VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

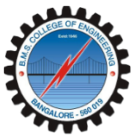
LAB REPORT
on

Computer Networks

Submitted by

Arvind Ashok (1BM21CS032)

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
JUN-2023 to SEP-2023

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Computer Networks Lab" carried out by **Arvind Ashok (1BM21CS032),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **Computer Networks - (22CS4PCCON)** work prescribed for the said degree.

Dr. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

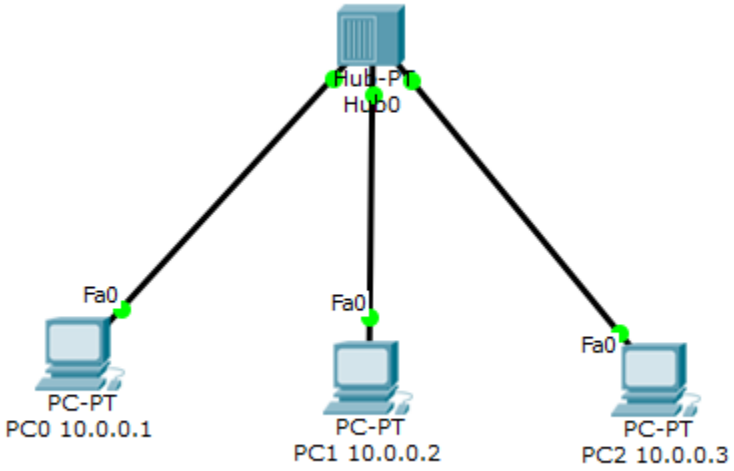
`

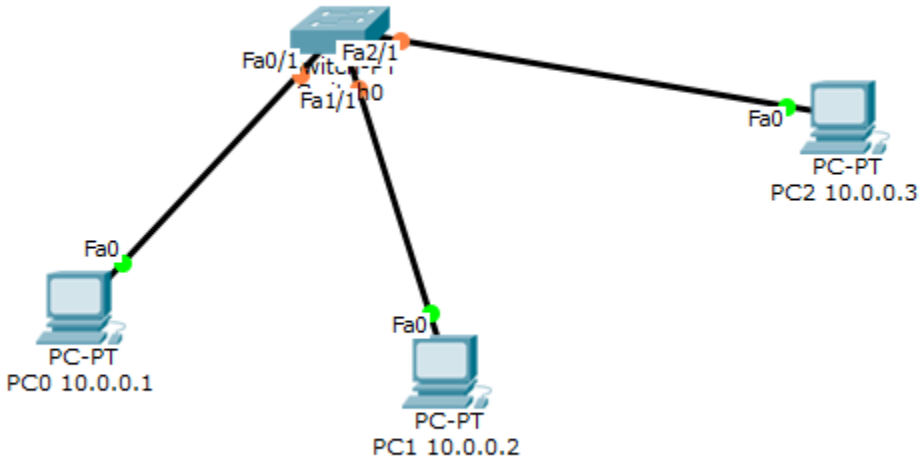
Index

Experiment 1

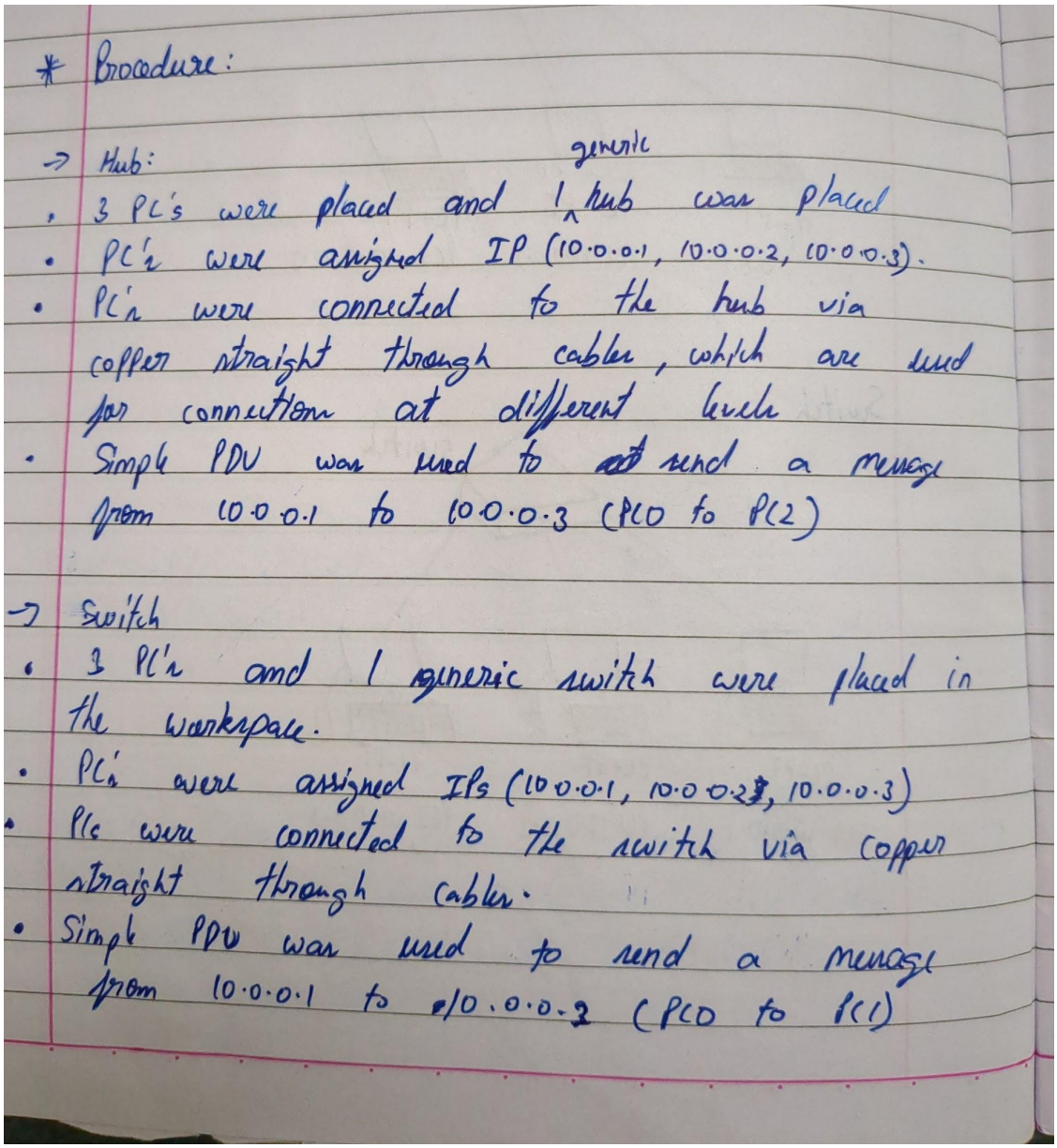
Aim of the program: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Topology:





Procedure:



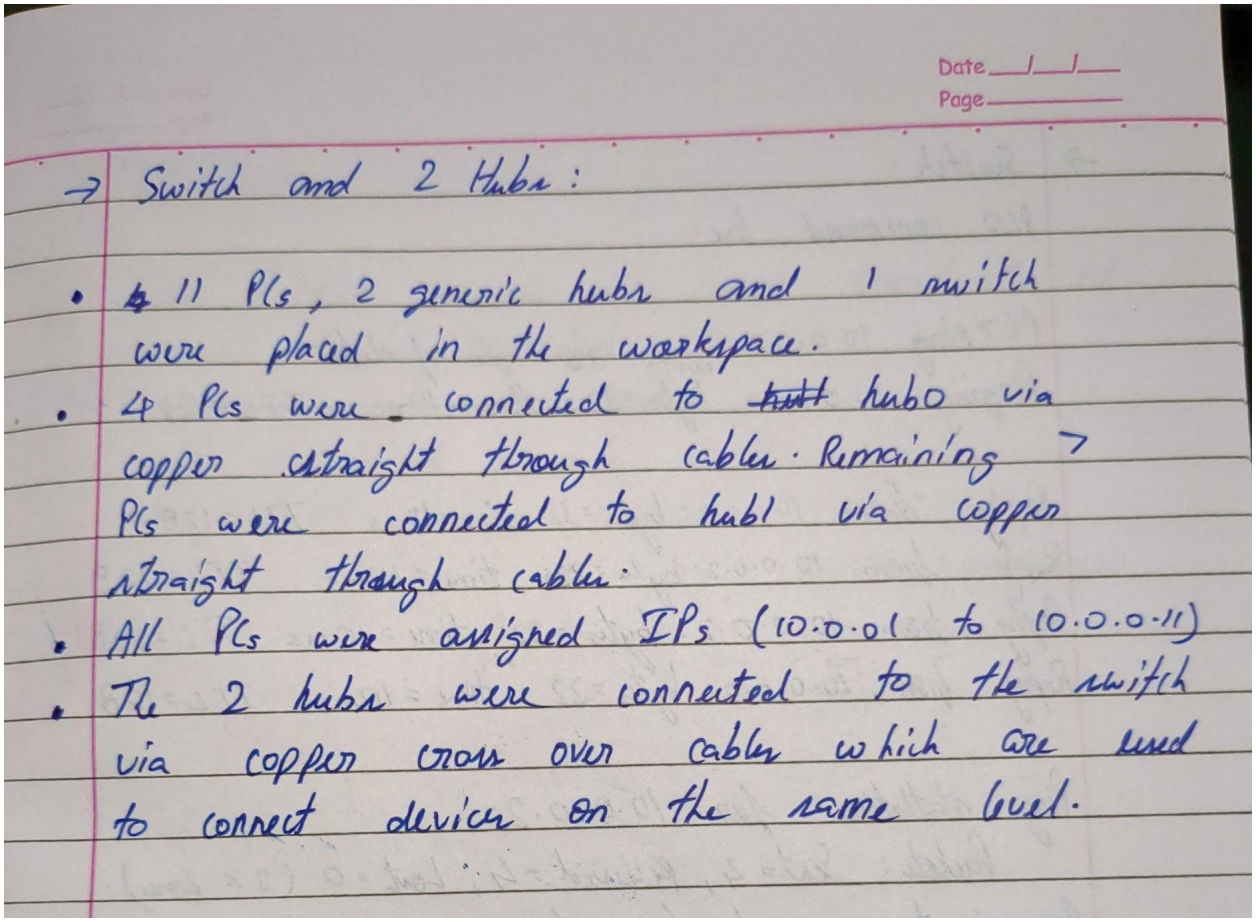
* Procedure:

→ Hub:

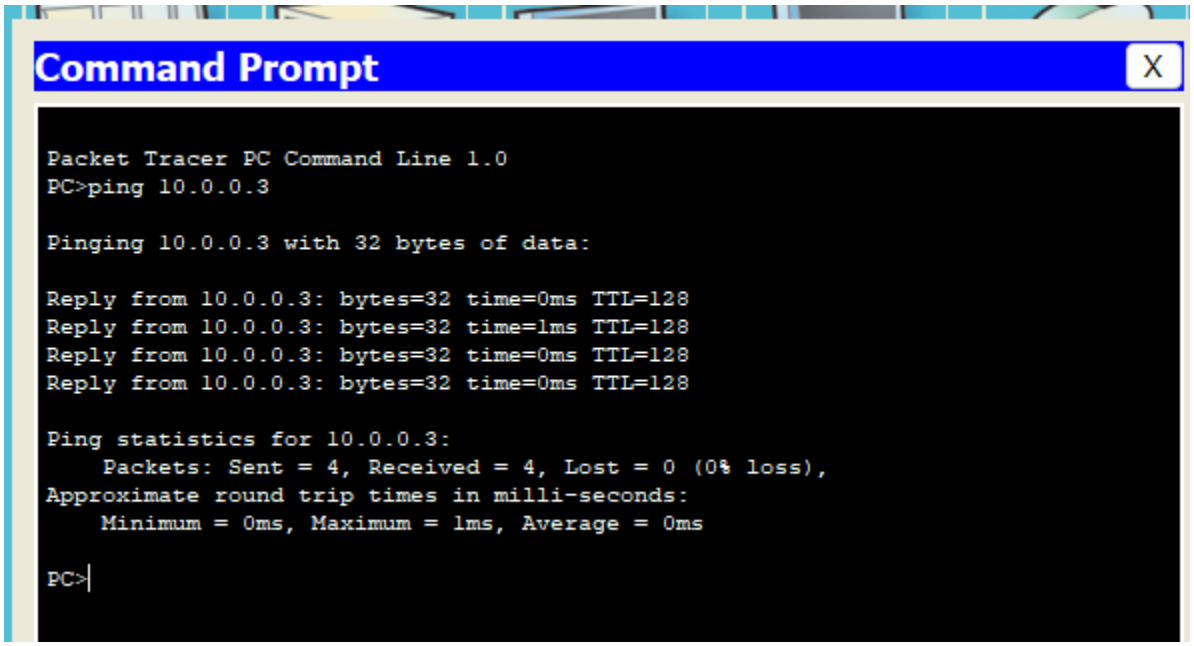
- 3 PC's were placed and 1 generic hub was placed
- PC's were assigned IP (10.0.0.1, 10.0.0.2, 10.0.0.3).
- PC's were connected to the hub via copper straight through cables, which are used for connections at different levels
- Simple PDU was used to send a message from 10.0.0.1 to 10.0.0.3 (PC0 to PC2)

→ Switch

- 3 PC's and 1 generic switch were placed in the workspace.
- PC's were assigned IPs (10.0.0.1, 10.0.0.2, 10.0.0.3)
- PCs were connected to the switch via copper straight through cables.
- Simple PDU was used to send a message from 10.0.0.1 to 10.0.0.3 (PC0 to PC1)

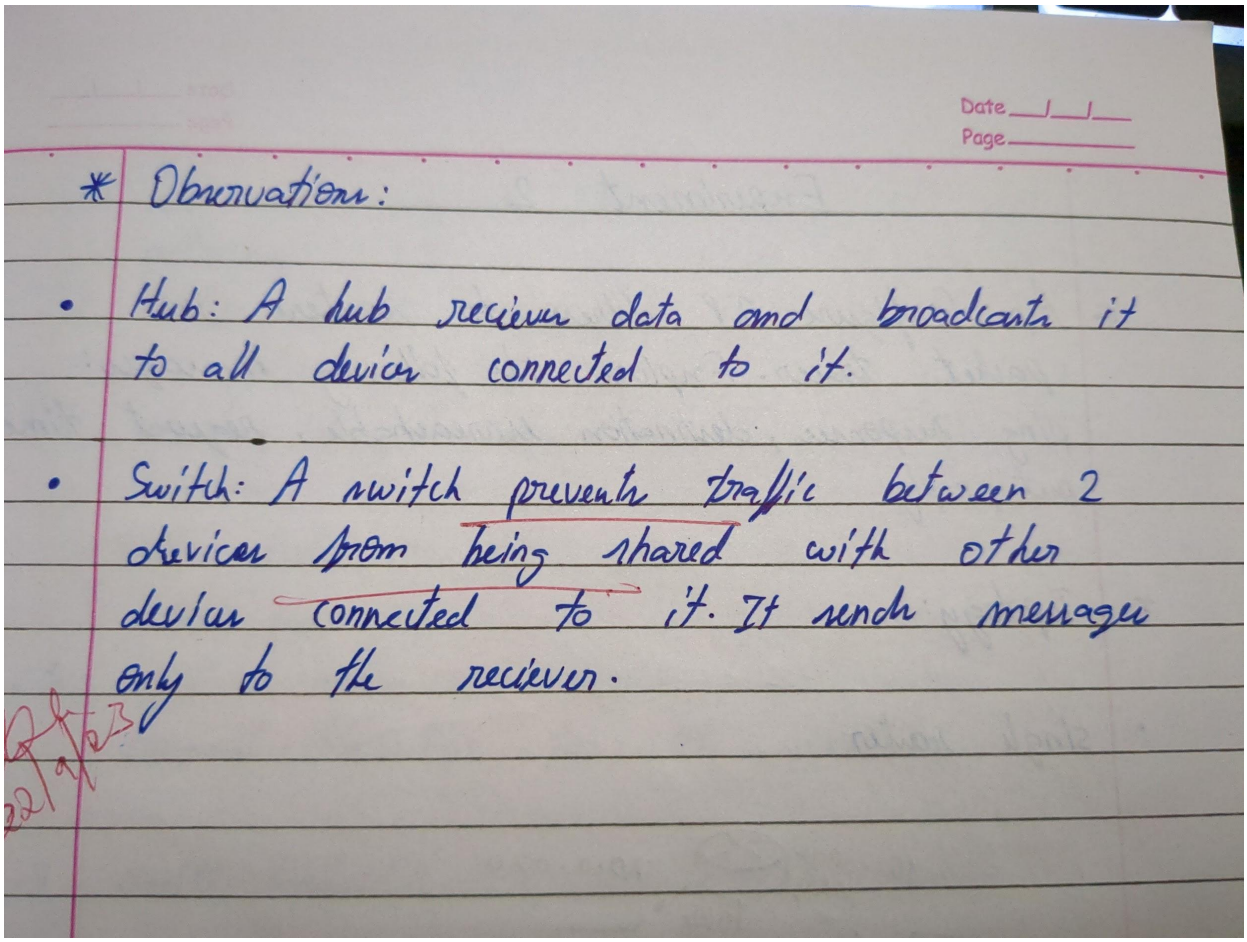

Date / /
Page
→ Switch and 2 Hubs:
• 11 PCs, 2 generic hubs and 1 switch were placed in the workspace.
• 4 PCs were connected to hub0 via copper straight through cables. Remaining 7 PCs were connected to hub1 via copper straight through cables.
• All PCs were assigned IPs (10.0.0.1 to 10.0.0.11)
• The 2 hubs were connected to the switch via copper cross over cables which are used to connect devices on the same level.

