VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (22CS4PCCON)

Submitted by

TANMAY BHARADWAJ (1BM22CS303)

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 (22CS4PCCON)"** carried out by **TANMAY BHARADWAJ (1BM22CS303),** 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. Sneha P Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

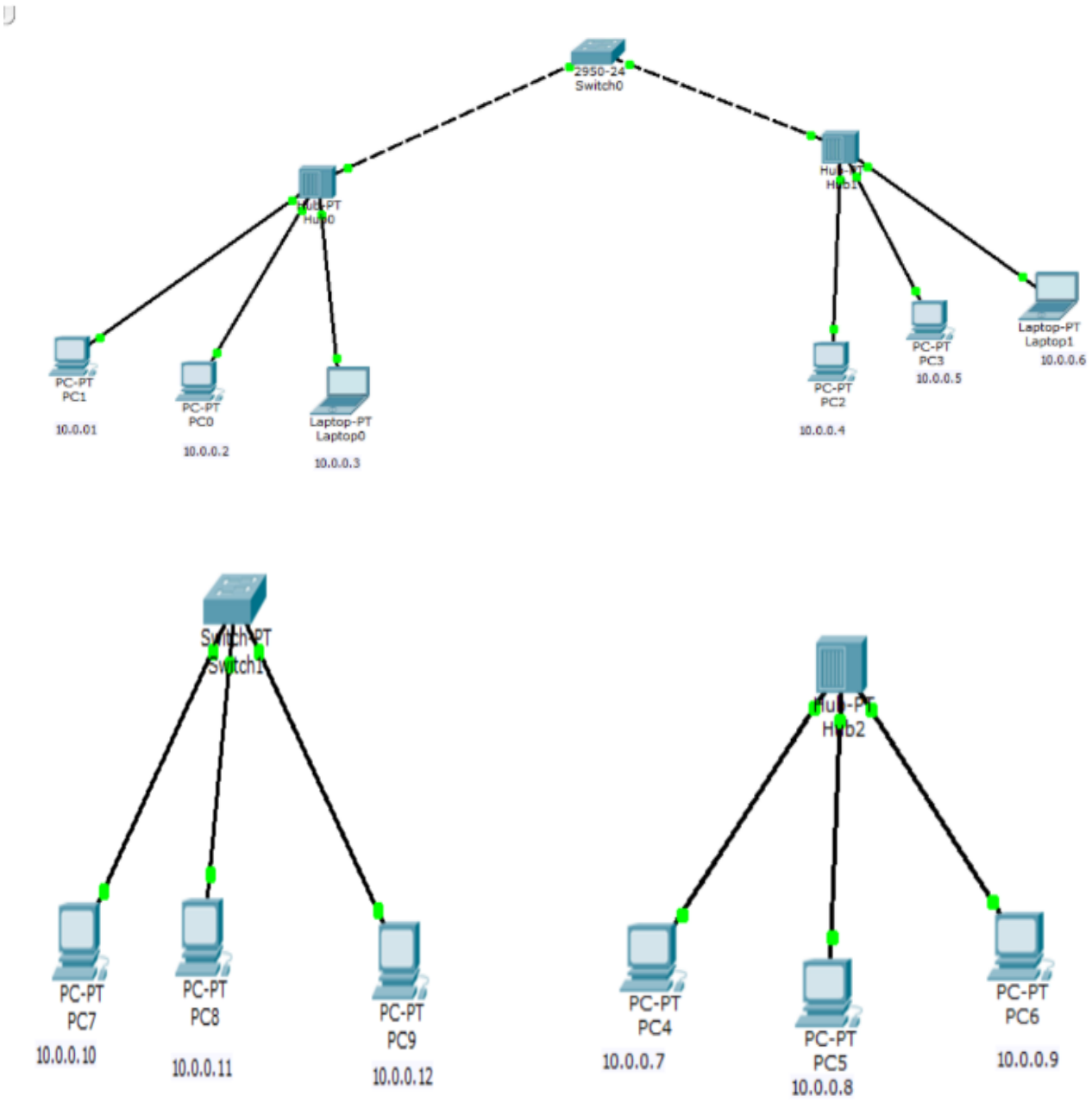
Index

Github Link : https://github.com/TanmayBj23/CN_LAB

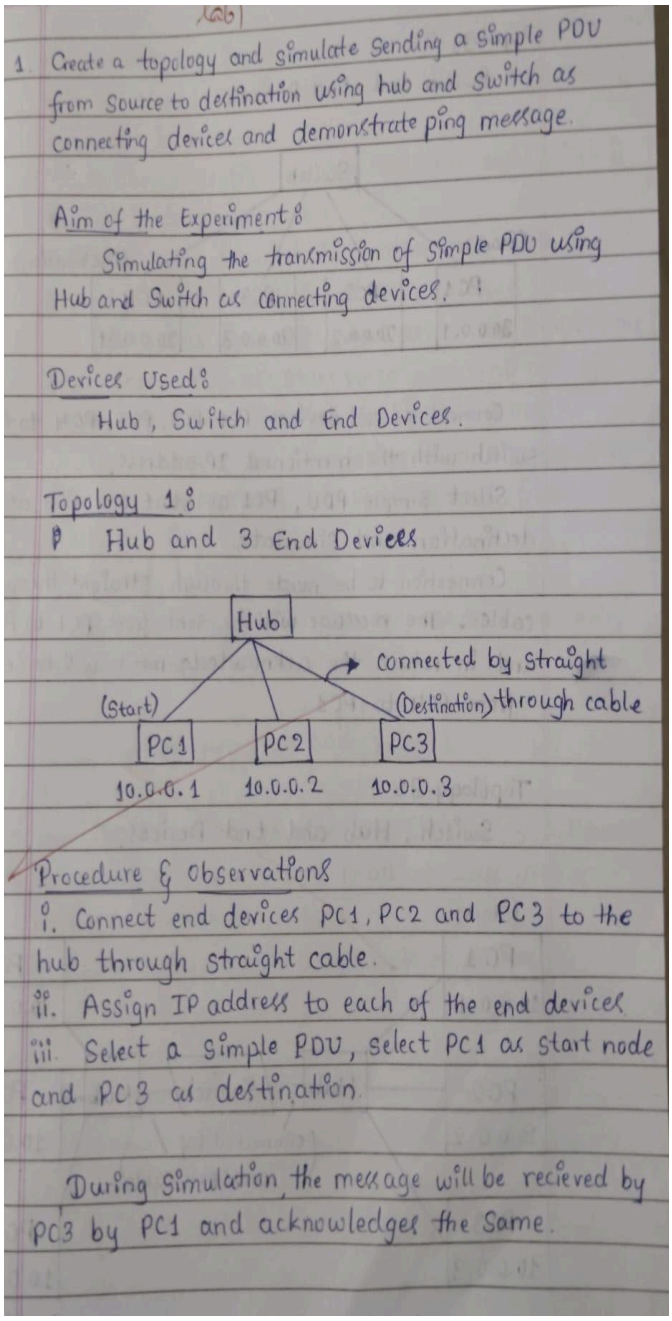
Program 1:

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

Topology:



Procedure and Observations:



Lab 1

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.

Aim of the Experiment:

Simulating the transmission of simple PDU using Hub and Switch as connecting devices.

Devices Used:

Hub, Switch and End Devices.

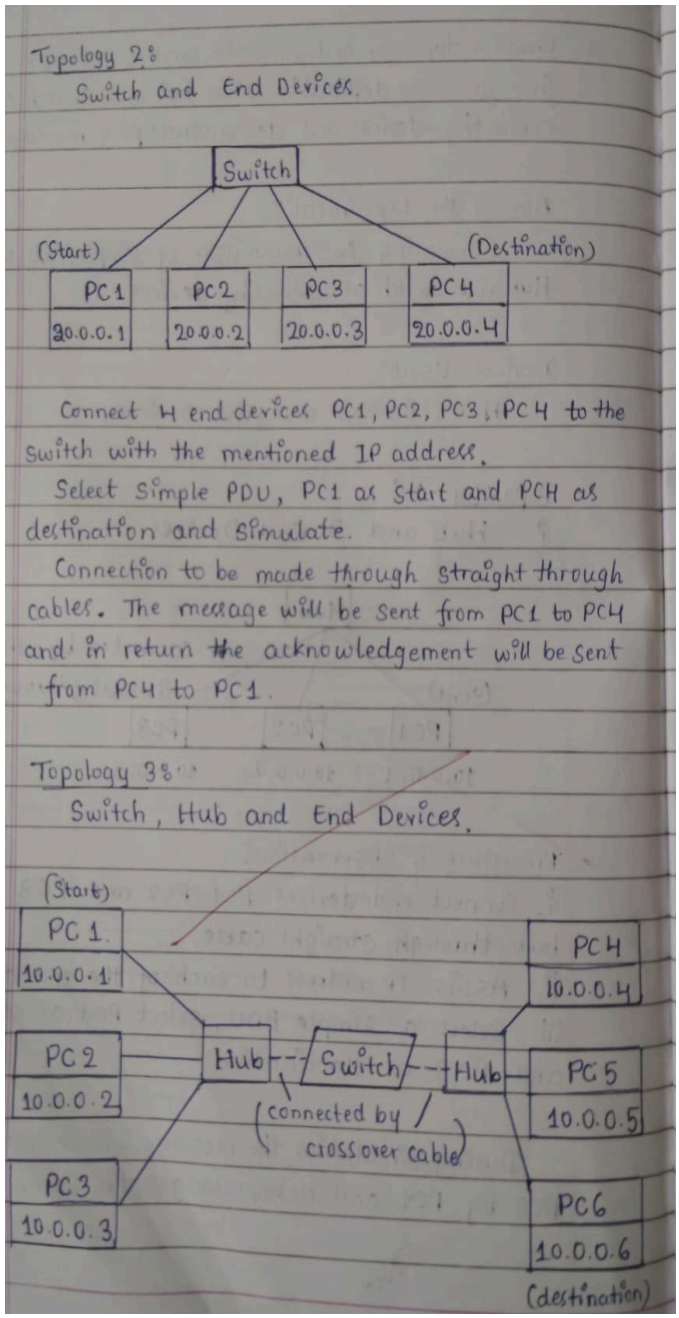
Topology 1:

Hub and 3 End Devices

Procedure & 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 recieved 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 the mentioned IP address.

Select simple PDU, PC1 as start and PC4 as 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 is connected to a Switch.

The connection between the Hub and Switch is through a cross over cable.

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

Select a simple PDU and assign any one of the first three PCs as start node and any one of the other three PCs as destination node.

Demonstrate the simulation and analyse the flow of message and acknowledgement from PC1 to PC6.

The successful ping message confirms the connectivity between the source and destination.

Difference between Hub and Switch:

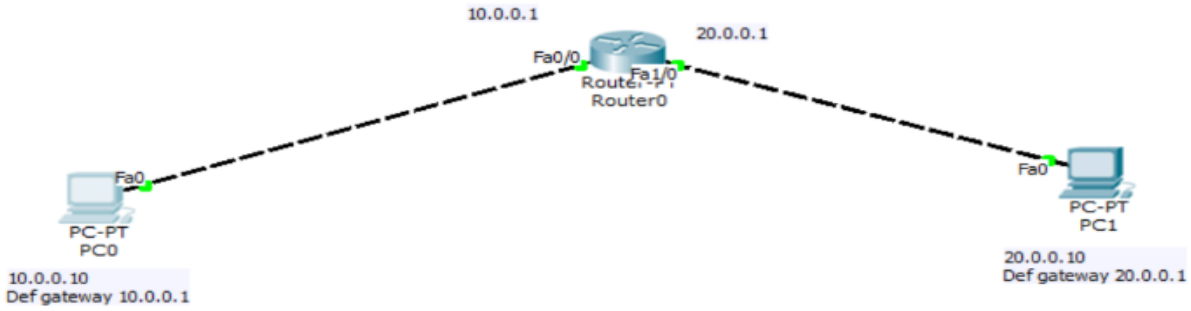
- Hub operates at the physical layer (Layer 1) of OSI model.
- It broadcasts data packets to all connected devices regardless of intended recipents.
- It is less efficient and supports lower speeds.

- Switch operates at Data Link layer (Layer 2) of OSI model.
- It broadcasts data packets only to specific device for which data is intended.
- It is more efficient and supports higher speeds.

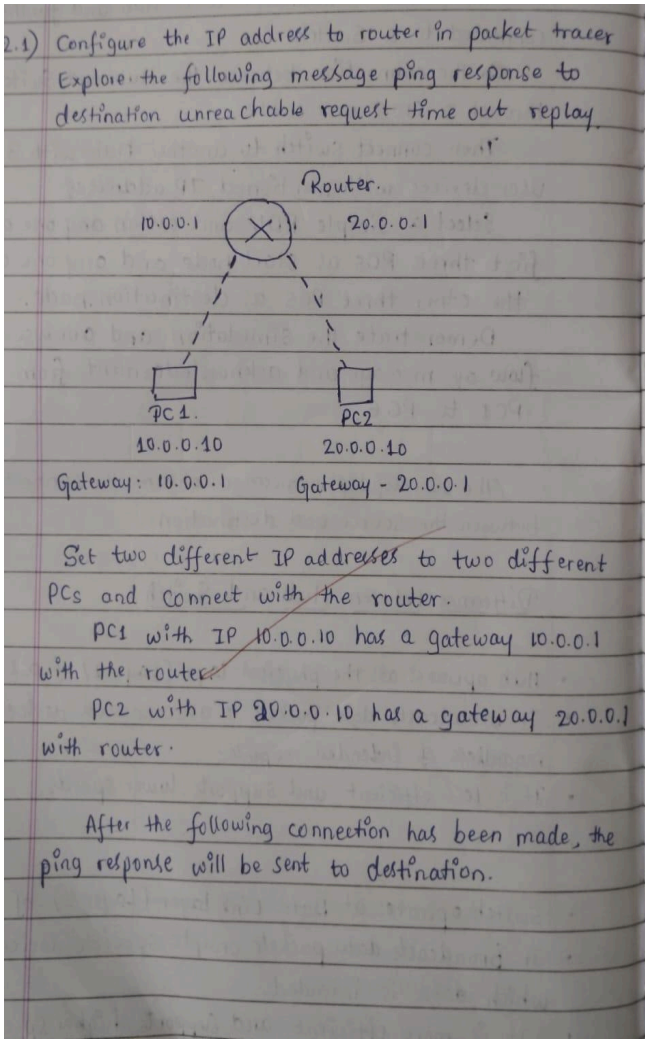
Program 2 :

