VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

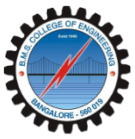
LAB REPORT
on

COMPUTER NETWORKS

Submitted by

RICHA RAGHAVENDRA (1BM21CS164)

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" carried out by **RICHA RAGHAVENDRA(1BM21CS164)** 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.

Prof. Swathi Sridharan
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Index

Sl. No.	Date	Experiment Title	Page No.
		CYCLE-I	
1.	15/06/2023	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message	5-10
2.	22/06/2023	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	11-14
3.	13/07/2023	Configure default route, static route to the Router	15-16
4.	13/07/2023	Configure DHCP within a LAN and outside LAN.	21-24
5.	20/07/2023	Configure RIP routing Protocol in Routers	25-27
6.	20/08/2023	Configure OSPF routing protocol	28-30
7.	27/07/2023	Demonstrate the TTL/ Life of a Packet	31-29
8.	03/08/2023	Configure Web Server, DNS within a LAN.	34-36
9.	10/08/2023	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	37-42
10.	10/08/2023	To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.	43-46
11.	10/08/2023	To construct a VLAN and make the PC's communicate among a VLAN	45-48
12.	10/08/2023	To construct a WLAN and make the nodes communicate wirelessly	51-55
		CYCLE-II	
13.	15/6/2023	Write a program for error detecting code using CRC-CCITT (16-bits).	56-60
14.	15/6/2023	Write a program for congestion control using Leaky bucket algorithm	62-66
15.	15/6/2023	Using TCP/IP sockets, write a client-server program to make client sending the file name	67-70

		and the server to send back the contents of the requested file if present	
16.	15/6/2023	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	71-75

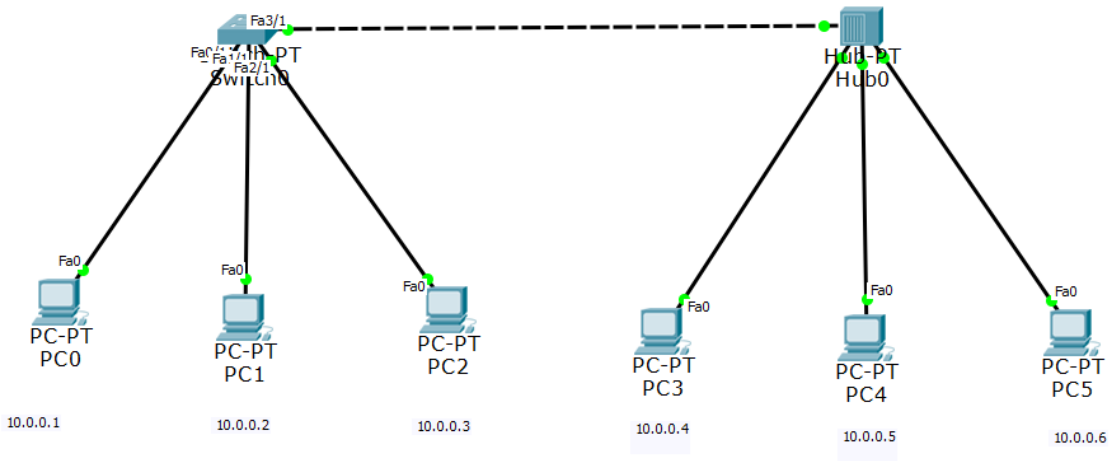
CYCLE 1:

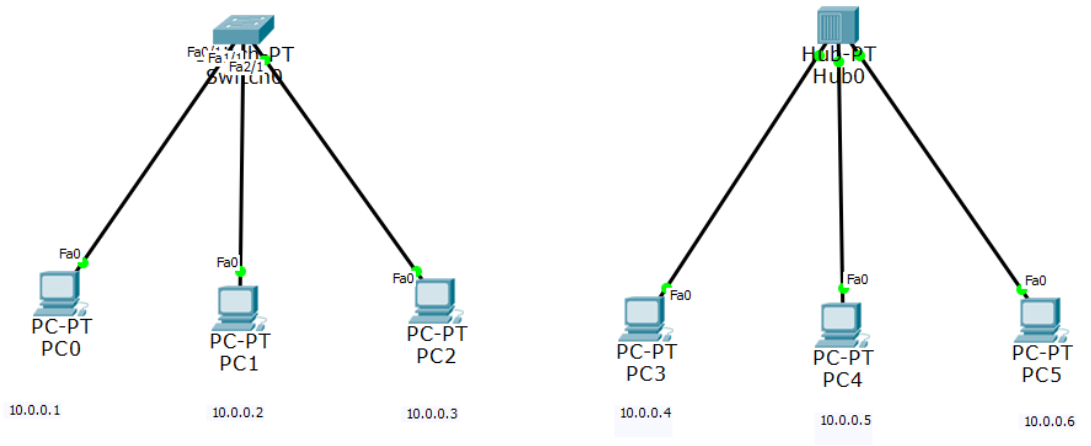
Experiment 1:

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

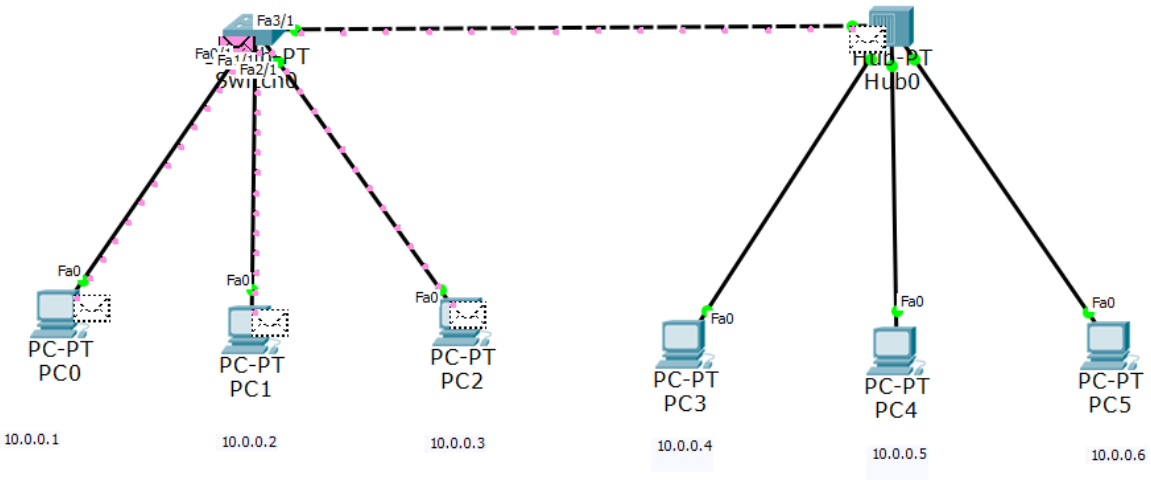
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message

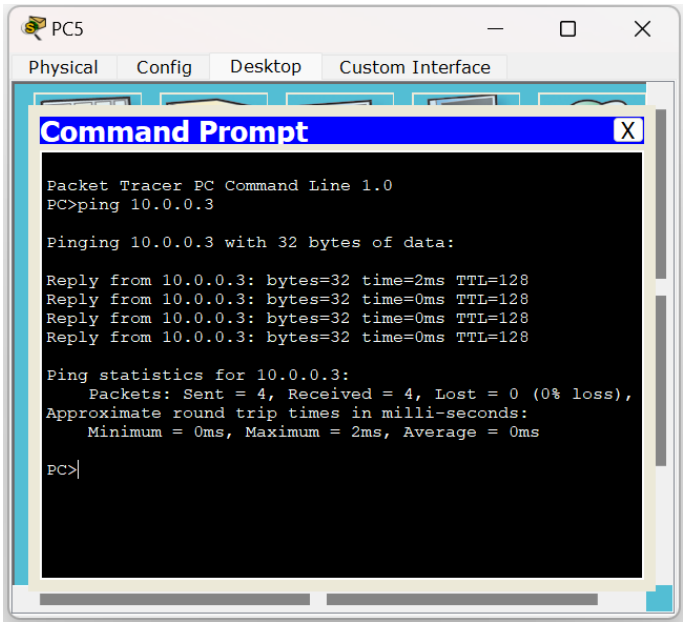
Topology:

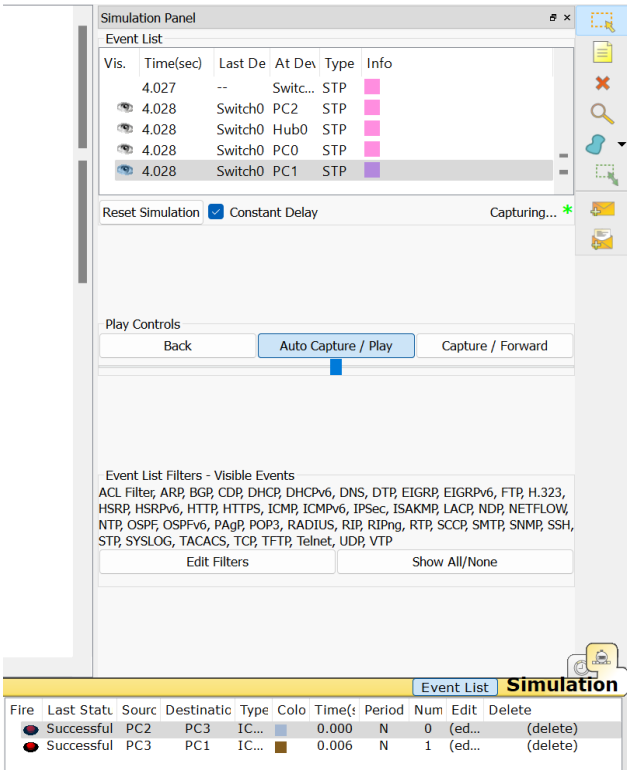



Hub-PT
Hub0
Fa0
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
10.0.0.5
10.0.0.6

Output:


Fa3/1
Fa2/1
Hub-PT
Hub0
Fa0
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
10.0.0.5
10.0.0.6





Observation:

EXPERIMENT - 1

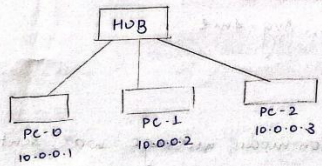
Create a topology & simulate sending a simple PDU from source to destination using hub & switch as connecting devices & demonstrate ping message.

AIM:

To make & understand topology using hub, switch & hybrid topology using both.

Network with Hub:

Topology:



PROCEDURE:

i) Select end users, here PC & generic hub.
ii) Click on end user → config → fast ethernet → assign IP address
iii) Send a simple PDU message from PC-0 to PC-2 in the simulation mode.
iv) In real-time mock click on PC-0 & under desktop, cmd, ping another PC.

RESULT:

(i) The simple PDU is sent from PC-0 to hub
(ii) Hub sends PDU to all ports except the incoming port. The PDU is rejected by other ports except destination port.

→

(iii) PC-2 sends an acknowledgement to hub
(iv) PC0 recieves this & transfer is completed

>> Ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data

Reply from 10.0.0.3 bytes=32 time=4ms TTL=128
Reply from " "
Reply from " "
Reply from 10.0.0.3 bytes=32 time=4ms TTL=128

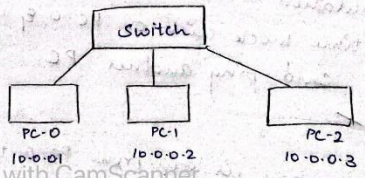
Ping statistics from 10.0.0.3

Packets sent = 4, received = 4, lost = 0.
Approx. round trip times in 4ms min = 4ms
max = 4ms, Avg = 4ms

OBSERVATION:

In simulation mode, message was sent from source to hub, this hub forwards the message to all devices connected w/ it. Only the destination device responds & receives the message whereas as all other devices reject it.

TOPOLOGY W/ SWITCH:



Procedure:

i) Add a generic switch & connect 3 PC's to it using copper straight through wire.

ii) Wait for PC & switch connections to be established

iii) Set IP address of all end devices by clicking on end user → config → fast ethernet → assign IP address.

iv) Send a simple PDU request from PC0 to PC2 in simulation mode. In realtime mode ping PC2 from PC0 Desktop → cmd.

Result:

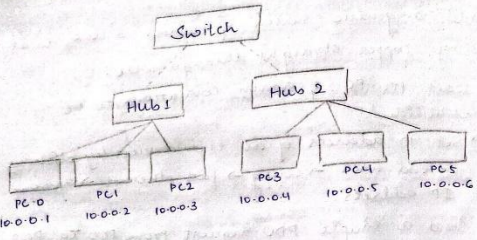
Source - PC0
Destination - PC2
} status successful

ping 10.0.0.3
Reply from 10.0.0.3 bytes = 32 time = 0ms TTL = 128
Reply from "
Reply from "
Reply from 10.0.0.3 bytes=32 time = 0ms TTL = 128
ping statistics for 10.0.0.3
packets : sent = 4 received lost = 0.
Approx. round trip time (ms)
Minimum = 0ms, max = 0ms, avg = 0ms.

OBSERVATION:

The simple PDU is sent from PC-0 to switch. Switch sends this data only to the destination device i.e. PC-2. The destination device acknowledges & receives the data.

HYBRID TOPOLOGY



Procedure:

i) Add generic hubs, generic switch & connect 3 end devices, here PC.

ii) Connect end devices to the 2 diff hubs & then connect 2 hubs to a switch using copper cross wire.

iii) Click on end device → config → fast ethernet → assign IP address

iv) Send PDU from source device to destination & wait for simulation to finish.

v) In real time, ping devices.

Result:

Source : PC-0
Destination : PC-5
} status successful.

Ping 10.0.0.6
Reply from 10.0.0.6 byte = 32 time = 0ms TTL = 128
"
"

Ping statistics for 10.0.0.6
Packets : Sent = 4 received = 4, loss = 0

Observations:

Sending message from PC-0 to PC-5

PC1 is the source node which sends the msg to hub 1. Hub 1 sends the message to other PC's connected w/ it. i.e PC1 & PC2 & also sends the message to the switch. PC1 & PC2 rejects the message & switch sends it to hub 2. which sends it to all end devices. PC5 accepts the msg & acknowledges it.

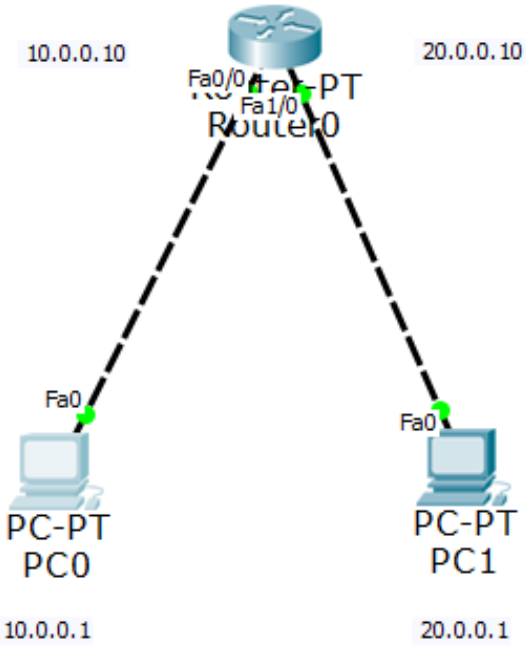
14/8/23.

Experiment 2:

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

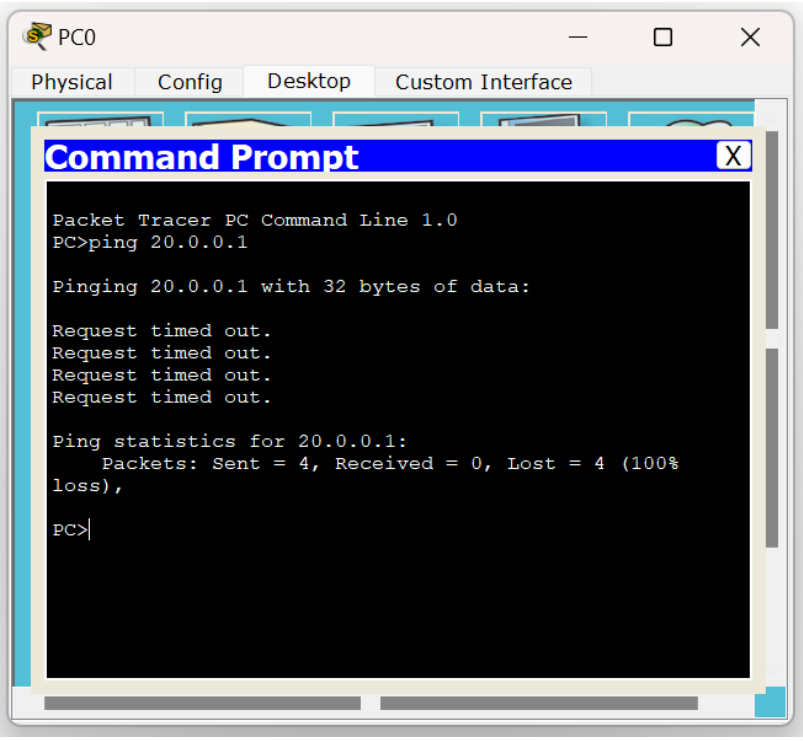
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Topology: (1 router & 2 PC'S)

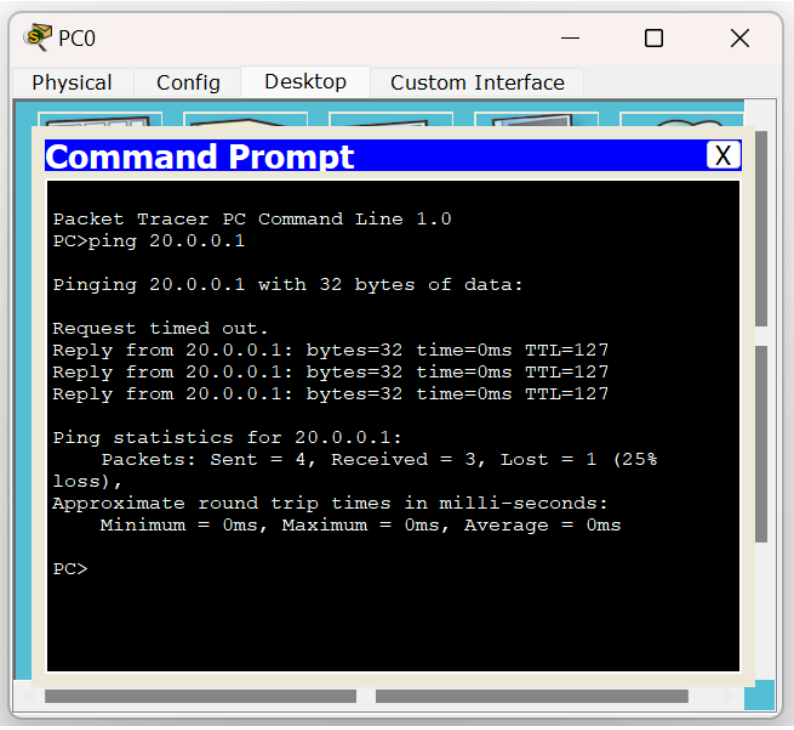


Output:

Before Configuring Gateway:



Before Configuring Gateway:

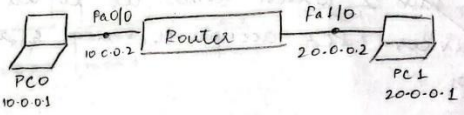


Observation:

EXPERIMENT - 2.

