VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPUTER NETWORKS

Submitted by

TANYA D SHETTY (1BM22CS337)

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 Sep
2024-Jan 2025

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**" carried out by Tanya D Shetty(1BM22CS337) who is 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 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Computer Networks Lab - (23CS5PCCON)** work prescribed for the said degree.

Spoorthi D M

Assistant Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

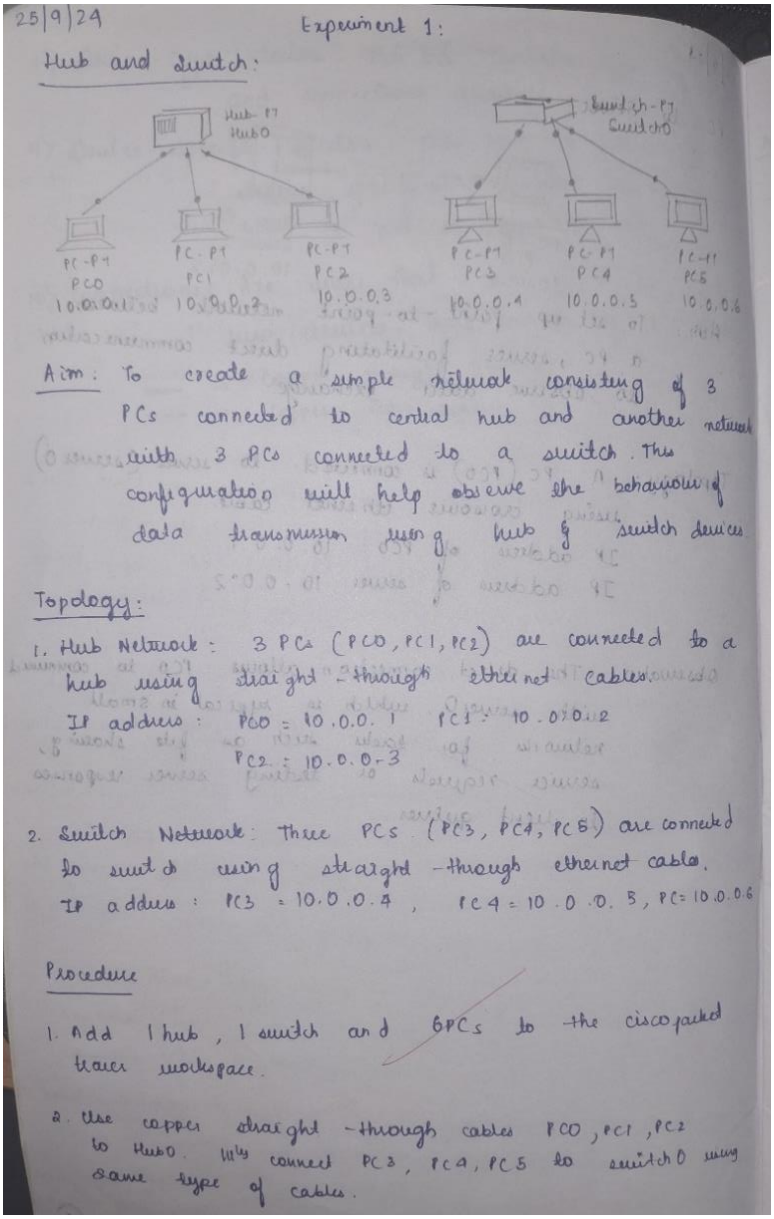
CYCLE 1

INDEX

CYCLE 2

Cycle -1
Experiment 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

Observation:



25/9/24

Experiment 1:

Hub and Switch:

Aim: To create a simple network consisting of 3 PCs connected to central hub and another network with 3 PCs connected to a switch. This configuration will help observe the behaviour of data transmission using hub & switch devices

Topology:

1. Hub Network: 3 PCs (PC0, PC1, PC2) are connected to a hub using straight-through ethernet cables.
   IP address: PC0 = 10.0.0.1 PC1 = 10.0.0.2
   PC2 = 10.0.0.3

2. Switch Network: Three PCs (PC3, PC4, PC5) are connected to switch using straight-through ethernet cable.
   IP address: PC3 = 10.0.0.4, PC4 = 10.0.0.5, PC5 = 10.0.0.6

Procedure

1. Add 1 hub, 1 switch and 6 PCs to the ciscopacket tracer workspace.
2. Use copper straight-through cables PC0, PC1, PC2 to Hub0. Then connect PC3, PC4, PC5 to switch0 using same type of cables.

3. Assign IP addresses to each PC & obtain subnet mask

4. Switch to simulation mode to observe data traffic behaviour when packets are sent between the devices

5. In hub, notice how hub broadcasts packets to all devices, causing potential traffic overload. In the switch network, observe how the switch forwards packets only to the intended recipient, reducing unnecessary traffic.

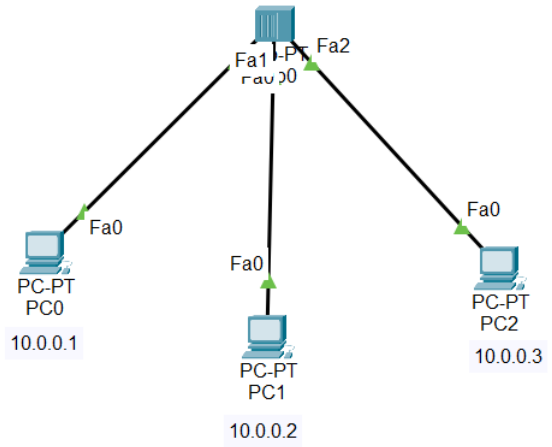
6. The hub broadcasts data to all connected devices leading to more network connected devices, while the switch efficiently sends data only to the correct device, optimizing performance.

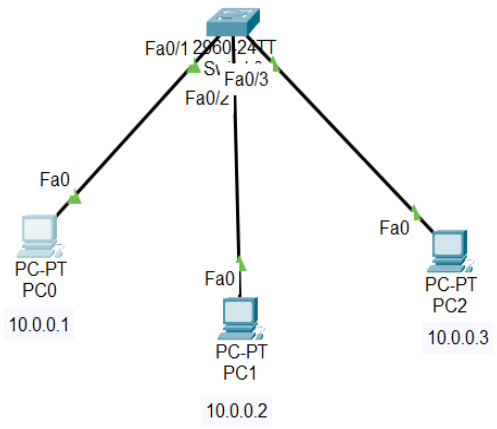
OBSERVATION

1. Hub broadcasts packets to all devices, which may cause unnecessary traffic

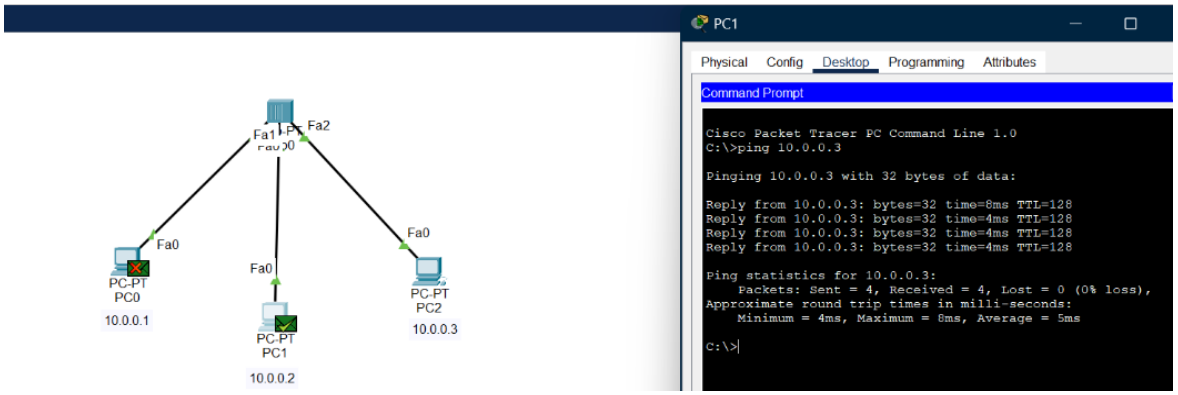
2. Switch forwards packets only to appropriate device by learning MAC addresses, making it more efficient in reducing traffic.

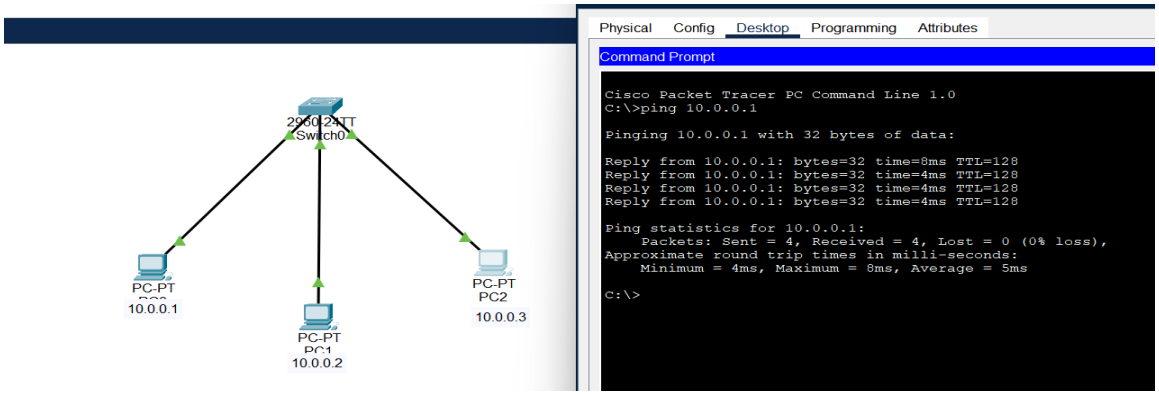
Topology:





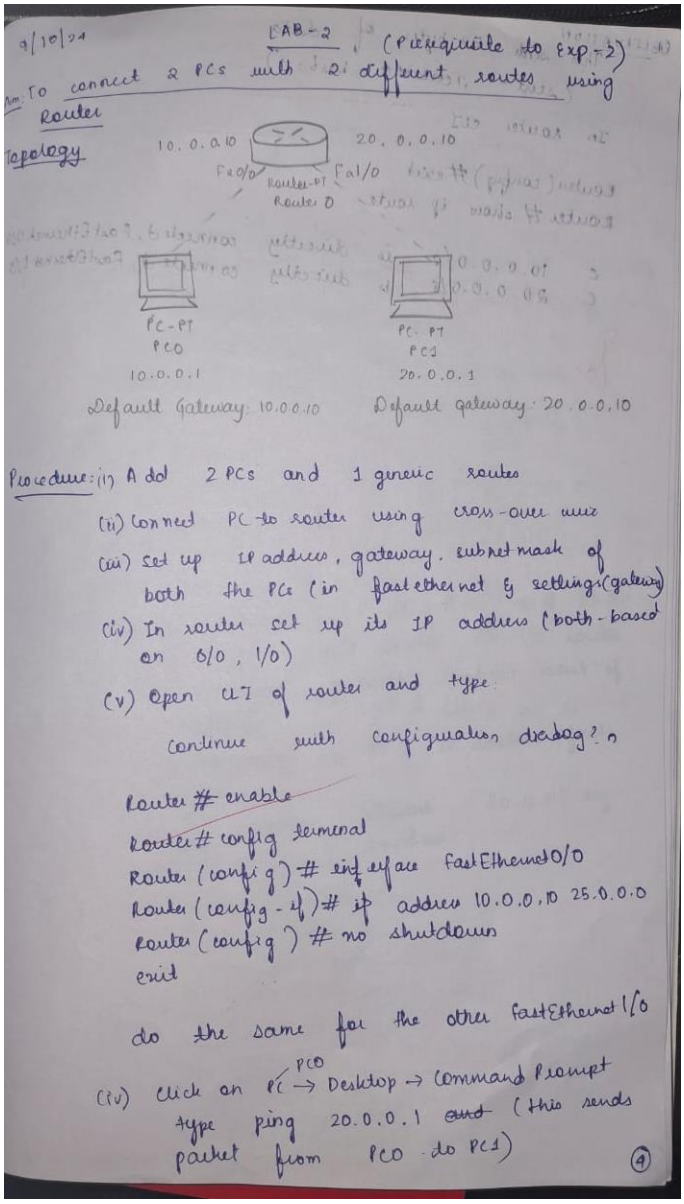
Output:

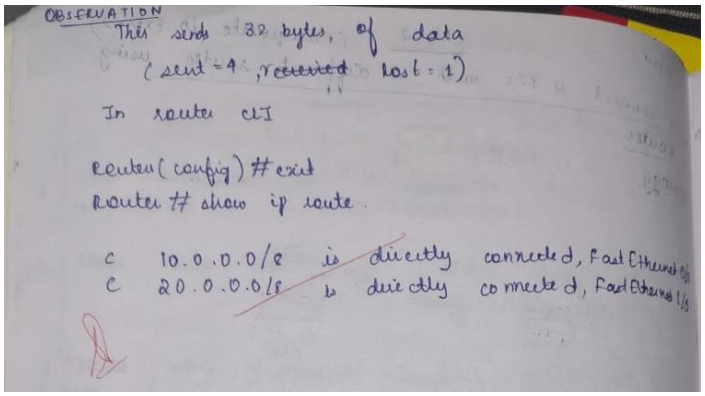

Fa1
Fa2
Fa0
Fa0
Fa0
PC-PT
PC0
10.0.0.1
PC-PT
PC1
10.0.0.2
PC-PT
PC2
10.0.0.3
PC1
Physical
Config
Desktop
Programming
Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=8ms TTL=128
Reply from 10.0.0.3: bytes=32 time=4ms TTL=128
Reply from 10.0.0.3: bytes=32 time=4ms TTL=128
Reply from 10.0.0.3: bytes=32 time=4ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 8ms, Average = 5ms
C:\>


Switch0
PC-PT
10.0.0.1
PC-PT
10.0.0.2
PC-PT
PC2
10.0.0.3
Physical
Config
Desktop
Programming
Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=8ms TTL=128
Reply from 10.0.0.1: bytes=32 time=4ms TTL=128
Reply from 10.0.0.1: bytes=32 time=4ms TTL=128
Reply from 10.0.0.1: bytes=32 time=4ms TTL=128
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 8ms, Average = 5ms
C:\>

Experiment 2:
Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Observation:

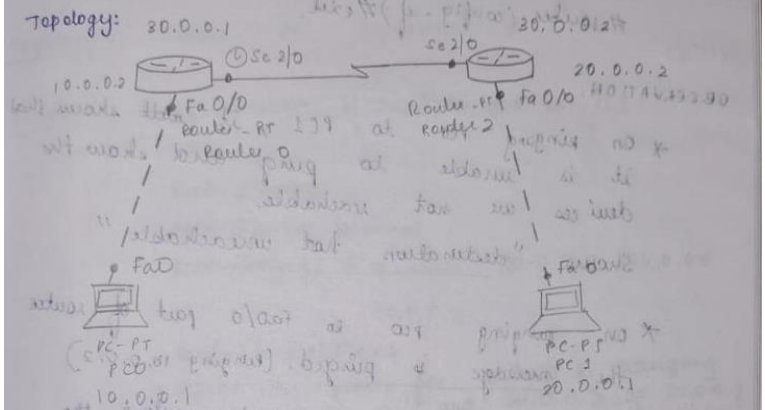




16/10/24 LAB-3 (Experiment 2)

AIM: To connect 2 routers belonging to different Network network to each other.

Topology:



Procedure:

(i) Create a 2 network of 10.0.0.0 and 20.0.0.0 by connecting a end device to the router

(ii) Set up IP address, gateway, subnet mask of both networks

(iii) In router set up the IP address (fast ethernet)

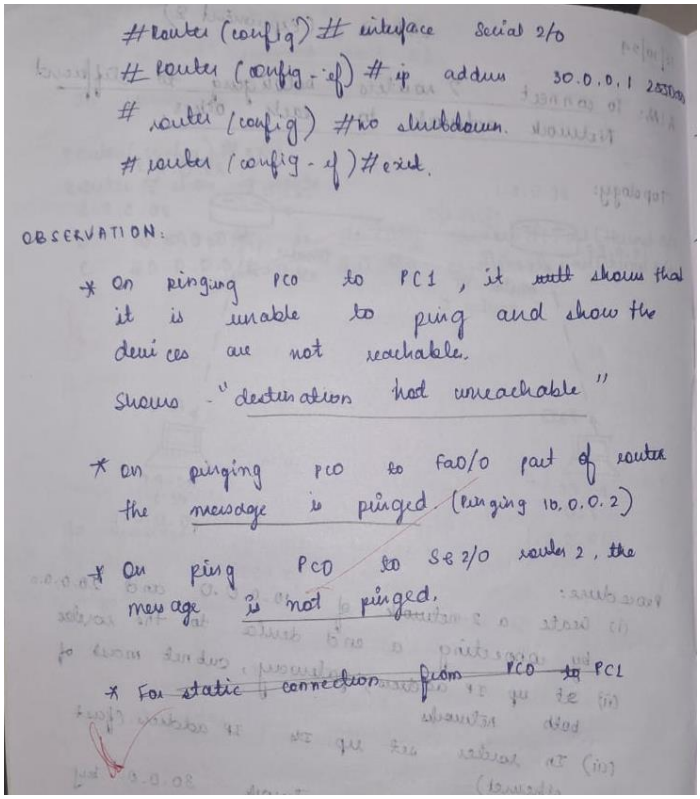
(iv) Set up another network 30.0.0.0 by connecting the 2 routers.

In CLI of router enter

Router #enable
Router # config terminal
Router (config) # interface fastEthernet 0/0
Router (config) # no shutdown
exit

do the same for router 2.

router # enable
router # config terminal



#router (config) # interface Serial 2/0
#router (config-if) # ip addrss 30.0.0.1 255.0.0.0
#router (config) #no shutdown.
#router (config-if) #exit.

OBSERVATION:

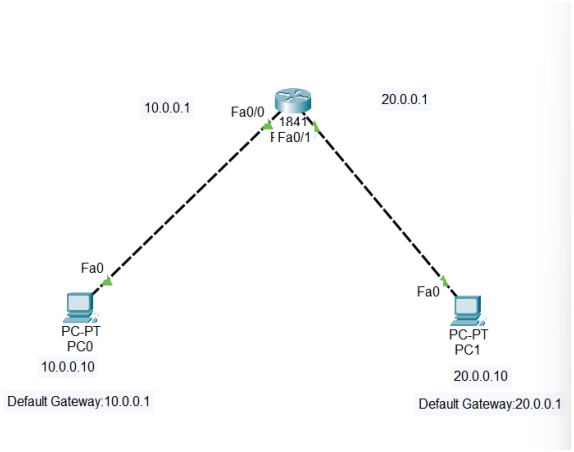
* On ringing PC0 to PC1, it shows that it is unable to ping and show the devices are not reachable.
Shows - "destination host unreachable"

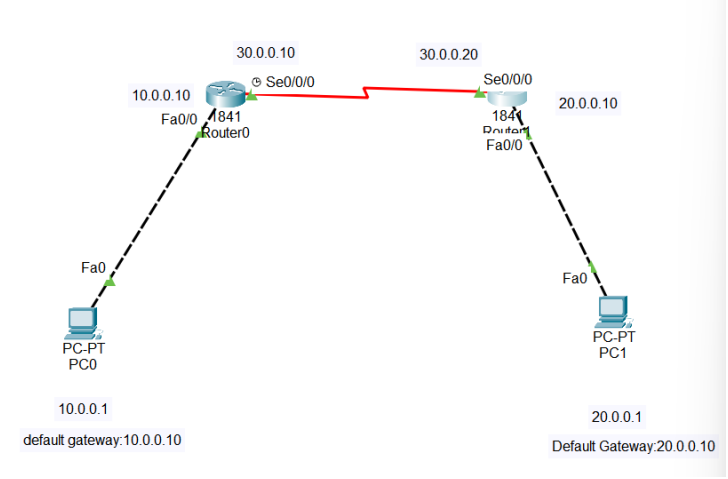
* on ringing PC0 to Fa0/0 part of router the message is pinged (Pinging 10.0.0.2)

* On ping PC0 to Se 2/0 router 2, the message is not pinged.

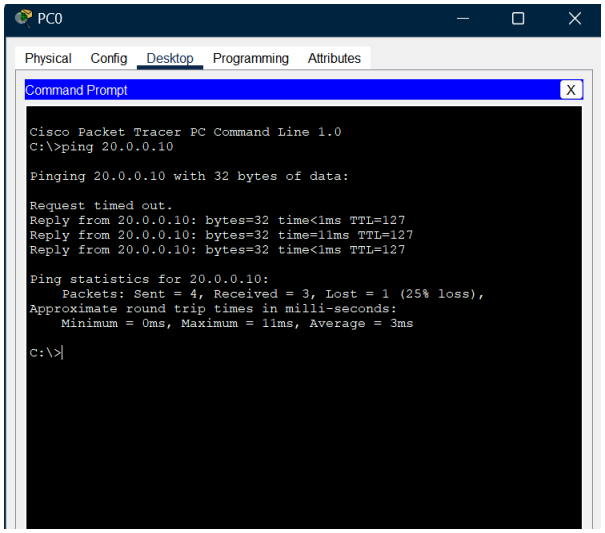
* For static connection from PC0 to PC1

Topology:

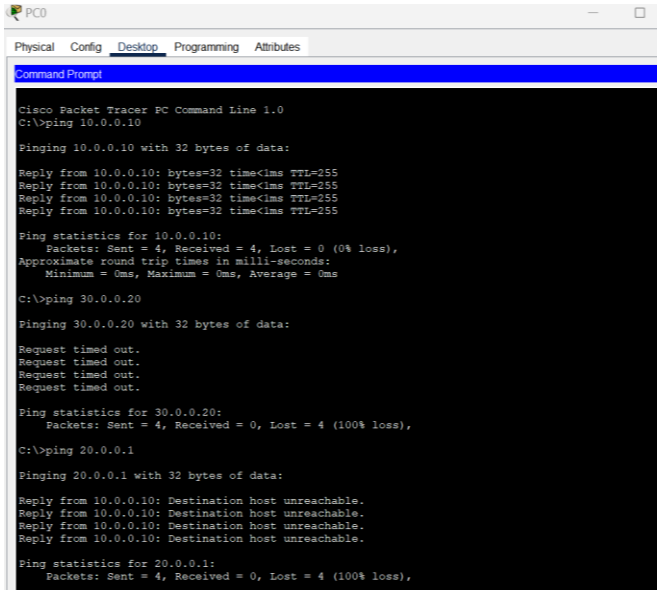

10.0.0.1
Fa0/0
1841
Fa0/1
20.0.0.1
Fa0
PC-PT
PC0
10.0.0.10
Default Gateway:10.0.0.1
Fa0
PC-PT
PC1
20.0.0.10
Default Gateway:20.0.0.1


30.0.0.10
30.0.0.20
10.0.0.10
Se0/0/0
Fa0/0
1841
Router0
Se0/0/0
1841
Fa0/0
20.0.0.10
Fa0
PC-PT
PC0
10.0.0.1
default gateway:10.0.0.10
Fa0
PC-PT
PC1
20.0.0.1
Default Gateway:20.0.0.10

Output:


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.10
Pinging 20.0.0.10 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.10: bytes=32 time<1ms TTL=127
Reply from 20.0.0.10: bytes=32 time=11ms TTL=127
Reply from 20.0.0.10: bytes=32 time<1ms TTL=127
Ping statistics for 20.0.0.10:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 11ms, Average = 3ms
C:\>

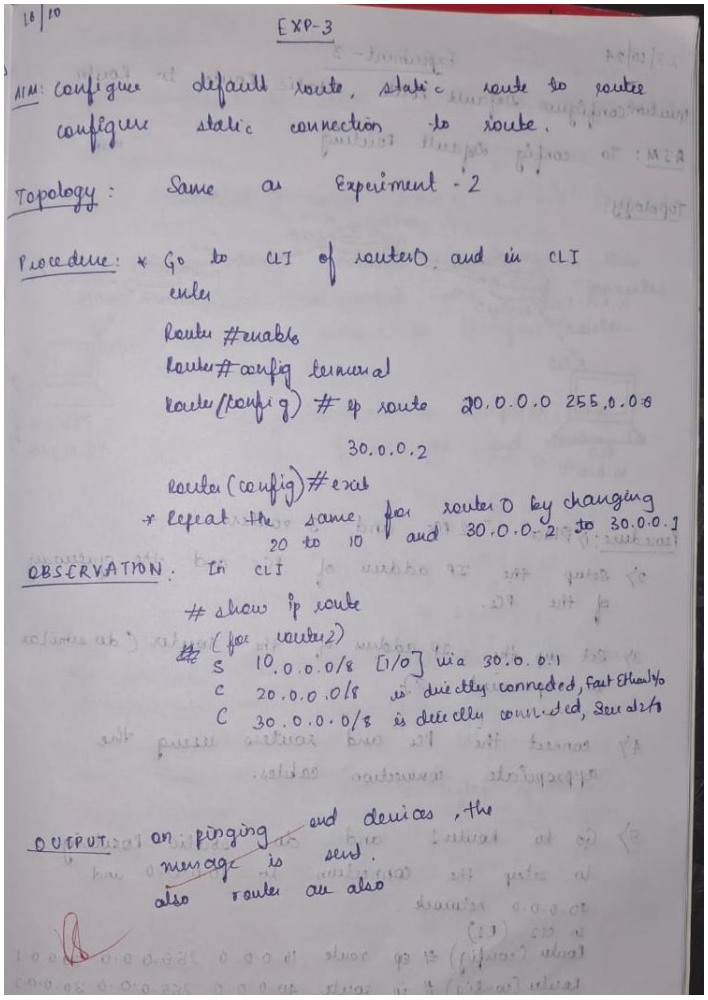

Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Press RETURN to get started.
Router>enable
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - 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 30.0.0.0/8 is directly connected, Serial0/0/0


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.10
Pinging 10.0.0.10 with 32 bytes of data:
Reply from 10.0.0.10: bytes=32 time<1ms TTL=255
Reply from 10.0.0.10: bytes=32 time<1ms TTL=255
Reply from 10.0.0.10: bytes=32 time<1ms TTL=255
Reply from 10.0.0.10: bytes=32 time<1ms TTL=255
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>ping 30.0.0.20
Pinging 30.0.0.20 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 30.0.0.20:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

Experiment 3:

Configure default route, static route to the Router

Observation:



10/10

EXP-3

AIM: configure default route, static route to router configure static connection to route.

