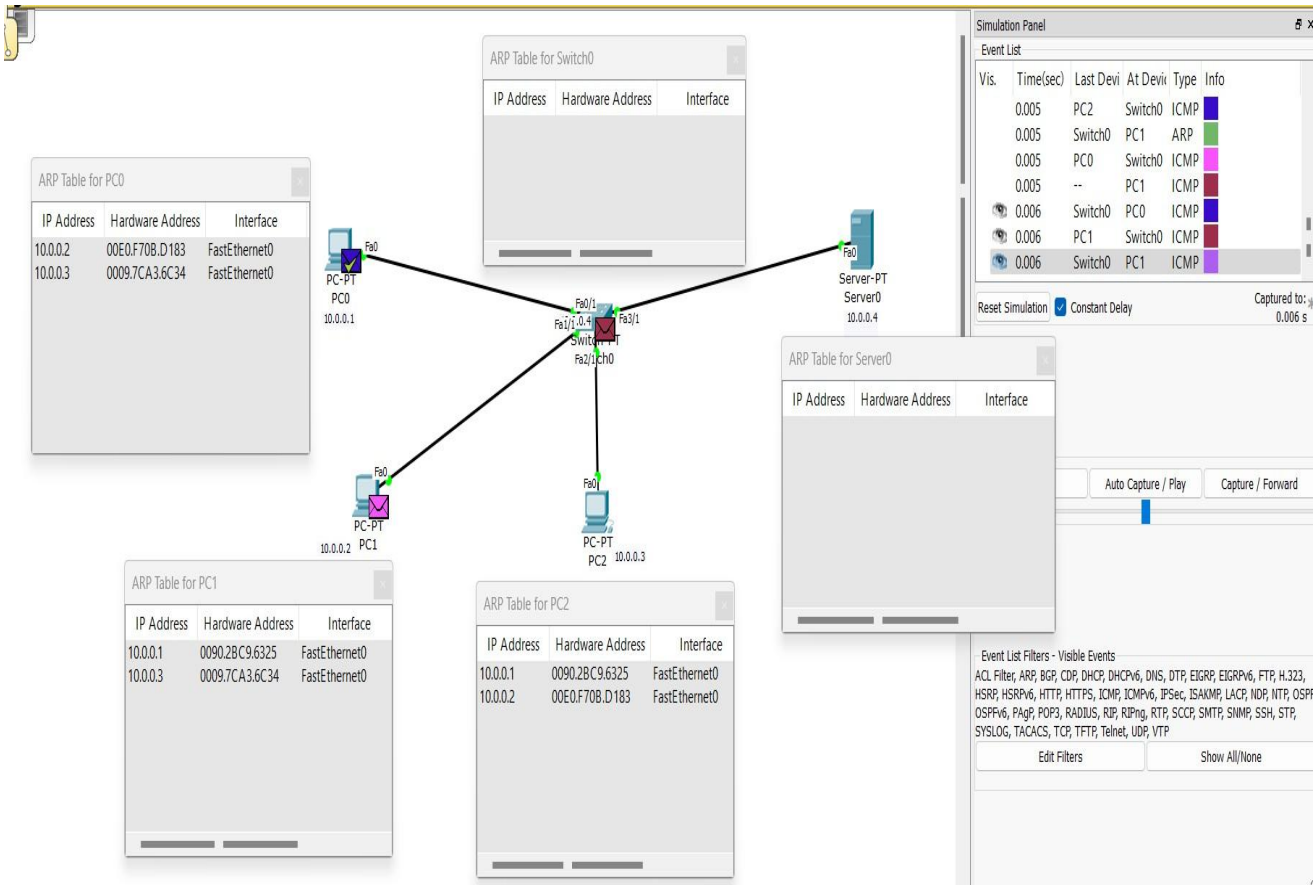
Topology



Procedure:
Select 1 Switch, 1 Server and 3 end devices / PCs
Configure the devices as shown in the topology
Select Inspect tool and click on a PC say PC0 then click on ARP table. An empty ARP table appears.
Do the same for other PCs, Server and Switch.
Select Simple PDU and choose source and destination, click on Simulation and keep checking the ARP table after every click on Capture / forward.
click on Switch - CLI and type : show mac address - table

Observation: Initially ARP of all devices observed empty. After every click on Capture/forward, in the ARP table update of ARP requests and replies are exchanged showing the mapping b/w IP addresses & MAC addresses of device involved in the communication.

30/12/24




```
Switch>show mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    0009.7ca3.6c34    DYNAMIC     Fa2/1
   1    0090.2bc9.6325    DYNAMIC     Fa0/1
   1    00e0.f70b.d183    DYNAMIC     Fa1/1
Switch>
```

Program 10:

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

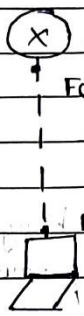
Topology :

Procedure and Observations:

classmate
Date
Page 25

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

Topology:



Procedure:

Select 1 router and 1 PC. Configure/setup the topology as shown goto CLI of router and type the following commands.

enable
config terminal
hostname CSE
enable secret password
interface fastethernet 0/0.
ip address 10.0.0.2 255.0.0.0
no shut
line vty 0 3.
login
password psw
exit
exit.