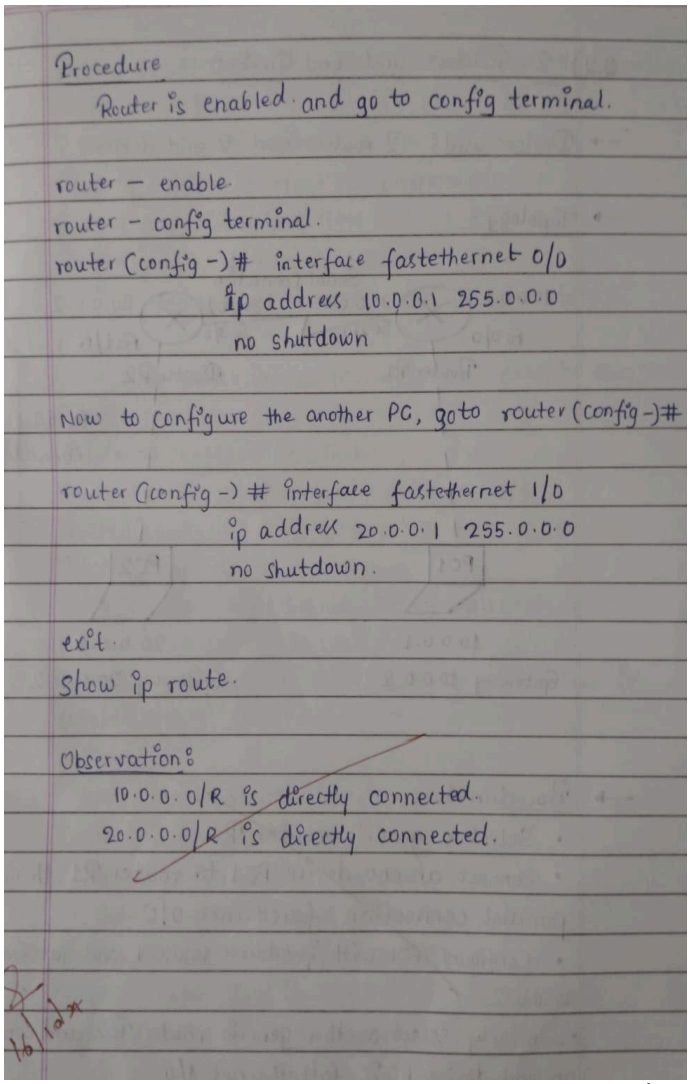
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

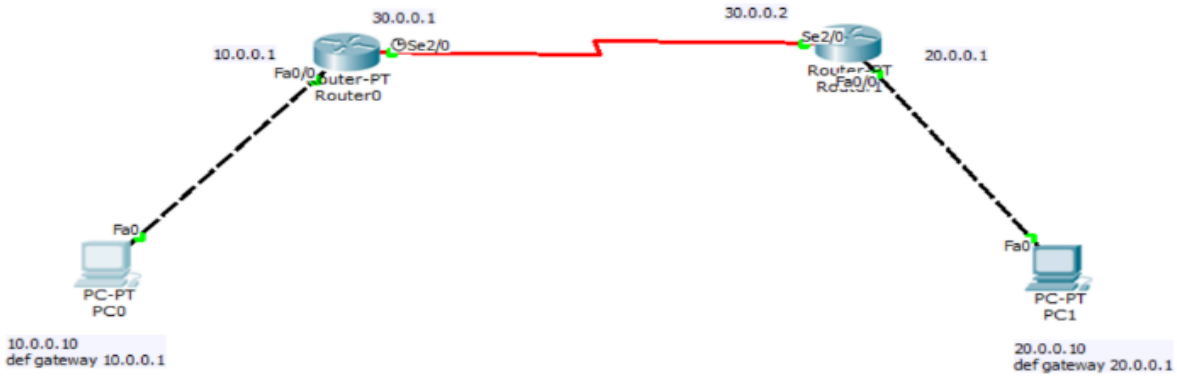
Topology:



Procedure and Observations:



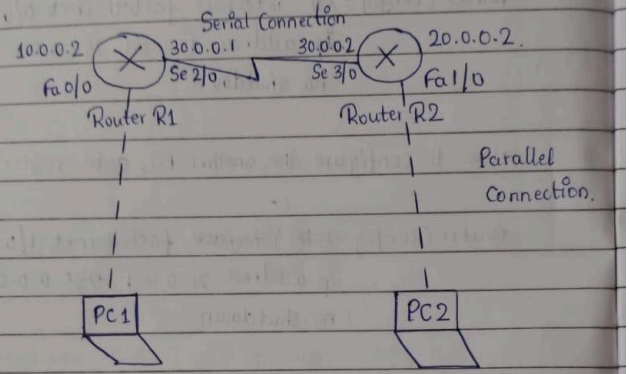




2.2) 2 Routers and End Devices.

→ Devices used: 2 routers and 2 end devices.

→ Topology:



→ 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 10.0.0.2.
- 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 following

```
Router > enable
Router# configure 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. goto CLI execute the same.

```
Router > enable
Router# configure 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 Router & 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 3/0 changed state to up"

Observations:

→ After setting up the mentioned topology, Now try to ping PC2 with PC1.

Open command prompt for PC1 type ping 20.0.0.1

→ Destination host unreachable.

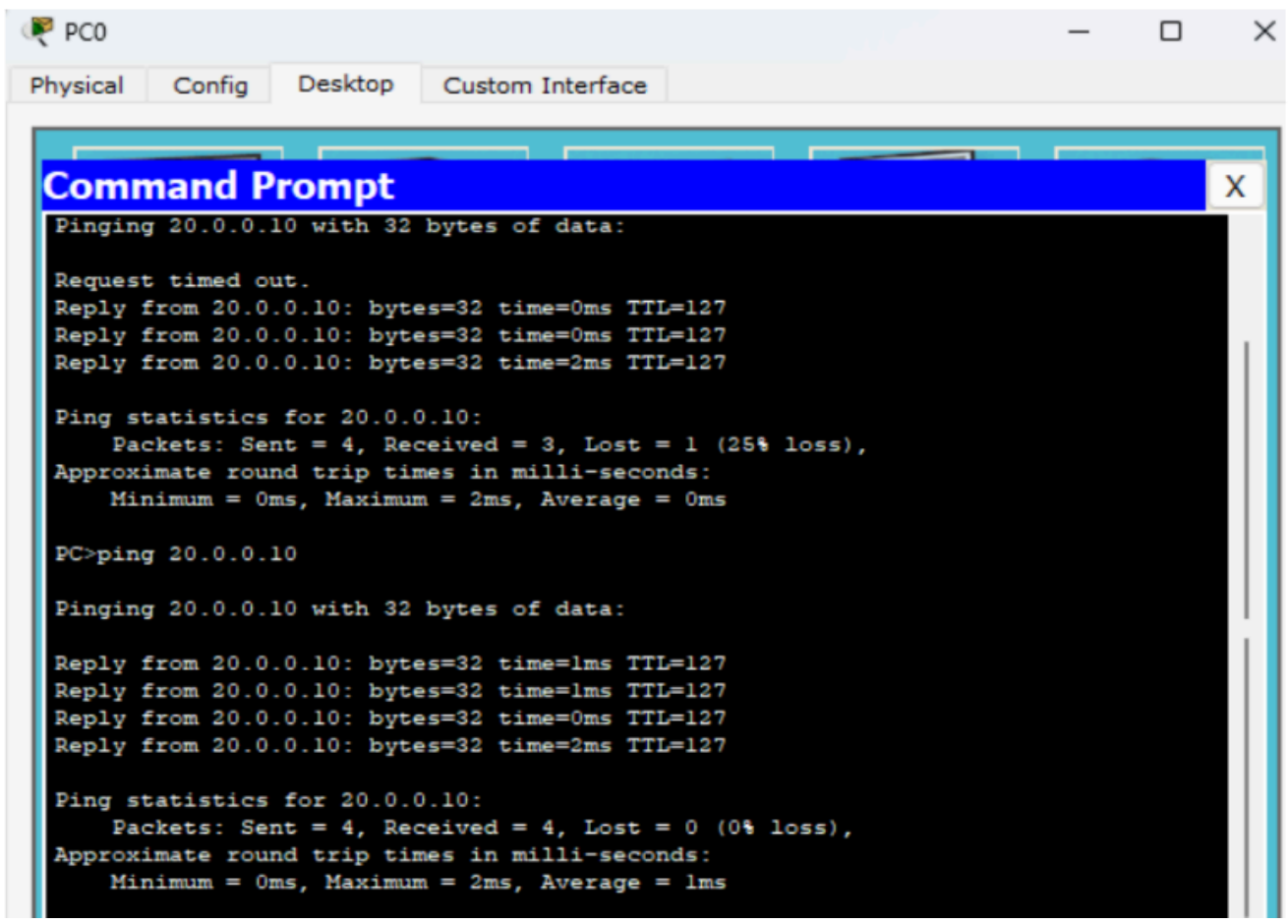
Packets Sent: 4 recieved: 0 lost: 4 Loss = 100%.

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

ping 30.0.0.1 → Successful.

Packets Sent: 4 recieved: 4 lost: 0 Lost = 0 0%.

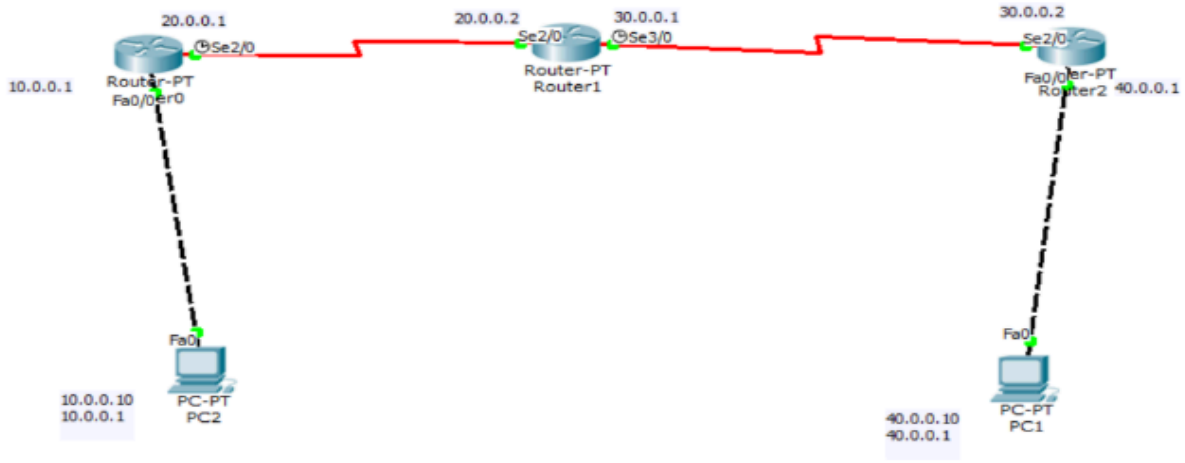
Hence although the routers were connected serially the end devices were unable to ping each other.

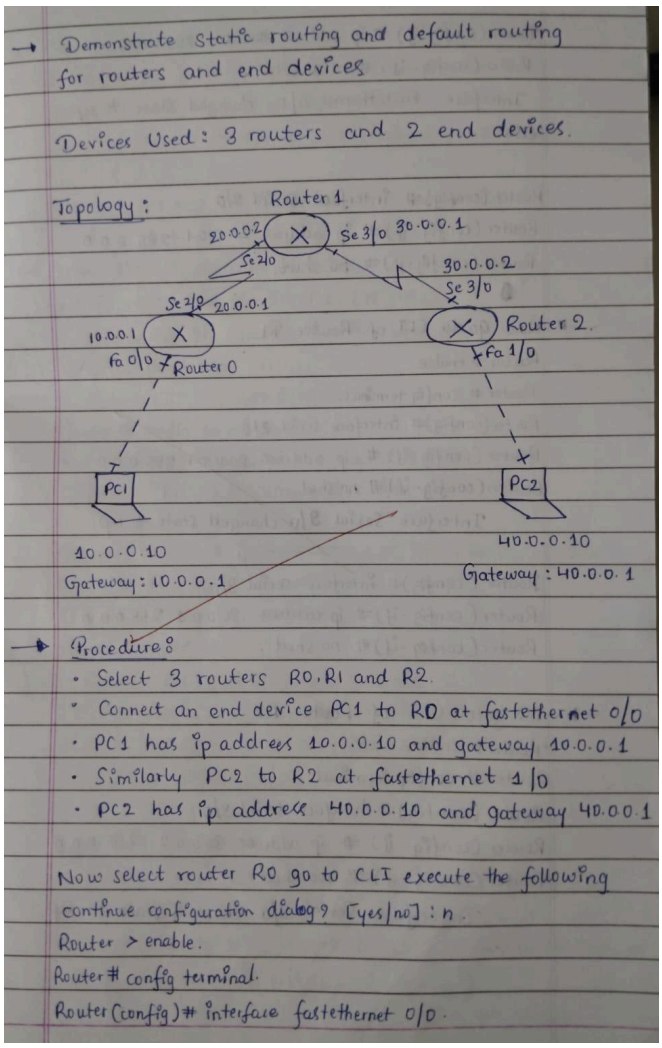


Program 3:

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

Topology, Procedure and Observations :





→ Demonstrate Static routing and default routing for routers and end devices

Devices Used : 3 routers and 2 end devices.

Topology :

→ Procedure :

- Select 3 routers R0, R1 and R2.
- Connect an end device PC1 to R0 at fastethernet 0/0
- PC1 has ip address 10.0.0.10 and gateway 10.0.0.1
- Similarly PC2 to R2 at fastethernet 1/0
- PC2 has ip address 40.0.0.10 and gateway 40.0.0.1

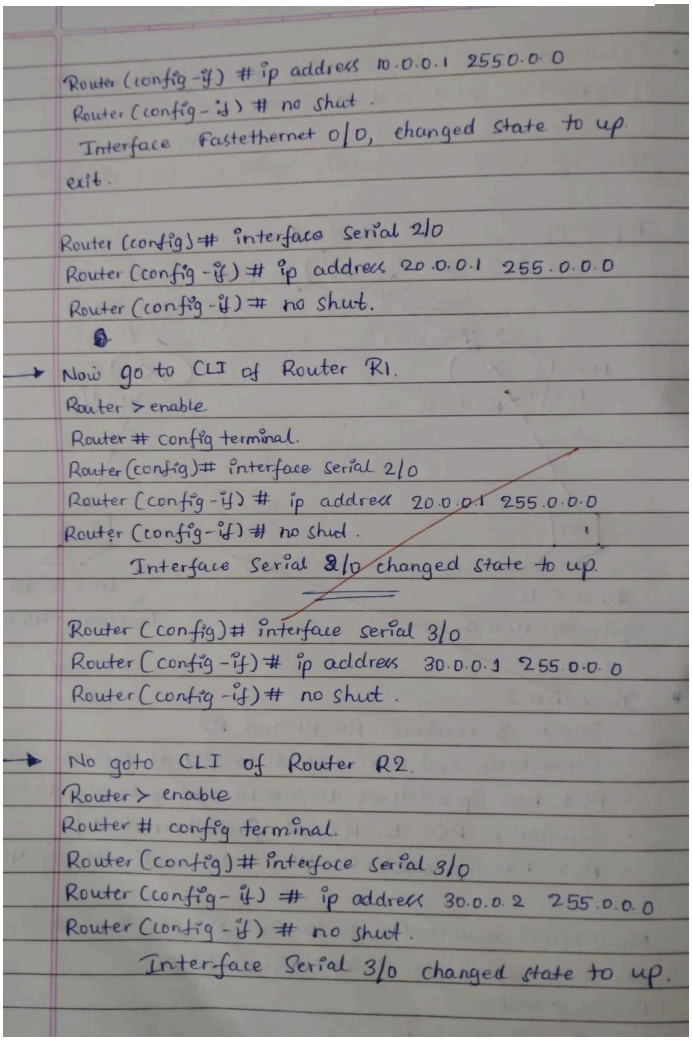
Now select router R0 go to CLI execute the following

continue configuration dialog? [yes/no] : n.