Topology: Same as Experiment - 2

Procedure: * Go to CLI of router0 and in CLI enter

Router #enable
Router# config terminal
Router(config) # ip route 20.0.0.0 255.0.0.0 30.0.0.2
Router(config)#exit

* Repeat the same for router0 by changing 20 to 10 and 30.0.0.2 to 30.0.0.1

OBSERVATION: In CLI

show ip route
(for router2)
S 10.0.0.0/8 [1/0] via 30.0.0.1
C 20.0.0.0/8 is directly connected, FastEthernet0
C 30.0.0.0/8 is directly connected, Serial2/1

OUTPUT: on pinging end devices, the message is sent.
also router are also

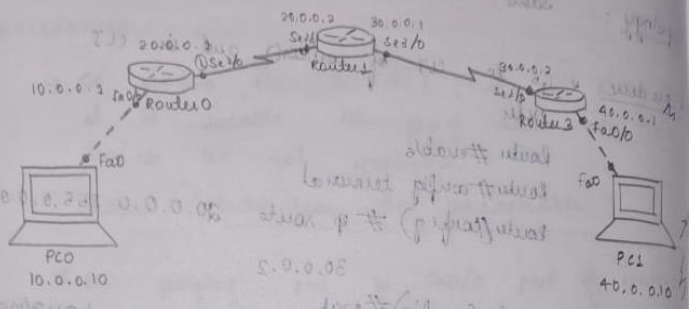
23/10/24

Experiment – 3

Question: Configure Default Route, Static Route to Router

AIM: To config Default Routing

Topology:



Procedure: 1) Place 2 PCs and 3 routers.

2) Setup the IP address of PCs and the gateway of the PCs.

3) Set up the IP address of the router (do similar to experiment 2)

4) connect the PCs and routers using the appropriate connection cables.

5) Go to Router 1 and do static routing to setup the connection to 10.0.0.0 and 40.0.0.0 network.

In CLI (R1)

```
Router (config) # ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router (config) # ip route 40.0.0.0 255.0.0.0 30.0.0.2
```

6) In Router 0 and 2 go to their CLI and enter the following commands.

CLI (Router0)

Router (config) # ip route 0.0.0.0 0.0.0.0 20.0.0.2

CLI (Router2)

Router (config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1

This step is called default routing, in this if any network or packet other than connected network will be passed to specified router.

7) Once the configuration is complete we can now ping from the end device to other.

PC0

ping 40.0.0.10

OBSERVATION:

1) on pinging from one end device to the other,

ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes = 32 time = 2ms TTL = 253
Reply from 40.0.0.10: bytes = 32 time: 2ms TTL = 253
Reply from 40.0.0.10 bytes = 32 time: 8ms TTL = 253
Reply from 40.0.0.10 bytes = 32 time = 10ms TTL = 253

ping statistics for 40.0.0.1:
Packets: sent = 4, Recieved = 4 lost = 0 (0% loss)

2) In Router0

Router # show ip route

Gateway of last route is 20.0.0.2 to network 0.0.0.0

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

(Similar output for Router2)

In Router1

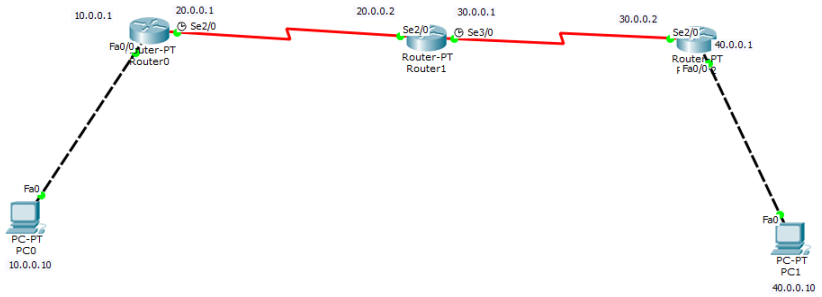
Router # show ip route

Gateway of last resort is not set

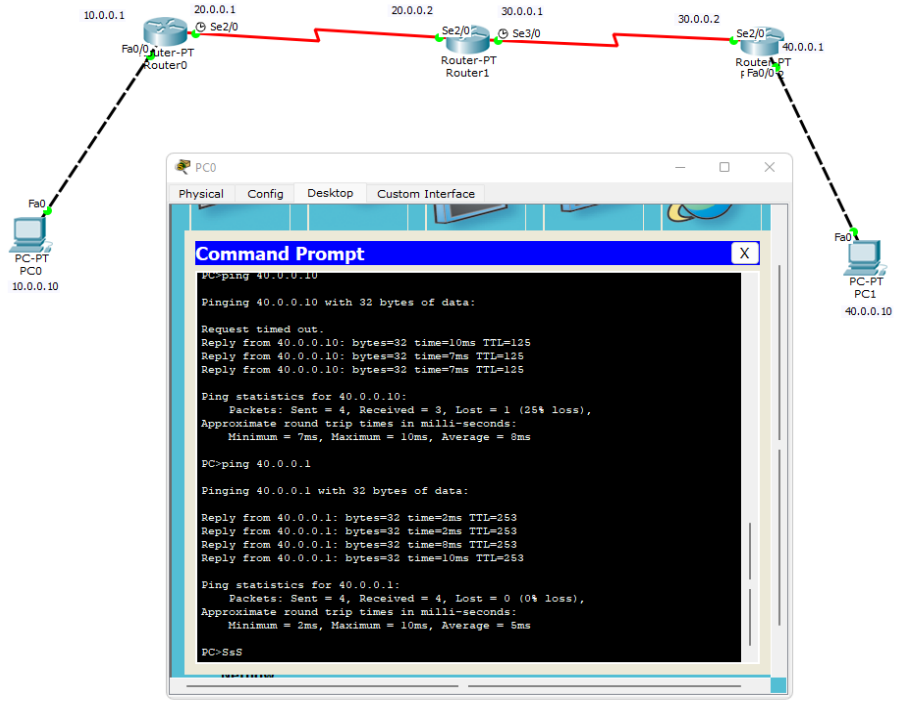
S 10.0.0.0/8 [1/0] via
C 20.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.0/8 is directly connected, Serial3/0
S 40.0.0.0/8 [1/0] via

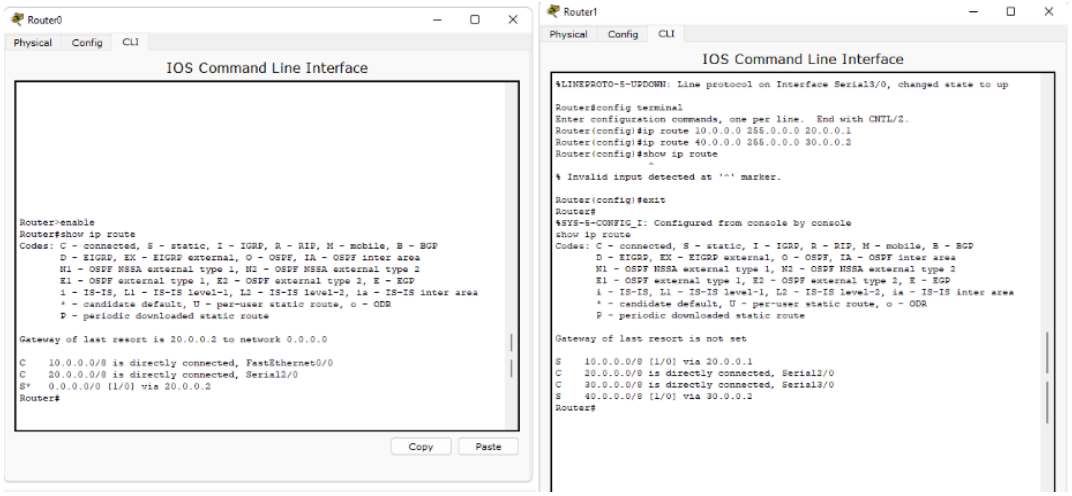
Through this experiment, we learnt on how to connect 3 or more networks by the concept of static and default routing and we also sent messages from end device to other.

Topology:



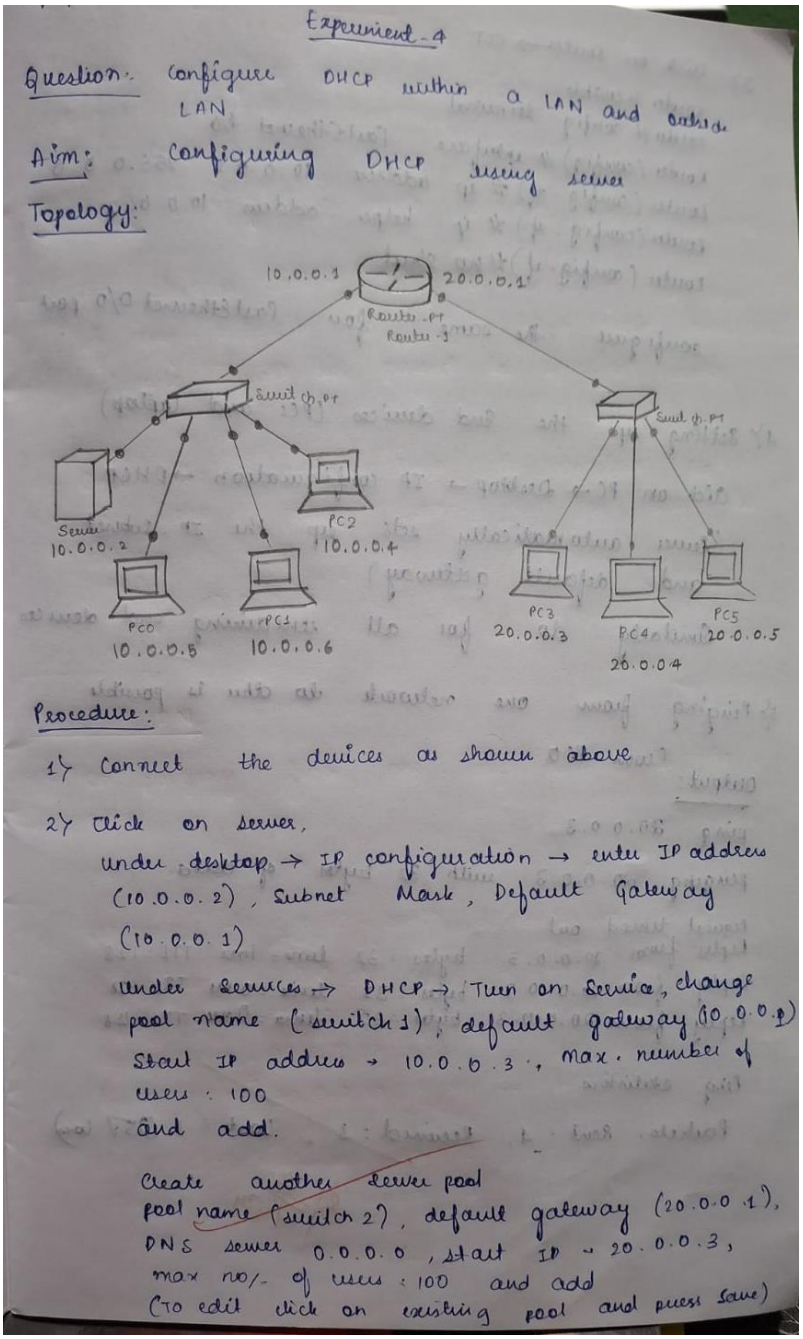
Output:

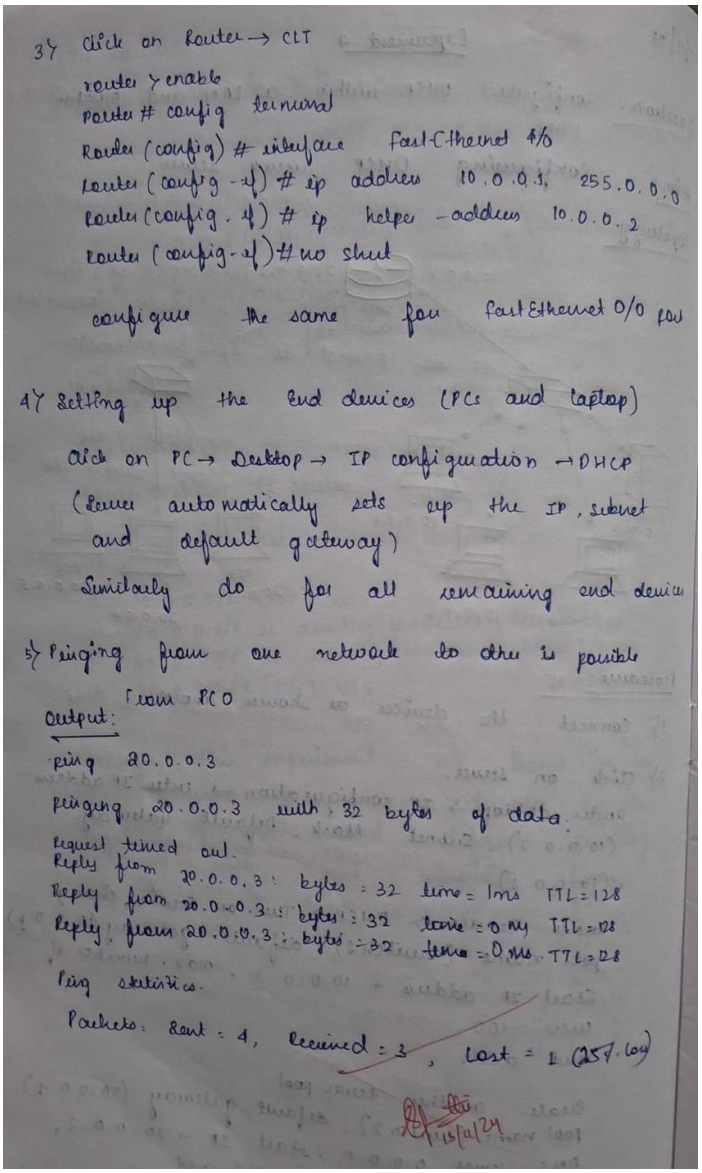



Router0
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external 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 20.0.0.2 to network 0.0.0.0
C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 20.0.0.0/8 is directly connected, Serial2/0
S* 0.0.0.0/0 [1/0] via 20.0.0.2
Router#
Copy
Paste
Router1
Physical
Config
CLI
IOS Command Line Interface
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router(config)#ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router(config)#show ip route
% Invalid input detected at '^' marker.
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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 20.0.0.1
C 20.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.0/8 is directly connected, Serial3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2
Router#

Experiment 4:
Configure DHCP within a LAN and outside LAN.

Observation:





3) Click on Router → CLT
router > enable
router # config terminal
Router (config) # interface Fast-Ethernet 4/5
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

configure the same for Fast Ethernet 0/0 port

4) Setting up the End devices (PCs and Laptop)

Click on PC → Desktop → IP configuration → DHCP
(Server automatically sets up the IP, subnet and default gateway)
Similarly do for all remaining end devices

5) Pinging from one network to other is possible

From PC0

Output:

ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data.

Request timed out.
Reply from 20.0.0.3: bytes = 32 time = 1ms TTL = 128
Reply from 20.0.0.3: bytes = 32 time = 0 ms TTL = 128
Reply from 20.0.0.3: bytes = 32 time = 0 ms TTL = 128

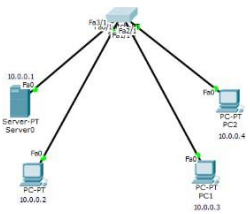
Ping statistics.

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

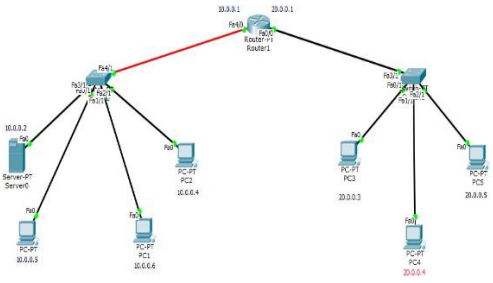
15/8/24

Topology:

(within Lan)

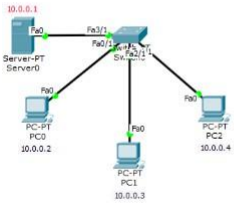


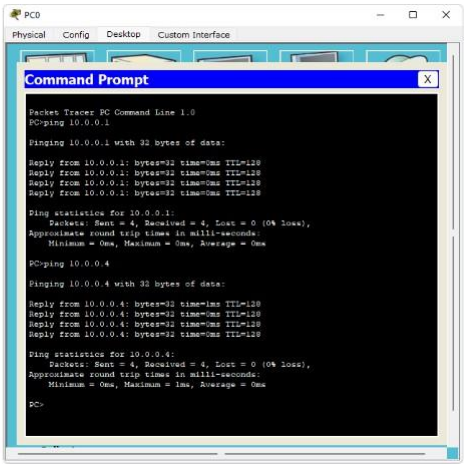
(outside Lan)



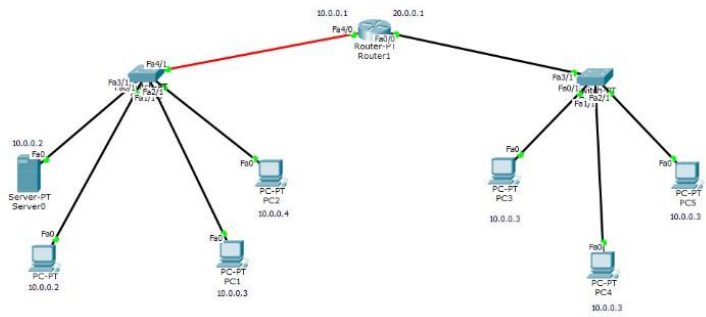
Output:

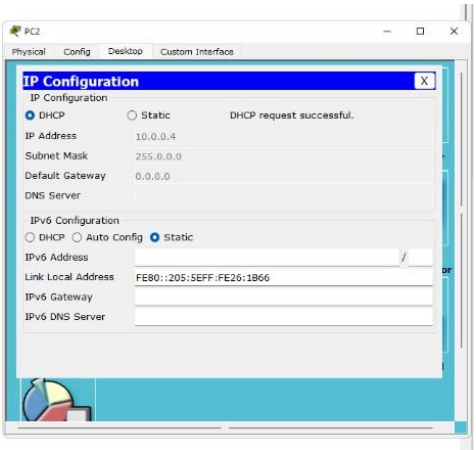
(within Lan)

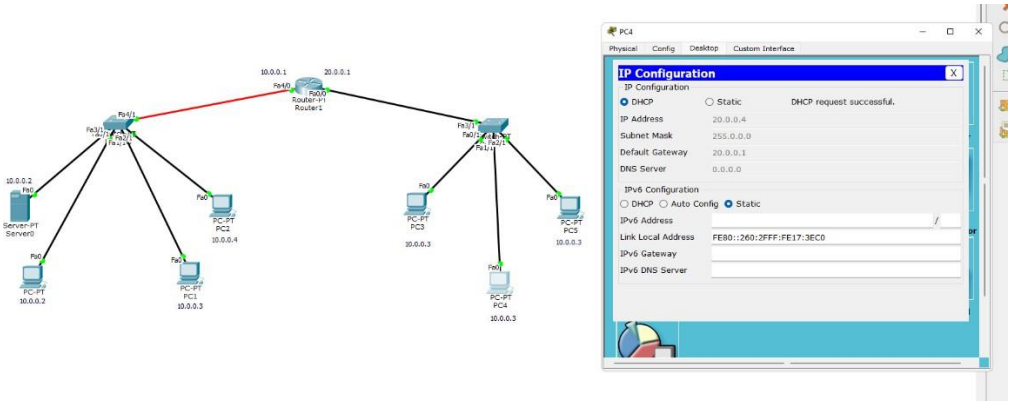


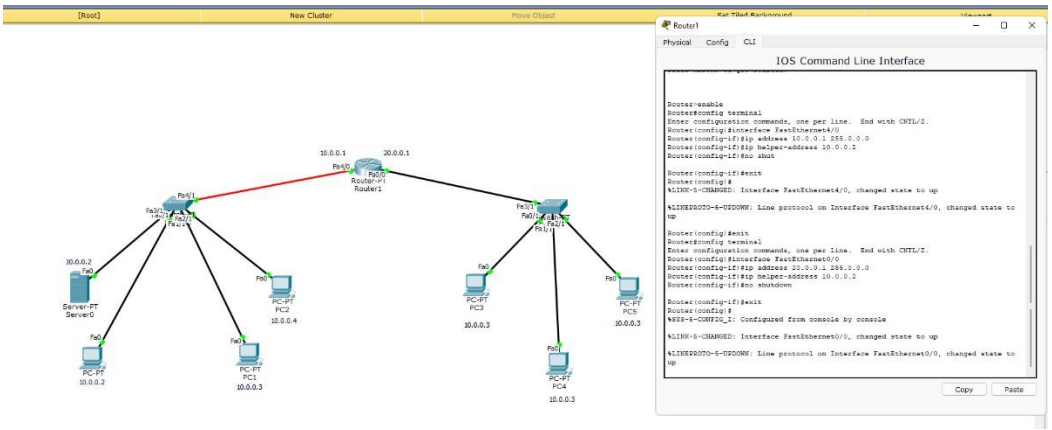


(outside Lan)



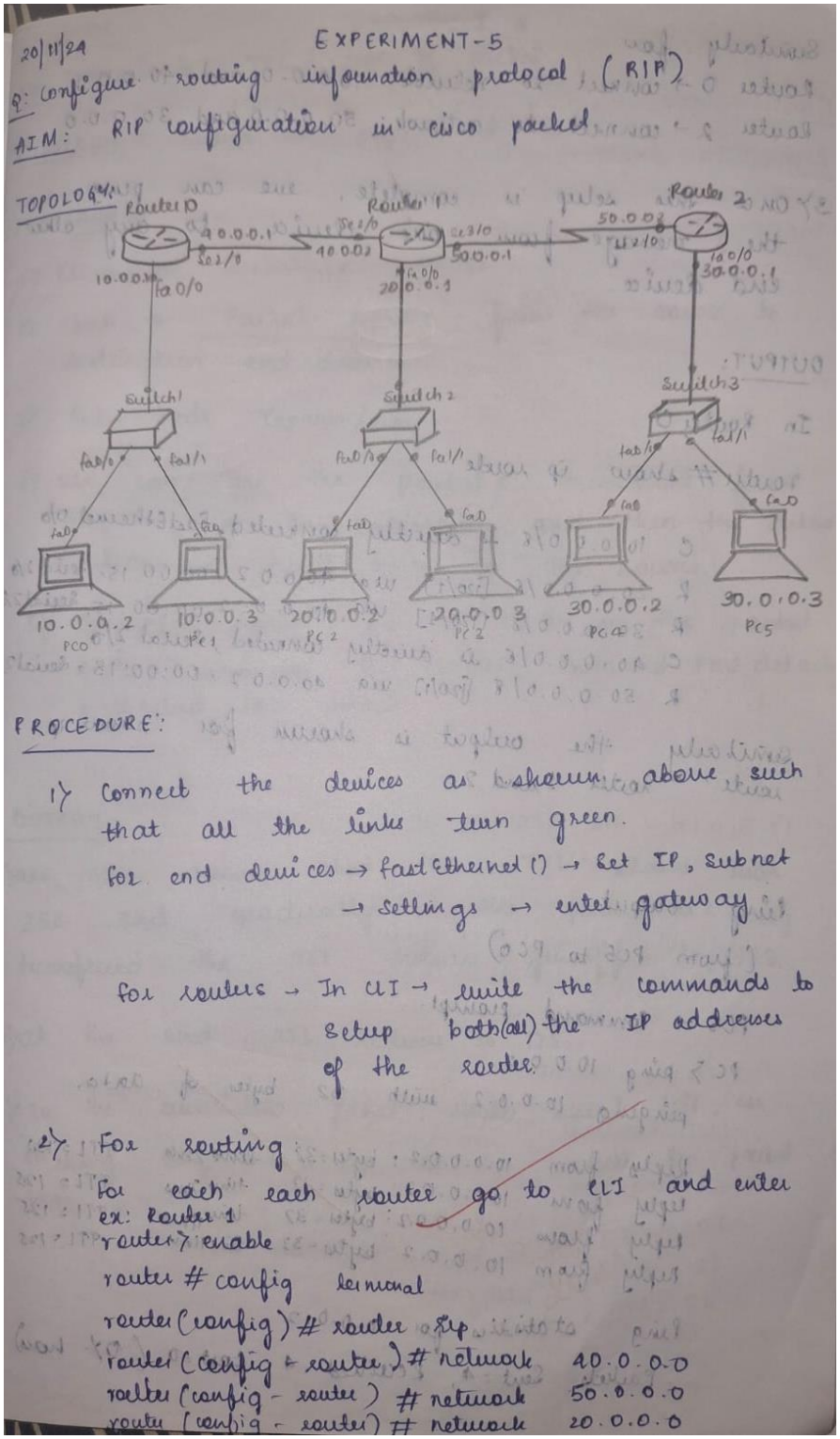



PC4
Physical
Config
Desktop
Custom Interface
IP Configuration
IP Configuration
DHCP
Static
DHCP request successful.
IP Address
20.0.0.4
Subnet Mask
255.0.0.0
Default Gateway
20.0.0.1
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::260:2FFF:FE17:3EC0
IPv6 Gateway
IPv6 DNS Server
10.0.0.1
20.0.0.1
Router-PT
Router1
Server-PT
Server0
PC-PT
PC1
PC2
PC3
PC4
PC5
10.0.0.2
10.0.0.3
10.0.0.4


[Root]
New Cluster
Move Object
Router1
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet4/0
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
Router(config-if)#exit
Router(config)#
%LINK-5-CHANGED: Interface FastEthernet4/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet4/0, changed state to up
Router(config)#exit
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
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 shutdown
Router(config-if)#exit
Router(config)#
%SYS-5-CONFIG_I: Configured from console by console
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
Copy
Paste
10.0.0.1
20.0.0.1
Router-PT
Router1
Server-PT
Server0
PC-PT
PC1
PC2
PC3
PC4
PC5
10.0.0.2
10.0.0.3
10.0.0.4

Experiment 5:
Configure RIP routing Protocol in Routers

Observation:



Similarly for
Router 0 → connect to network 10.0.0.0 and 40.0.0.0
Router 2 → connect to network 50.0.0.0 and 30.0.0.0

3) Once this setup is complete, we can ping the message from one device to any other end device.