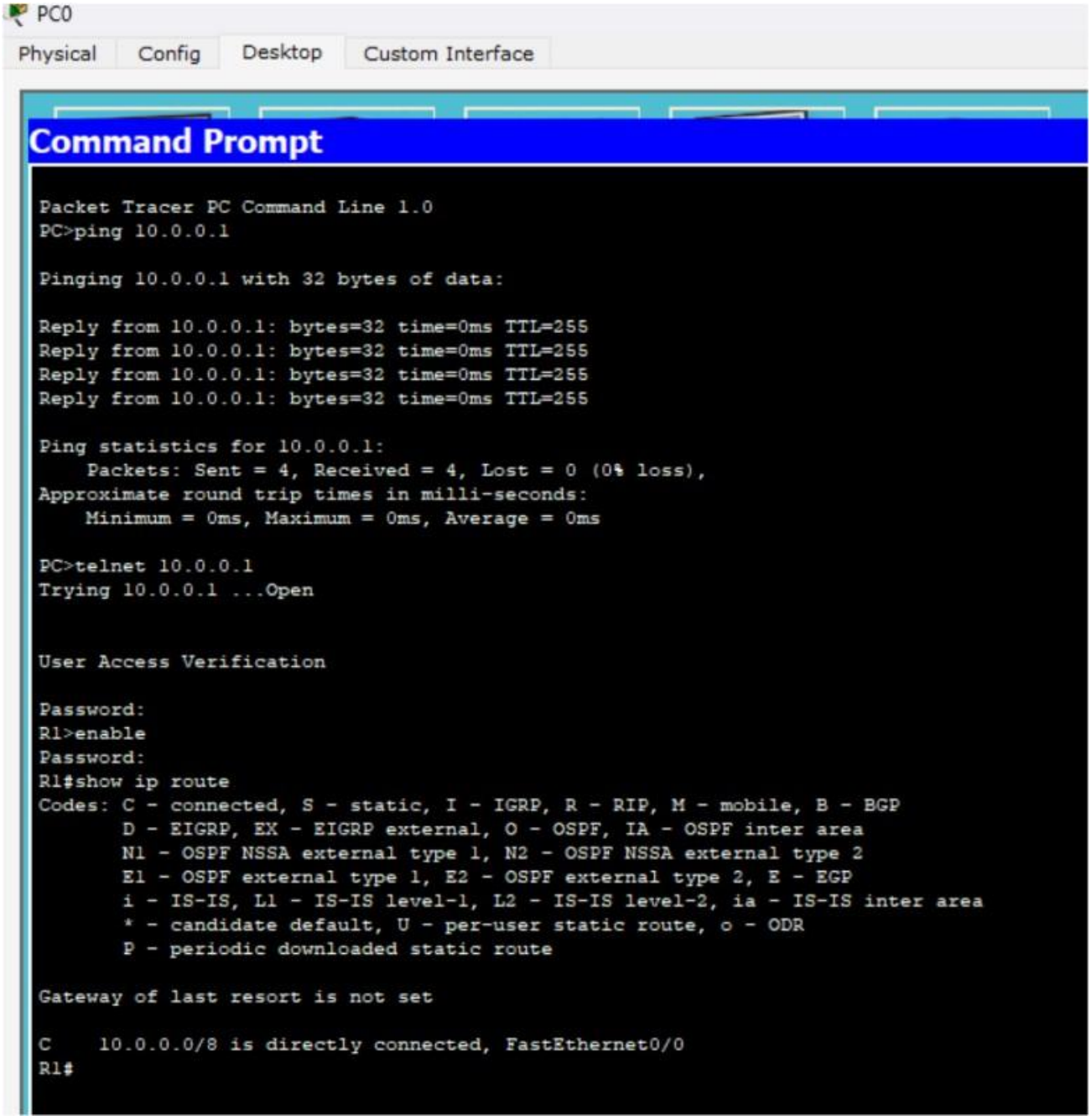
Now goto command prompt PC0, enquire ping 10.0.0.2 is successful.

Adgb in the cmd of PC, type telnet 10.0.0.2 give password which was set and observe the changes

Observation

telnet 10.0.0.2
trying 10.0.0.2 ... open
use Access verification
Password psw
Here the name PC changes to CSE which was the hostname execute the commands.
enable
password psw
Show ip route
10.0.0.0/8 is directly connected. fastethernet 0/0.