Router > enable.

Router # config terminal.

Router (config) # interface fastethernet 0/0.



Router (config-if) # ip address 10.0.0.1 255.0.0.0

Router (config-if) # no shut.

Interface Fastethernet 0/0, changed state to up

exit.

Router (config) # interface serial 2/0

Router (config-if) # ip address 20.0.0.1 255.0.0.0

Router (config-if) # no shut.

→ Now go to CLI of Router R1.

Router > enable.

Router # config terminal.

Router (config) # interface serial 2/0

Router (config-if) # ip address 20.0.0.1 255.0.0.0

Router (config-if) # no shut.

Interface Serial 2/0 changed state to up

Router (config) # interface serial 3/0

Router (config-if) # ip address 30.0.0.1 255.0.0.0

Router (config-if) # no shut.

→ No goto CLI of Router R2.

Router > enable

Router # config terminal.

Router (config) # interface serial 3/0

Router (config-if) # ip address 30.0.0.2 255.0.0.0

Router (config-if) # no shut.

Interface Serial 3/0 changed state to up.

Router(config)# interface fast ethernet 1/0
Router(config-if)# ip address 40.0.0.1 255.0.0.0
Router(config-if)# no shut
Interface FastEthernet 1/0 changed state to up.

Hence topology configuration is completed.

Now there is no communication b/w R0, R1, R2 as R1 is not aware of R0, R2 beside.
-We establish communication b/w them using Static Routing.
Goto CLI of router1 again.
Router# config terminal.
Router(config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1.
→ R0, R1 communication achieved.

Router(config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2
→ R1, R2 communication achieved.

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, Serial 2/0
C 30.0.0.0/8 is directly connected, Serial 3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2.

Dynamic Routing
To establish communication b/w R0 and R2
Goto CLI of R0,
Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
connection done.

Similarly goto CLI of router 2,
Router(config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1
Hence communication done.

Now go to Desktop of PC1.
ping 40.0.0.10 (ip address of PC2)

Packets: Sent = 4 recieved = 4 lost = 0 0% Loss.

Hence static routing and default routing is achieved.

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 = 6ms TTL = 125
Reply from 40.0.0.10: bytes = 32 time = 9ms TTL = 125
Reply from 40.0.0.10: bytes = 32 time = 6ms TTL = 125