Aim: To configure IP address in router in packet tracer. Explore the following messages: ping responses, destination unreachables, request timed out, reply.

Topology 1:



Procedure:

(i) Use a generic router & connect 2 end devices
(ii) Set the IP addresses as 10.0.0.1 & 20.0.0.1
(iii) Go to CLI & enter 'no'
(v) Type 'enable'
(v) Type 'configure terminal'
'interface Fa0/0'
'ip address 10.0.0.2 255.0.0.0'
'no shutdown'
'exit'

Repeat steps for Fa1/0 interface
Ping PC1 from PC0

vi) Set the output gateway for PC0 as 10.0.0.2 & 20.0.0.2 for PC1
vii) Ping PC1 again for PC0

RESULT

Before setting Default Gateway:

ping 20.0.0.1.

Request timed out

"

"

Ping statistics for 10.0.0.2

Packets sent = 4, received = 0, loss = 4

After setting Default Gateway:

Ping 20.0.0.1

Reply from 20.0.0.1 time = 0ms TTL = 128

"

"

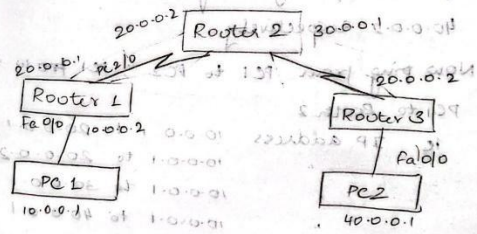
"

Ping statistics for 10.0.0.2

Packets : sent = 4, received = 0; loss = 0.

3 Routers together and an End Device.

Topology:



Procedure:

i) Place 3 generic Routers & 2 end devices. The end devices are connected to router 1 & Router 3. Router 2 is connected to 1 & 3.

ii) Set the IP address of end devices. Here each end device is considered to be a separate network

iii) Go to CLI & configure IP address of 10.0.0.2

iv) Similarly it is configured for fa 2/0 as 20.0.0.1

v) Repeat steps iii & iv for router 2 & 3 as:

Router 1 fa 0/0 → 10.0.0.2
se 0/0 → 20.0.0.2

Router 2 se 2/0 → 20.0.0.2
se 3/0 → 30.0.0.1

Router 3 se 3/0 → 30.0.0.2
fa 0/0 → 40.0.0.2

vi) Now in PC1 & PC2 set the IP address of the output gateway as 10.0.0.2 & 40.0.0.2 respectively

vii) Now ping from PC1 to PC2; PC1 ~~to~~ to R2; PC1 to Router 2.

ie IP address 10.0.0.1 to 20.0.0.1
" 10.0.0.1 to 20.0.0.2
" 10.0.0.1 to 30.0.0.1
" 10.0.0.1 to 40.0.0.1

viii) Write down the ping outputs you get.

RESULT:

a) Ping 20.0.0.2

Request timed out
"
"
"

Ping statistics for 20.0.0.2

Packets: Sent = 9, Received = 0, Loss = 4

b) Ping 20.0.0.1

Reply from 20.0.0.1 time = 0ms
"
"
"

c) ping 30.0.0.1

Reply from 20.0.0.2 Destination host unreachable
"
"
"

Ping statistics from 30.0.0.1

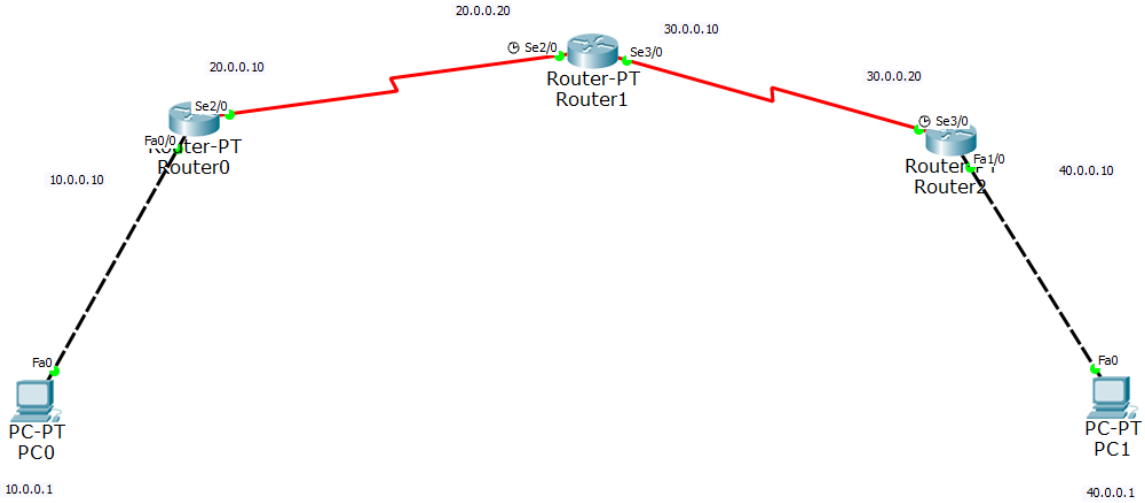
Packets: Sent = 4, Received = 0, lost = 4

Experiment 3:

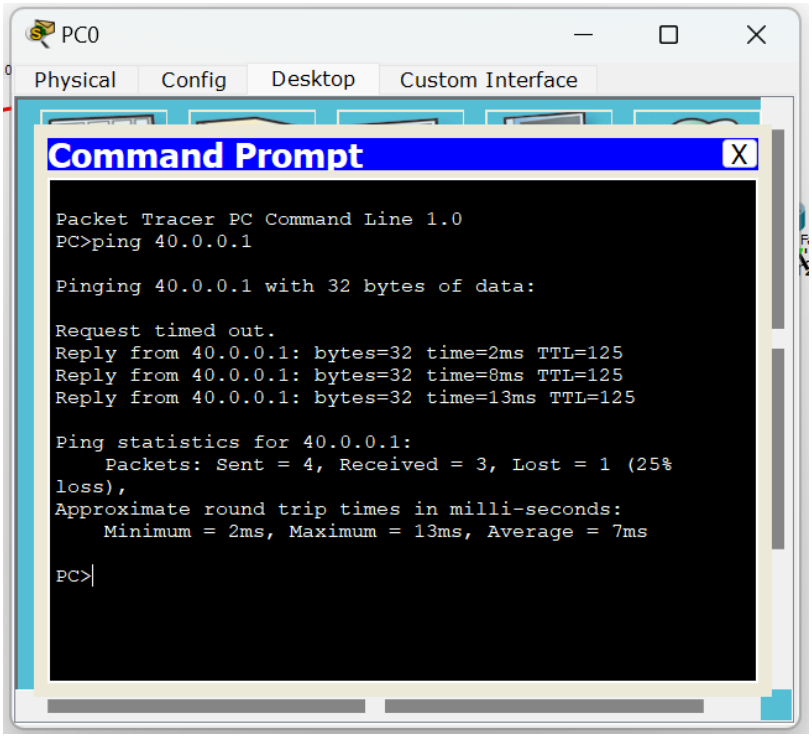
Configure default route, static route to the Router

Aim: Configure default route, static route to the Router

Topology:



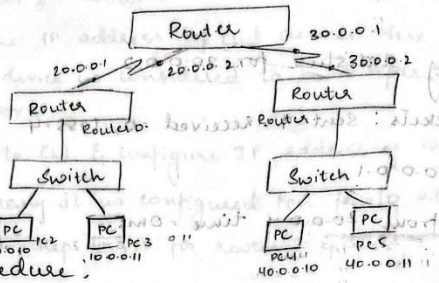
Output:



Observation:

EXPERIMENT - 3

Aim: Configure default route, static route to the router.



Procedure:

Adding Static Routers:

1. To router 1 for networks 20.0.0.0 & 40.0.0.0

```
# ip route 40.0.0.0 255.0.0.0 30.0.0.1
# ip route 20.0.0.0 255.0.0.0 30.0.0.1
```

2. To router 3 for networks 10.0.0.0 & 20.0.0.0

```
# ip route 10.0.0.0 255.0.0.0 30.0.0.2
# ip route 20.0.0.0 255.0.0.0 40.0.0.2
```

3. To router 2 for networks 10.0.0.0 & 30.0.0.0

```
# ip route 30.0.0.0 255.0.0.0 40.0.0.1
# ip route 10.0.0.0 255.0.0.0 40.0.0.1
```

Observation:

The ip routes have been added to each router which can be seen by running 'show ip route' command.

Router 1:

Show ip route

C 10.0.0.0/8 is directly connected, FastEthernet 0/0

S 20.0.0.0/8 [1/0] via 30.0.0.1

C 30.0.0.0/8 is directly connected Serial 2/0

S 40.0.0.0/8 [1/0] via 30.0.0.1

Router 2.

Show ip route

S 10.0.0.0/8 [1/0] via 40.0.0.1

C 20.0.0.0/8 is directly connected, FastEthernet 0/0

S 30.0.0.0/8 is ~~directly connected, Serial 2/0~~ [1/0] via 40.0.0.1

C 40.0.0.0/8 is directly connected, Serial 2/0

Router 3

show ip route

S 10.0.0.0/8 [1/0] via 30.0.0.2

S 20.0.0.0/8 [1/0] via 40.0.0.2

C 30.0.0.0/8 is directly connected Serial 2/0

C 40.0.0.0/8 is directly connected, Serial 3/0

OUTPUT :

The ping requests to all networks are successful

From PC0

→ ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data.

Reply from 20.0.0.1 : bytes = 32 time = 10ms TTL = 125
"
"
"

Ping statistics for 20.0.0.0.1

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

→ ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data

Reply from 40.0.0.2 bytes = 32, time = 7ms TTL = 253
"
"
"

Ping statistics

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

From PC1.

→ ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data

Reply from 30.0.0.2 : bytes = 32 time = 2ms TTL = 253
"
"
"
time = 9ms
time = 6ms
time = 7ms

Ping statistics for 30.0.0.2

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

→ ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data

Reply from 10.0.0.1 : bytes = 32 time = 7ms TTL = 125

"

"

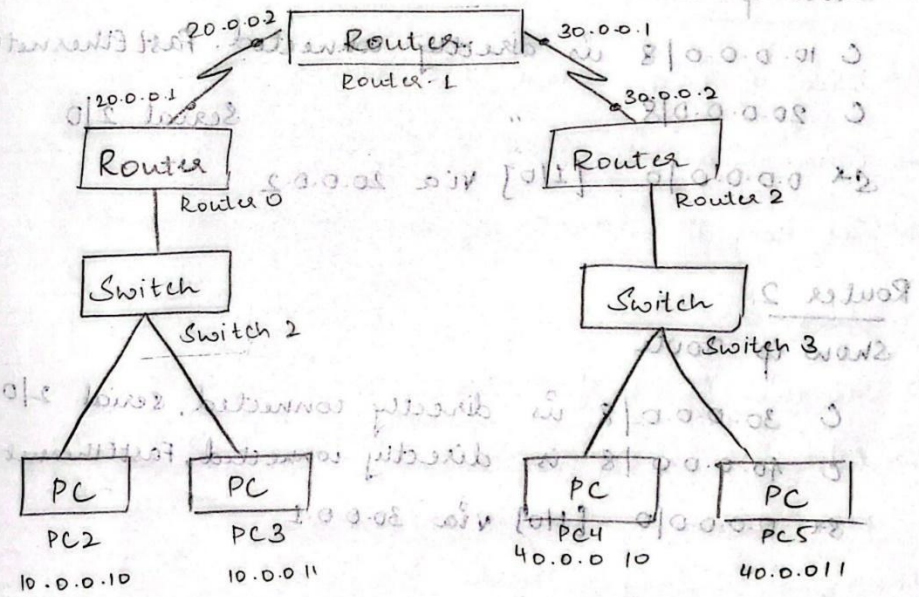
"

Ping statistics for 10.0.0.1

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

Default Routing:

Topology:



Procedure:

- Configuring default route to Router 0 & Router 2

Router 0

ip route 0.0.0.0 0.0.0.0 20.0.0.2

Router 2

ip route 0.0.0.0 0.0.0.0 30.0.0.1

- Configuring 2 Static routes to Router 1

Router 1

ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2

Observation:

The default routes 0 & router 2 & static routes to router 1 have been added

Router 0

show ip route

C 10.0.0.0/8 is directly connected. FastEthernet0/0
C 20.0.0.0/8 " Serial 2/0
S* 0.0.0.0/0 [1/0] via 20.0.0.2

Router 2

show ip route

C 30.0.0.0/8 is directly connected, serial 2/0
C 40.0.0.0/8 is directly connected, FastEthernet
S* 0.0.0.0/0 [1/0] via 30.0.0.1

Router 1

Show ip route

S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected, serial 2/0
C 30.0.0.0/8 " , serial 3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2

OUTPUT

Ping request : From PC5

→ ping 10.0.0.11

Pinging 10.0.0.11 with 32 bytes of data

Reply from 10.0.0.11 : bytes = 32 time = 10ms TTL = 125

" time = 6ms "

" time = 8ms "

" time = 5ms "

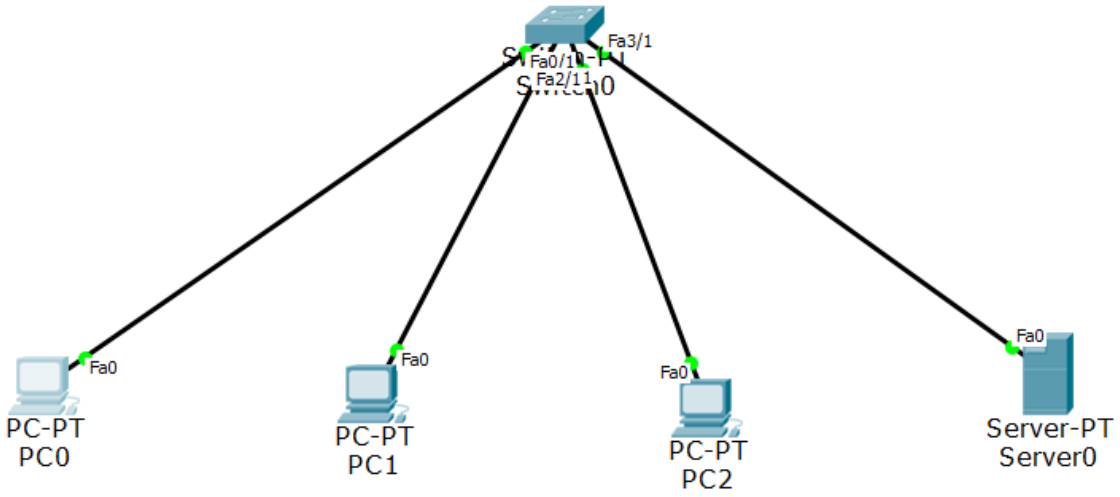
Experiment 4:

Configure DHCP within a LAN and outside LAN.