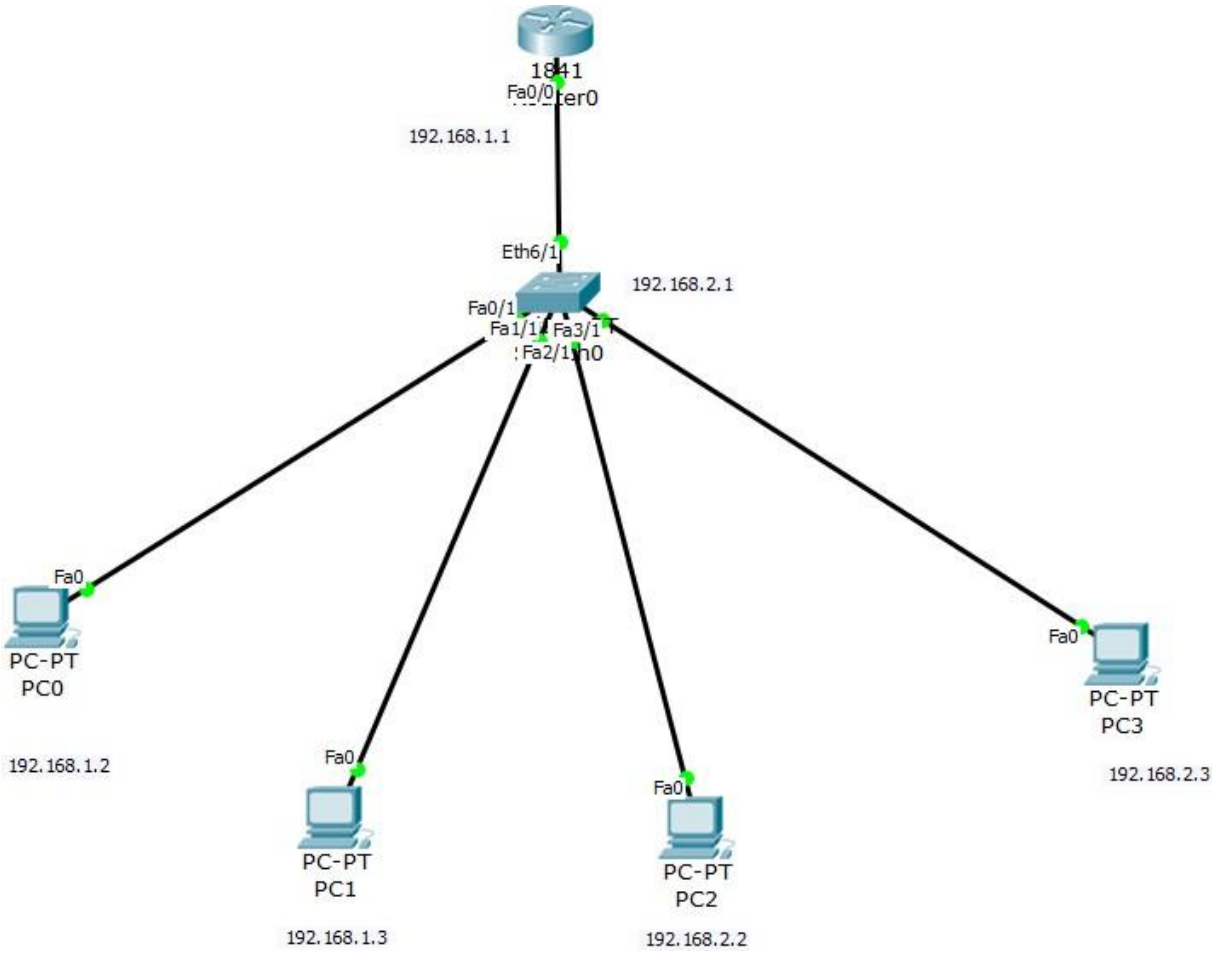
30/12/24


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
Gateway of last resort is not set
C 10.0.0.0/8 is directly connected, FastEthernet0/0
R1#

Program 11:

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

Topology:



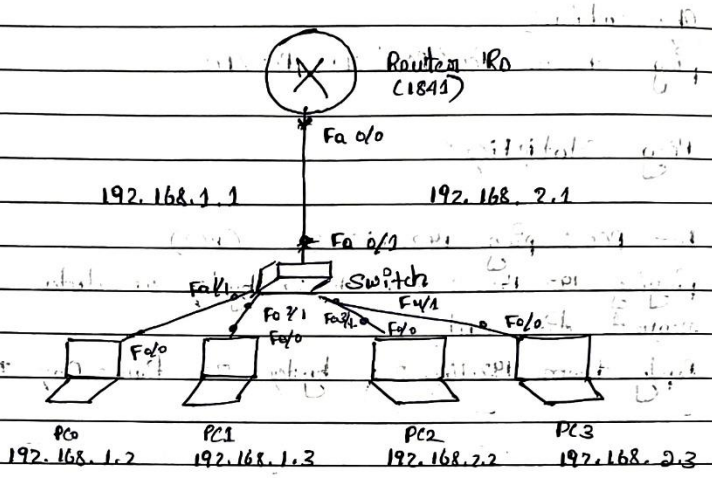
Procedure and Observations:

Lab - 8

9) Construct a VLAN and make the PC's communicate among a VLAN

Devices used : 1 router, 1 switch, 4 PC's

Topology :



Procedure : Set up the topology as seen above.
Use specific router 1841

→ Assign IP addresses to the PCs as shown in topology

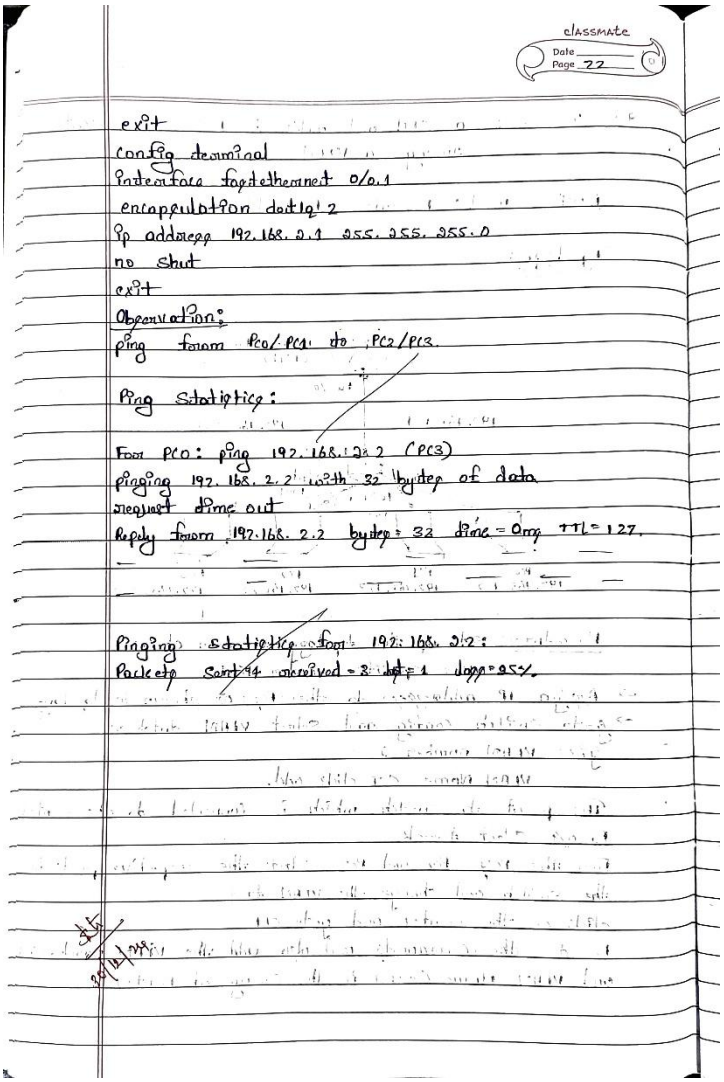
→ Go to switch, config and select VLAN database
Give VLAN number 2
VLAN Name : CSE click add.