Output:


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=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>

```
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

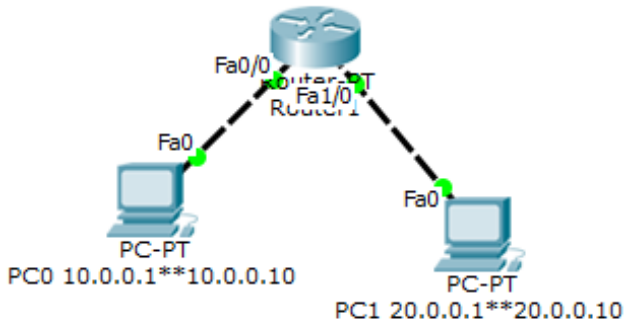
Observation:

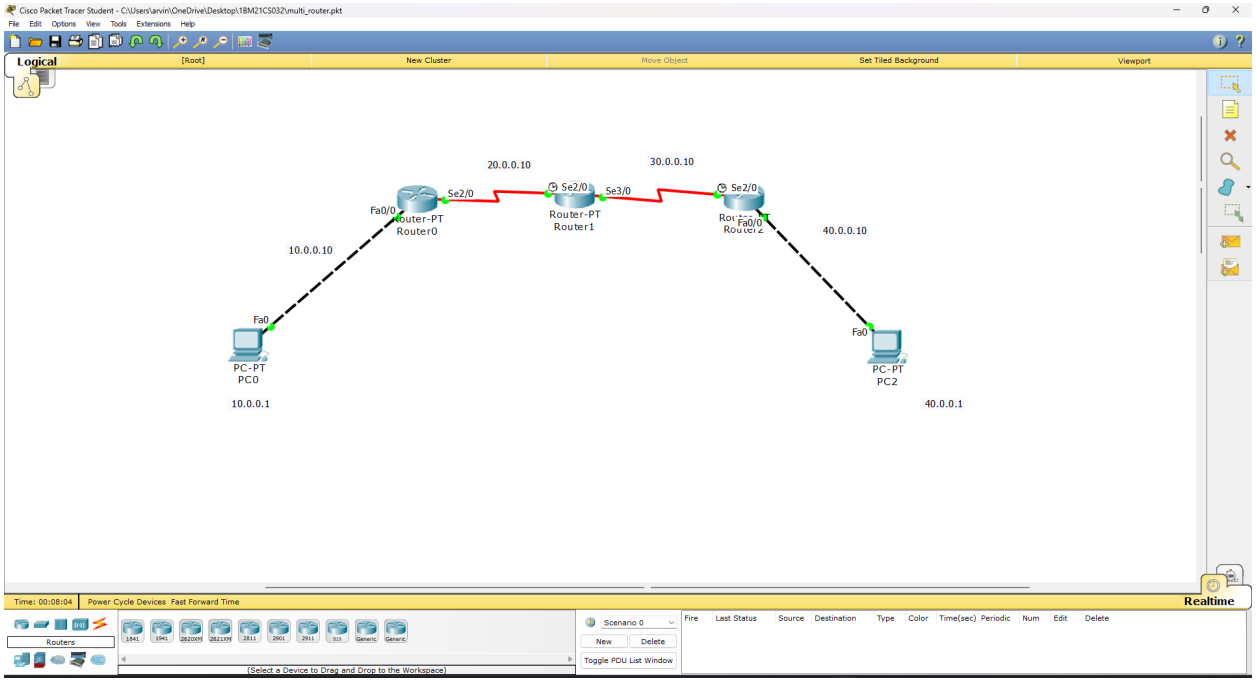


Experiment 2

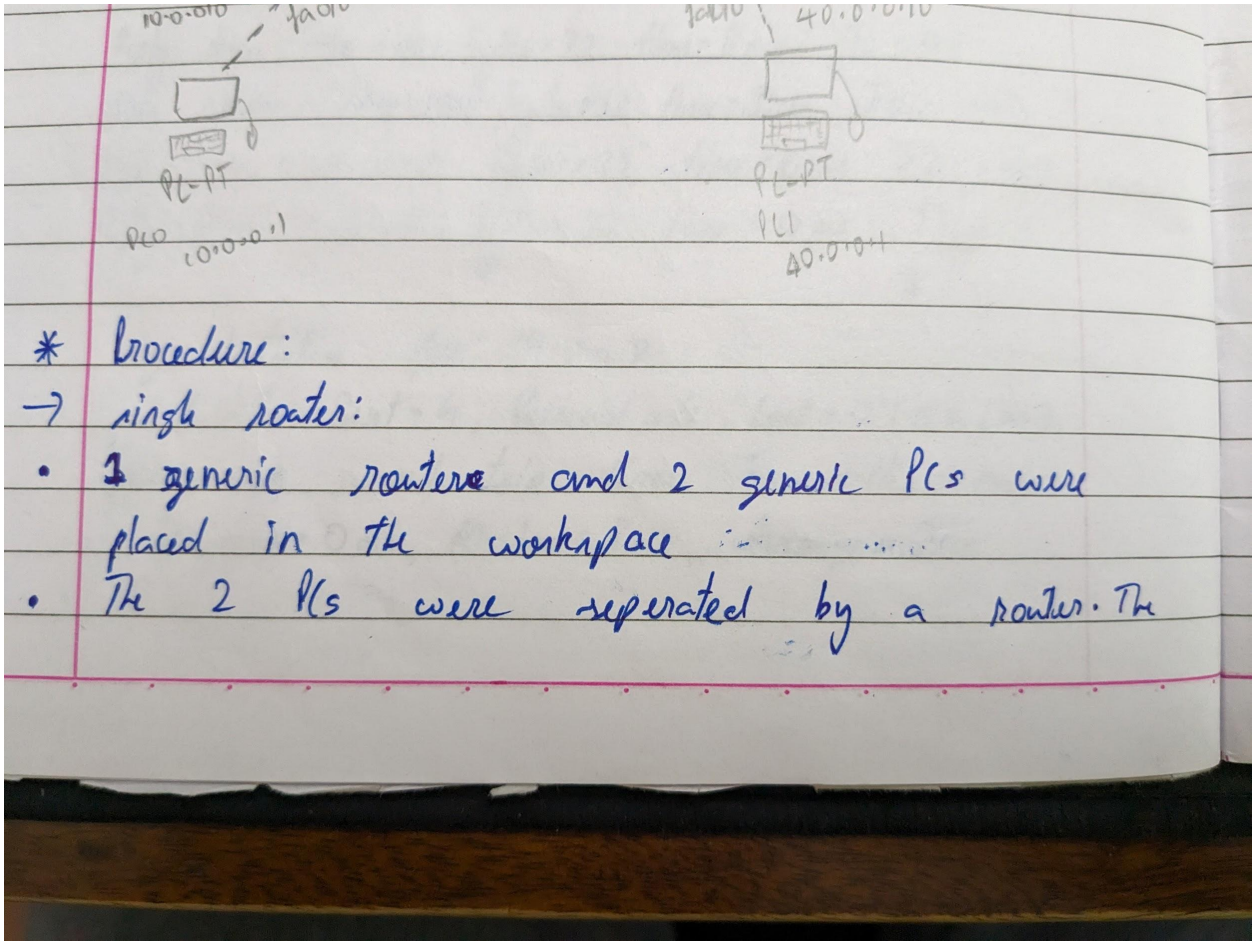
Aim of the program: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Topology:





Procedure:



* Procedure:

→ single router:

- 1 generic router and 2 generic PCs were placed in the workspace
- The 2 PCs were seperated by a router. The

Date __/__/__
Page _____

PC's were assigned IP 10.0.0.1, 20.0.0.1 and gateways 10.0.0.10 and 20.0.0.10 respectively.

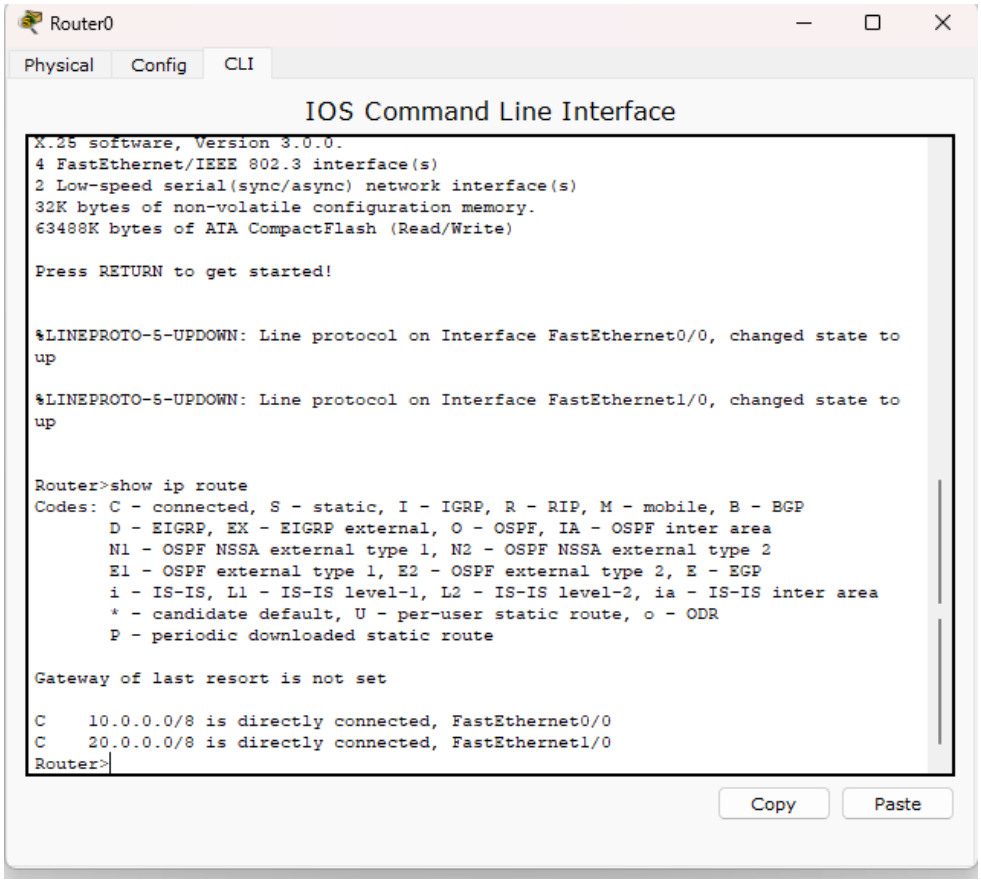
- Routers were configured from CLI using the following commands:
  enable, config t, interface fastethernet 0/0, IP address 10.0.0.10 255.0.0.0. The same was repeated for both ports. No shut was also used after each one.
- Copper cross over cables were used to connect the PCs to the routers.

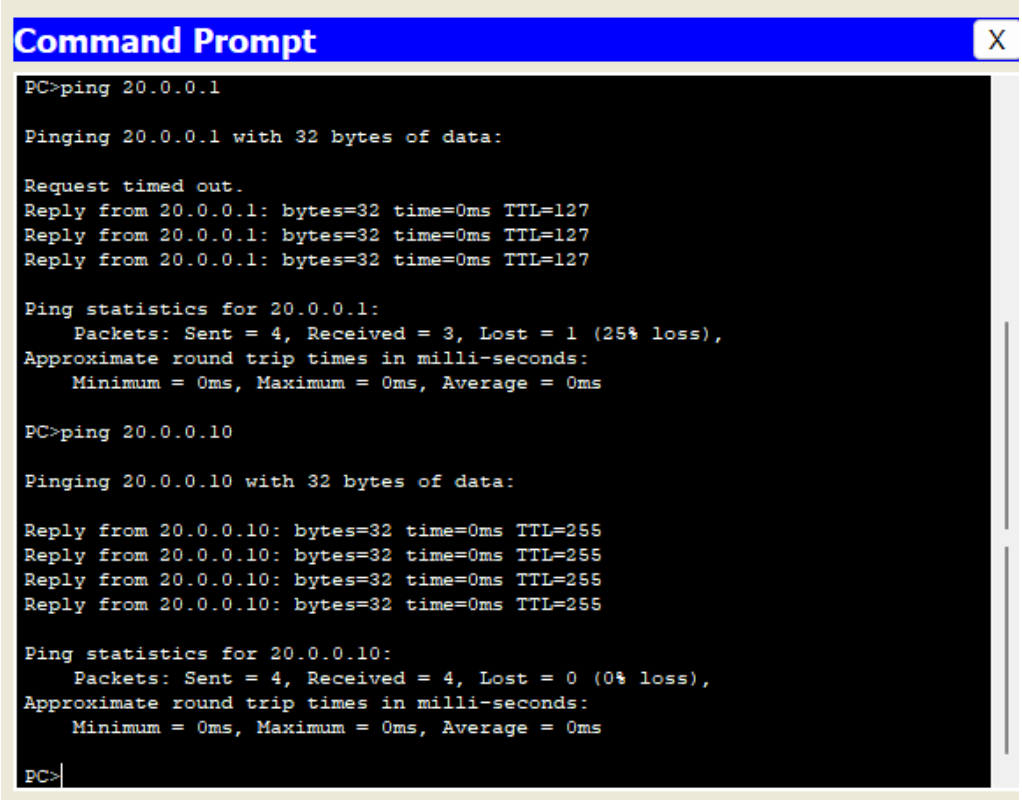
→ multi router:

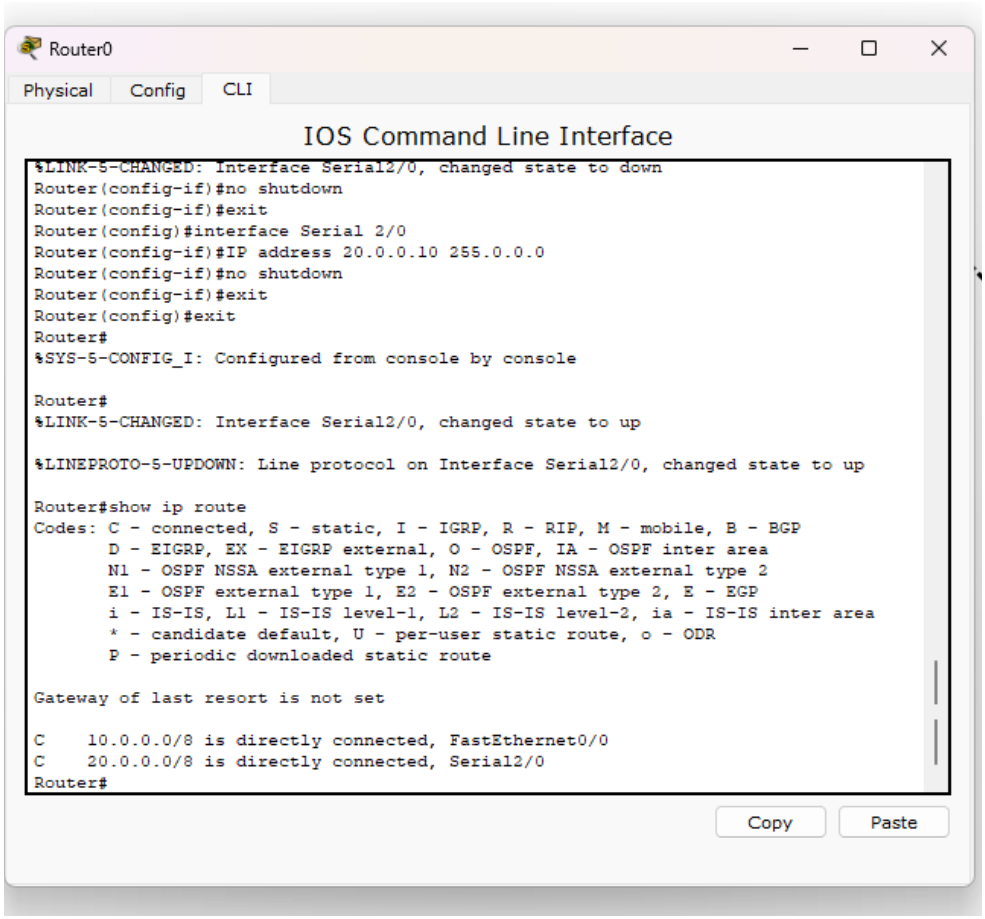
- 3 generic routers and 2 generic PC's were placed in the workspace. The 2 PC's were seperated by 3 routers.
- The PCs were assigned IP 10.0.0.1 and 40.0.0.1 and gateways 10.0.0.10 and 40.0.0.10 respectively.
- Routers were confired for fastethernet as done before. The routers were then configured for serial using: enable, config t, interface serial 2/0, IP address 20.0.0.10 255.00.0, no shut. The same was done for 30.0.0.10 and 40.0.0.10. &
- The routers were then connected using Serial DTE cables to the serial ports 2/0, 3/0.

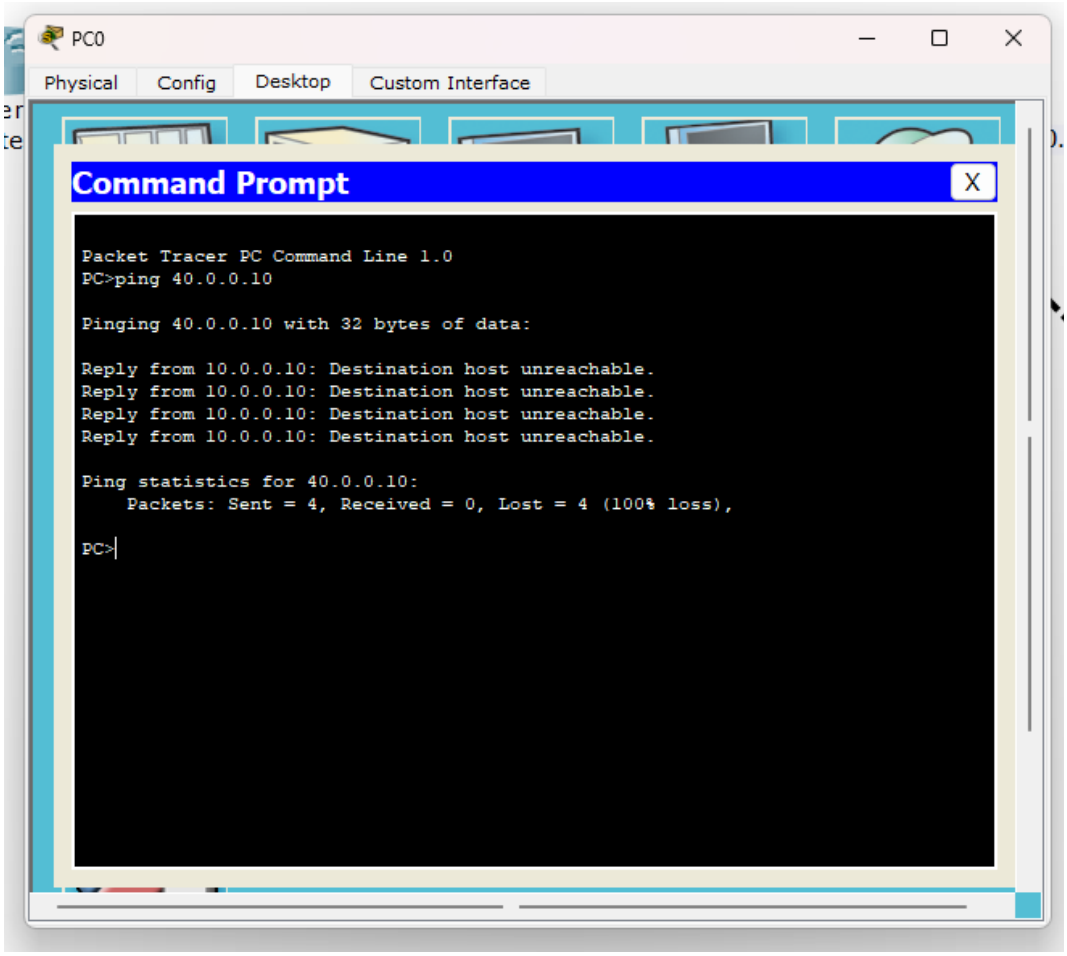
* Result:

Output:


Router0
Physical
Config
CLI
IOS Command Line Interface
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)
Press RETURN to get started!
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up
Router>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
Gateway of last resort is not set
C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 20.0.0.0/8 is directly connected, FastEthernet1/0
Router>
Copy
Paste


Command Prompt
X
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, 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=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
Reply from 20.0.0.10: bytes=32 time=0ms TTL=255
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 = 0ms, Average = 0ms
PC>


Router0
Physical
Config
CLI
IOS Command Line Interface
%LINK-5-CHANGED: Interface Serial2/0, changed state to down
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface Serial 2/0
Router(config-if)#IP address 20.0.0.10 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
Gateway of last resort is not set
C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 20.0.0.0/8 is directly connected, Serial2/0
Router#
Copy
Paste



Observation:

Date __/__/__
Page _____

* Observations:

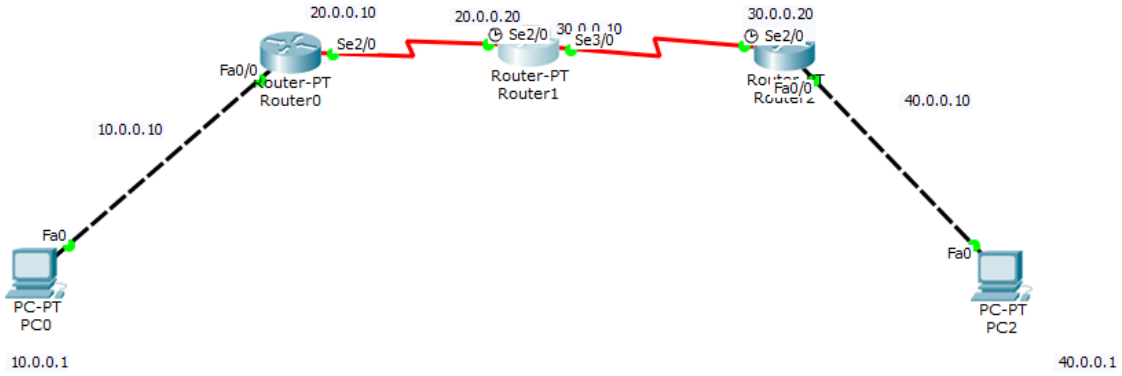
- The router connects LAN to the internet. It helps in communication between multiple network - In this case (10.0.0.10, 20.0.0.10, 30.0.0.10, 40.0.0.10)
- It determines the best path from the available paths.
- Packets are forwarded to the next network based on the information in routing table
- Request timed out is a response given when pinging a network and no response is recieved. It implies that the ICMP packet reached a host, but the reply could not reach the requesting host. This could be due to packet loss
- Destination host unreachable is when the host we try to ping is down. This is usually due to lack of router from source to destination.

26/6/23

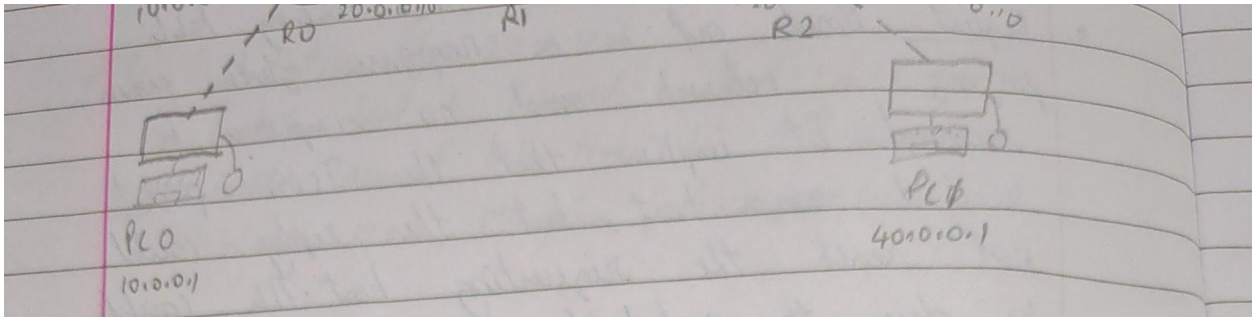
Experiment 3

Aim of the program: Configure default route, static route to the Router

Topology:



Procedure:

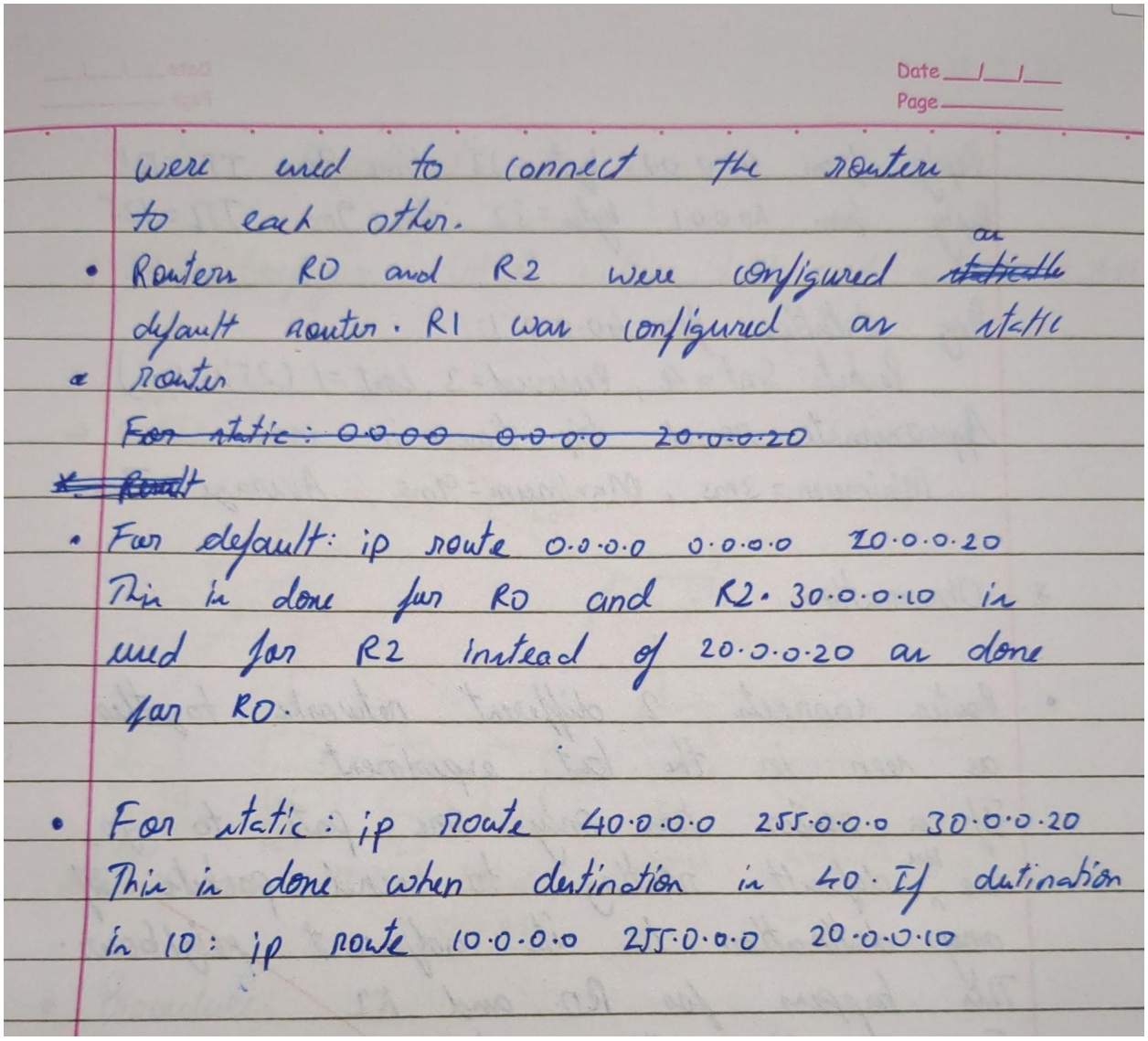


* Procedure:

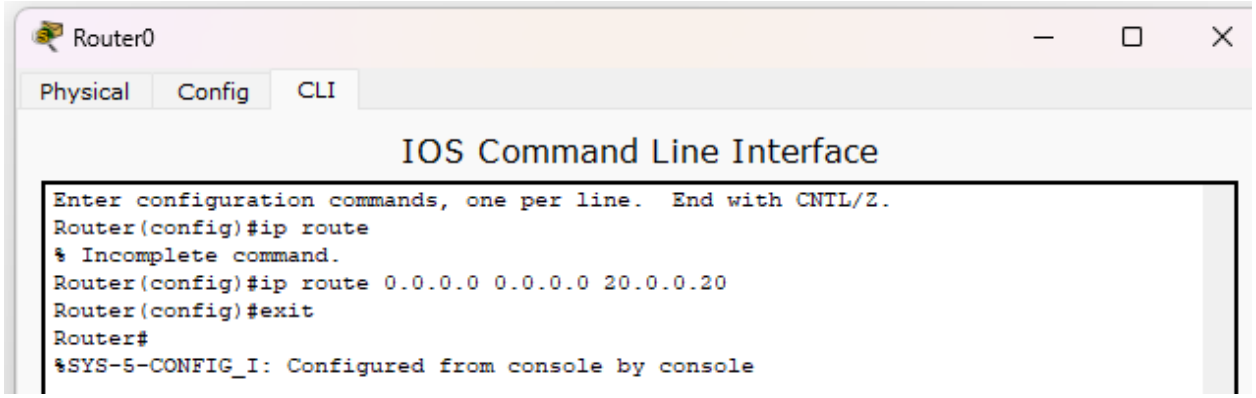
- 3 generic routers and 2 generic PCs are placed in the workspace. The PCs are separated by the 3 routers.
- The PCs are assigned IPs 10.0.0.1 and 40.0.0.1 and gateways 10.0.0.10 and 40.0.0.10 respectively.
- Routers were configured from the CLI using:

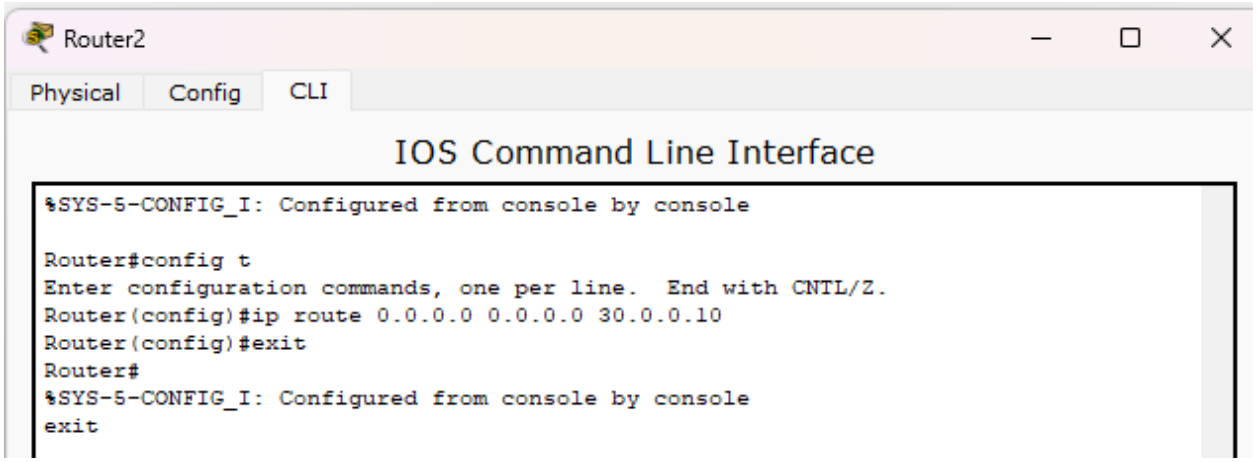
enable
config t
interface ~~fastethernet~~ <port>
IP address <ip> <subnet mask>
no shut
exit
exit. This is repeated for each port.

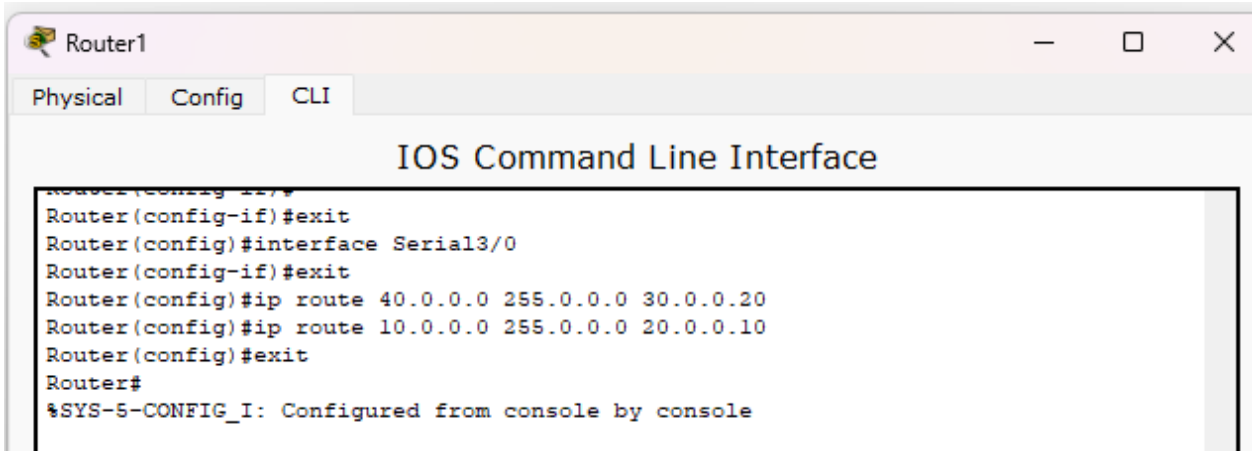
- Copper cross over cables were used to connect the PC to router. Serial DTE cable


were used to connect the routers
to each other.
• Routers R0 and R2 were configured as default router. R1 was configured as static router
For static: 0.0.0.0 0.0.0.0 20.0.0.20
• For default: ip route 0.0.0.0 0.0.0.0 20.0.0.20
This is done for R0 and R2. 30.0.0.10 is used for R2 instead of 20.0.0.20 as done for R0.
• For static: ip route 40.0.0.0 255.0.0.0 30.0.0.20
This is done when destination is 40. If destination is 10: ip route 10.0.0.0 255.0.0.0 20.0.0.10

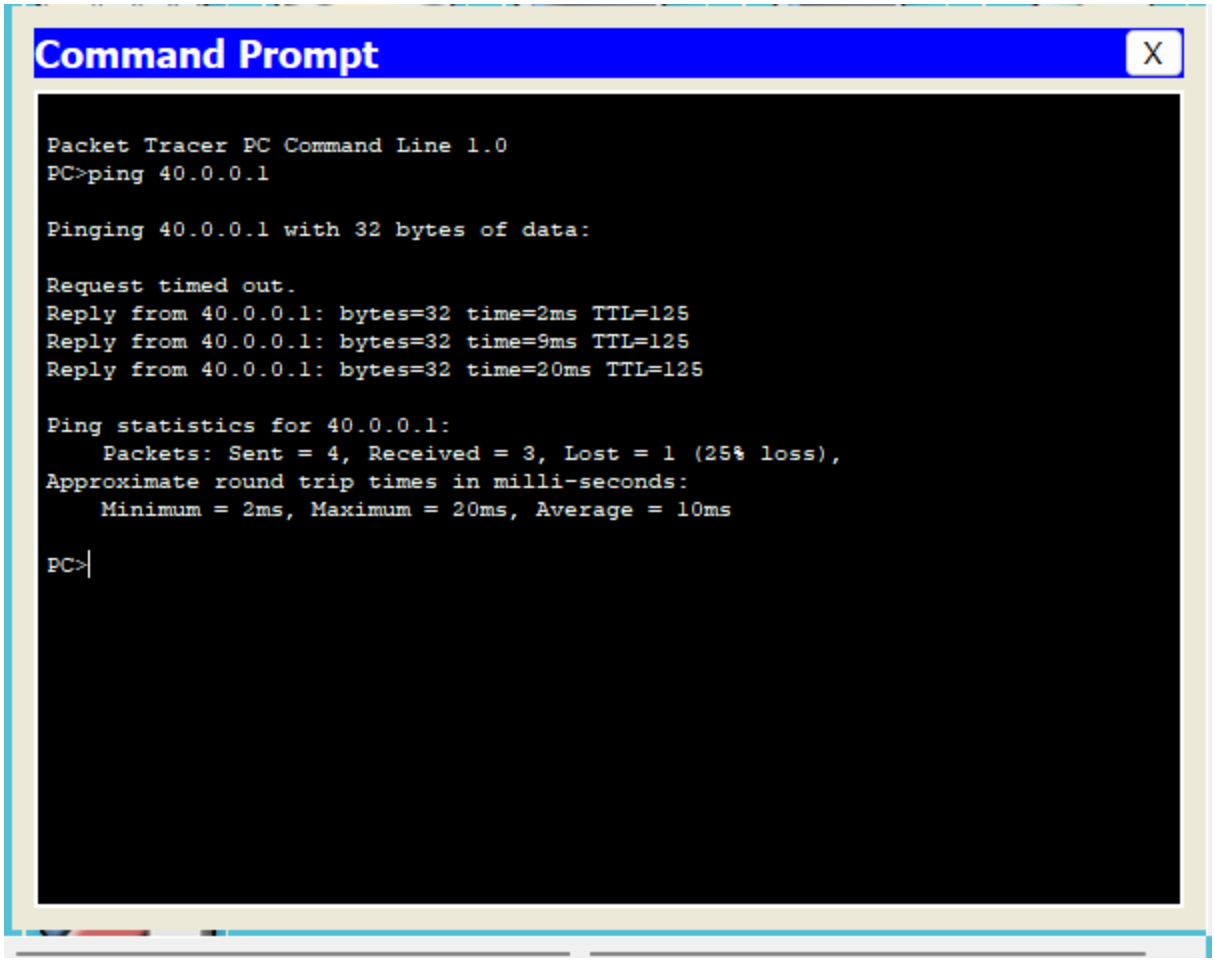
Output:


Router0
Physical
Config
CLI
IOS Command Line Interface
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route
% Incomplete command.
Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.20
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

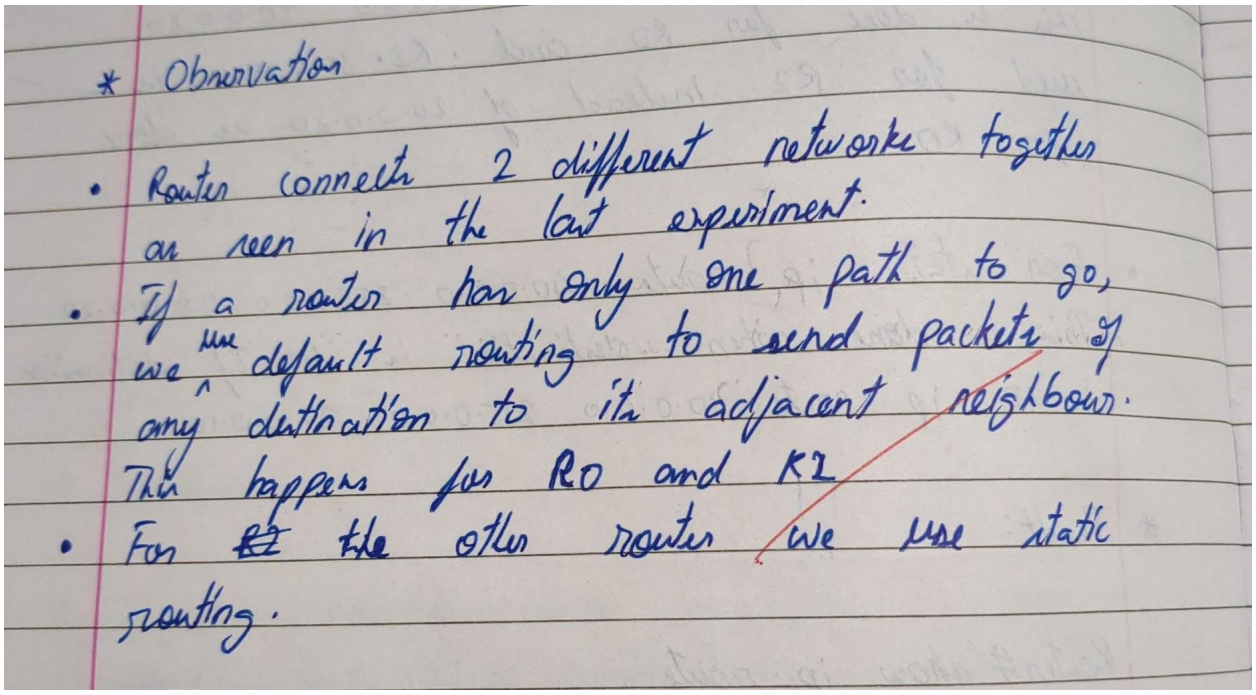