Command Prompt

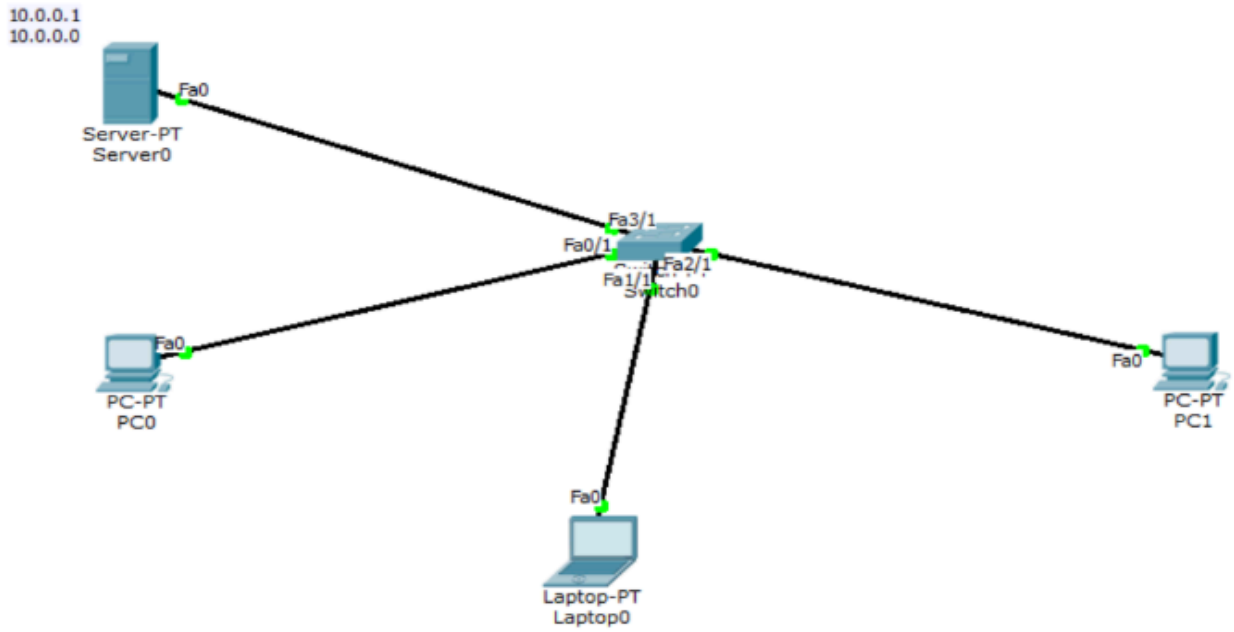
```
Pinging 40.0.0.10 with 32 bytes of data:

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:

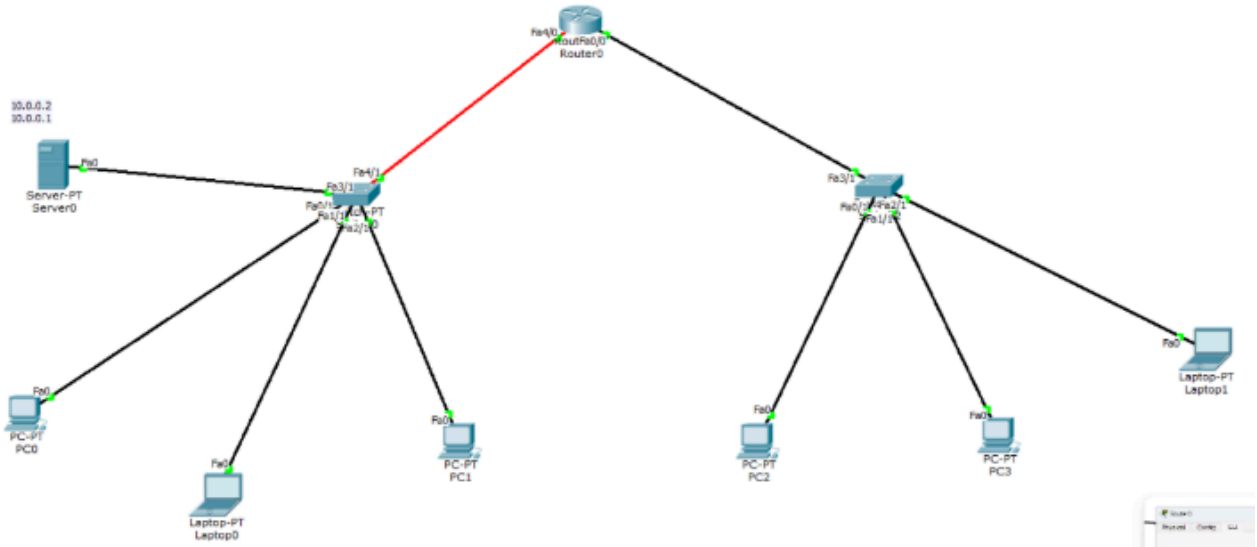
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

Within LAN



Outside LAN

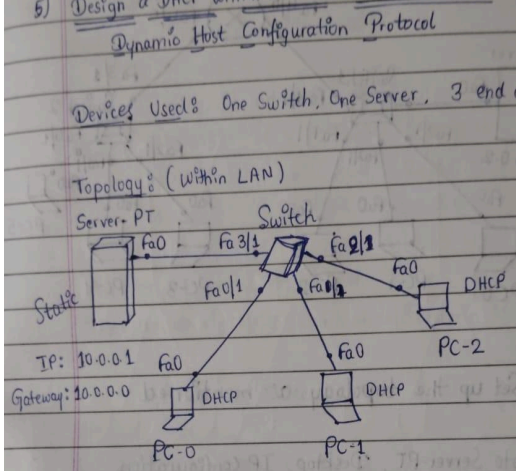


Procedure and Observation:

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

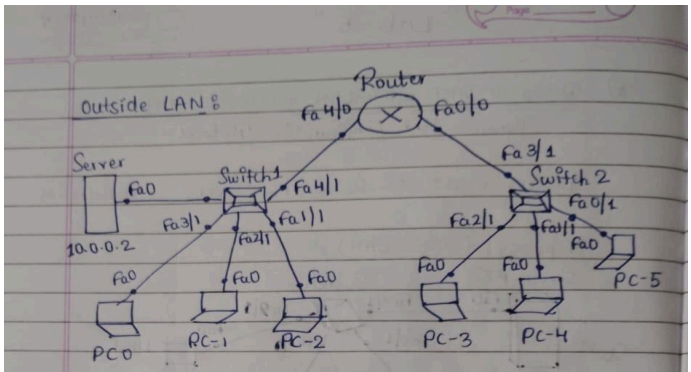
Devices Used: One Switch, One Server, 3 end devices

Topology: (within LAN)



→ Procedure:
Set up the topology as mentioned.
Goto Server IP configuration (Desktop)
Select IP address 10.0.0.1, 255.0.0.0, 10.0.0.0
Then to set up DHCP, goto Config, Servicer Select DHCP.
Make DHCP to all three PCs.
Dynamic ip addresses will be assigned.

Now ping PC-0 with PC-1 or PC-2.
PC-0 had 10.0.0.2 So goto Command Prompt
→ Ping was Successful ping 10.0.0.3



Set up the topology as mentioned

Goto 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

→ Interface established with Switch1 & Router

Similarly for Switch 2,
goto Router CLI, Router (config) # interface fastethernet 0/0
Router (config-if) # ip address 20.0.0.1 255.0.0.0
Router (config-if) # ip helper-address 10.0.0.2
Router (config-if) # no shut

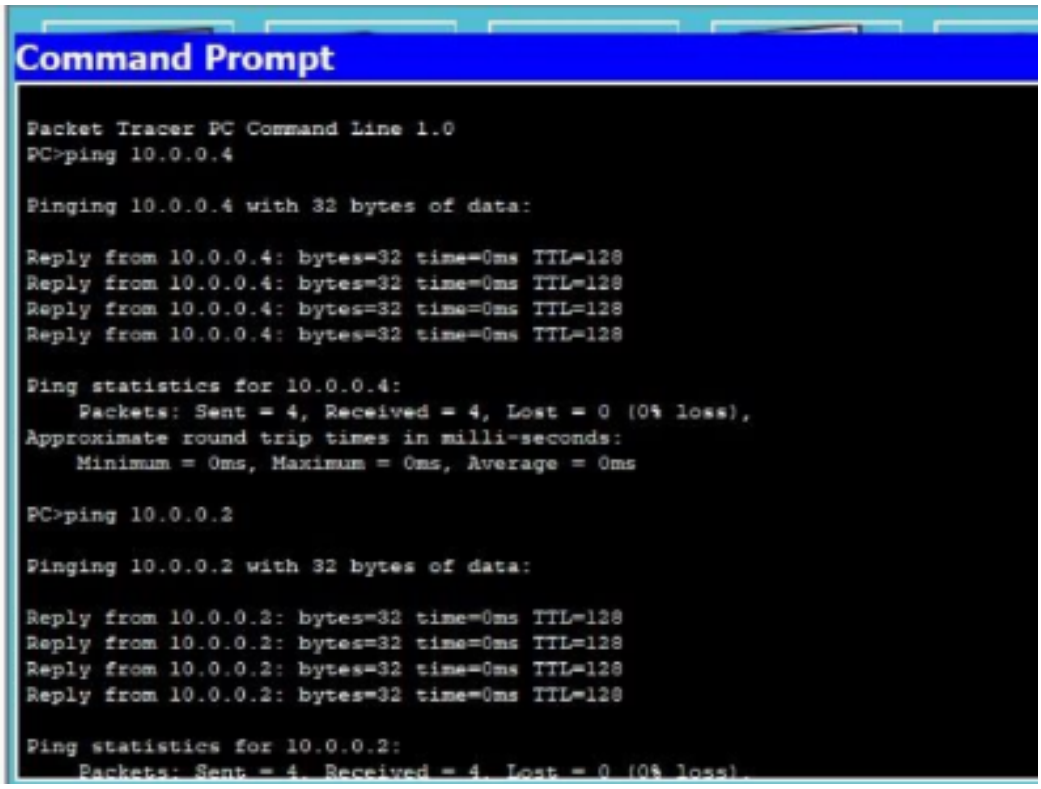
∴ Both networks have established connection with Switch1, Switch2 with router.

we can notice ip address of

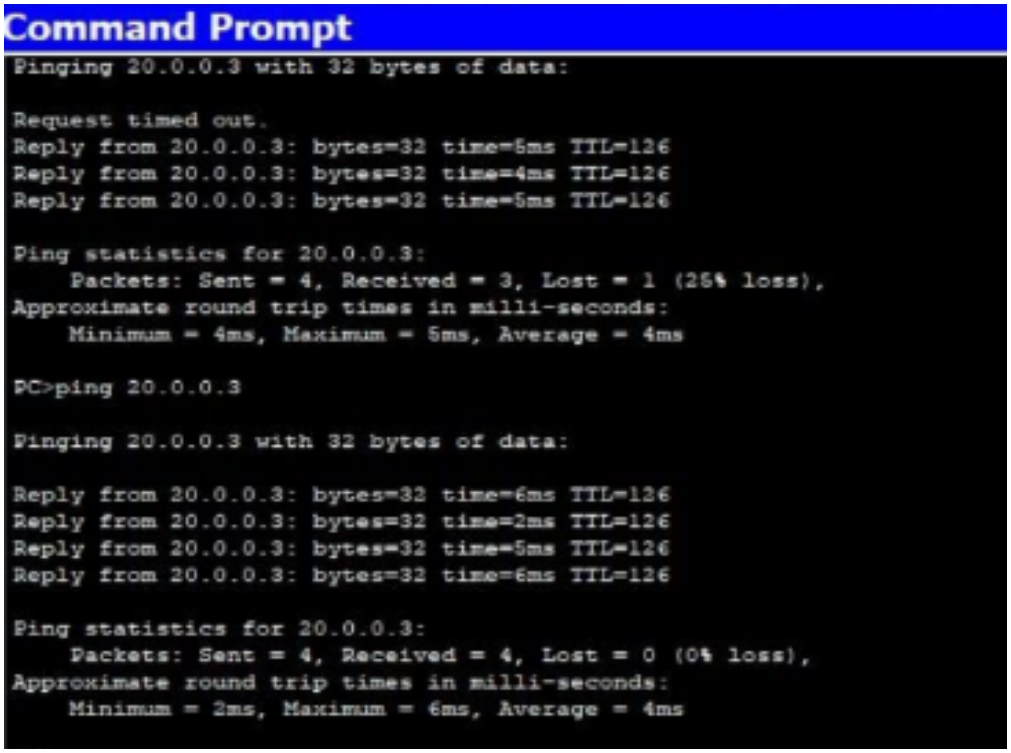
PC0 - 10.0.0.5	PC3 - 20.0.0.4
PC1 - 10.0.0.6	PC4 - 20.0.0.3
PC2 - 10.0.0.3	PC5 - 20.0.0.5

Now try to ping PC0 with PC5.
Goto Desktop of PC0, Command Prompt.

Ping was Successful



Within LAN

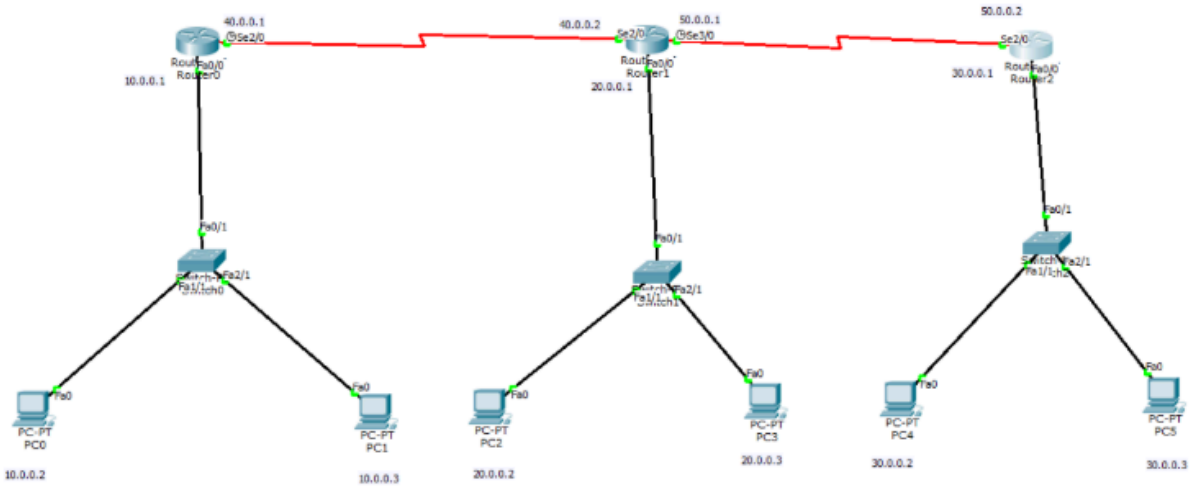


Outside LAN

Program 5:

Aim: Configure RIP routing Protocol in Routers.

Topology:



Procedure and Observation:

Procedure: Set up the topology as mentioned.

Set the ip configuration for routers.

Router R0 - 10.0.0.1 , R1 - 20.0.0.1, R2 - 30.0.0.1

PC0 - 10.0.0.2
PC1 - 10.0.0.3
} Gateway 10.0.0.1

PC2 - 20.0.0.2
PC3 - 20.0.0.3
} Gateway 20.0.0.1

PC4 - 30.0.0.2
PC5 - 30.0.0.3
} Gateway 30.0.0.1.

To establish connection between routers, serially Goto CLI of routers and configure.

R0 - Se 2/0 - 40.0.0.1

R1 - Se 2/0 - 40.0.0.2 , Se 3/0 - 50.0.0.1

R2 - Se 2/0 - 50.0.0.2

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

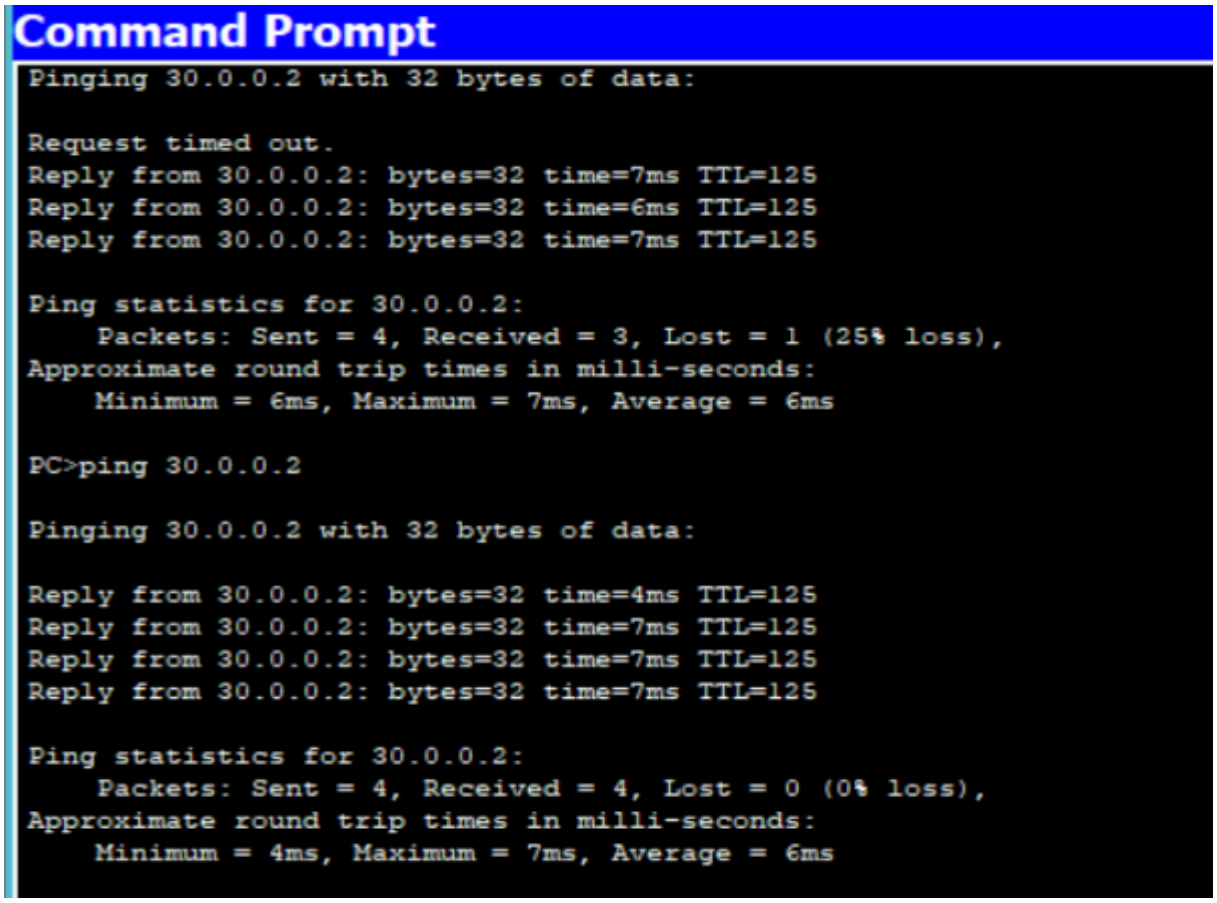
Router (config) # router rip
Router (config - router) # network 30.0.0.0
network 50.0.0.0.

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 = 9ms TTL = 125.

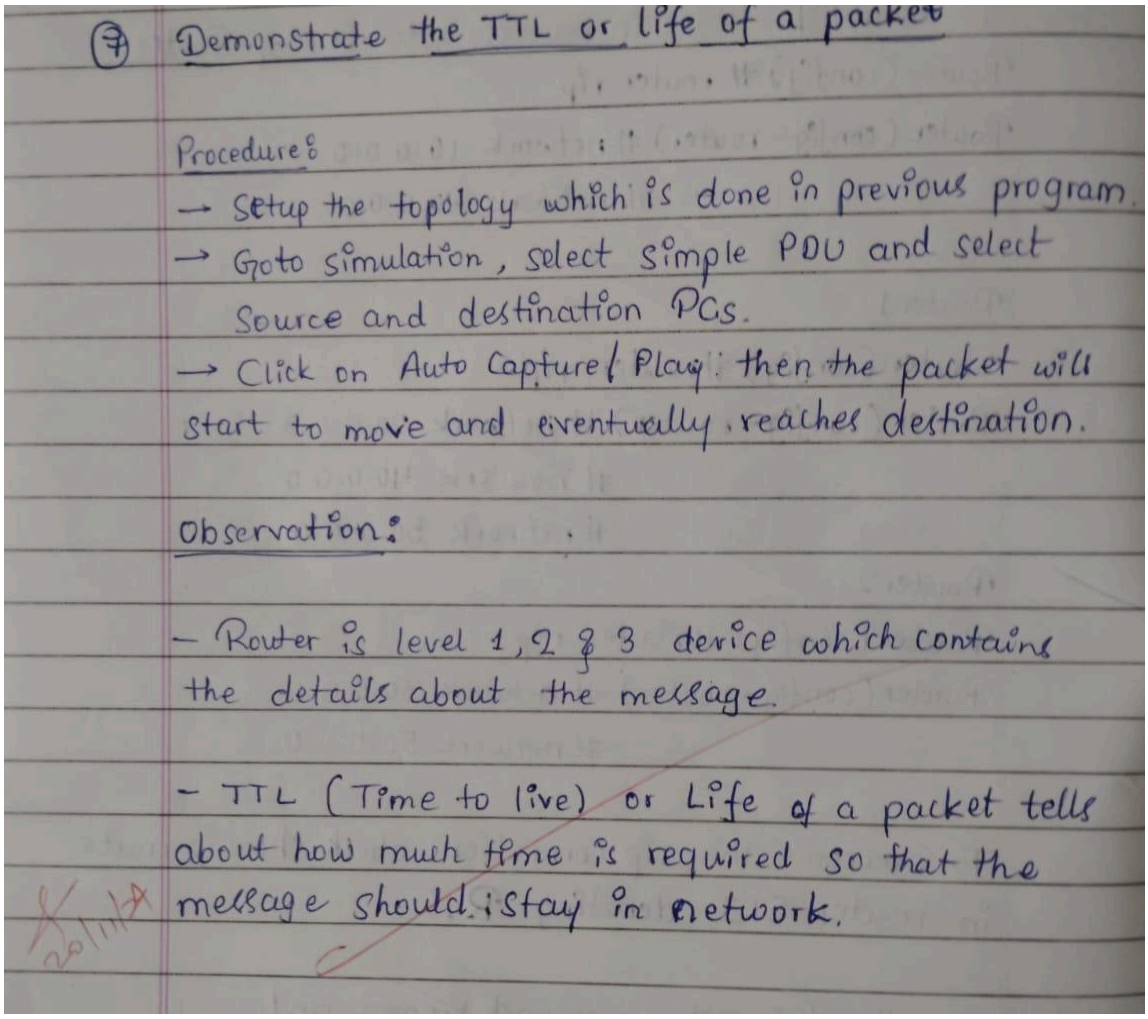
Ping statistics for 30.0.0.2:
Packets : Sent = 4, Recieved : 4, Lost = 0 (0% Loss)
App. round trip times in ms:
Min = 7ms, Max = 9ms, Avg = 7ms
∴ Ping was Successful


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 or life of a packet

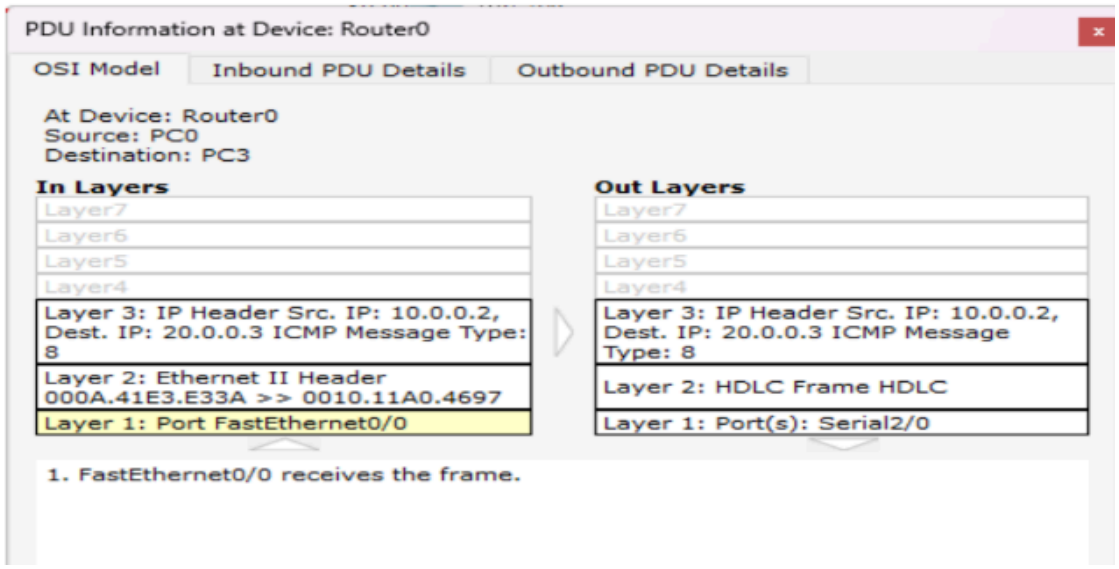
Procedure:

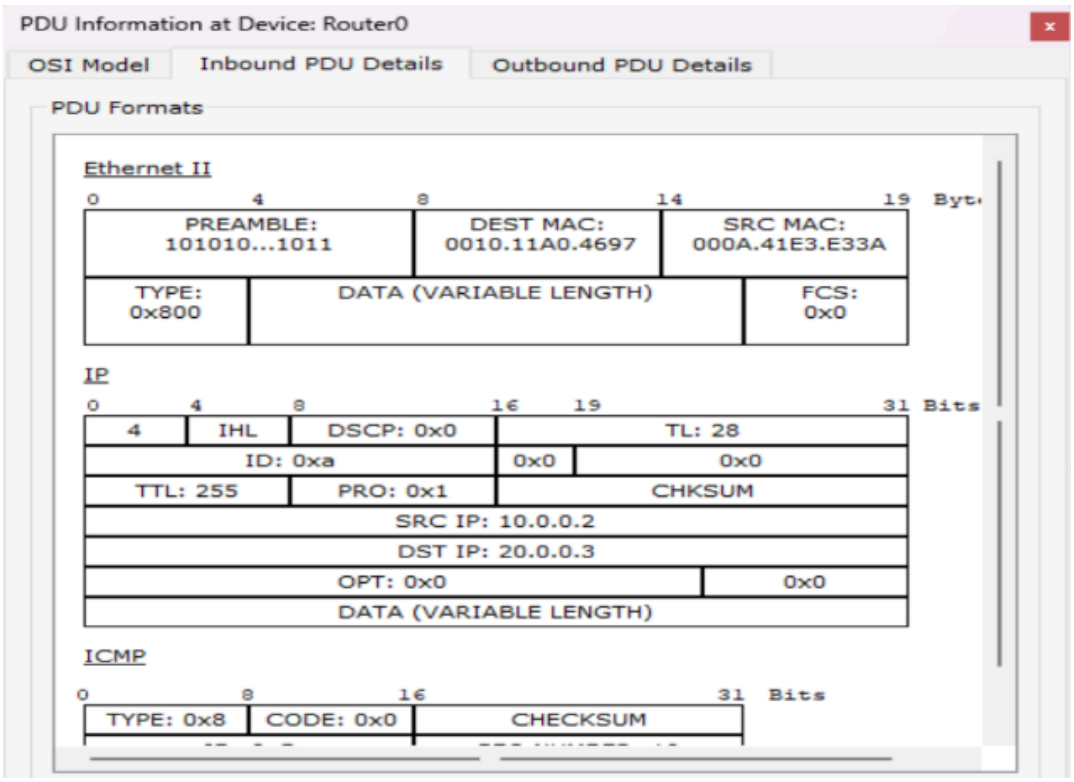
→ Setup the topology which is done in previous program.