The port to switch which is connected to the router Fa 0/1 select trunk

For the PC's, PC2 and PC3. Select the respective ports in the switch and change the VLAN to 2

click on the router and goto CLI

Execute the commands and also add the VLAN number (2) and VLAN name (CSE) in the config of Router.

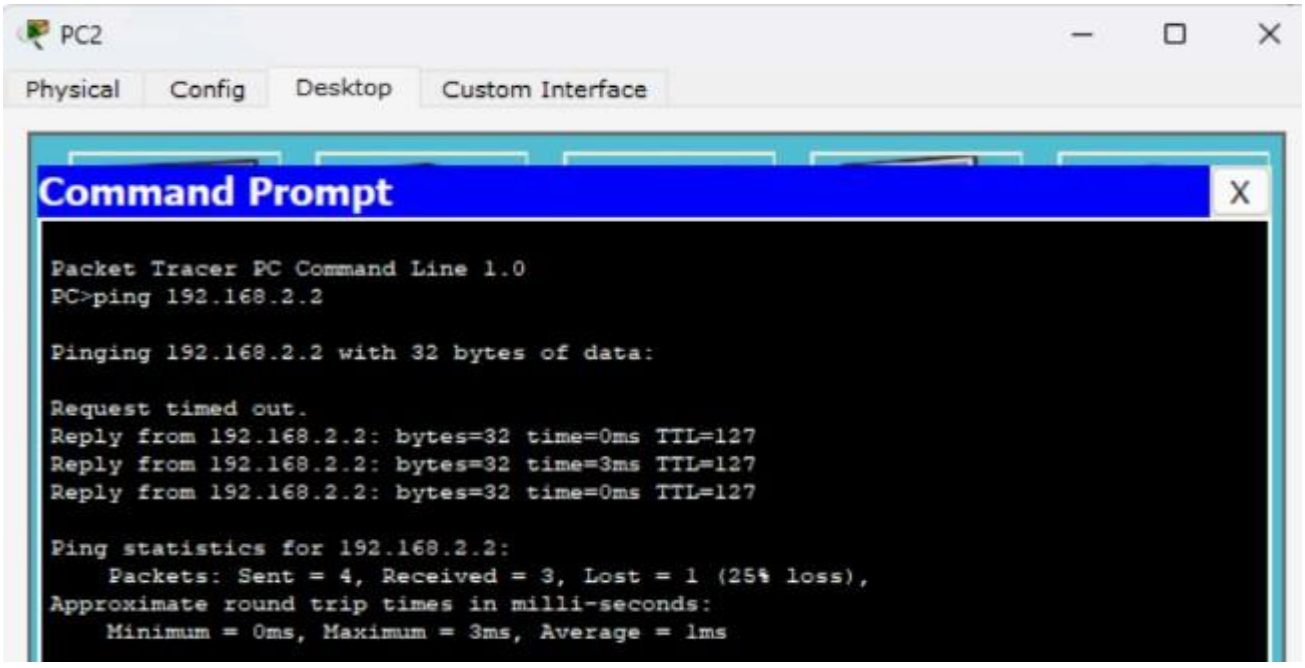


exit
config terminal
interface fastethernet 0/0.1
encapsulation dot1q 2
ip address 192.168.2.1 255.255.255.0
no shut
exit
Observation:
ping from PC0/PC1 to PC2/PC3.

Ping Statistics:

For PC0: ping 192.168.2.2 (PC3)
pinging 192.168.2.2 with 32 bytes of data
request time out
Reply from 192.168.2.2 bytes=32 time=0ms TTL=127

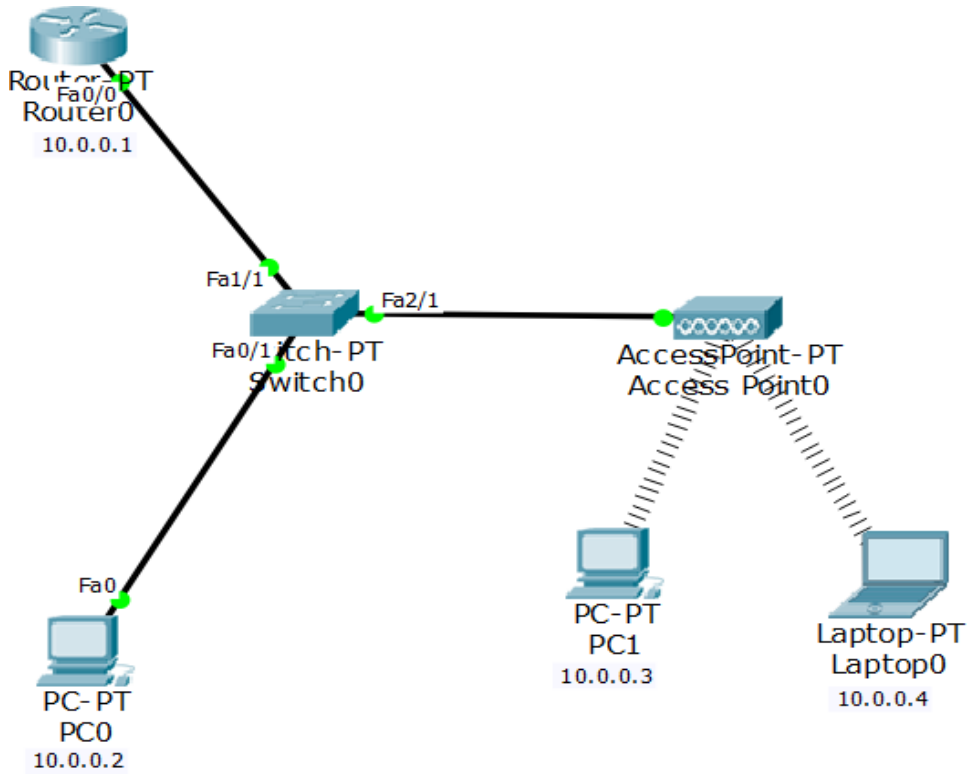
Pinging statistics for 192.168.2.2:
Packets: Sent=4 received=3 lost=1 loss=25%