```
Router>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 20.0.0.10
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.20
Router>
```


Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=9ms TTL=125
Reply from 40.0.0.1: bytes=32 time=20ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 20ms, Average = 10ms
PC>

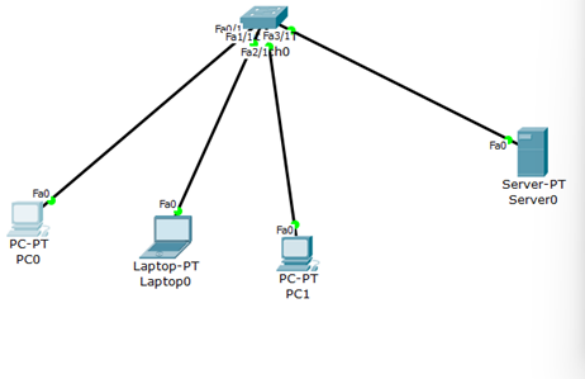
Observation:



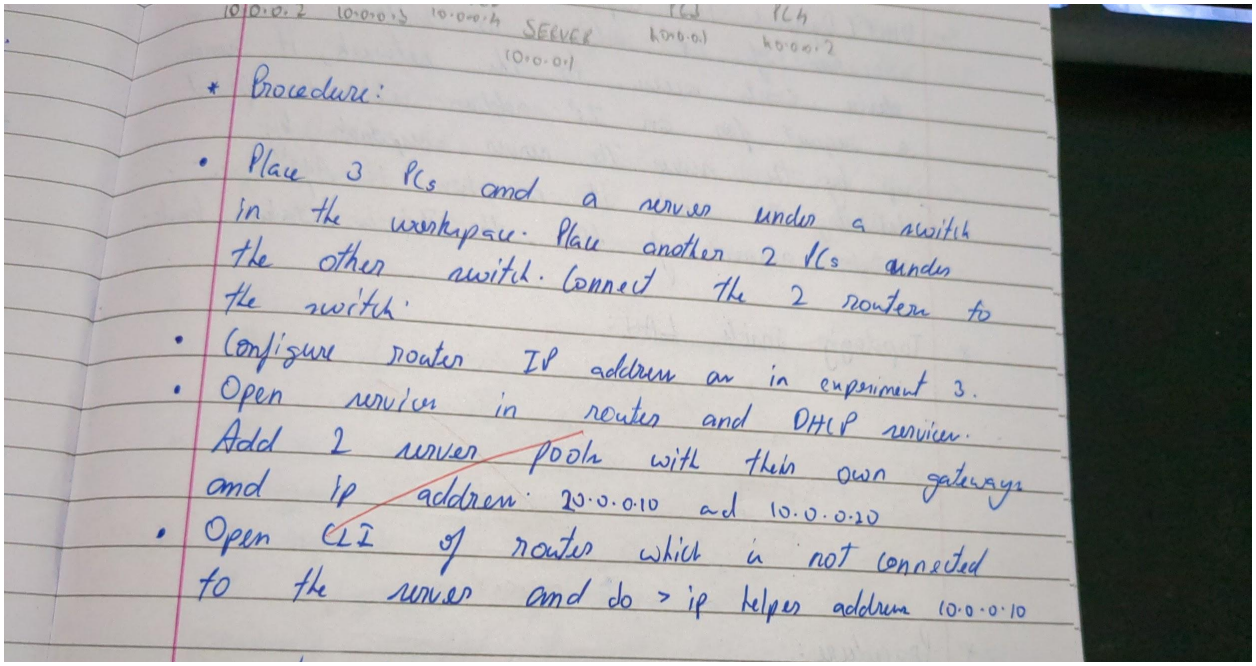
Experiment 4

Aim of the program: Configure DHCP within a LAN and outside LAN

Topology:

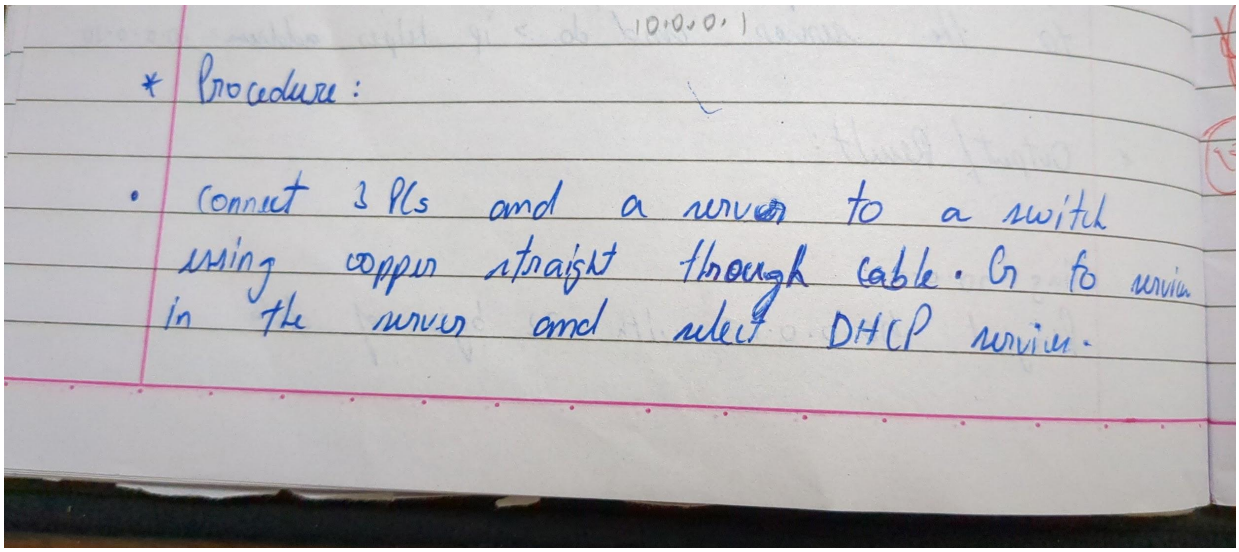


Procedure:



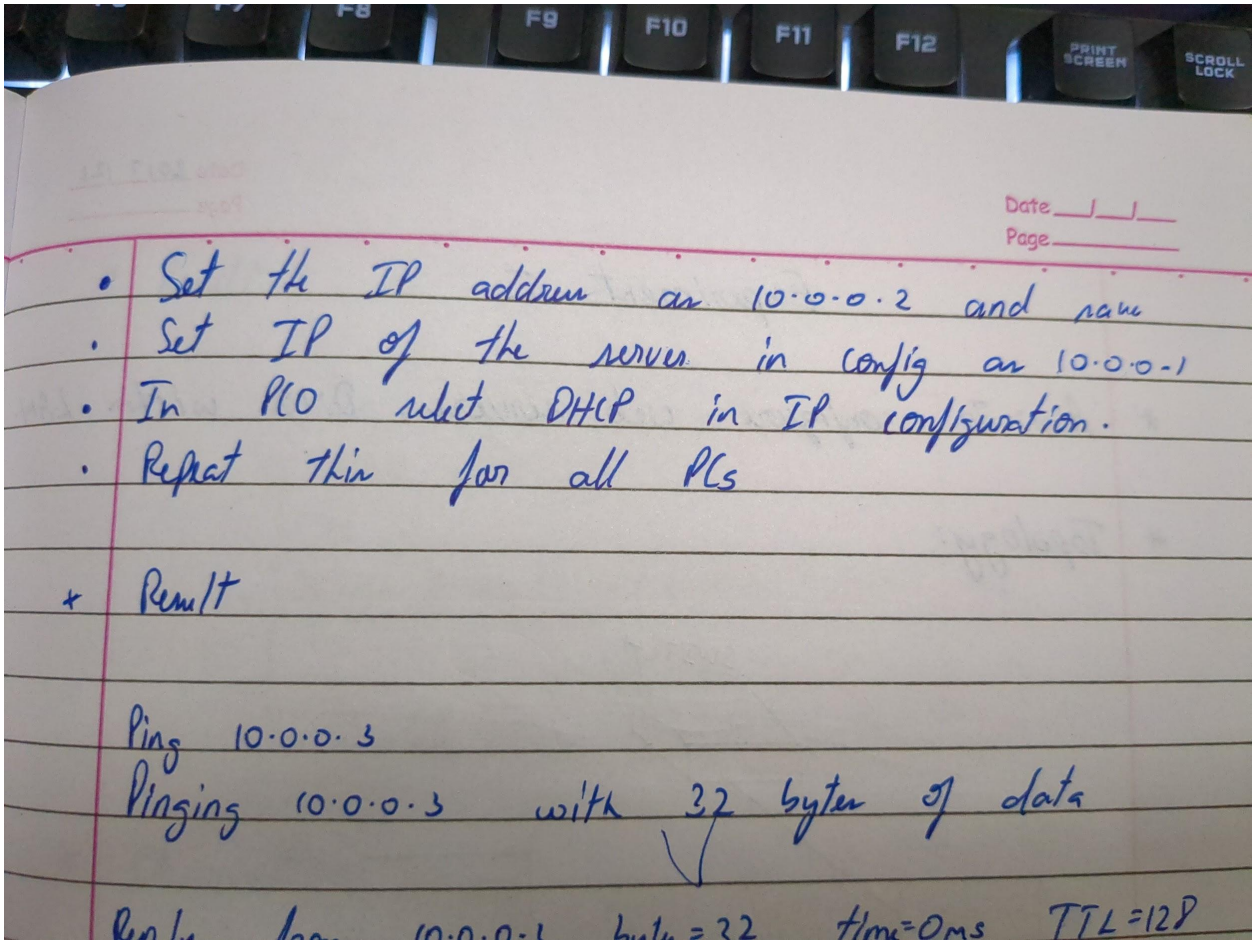
* Procedure:

- Place 3 PCs and a server under a switch in the workspace. Place another 2 PCs under the other switch. Connect the 2 routers to the switch.
- Configure router IP address as in experiment 3.
- Open service in router and DHCP service. Add 2 server pool with their own gateways and ip address: 20.0.0.10 and 10.0.0.20
- Open CLI of router which is not connected to the server and do > ip helper address 10.0.0.10

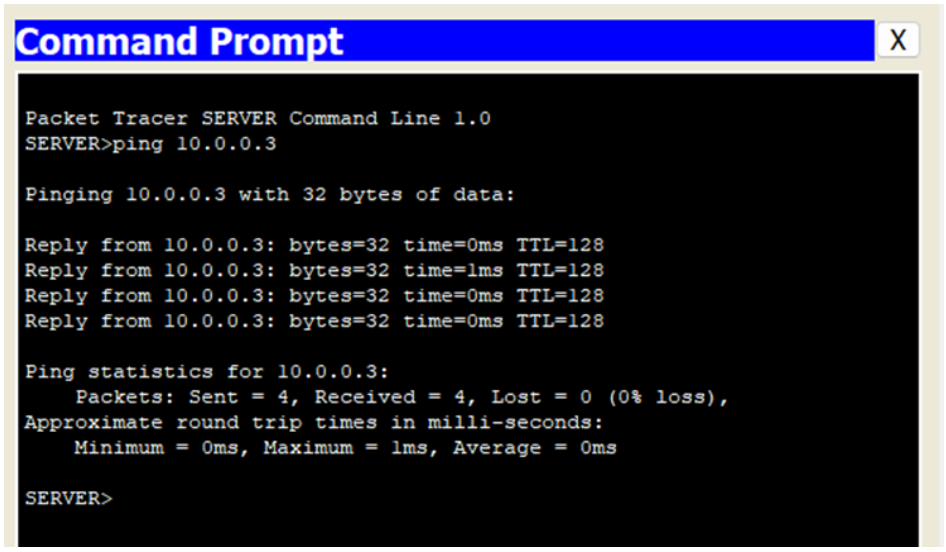


* Procedure:

- Connect 3 PCs and a server to a switch using copper straight through cable. Go to service in the server and select DHCP service.



Output:



Observation

* Observation:

- DHCP (Dynamic host configuration protocol) helps with allocation of IP address to end-users. When a device wants access to the network, it sends a request for an IP address which is picked up by the server. The server responds by delivering IP and it monitors it. After a certain amount of time, the IP is taken back.

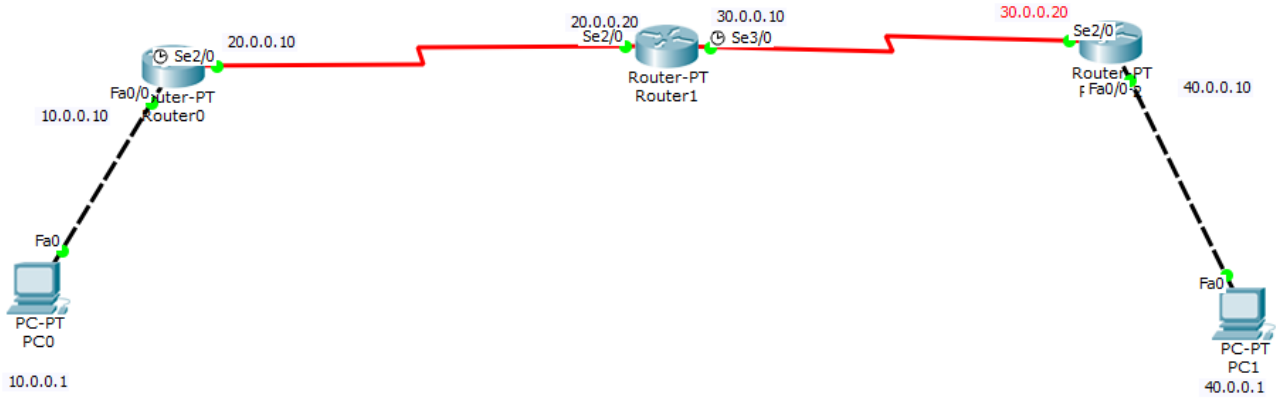
* Observation:

- DHCP dynamically assigns an IP address to any device or node.
- It is a client server protocol where the servers manages a pool of unique IP address
- It responds to the client request by providing IP configuration information from address pools, previously specified by a network administrator

Experiment 5

Aim of the program: Configure RIP routing Protocol in Routers

Topology:



Procedure:

* Procedure:

- Place 2 PCs and 3 routers in the workspace. Connect them as shown in the topology using the appropriate cables.
- Assign IP addresses 10.0.0.1 and 40.0.0.1 and gateways 10.0.0.10 and 40.0.0.10 to the PCs respectively.
- Setup the routers as done normally to configure IPs in CLI.