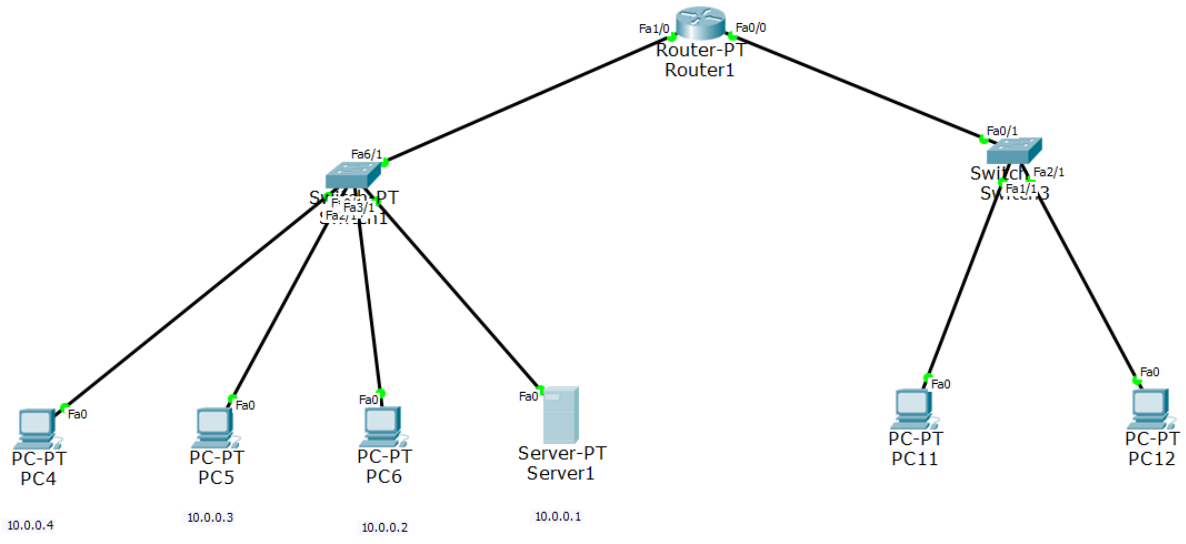
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

(Within a LAN)

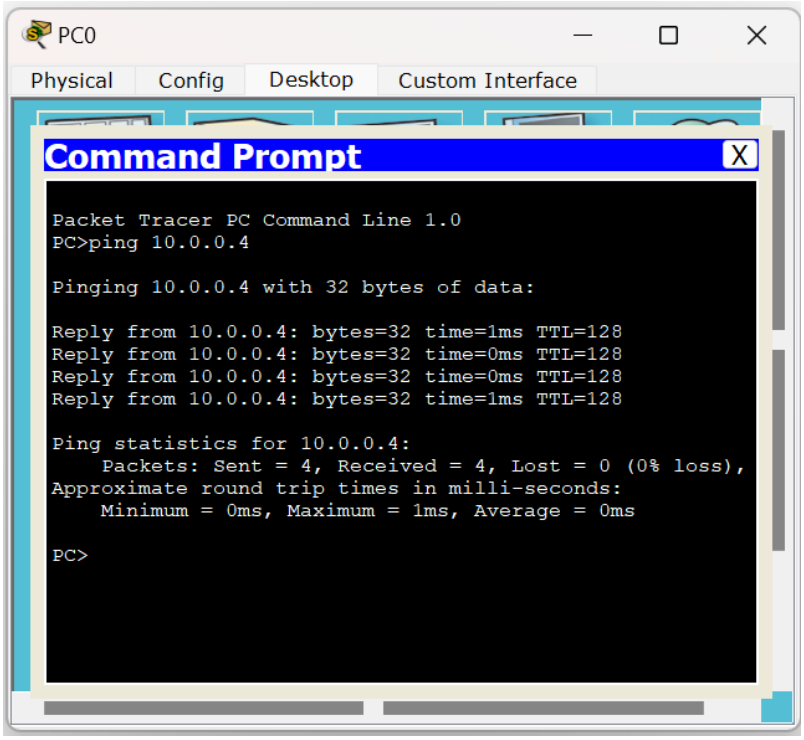


(Outside a LAN)

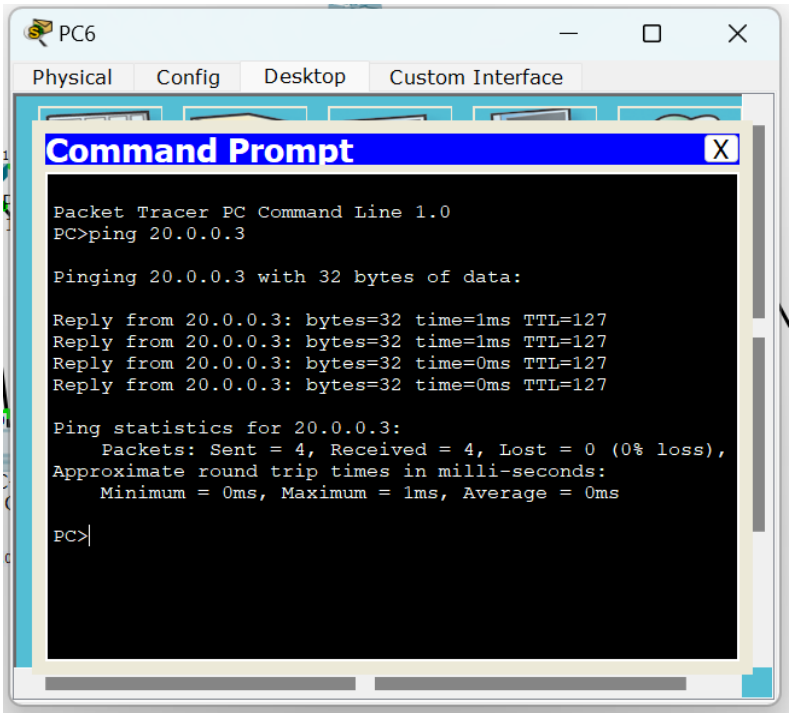


Output:

(Within a LAN)



(Outside a LAN)

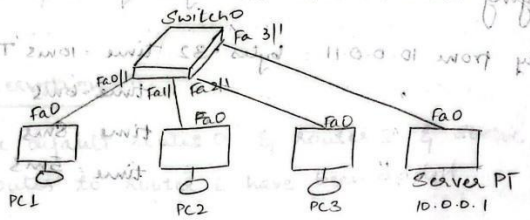


Observation:

EXPERIMENT – 4

Aim: Configure DHCP within a LAN & outside LAN

Topology: Within a LAN



Procedure:

→ Drag & drop 3 PCs, 1 switch & 1 server
→ Connect all the PCs & server to the switch
→ Set the IP address of the server normally
→ Click on server services. Select DHCP.
→ Set the start IP address say 10.0.0.2
→ Add the configuration
→ Now click on each PC & go to config & select IP configuration as DHCP

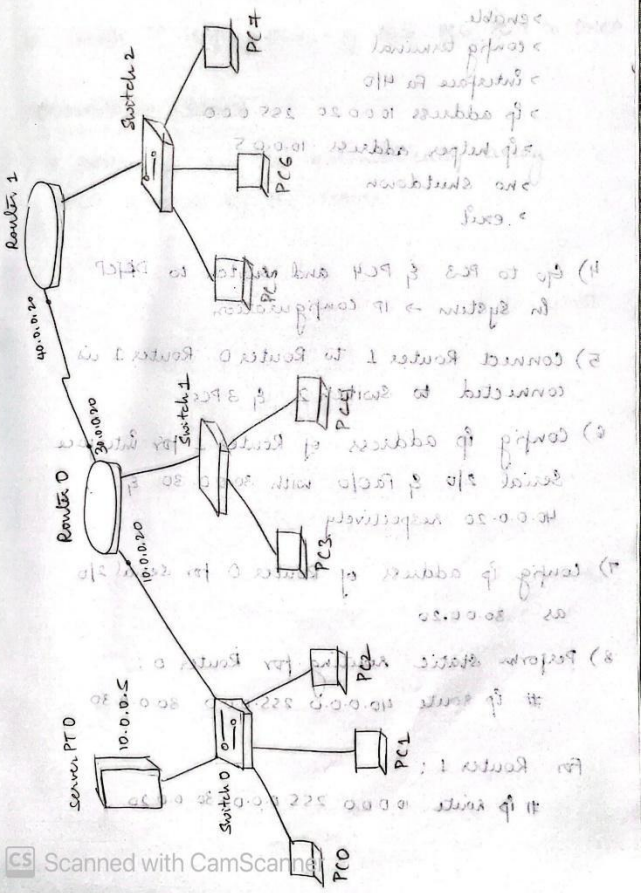
Observation:

→ All the PC's connected to the switch & server gets automatically assigned IP address starting from 10.0.0.2.
→ PC1 is assigned 10.0.0.2
→ PC2 " 10.0.0.3
→ PC3 " 10.0.0.4

Result:

All the PCs are assigned an IP address dynamically

Aim: To configure DHCP outside a LAN



Procedure

1) Repeat the procedures we did for LAN
2) Now add 2 routers, another set of PC & switch 1
3) Config the ip address of ports Fa 4/0 & Fa0/0 of Router 0.

```
>enable
>config terminal
>interface Fa 4/0
>ip address 10.0.0.20 255.0.0.0
>ip helper address 10.0.0.5
>no shutdown
>exit
```

4) Go to PC3 & PC4 and switch to DHCP in System → IP configuration
5) Connect Router 1 to Router 0. Router 1 is connected to switch 2 & 3 PCs
6) Config ip address of Router 1 for interface Serial 2/0 & Fa0/0 with 30.0.0.30 & 40.0.0.20 respectively
7) Config ip address of Router 0 for Serial 2/0 as 30.0.0.20
8) Perform Static routing for Router 0 :

```
# ip route 40.0.0.0 255.0.0.0 30.0.0.30
```

For Router 1 :

```
# ip route 10.0.0.0 255.0.0.0 30.0.0.20
```


9) Go to Server & create 12 more Server pools w/ different names

	Default Gateway	DNS	Start IP	Subnet
Server Pool 1	10.0.0.20	10.0.0.5	10.0.0.10	255.0.0.0
Server Pool 2	10.0.0.20	10.0.0.5	20.0.0.20	255.0.0.0
Server Pool 3	10.0.0.20	10.0.0.5	40.0.0.40	255.0.0.0

10. Switch IP configuration of PC5, PC6, PC7 to DHCP

Observation / Result

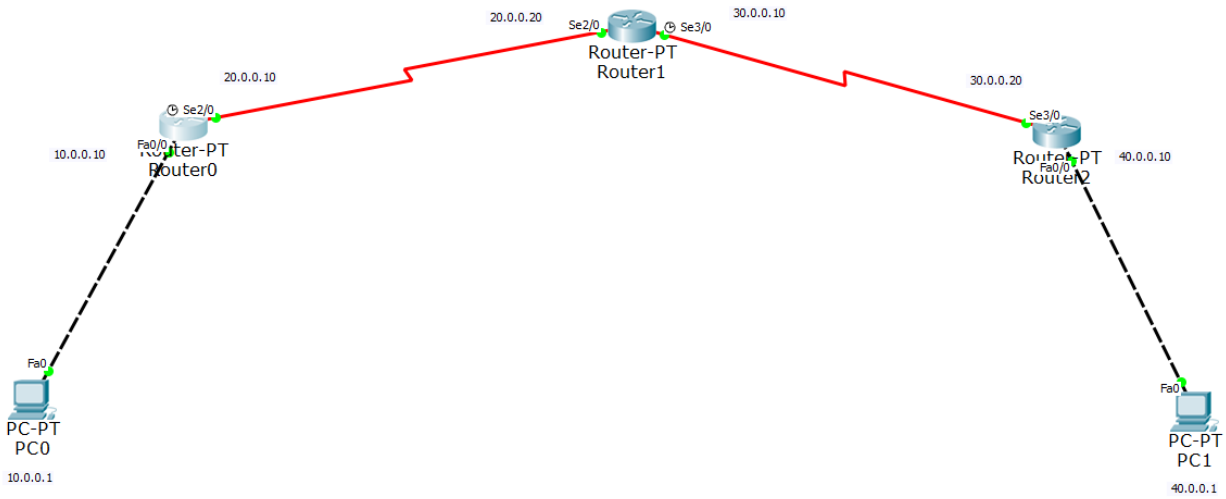
IP Addresses are set automatically using DHCP protocol by the server.

Experiment 5:

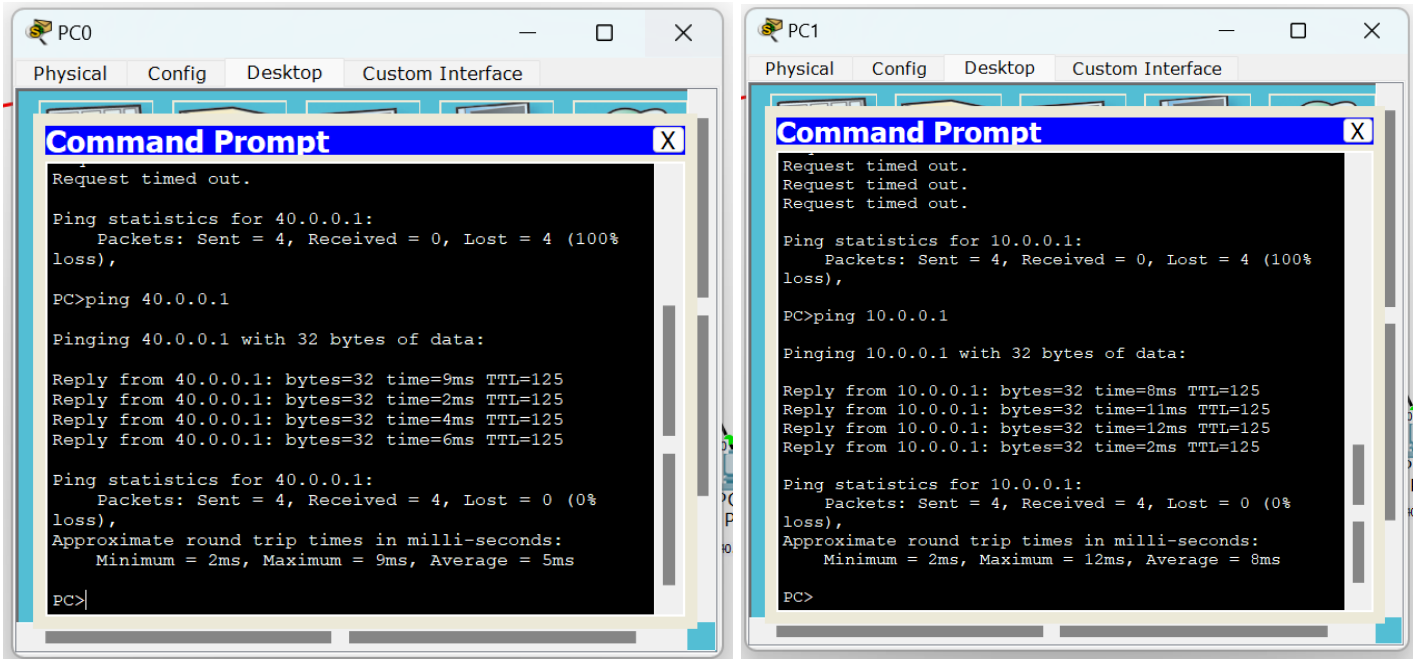
Configure RIP routing Protocol in Routers

Aim: Configure RIP routing Protocol in Routers

Topology:



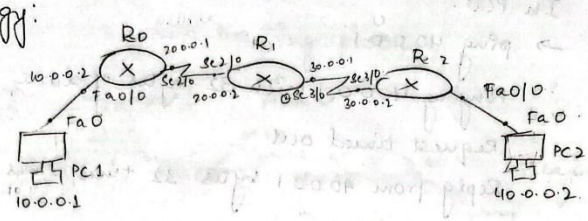
Output:



Observation:

EXPERIMENT – 6

Topology:



Aim: Configure RIP routing protocols in routers.

Procedure:

1. Create a topology with 2PC's, 3 routers
2. Configure PC with ip address & gateway
3. Configure routers:

For R0: enable
config terminal
interface fa0/0
ip address 10.0.0.1 255.00.0
interface se2/0
ip address 20.0.0.1
encapsulation ppp
clock rate 64000
no shut
exit.

4. For R0: enable
show ip route
config terminal
router rip
network 10.0.0.0
network 20.0.0.0.
exit.

Output:

In PC0:

→ 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

"

"

Ping statistics for 40.0.0.1

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

→ ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Reply from 40.0.0.1 bytes = 32 time = 2ms TTL = 125

"

"

Ping statistics for 40.0.0.1

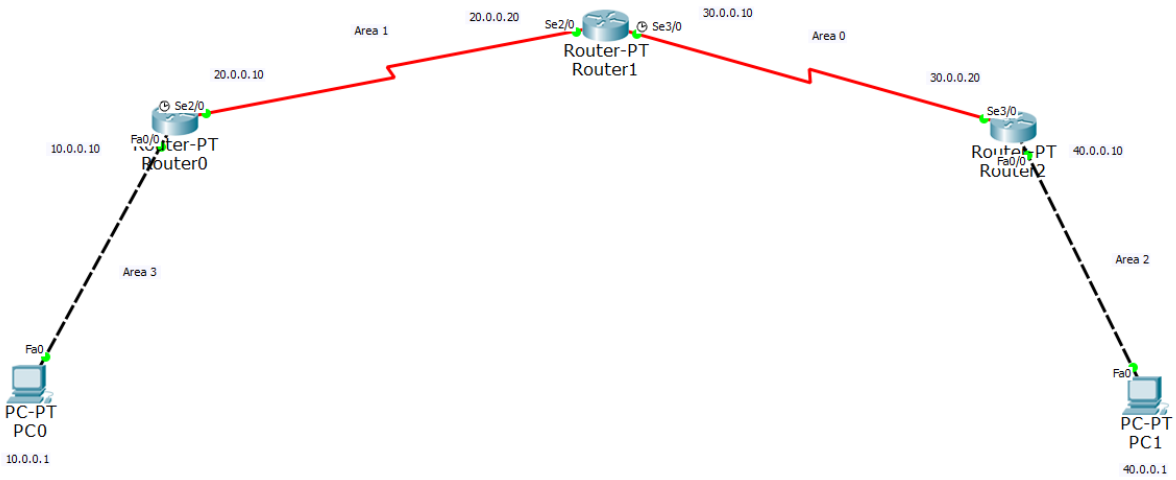
Packets ~~= 4~~ Sent = 4 Received = 4 Lost = 0 (0% lost)

Experiment 6:

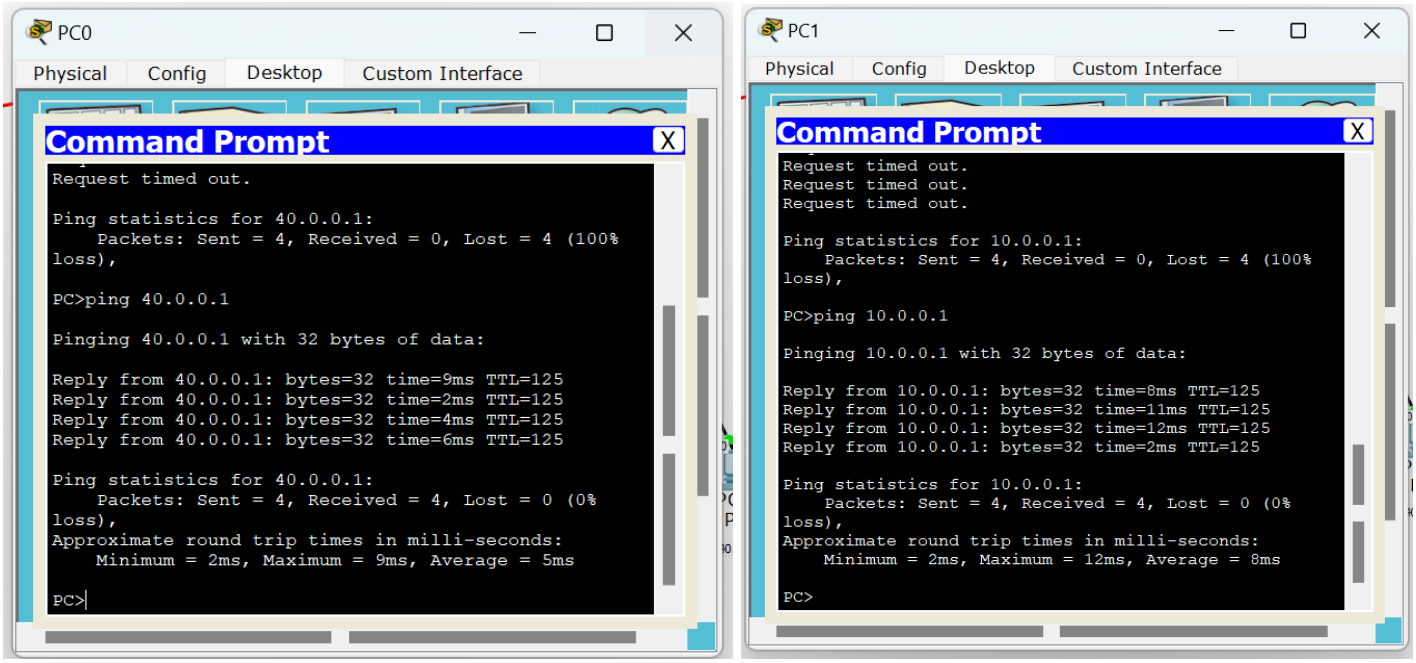
Configure OSPF routing protocol

Aim: Configure OSPF routing protocol

Topology:



Output:

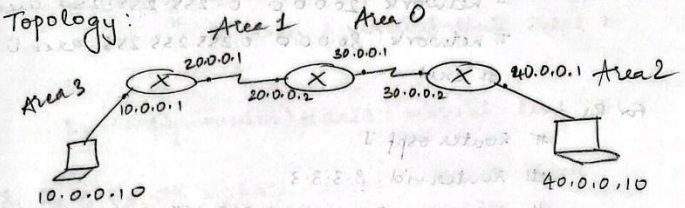


Observation:

EXPERIMENT - 7

Aim: Configure OSPF routing protocol

Topology:



Procedure:

1. Create a topology using 3 routers & 2 PC's
2. Configure ip addresses for all interfaces.

In R1, # Interface fa2/0.
ip address 10.0.0.1 255.0.0.0
no shutdown
exit
Interface se1/0
ip address 20.0.0.1 255.0.0.0
encapsulation ppp
clock state 64000
no shutdown
exit.