→ Goto simulation, select simple PDU and select Source and destination PCs.
→ Click on Auto Capture/ Play: then the packet will start to move and eventually reaches destination.

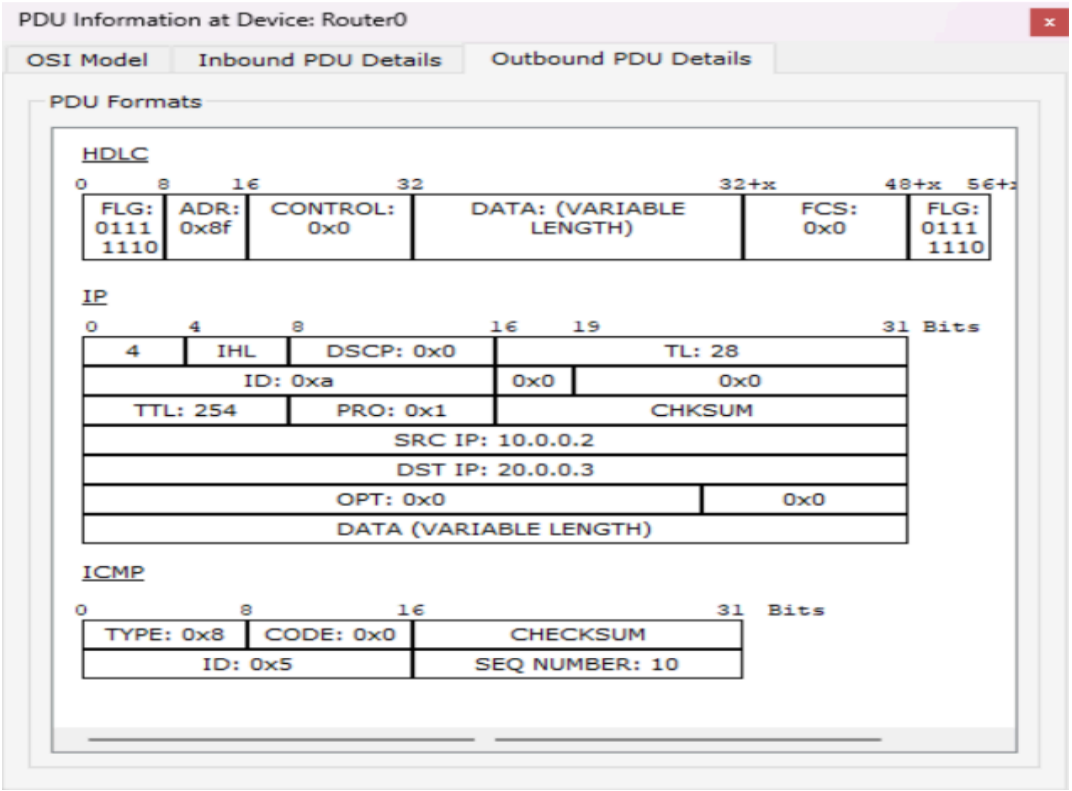
Observation:

- 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.



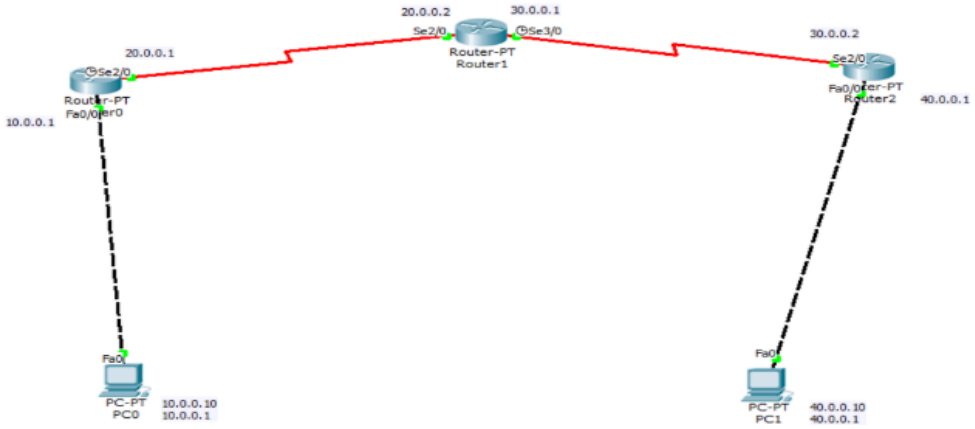

PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Byt
PREAMBLE: 101010...1011
DEST MAC: 0010.11A0.4697
SRC MAC: 000A.41E3.E33A
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
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 56+
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

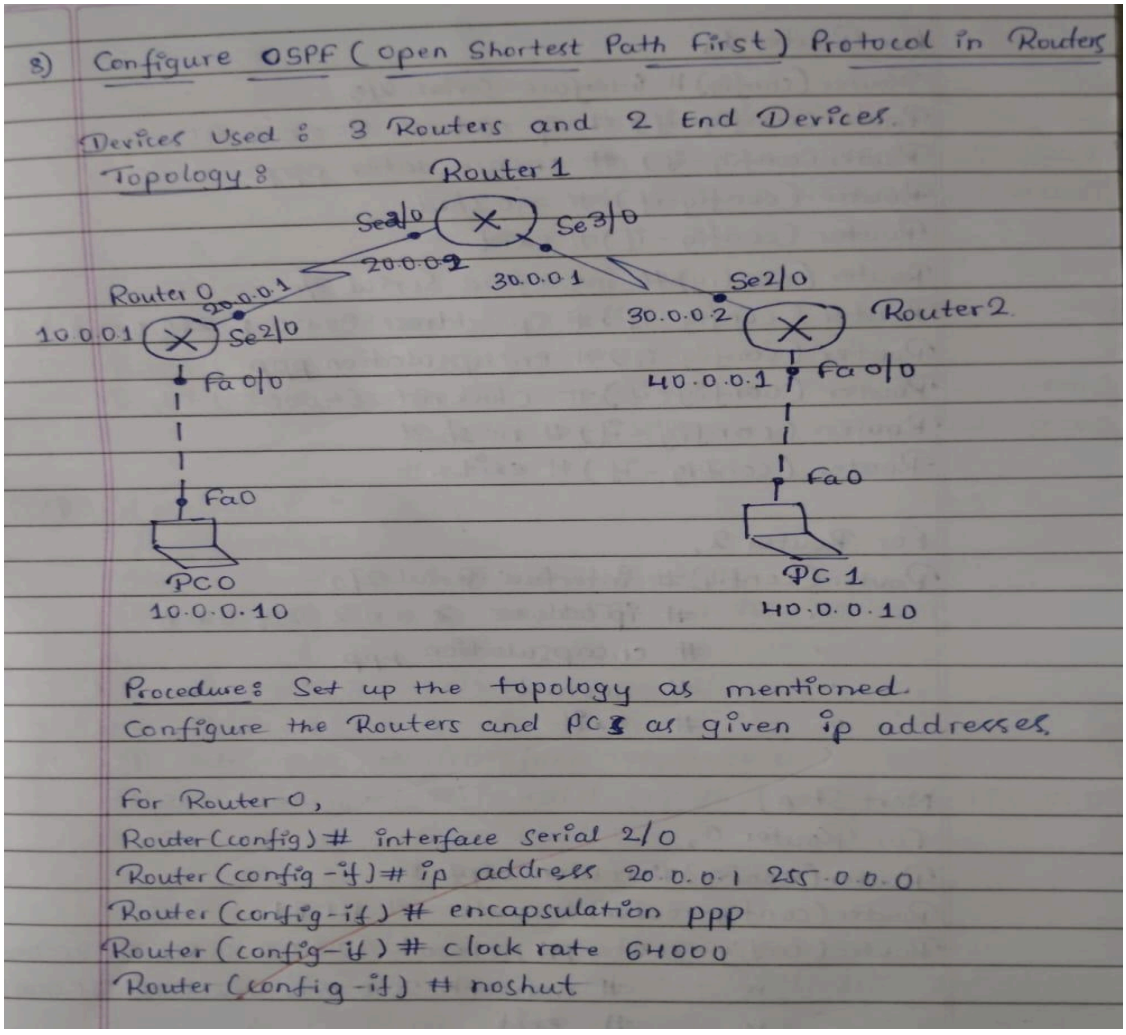
Program 7:

Aim: Configure OSPF routing protocol**.**

Topology:



Procedure and Observation:



For Router 1,
Router (config)# interface Serial 2/0
Router (config-if) # ip address 20.0.0.2 255.0.0.0
Router (config-if) # encapsulation PPP
Router (config-if) # no shut
Router (config-if)# exit
Router (config)# interface Serial 3/0
Router (config-if)# ip address 30.0.0.1 255.0.0.0
Router (config-if)# encapsulation ppp
Router (config-if)# clock rate 64000
Router (config-if)# no shut
Router (config-if)# exit.

For Router 2,
Router (config)# interface Serial 2/0
" # ip address 30.0.0.2 255.0.0.0
" # encapsulation ppp
" # noshut
" # exit

Next Step,
For Router 0,
Router (config)# router ospf 1
Router (config-router)# router-id 1.1.1.1
Router (config-router)# network 10.0.0.0 0.255.255.255 area 3
" # network 20.0.0.0 0.255.255.255 area 1
" # exit

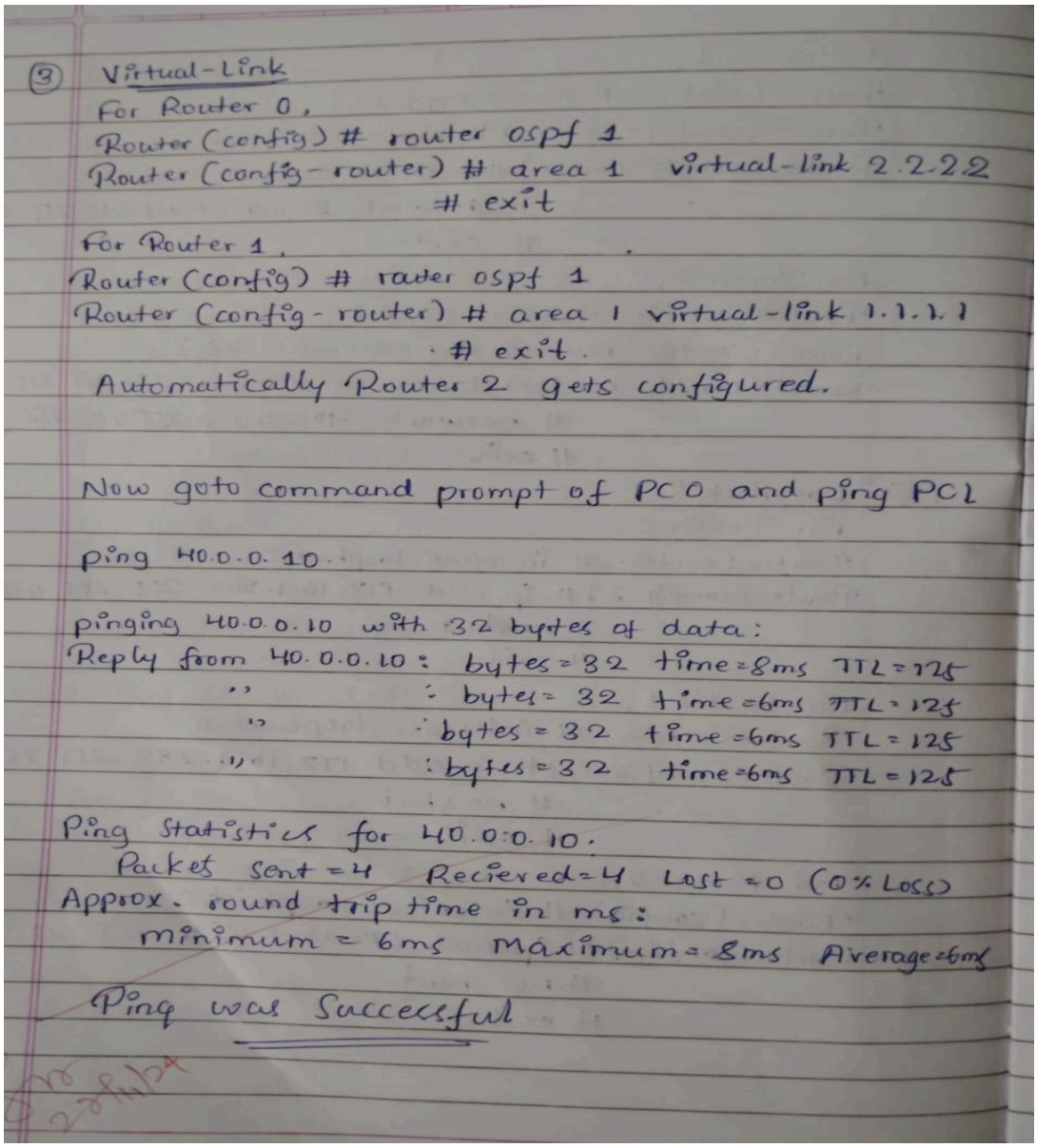
For Router 1,
Router (config) # router ospf 1
Router (config-router)# router-id 2.2.2.2
Router (config-router)# network 20.0.0.0 0.255.255.255 area 1
" # network 30.0.0.0 0.255.255.255 area 0
exit.

For Router 2,
Router (config) #router ospf 1
Router (config-router) # router-id 3.3.3.3
Router (config-router) # network 30.0.0.0 0.255.255.255 area 0
network 40.0.0.0 0.255.255.255 area 2
exit.

(2) "Loopback"
For Router 0,
Router (config) # interface loopback 0
Router (config-if) # ip add 172.16.1.252 255.255.0.0
noshut
exit.

For Router 1,
Router (config) # interface loopback 0
Router (config-if)# ip add 172.16.1.253 255.255.0.0
no shut
exit.

For Router 2,
Router (config) # interface loopback 0
Router (config-if)# ip add 172.16.1.254 255.255.0.0
no shut
exit.



③ Virtual-Link

For Router 0,

Router (config) # router ospf 1

Router (config-router) # area 1 virtual-link 2.2.2.2

exit

For Router 1,

Router (config) # rater ospf 1

Router (config-router) # area 1 virtual-link 1.1.1.1

exit.

Automatically Router 2 gets configured.

Now goto command prompt of PC 0 and ping PC1

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

" : bytes=32 time=6ms TTL=125

" : bytes=32 time=6ms TTL=125

" : bytes=32 time=6ms TTL=125

Ping statistics for 40.0.0.10.

Packets sent=4 Recieved=4 Lost=0 (0% Loss)

Approx. round trip time in ms:

minimum = 6ms Maximum = 8ms Average=6ms

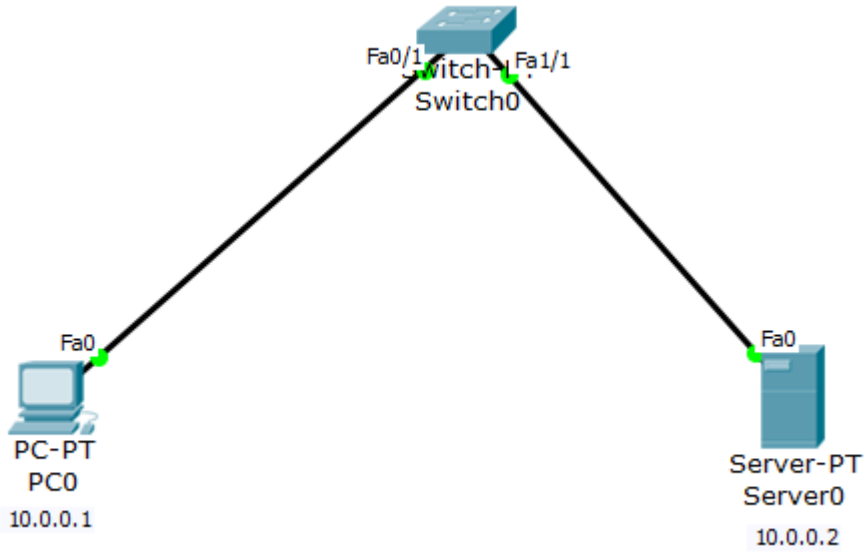
Ping was Successful