OUTPUT:

In Router 0

router # show ip route

C 10.0.0.0/8 is directly connected, FastEthernet 0/0
R 20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:15, Serial 2/0
R 30.0.0.0/8 [120/1] via 40.0.0.2, 00:00:15, Serial 2/0
C 40.0.0.0/8 is directly connected, Serial 2/0
R 50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:15, Serial 2/0

Similarly the output is shown for show ip route, router 1 and 2

Ping output
(from PC5 to PC0)

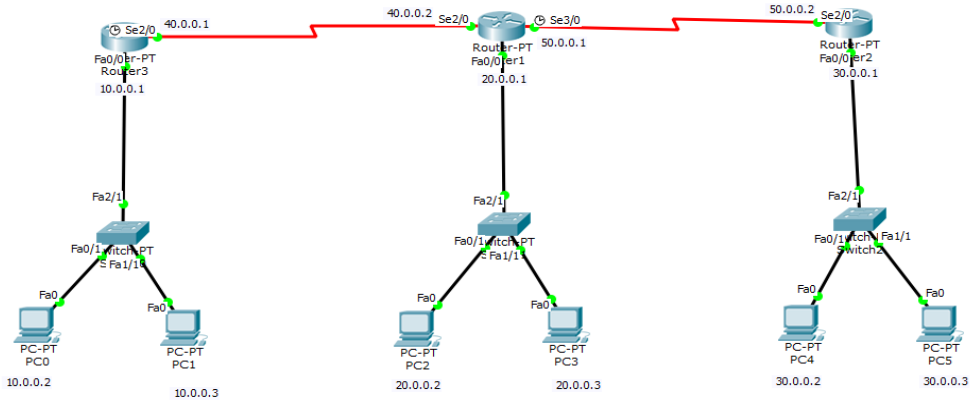
PC5 → command prompt.

PC > ping 10.0.0.2
pinging 10.0.0.2 with 32 bytes of data,

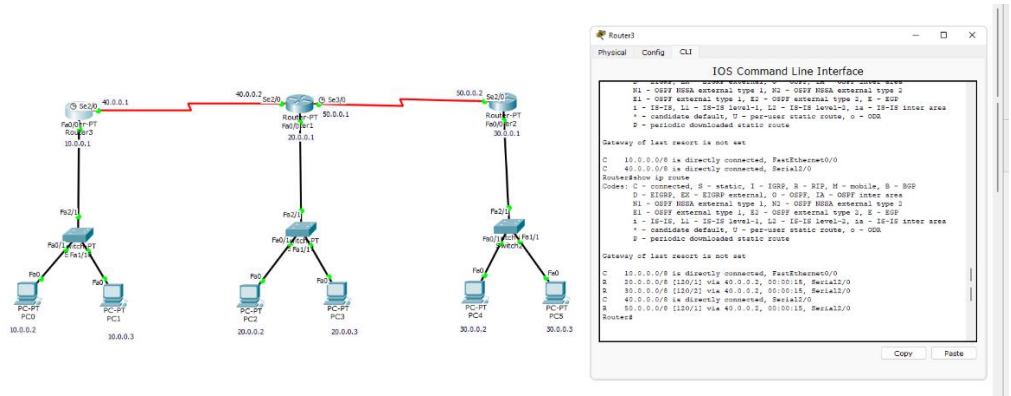
Reply from 10.0.0.0.2: bytes=32 time 2ms TTL=125
Reply from 10.0.0.0.2: bytes=32 time 7ms TTL=125
Reply from 10.0.0.0.2: bytes=32 time 1ms TTL=125
Reply from 10.0.0.2 bytes=32 time 11ms TTL=125

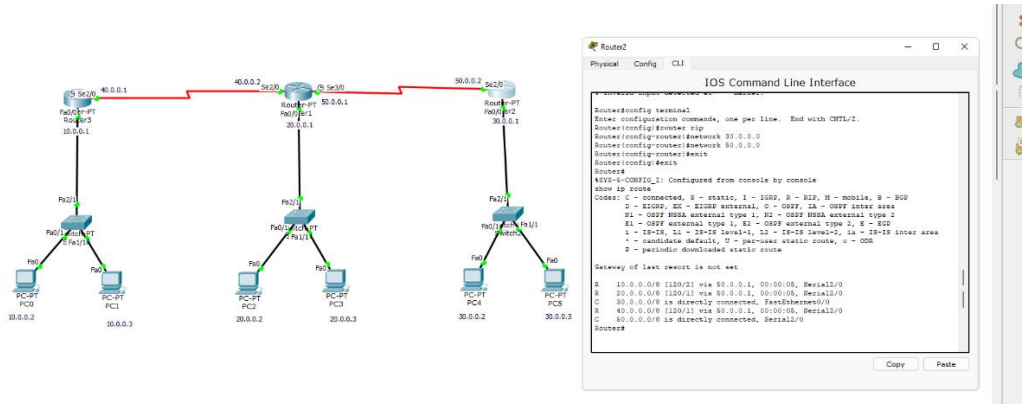
Ping statistics for 10.0.0.2
Packets Sent = 4, Received = 4, Lost = 0 (0% loss)

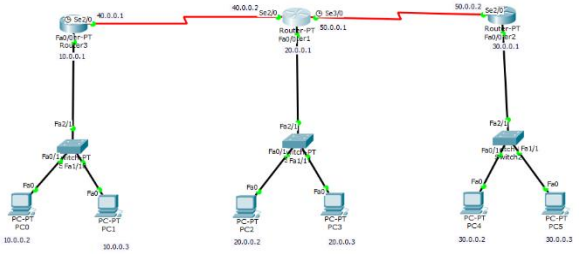
Topology:

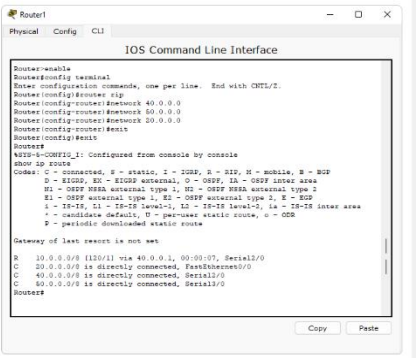


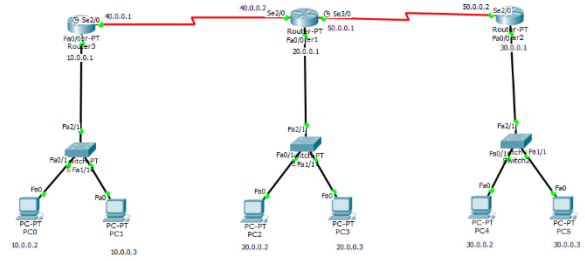
Output:

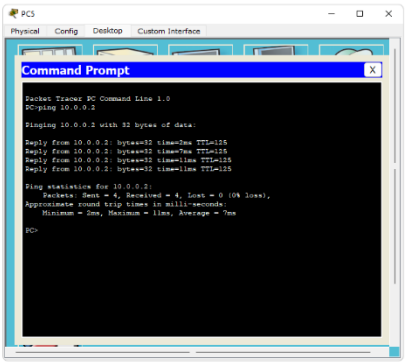





40.0.0.1
40.0.0.2
50.0.0.2
10.0.0.1
20.0.0.1
30.0.0.1
50.0.0.1
10.0.0.2
10.0.0.3
20.0.0.2
20.0.0.3
30.0.0.2
30.0.0.3

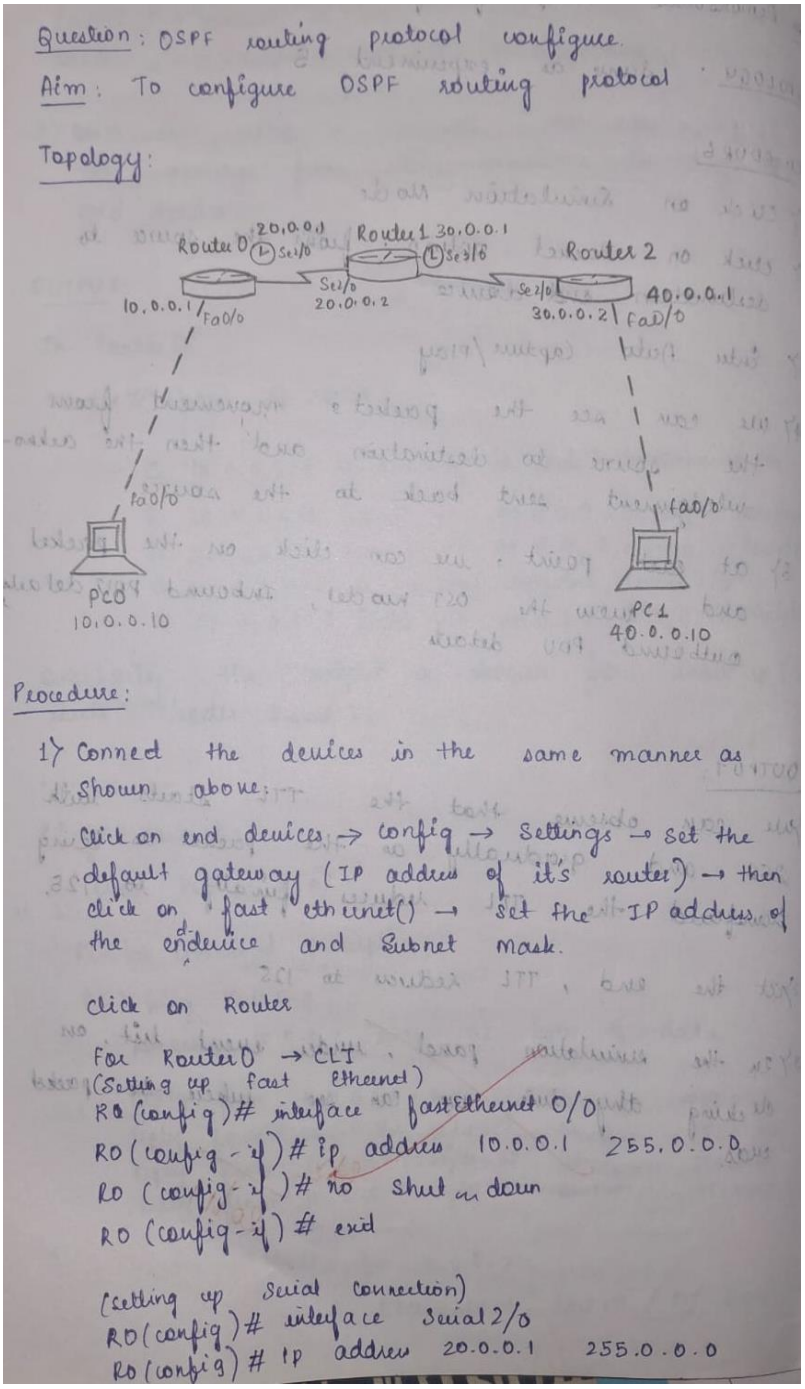

Router1
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 40.0.0.0
Router(config-router)#network 50.0.0.0
Router(config-router)#network 20.0.0.0
Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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
R 10.0.0.0/8 [120/1] via 40.0.0.1, 00:00:07, Serial2/0
C 20.0.0.0/8 is directly connected, FastEthernet0/0
C 40.0.0.0/8 is directly connected, Serial2/0
C 50.0.0.0/8 is directly connected, Serial3/0
Router#
Copy
Paste


40.0.0.1
40.0.0.2
50.0.0.2
10.0.0.1
20.0.0.1
30.0.0.1
50.0.0.1
10.0.0.2
10.0.0.3
20.0.0.2
20.0.0.3
30.0.0.2
30.0.0.3


PC5
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
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=3ms TTL=125
Reply from 10.0.0.2: bytes=32 time=7ms TTL=125
Reply from 10.0.0.2: bytes=32 time=11ms TTL=125
Reply from 10.0.0.2: bytes=32 time=11ms TTL=125
Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 11ms, Average = 7ms
PC>

Experiment 6:
Configure OSPF routing protocol

Observation:



R0 (config-if) # encapsulation ppp
R0 (config-if) # clock rate 64000
R0 (config-if) # no shut down
R0 (config-if) # exit.

Similarly we set up the IP's of R1 and R2 while the setup of fast Ethernet remains same, the setting up of Serial connections has 2 extra lines (encapsulation ppp, clock rate 64000)
clock rate 6400 must only be written if the Serialy connected port show a (D) symbol.
Here, we write clock rate command for
R0 Serial 2/0, R1 Serial 3/0.

After this step all the connections must have turned green.

2) To enable IP routing by configuring ospf routing protocol in all routers.

Router R0 → CLI
R0(config) # router ospf 1
R0(config-router) # router-id 1.1.1.1
R0 (config-router) # network 10.0.0.0 0.255.255.255 area 3
R0 (config-router) # network 20.0.0.0 0.255.255.255 area 0
R0 (config-router) # exit

Router R1 - CLI
R1 (config) # router ospf 1
R1 (config-router) # router-id 2.2.2.2
R1 (config-router) # network 20.0.0.0 0.255.255.255 area 1
R1 (config-router) # network 30.0.0.0 0.255.255.255 area 0
R1 (config-router) # exit

In router R2 →CLI

```
R2(config)# router ospf 1
R2(config-router)# router-id 3.3.3.3
R2(config-router)#network 30.0.0.0 0.255.255.255 area 0
R2(config-router)# network 40.0.0.0 0.255.255.255 area 2
R2(config-router)#exit
```

3) Once the setting up of networking area is done we configure loopback address to router.

```
R0(config-if)# interface loopback 0
R0(config-if)# ip add 172.16.1.252 255.255.0.0
R0(config-if)# no shutdown

R1(config-if)# interface loopback 0
R1(config-if)# ip add 172.16.1.253 255.255.0.0
R1(config-if)# no shutdown

R2(config-if)# interface loopback 0
R2(config-if)# ip add 172.16.1.254 255.255.0.0
R2(config-if)# no shutdown
```

4) On checking routing table of R2 using show ip route we can see that R2 doesn't know about area 3.
Gateway of last resort is not set

```
O IA 20.0.0.0/8 [110/125] via 30.0.0.1, Serial 3/0
C   40.0.0.0/8 is directly connected, FastEthernet 0/0
C   30.0.0.0/8 is directly connected, Serial 2/0
```

Since R2 doesn't know about area 3, we have to create a virtual link between R0 and R1.

5) Creating virtual link between R1, R0

In Router R0

```
R0(config) # router ospf 1
R0(config-router) # area 1 virtual-link 2.2.2.2
R0(config-router) # exit
```

In Router R1

```
R1(config) #router ospf 1
R1(config-router) # area 1 virtual-link 1.1.1.1
R1(config-router) # exit
```

6) Now, check routing table of R2

Once all these steps are completed, the message can now be pinged from 1 end-device to other.

OBSERVATION

In R2

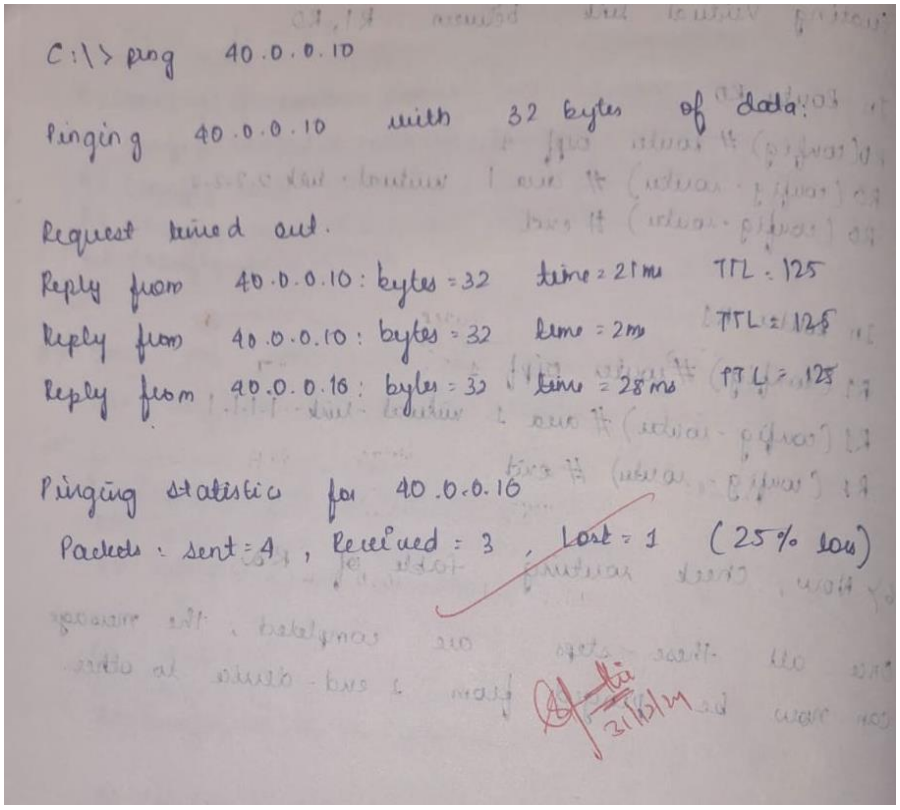
```
Router # show ip route
O IA   20.0.0.0/8 [110/128] via 30.0.0.1, 00:57:25, Serial 2/0
C      40.0.0.0/8 is directly connected, FastEthernet 0/0
O IA   10.0.0.0/8 [110/129] via 30.0.0.1 00:57:05, Serial 2/0
C      30.0.0.0/8 is directly connected, Serial 2/0
C      172.16.0.0/16 is directly connected, Loopback0
```

Similarly the output is shown for Router 0 and 1

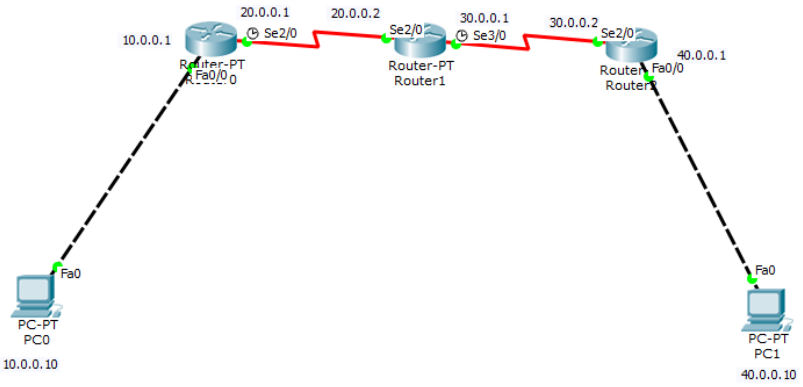
Ping output

(from PC0 to PC1)

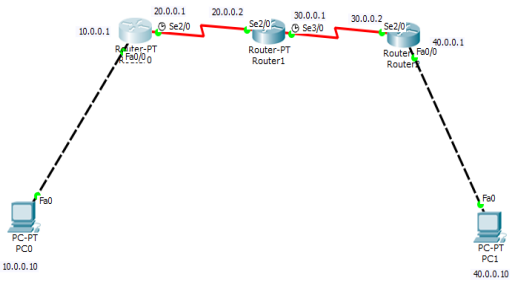
PC0 → Command prompt

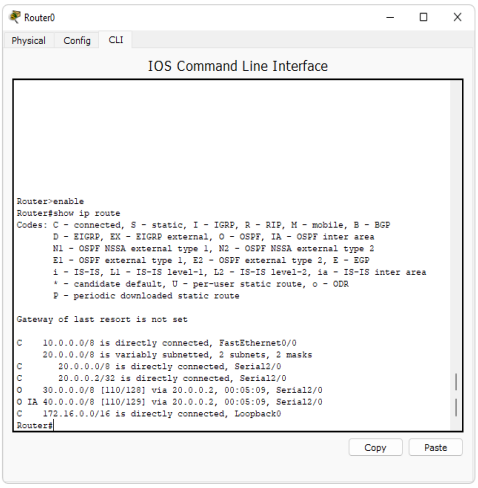

C:\> ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.10: bytes=32 time=21ms TTL=125
Reply from 40.0.0.10: bytes=32 time=2ms TTL=125
Reply from 40.0.0.10: bytes=32 time=28ms TTL=125
Pinging statistics for 40.0.0.10
Packets: Sent=4, Received=3, Lost=1 (25% loss)

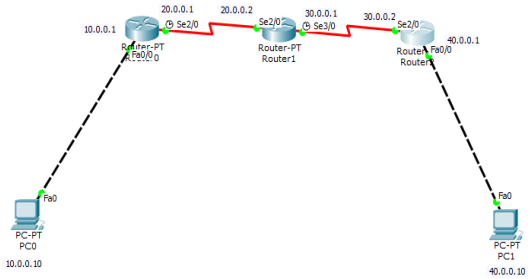
Topology:

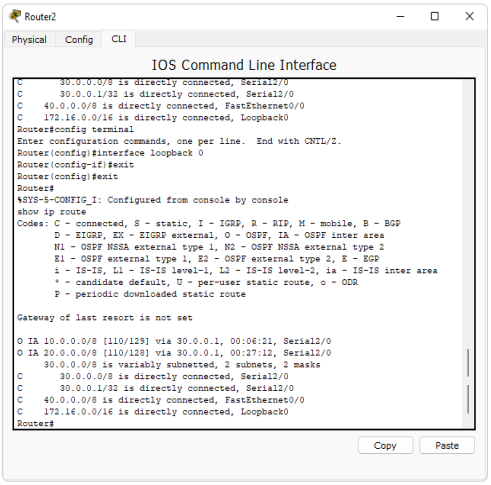

20.0.0.1
Se2/0
20.0.0.2
Se2/0
30.0.0.1
Se3/0
30.0.0.2
Se2/0
10.0.0.1
Router-PT
Router1
Fa0/0
Router-PT
Router1
Router
Router1
Fa0/0
40.0.0.1
Fa0
PC-PT
PC0
10.0.0.10
Fa0
PC-PT
PC1
40.0.0.10

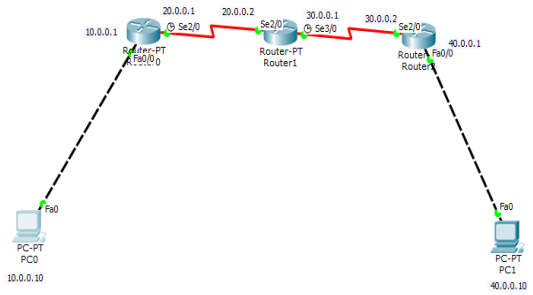
Output:

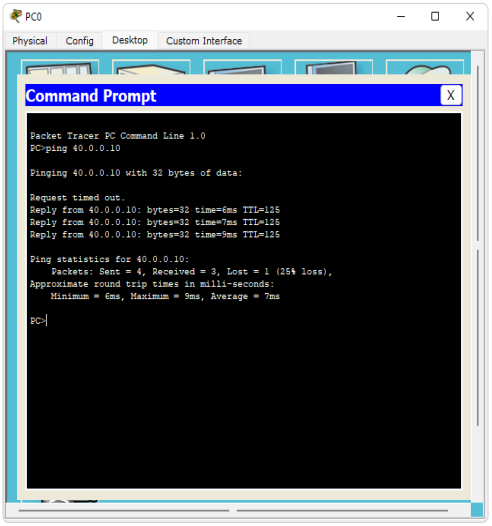

20.0.0.1
20.0.0.2
30.0.0.1
30.0.0.2
10.0.0.1
Se2/0
Se3/0
Fa0/0
40.0.0.1
Router-PT
Router1
Router-PT
Router2
Router-PT
Router0
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10

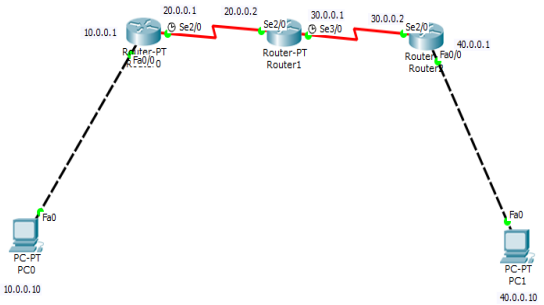

Router0
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external 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
O 30.0.0.0/8 [110/128] via 20.0.0.2, 00:05:09, Serial2/0
O IA 40.0.0.0/8 [110/129] via 20.0.0.2, 00:05:09, Serial2/0
C 172.16.0.0/16 is directly connected, Loopback0
Router#
Copy
Paste