```
enable
config t
interface portname
ip address ip subnet
no shut
exit
```

encapsulation ppp
clock rate 64000

Date __/__/__
Page _____

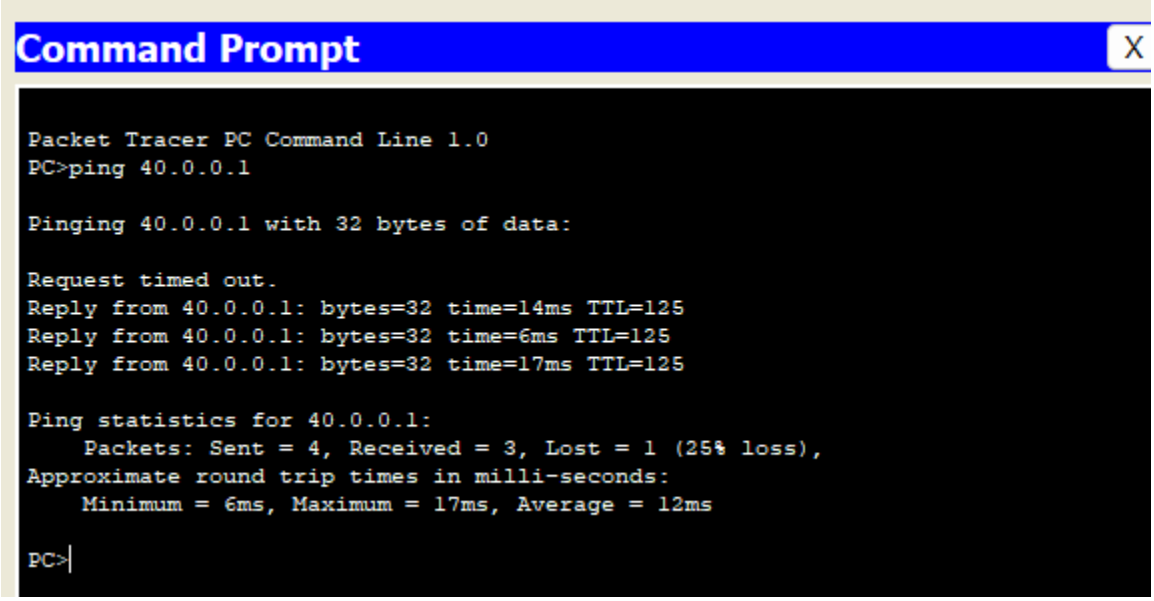
- next, setup the hops using RIP

```
router rip
network ip
network ip
```

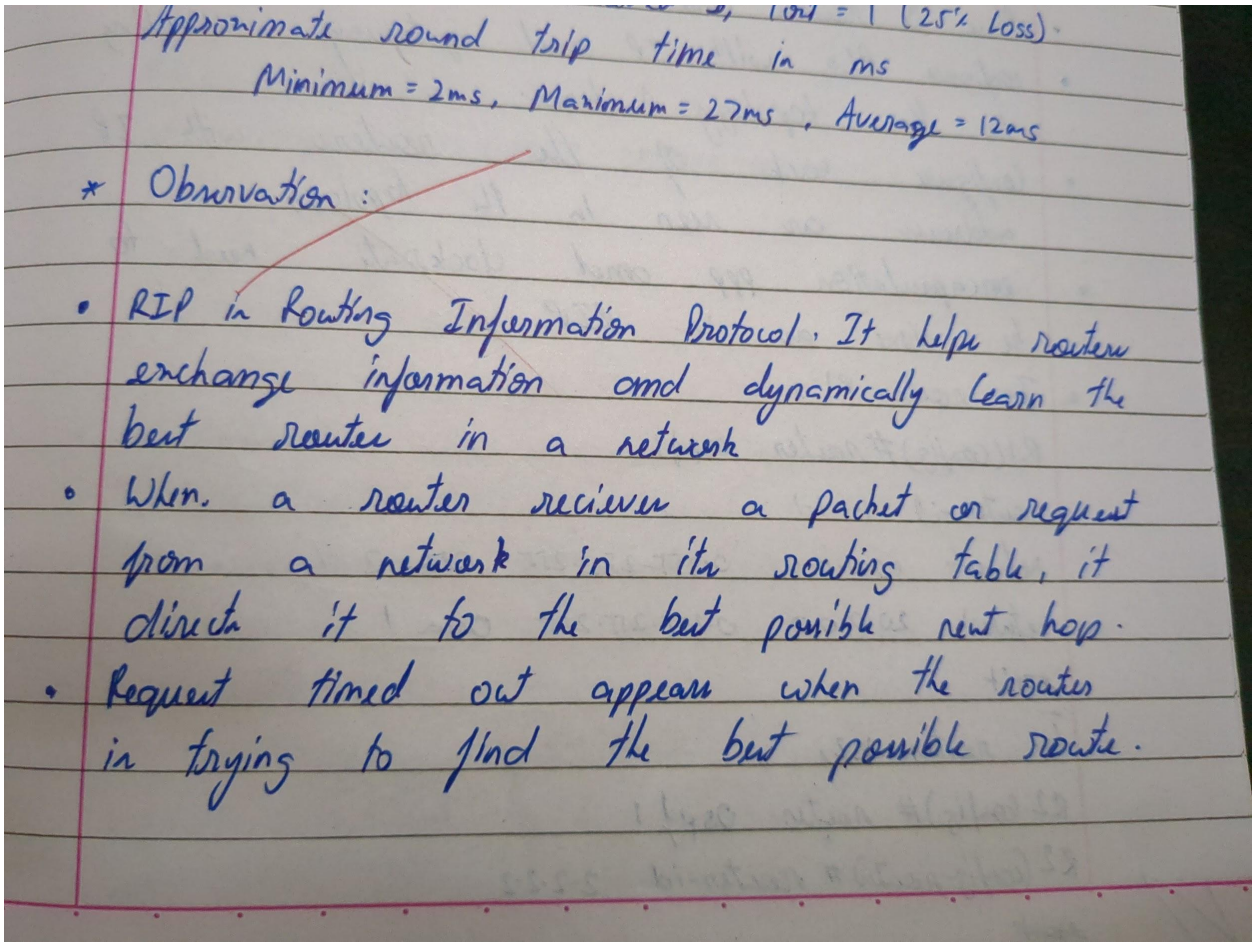
- next, check ip route using: show ip route

Output:

```
Router>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
     20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       20.0.0.0/8 is directly connected, Serial2/0
C       20.0.0.20/32 is directly connected, Serial2/0
R    30.0.0.0/8 [120/1] via 20.0.0.20, 00:00:16, Serial2/0
R    40.0.0.0/8 [120/2] via 20.0.0.20, 00:00:16, Serial2/0
Router>
```



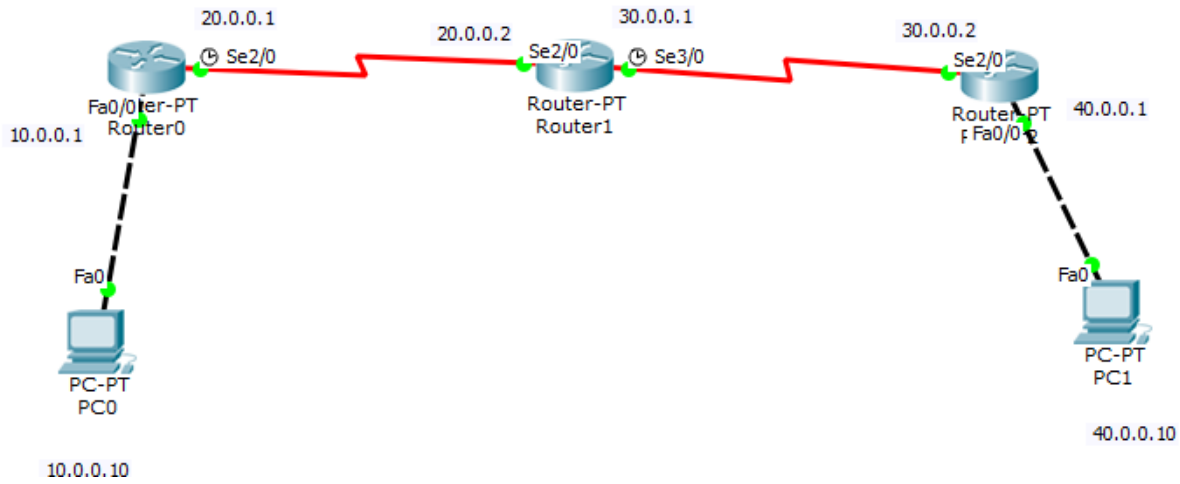
Observation



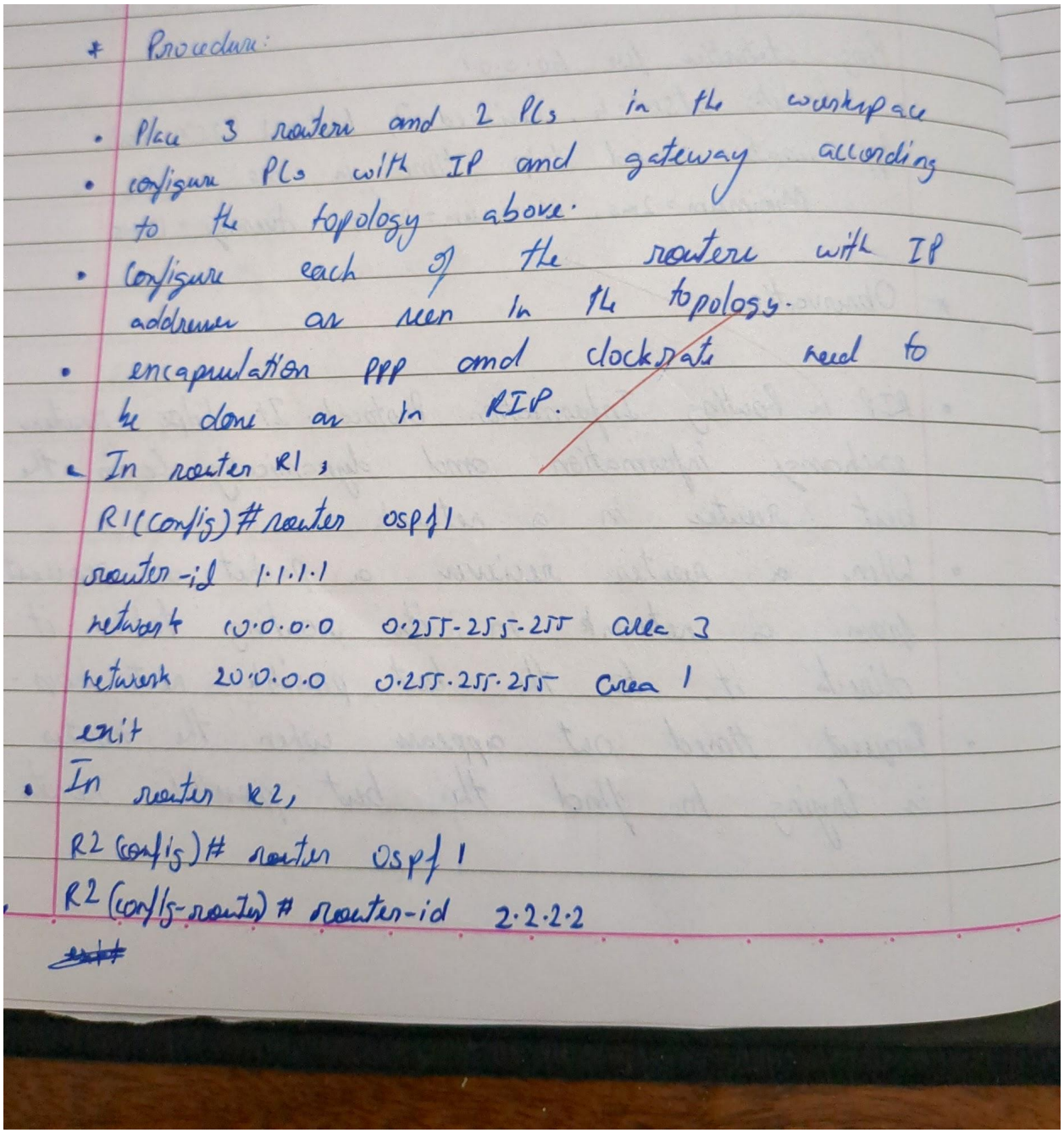
Experiment 6

Aim of the program: Configure OSPF routing protocol

Topology:



Procedure:



* Procedure:

- Place 3 routers and 2 PCs in the workspace
- configure PCs with IP and gateway according to the topology above.
- Configure each of the routers with IP addresses as seen in the topology.
- encapsulation PPP and clock rate need to be done as in RIP.
- In router R1,

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

Date __/__/__
Page

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,
router ospf 1
router-id 3.3.3.3
network 30.0.0.0 0.255.255.255 area 0
network 400.0.0.0 0.255.255.255 area 2
exit

R1(config-if)# interface loopback 0
ip address 172.16.1.252 255.255.0.0
no shut

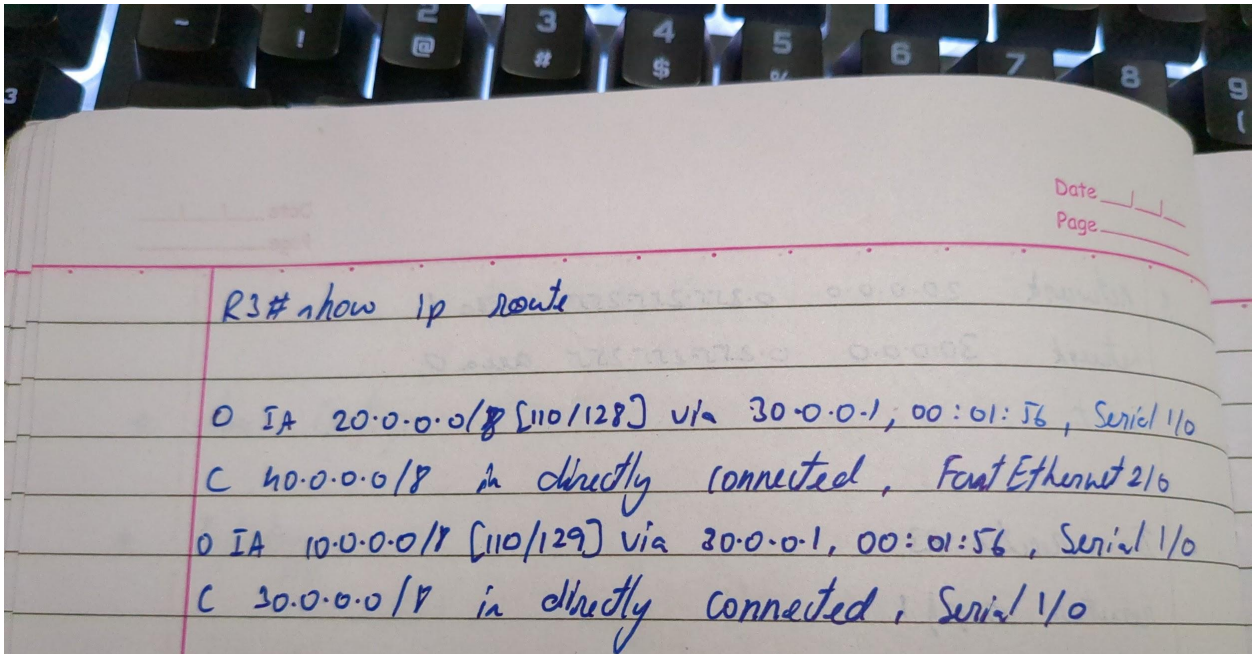
R2(config-if)# interface loopback 0
ip address 172.16.1.253 255.255.0.0
no shut

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

R3# show ip route

- In Router R1
R1(config)# router ospf 1
R1(config-router)# area 1 virtual-link 2.2.2.2

- In router R2
R2(config)# router ospf 2
area 1 virtual-link 1.1.1.1
exit



Output:

```
Router>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
     20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       20.0.0.0/8 is directly connected, Serial2/0
C       20.0.0.2/32 is directly connected, Serial2/0
R    30.0.0.0/8 [120/1] via 20.0.0.2, 00:00:01, Serial2/0
R    40.0.0.0/8 [120/2] via 20.0.0.2, 00:00:01, Serial2/0
C    172.16.0.0/16 is directly connected, Loopback0
Router>
```

```
Router>show ip ospf database
            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
1.1.1.1         1.1.1.1         110         0x80000002 0x00608c 1

                Summary Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        1.1.1.1         130         0x80000001 0x00f25d
20.0.0.0        1.1.1.1         120         0x80000002 0x00e61f

                Router Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum Link count
2.2.2.2         2.2.2.2         119         0x80000004 0x004341 2
1.1.1.1         1.1.1.1         110         0x80000005 0x00a1e6 2

                Summary Net Link States (Area 1)
Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        1.1.1.1         130         0x80000001 0x00f25d
30.0.0.0        2.2.2.2         115         0x80000001 0x0048b0
40.0.0.0        2.2.2.2         115         0x80000002 0x00cd1f

                Router Link States (Area 3)

Link ID         ADV Router      Age         Seq#       Checksum Link count
1.1.1.1         1.1.1.1         134         0x80000001 0x00f749 1
Router>
```

Command Prompt X

```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time=8ms TTL=253
Reply from 40.0.0.1: bytes=32 time=15ms TTL=253
Reply from 40.0.0.1: bytes=32 time=14ms TTL=253
Reply from 40.0.0.1: bytes=32 time=7ms TTL=253