```
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
```

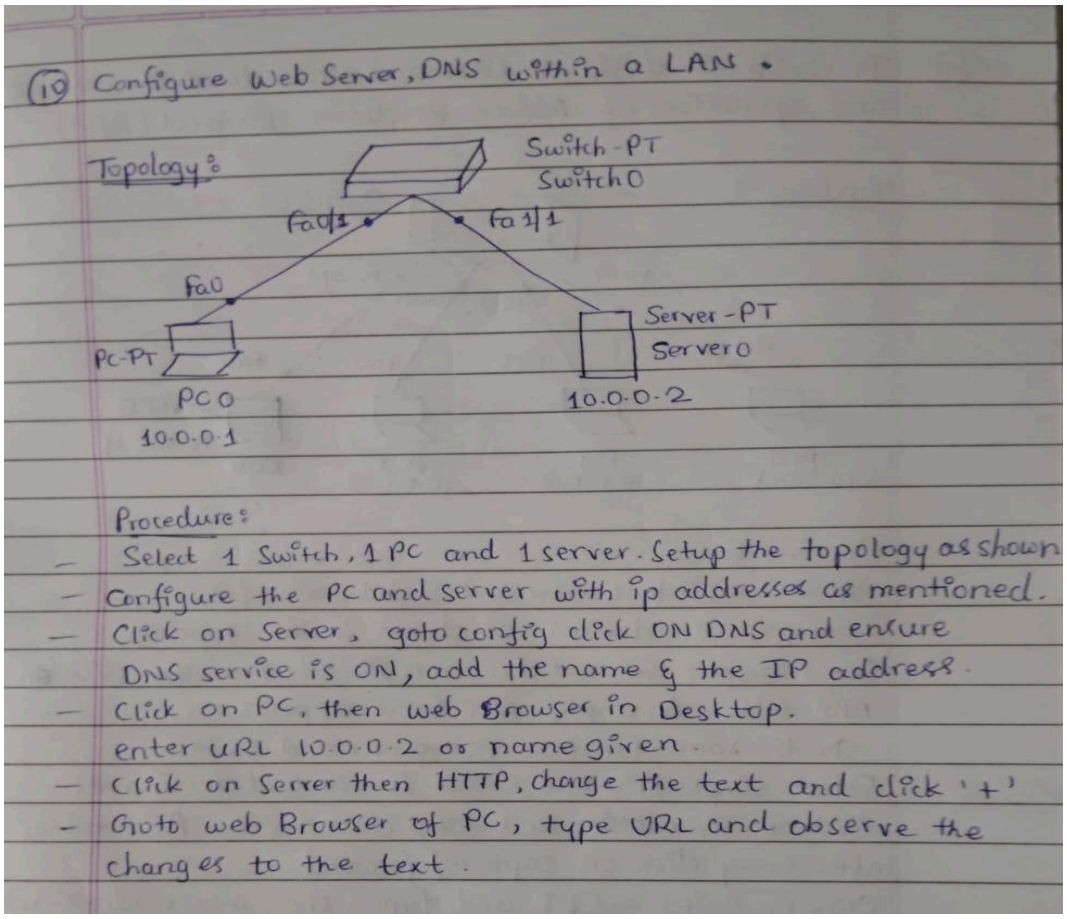
Program 8:

Aim: Configure Web Server, DNS within a LAN**.**

Topology:



Procedure and Observations:

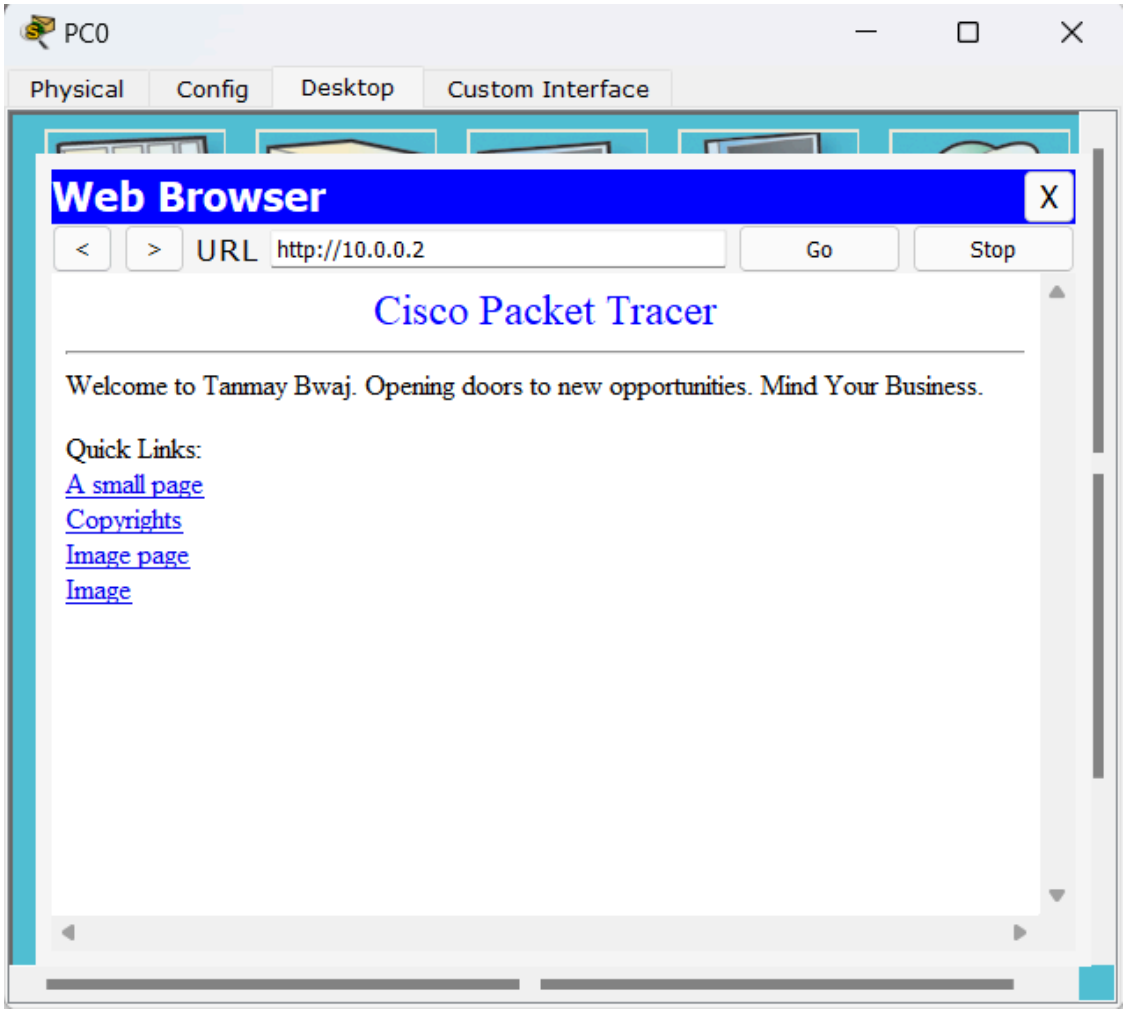


Observation:

DNS resolves domain names to IP addresses, allowing clients to access the web server 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 resource URL

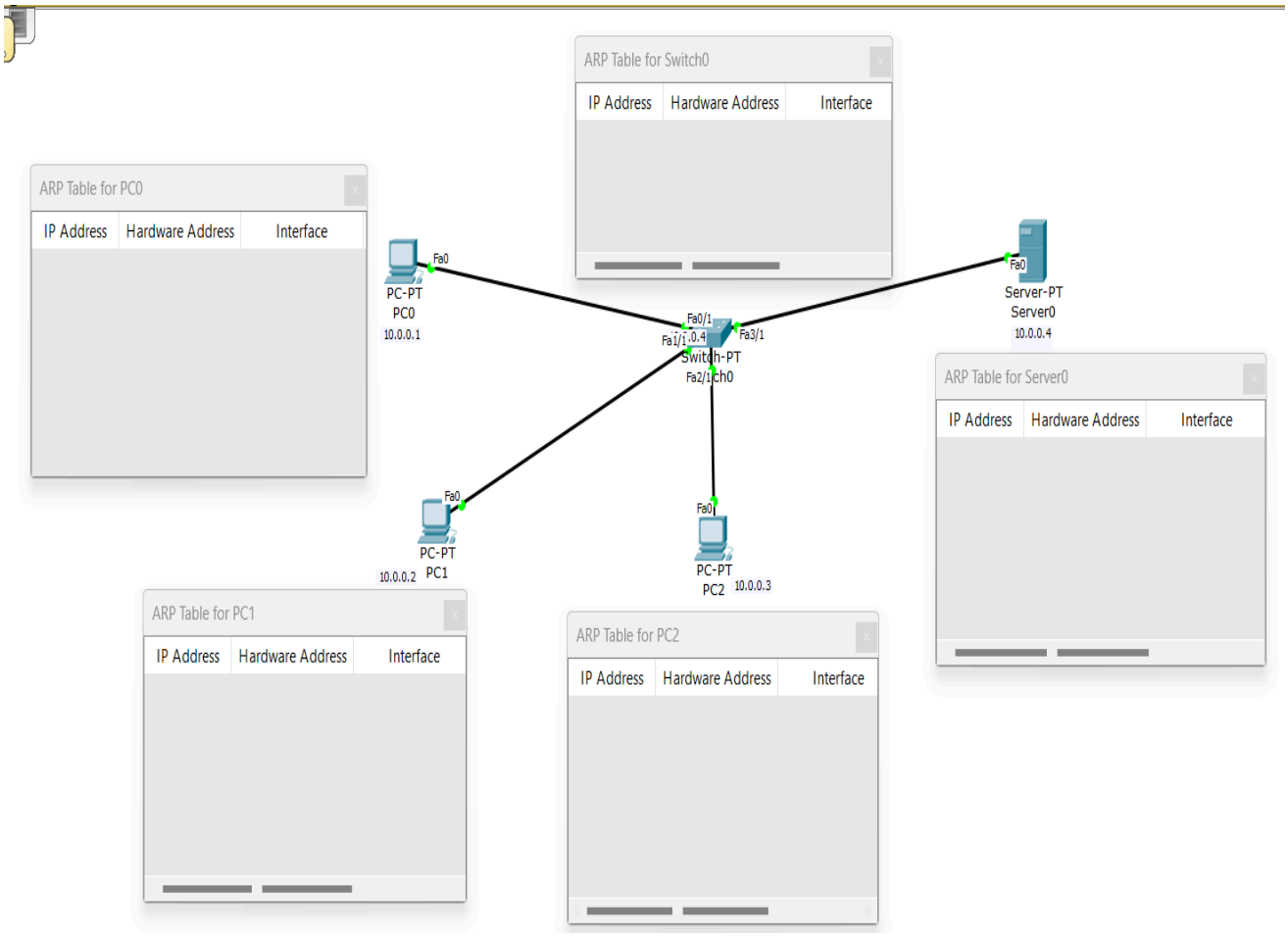
– The DNS server could hence be configured within a LAN by enabling the DNS.



Program 9:

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

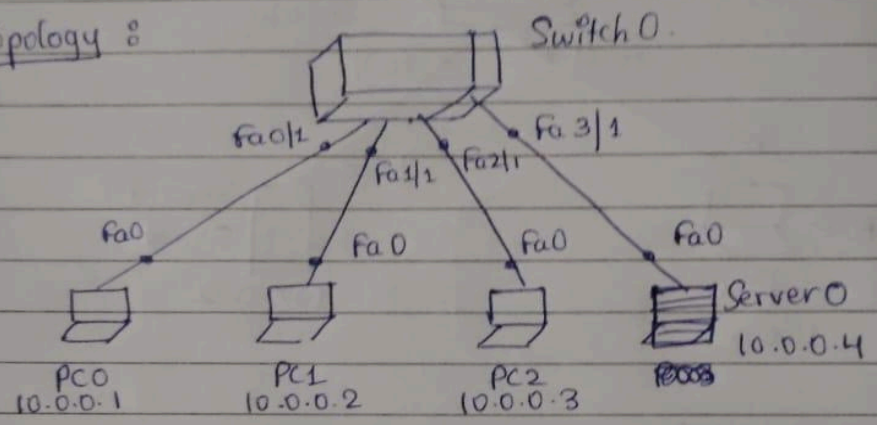
Topology:



Procedure and Observations:

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

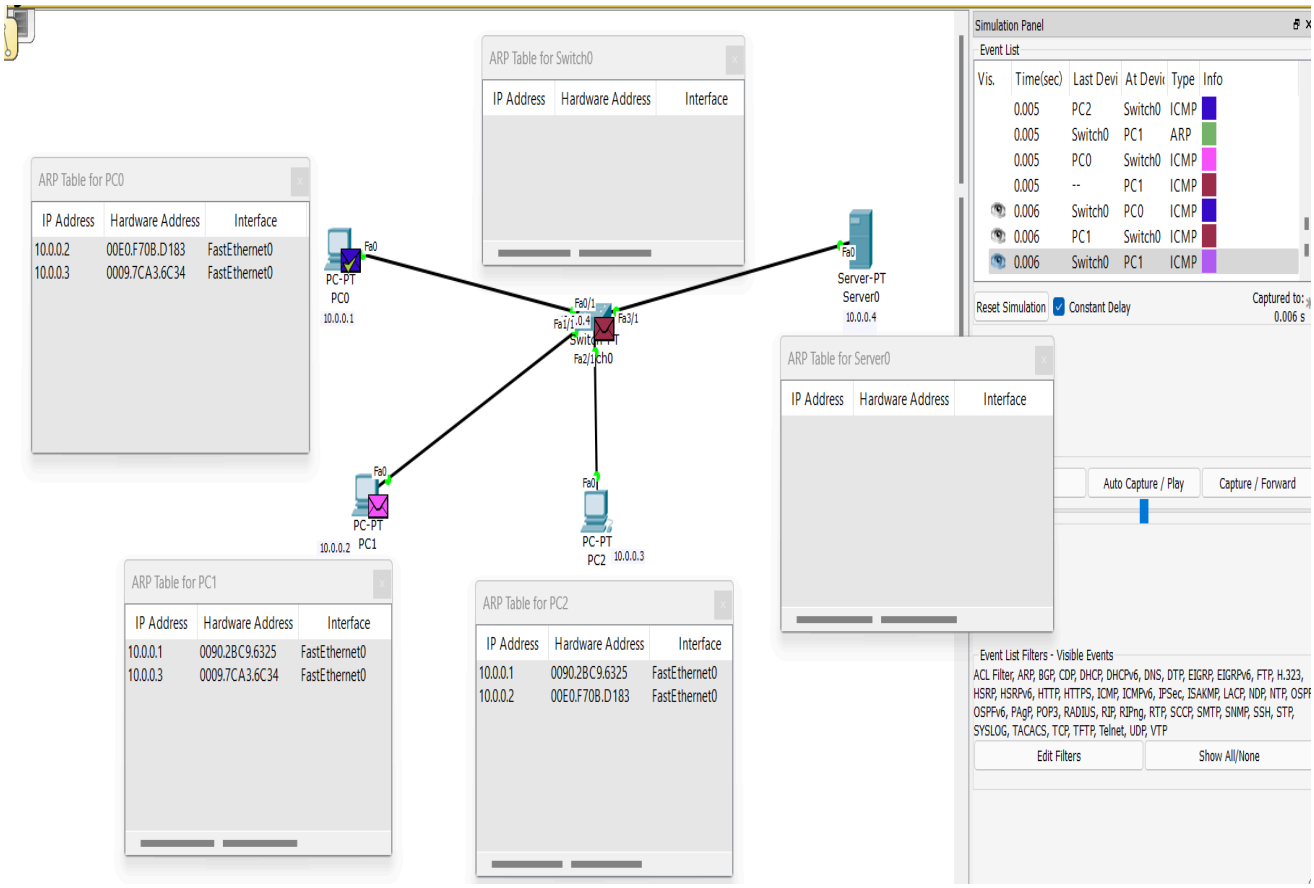
Topology :



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 PC's, server and switch.
Select simple PDU and choose source and destination.
Click on simulation and keep checking the ARP tables after every click on capture/forward.
Click on Switch – CLI and type : show mac address-table.

Observation : Initially ARP table of all devices observed is empty
After every click on capture / forward, in the ARP tables update as ARP requests and replies are exchanged showing the mapping between IP addresses & MAC addresses of devices involved in the communication.