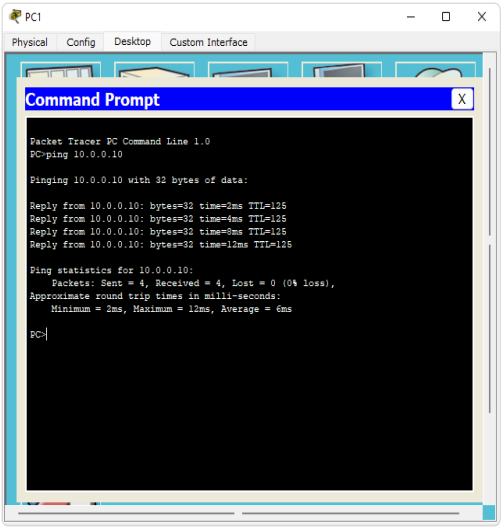

20.0.0.1
20.0.0.2
30.0.0.1
30.0.0.2
10.0.0.1
Se2/0
Se3/0
Fa0/0
40.0.0.1
Router-PT
Router1
Router-PT
Router2
Router-PT
Router0
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10


Router2
Physical
Config
CLI
IOS Command Line Interface
C 30.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.1/32 is directly connected, Serial2/0
C 40.0.0.0/8 is directly connected, FastEthernet0/0
C 172.16.0.0/16 is directly connected, Loopback0
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface loopback 0
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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
O IA 10.0.0.0/8 [110/129] via 30.0.0.1, 00:06:21, Serial2/0
O IA 20.0.0.0/8 [110/128] via 30.0.0.1, 00:27:12, Serial2/0
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 30.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.1/32 is directly connected, Serial2/0
C 40.0.0.0/8 is directly connected, FastEthernet0/0
C 172.16.0.0/16 is directly connected, Loopback0
Router#
Copy
Paste


20.0.0.1
20.0.0.2
30.0.0.1
30.0.0.2
10.0.0.1
Se2/0
Se3/0
Fa0/0
40.0.0.1
Router-PT
Router1
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=9ms TTL=125
Ping statistics for 40.0.0.10:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 9ms, Average = 7ms
PC>

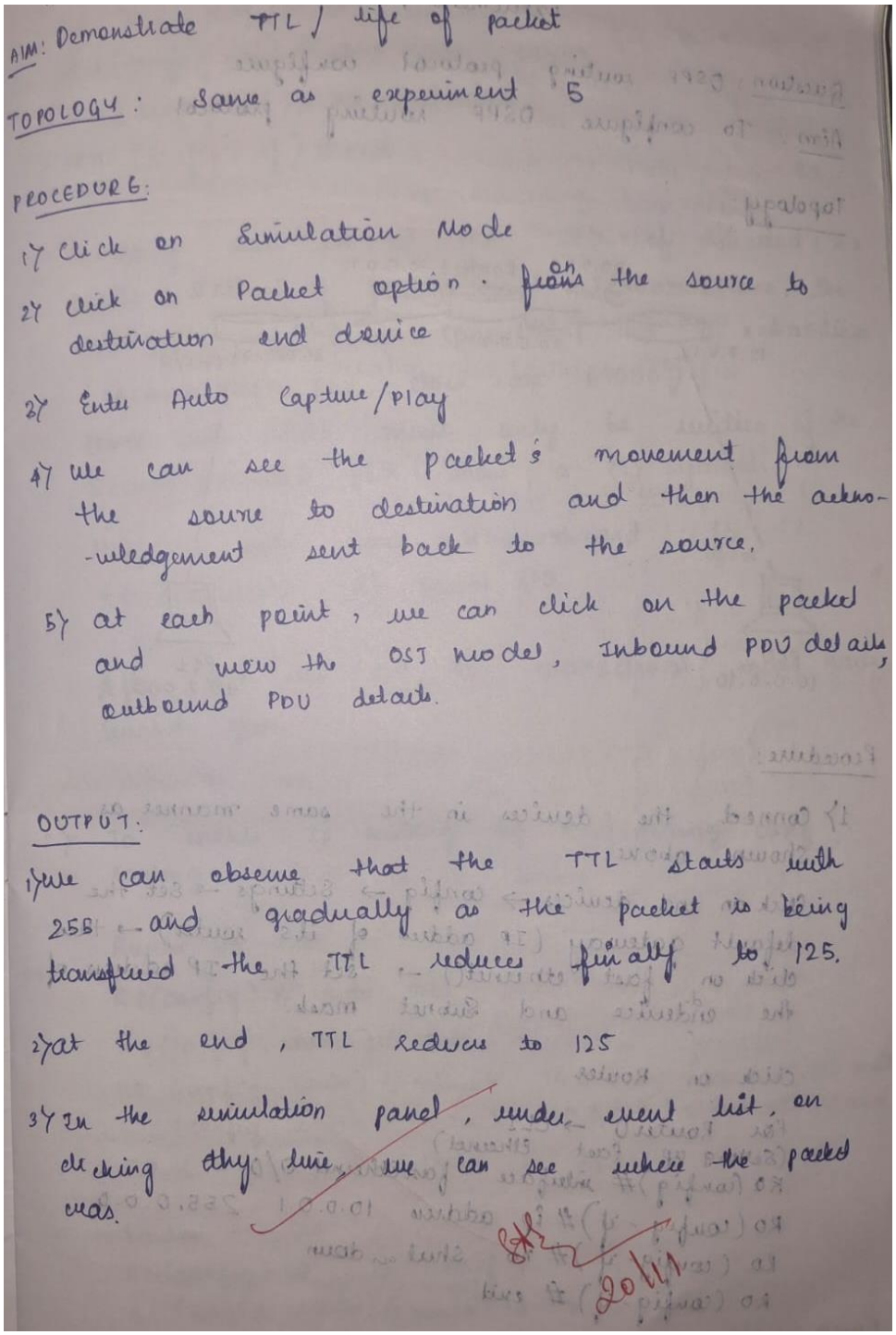

20.0.0.1
20.0.0.2
30.0.0.1
30.0.0.2
10.0.0.1
Se2/0
Se3/0
Fa0/0
40.0.0.1
Router-PT
Router1
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10


PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.10
Pinging 10.0.0.10 with 32 bytes of data:
Reply from 10.0.0.10: bytes=32 time=2ms TTL=125
Reply from 10.0.0.10: bytes=32 time=4ms TTL=125
Reply from 10.0.0.10: bytes=32 time=8ms TTL=125
Reply from 10.0.0.10: bytes=32 time=12ms TTL=125
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 12ms, Average = 6ms
PC>

Experiment 7:

Demonstrate the TTL/ Life of a Packet

Observation :



AIM: Demonstrate TTL / life of packet

TOPOLOGY: Same as experiment 5

PROCEDURE:

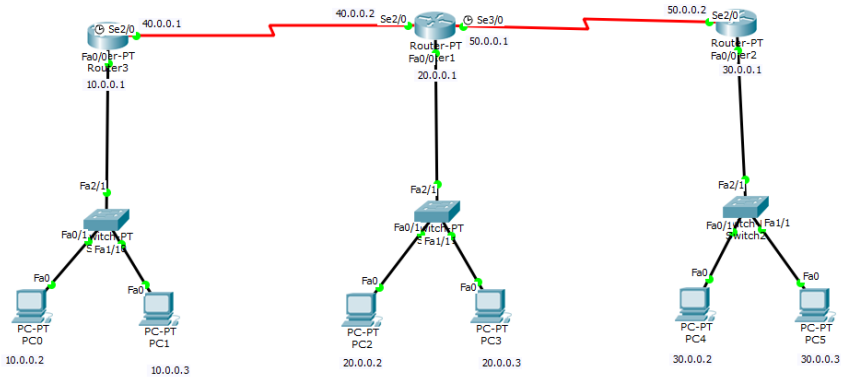
1) Click on Simulation Mode
2) Click on Packet option. From the source to destination end device
3) Enter Auto Capture/Play
4) We can see the packet's movement from the source to destination and then the acknowledgement sent back to the source.
5) At each point, we can click on the packet and view the OSI model, Inbound PDU details, outbound PDU details.

OUTPUT:

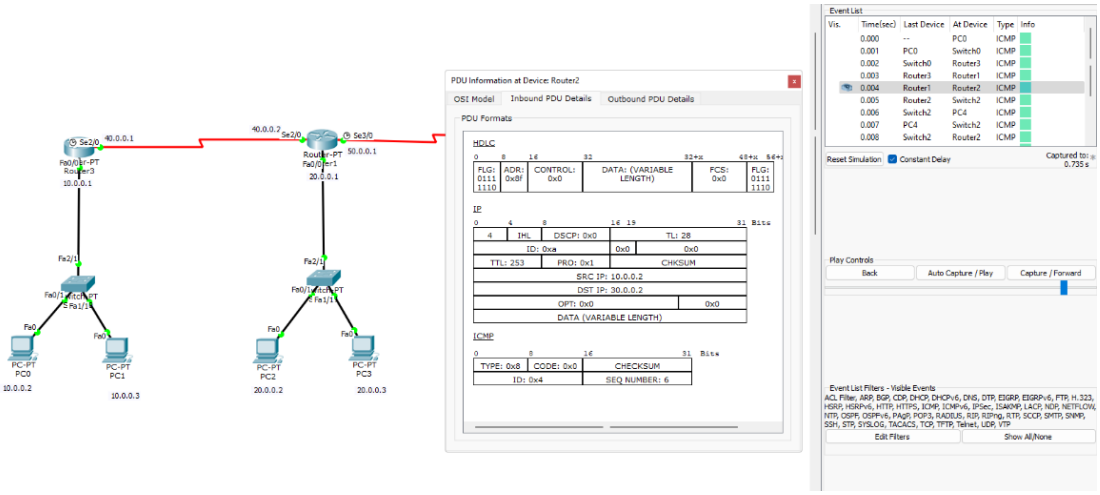
1) We can observe that the TTL starts with 255 and gradually as the packet is being transferred the TTL reduces finally to 125.
2) At the end, TTL reduces to 125
3) In the simulation panel, under event list, on clicking any line, we can see where the packet was.

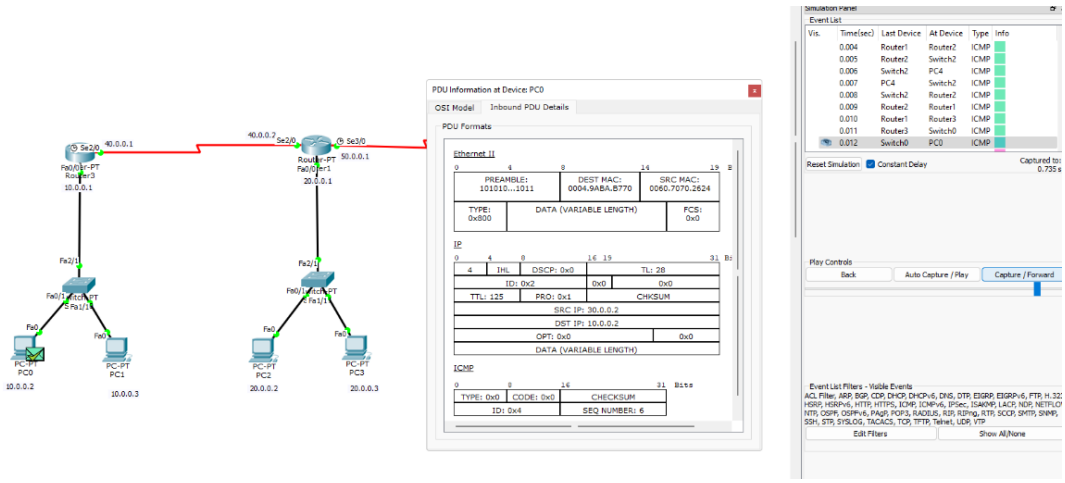
8/12/2021

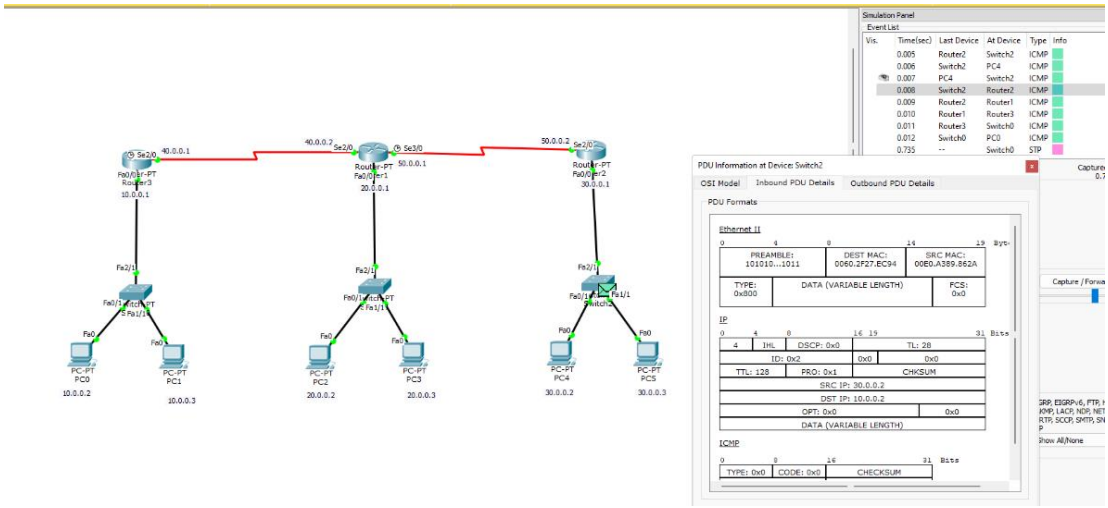
Topology:



Output:



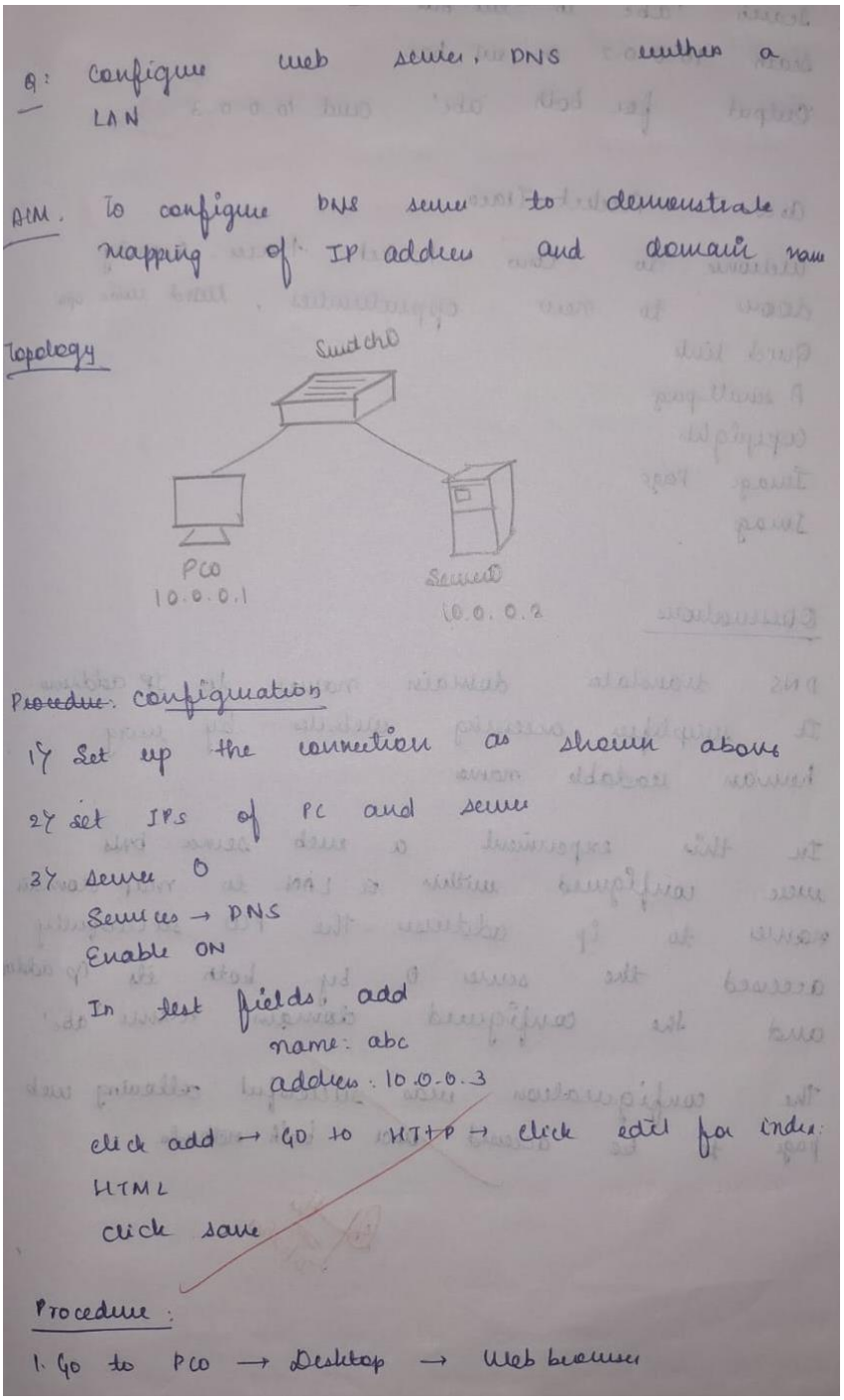

Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type Info
0.004 Router1 Router2 ICMP
0.005 Router2 Switch2 ICMP
0.006 Switch2 PC4 ICMP
0.007 PC4 Switch2 ICMP
0.008 Switch2 Router2 ICMP
0.009 Router2 Router1 ICMP
0.010 Router1 Router3 ICMP
0.011 Router3 Switch0 ICMP
0.012 Switch0 PC0 ICMP
Reset Simulation
Constant Delay
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
Edit Filters
Show All/None
PDU Information at Device: PC0
OSI Model
Inbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 0004.9ABA.B770
SRC MAC: 0060.7070.2624
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
4 IHL DSCP: 0x0 TL: 28
ID: 0x2 0x0 0x0
TTL: 125 PRO: 0x1 CHKSUM
SRC IP: 30.0.0.2
DST IP: 10.0.0.2
OPT: 0x0 0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: 0x0 CODE: 0x0 CHECKSUM
ID: 0x4 SEQ NUMBER: 6
Router-PT
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
10.0.0.2
10.0.0.3
20.0.0.2
20.0.0.3


Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type Info
0.005 Router2 Switch2 ICMP
0.006 Switch2 PC4 ICMP
0.007 PC4 Switch2 ICMP
0.008 Switch2 Router2 ICMP
0.009 Router2 Router1 ICMP
0.010 Router1 Router3 ICMP
0.011 Router3 Switch0 ICMP
0.012 Switch0 PC0 ICMP
0.735 -- Switch0 STP
PDU Information at Device: Switch2
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 0060.2F27.EC94
SRC MAC: 00E0.A3B9.862A
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
4 IHL DSCP: 0x0 TL: 28
ID: 0x2 0x0 0x0
TTL: 128 PRO: 0x1 CHKSUM
SRC IP: 30.0.0.2
DST IP: 10.0.0.2
OPT: 0x0 0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: 0x0 CODE: 0x0 CHECKSUM
Router-PT
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
PC-PT PC4
PC-PT PC5
10.0.0.2
10.0.0.3
20.0.0.2
20.0.0.3
30.0.0.2
30.0.0.3

Experiment 8:

Configure Web Server, DNS within a LAN

Observation:



2. Search 'abc' in url bar
3. Search 10.0.0.2 in url bar
Output for both 'abc' and 10.0.0.3

Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening door to new opportunities. Mind wide open
Quick Links
A small page
Copyrights
Image Page
Image

Observations

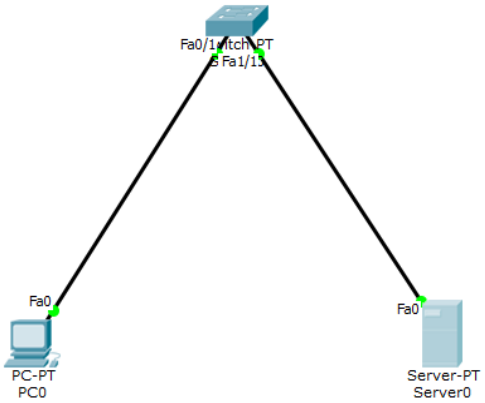
DNS translates domain names to IP addresses. It simplifies accessing websites by using human readable names.

In this experiment a web server DNS were configured within a LAN to map domain names to ip addresses. The PC0 successfully accessed the server 0 by both its ip address and the configured domain name 'abc'.

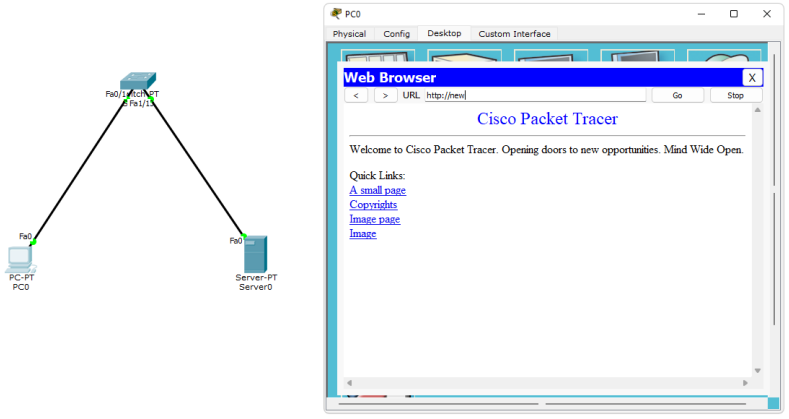
The configuration was successful allowing web page to be accessed via both methods.