Program 12:

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

Topology:

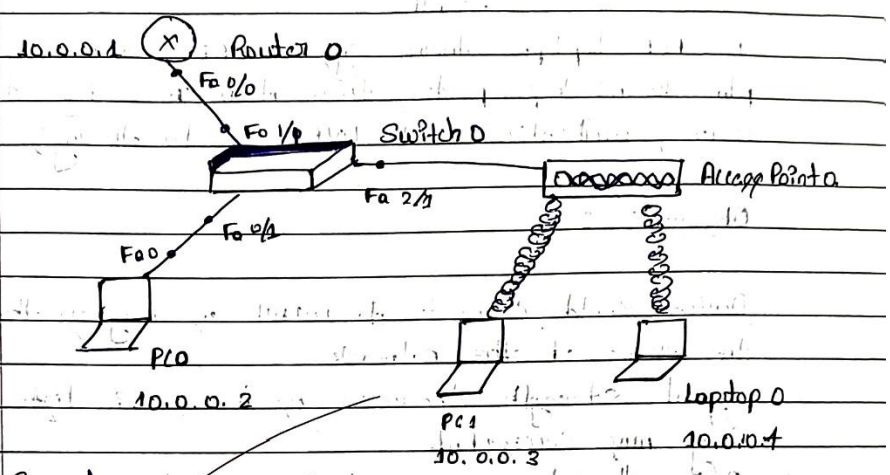


Procedure and Observations:

classmate
Date
Page 27.

13) To construct a WLAN and make the nodes communication wirelessly.

Topology:



Procedure:

Select 1 router, 2 PCs, 1 laptop, 0 switch and an access point.
Set up the topology as shown (Initially no connection b/w end point & Access point)
Configure the devices with their IP addressing as shown.
To configure access point, goto port 1.
give a name to SSID as CSE
Select WEP and give 10 digit WEP key as 1234567890
ensure the port status is ON
Configure PC1 and laptop with wireless standards.
For PC0, switch off, drag existing PT-HOST-NM-1AM to unported slotted in LHS.
Drag WMP300N wireless interface to the empty port
Switch on PC of Access Point 0.

classmate
Date
Page 28

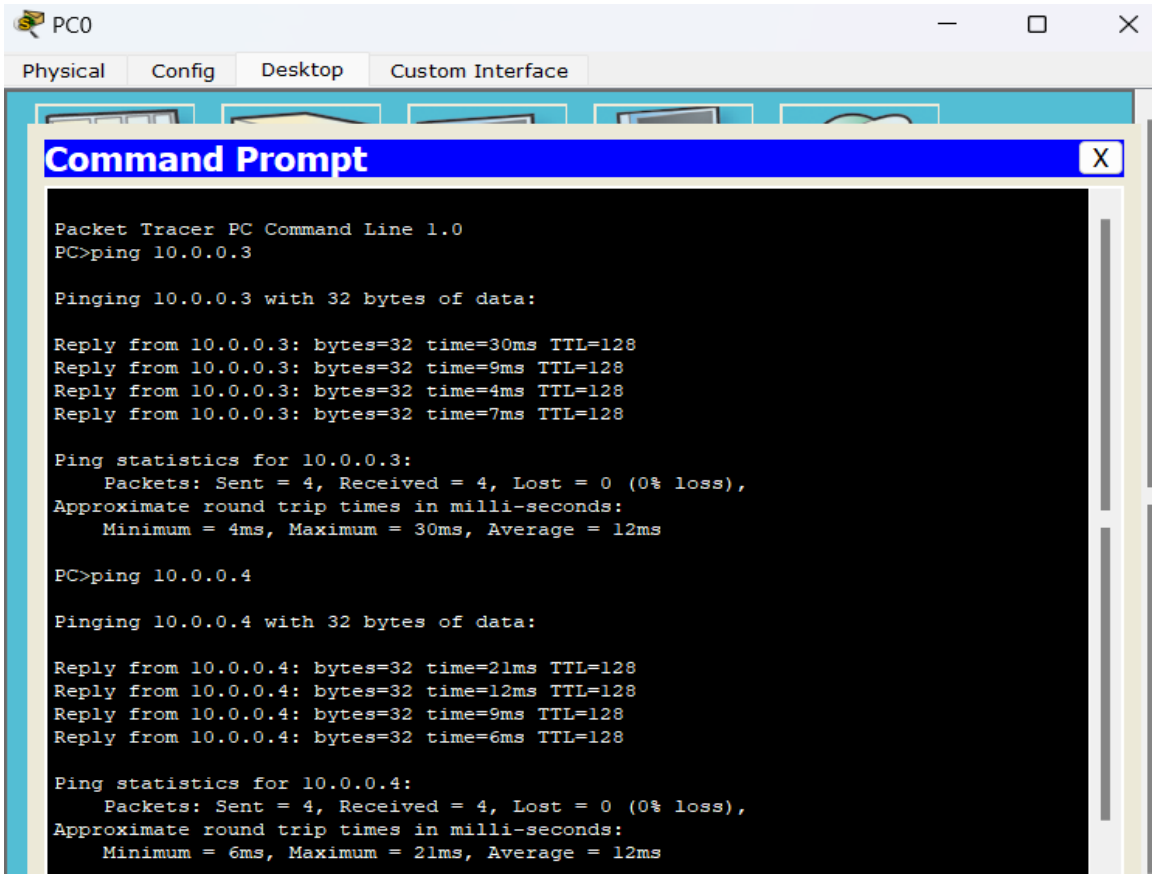
In the Config tab a new wireless interface would have been added configure SSID, WEP, WEP Key, IP address & gateway for the device [Port status should be set 'ON' Set SSID name & Do similar Process from Laptop ping from every device to all other device and observe the result.