```
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:

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 fastetherent 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, ensure ping 10.0.0.2 is successful.

Also 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

User Access verification

Password: psw.

Here the name PC changes to CSE which was the hostname.

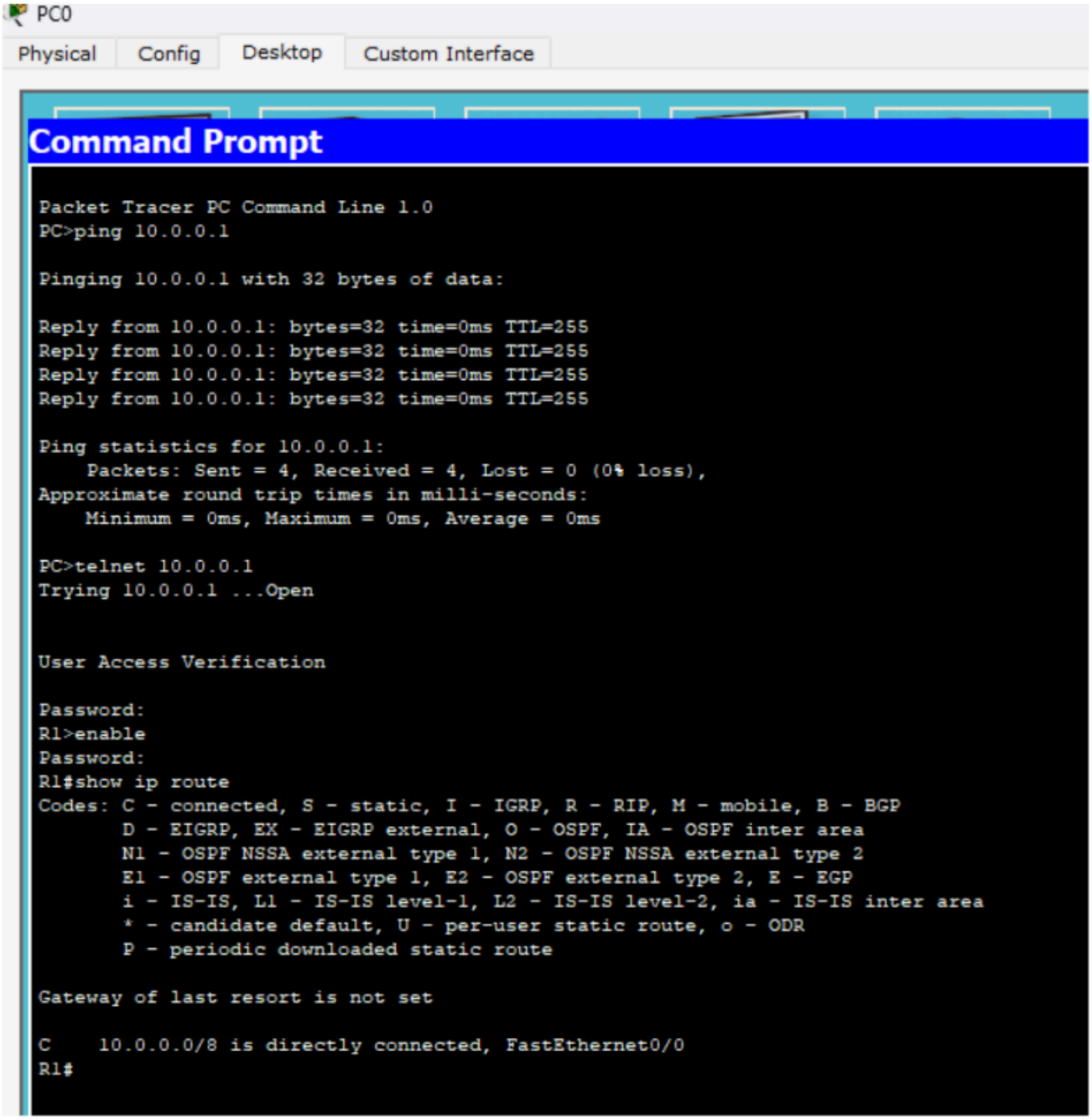
execute the commands:

enable

password: psw

Show ip route

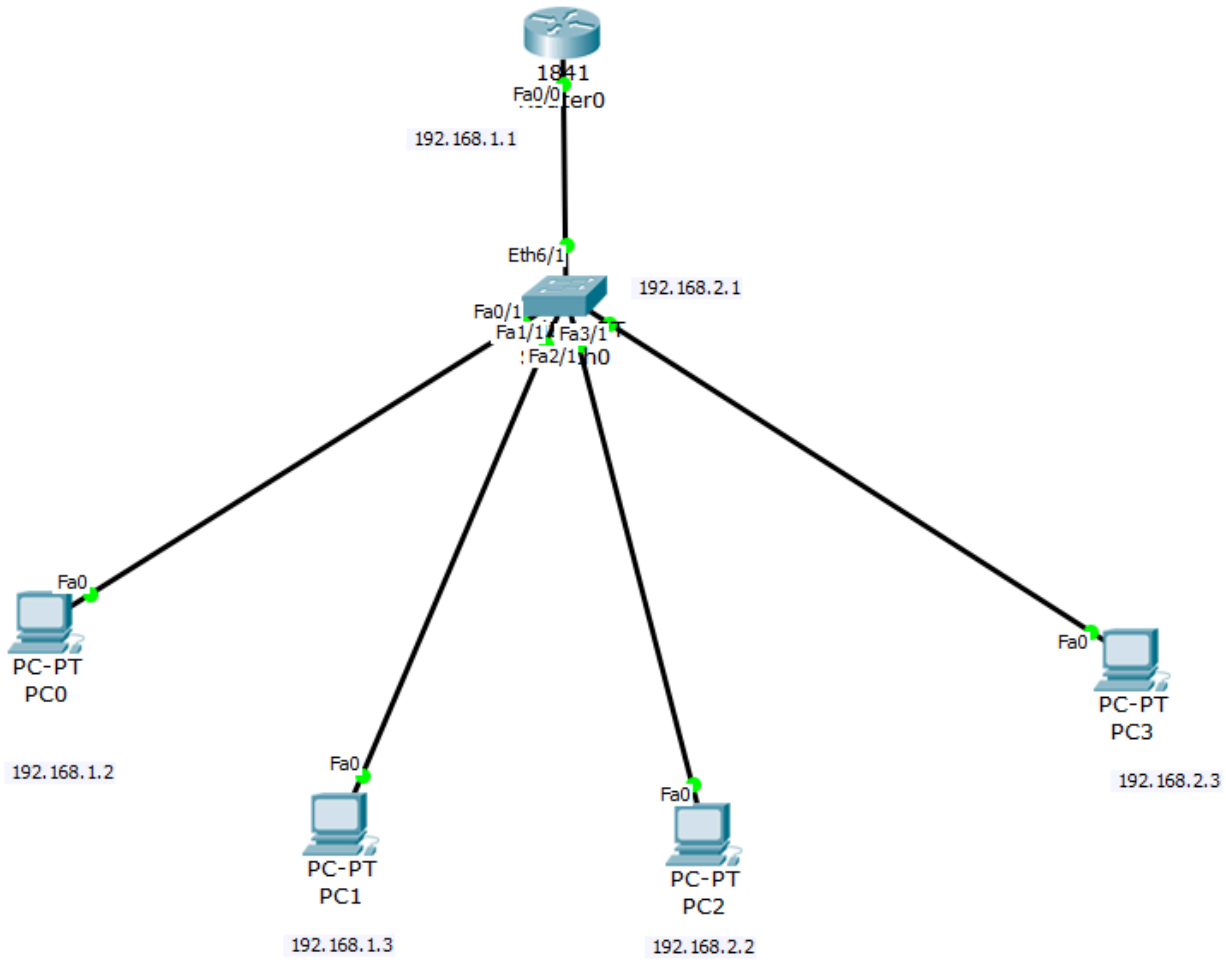
10.0.0.0/8 is directly connected, fastethernet 0/0.


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#

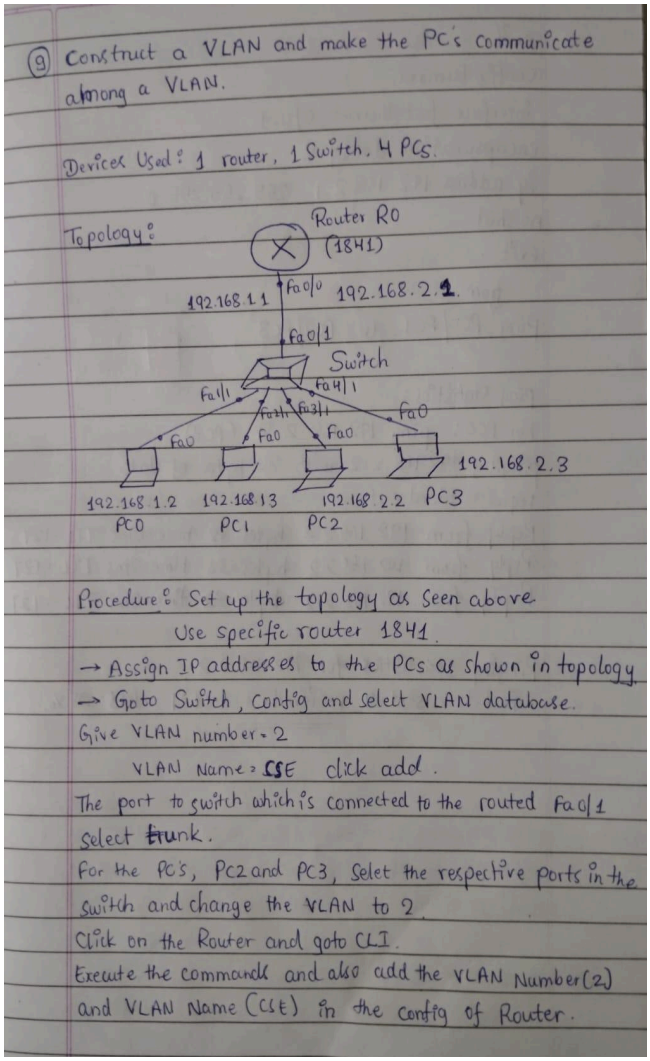
Program 11:

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

Topology:



Procedure and Observations:

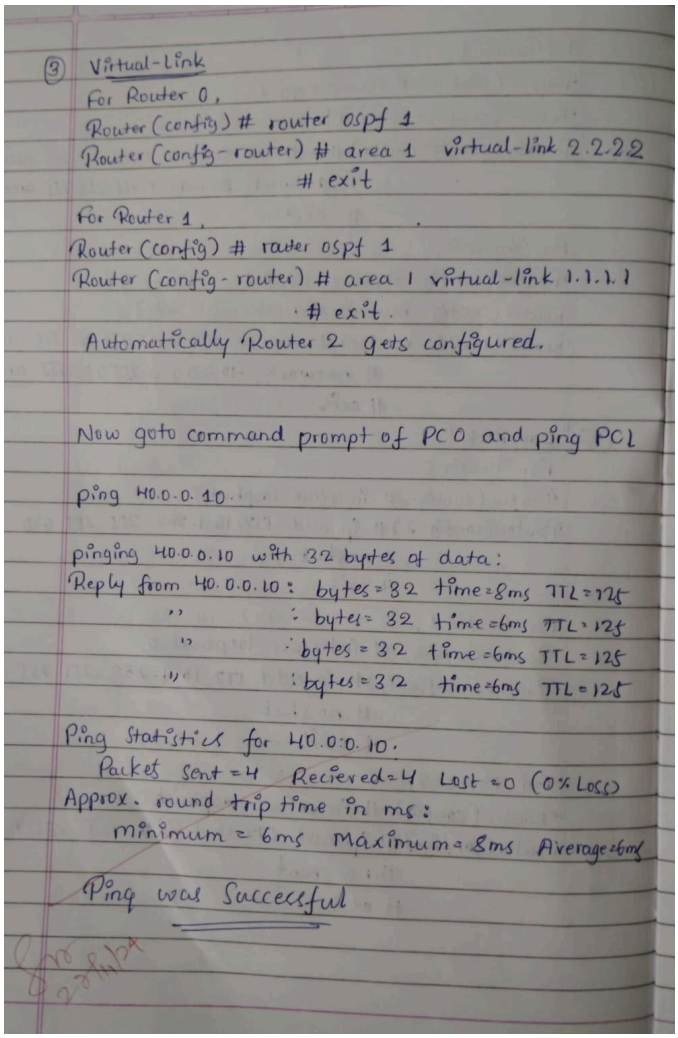


⑨ Construct a VLAN and make the PCs communicate among a VLAN.