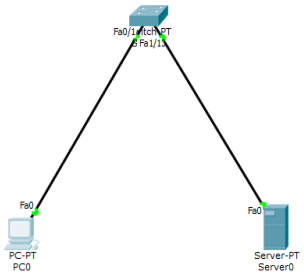
31/5/24

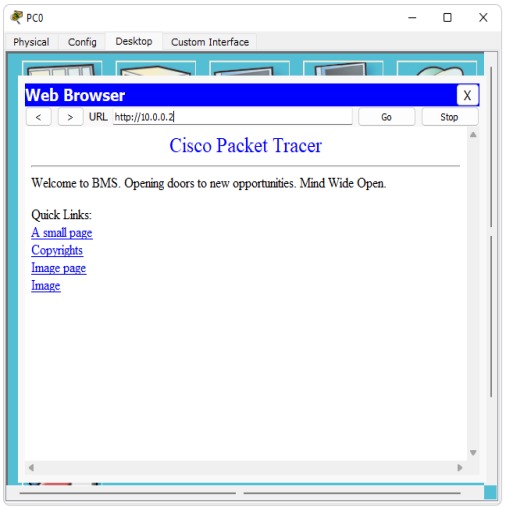
Topology:

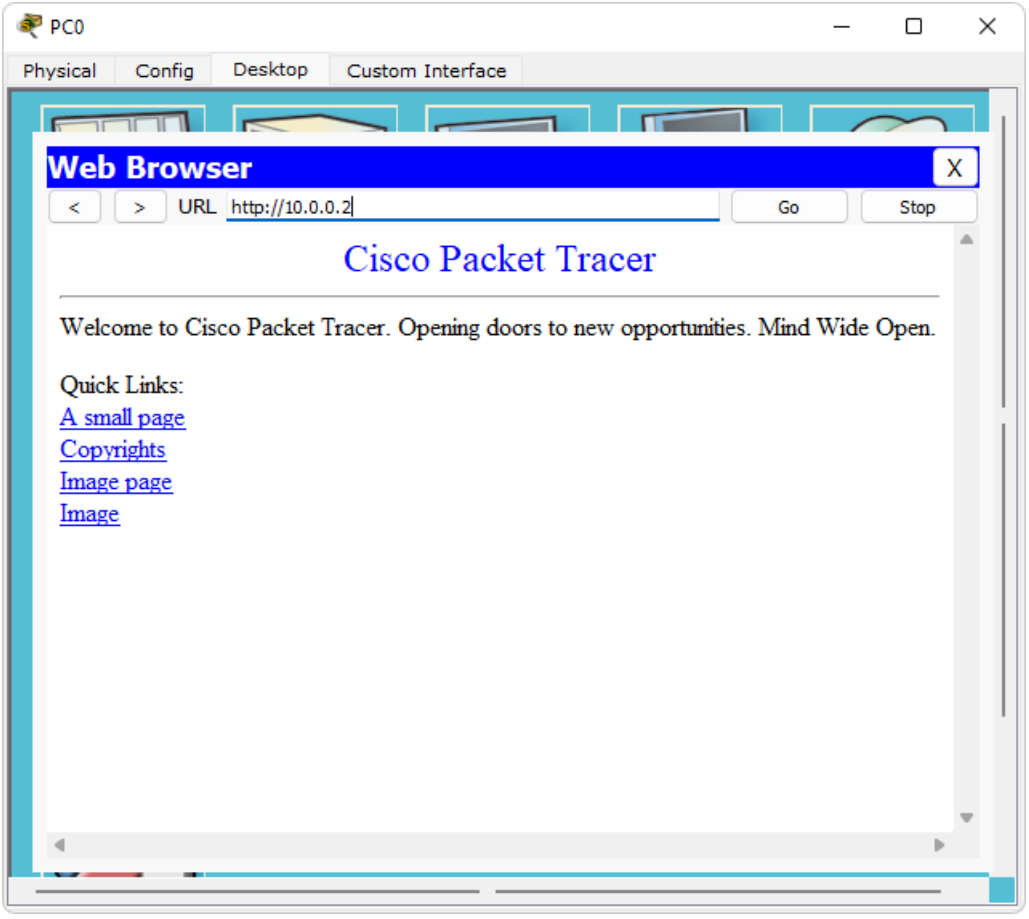


Output:




Fa0
PC-PT
PC0
Fa0
Server-PT
Server0


PC0
Physical
Config
Desktop
Custom Interface
Web Browser
URL
http://10.0.0.2
Go
Stop
Cisco Packet Tracer
Welcome to BMS. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image


PC0
Physical
Config
Desktop
Custom Interface
Web Browser
URL
http://10.0.0.2
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image

Experiment 9:

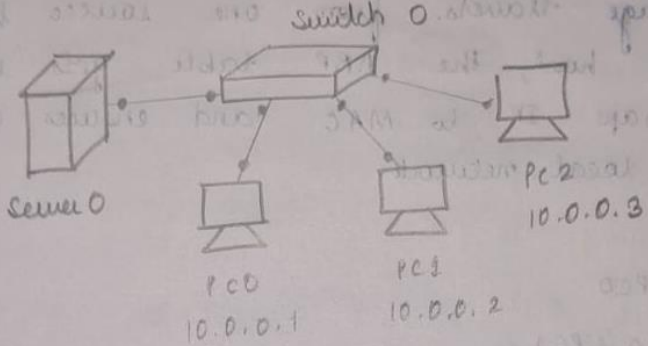
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

Observation:

Q: To construct a simple LAN and understand concept and operation of ARP

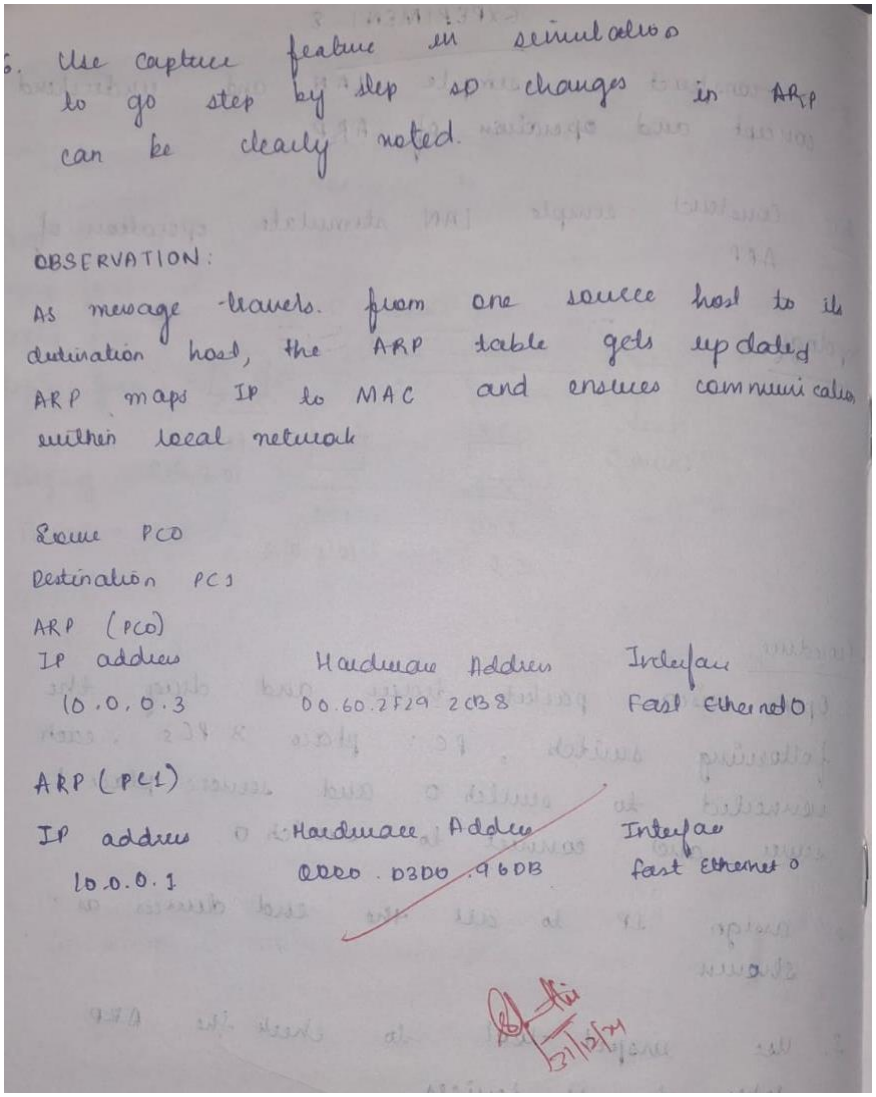
Aim: Construct simple LAN simulate operation of ARP

Topology:



Procedure:

1. Open cisco packet tracer and drag the following switch, PC: place 3 PC's, each connected to switch 0 and server: place 1 server and connect to switch 0
2. assign IP to all the end devices as shown
3. Use inspect tool to check the ARP table of all devices
4. Can also use CLI (of end devices) to check arp table (command - arp -a)
5. CLI of switch, show mac address-table displays table



5. Use capture feature in simulation to go step by step so changes in ARP can be clearly noted.

OBSERVATION:

As message travels from one source host to its destination host, the ARP table gets updated. ARP maps IP to MAC and ensures communication within local network

Source PC0

Destination PC1

ARP (PC0)

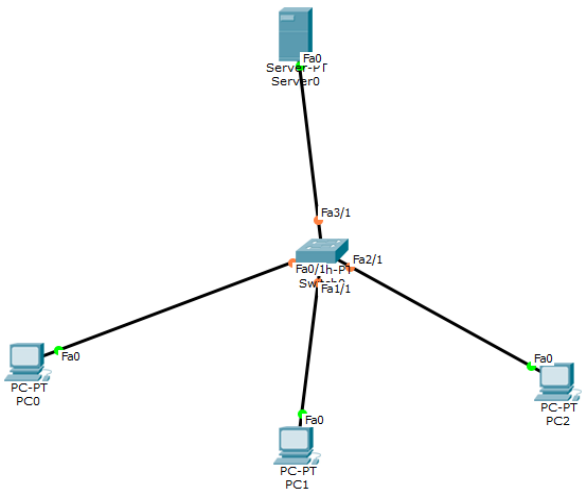
IP address	Hardware Address	Interface
10.0.0.3	0060.2F29.2CB8	Fast Ethernet0

ARP (PC1)

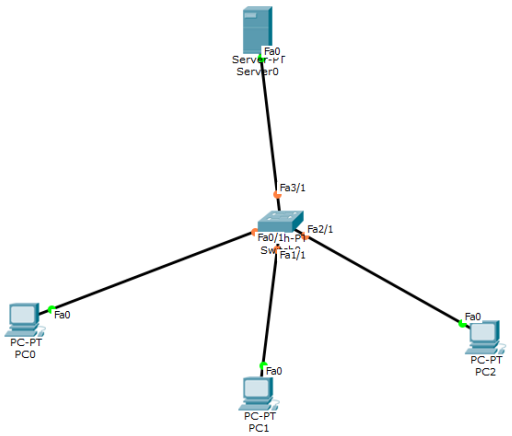
IP address	Hardware Address	Interface
10.0.0.1	0000.0D0D.96DB	Fast Ethernet0

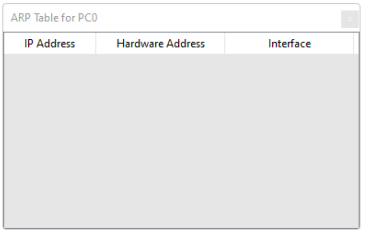
31/12/24

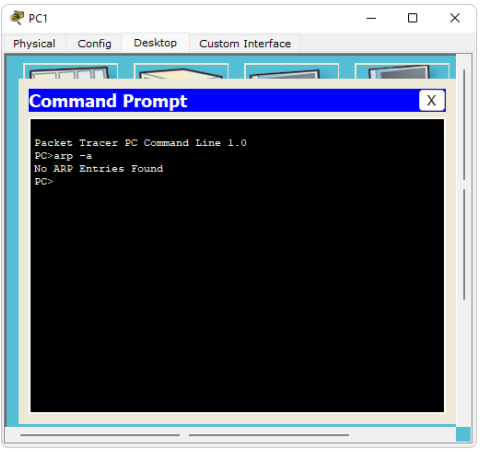
Topology:

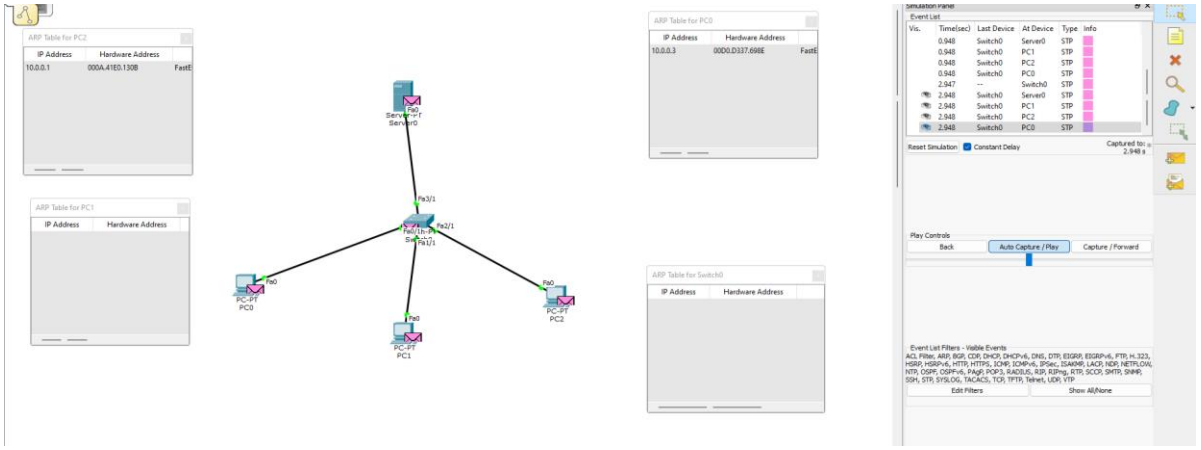


Output:


Fa0
Server0
Fa3/1
Fa2/1
Fa1/1
Fa0
PC-PT
PC0
Fa0
PC-PT
PC2
Fa0
PC-PT
PC1


ARP Table for PC0
IP Address
Hardware Address
Interface

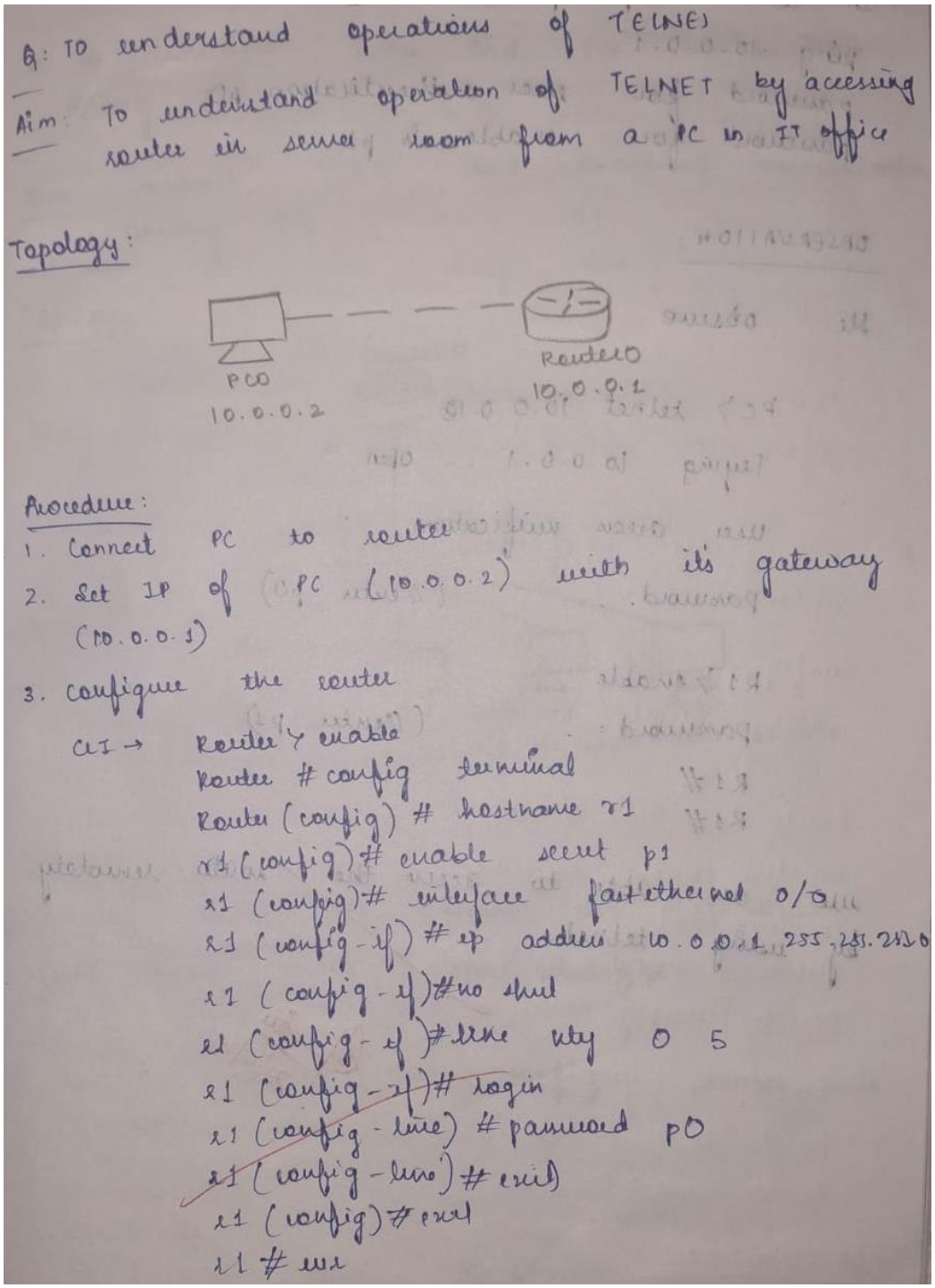

PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>arp -a
No ARP Entries Found
PC>



Experiment 10:

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

Observation:



In command prompt (PC0)

ping 10.0.0.1

password for user authentication p0

password for enable is p1

OBSERVATION:

We observe

PC> telnet 10.0.0.1

Trying 10.0.0.1 ... Open

User access verification

Password: ... (Enter p0)

R1>enable

Password: ... (Enter p1)

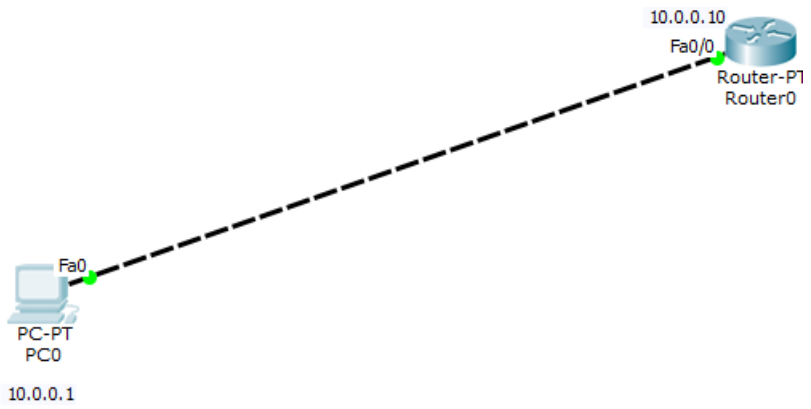
R1#

R1#

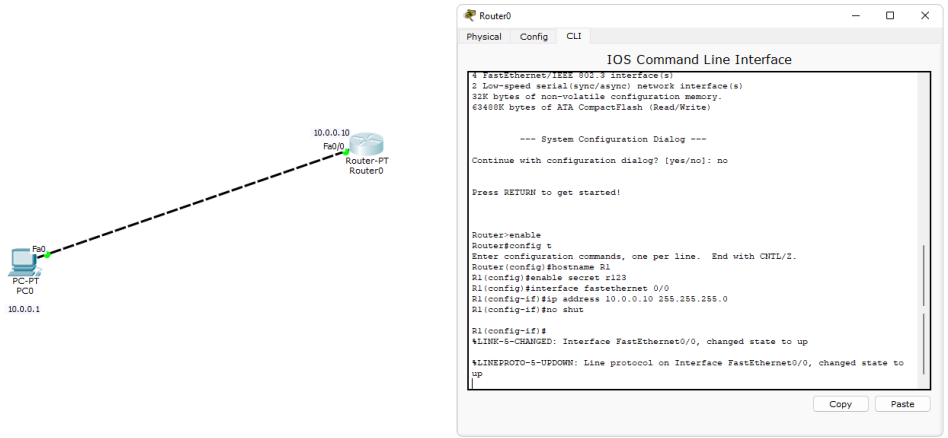
we are able to access the router remotely by using telnet.

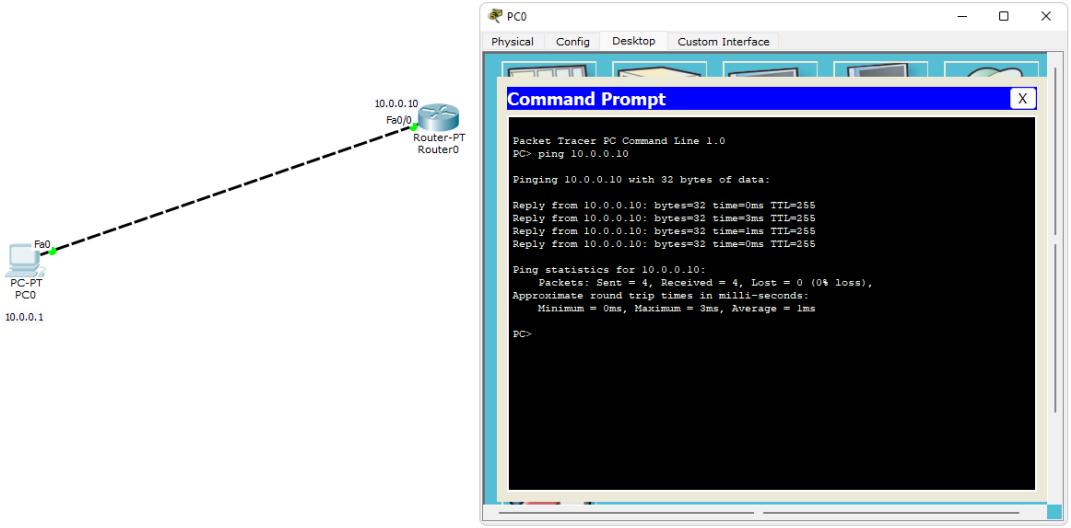
31/12/24

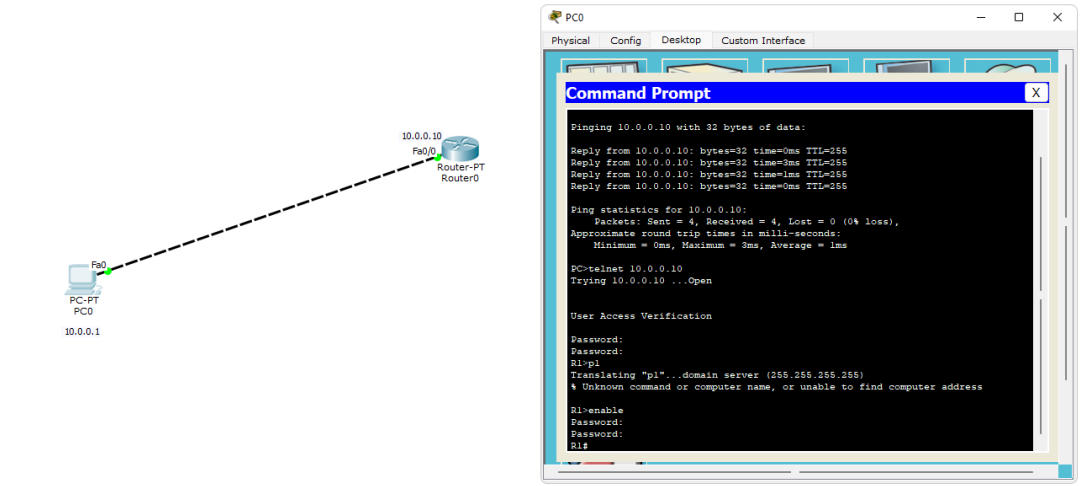
Topology:



Output:

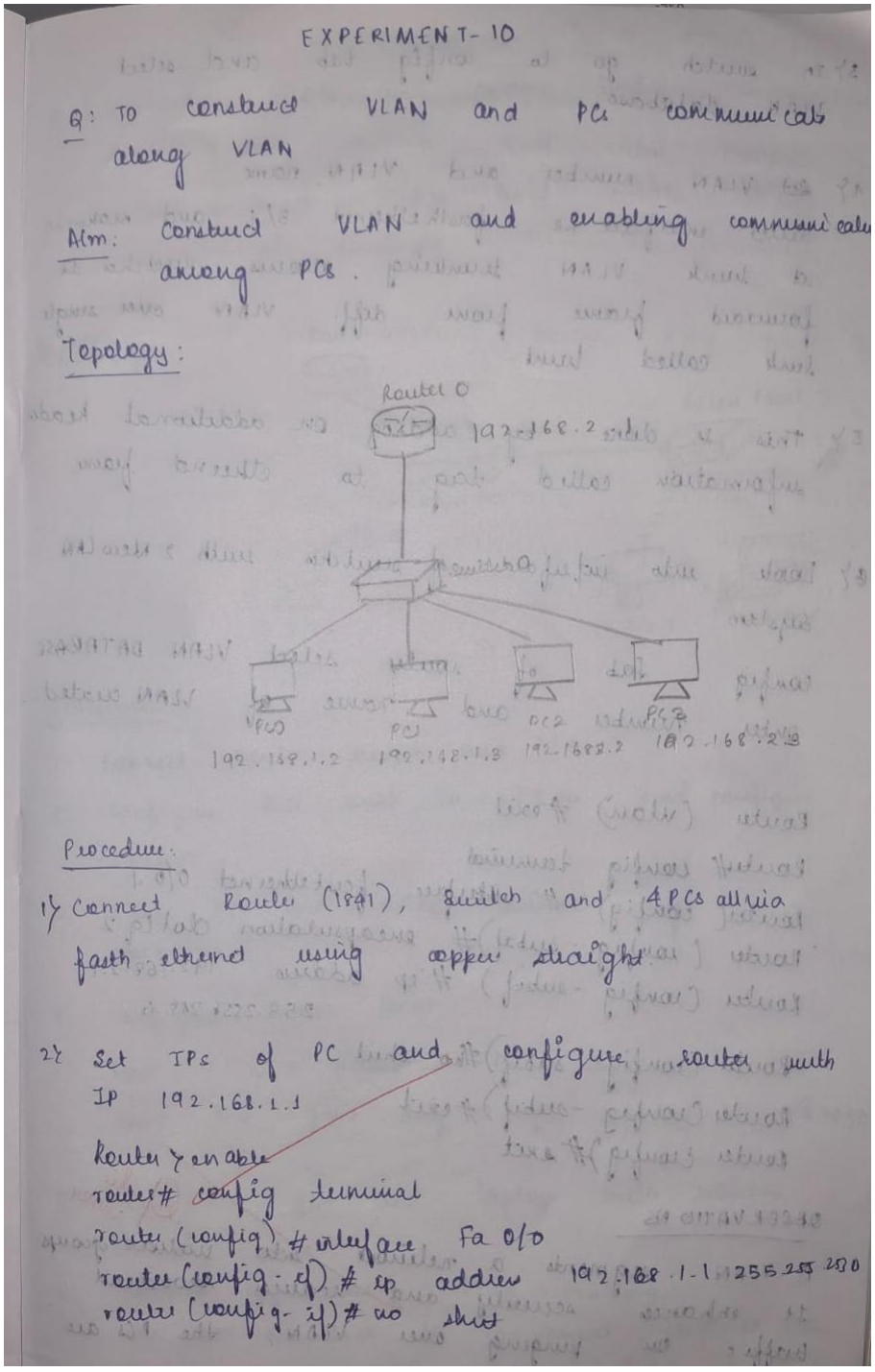


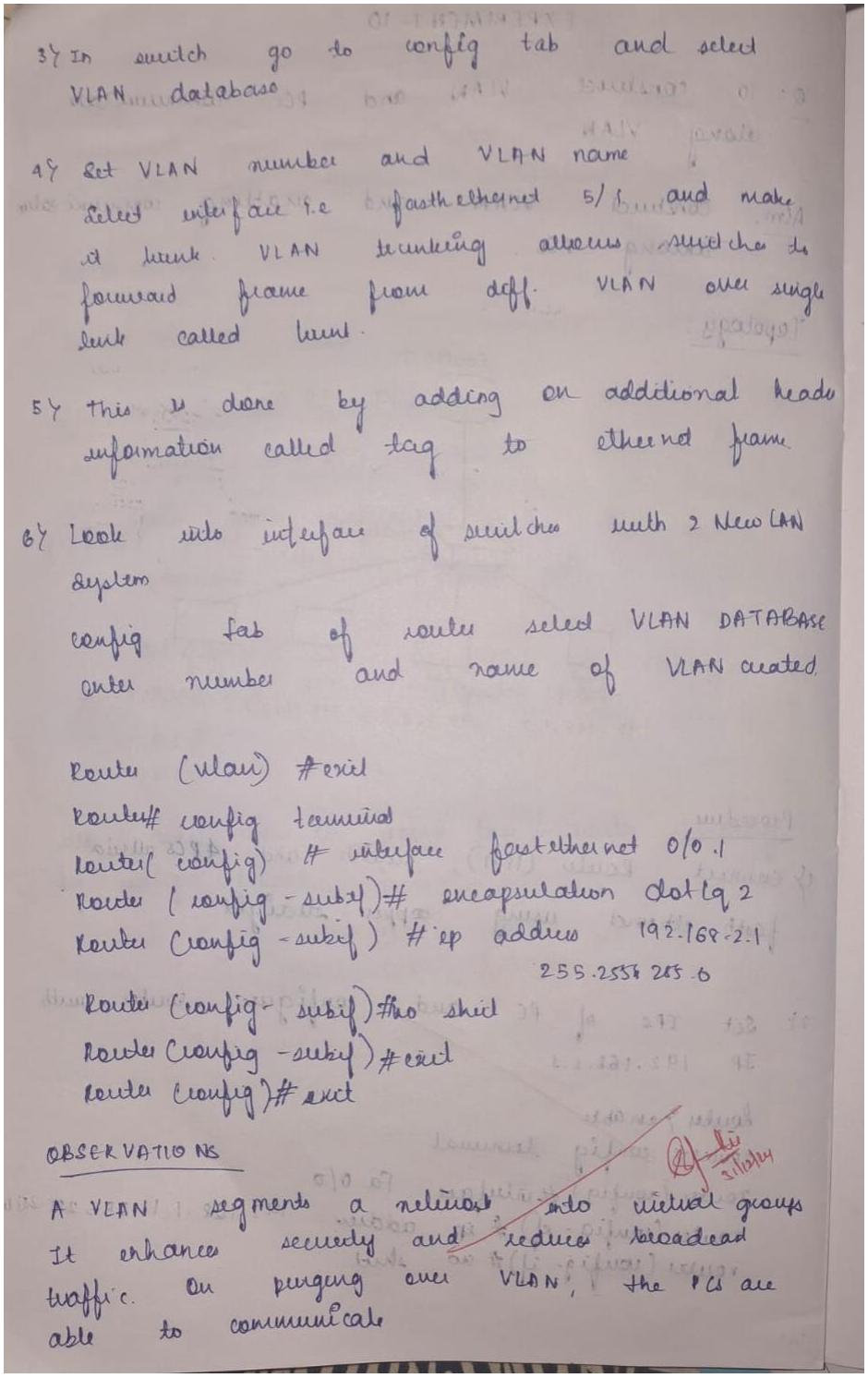

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC> ping 10.0.0.10
Pinging 10.0.0.10 with 32 bytes of data:
Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Reply from 10.0.0.10: bytes=32 time=3ms TTL=255
Reply from 10.0.0.10: bytes=32 time=1ms TTL=255
Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 3ms, Average = 1ms
PC>
10.0.0.10
Fa0/0
Router-PT
Router0
Fa0
PC-PT
PC0
10.0.0.1