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 15ms, Average = 11ms

PC>
```

Observation

Ping statistics for ...
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milliseconds:
Minimum = 2ms, Maximum = 11ms, Average = 5ms

* Observation:

OSPF 3/8/22

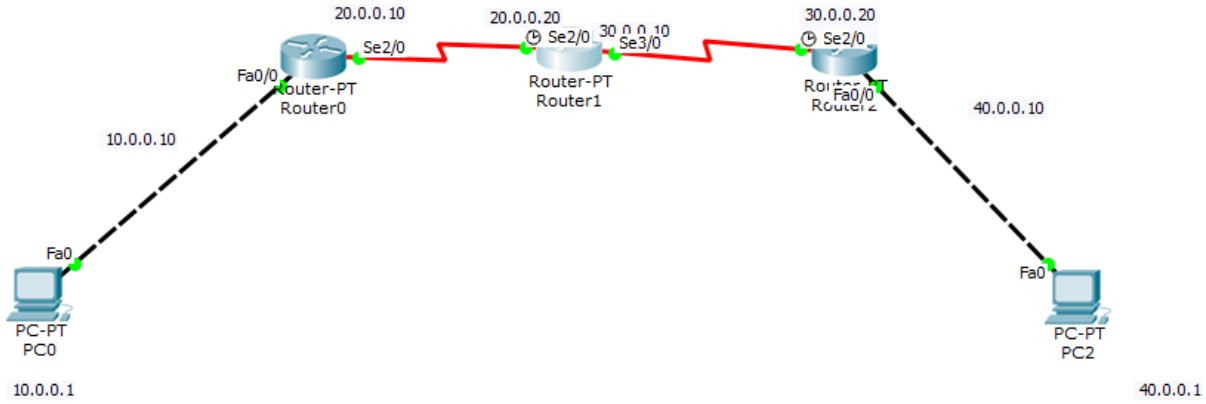
OSPF is Open Shortest Path First. It is a protocol which finds the best routing path between source and destination routers. It uses its own shortest path algorithm

- Networks are divided into areas. Backbone (area0) forms core of the OSPF network. Other networks are connected to the backbone.

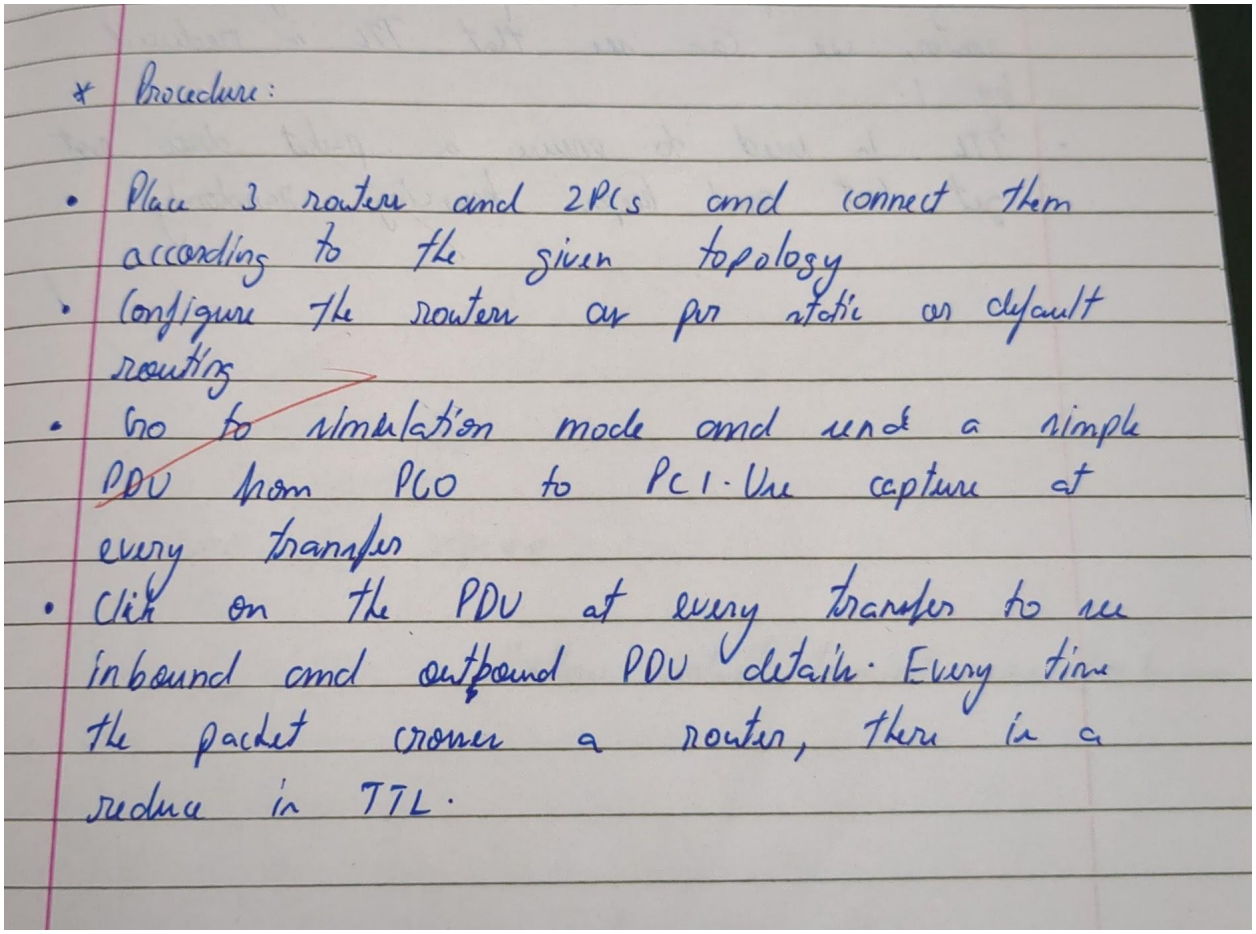
Experiment 7

Aim of the program: Demonstrate the TTL/ Life of a Packet

Topology:



Procedure:


* Procedure:
• Place 3 routers and 2PCs and connect them according to the given topology
• Configure the routers as per static or default routing
• Go to simulation mode and send a simple PDU from PC0 to PC1. Use capture at every transfer
• Click on the PDU at every transfer to see inbound and outbound PDU details. Every time the packet crosses a router, there is a reduce in TTL.

Output:

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+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: 0x2			0x0	0x0	
TTL: 254		PRO: 0x1	CHKSUM		
SRC IP: 10.0.0.1					
DST IP: 40.0.0.1					
OPT: 0x0				0x0	
DATA (VARIABLE LENGTH)					

ICMP

0	8	16	31 Bits
TYPE: 0x8	CODE: 0x0	CHECKSUM	
ID: 0x3		SEQ NUMBER: 2	

PDU Information at Device: Router1

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

HDLC

0	8	16	32	32+x	48+x	56+x
FLG: 0111 1110	ADR: 0x8f	CONTROL: 0x0	DATA: (VARIABLE LENGTH)		FCS: 0x0	FLG: 0111 1110

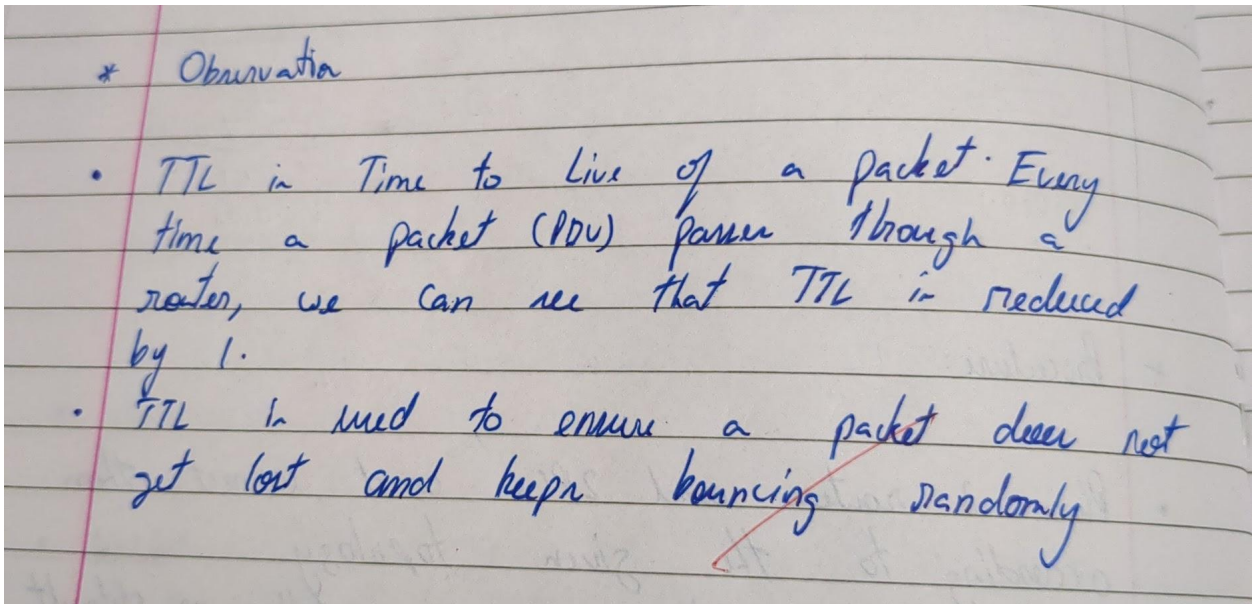
IP

0	4	8	16	19	31 Bits
4	IHL	DSCP: 0x0	TL: 28		
ID: 0x2			0x0	0x0	
TTL: 253		PRO: 0x1	CHKSUM		
SRC IP: 10.0.0.1					
DST IP: 40.0.0.1					
OPT: 0x0				0x0	
DATA (VARIABLE LENGTH)					

ICMP

0	8	16	31 Bits
TYPE: 0x8	CODE: 0x0	CHECKSUM	
ID: 0x3		SEQ NUMBER: 2	

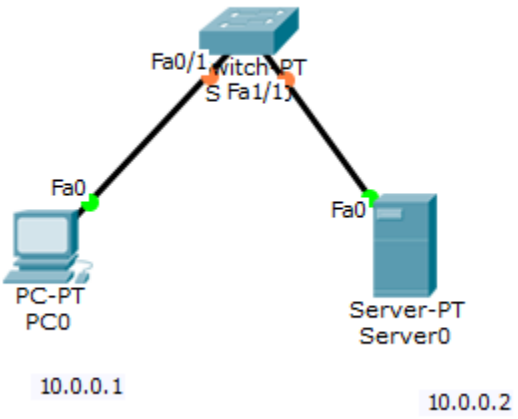
Observation



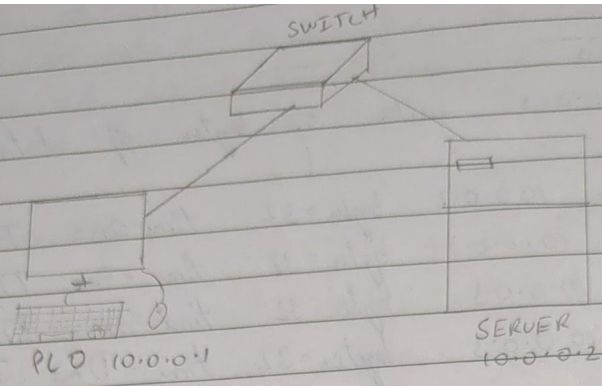
Experiment 8

Aim of the program: Configure Web Server, DNS within a LAN

Topology:



Procedure:

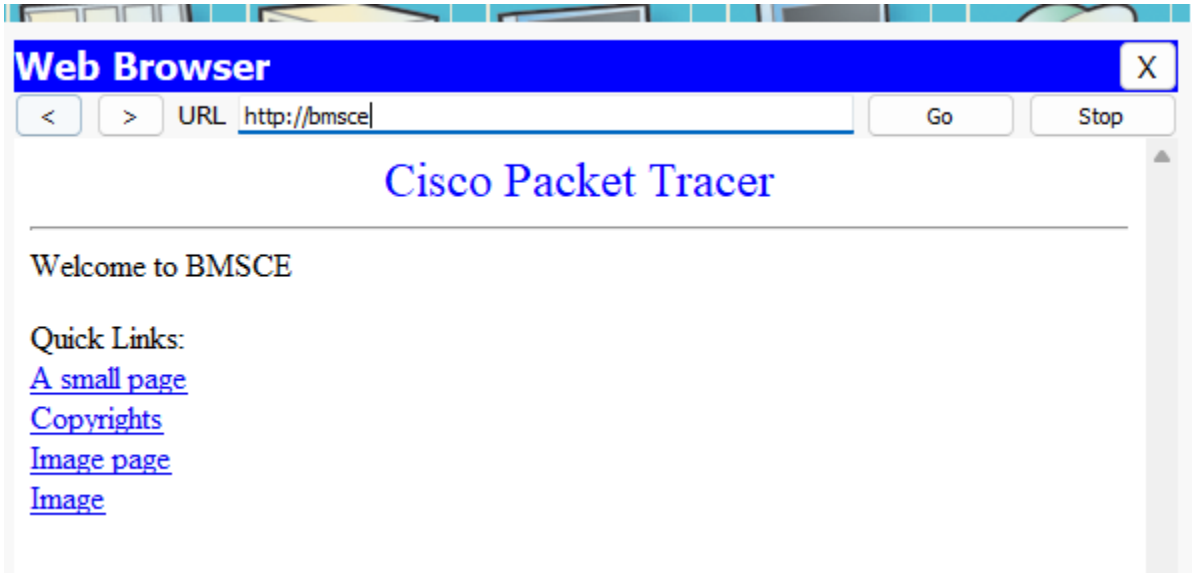


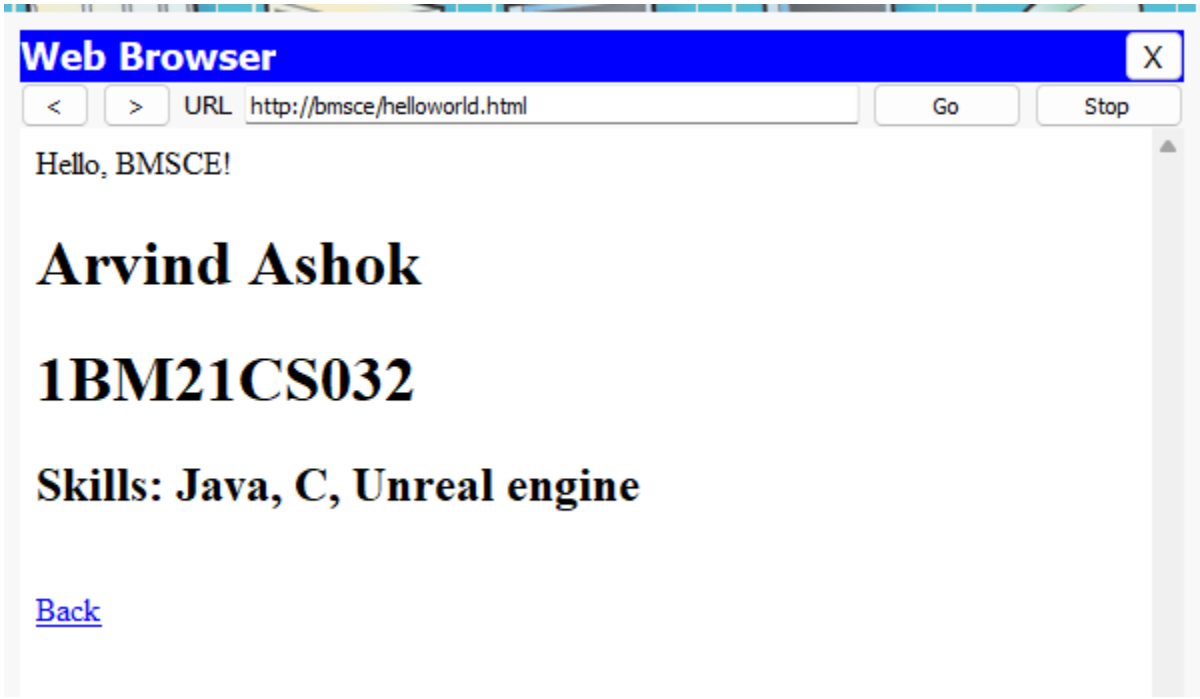
* Procedure:

- Place a PC, server and switch in the work space. Connect the PC and server to the switch to create a LAN
- Turn on HTTP service in the service of settings of the server.
- In the index.html file in file manager, add your information as seen in a resume.
- Go to DNS service and turn it on. Add a path 'bmsce.com' and address 10.0.0.2.
- In the web browser of the PC, type bmsce.com or 10.0.0.2 to view the website..

Output:

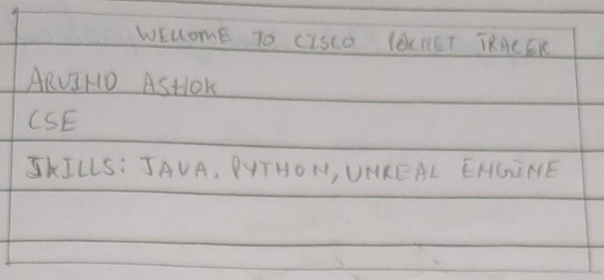
No.	Name	Type	Detail
0	bmsce	A Record	10.0.0.2





Observation

* Result:



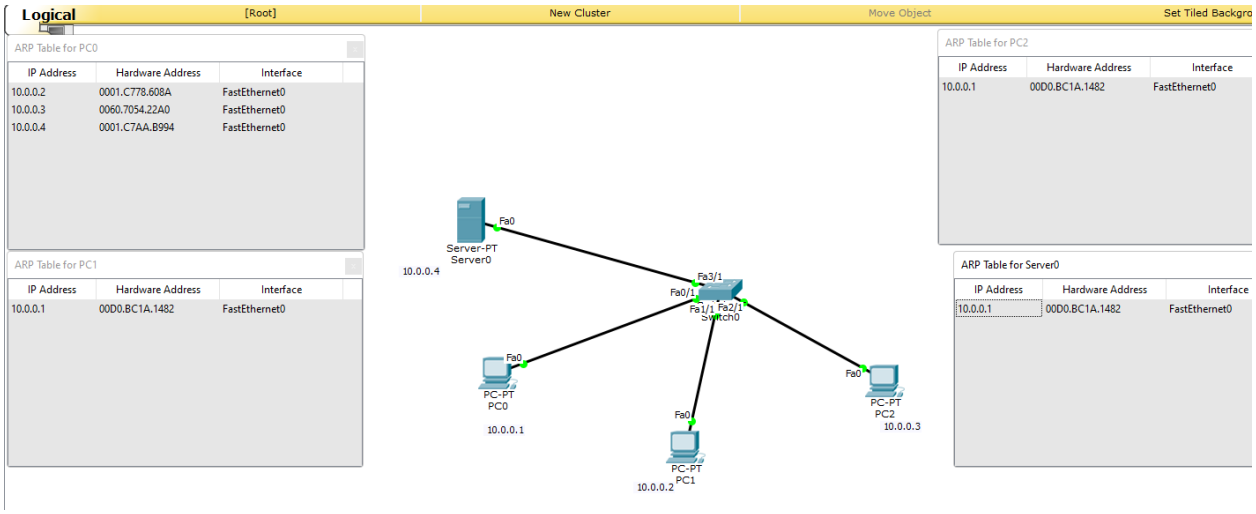
* Observation :

- LAN is a local area network formed by switch, server and PC.
- DNS is Domain Name Server. It is responsible for linking domain name like 'bmsce.com' with its associated ip address. In this case the IP address is 10.0.0.2.
- When the domain or IP is searched, the DNS responds by displaying the website in this case.

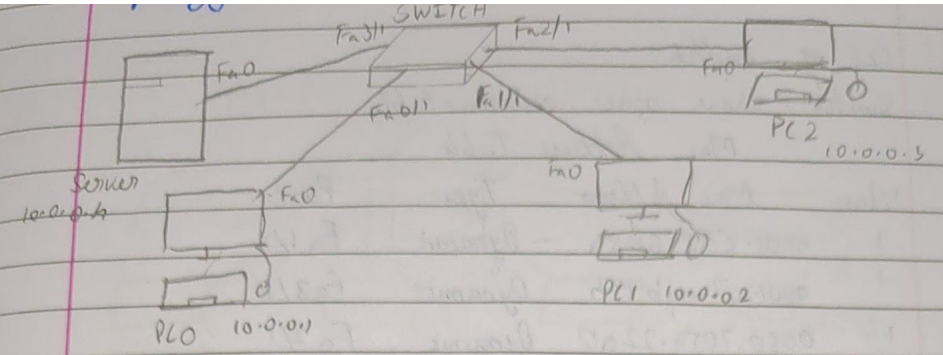
Experiment 9

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

Topology:



Procedure:



* Procedure:

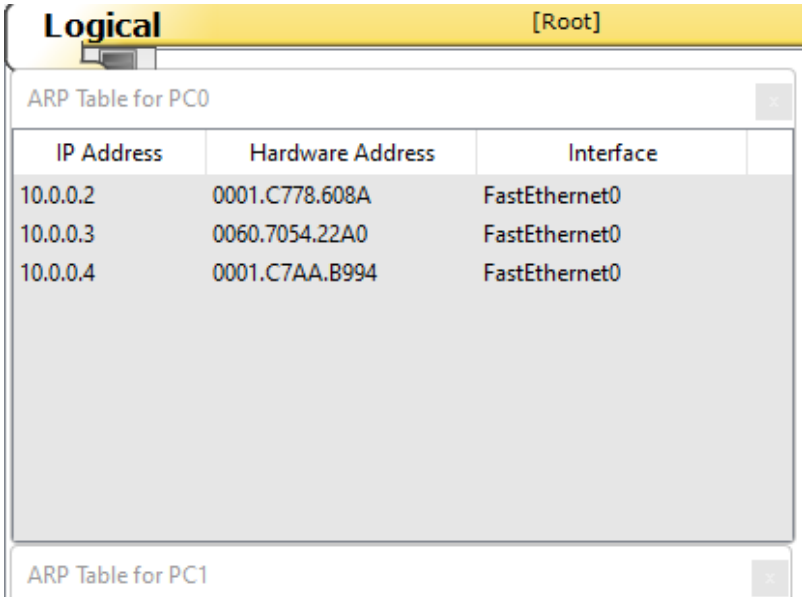
- Place a generic switch, 3 generic PCs and 1 hub in the workspace.
- Connect the PCs and hub to the switch as shown in the above topology. Set IP address for the devices.
- Next, go to the command prompt of any PC and type: arp -a
  Initially there will be no arp entries.
- Go to the command prompt and ping all the connected devices using the IP address one by one. Go to simulation mode in each case and click capture to see packets moving.
- Using inspect, open ARP table for each device.
- In CLI of switch, type show mac address-table to view mac address table.

Output:

```
Packet Tracer PC Command Line 1.0
PC>arp -a
No ARP Entries Found
PC>
```

```
PC>arp -a
  Internet Address      Physical Address      Type
  10.0.0.2              0001.c778.608a        dynamic
  10.0.0.3              0060.7054.22a0        dynamic
  10.0.0.4              0001.c7aa.b994        dynamic
```

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

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

   1    0001.c778.608a    DYNAMIC     Fa1/1
   1    0001.c7aa.b994    DYNAMIC     Fa3/1
   1    0060.7054.22a0    DYNAMIC     Fa2/1
   1    00d0.bc1a.1482    DYNAMIC     Fa0/1
```



Observation

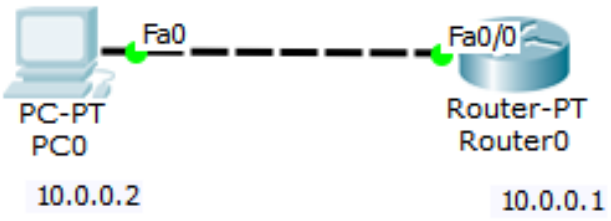
* Observation:

- ARP is Address Resolution Protocol. It uses the IP address to find out the physical address/MAC address/Link layer address.
- When we ping a device using IP, ARP sends an ARP request to the given IP, which responds by giving its MAC address. This MAC address is stored in the MAC address table.

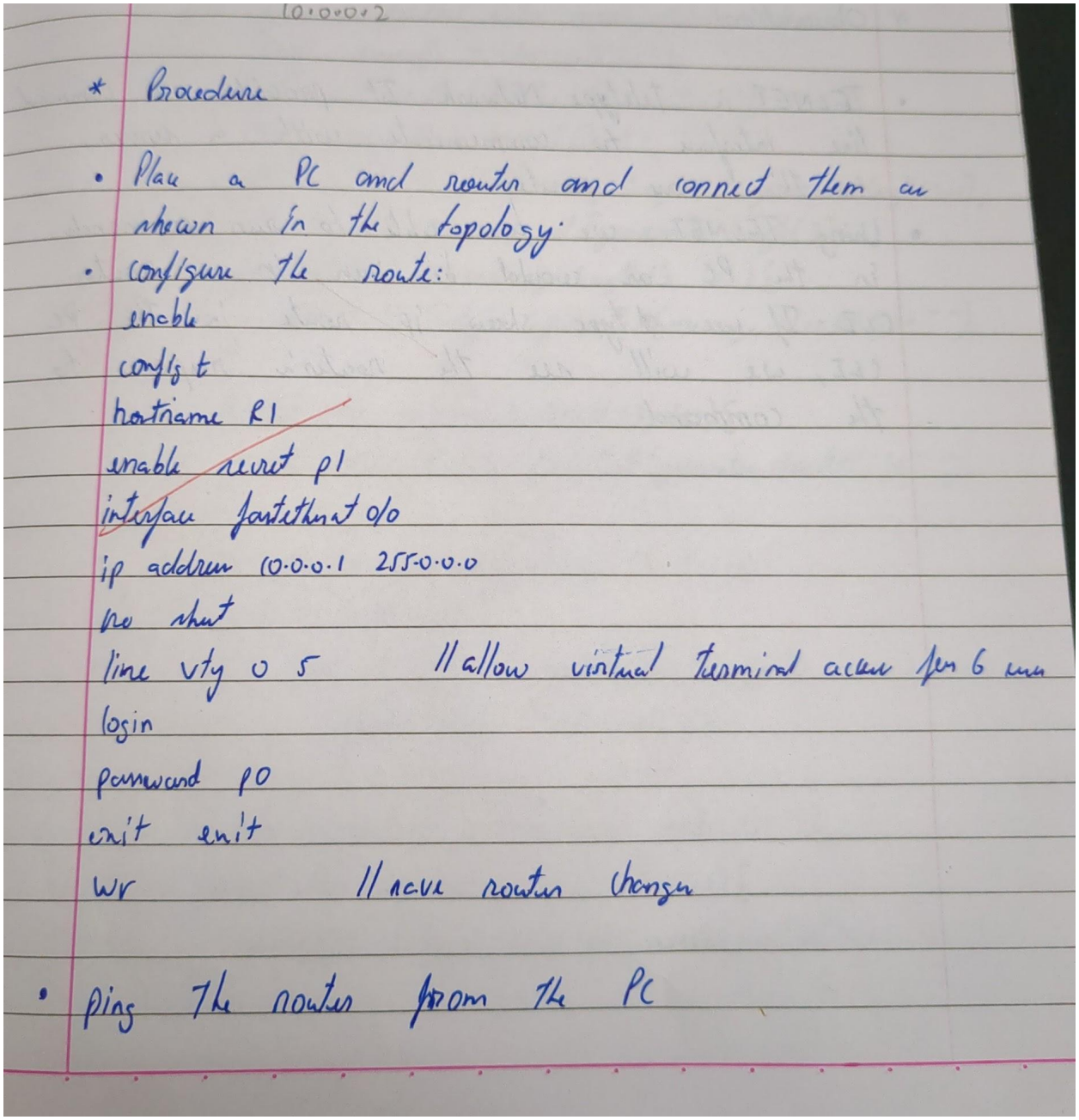
Experiment 10

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

Topology:



Procedure:


10.0.0.2
* Procedure
• Place a PC and router and connect them as shown in the topology.
• configure the route:
enable
config t
hostname R1
enable secret p1
interface fastethernet 0/0
ip address 10.0.0.1 255.0.0.0
no shut
line vty 0 5 // allow virtual terminal access for 6 users
login
password p0
exit exit
wr // save router changes
• ping the router from the PC