In PCs, laptops, turn systems 'OFF', remove the ports, place the wireless port & turn it on. In Config, set same SSID & authentication to WEP & enter the key).

Observation:

- Device could connect to WLAN as long as they are in the range of the network.
- Signal strength decrease as increase in distance
- Ping was successful.
- After the setup, PCs & laptops wireless connection are observed with Access point to indicating successfull wireless connection.

Ans
30/1/24


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=30ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=4ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms 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 = 4ms, Maximum = 30ms, Average = 12ms
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=21ms TTL=128
Reply from 10.0.0.4: bytes=32 time=12ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms 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 = 6ms, Maximum = 21ms, Average = 12ms

CYCLE - 2

Program 13:

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

```
#include <iostream>
#include <string.h>
using namespace std;

int crc(char *ip, char *op, char *poly, int mode)
{
strcpy(op, ip);
if (mode) {
   for (int i = 1; i < strlen(poly); i++)
       strcat(op, "0");
}
/* Perform XOR on the msg with the selected polynomial */
for (int i = 0; i < strlen(ip); i++) {
     if (op[i] == '1') {
          for (int j = 0; j < strlen(poly); j++) {
               if (op[i + j] == poly[j])
                    op[i + j] = '0';
    else
          op[i + j] = '1';
}}}
/* check for errors. return 0 if error detected */
for (int i = 0; i < strlen(op); i++)
       if (op[i] == '1') return 0;
return 1;
}

int main(){
     char ip[50], op[50], recv[50];
     /* x 16 + x12 + x5 + 1 */
      char poly[] = "10001000000100001";
      cout << "Enter the input message in binary"<< endl;
      cin >> ip;
      crc(ip, op, poly, 1);
      cout << "The transmitted message is: " << ip << op + strlen(ip) << endl;
      cout << "Enter the received message in binary" << endl;
      cin >> recv;
      if (crc(recv, op, poly, 0))
           cout << "No error in data" << endl;
     else
           cout << "Error in data transmission has occurred" << endl;
     return 0;
}
```

Observations:

Part B

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

Solution:

```
#include <iostream>
#include <string.h>
using namespace std;

int crc(char *ip, char *op, char *poly, int mode)
{
  strcpy(op, ip);
  if(mode){
    for(int i=1; i<strlen(poly); i++)
      strcat(op, "0");
  }
  for(int i=0; i<strlen(ip); i++){
    if(op[i] == '1'){
      for(int j=0; j<strlen(poly); j++){
        if(op[i+j] == poly[j])
          op[i+j] = '0';
        else
          op[i+j] = '1';
      }
    }
  }
  for(int i=0; i<strlen(op); i++)
    if(op[i] == '1')
      return 0;
  return 1;
}
```

```
int main()
{
  char ip[50], op[50], recv[50];
  char poly[] = "10001000000100001";

  cout << "Enter the input message in binary" << endl;
  cin >> ip;
  crc(ip, op, poly, 1);
  cout << "The transmitted message is: " << ip << op << endl;
  cout << "Enter the received message in binary" << endl;
  cin >> recv;
  if (!crc(recv, op, poly, 0))
    cout << "No error in data" << endl;
  else
    cout << "Error in data transmission has occurred" << endl;
  return 0;
}
```

Output 1

Enter the input message in binary 1111101
The transmitted message is: 1111101011010111001110110
Enter the received message in binary 1111101
No error in data.

Output 2.

Enter the input message in binary: 1111101
The transmitted message is: 1111101011010111001110110
Enter the received message in binary 1110
Error in data transmission has occurred.

30/12/24

Program 14:

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

Algorithm:
1. Start
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted.
   (Reject packets whosesize is greater than the bucket size.)
6. Stop

Code:
```
#include <iostream>
#include <string.h>
using namespace std;

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 10
int rand(int a){
     int rn = (random() % 10) % a;
     return rn == 0 ? 1 : rn;
}
int main() {
   int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
   for(i = 0; i<NOF_PACKETS; ++i)
        packet_sz[i] = rand(6) * 10;
   for(i = 0; i<NOF_PACKETS; ++i)
       printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
   printf("\nEnter the Output rate:");
   scanf("%d", &o_rate);
   printf("Enter the Bucket Size:");
   scanf("%d", &b_size);
   for(i = 0; i<NOF_PACKETS; ++i){
       if( (packet_sz[i] + p_sz_rm) > b_size)
           if(packet_sz[i] > b_size)/*compare the packet size with bucket size*/
               printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity
                           (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
           else
                printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
       else {
           p_sz_rm += packet_sz[i];
           printf("\n\nIncoming Packet size: %d", packet_sz[i]);
           printf("\nBytes remaining to Transmit: %d", p_sz_rm);
           p_time = rand(4) * 10;
```

```
            printf("\nTime left for transmission: %d units", p_time);
           for(clk = 10; clk <= p_time; clk += 10) {
               sleep(1);
                if(p_sz_rm) {
                     if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                         op = p_sz_rm, p_sz_rm = 0;
                     else
                         op = o_rate, p_sz_rm -= o_rate;
                      printf("\nPacket of size %d Transmitted", op);
                      printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
                else {
                   printf("\nTime left for transmission: %d units", p_time-clk);
                   printf("\nNo packets to transmit!!");
}}}}
return 0;
}
```