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Pinging 10.0.0.10 with 32 bytes of data:
Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Reply from 10.0.0.10: bytes=32 time=3ms TTL=255
Reply from 10.0.0.10: bytes=32 time=1ms TTL=255
Reply from 10.0.0.10: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 3ms, Average = 1ms
PC>telnet 10.0.0.10
Trying 10.0.0.10 ...Open
User Access Verification
Password:
Password:
R1>p1
Translating "p1"...domain server (255.255.255.255)
% Unknown command or computer name, or unable to find computer address
R1>enable
Password:
Password:
R1#
10.0.0.10
Fa0/0
Router-PT
Router0
Fa0
PC-PT
PC0
10.0.0.1

Experiment 11:
To construct a VLAN and make the PC's communicate among a VLAN

Observation:





3) In switch go to config tab and select VLAN database

4) Set VLAN number and VLAN name
Select interface i.e fastethernet 5/1 and make it trunk. VLAN trunking allows switches to forward frame from diff. VLAN over single link called trunk.

5) This is done by adding on additional header information called tag to ethernet frame.

6) Look into interface of switches with 2 New LAN system

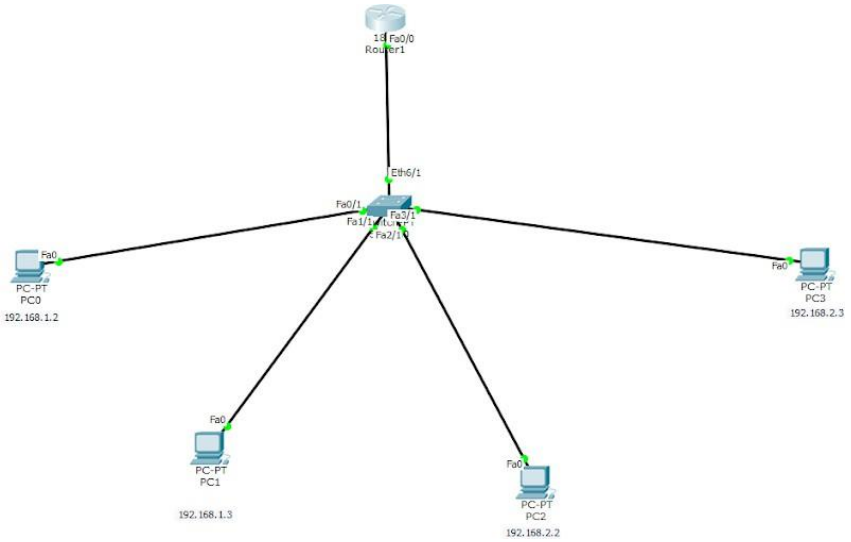
config tab of router select VLAN DATABASE enter number and name of VLAN created.

Router (vlan) #exit
Router# config terminal
Router(config) # interface fastethernet 0/0.1
Router (config-subif)# encapsulation dot1q 2
Router (config-subif) # ip address 192.168.2.1 255.255.255.0
Router (config-subif) #no shut
Router (config-subif) #exit
Router (config) # exit

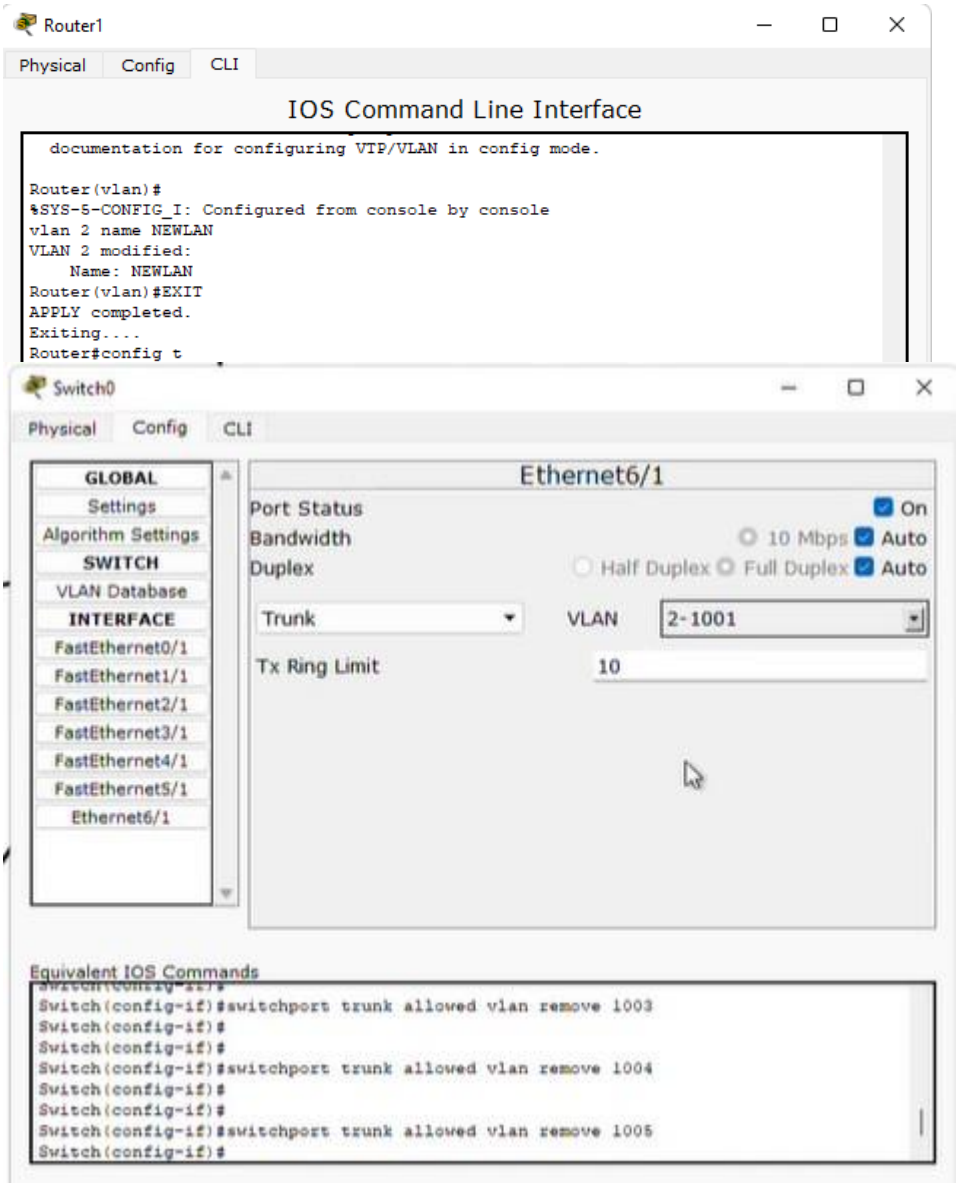
OBSERVATIONS

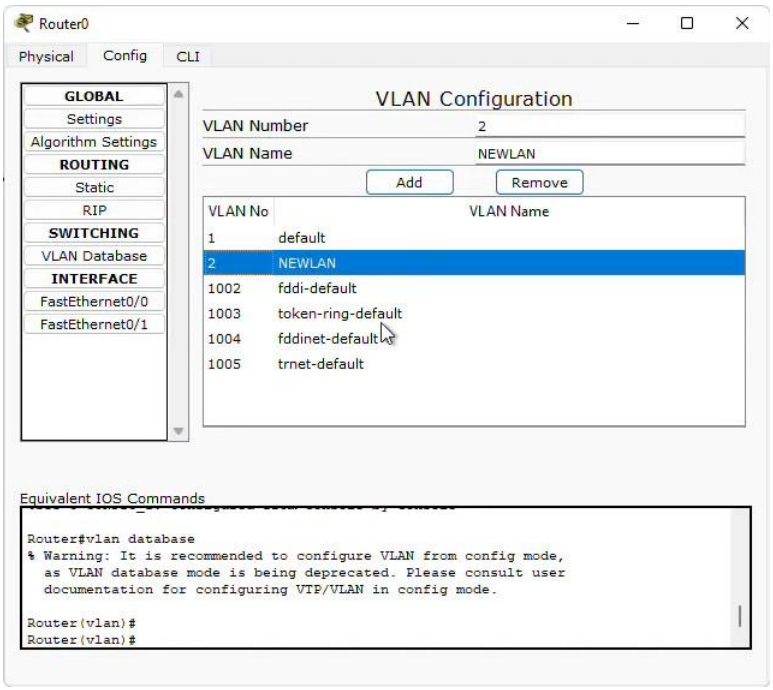
A VLAN segments a network into virtual groups It enhances security and reduces broadcast traffic. On pinging over VLAN, the PCs are able to communicate

Topology:

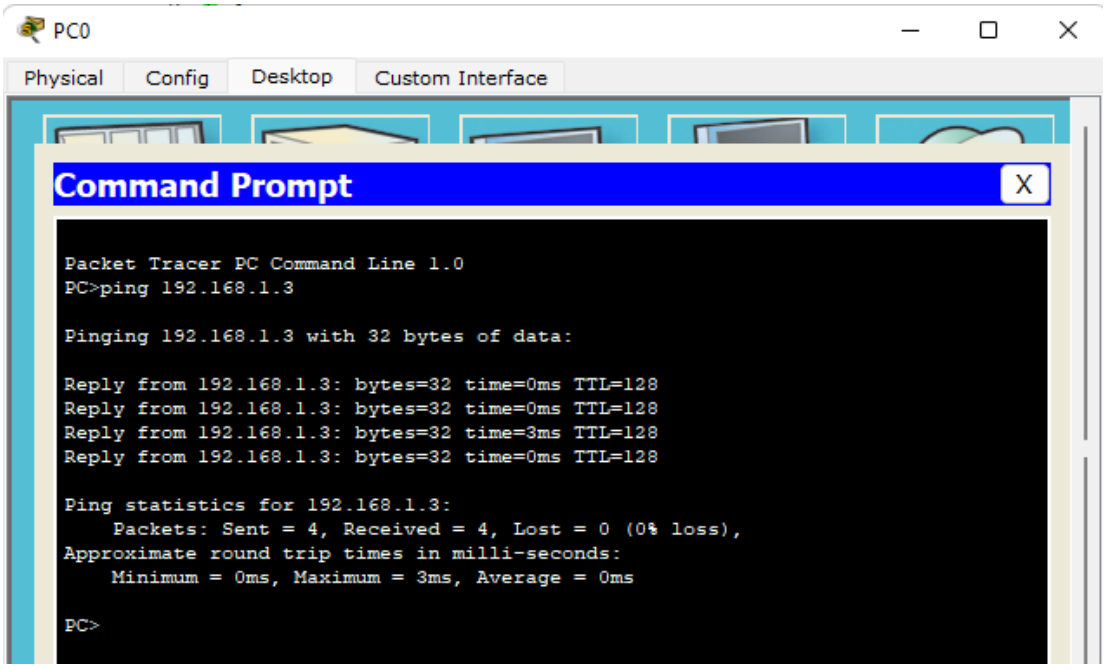

Fa0/0
Router1
Eth6/1
Fa0/1
Fa3/1
Fa2/1
Fa0
PC-PT
PC0
192.168.1.2
Fa0
PC-PT
PC3
192.168.2.3
Fa0
PC-PT
PC1
192.168.1.3
Fa0
PC-PT
PC2
192.168.2.2

Output:


Router1
Physical
Config
CLI
IOS Command Line Interface
documentation for configuring VTP/VLAN in config mode.
Router(vlan)#
%SYS-5-CONFIG_I: Configured from console by console
vlan 2 name NEWLAN
VLAN 2 modified:
Name: NEWLAN
Router(vlan)#EXIT
APPLY completed.
Exiting....
Router#config t
Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet1/1
FastEthernet2/1
FastEthernet3/1
FastEthernet4/1
FastEthernet5/1
Ethernet6/1
Ethernet6/1
Port Status
On
Bandwidth
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Trunk
VLAN
2-1001
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config-if)#switchport trunk allowed vlan remove 1003
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport trunk allowed vlan remove 1004
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport trunk allowed vlan remove 1005
Switch(config-if)#

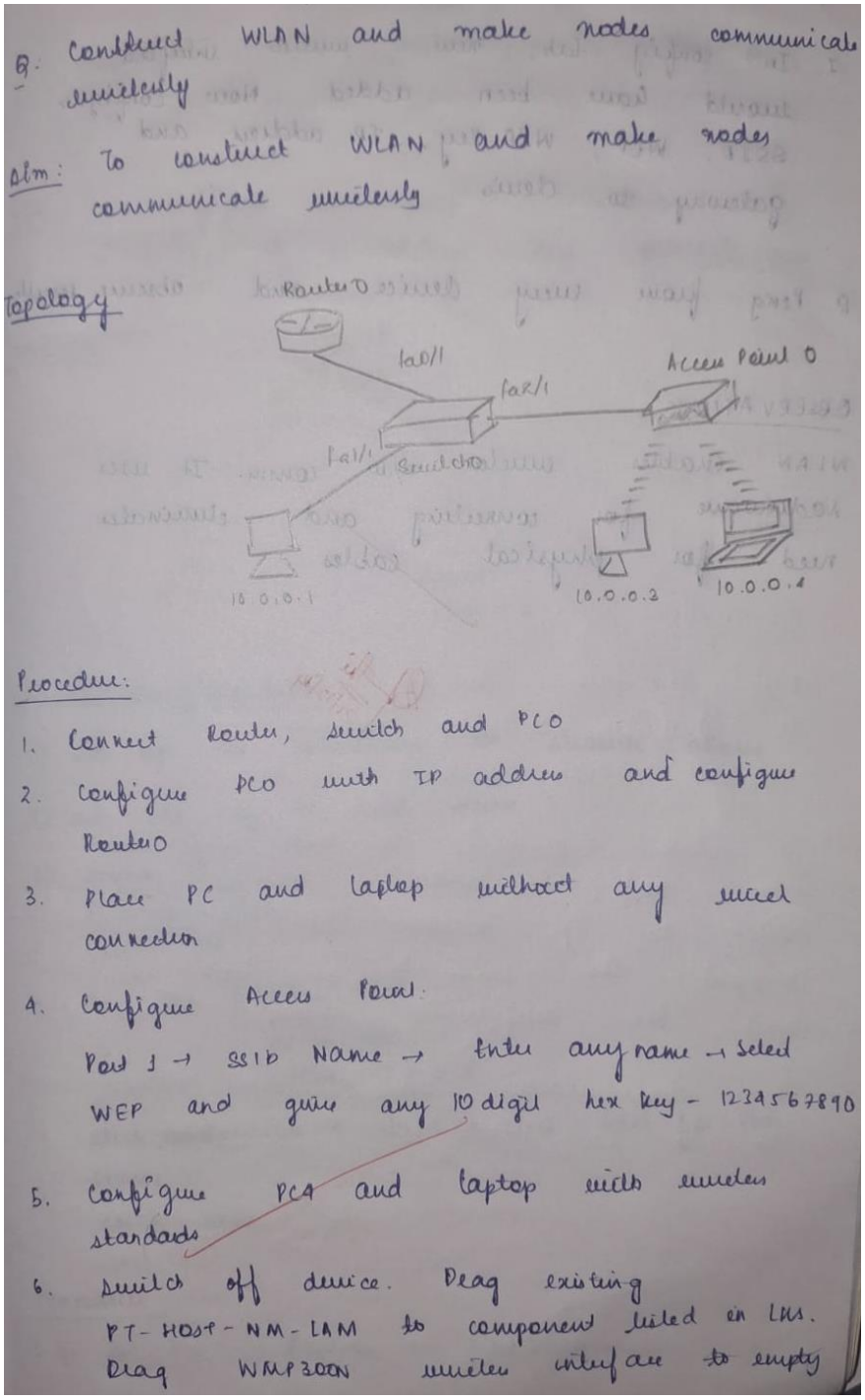


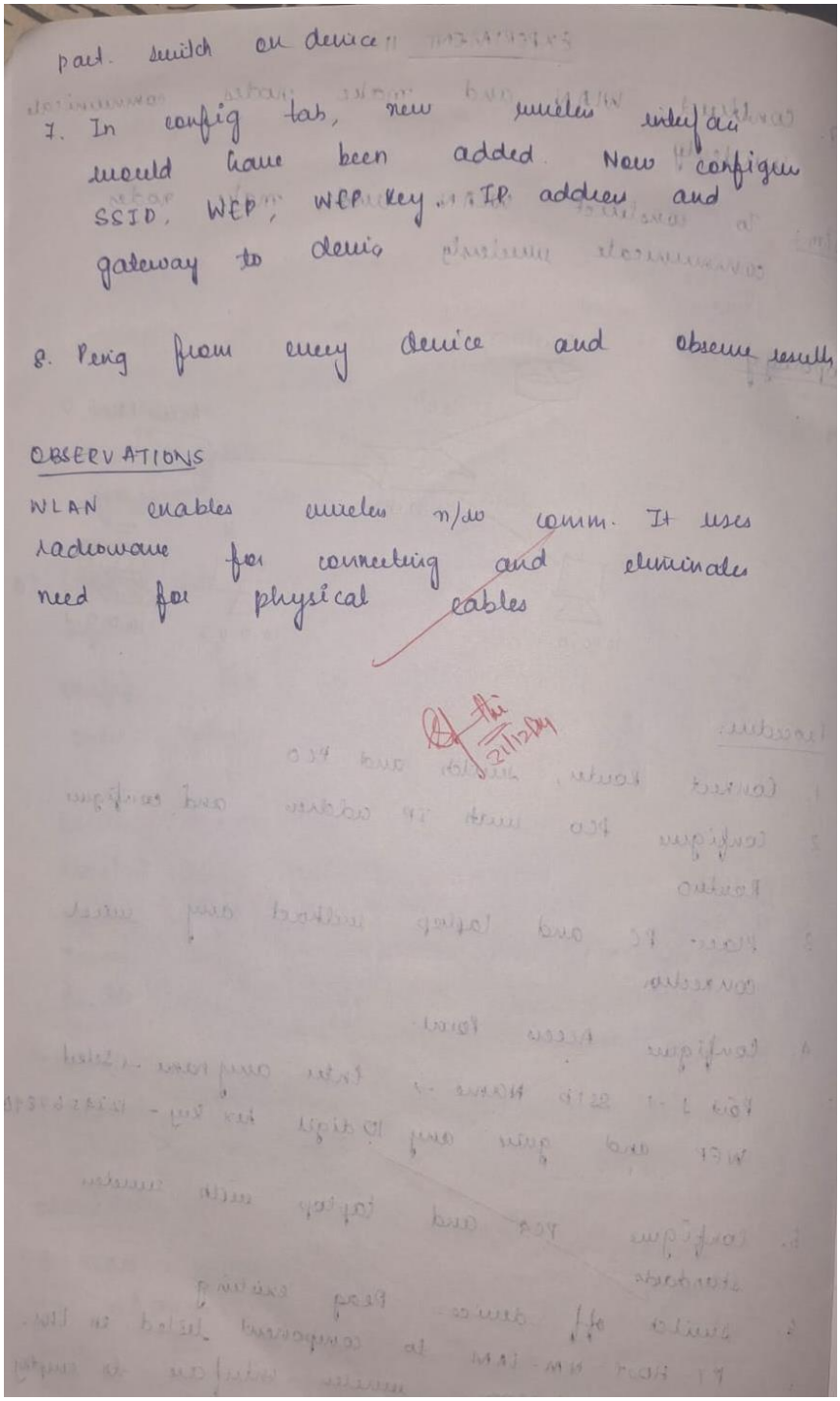

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to
up
exit
Router(config)#
Router(config)#exit
Router#vlan database
% Warning: It is recommended to configure VLAN from config mode,
  as VLAN database mode is being deprecated. Please consult user
  documentation for configuring VTP/VLAN in config mode.