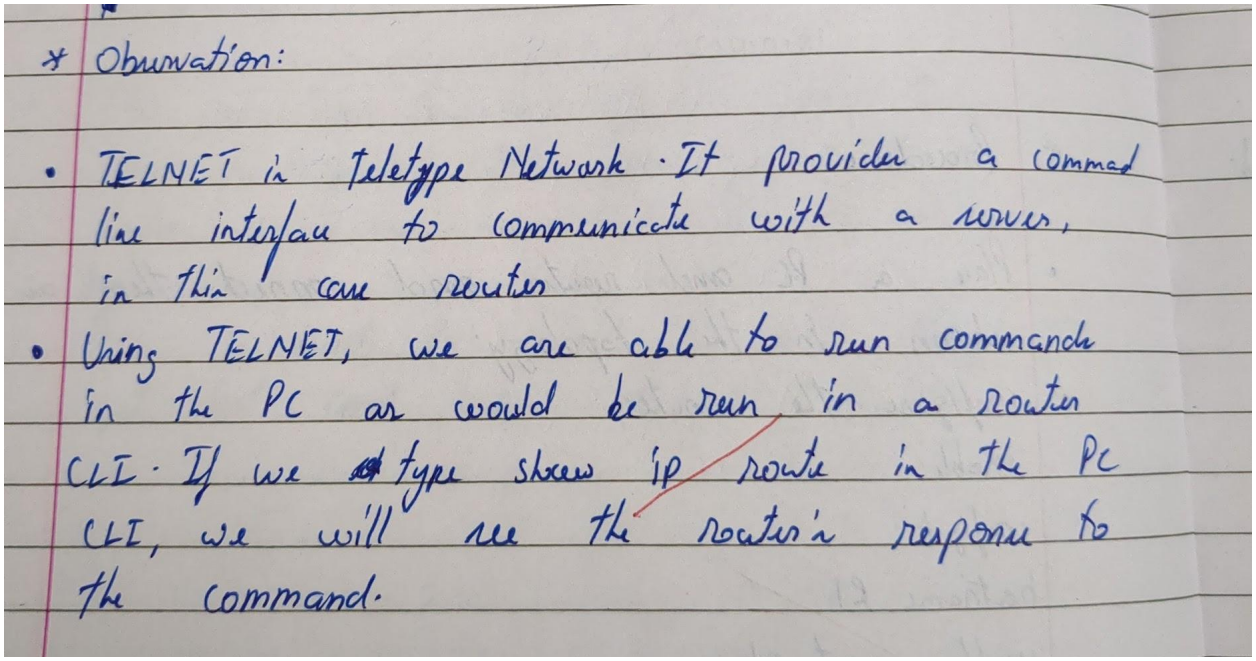
Output:

```
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname rl
rl(config)#enable secret pl
rl(config)#interface fa0/0
rl(config-if)#ip add 10.0.0.0 255.0.0.0
Bad mask /8 for address 10.0.0.0
rl(config-if)#ip add 10.0.0.1 255.0.0.0
rl(config-if)#no shut

rl(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

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

rl(config-if)#line vty 0 5
rl(config-line)#login
% Login disabled on line 132, until 'password' is set
% Login disabled on line 133, until 'password' is set
% Login disabled on line 134, until 'password' is set
% Login disabled on line 135, until 'password' is set
% Login disabled on line 136, until 'password' is set
% Login disabled on line 137, until 'password' is set
rl(config-line)#password p0
rl(config-line)#
rl(config-line)#exit
rl(config)#exit
rl#
%SYS-5-CONFIG_I: Configured from console by console

rl#wr
Building configuration...
[OK]
```

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


User Access Verification

Password:
r1>enable
Password:
Password:
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#exit
```

Observation



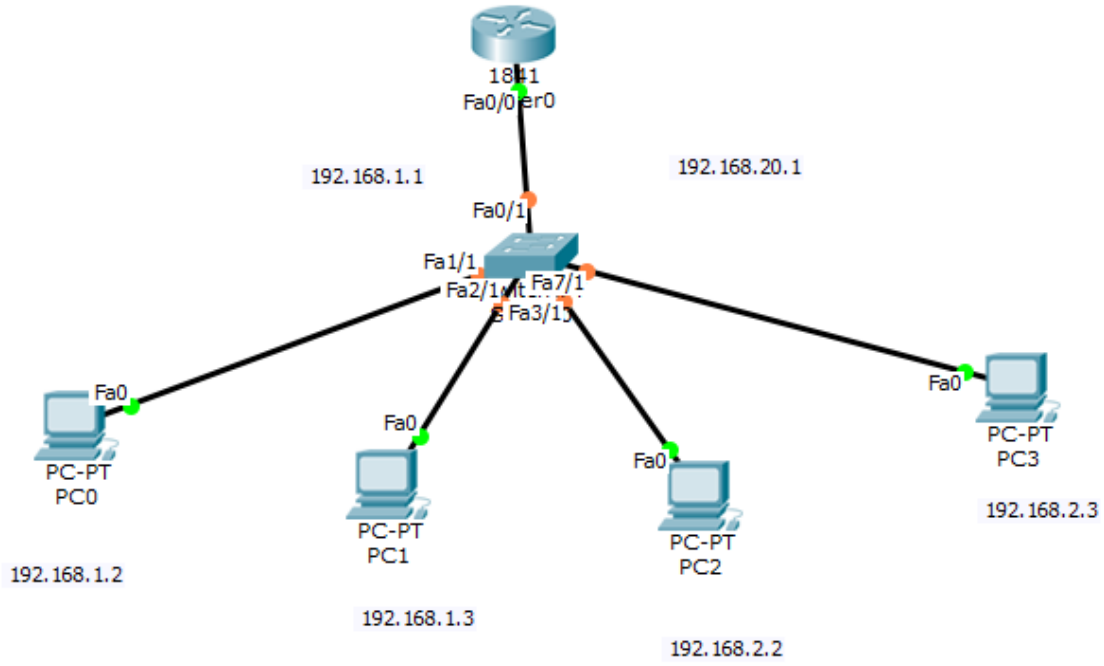
* Observation:

- TELNET is Teletype Network. It provides a command line interface to communicate with a server, in this case routers
- Using TELNET, we are able to run commands in the PC as would be run in a router CLI. If we type show ip route in the PC CLI, we will see the router's response to the command.

Experiment 11

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

Topology:

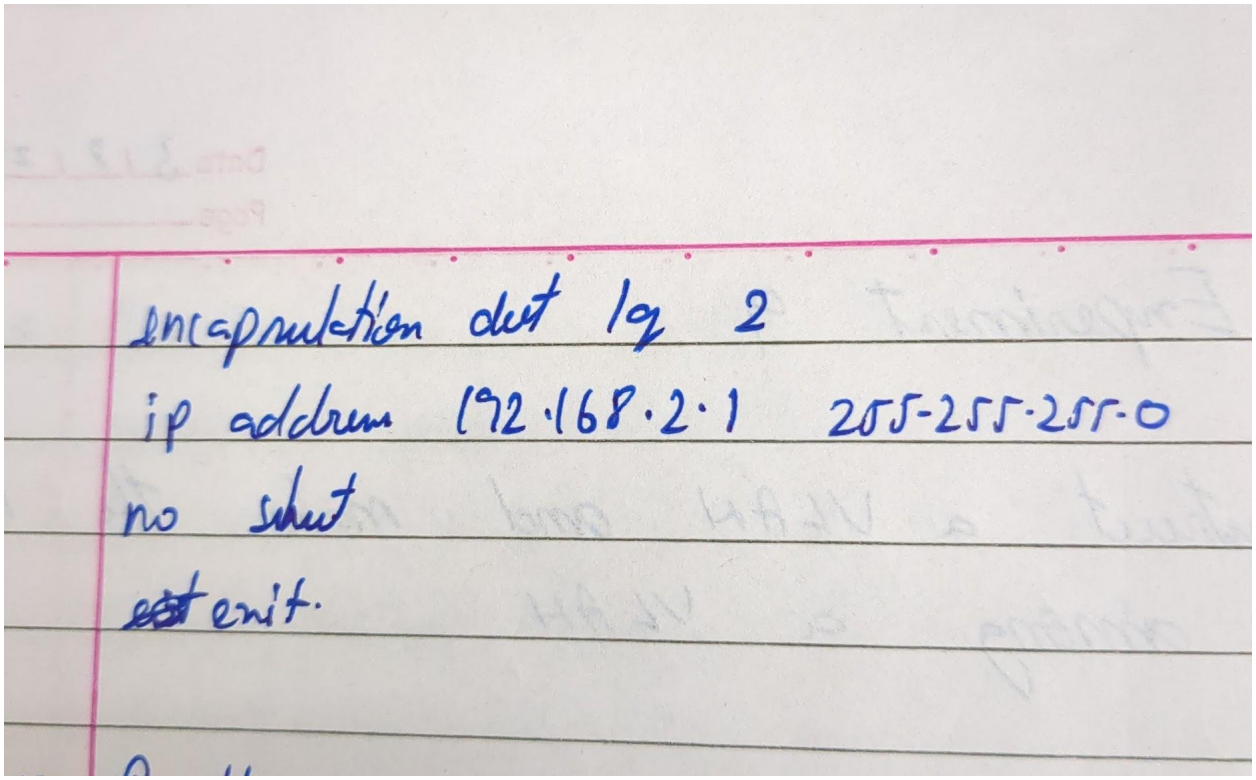


Procedure:

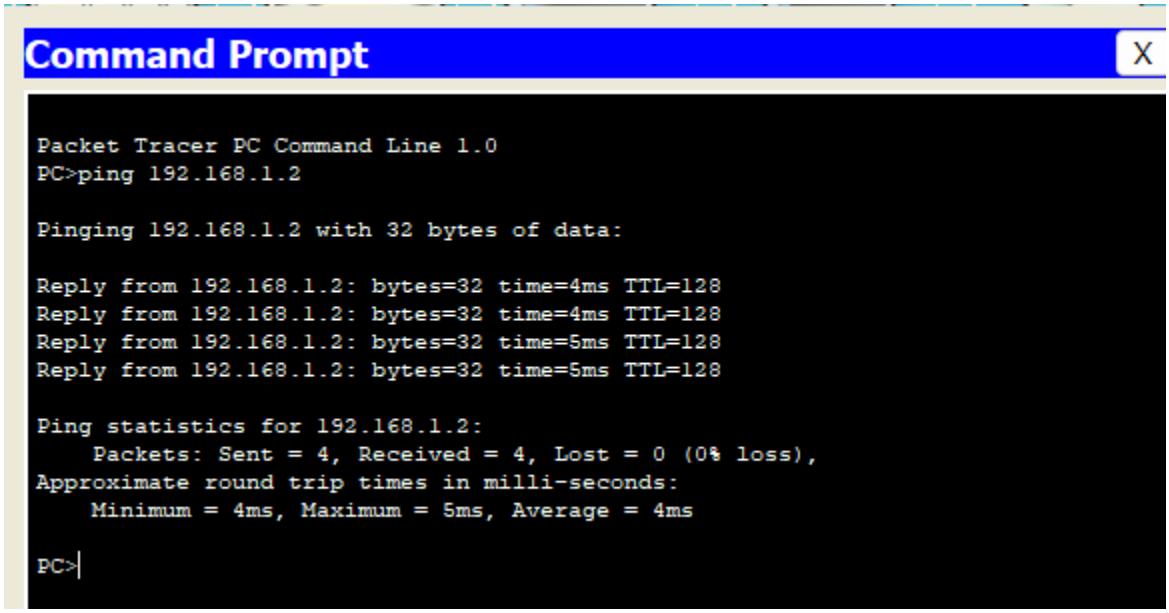
* Procedure:

- Place a router (1841), a switch, and 4 PCs in the workspace and connect them according to the topology.
- In switch, go to config and select VLAN. Select FastEthernet and make it Trunk. This allows switch to forward frames from different VLANS over a single link.
- Go to config of router and select VLAN and enter the number and name of VLAN.
- Go to CLI

```
Router# config t
Router(config)# interface FastEthernet 0/0.1
Router(config-subif)#
```



Output:



Observation

* Observation

- VLAN is Virtual Local Area Network. Here, a network is subdivided into smaller network for a group of devices
- It is in the same physical network. It operates at layer 2 of the network (DLL).

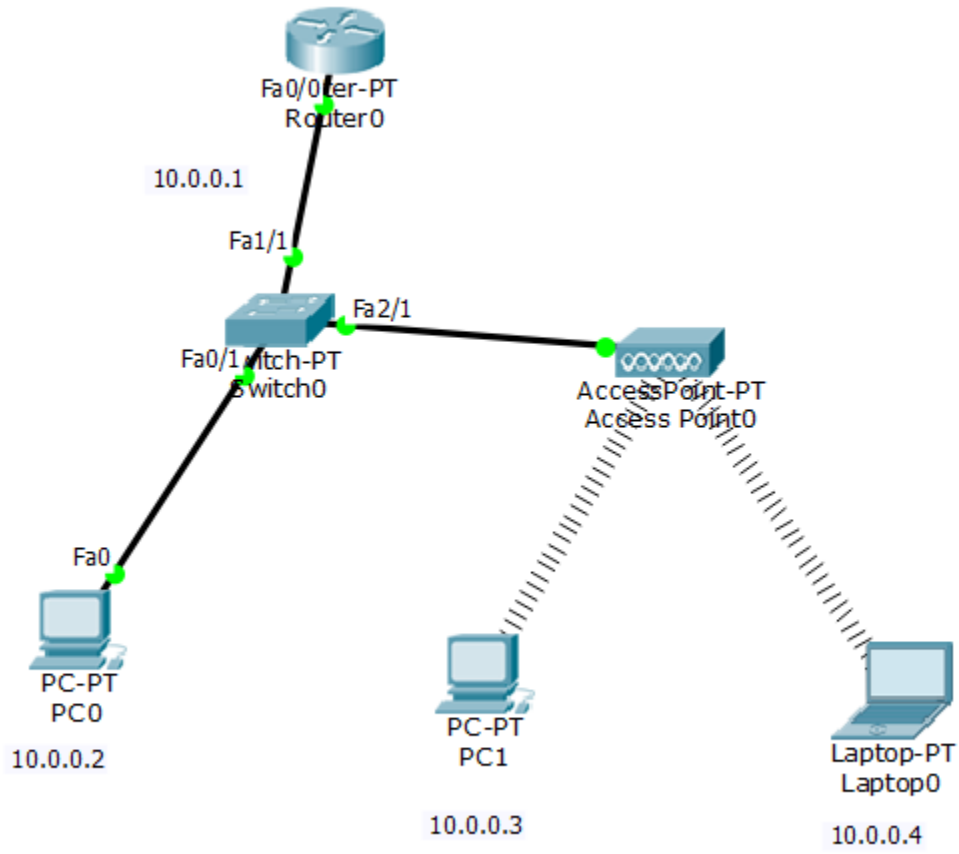
VLANs improve network security

A subset of devices share the same physical LAN, but are subdivided in the data link layer

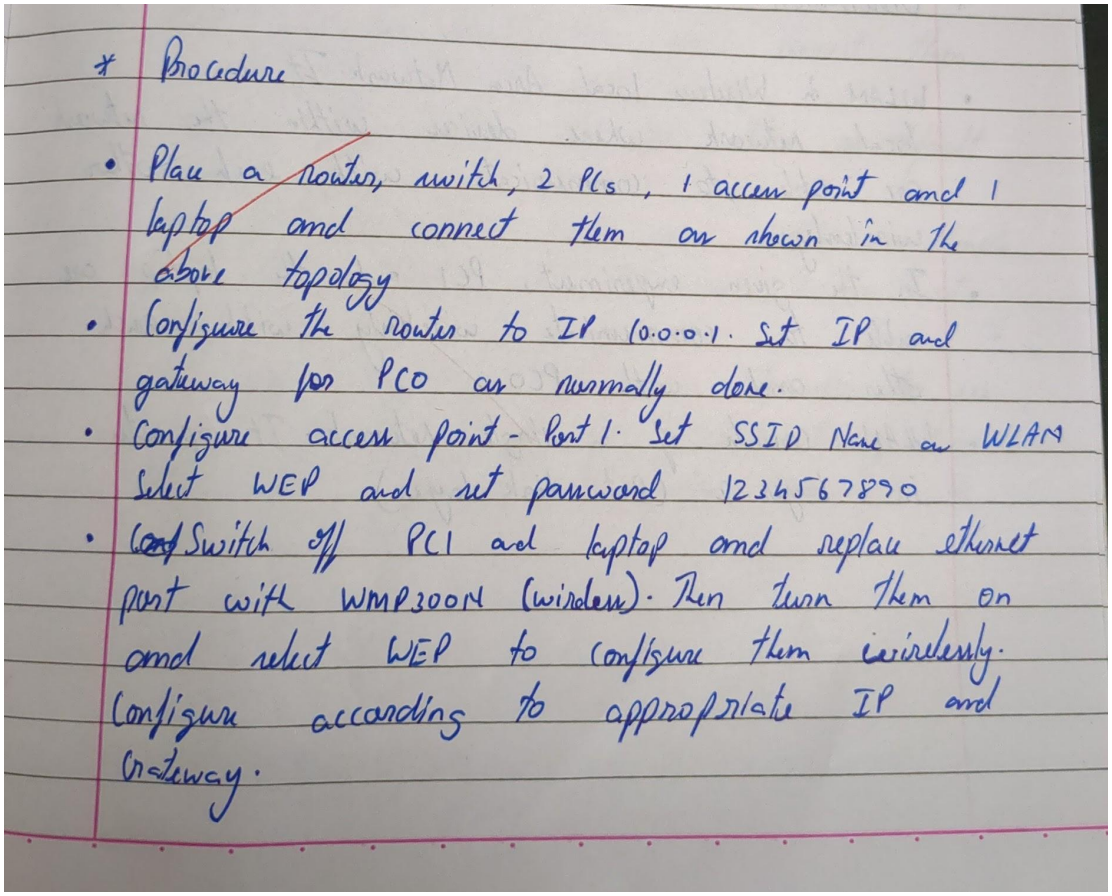
Experiment 12

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

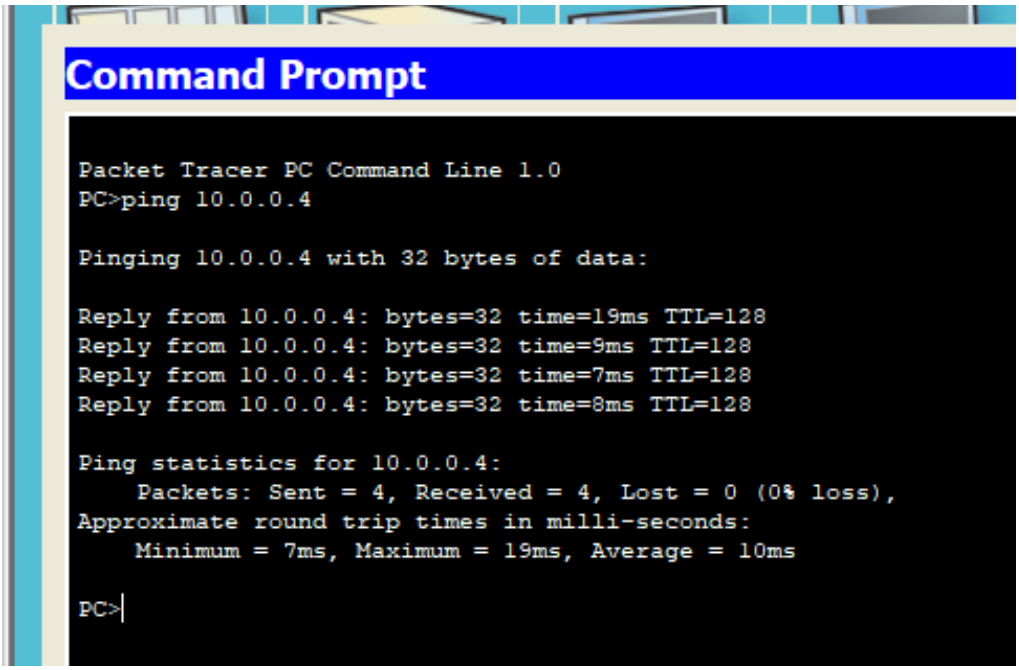
Topology:



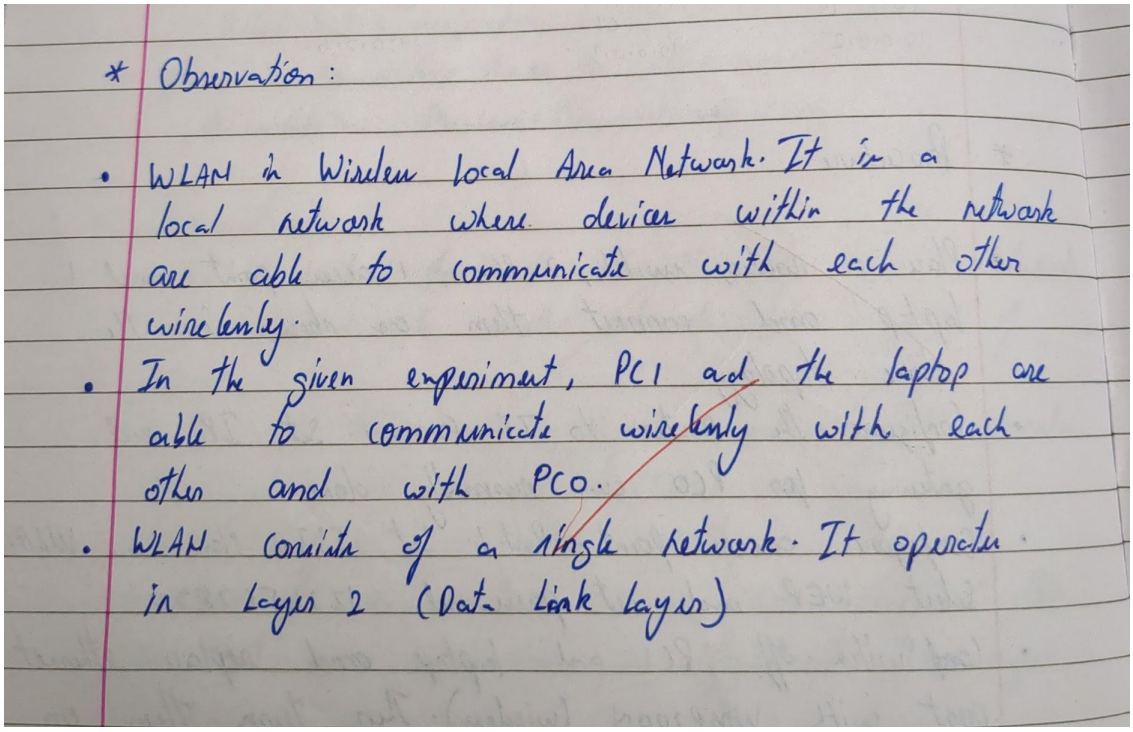
Procedure:


* Procedure
• Place a router, switch, 2 PCs, 1 access point and 1 laptop and connect them as shown in the above topology
• Configure the router to IP 10.0.0.1. Set IP and gateway for PC0 as normally done.
• Configure access point - Port 1. Set SSID Name as WLAN Select WEP and set password 1234567890
• Switch off PC1 and laptop and replace ethernet port with WMP300N (wireless). Then turn them on and select WEP to configure them wirelessly. Configure according to appropriate IP and Gateway.

Output:


Command Prompt
Packet Tracer PC Command Line 1.0
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=19ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=8ms 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 = 7ms, Maximum = 19ms, Average = 10ms
PC>

Observation



* Observation :

- WLAN is Wireless local Area Network. It is a local network where devices within the network are able to communicate with each other wirelessly.
- In the given experiment, PC1 and the laptop are able to communicate wirelessly with each other and with PC0.
- WLAN consists of a single network. It operates in Layer 2 (Data Link Layer)

CYCLE 2

Experiment 13

Program