Enable ip routing by configuring ospf routing protocol in all routers.

In R1,
R1 (config)# router ospf 1.
R1 (config-router)# 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 R_2,

R_2(config) # router ospf 1

router-id 2.2.2.2

network 20.0.0.0 0.255.255.255 area

network 30.0.0.0 0.255.255.255 area 0

exit

For R_3,

router ospf 1.

router-id 3.3.3.3

network 30.0.0.0 0.255.255.255 area 0

network 40.0.0.0 0.255.255.255 area 3

exit

Router id → identifies router.

4. Check routing table of R_1

Router # show ip route

5. Configure loopback address to routers.

R_1(config-if) # interface loopback 0

ip add 172.16.1.252 255.255.0.0

no shut

R_2(config-if) # interface loopback 0

add 172.16.1.253 255.255.0.0

no shut

R_3(config-if) # interface loopback 0

ip add 172.16.1.254 255.255.0.0

no shut

6. Now, check routing table of R_3

R_3 # show ip route

7. Create virtual link b/w R_1, R_2 to connect area 3 to area 0

R_1 :

R_1 (config) # router ospf 1

area 1 virtual-link 2.2.2.2

R_2 :

R_2 (config-router) # area 1 virtual-link 1.1.1.1

8. R_2 & R_3 get updates about area 3. Check routing table of R_3

show ip route

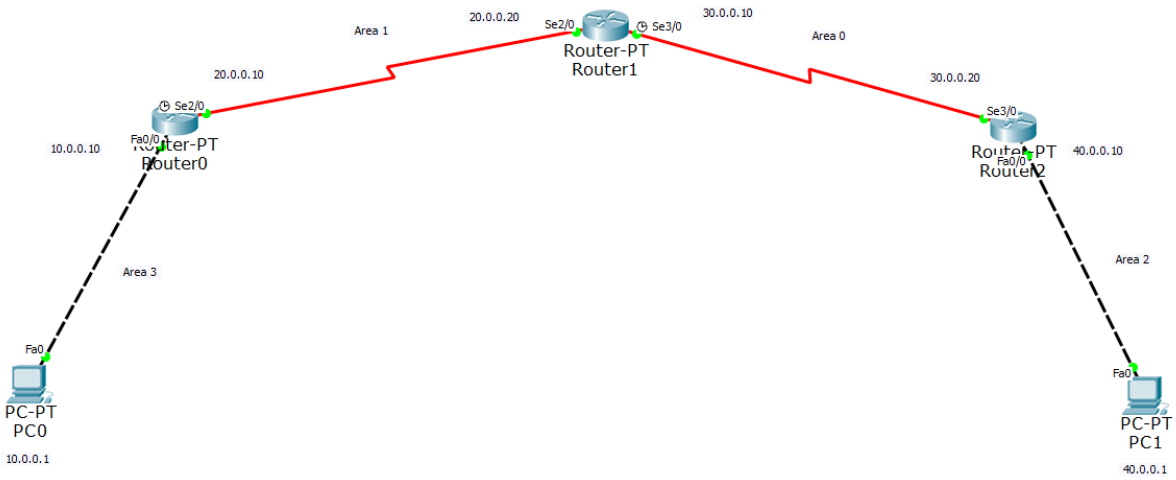
9. Check connectivity b/w host 10.0.0.10 to 40.0.0.10

Experiment 7:

Demonstrate the TTL/ Life of a Packet

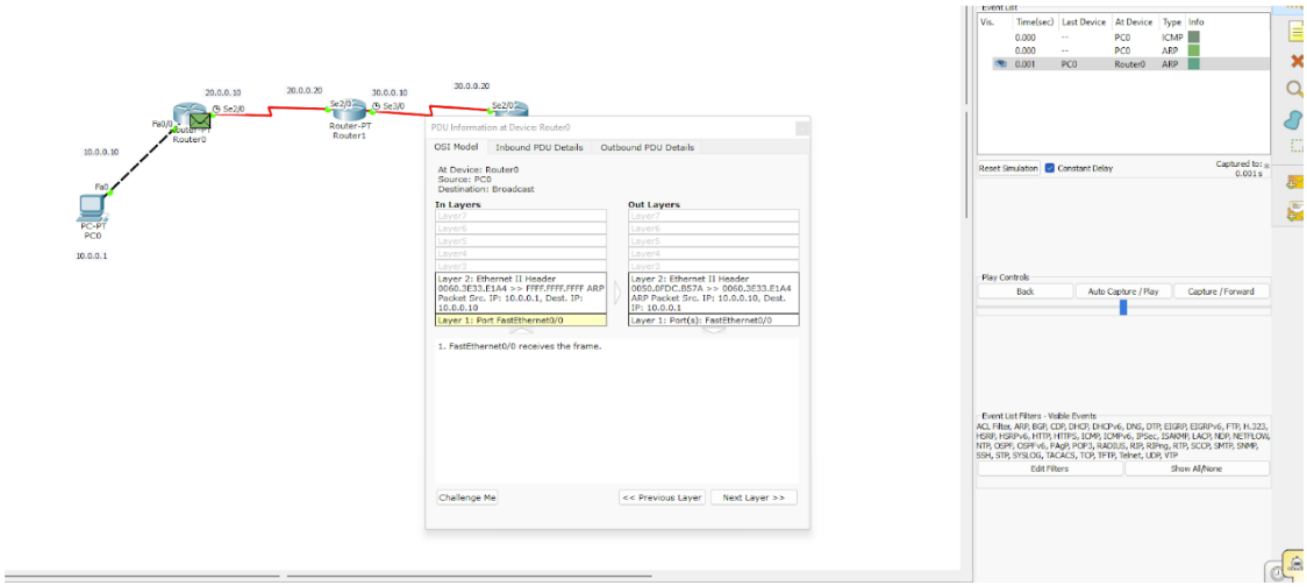
Aim: Demonstrate the TTL/ Life of a Packet

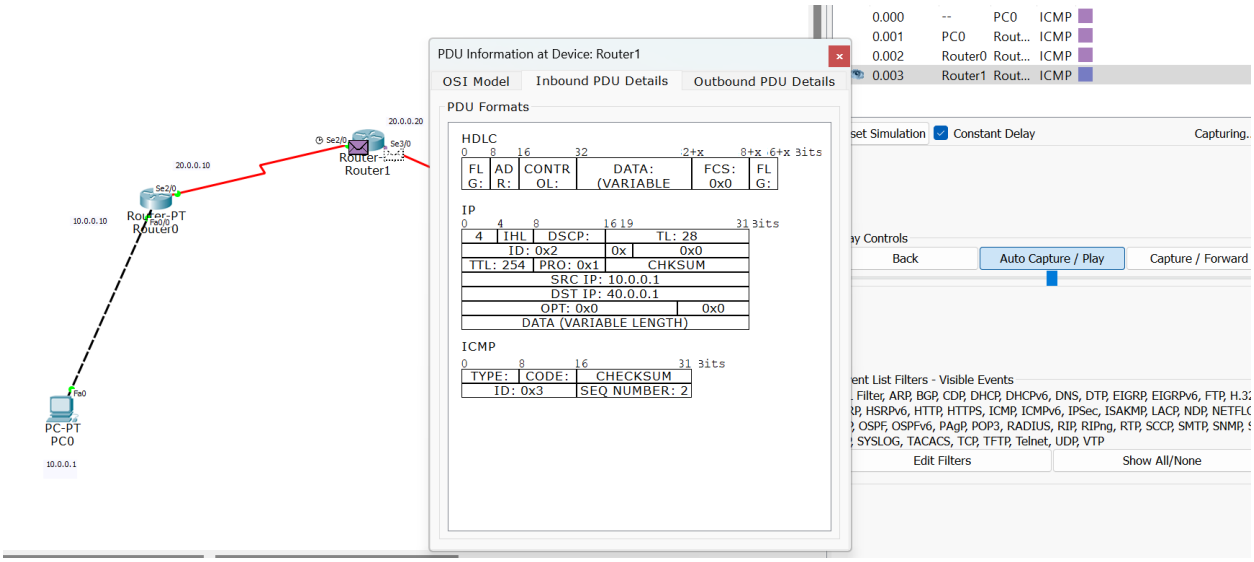
Topology:

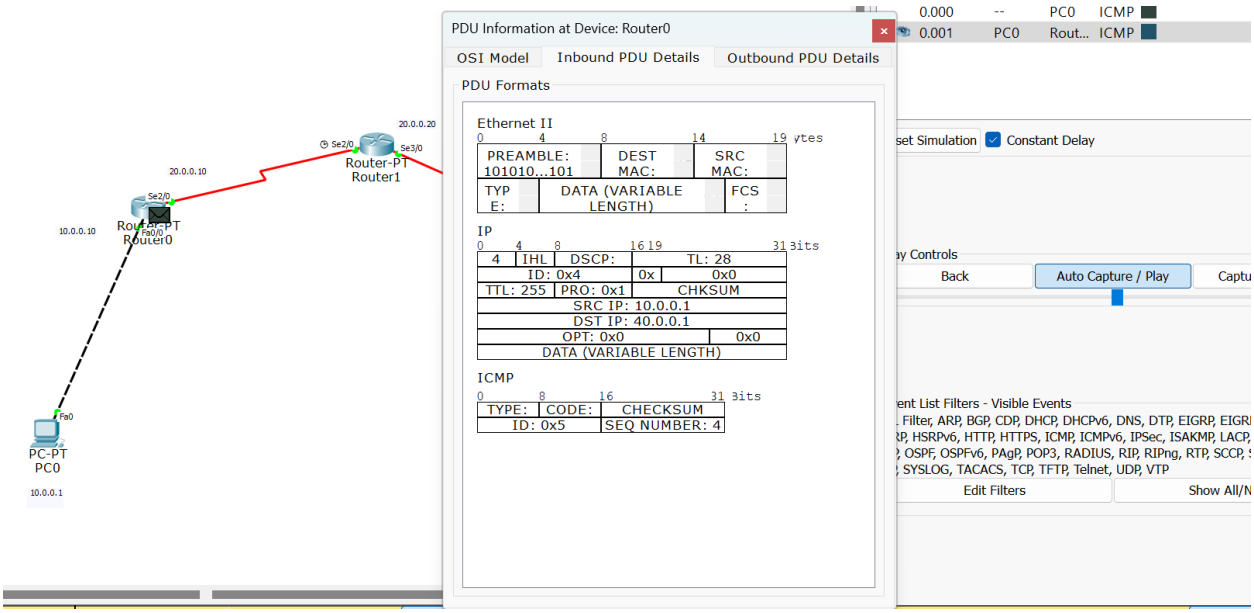


Output:

Simple PDU sent from PC0 to PC1 in simulation mode.




PDU Information at Device: Router1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
HDLC
FLG: | ADR: | CONTROL: | DATA: (VARIABLE | FCS: 0x0 | FLG:
IP
4 | IHL | DSCP: | TL: 28
ID: 0x2 | 0x | 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
TYPE: | CODE: | CHECKSUM
ID: 0x3 | SEQ NUMBER: 2
Router-PT Router1
Router-PT Router0
PC-PT PC0
20.0.0.20
20.0.0.10
10.0.0.10
10.0.0.1
0.000 -- PC0 ICMP
0.001 PC0 Rout... ICMP
0.002 Router0 Rout... ICMP
0.003 Router1 Rout... ICMP
Constant Delay
Capturing.
Back
Auto Capture / Play
Capture / Forward
Edit Filters
Show All/None

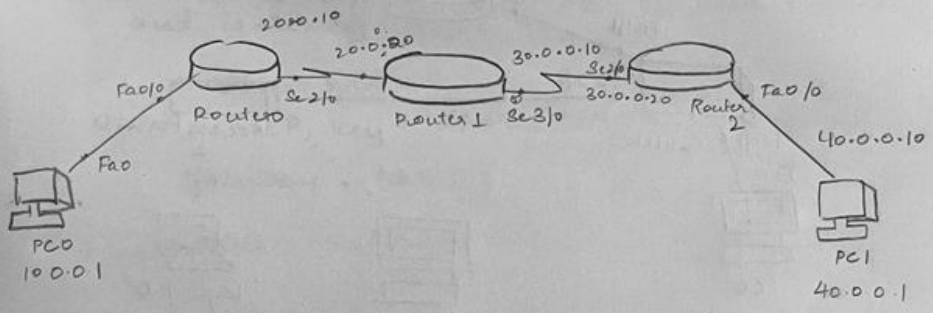

PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...101 | DEST MAC: | SRC MAC:
TYPE: | DATA (VARIABLE LENGTH) | FCS:
IP
4 | IHL | DSCP: | TL: 28
ID: 0x4 | 0x | 0x0
TTL: 255 | PRO: 0x1 | CHKSUM
SRC IP: 10.0.0.1
DST IP: 40.0.0.1
OPT: 0x0 | 0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: | CODE: | CHECKSUM
ID: 0x5 | SEQ NUMBER: 4
Router-PT Router1
Router-PT Router0
PC-PT PC0
20.0.0.20
20.0.0.10
10.0.0.10
10.0.0.1
0.000 -- PC0 ICMP
0.001 PC0 Rout... ICMP
Constant Delay
Back
Auto Capture / Play
Edit Filters

Observation:

EXPERIMENT - 10

Aim: To demonstrate TTL / life of packet

Topology:



Procedure:

1. Create a topology with 2PC's & 3 router
2. Configure the devices as per static / default / dynamic routing
3. In simulation mode, send simple PDU from one PC to another
4. Use the capture button to capture every transfer.
5. Click on the PDU during every transfer to see the inbound & outbound PDU details.

Observation:

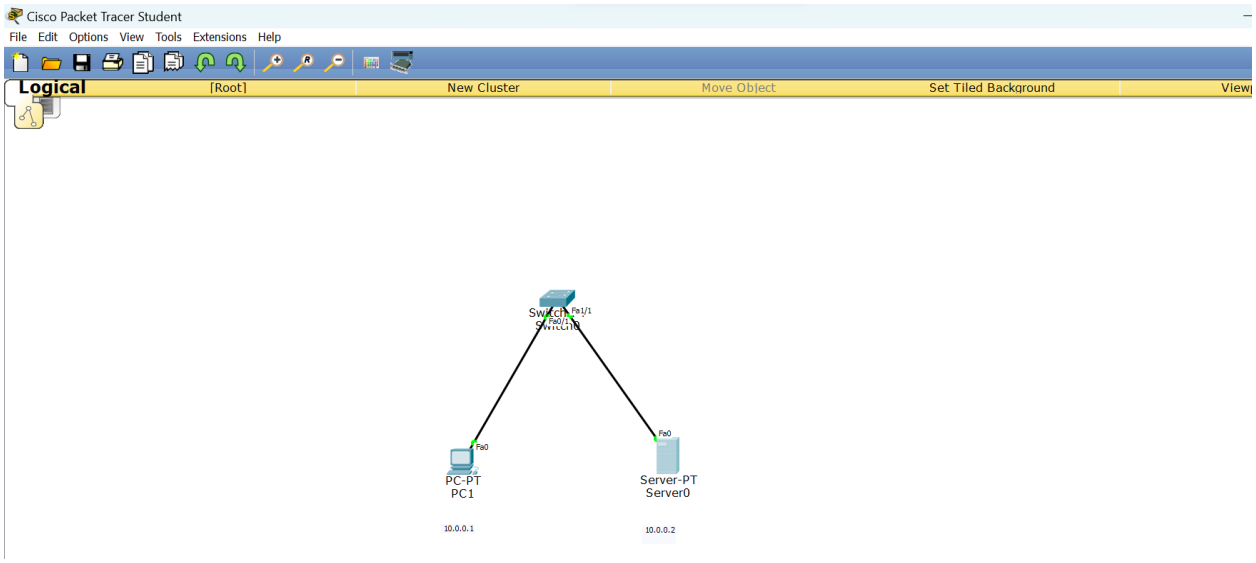
Difference of 1 in TTL when the PDU crosses every router.

Experiment 8:

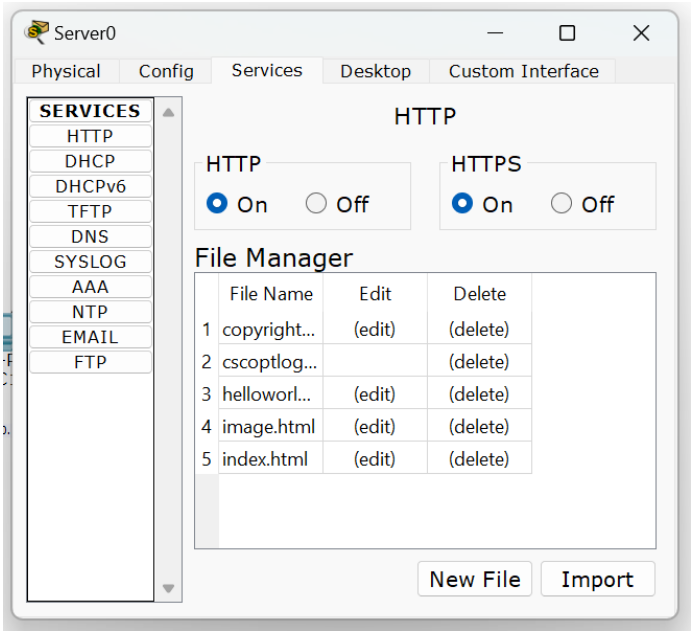
Configure Web Server, DNS within a LAN.