OUTPUT:

```
packet[0]:30 bytes
packet[1]:10 bytes
packet[2]:10 bytes
packet[3]:50 bytes
packet[4]:30 bytes
packet[5]:50 bytes
packet[6]:10 bytes
packet[7]:20 bytes
packet[8]:30 bytes
packet[9]:10 bytes
Enter the Output rate:100
Enter the Bucket Size:50
Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 20 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 30 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!
Incoming Packet size: 10
Bytes remaining to Transmit: 10          Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 50
Bytes remaining to Transmit: 50
Time left for transmission: 10 units
Packet of size 50 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 30 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 50
```

Bytes remaining to Transmit: 50
Time left for transmission: 20 units
Packet of size 50 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 20 units
Packet of size 20 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 20 units
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!
Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

2 Write a program for congestion control using Leaky bucket algorithm.

Algorithm: 1. Start
2. Set the bucket size or the buffer size
3. Set the output rate
4. Transmit the packets such that there is no overflow
5. Repeat the process of transmission until all packets are transmitted.
6. Stop.

```
#include <iostream>
#include <string.h>
using namespace std;
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NOF_PACKETS 10
int rand(int a) {
    int rn = (random() % 10) % a;
    return rn == 0 ? 1 : rn;
}
int main() {
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate;
p_sz = rm = 0, p_sz, p_time, op;
    for (i=0; i<NOF_PACKETS; ++i)
        packet_sz[i] = rand(6)*10;
    for (i=0; i<NOF_PACKETS; ++i)
        printf("\n packet[%d]: %d bytes\t", i, packet_sz[i]);
        printf("\n Enter the Output rate:");
        scanf("%d", &o_rate);
        printf("Enter the Bucket Size:");
        scanf("%d", &b_size);
        for (i=0; i<NOF_PACKETS; ++i)
```

```
{
    if((packet_sz[i] + p_sz_rm) > b_size)
    if (packet_sz[i] > b_size)
      printf("\n In Incoming packet size (%d bytes) is
Greater than bucket capacity (%d bytes) - PACKET REJECTED",
packet_sz[i], b_size);
    else
      printf("\n In Bucket capacity exceeded - PACKETS
REJECTED!!");
    else {
      printf
      p_sz_rm += packet_sz[i];
      printf("\n In Incoming Packet size: %d", packet_sz[i]);
      printf("\n Bytes remaining to transmit: %d",
p_sz_rm);
      p_time = rand(4)*10;
      printf("\n Time left for transmission: %d units",
p_time);
      for (clk = 10; clk <= p_time; clk += 10)
      {
        sleep(1);
        if (p_sz_rm)
        {
        if (p_sz_rm <= 10_rate)
          p_sz_rm = p_sz_rm = 0;
        else
          op = 0_rate; p_sz_rm -= 0_rate;
        printf("\n Packet of size %d Transmitted", op);
        printf("\n Bytes Remaining to Transmit: %d", p_sz_rm);
      }
      else {
        printf("\n Time left for transmission: %d units", p_time-clk
        printf("\n No packets to transmit");
      }
    }
  }
```

packet[0]: 30 bytes
packet[1]: 10 bytes
packet[2]: 10 bytes
packet[3]: 50 bytes
packet[4]: 30 bytes
packet[5]: 50 bytes
packet[6]: 10 bytes
packet[7]: 20 bytes
packet[8]: 30 bytes
packet[9]: 10 bytes

Enter the Output rate: 100
Enter the Bucket Size: 50
Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 20 units
Packet of size 30 Transmitted --- Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10
Packet of size 10 Transmitted --- Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 50
Bytes remaining to Transmit: 50
Time left for transmission: 10 units
Packet of size 50 Thousand --- Bytes Remaining to Transmit

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 30 units.
Packet of size 30 transmitted --- Bytes Remaining to Transmit: 0
Time left for transmission: 10 units.
No packets to transmit!!
Time left for transmission: 0 units.
No packets to transmit!!

Incoming Packet size: 50
Bytes remaining to Transmit: 50.
Time left for transmission: 20 units.
Packet of size 50 Transmitted --- Bytes Remaining to Transmit: 0
Time left for transmission: 0 units.
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10.
Time left for transmission: 10 unit.
Packet of size 10 Transmitted --- Bytes Remaining to Transmit: 0.

30/12/24

Program 15:

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

Algorithm:

Client Side

1. Start.
2. Create a socket using the socket() system call.
3. Connect the socket to the server's address using the connect() system call.
4. Send the filename of the required file using the send() system call.
5. Read the contents of the file sent by the server using the recv() system call.
6. Stop.

Code:

```
#include <unistd.h>
int main()
{
    int soc, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    /* socket creates an endpoint for communication and returns a file descriptor */
    soc = socket(PF_INET, SOCK_STREAM, 0);
    /*
    * sockaddr_in is used for ip manipulation
    * we define the port and IP for the connection.
    */
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    /* keep trying to establish connection with server */
    while(connect(soc, (struct sockaddr *) &addr, sizeof(addr))) ;
        printf("\nClient is connected to Server");
    printf("\nEnter file name: ");
    scanf("%s", fname);
     /* send the filename to the server */
    send(soc, fname, sizeof(fname), 0);
    printf("\nRecieved response\n");0
     /* keep printing any data received from the server */
     while ((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
         printf("%s", buffer);
     return 0;
}
```

Algorithm:

Server Side

1. Start.
2. Create a socket using socket() system call.
3. Bind the socket to an address using bind() system call.
4. Listen to the connection using listen() system call.
5. accept connection using accept()
6. Receive filename and transfer contents of file with client.
7. Stop.