Devices Used: 1 router, 1 Switch, 4 PCs.

Topology:

Procedure: Set up the topology as seen above
Use specific router 1841.
→ Assign IP addresses to the PCs as shown in topology
→ Goto Switch, config and select VLAN database.
Give VLAN number = 2
VLAN Name = CSE click add.
The port to switch which is connected to the routed Fa0/1 select trunk.
For the PCs, PC2 and PC3, Selet 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.



⑨ Virtual-Link
For Router 0,
Router (config) # router ospf 1
Router (config-router) # area 1 virtual-link 2.2.2.2
exit
For Router 1,
Router (config) # rater ospf 1
Router (config-router) # area 1 virtual-link 1.1.1.1
exit.
Automatically Router 2 gets configured.

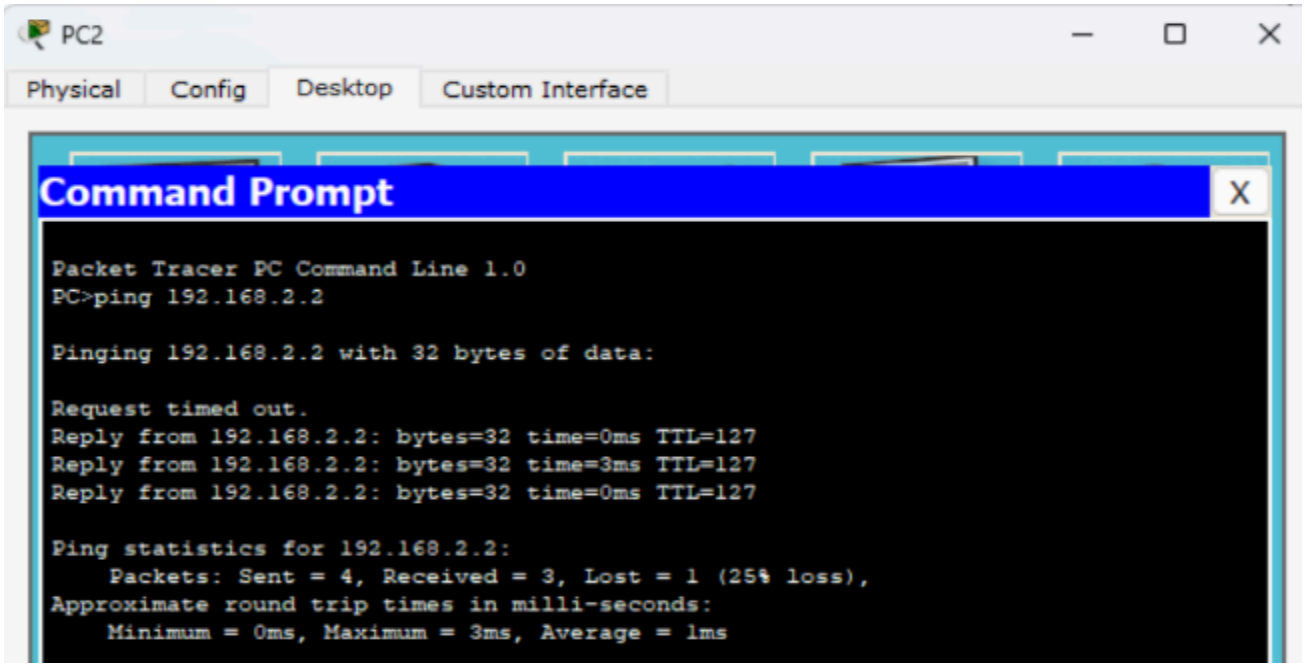
Now goto command prompt of PC 0 and ping PC1

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
" : bytes=32 time=6ms TTL=125
" : bytes=32 time=6ms TTL=125
" : bytes=32 time=6ms TTL=125

Ping statistics for 40.0.0.10.
Packet sent=4 Recieved=4 Lost=0 (0% Loss)
Approx. round trip time in ms:
minimum = 6ms Maximum = 8ms Average=6ms

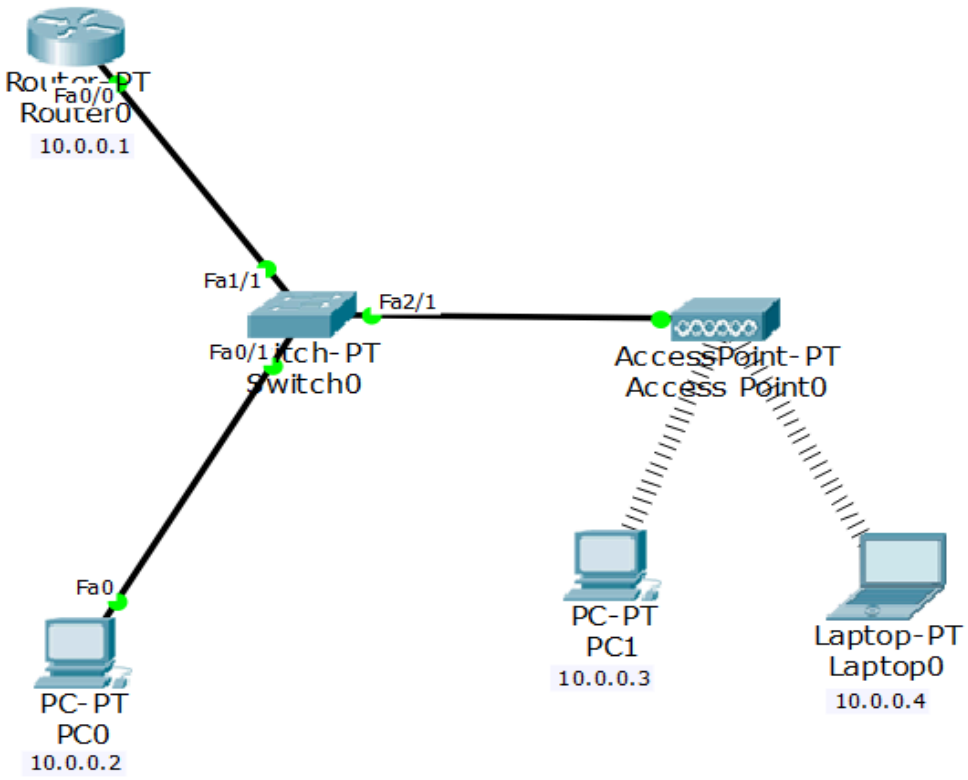
Ping was Successful



Program 12:

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

Topology:



Procedure and Observations:

Procedure:

Select 1 router, 2 PCs, 1 laptop, a switch and an access point

Set up the topology as shown. (initially no connection b/w end point & Accesspoint)

Configure the devices with their ip addresses as shown.

To configure access point, goto port 1.

give a name to SSID as CSE

Selet 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 unpanset listed in LHS.

Drag WMP300N wireless interface to the empty port.

Switch on PC.

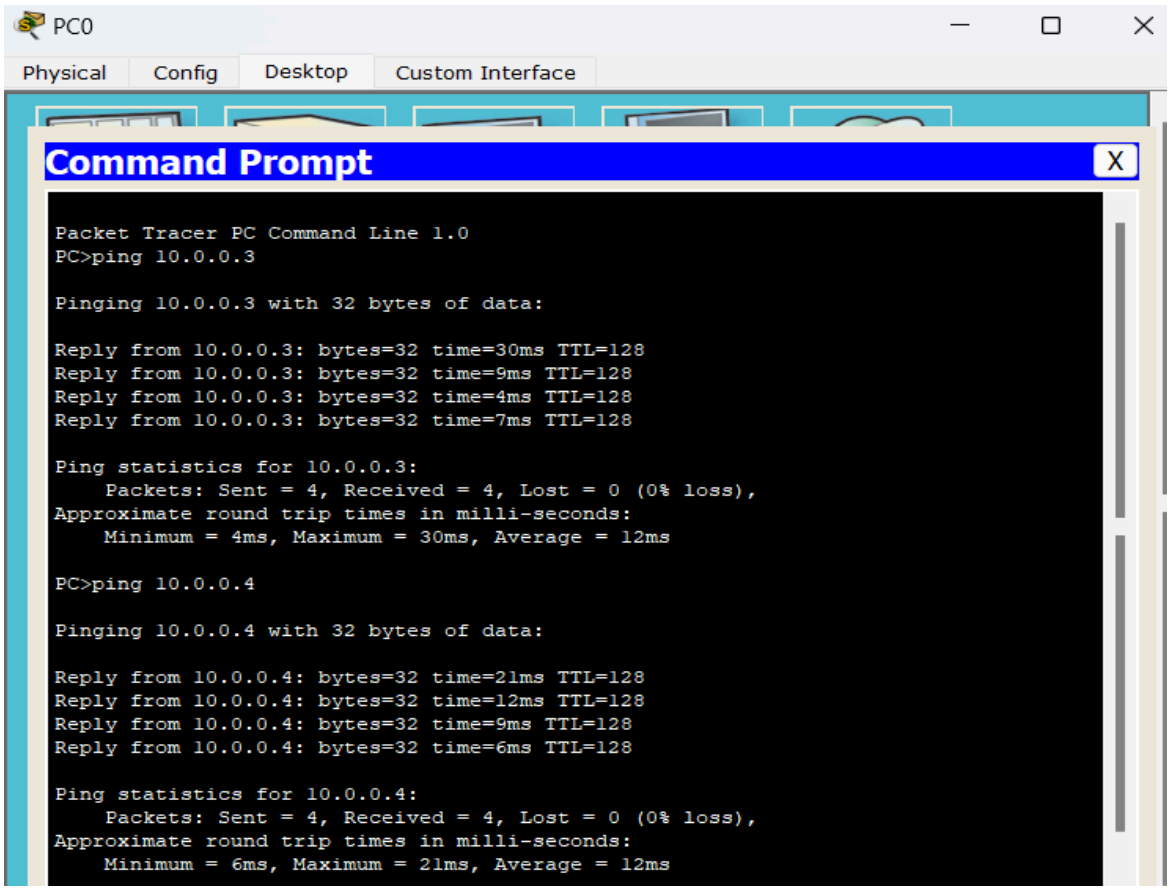
of AccessPoint0

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 SSIPname & channel authentication WEP & set key]

- Do similar process for Laptop.
- Ping from every device to all other devices and observe the result.
- In PC1, laptop0, turn system 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, PC1 & laptop0 wireless connections were observed with AccessPoint0 indicating Successful wireless connection.



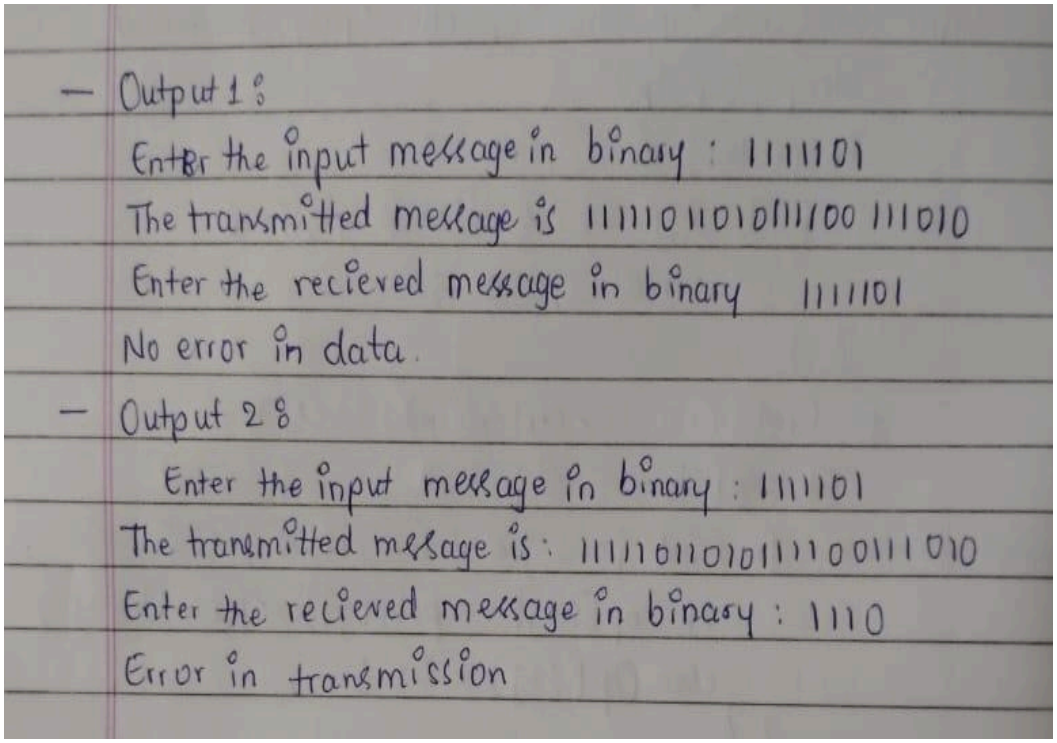
CYCLE - 2

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:



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!!

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.

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
```