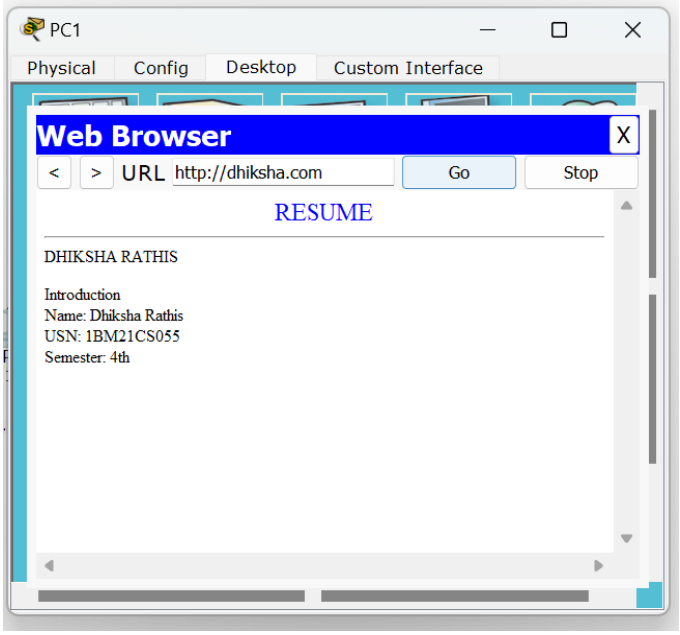
Aim: Configure Web Server, DNS within a LAN.

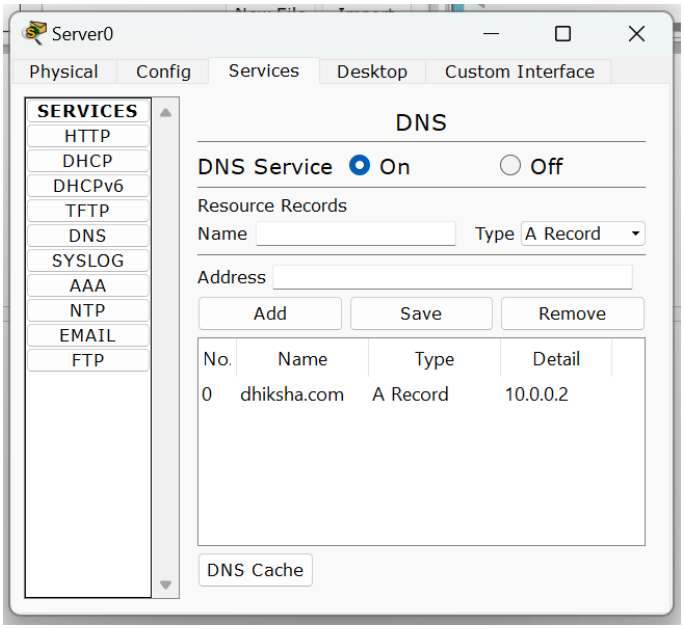
Topology:

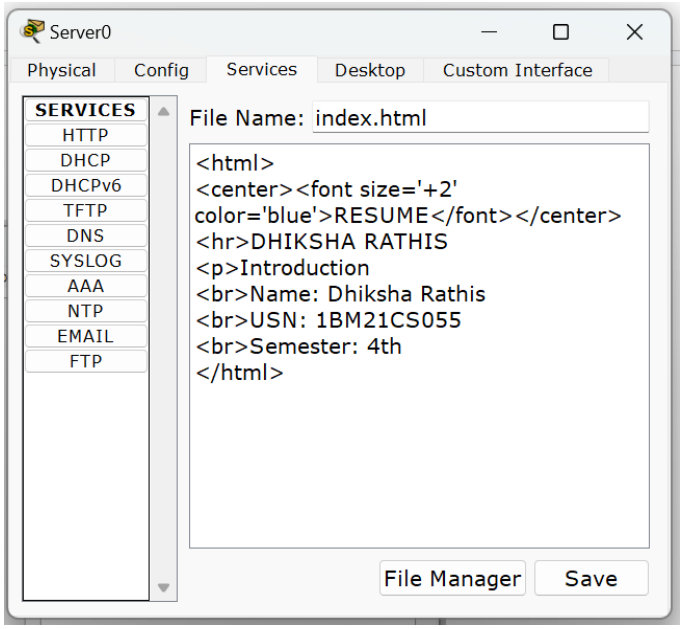


Output:





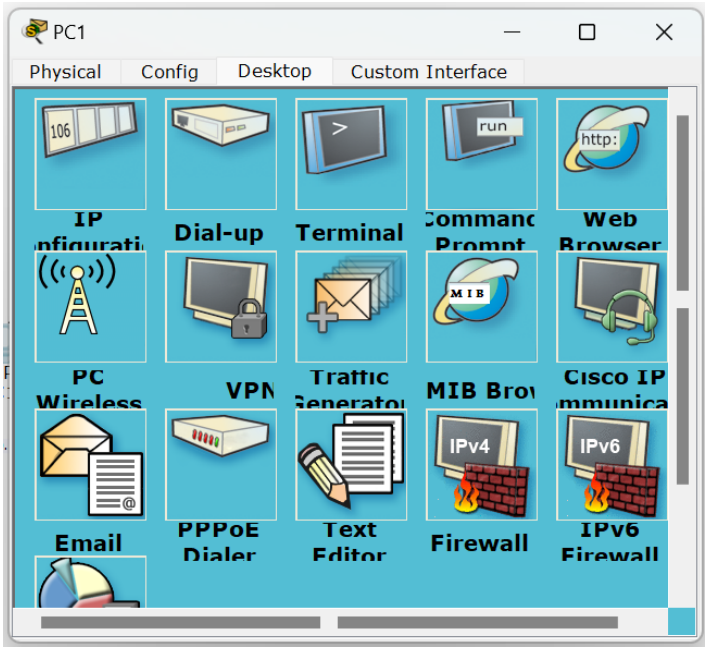

Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
DNS
DNS Service
On
Off
Resource Records
Name
Type
A Record
Address
Add
Save
Remove
No.
Name
Type
Detail
0
dhiksha.com
A Record
10.0.0.2
DNS Cache

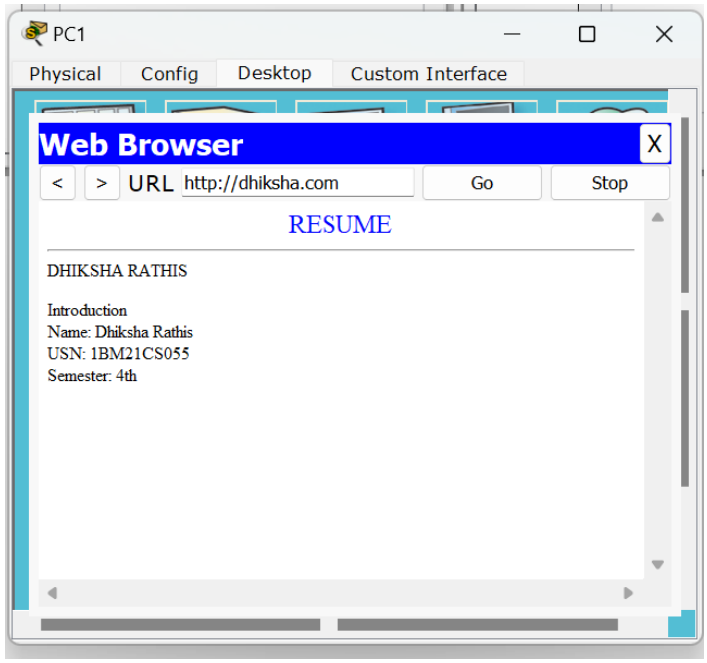

Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
File Name: index.html
<html>
<font size='+2'
color='blue'>RESUME
<hr>DHIKSHA RATHIS
<p>Introduction

Name: Dhiksha Rathis

USN: 1BM21CS055

Semester: 4th
</html>
File Manager
Save


PC1
Physical
Config
Desktop
Custom Interface
IP
Dial-up
Terminal
Command
Prompt
Web
Browser
PC
Wireless
VPN
Traffic
MIB Bro
Cisco IP
mmunica
Email
PPPoE
Dialer
Text
Editor
Firewall
IPv6
Firewall

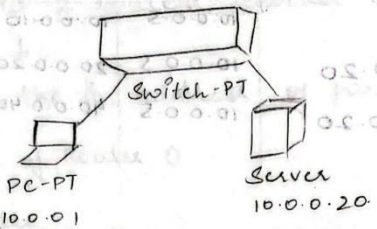

PC1
Physical
Config
Desktop
Custom Interface
Web Browser
X
URL
http://dhiksha.com
Go
Stop
RESUME
DHIKSHA RATHIS
Introduction
Name: Dhiksha Rathis
USN: 1BM21CS055
Semester: 4th

Observation:

EXPERIMENT - 5.

Aim: To configure web server, DNS within a LAN.

Topology:



Procedure:

1. Create topology using PC, server & a switch
2. Configure PC & server (IP addr + Gateway)
3. Open configure DNS of server with name & URL
4. Edit index.html to display name & USN.

Observation:

Web Browser
URL http://richa.com
USN: 1BM21CS164.
Name Richa.

Result:

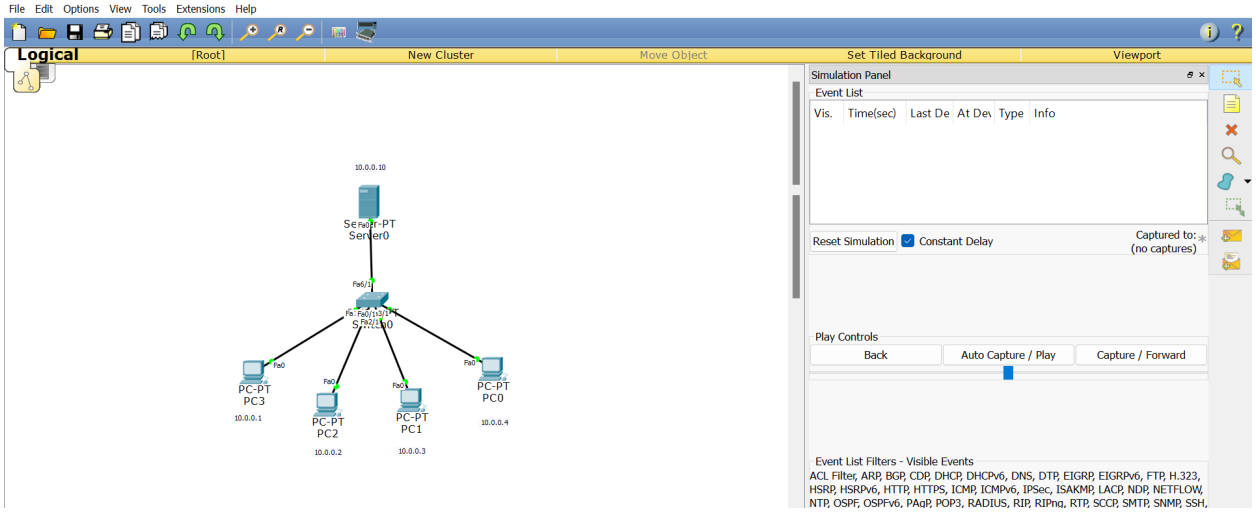
On entering the name of the URL, the specific information set on servers DNS is displayed.

Experiment 9:

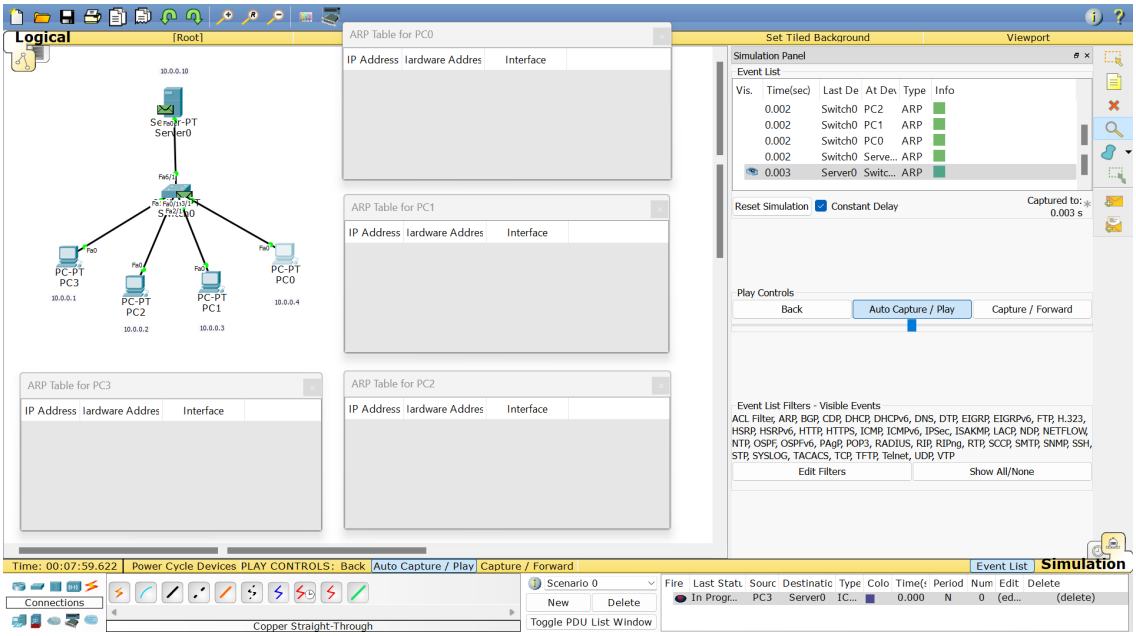
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

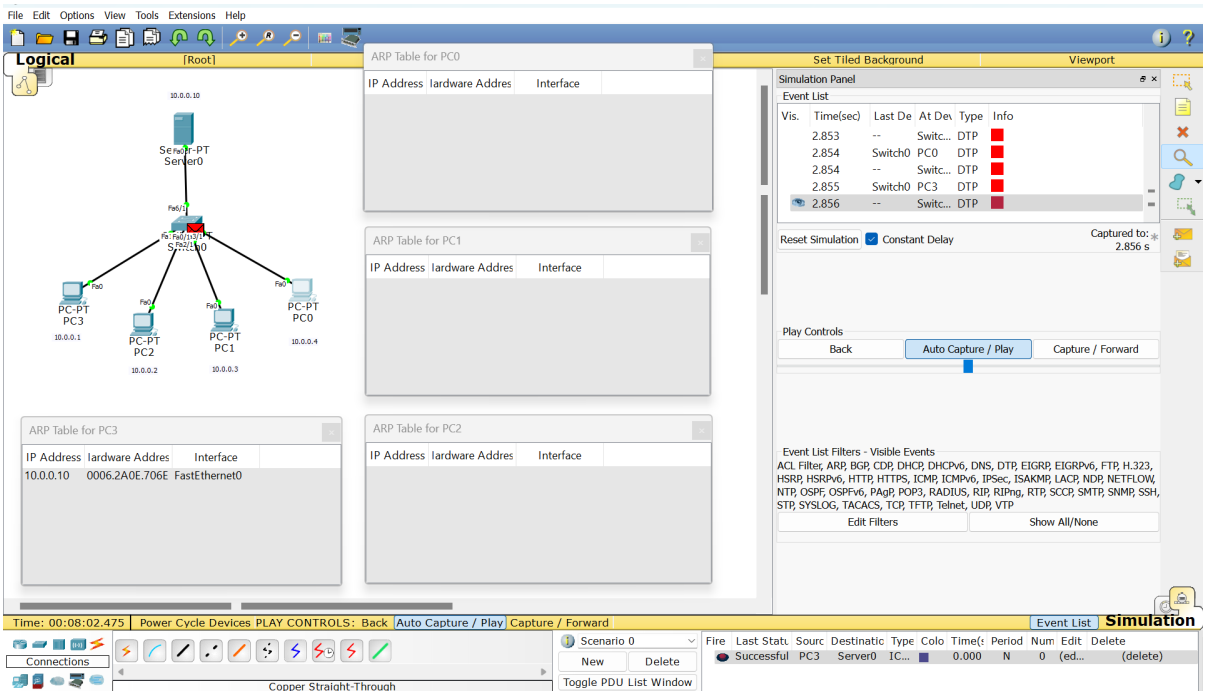
Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

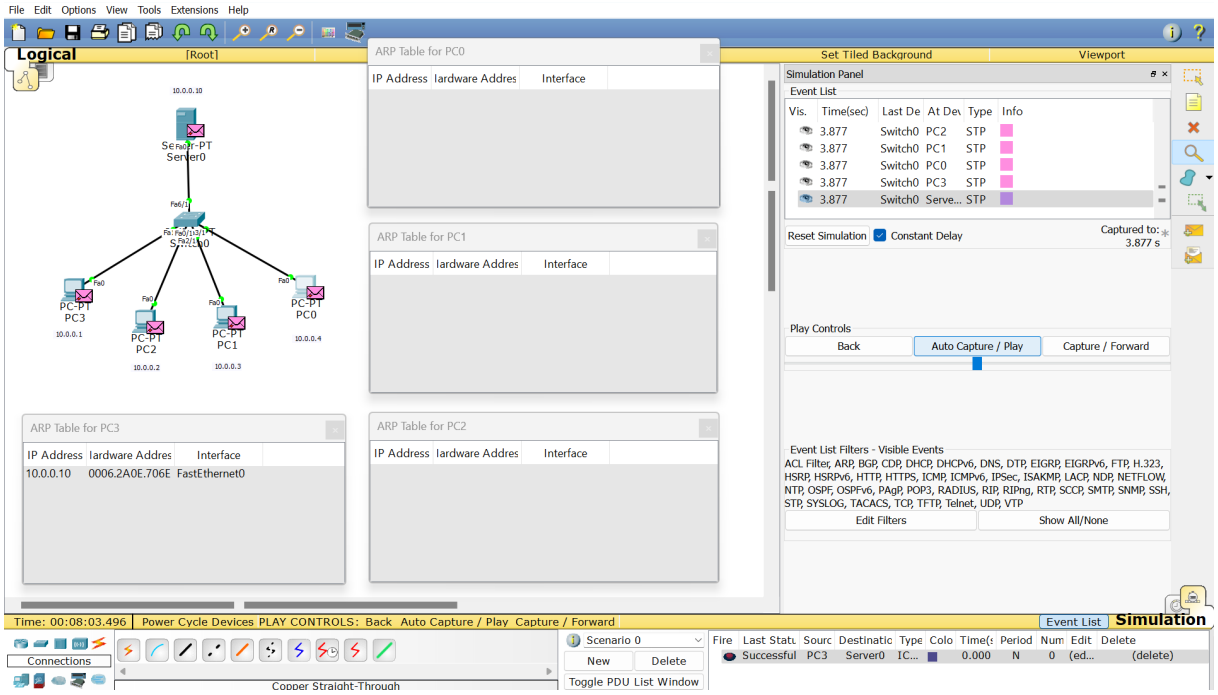
Topology:

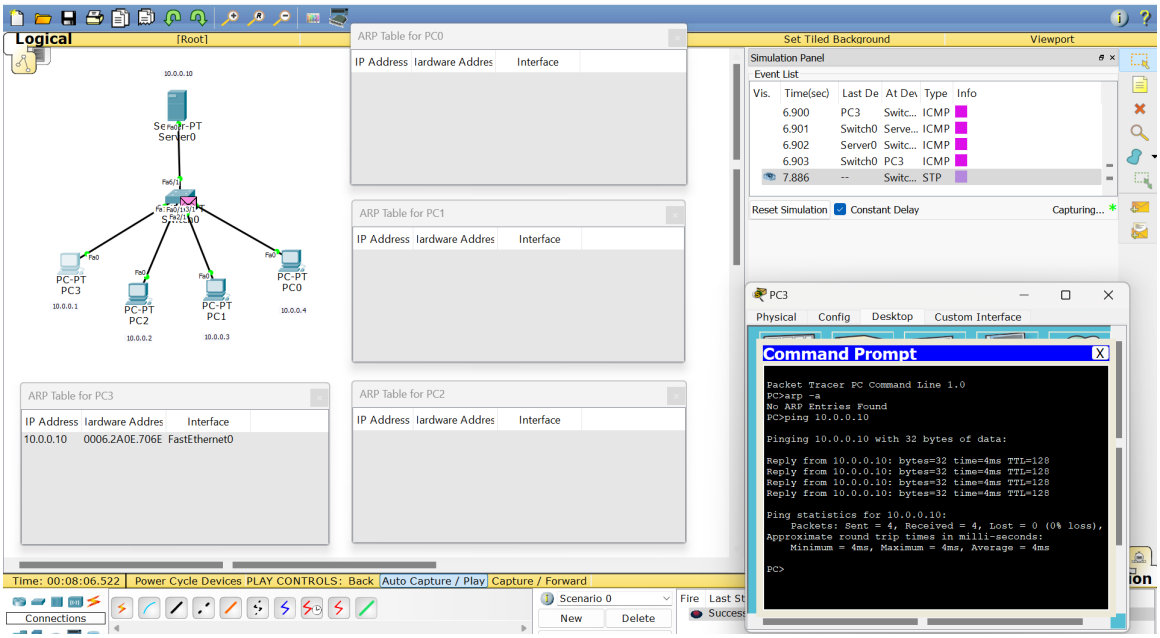


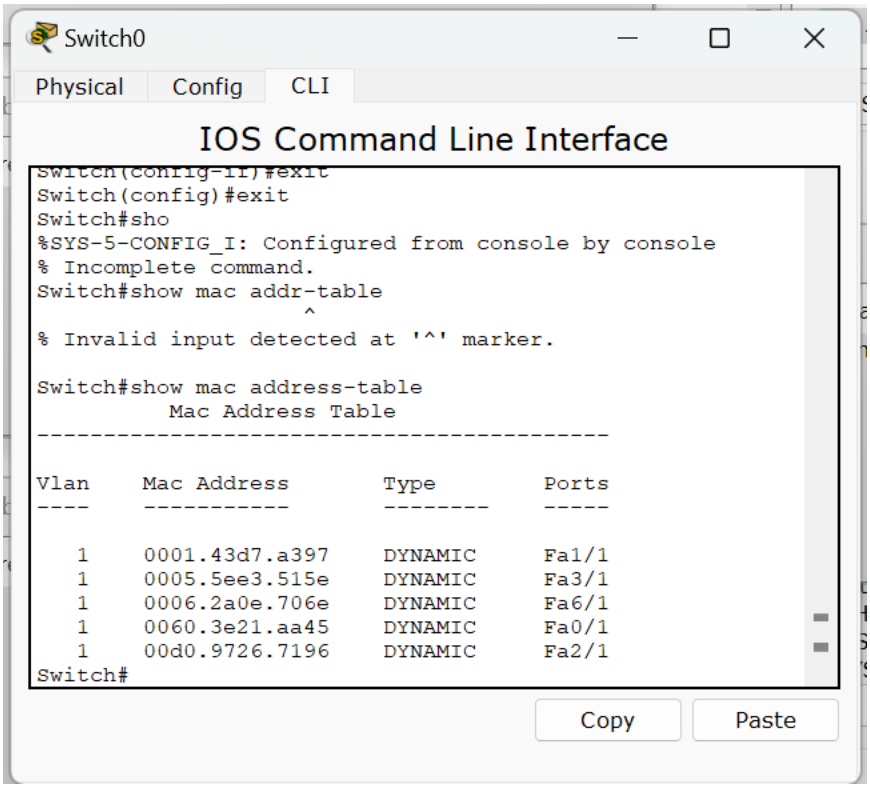
Output:

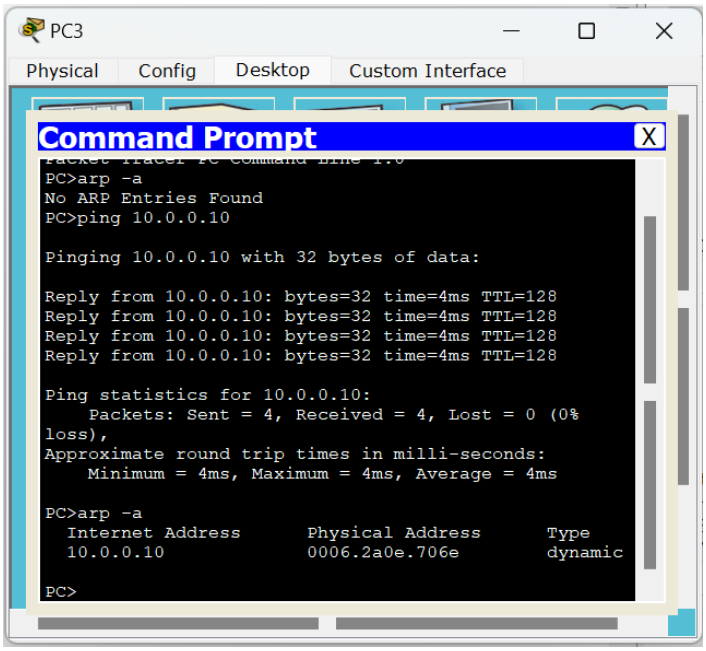

Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
ARP Table for PC1
ARP Table for PC2
ARP Table for PC3
IP Address
Hardware Addres
Interface
Simulation Panel
Event List
Vis.
Time(sec)
Last De
At Dev
Type
Info
0.002 Switch0 PC2 ARP
0.002 Switch0 PC1 ARP
0.002 Switch0 PC0 ARP
0.002 Switch0 Serve... ARP
0.003 Server0 Switc... ARP
Reset Simulation
Constant Delay
Captured to: 0.003 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:07:59.622
Power Cycle Devices
PLAY CONTROLS: Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Connections
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Statu
Sourc
Destinatic
Type
Colo
Time(s
Period
Num
Edit
Delete
In Progr... PC3 Server0 IC... 0.000 N 0 (ed... (delete)

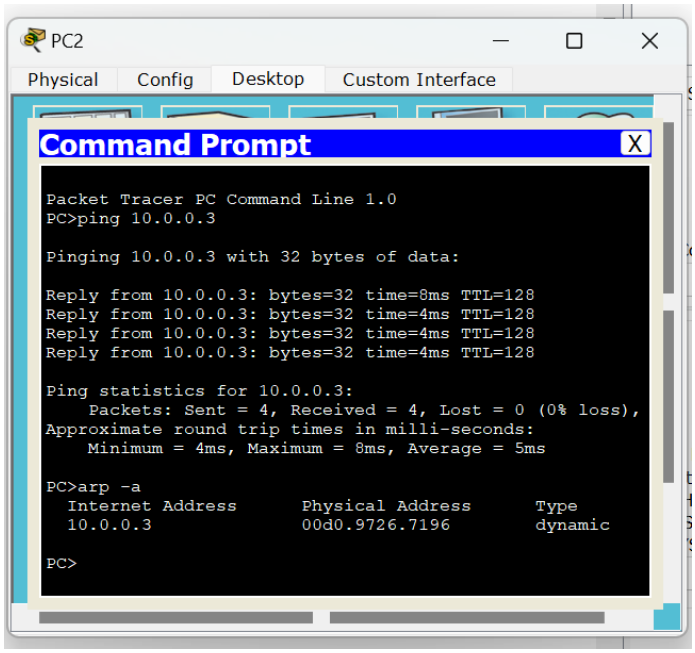

File Edit Options View Tools Extensions Help
Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
ARP Table for PC1
ARP Table for PC2
ARP Table for PC3
IP Address
Hardware Addres
Interface
10.0.0.10 0006.2A0E.706E FastEthernet0
Simulation Panel
Event List
Vis.
Time(sec)
Last De
At Dev
Type
Info
2.853 -- Switc... DTP
2.854 Switch0 PC0 DTP
2.854 -- Switc... DTP
2.855 Switch0 PC3 DTP
2.856 -- Switc... DTP
Reset Simulation
Constant Delay
Captured to: 2.856 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:08:02.475
Power Cycle Devices
PLAY CONTROLS: Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Connections
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Statu
Sourc
Destinatic
Type
Colo
Time(s
Period
Num
Edit
Delete
Successful PC3 Server0 IC... 0.000 N 0 (ed... (delete)

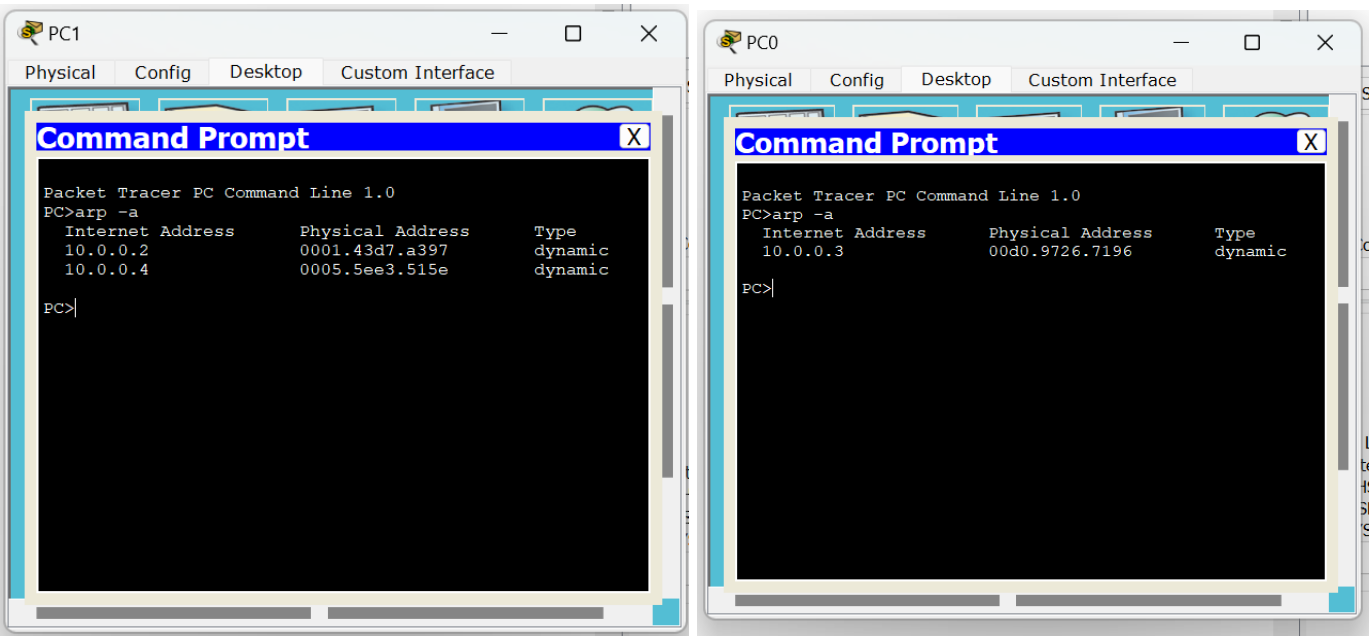

File Edit Options View Tools Extensions Help
Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
IP Address
Interface
ARP Table for PC1
ARP Table for PC3
10.0.0.10 0006.2A0E.706E FastEthernet0
ARP Table for PC2
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type Info
3.877 Switch0 PC2 STP
3.877 Switch0 PC1 STP
3.877 Switch0 PC0 STP
3.877 Switch0 PC3 STP
3.877 Switch0 Serve... STP
Reset Simulation
Constant Delay
Captured to: 3.877 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:08:03.496
Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List
Simulation
Connections
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Statu Sourc Destinatic Type Colo Time(s Period Num Edit Delete
Successful PC3 Server0 IC... 0.000 N 0 (ed... (delete)


Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
ARP Table for PC1
ARP Table for PC3
10.0.0.10 0006.2A0E.706E FastEthernet0
ARP Table for PC2
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type Info
6.900 PC3 Switc... ICMP
6.901 Switch0 Serve... ICMP
6.902 Server0 Switc... ICMP
6.903 Switch0 PC3 ICMP
7.886 -- Switc... STP
Reset Simulation
Constant Delay
Capturing...
PC3
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>arp -a
No ARP Entries Found
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=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 4ms, Average = 4ms
PC>
Time: 00:08:06.522
Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Connections
Scenario 0
New
Delete


Switch0
Physical
Config
CLI
IOS Command Line Interface
Switch(config-if)#exit
Switch(config)#exit
Switch#sho
%SYS-5-CONFIG_I: Configured from console by console
% Incomplete command.
Switch#show mac addr-table
^
% Invalid input detected at '^' marker.
Switch#show mac address-table
Mac Address Table
Vlan Mac Address Type Ports
1 0001.43d7.a397 DYNAMIC Fa1/1
1 0005.5ee3.515e DYNAMIC Fa3/1
1 0006.2a0e.706e DYNAMIC Fa6/1
1 0060.3e21.aa45 DYNAMIC Fa0/1
1 00d0.9726.7196 DYNAMIC Fa2/1
Switch#
Copy
Paste


PC3
Physical
Config
Desktop
Custom Interface
Command Prompt
PC>arp -a
No ARP Entries Found
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=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Reply from 10.0.0.10: bytes=32 time=4ms TTL=128
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 4ms, Average = 4ms
PC>arp -a
Internet Address Physical Address Type
10.0.0.10 0006.2a0e.706e dynamic
PC>


PC2
Physical
Config
Desktop
Custom Interface
Command Prompt
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=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
PC>arp -a
Internet Address Physical Address Type
10.0.0.3 00d0.9726.7196 dynamic
PC>

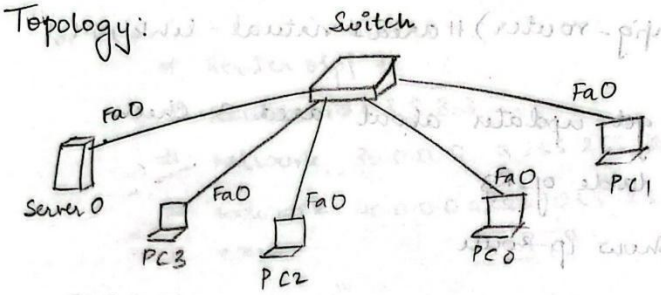


Observation:

EXPERIMENT - 8

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

Topology:



Procedure:

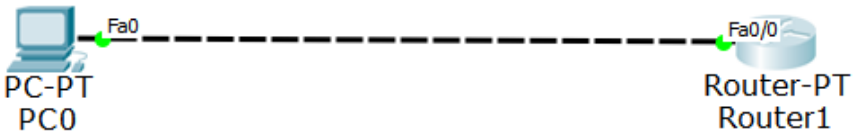
1. Create a topology of 4 PC's & a server
2. IP address assigned to all.
3. Connect them via switch
4. Use inspect tool on PC to see the ARP table
5. Command in CLI for the same is arp -a
6. Initially, the ARP table is empty.
7. In CLI of switch - show mac address table
   to see how the switch learns from transactions and builds the address table
8. Use capture button in simulation panel to go step-by-step so that changes in ARP can be clearly noted
9. Observe switch as well as nodes, update the ARP table as & when a new communication starts

Experiment 10:

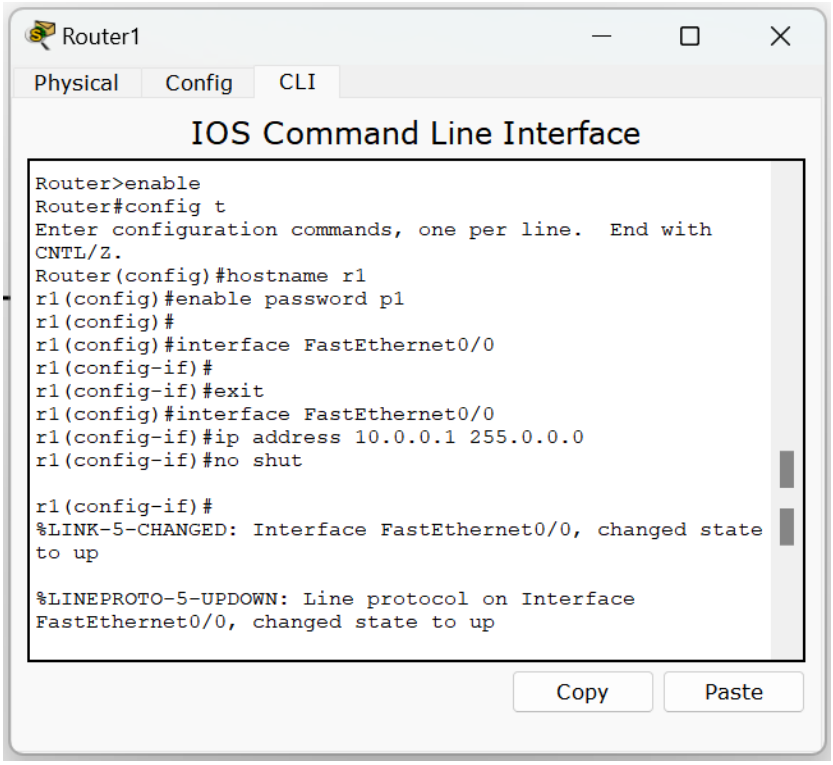
To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.