Code:

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
    int welcome, new_soc, fd, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    welcome = socket(PF_INET, SOCK_STREAM, 0);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
    printf("\nServer is Online");
    /* listen for connections from the socket */
    listen(welcome, 5);
    /* accept a connection, we get a file descriptor */
   new_soc = accept(welcome, NULL, NULL);
   /* receive the filename */
   recv(new_soc, fname, 50, 0);
   printf("\nRequesting for file: %s\n", fname);
   /* open the file and send its contents */
   fd = open(fname, O_RDONLY);
   if (fd < 0)
       send(new_soc, "\nFile not found\n", 15, 0);
   else
      while ((n = read(fd, buffer, sizeof(buffer))) > 0)
  send(new_soc, buffer, n, 0);
  printf("\nRequest sent\n");
  close(fd);
  return 0;
}
```

OUTPUT:

Server is Online.
Requesting for file : test.txt
Request sent.

Client is connected to server
Enter file name : test.txt
Received Response
Hello World.

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

Client side

```
#include <unistd.h>
int main() {
    int soc, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    soc = socket(PF_INET, SOCK_STREAM, 0);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    while(connect(soc, (struct sockaddr *) &addr, sizeof(addr)) == printf("\n Client is connected to Server");
    printf("\n Enter file name :");
    scanf("%s", fname);
    send(soc, fname, sizeof(fname), 0);
    printf("\n Received response \n");
    while((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
    printf("%s", buffer);
    return 0;
}
```

Server side

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>
int main() {
    int welcome, new_soc, fd, n;
```

```
char buffer[1024], fname[50];
struct sockaddr_in addr;
welcome = socket(PF_INET, SOCK_STREAM, 0);
addr.sin_family = AF_INET;
addr.sin_port = htons(7890);
addr.sin_addr.s_addr = inet_addr("127.0.0.1");
bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
printf("\n Server is Online");
listen(welcome, 5);
new_soc = accept(welcome, NULL, NULL);
recv(new_soc, fname, 50, 0);
printf("\n Requesting for file: %s\n", fname);
fd = open(fname, O_RDONLY);
if(fd<0)
send(new_soc, "\n File not found\n", 15, 0);
else
while((n = read(fd, buffer, sizeof(buffer)))>0)
send(new_soc, buffer, n, 0);
printf("\n Request sent\n");
close(fd);
return 0;
}
```

Output:

```
Server is online
Requesting for file: dept.txt
Request sent
```

30/9/24

```
Client is connected to server.
Enter file name: dept.txt
Received response.
Hello world.
```

4) Ping UDP

make client

Send back the

```
// Server pro
#include <add
#include <sta
#include <sy
#include <un
#include <sy
#include <ne
#define POR
#define MA
int main() {
  char buff
  char *mes
  int listen
  struct soc
  bzero(&s
  listenfd =
  servaddr.
  servaddr.
  servaddr.
  bind(liste
  sizeof(serv
  len = size
  int n = 0
  (struct so
  buffer[n]
  puts(buff
  send to Cl
(&cliaddr, size
}
```

Program 16:

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

Code:

```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Client";
    int listenfd, len;
    struct sockaddr_in servaddr, cliaddr;
    bzero(&servaddr, sizeof(servaddr));
    // Create a UDP Socket
    listenfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    // bind server address to socket descriptor
    bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    //receive the datagram
    len = sizeof(cliaddr);
    int n = recvfrom(listenfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len);
    //receive message from server
    buffer[n] = '\0';
    puts(buffer);
    // send the response
    sendto(listenfd, message, MAXLINE, 0,(struct sockaddr*)&cliaddr, sizeof(cliaddr));
}


// udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
```

```
#include<unistd.h>
#include<stdlib.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Server";
    int sockfd, n;
    struct sockaddr_in servaddr;
    // clear servaddr
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    // create datagram socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    // connect to server
   if(connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        printf("\n Error : Connect Failed \n");
        exit(0);
    }
    // request to send datagram
    // no need to specify server address in sendto
    // connect stores the peers IP and port
    sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)NULL, sizeof(servaddr));
    // waiting for response
    recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)NULL, NULL);
    puts(buffer);
    // close the descriptor
    close(sockfd);
}
```

Output:

```
//Server output
Server is Online.
Hello Server

//Client Output
Hello Client
```

4) 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.

// Server program for udp connection.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
int main() {
    char buffer[100];
    char *message = "Hello Client";
    int listenfd, len;
    struct sockaddr_in servaddr, cliaddr;
    bzero(&servaddr, sizeof(servaddr));
    listenfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    bind(listenfd, (struct sockaddr *) &servaddr,
sizeof(servaddr));
    len = sizeof(cliaddr);
    int n = recvfrom(listenfd, buffer, sizeof(buffer),
0, (struct sockaddr *) &cliaddr, &len);
    buffer[n] = '\0';
    puts(buffer);
    sendto(listenfd, message, MAXLINE, 0, (struct sockaddr
*) &cliaddr, sizeof(cliaddr));
}
```

classmate
Date
Page 38

Udp client driver program.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 5000
#define MAXLINE 1000
int main(){
    char buffer[100];
    char *message = "Hello Server";
    int sockfd, n;
    struct sockaddr_in servaddr;
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if(connect(sockfd, (struct sockaddr*)&servaddr,
sizeof(servaddr))<0)
    {
        printf("\n Error: Connect Failed \n");
        exit(0);
    }
    sendto(sockfd, message, MAXLINE, 0, (struct
sockaddr*) NULL, sizeof(servaddr));
    recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*) NULL, NULL);
    puts(buffer);
    close(sockfd);
}
```