Router(vlan)#
%SYS-5-CONFIG_I: Configured from console by console
vlan 2 name NEWLAN
VLAN 2 modified:
    Name: NEWLAN
Router(vlan)#EXIT
```



Experiment 12:
To construct a WLAN and make the nodes communicate wirelessly

Observation:





part. switch on device

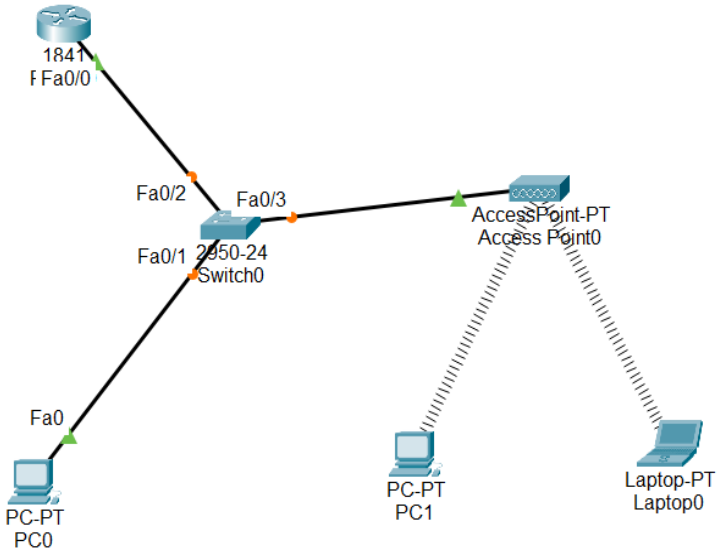
7. In config tab, new wireless interface would have been added. Now configure SSID, WEP, WEP key, IP address and gateway to device

8. Ping from every device and observe results

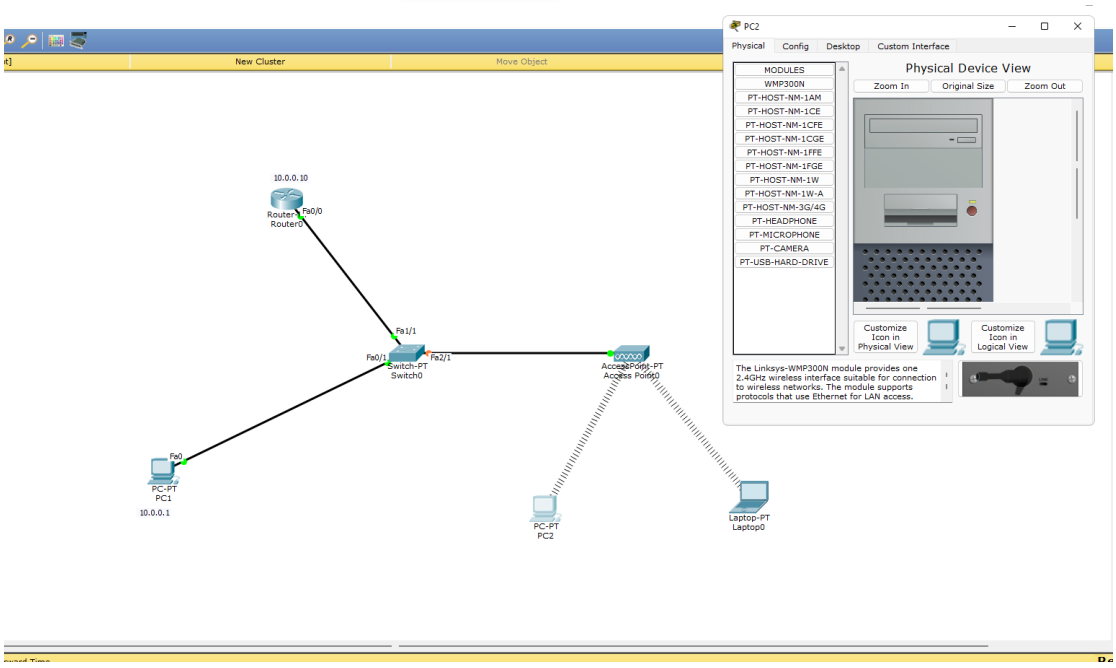
OBSERVATIONS

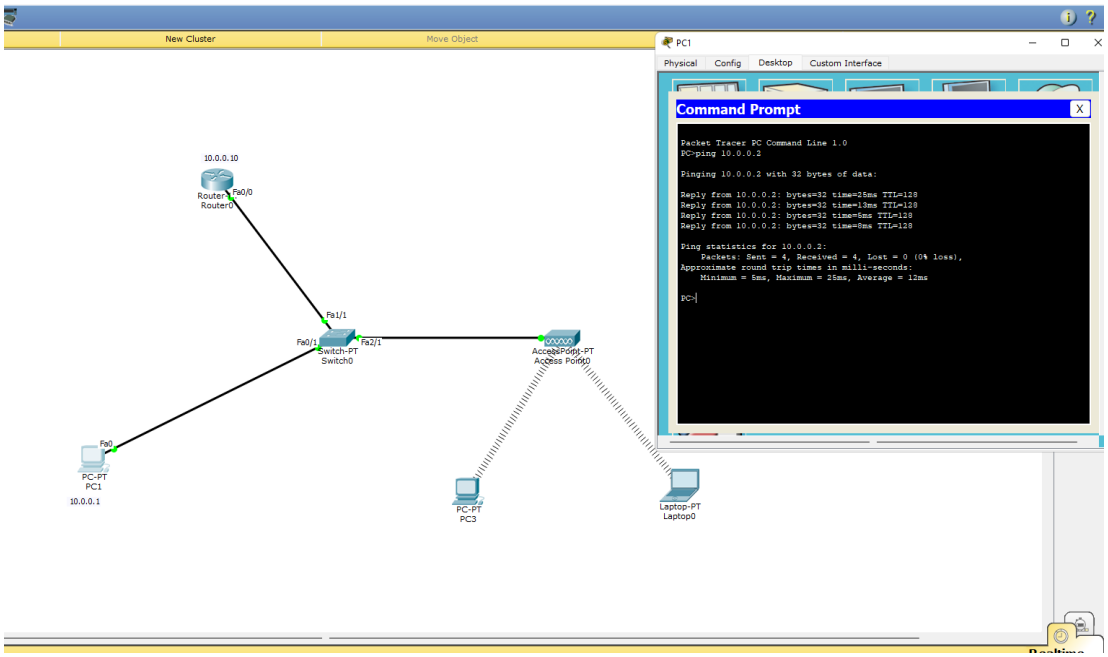
WLAN enables wireless n/w comm. It uses radiowave for connecting and eliminates need for physical cables

Topology:



Output:



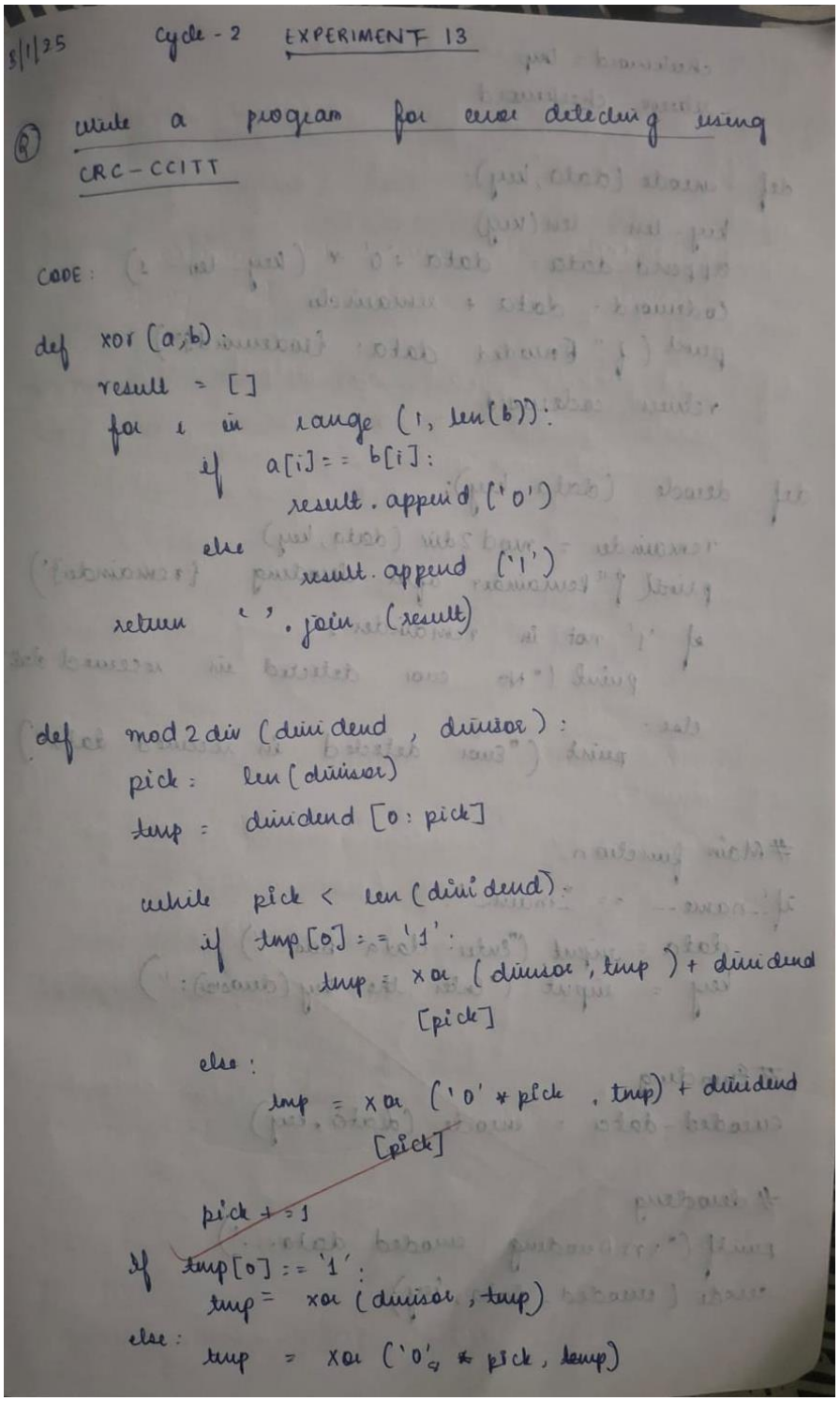

New Cluster
Move Object
10.0.0.10
Router-PT
Router0
Fa0/0
Fa1/1
Fa0/1
Switch-PT
Switch0
Fa2/1
Access Point-PT
Access Point0
Fa0
PC-PT
PC1
10.0.0.1
PC-PT
PC3
Laptop-PT
Laptop0
PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=25ms TTL=128
Reply from 10.0.0.2: bytes=32 time=13ms TTL=128
Reply from 10.0.0.2: bytes=32 time=5ms TTL=128
Reply from 10.0.0.2: bytes=32 time=8ms 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 = 5ms, Maximum = 25ms, Average = 12ms
PC>

Cycle-2

Program 1:

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

Observation:

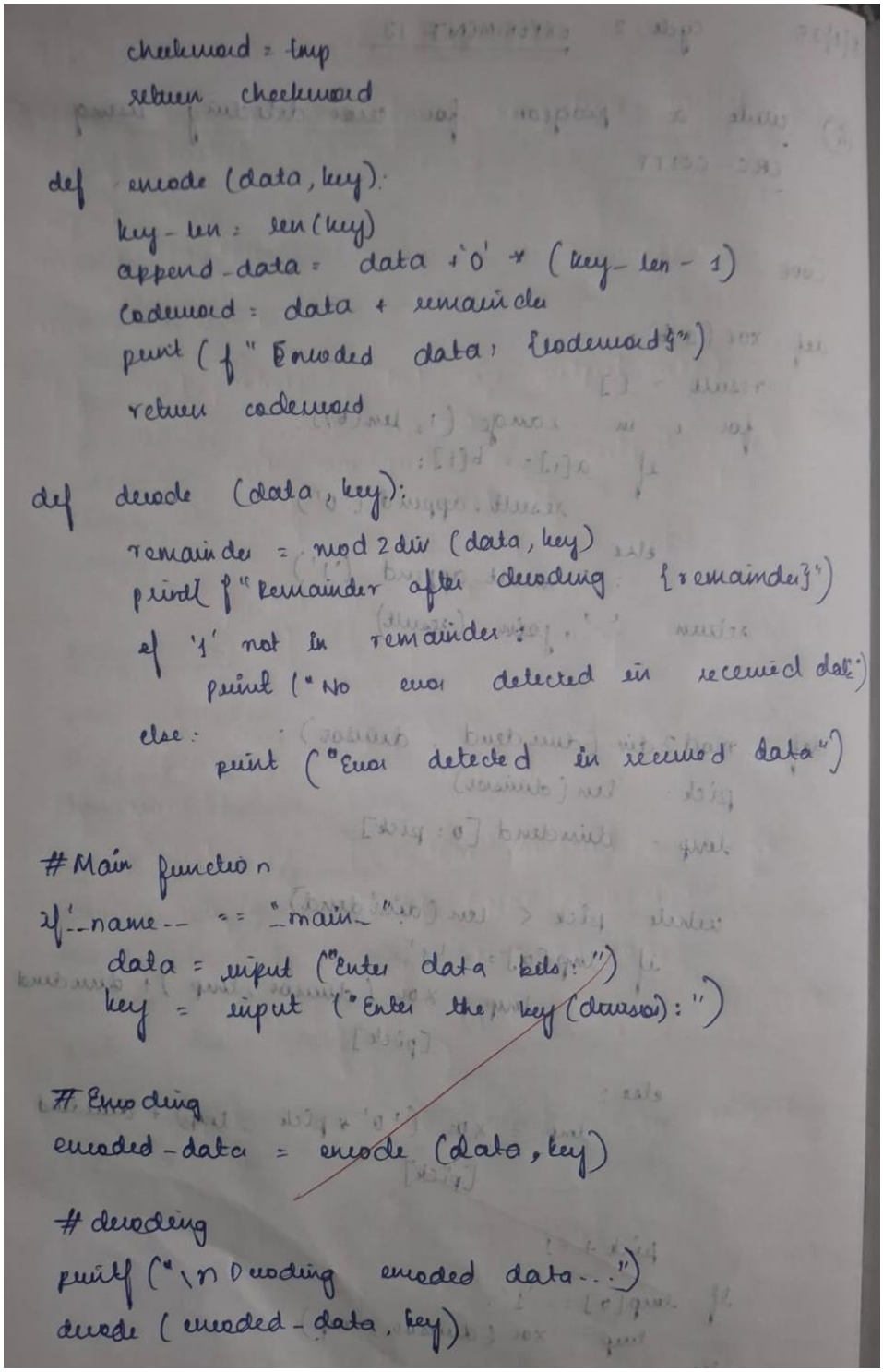


5/1/25 Cycle - 2 EXPERIMENT 13

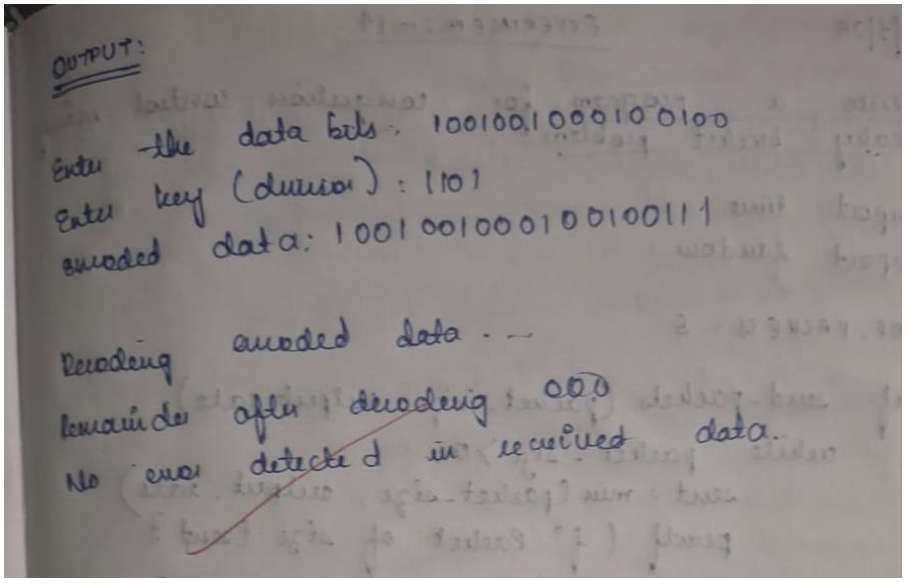
① Write a program for error detecting using CRC-CCITT

CODE:

```
def xor(a,b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else
            result.append('1')
    return ''.join(result)

def mod2div(dividend, divisor):
    pick: len(divisor)
    tmp: dividend[0:pick]

    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend
                  [pick]
        else:
            tmp = xor('0' * pick, tmp) + dividend
                  [pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)
```



```
    checkword = tmp
    return checkword

def encode(data, key):
    key_len = len(key)
    append_data = data + '0' * (key_len - 1)
    codeword = data + remainder
    print(f"Encoded data: {codeword}")
    return codeword

def decode(data, key):
    remainder = mod2div(data, key)
    print(f"Remainder after decoding: {remainder}")
    if '1' not in remainder:
        print("No error detected in received data")
    else:
        print("Error detected in received data")

#Main function
if __name__ == "__main__":
    data = input("Enter data bits: ")
    key = input("Enter the key (divisor): ")

#Encoding
encoded_data = encode(data, key)

#decoding
print("\nDecoding encoded data...")
decode(encoded_data, key)
```



Code:

```
def crc_ccitt_16_bitstream(bitstream: str, poly: int = 0x1021, init_crc: int =
0xFFFF) -> int:
    """
    Calculate the 16-bit CRC-CCITT checksum for a given binary string.
    """
    crc = init_crc
    for bit in bitstream:
        crc ^= int(bit) << 15  # Align the bit with CRC's uppermost bit
        for _ in range(8):  # Process each bit
            if crc & 0x8000:  # Check if the leftmost bit is set
                crc = (crc << 1) ^ poly
            else:
                crc <<= 1
            crc &= 0xFFFF  # Ensure CRC remains 16-bit
    return crc


def append_crc_to_bitstream(bitstream: str) -> str:
    """
    Append the calculated 16-bit CRC to the given bitstream.
    """
    crc = crc_ccitt_16_bitstream(bitstream)
    crc_bits = f"{crc:016b}"  # Convert CRC to a 16-bit binary string
    return bitstream + crc_bits


def verify_crc_bitstream(bitstream_with_crc: str) -> bool:
    """
    Verify the CRC of the given bitstream with CRC appended.
    """
    if len(bitstream_with_crc) < 16:
        return False  # Not enough bits to contain CRC
    data, received_crc = bitstream_with_crc[:-16], bitstream_with_crc[-16:]
```

```
    calculated_crc = crc_ccitt_16_bitstream(data)
    return calculated_crc == int(received_crc, 2)


# Main Program
if __name__ == "__main__":
    # User input for original bitstream
    message_bits = input("Enter the original bitstream (e.g., 11010011101100):
").strip()

    # Validate input
    if not all(bit in "01" for bit in message_bits):
        print("Invalid input. Please enter a binary bitstream (e.g.,
11010011101100).")
    else:
        # Calculate and append CRC
        bitstream_with_crc = append_crc_to_bitstream(message_bits)
        print(f"Transmitted bitstream with CRC: {bitstream_with_crc}")

        # User input for received bitstream
        user_bitstream = input("Enter the received bitstream for verification:
").strip()

        # Validate received input
        if not all(bit in "01" for bit in user_bitstream):
            print("Invalid input. Please enter a valid binary bitstream.")
        elif len(user_bitstream) < 16:
            print("Invalid input. Received bitstream must include at least 16
bits for CRC.")
        else:
            # Verify CRC
            is_valid = verify_crc_bitstream(user_bitstream)
            if is_valid:
                print("No errors detected. CRC valid.")
            else:
                print("Error detected! CRC invalid.")
```

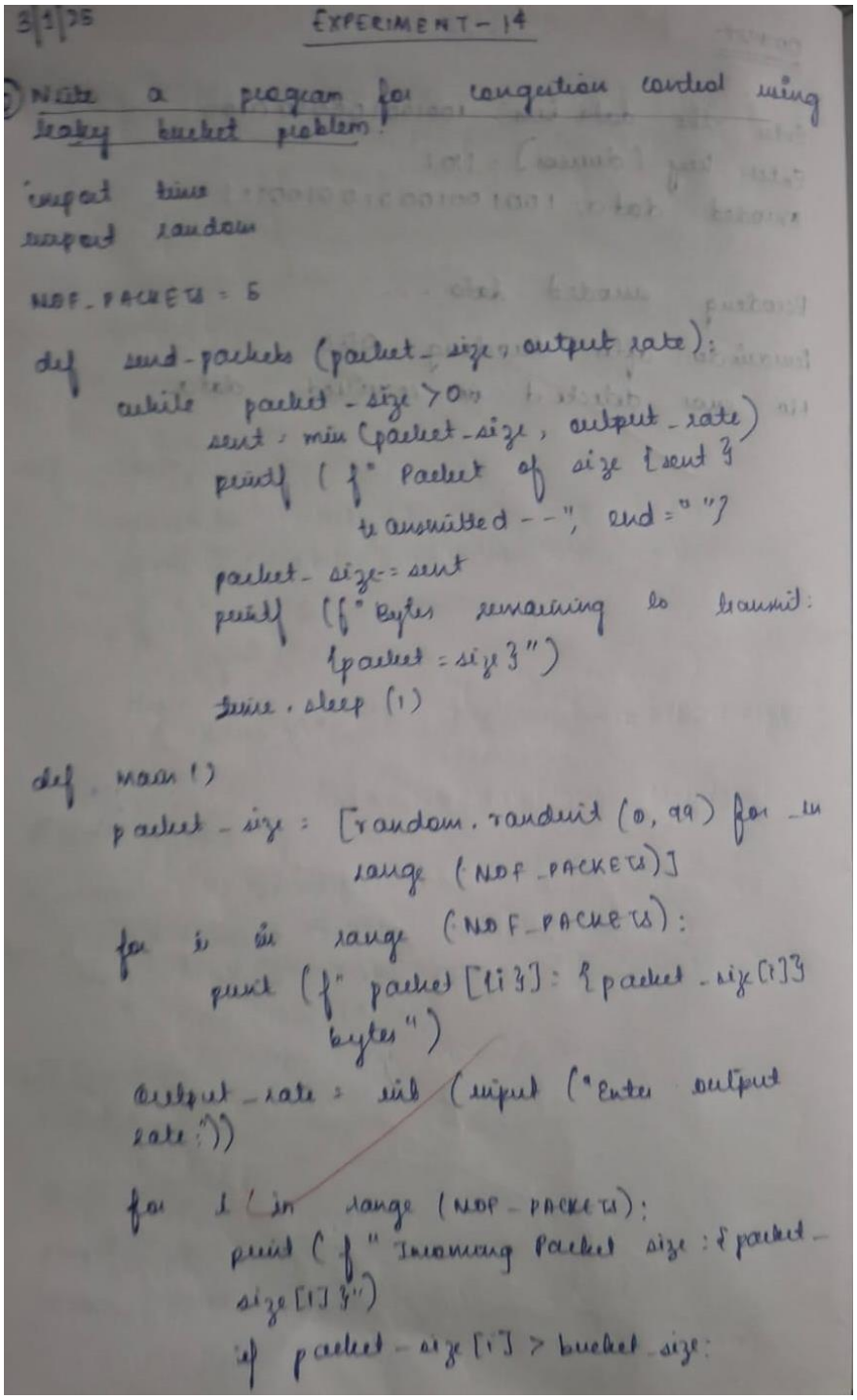
Output:

```
PS C:\Users\HP\OneDrive\Desktop\cn lab> python crc-ccitt.py
Enter the original bitstream (e.g., 11010011101100): 11010011101100
Transmitted bitstream with CRC: 1101001110110010100110001001000
Enter the received bitstream for verification: 1101000000000010100001
Error detected! CRC invalid.
```

Program 2:

Write a program for congestion control using Leaky bucket algorithm.

Observation:

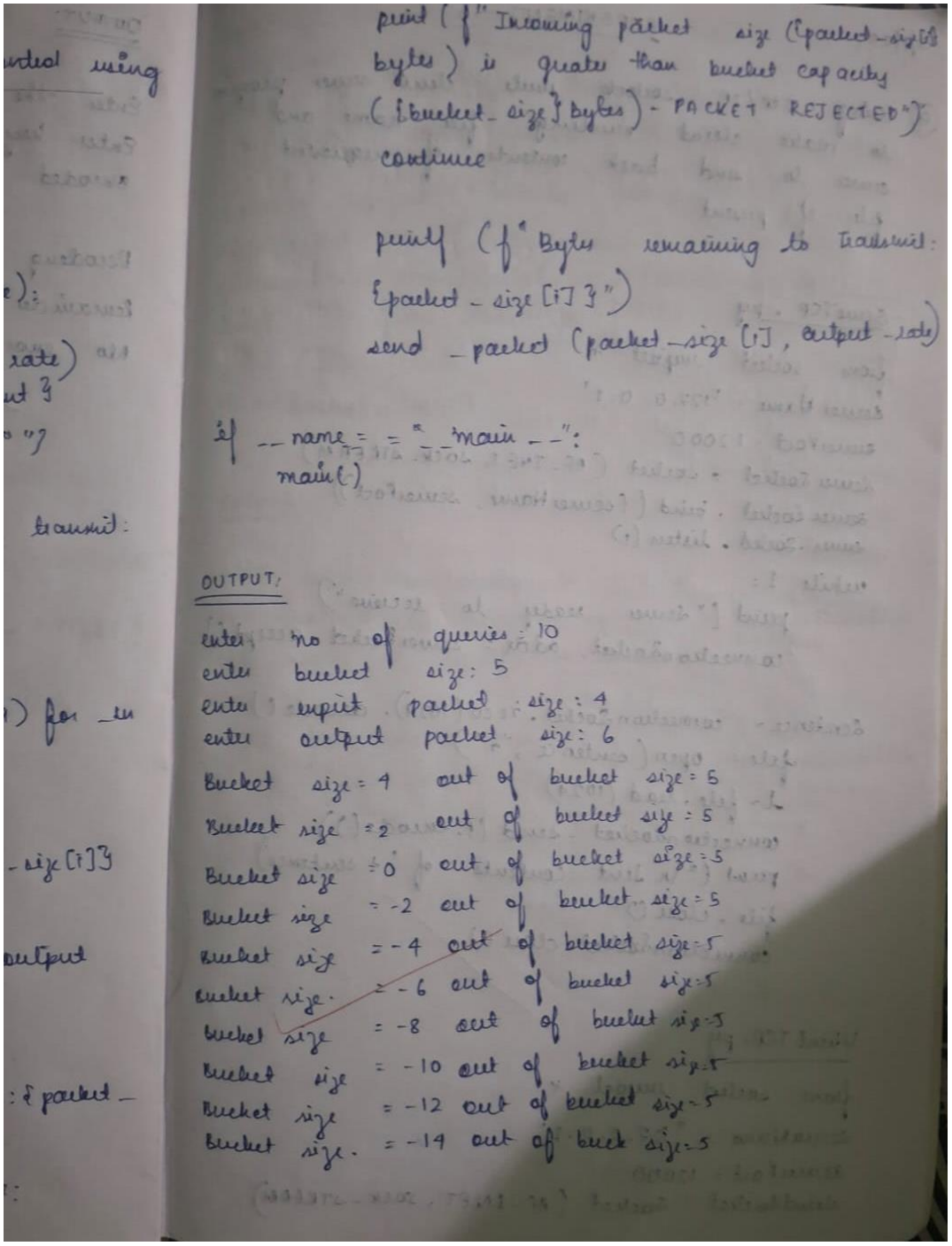


3/1/25

EXPERIMENT - 14

1) Write a program for congestion control using leaky bucket problem.