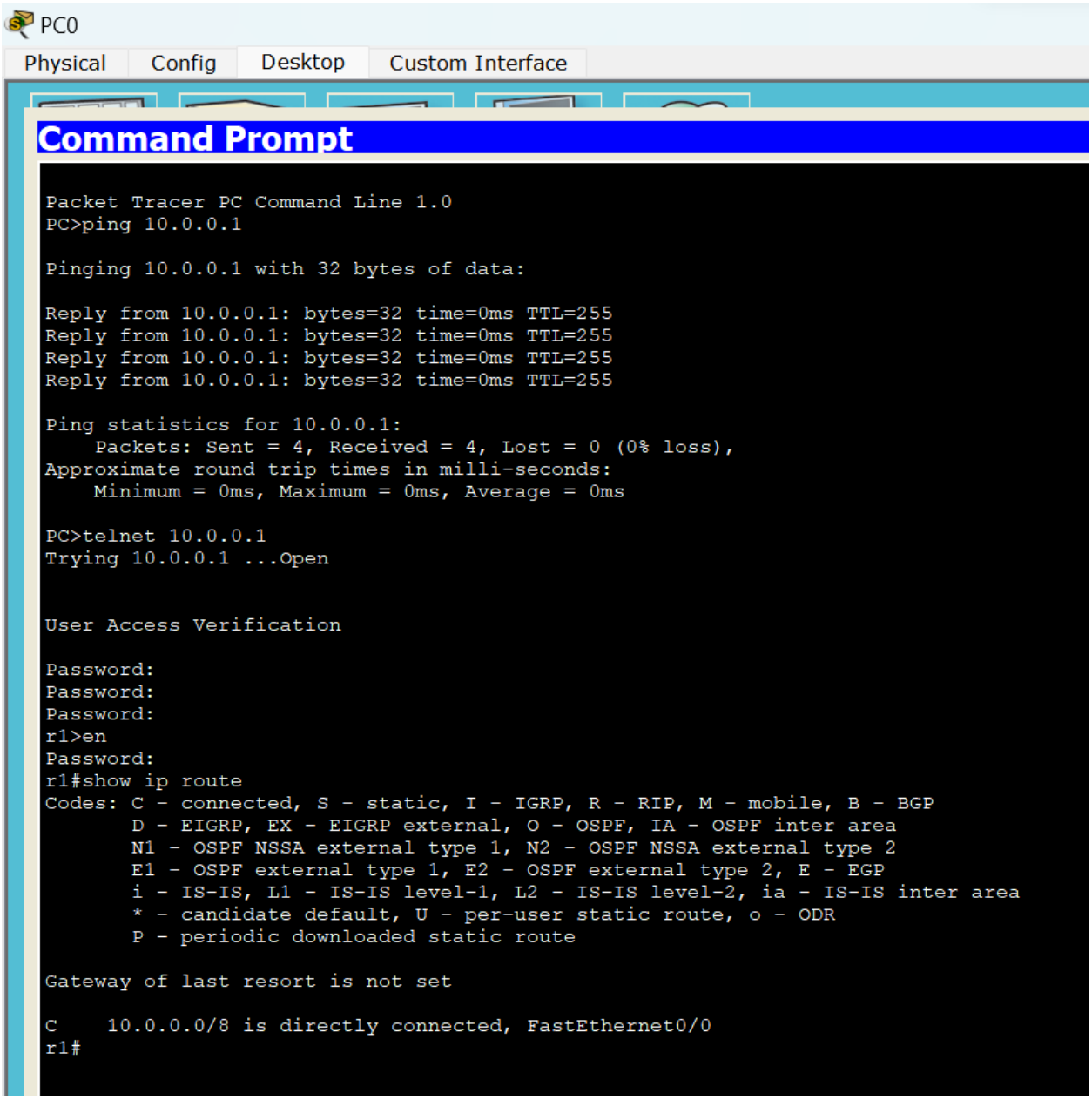
Aim: To understand the operation of TELNET by accessing the router in server room from a PC in the IT office.

Topology:



Output:



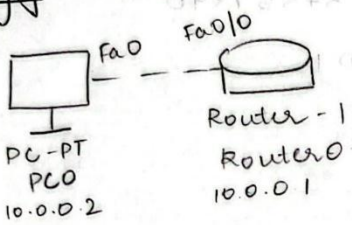

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open
User Access Verification
Password:
Password:
Password:
r1>en
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#

Observation:

EXPERIMENT - 12:

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

Topology:



Procedure:

1) Create the topology as shown above
   wire used - copper cross over

2) Configure the PC
   IP address = 10.0.0.2
   gateway = 10.0.0.1

3) In Router0 CLI

```
Router >en
Router #config t
Router(config)# hostname r1
r1(config)# enable secret p1
r1(config)# int fa0/0
r1(config-if)# ip address 10.0.0.1 255.0.0.0
r1(config-if)# no shut
r1(config-if)# line vty 0 5  -- for 6 users
r1(config-line)# login
r1(config-line)# password p0
r1(config-line)# exit
r1(config)# exit.
```

Ping Output & Command prompt of PC0

We can successfully ping 10.0.0.1 from PC0
In PC0 command prompt

```
PC> telnet 10.0.0.1
Trying 10.0.0.1 ... open
User Access Verification
Password : p0
R1> en
Password : p1
R1# show ip route
```

Observation:

→ The admin in PC is able to run commands as run in router CCI & see the result from the PC

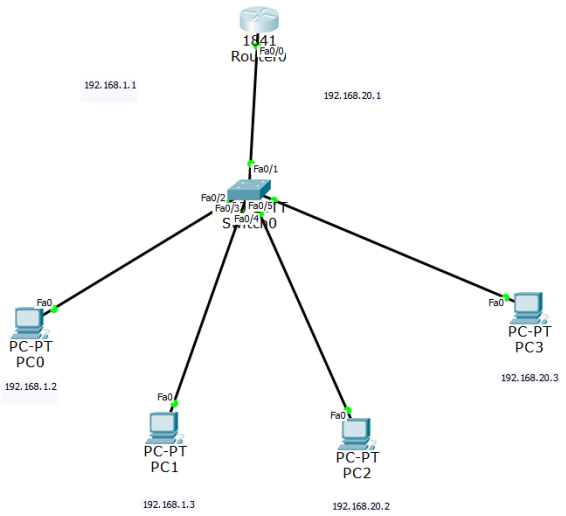
→ Thus with help of TELNET, we access the router in server from a PC

Experiment 11:

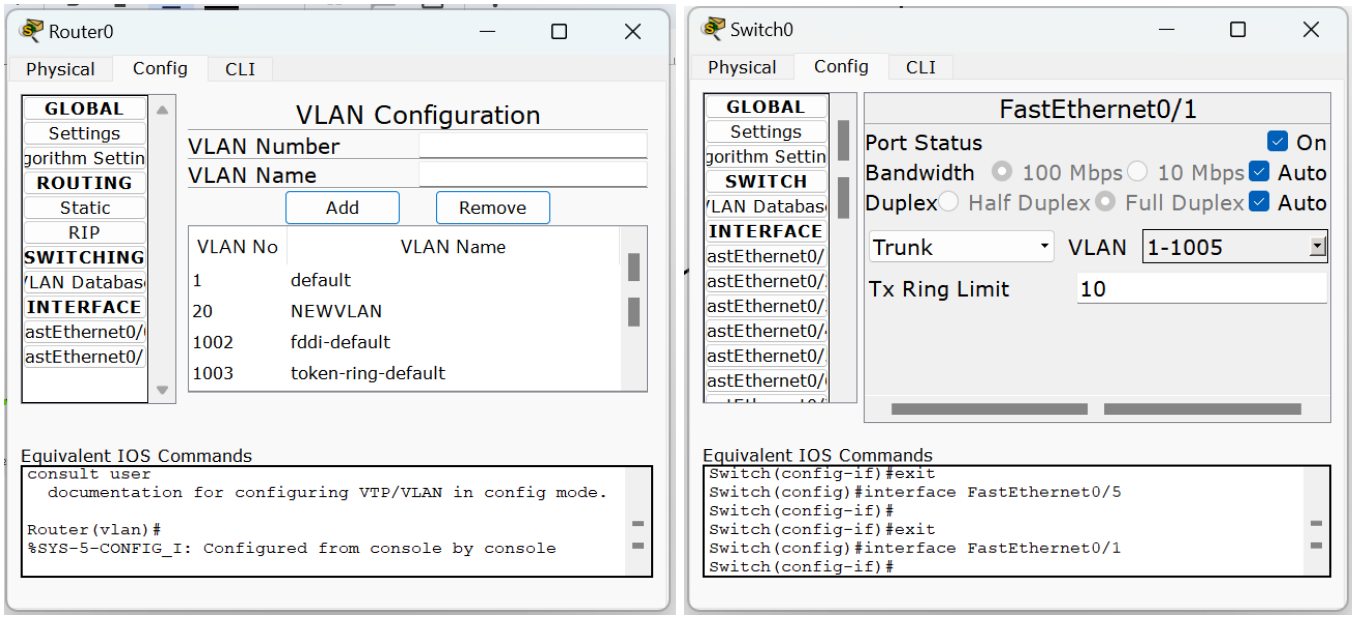
To construct a VLAN and make the PC's communicate among a VLAN

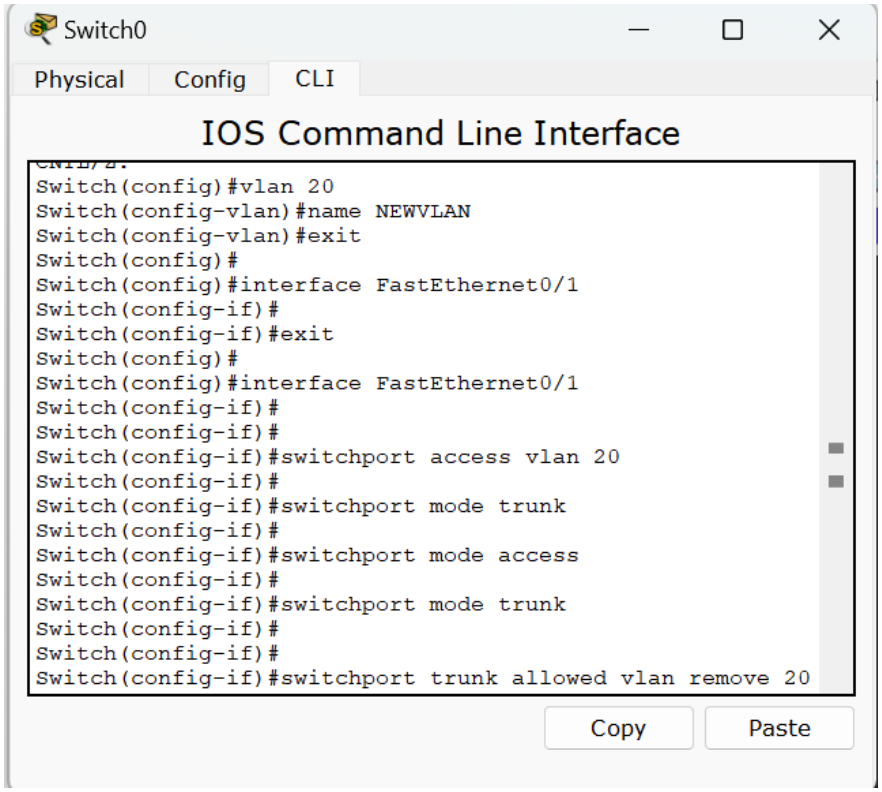
Aim: To construct a VLAN and make the PC's communicate among a VLAN

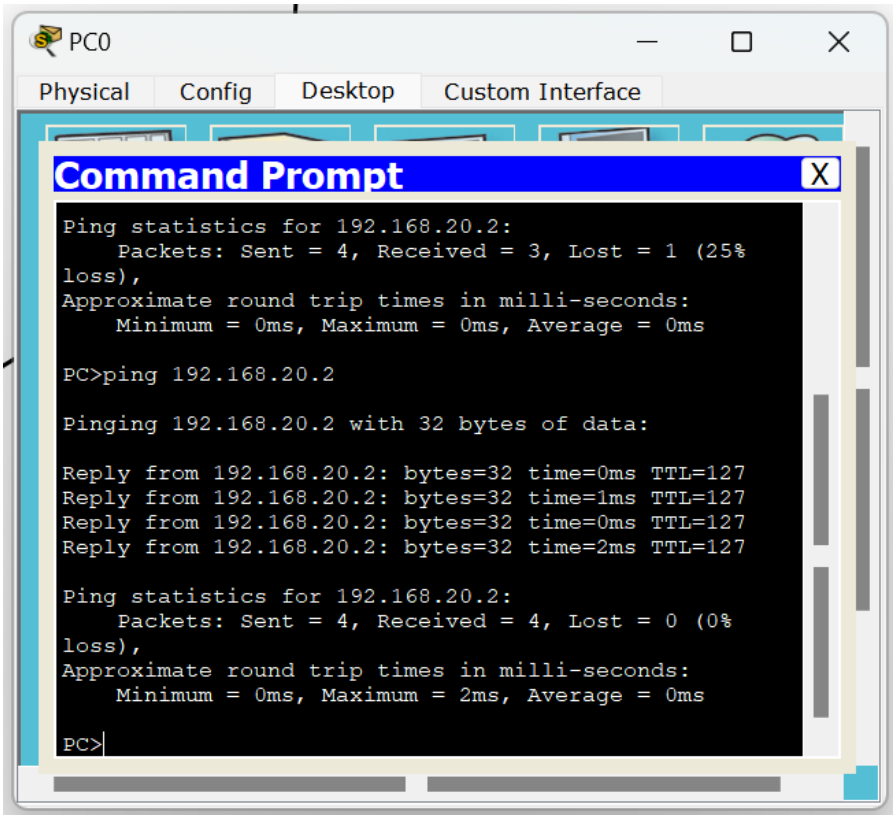
Topology:



Output:




Switch0
Physical
Config
CLI
IOS Command Line Interface
Switch(config)#vlan 20
Switch(config-vlan)#name NEWVLAN
Switch(config-vlan)#exit
Switch(config)#
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport access vlan 20
Switch(config-if)#
Switch(config-if)#switchport mode trunk
Switch(config-if)#
Switch(config-if)#switchport mode access
Switch(config-if)#
Switch(config-if)#switchport mode trunk
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport trunk allowed vlan remove 20
Copy
Paste

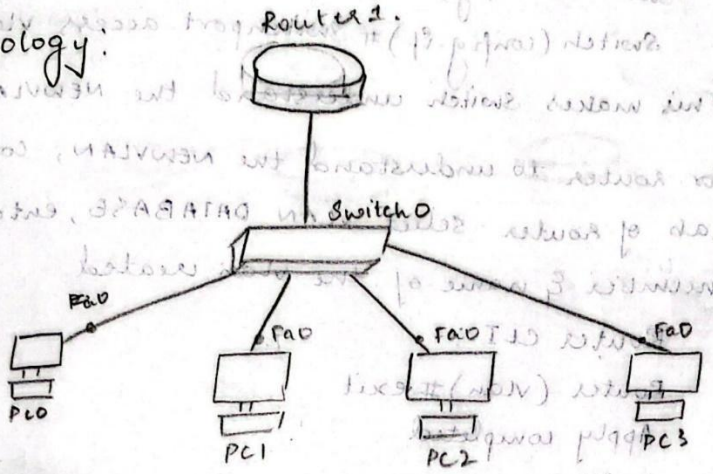

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Ping statistics for 192.168.20.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25%
loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>ping 192.168.20.2
Pinging 192.168.20.2 with 32 bytes of data:
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=1ms TTL=127
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=2ms TTL=127
Ping statistics for 192.168.20.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>

Observation:

EXPERIMENT - 9

Aim: To create Virtual LAN (VLAN)

Topology:



Procedure:

1). Create topology using 4 PC's, a switch & choose the 1841 router.

2) In switch, go to config tab and select VLAN database

3) Give any VLAN number, say 2. Include any name, say NEW VLAN.

```
Switch (config) # vlan 2
                # name NEWVLAN
                # exit
                # interface Fa4/1
```

4) Select the interface fast ethernet 4/1 and make it the trunk

VLAN trunking allows switches to foward frame from different VLANs over a single link called trunk. This is done by adding an additional header information called tag to the Ethernets frame. The process of adding this small header is called VLAN tagging.