```
import java.util.Scanner;


class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        // Input DATA
        System.out.println("Enter Message/Data bits");
        String message = in.nextLine();
        System.out.println("Enter Generator");
        String generator = in.nextLine();
        int data[] = new int[message.length() + generator.length() - 1];
        int divisor[] = new int[generator.length()];

        for (int i = 0; i < message.length(); i++) {
            data[i] = Integer.parseInt(message.charAt(i) + "");
        }
        for (int i = 0; i < generator.length(); i++) {
            divisor[i] = Integer.parseInt(generator.charAt(i) + "");
        }

        // CRC calculation
        for (int i = 0; i < message.length(); i++) {
```

```
    if (data[i] == 1) // if 0, you can just skip
        for (int j = 0; j < divisor.length; j++)
            data[i + j] ^= divisor[j];
}
// Display CRC
System.out.println("The checksum code is: ");
for (int i = 0; i < message.length(); i++)
    data[i] = Integer.parseInt(message.charAt(i) + "");

for (int i = 0; i < data.length; i++)
    System.out.print(data[i]);

System.out.println();

// CHECK for input CRC code
System.out.println("Enter Checksum code");
message = in.nextLine();
System.out.println("Enter generator");
generator = in.nextLine();
data = new int[message.length() + generator.length() - 1];
divisor = new int[generator.length()];

for (int i = 0; i < message.length(); i++)
    data[i] = Integer.parseInt(message.charAt(i) + "");

for (int i = 0; i < generator.length(); i++)
```

```
            divisor[i] = Integer.parseInt(generator.charAt(i) + "");

        // Calculator of remainder
        for (int i = 0; i < message.length(); i++) {
            if (data[i] == 1)
                for (int j = 0; j < divisor.length; j++)
                    data[i + j] ^= divisor[j];
        }

        // validity of data
        boolean valid = true;
        for (int i = 0; i < data.length; i++){
            if (data[i] == 1) {
                valid = false;
                break;
            }
        }
        if (valid == true) {
            System.out.println("Data stream is valid");
        }
        else {
            System.out.println("Data stream is not valid, CRC error");
        }
        in.close();
    }
}
```

Program

using CRC-CCITT (16-bits)

* Java program:

```
import java.util.Scanner;

Class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter message/data bits:");
        String message = in.nextLine();
        System.out.println("Enter Generator:");
        String generator = in.nextLine();
        int data[] = new int[message.length() + generator.length()-1];
        int divisor[] = new int[generator.length()];
        for(int i=0; i<message.length(); i++) {
            data[i] = Integer.parseInt(message.charAt(i)+"");
        }
        for(int i=0; i<generator.length(); i++) {
            divisor[i] = Integer.parseInt(generator.charAt(i)+"");
        }
        for(int i=0; i<message.length(); i++) {
            if(data[i]==1)
                for(int j=0; j<divisor.length; j++)
                    data[i+j] ^= divisor[j];
        }
        System.out.println("The checksum code is:");
        for(int i=0; i<message.length(); i++) {
            data[i] = Integer.parseInt(message.charAt(i)+"");
        }
```

```
        for (int i=0; i<data.length; i++)
            System.out.println(data[i]);
        System.out.println();

        System.out.println("Enter checksum code: ");
        message = in.nextLine();
        System.out.println("Enter generator: ");
        generator = in.nextLine();
        data
        for (int i=0; i<message.length(); i++)
            data[i] = Integer.parseInt(message.charAt(i) + "");

        for (int i=0; i<generator.length(); i++)
            divisor[i] = Integer.parseInt(generator.charAt(i) + "");

        for (int i=0; i<message.length(); i++){
            if (data[i]==1)
                for (int j=0; j<divisor.length; j++)
                    data[i+j] ^= divisor[j];
        }

        boolean valid=true;
        for (int i=0; i<data.length; i++){
            if (data[i]==1){
                valid=false;
                break;
            }
        }

        if (valid) System.out.println("Data stream is valid!");
        else System.out.println("Data stream is invalid! CRC error");
        in.close();
        return 0;
    }
}
```

Output

```
PS C:\Users\arvin\OneDrive\Desktop\1BM21CS032> java Main
Enter Message/Data bits
10110011010
Enter Generator
1011
The checksum code is:
10110011010010
Enter Checksum code
10110011010010
Enter generator
1011
Data stream is valid
PS C:\Users\arvin\OneDrive\Desktop\1BM21CS032> java Main
Enter Message/Data bits
10110011010
Enter Generator
1011
The checksum code is:
10110011010010
Enter Checksum code
10110011010011
Enter generator
1011
Data stream is not valid, CRC error
```

Experiment 14

Program

```
#include <stdio.h>
#include <stdlib.h>

#define capacity 50

void main() {
   int timeLimit = 10;
   int bucketCapacity = 0, outputRate = 5;

   while(timeLimit < 20) {
      int newPacket;
      printf("\nenter new packet size: ");
      scanf("%d", &newPacket);

      if(newPacket < capacity) {
         bucketCapacity = bucketCapacity + newPacket;
         printf("\nbucket capacity currently: %d", bucketCapacity);
         bucketCapacity = bucketCapacity - outputRate;
         printf("\nbucket capacity after output: %d", bucketCapacity);
         timeLimit++;
      }
      else if(newPacket > capacity || (newPacket + bucketCapacity) > capacity) {
         printf("\nnew packet cannot be added to bucket");
```

```
            bucketCapacity = bucketCapacity - outputRate;
            printf("\nbucket capacity after output: %d", bucketCapacity);
        }
        else if(bucketCapacity < 0) {
            bucketCapacity = 0;
            printf("\nbucket capacity after output: %d", bucketCapacity);
            timeLimit++;
            exit(0);
        }
    }
}
```

Program

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

C program:

```
#include <stdio.h>
#include <stdlib.h>
#define capacity 10

void main(){
  int timeLimit = 10, bucketCapacity = 0, outputRate = 5;
  while (timeLimit < 20){
    int newPacket;
    printf("Enter new packet size: ");
    scanf("%d", &newPacket);

    if (newPacket < capacity){
      bucketCapacity += newPacket;
      printf("\nBucket capacity: %d", bucketCapacity);
      bucketCapacity -= outputRate;
      printf("\nBucket capacity after output: %d",
                                    bucketCapacity);
      timeLimit++;
    }

    else if(newPacket > capacity || (newPacket + bucketCapacity)
      > capacity){
      printf("\nNew packet cannot be added to packet");
      bucketCapacity -= outputRate;
      printf("\nBucket capacity after output: %d", bucketCapacity);
    }  timeLimit++;
  }else{
    bucketCapacity = 0;
```

```
            printf("\n Bucket capacity after output: %d",
                bucket(capacity);
            timeLimit++;
            exit(0);
        }
    }
}
```

* Output:

Enter new packet size: 15
bucket capacity: 10
Enter ~~packet~~ new packet size: 27
new packet cannot be added to bucket
bucket capacity after output: 5

Output

```
enter new packet size: 20

bucket capacity currently: 20
bucket capacity after output: 15
enter new packet size: 10

bucket capacity currently: 25
bucket capacity after output: 20
enter new packet size: 2

bucket capacity currently: 22
bucket capacity after output: 17
enter new packet size: 2

bucket capacity currently: 19
bucket capacity after output: 14
enter new packet size: 2

bucket capacity currently: 16
bucket capacity after output: 11
enter new packet size: 1

bucket capacity currently: 12
bucket capacity after output: 7
enter new packet size: 1
```

Experiment 15

Program

ClientTCP.py

```
from socket import *
serverName = &#39;127.0.0.1&#39;
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input(&quot;\nEnter file name: &quot;)
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print (&#39;\nFrom Server:\n&#39;)
print(filecontents)
clientSocket.close()
```

ServerTCP.py

```
from socket import *
serverName=&quot;127.0.0.1&quot;
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print (&quot;The server is ready to receive&quot;)
connectionSocket, addr = serverSocket.accept()
```

```
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,&amp;quot;r&amp;quot;)
l=file.read(1024)
connectionSocket.send(l.encode())
print (&amp;#39;\nSent contents of &amp;#39; + sentence)
file.close()
connectionSocket.close()
```

Program

* using ... to make client sending ... server to send back the contents of the requested file if present.

* Python3 code :

- ServerTCP.py

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

- ClientTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
```

Date
Page

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\n Enter file name :")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("\n From Server: \n")
print(filecontents)
clientSocket.close
```

* Output:

```
>>>
== RESTART: C:/Users/bmca/Desktop/IBM21CS032/ServerTCP.p
The Server is ready to receive
```

Output

```
Enter file name: ServerTCP.py

From Server:

from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file=open(sentence,"r")
    l=file.read(1024)

    connectionSocket.send(l.encode())
    print ('\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()
```

Experiment 16

Program

ClientUDP.py

```
from socket import *
serverName = &amp;quot;127.0.0.1&amp;quot;
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input(&amp;quot;\nEnter file name: &amp;quot;)
clientSocket.sendto(bytes(sentence,&amp;quot;utf-8&amp;quot;),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print (&amp;#39;\nReply from Server:\n&amp;#39;)
print (filecontents.decode(&amp;quot;utf-8&amp;quot;))
# for i in filecontents:
# print(str(i), end = &amp;#39;&amp;#39;)
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind((&amp;quot;127.0.0.1&amp;quot;, serverPort))
print (&amp;quot;The server is ready to receive&amp;quot;)
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
```

```
sentence = sentence.decode(&amp;quot;utf-8&amp;quot;)
file=open(sentence,&amp;quot;r&amp;quot;)
con=file.read(2048)
serverSocket.sendto(bytes(con,&amp;quot;utf-8&amp;quot;),clientAddress)
print (&amp;#39;\nSent contents of &amp;#39;, end = &amp;#39; &amp;#39;)
print (sentence)
# for i in sentence:
# print (str(i), end = &amp;#39;&amp;#39;)
file.close()
```

Program

to make client ... server to send back contents of the requested file if present.

* Python3 Code:

- ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name: ")
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print("\n Reply from server: \n")
print(fileContents.decode("utf-8"))
clientSocket.close()
```

- ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSock.bind(("127.0.0.1", serverPort))
print("The server is ready to recieve")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
```

Date __/__/____
Page _____

```
con = file.read(2048)
serverSocket.sendto(bytes(con, "utf-8"), clientAddrum)
print("\n Sent contents of ", end = "")
print(sentence)
file.close()
```

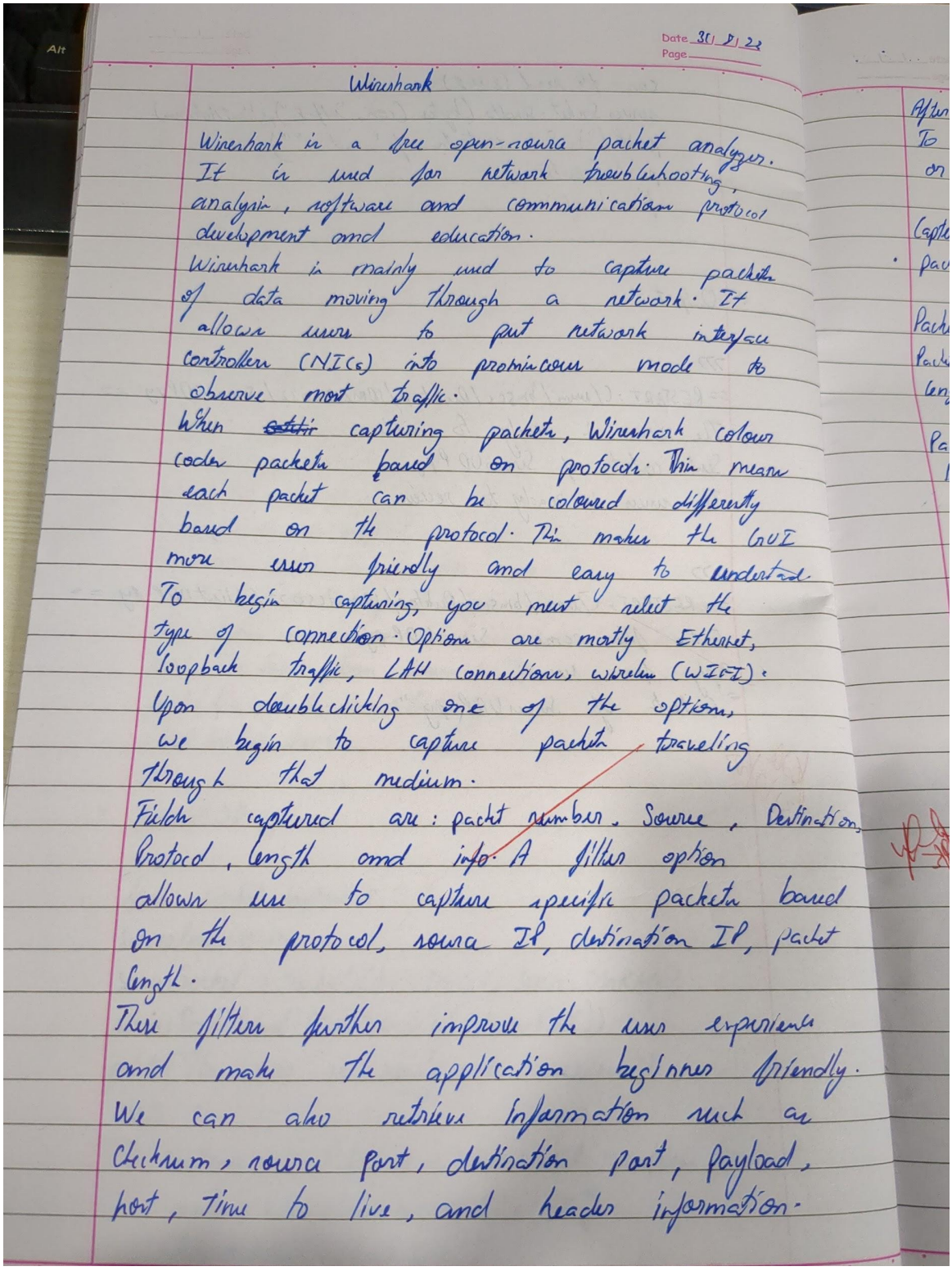
Output:

```
>>>
== RESTART: C:/Users/bmsce/Desktop/1DM21IS032/ServerUDP.py ==
The server is ready to recieve
Sent contents of ServerUDP.py
The server is ready to recieve
```

Output

```
Enter file name:  ServerUDP.py

Reply from Server:

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

while 1:
     print ("The server is ready to receive")
     sentence, clientAddress = serverSocket.recvfrom(2048)
     sentence = sentence.decode("utf-8")
     file=open(sentence,"r")
     l=file.read(2048)

     serverSocket.sendto(bytes(l,"utf-8"),clientAddress)

     print ('\nSent contents of ', end = ' ')
     print (sentence)
    #  for i in sentence:
        #  print (str(i), end = '')
     file.close()
```

Wireshark



Date 30/9/23
Page

Wireshark

Wireshark is a free open-source packet analyzer. It is used for network troubleshooting, analysis, software and communication protocol development and education.

Wireshark is mainly used to capture packets of data moving through a network. It allows users to put network interface controllers (NICs) into promiscuous mode to observe most traffic.

When ~~setting~~ capturing packets, Wireshark colour codes packets based on protocols. This means each packet can be coloured differently based on the protocol. This makes the GUI more user friendly and easy to understand.

To begin capturing, you must select the type of connection. Options are mostly Ethernet, loopback traffic, LAN connections, wireless (WIFI). Upon double clicking one of the options, we begin to capture packets traveling through that medium.

Fields captured are: packet number, Source, Destination, Protocol, length and info. A filter option allows us to capture specific packets based on the protocol, source IP, destination IP, packet length.

These filters further improve the user experience and make the application beginner friendly.

We can also retrieve information such as checksum, source port, destination port, payload, hops, Time to live, and header information.

Date __/__/__
Page

After capturing, we can save the capture. To stop capture we can use either Ctrl + E or the stop button.

Captured data interface contains 3 main sections:
- packet list
- packet details
- packet bytes

Packet list contains all packets captured. It includes Packet no., time, source IP, destination IP, protocol, length in bytes, additional info on the packet.

Packet details is an extension of packet list and it shows more information of the packets. Some of the information is source port and destination port number, checksum and time to live

Packet bytes displays the raw data of the selected packet in hexadecimal view. Hex dump contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset. Upon right clicking, we can display this information in bits