```
input time
input random

NOF_PACKETS = 5

def send_packets(packet_size, output_rate):
    while packet_size > 0:
        sent = min(packet_size, output_rate)
        print(f" Packet of size {sent}
              transmitted --", end=" ")
        packet_size -= sent
        print(f" Bytes remaining to transmit:
              {packet_size}")
        time.sleep(1)

def main():
    packet_size = [random.randint(0, 99) for _in
                   range(NOF_PACKETS)]
    for i in range(NOF_PACKETS):
        print(f" packet[{i}]: {packet_size[i]}
              bytes")
    output_rate = int(input("Enter output
    rate:"))
    for i in range(NOF_PACKETS):
        print(f" Incoming Packet size: {packet_
        size[i]}")
        if packet_size[i] > bucket_size:
```



print (f" Incoming packet size ({packet_size}
bytes) is greater than bucket capacity
({bucket_size} bytes) - PACKET REJECTED")
continue

print (f" Bytes remaining to Transmit:
{packet_size[i]}")
send_packet (packet_size[i], output_rate)

if __name__ == "__main__":
main()

OUTPUT:

enter no of queries : 10
enter bucket size: 5
enter input packet size : 4
enter output packet size: 6
Bucket size = 4 out of bucket size = 5
Bucket size = 2 out of bucket size = 5
Bucket size = 0 out of bucket size = 5
Bucket size = -2 out of bucket size = 5
Bucket size = -4 out of bucket size = 5
Bucket size = -6 out of bucket size = 5
Bucket size = -8 out of bucket size = 5
Bucket size = -10 out of bucket size = 5
Bucket size = -12 out of bucket size = 5
Bucket size = -14 out of buck size = 5

Code:

```
storage=0
noofqueries=int(input("Enter no of queries:"))
bucketsize=int(input("Enter bucket size:"))
inputpktsize=int(input("Enter input packet size:"))
outputpktsize=int(input("Enter output packet size:"))
for i in range(0,noofqueries):
    sizeleft=bucketsize-storage
    if inputpktsize<=sizeleft:
        storage+=inputpktsize
    else:
        print("Packet loss=", inputpktsize)
```

```
    print(f"Bucket size={storage}out of bucket size={bucketsize}")
    storage-=outputpktsize
```

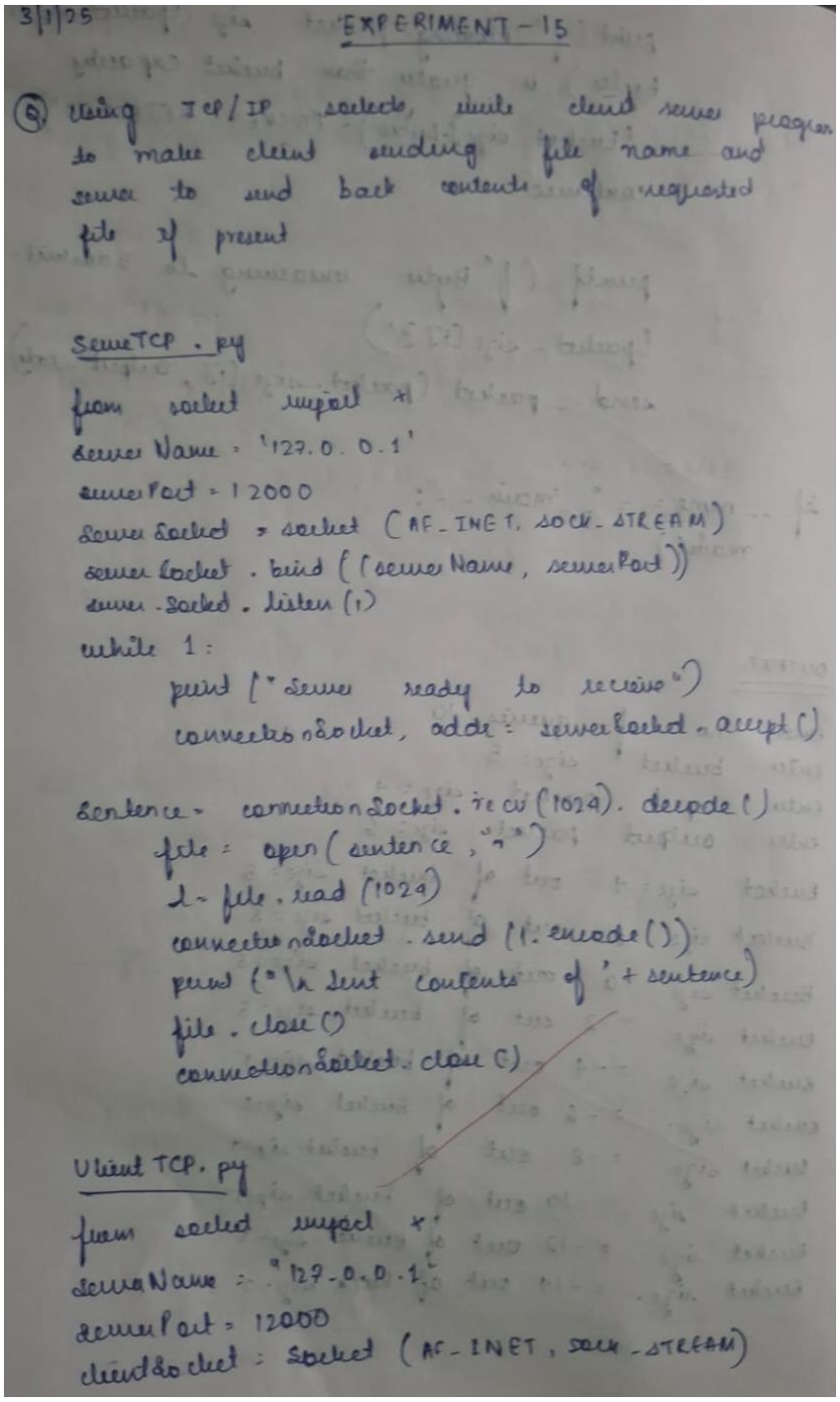
Output:

```
PS C:\Users\Dell\OneDrive\Desktop\code> python leakybucketalgorithm.py
Enter no of queries:10
Enter bucket size:5
Enter input packet size:4
Enter output packet size:6
Bucket size=4out of bucket size=5
Bucket size=2out of bucket size=5
Bucket size=0out of bucket size=5
Bucket size=-2out of bucket size=5
Bucket size=-4out of bucket size=5
Bucket size=-6out of bucket size=5
Bucket size=-14out of bucket size=5
```

Program 3:

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

Observation:



3/1/25

EXPERIMENT - 15

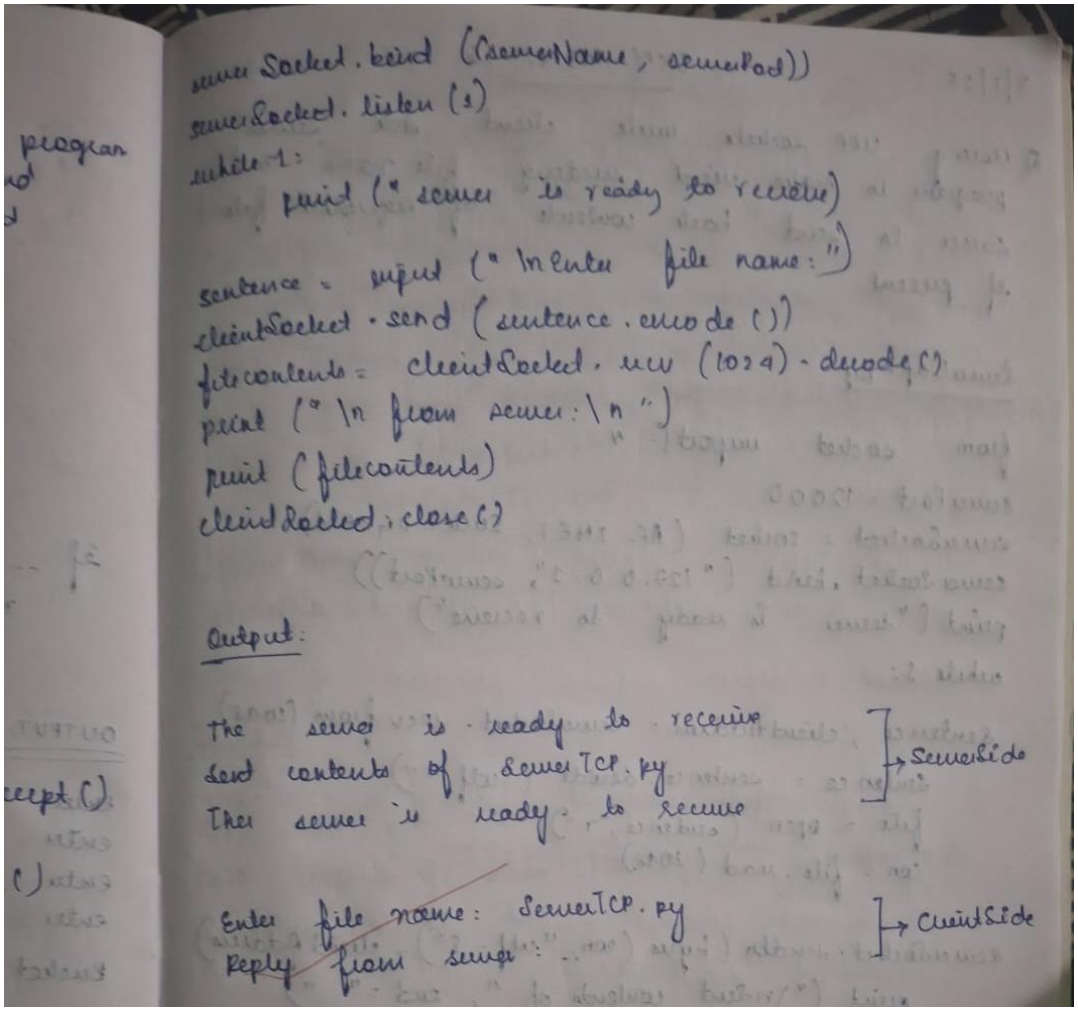
(8) Using TCP/IP sockets, write client server program to make client sending file name and server to send back contents of requested file if present

ServeTCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
server.Socket.listen(1)
while 1:
    print("Server 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()
```

Client TCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
```



Servertcp.py

```
from socket import *
serverName="127.0.0.1"
serverPort = 14000
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()
```

Clienttcp.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 14000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ('\nFrom Server:\n')
print(filecontents)
clientSocket.close()
```

Output:

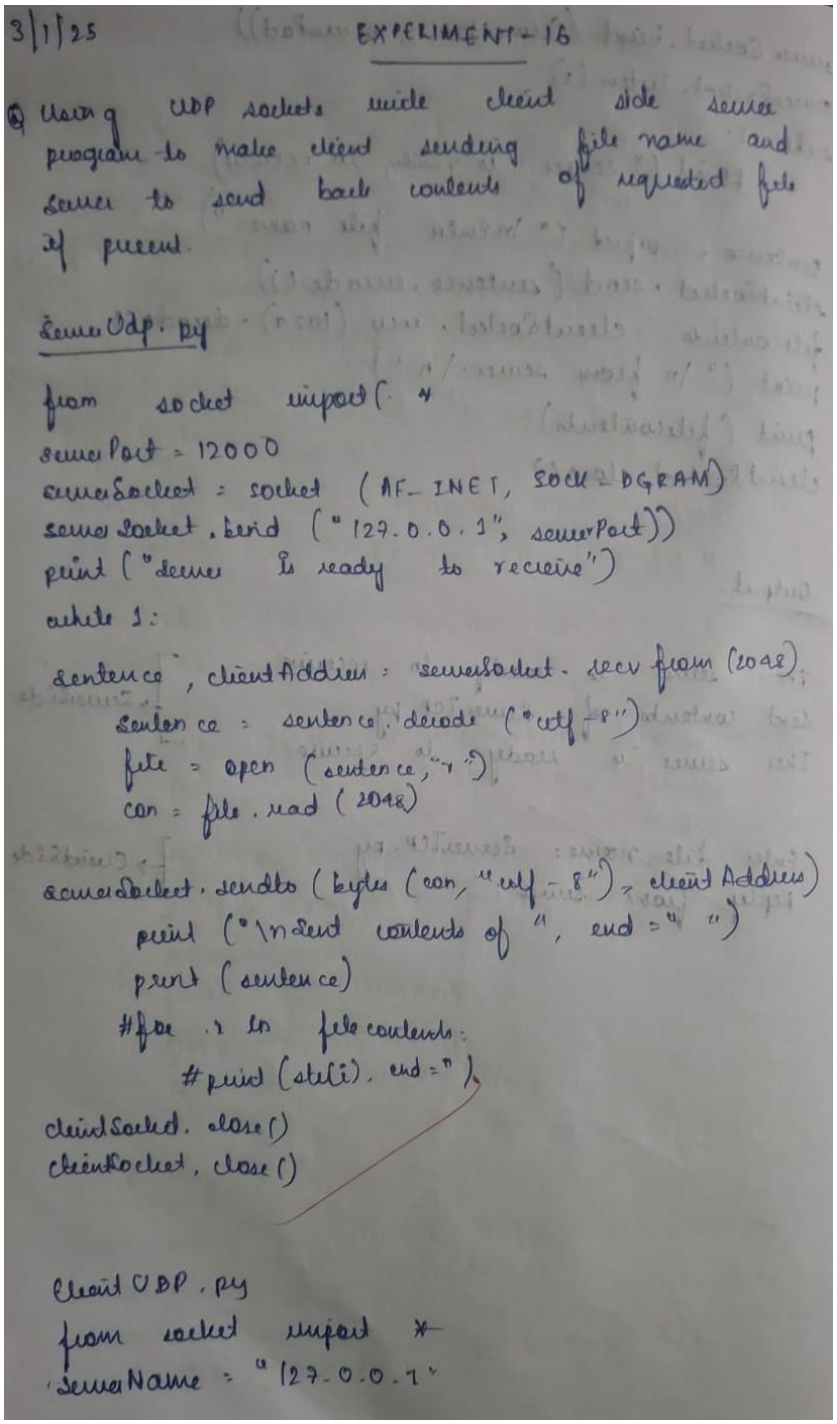
```
PS C:\Users\Dell\OneDrive\Desktop\code> python tcpserver.py
The server is ready to receive

Sent contents of example.txt
The server is ready to receive
```

```
PS C:\Users\Dell\OneDrive\Desktop\code> python tcpclient.py

Enter file name: example.txt

From Server:

Hello, this is a sample file.
It is used for testing the TCP server.
```

Program 4:

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

Observation:



3/1/25 EXPERIMENT - 16

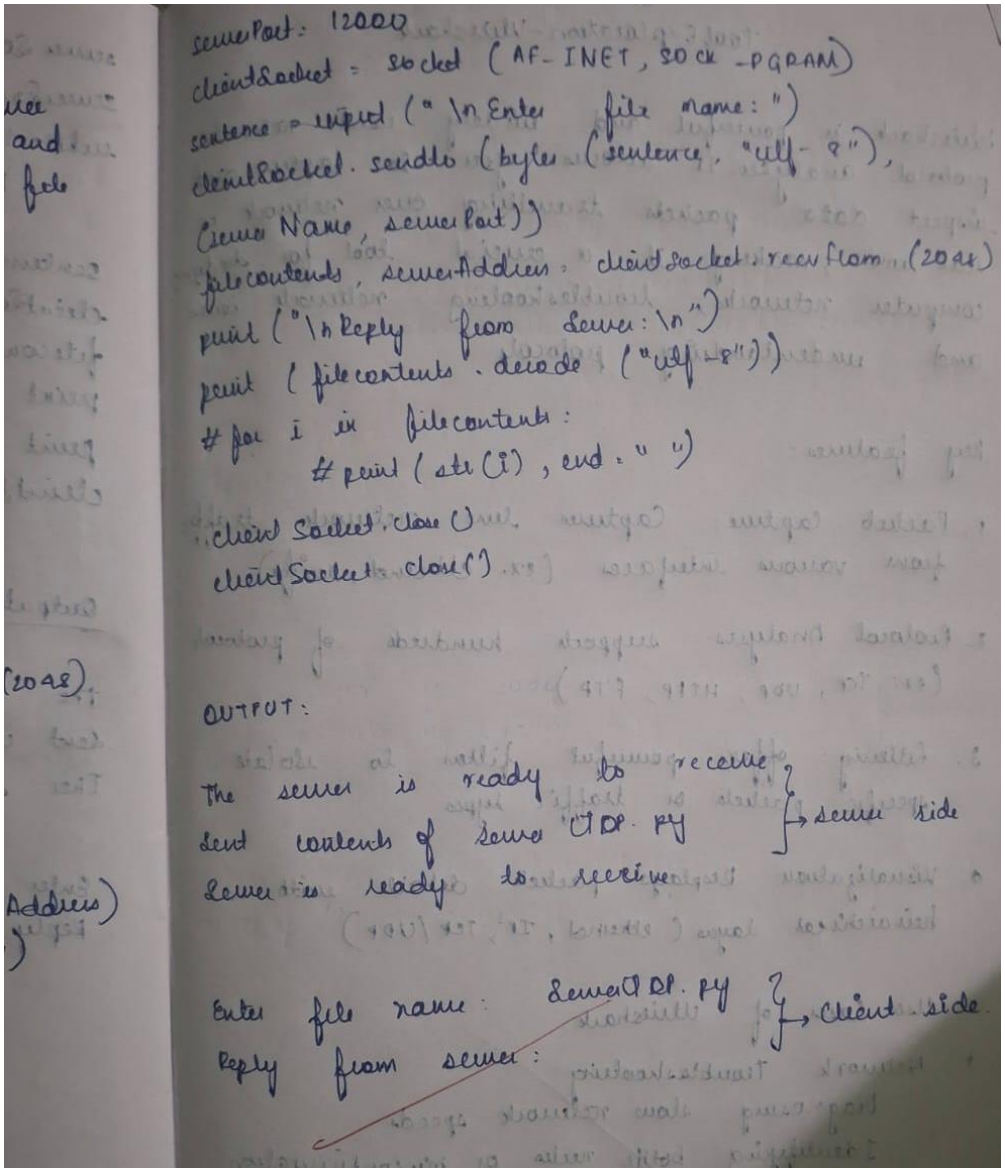
Q Using UDP sockets write client side server program to make client sending file name and server to send back contents of requested file if present.

Server Udp. py

```
from socket import (*
serverPort = 12000
serverSocket = socket (AF_INET, SOCK_DGRAM)
serverSocket.bind ("127.0.0.1", serverPort))
print ("Server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode ("utf-8")
    file = open (sentence, "r")
    con = file.read (2048)
serverSocket.sendto (bytes (con, "utf-8"), clientAddress)
    print ("\n Sent contents of ", end = " ")
    print (sentence)
    #for i in file contents:
        #print (str(i), end = " ")
clientSocket.close()
clientSocket.close()
```

Client UDP. py

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



Serverudp.py

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

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

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

```
    file.close()
```

Clienttudp.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name:  ")

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))

filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
# for i in filecontents:
    # print(str(i), end = '')
clientSocket.close()
clientSocket.close()
```

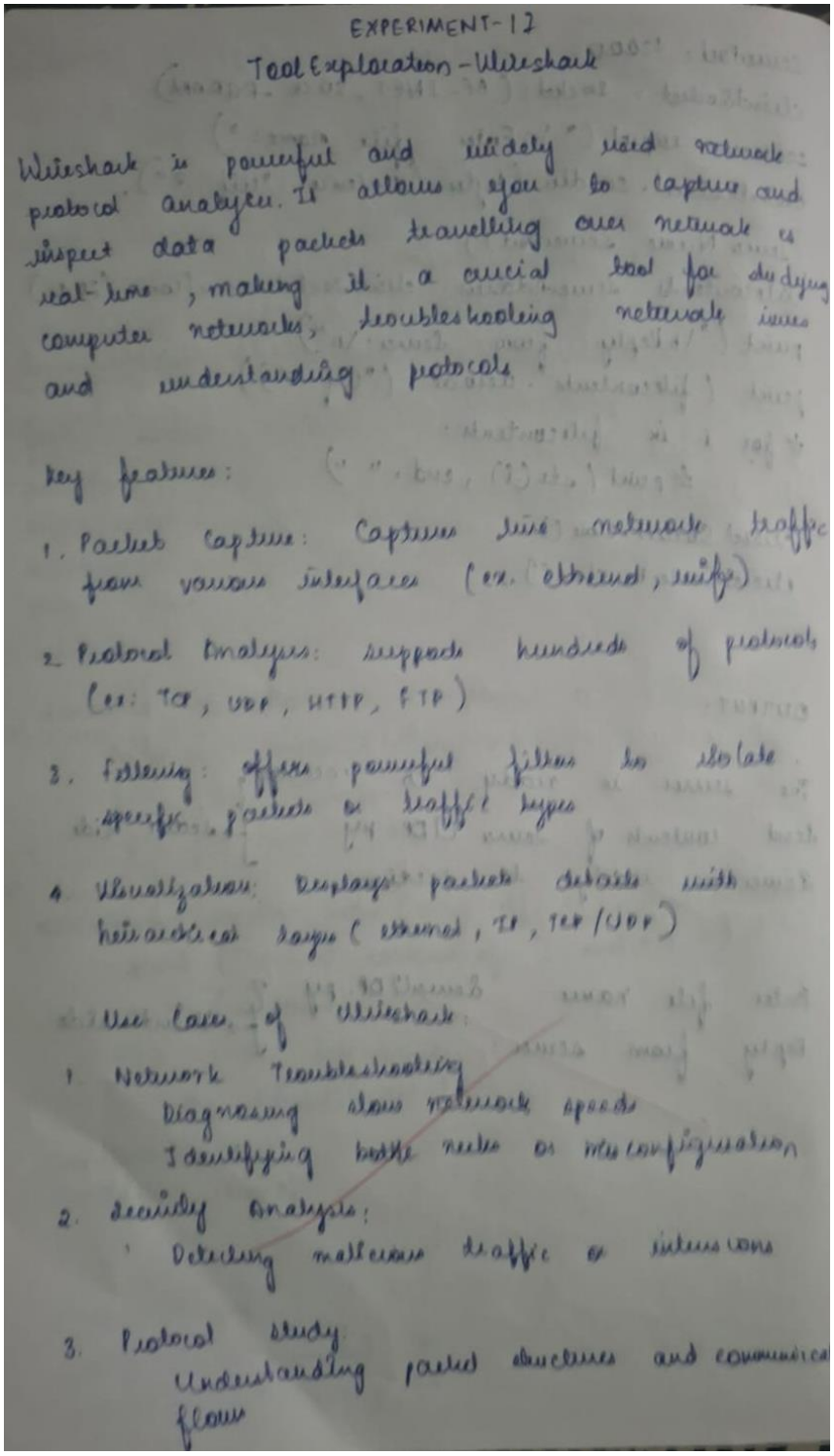
Output:

```
PS C:\Users\Dell\OneDrive\Desktop\code> python udpserver.py
The server is ready to receive

Sent contents of  example.txt
```

```
PS C:\Users\Dell\OneDrive\Desktop\code> python udpclient.py

Enter file name:  example.txt

Reply from Server:

Hello, this is a sample file.
It is used for testing the TCP server.
```

Program 5:

Tool Exploration –Wireshark



EXPERIMENT-12

Tool Exploration - Wireshark

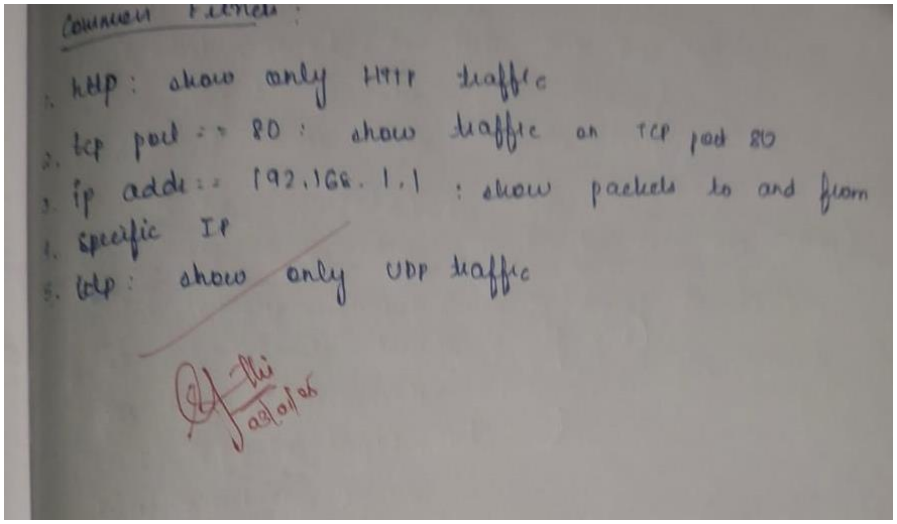
Wireshark is powerful and widely used network protocol analyzer. It allows you to capture and inspect data packets travelling over network in real-time, making it a crucial tool for studying computer networks, troubleshooting network issues and understanding protocols.

Key features:

1. Packet Capture: Captures live network traffic from various interfaces (ex. ethernet, wifi)
2. Protocol Analysis: supports hundreds of protocols (ex: TCP, UDP, HTTP, FTP)
3. Filtering: offers powerful filters to isolate specific packets or traffic types
4. Visualization: Displays packet details with hierarchical layers (ethernet, IP, TCP/UDP)

Use Cases of Wireshark:

1. Network Troubleshooting
   Diagnosing slow network speeds
   Identifying bottle necks or misconfiguration
2. Security Analysis:
   Detecting malicious traffic or intrusions
3. Protocol study
   Understanding packet structures and communication flows


Common Packets:
1. http: show only HTTP traffic
2. tcp port == 80 : show traffic on TCP port 80
3. ip addr == 192.168.1.1 : show packets to and from
4. specific IP
5. udp: show only UDP traffic