5) Look into the interfaces of the switches with the 2 NEWVLAN systems.

```
Switch (config) # interface fa2/1
Switch (config-if) # switchport access vlan2
```

This makes switch understand the NEWVLANS

6) For router to understand the NEWVLAN, config tab of router select VLAN DATABASE, enter the number & name of the vlan created

Router CLI:

```
Router (vlan) # exit
Apply completed
Exiting...
Router # config t
Router (config) # interface fa 0/0.1000
Router (config-subif) # encapsulation dot1q 2
                      # ip address 192.168.2.1
                        255.255.255.0
                      # no shut
                      # exit
```

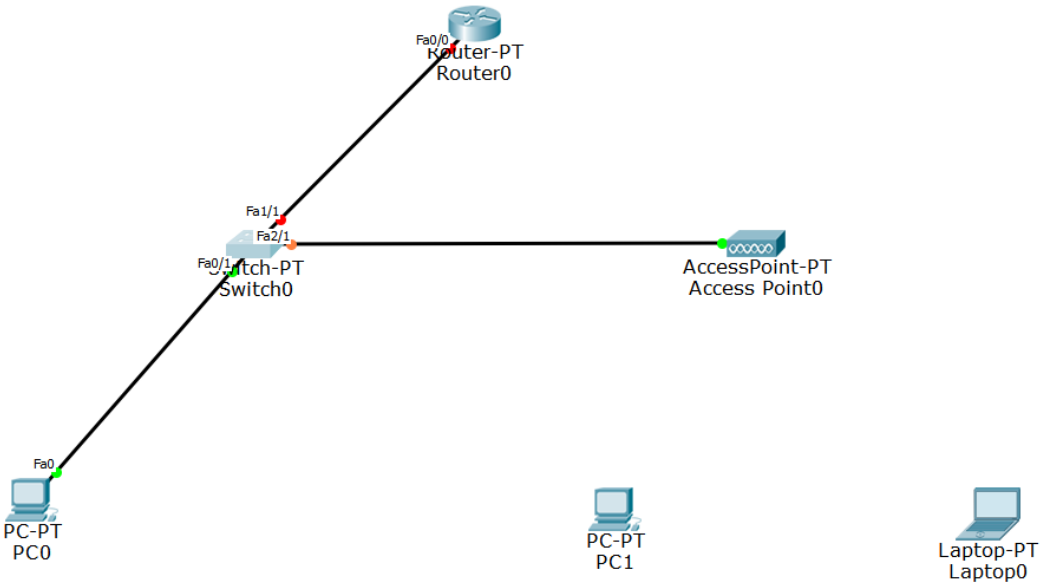
Experiment 12:

To construct a WLAN and make the nodes communicate wirelessly

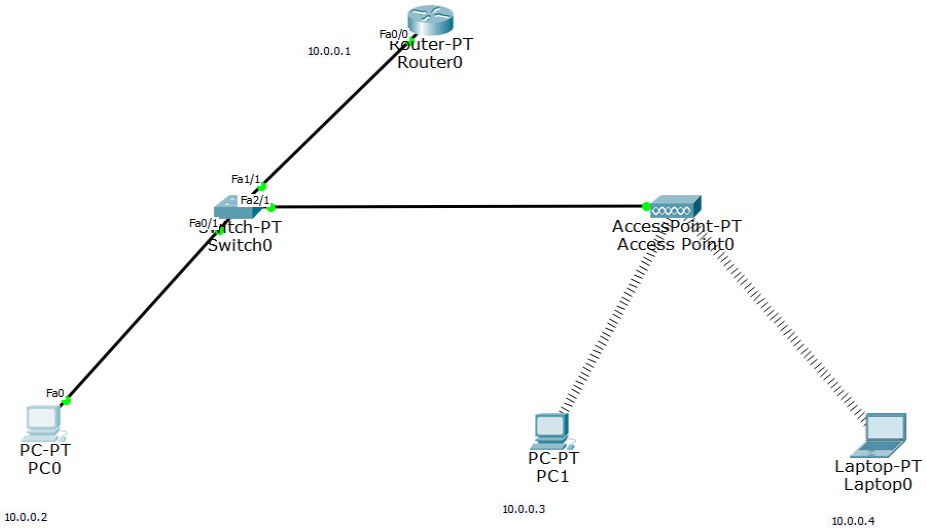
Aim: To construct a WLAN and make the nodes communicate wirelessly.

Topology:

(Initial)

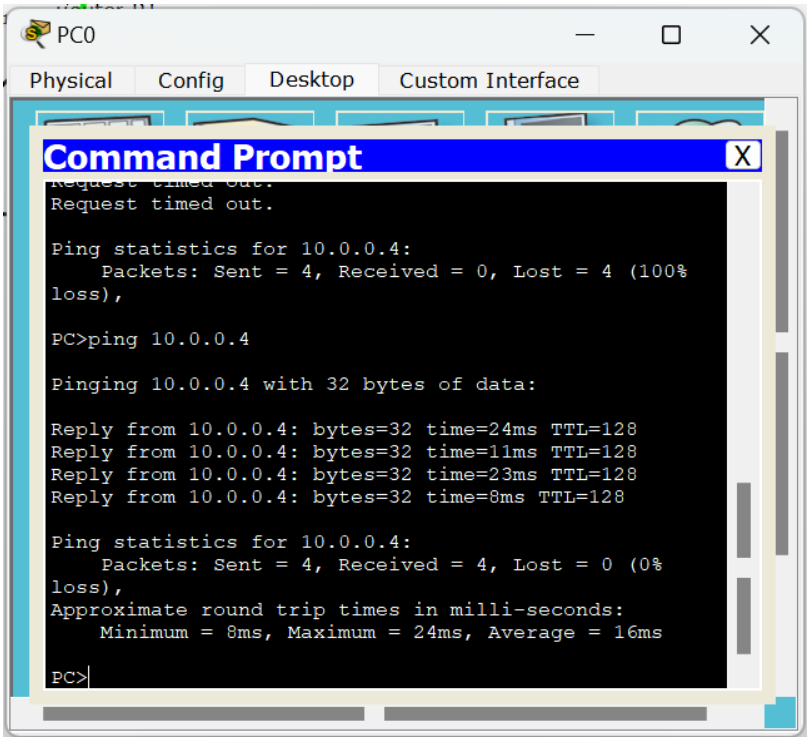


(Final)

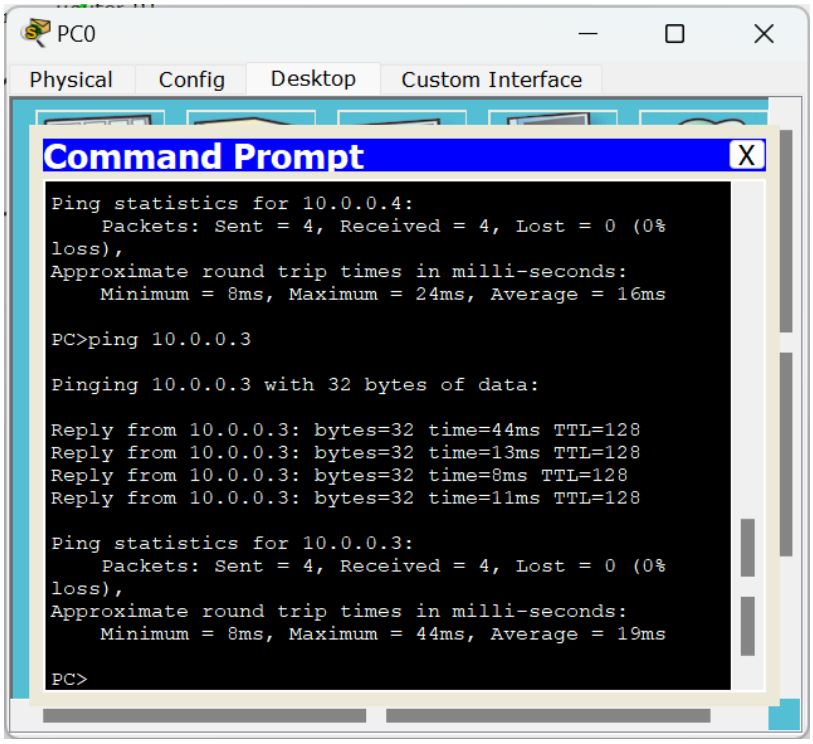


Output:

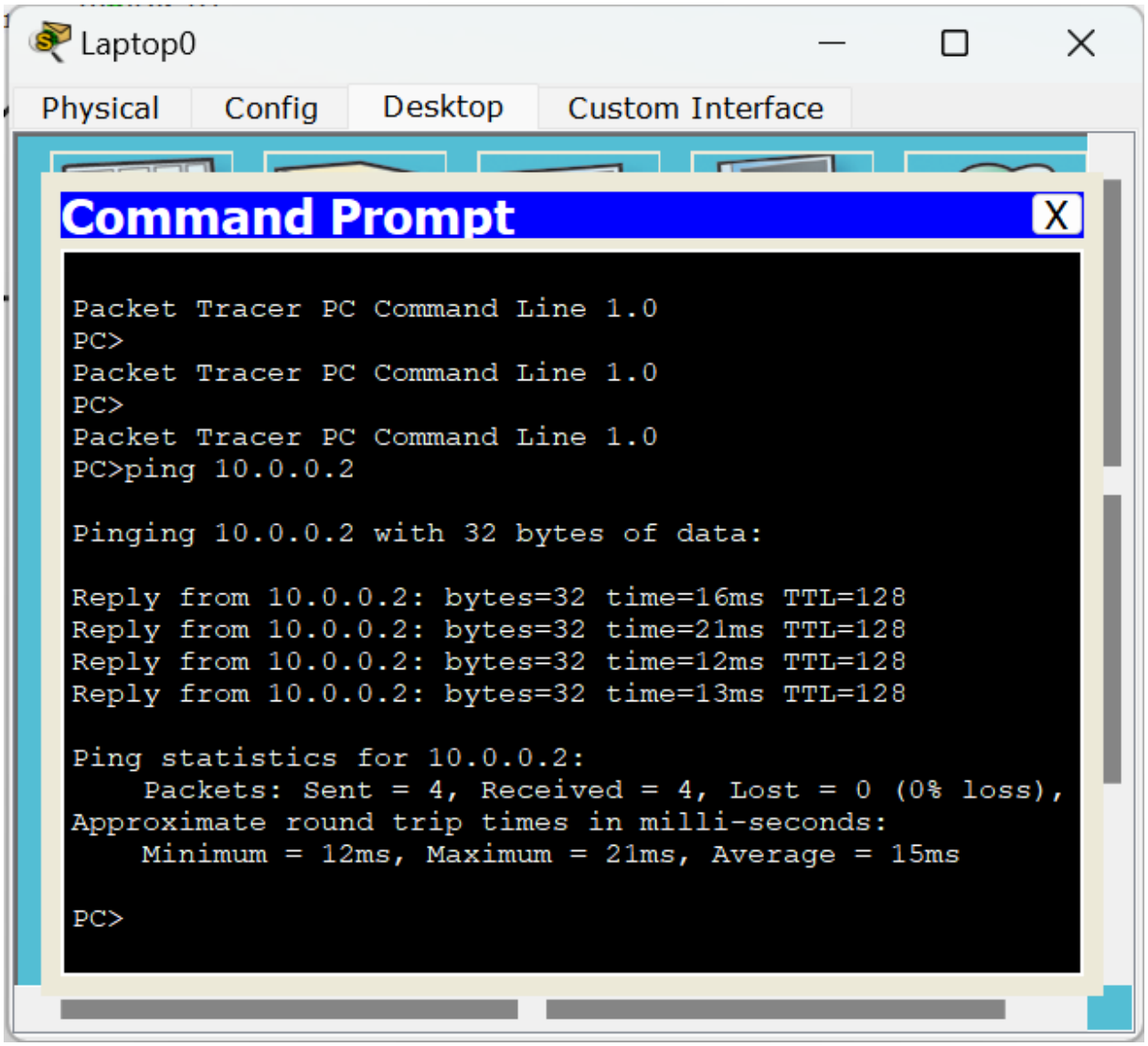
PC0 TO LAPTOP



PC0 TO PC1



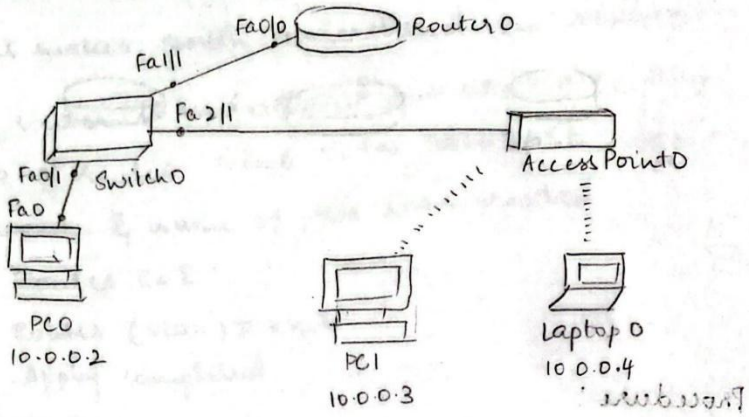
LAPTOP TO PC0



Observation:

EXPERIMENT - 11

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



Procedure:

1) Create the topology as seen above
2) Configure PC3
   IP address: 10.0.0.2
3) Configure Router 0. Set fastethernet 0/0 as 10.0.0.1 in usual method
4) Configure the Access Point → Port1 → Config
   → SSID = WLAN
   Select WEP and give any 10 digit hex
   = 1234567890
5) To configure PC4 & laptop with wireless standards.
   - Switch off the device. Drag the existing PT-HOST-NM-1AM to the LHS
   - Drag WMP300N wireless interface to the empty port.

- Switch on the device

6) In the config tab, a new wireless Interface would have been added

7) Configure SSID, WEP, WEP key, IP address. and Gateway to both PC & and Laptop.

SSID = WLAN

WEP key = 1234567890

Gateway - 10.0.0.1

Ping output : (In PC0)

PC> ping 10.0.0.4

Pinging 10.0.04 with 32 bytes of data.

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

"

"

"

Observations :

- We can ping from every service to every other service successfully.
- Thus the wireless service is successful.
- In the final topology, we observe stripped lines from access point to PC4 & laptop indicating the establishment of wireless connection.

CYCLE 2:

Experiment 13:

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

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

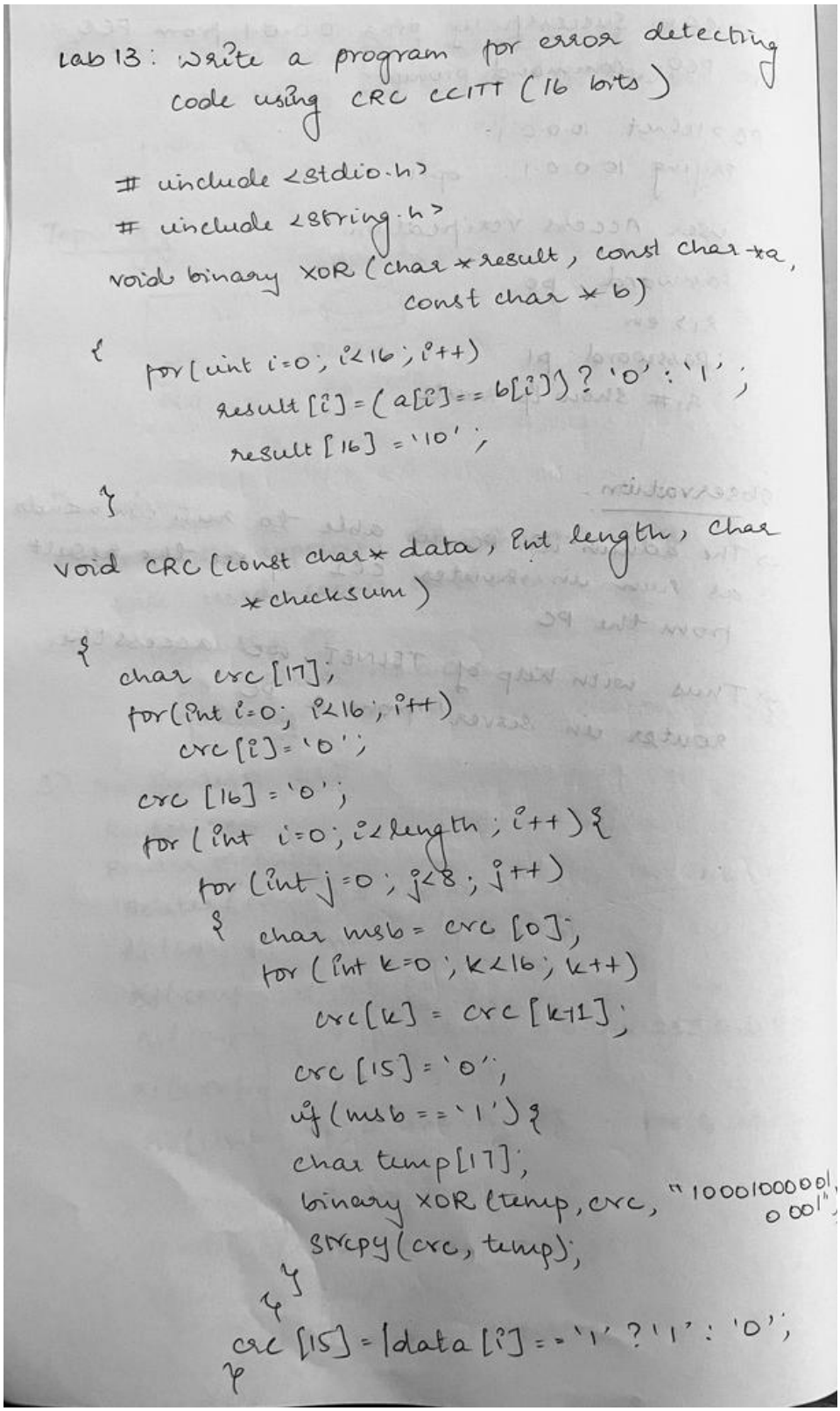
Code:

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

// CRC-CCITT polynomial: x^16 + x^12 + x^5 + 1 (0x1021)
//#define CRC_POLY 0x1021

// Function to perform bitwise XOR on binary strings
void binaryXOR(char *result, const char *a, const char *b) {
    for (int i = 0; i < 16; i++) {
        result[i] = (a[i] == b[i]) ? '0' : '1';
    }
    result[16] = '\0';
}

// Function to calculate CRC-CCITT checksum
void calculateCRC(const char *data, int length, char *checksum) {
    char crc[17];
    for (int i = 0; i < 16; i++) {
        crc[i] = '0';
```

```
    }
    crc[16] = '\0';

    for (int i = 0; i < length; i++) {
        for (int j = 0; j < 8; j++) {
            char msb = crc[0];
            for (int k = 0; k < 16; k++) {
                crc[k] = crc[k + 1];
            }
            crc[15] = '0';

            if (msb == '1') {
                char temp[17];
                binaryXOR(temp, crc, "1000100000100001"); // CRC_POLY in
binary
                strcpy(crc, temp);
            }
        }
        crc[15] = (data[i] == '1') ? '1' : '0';
    }

    strcpy(checksum, crc);
}

int main() {
```

```
char data[100]; // Replace with your actual data
printf("Enter data in binary: ");
scanf("%s", data);
int dataLength = strlen(data);
char checksum[17];
calculateCRC(data, dataLength, checksum);

printf("Calculated CRC: %s\n", checksum);
// Simulating error by changing a bit
// data[2] ^= 0x01; // Uncomment this line to introduce an error

// Verify the received data
char receivedChecksum[17];
printf("Enter received CRC: ");
scanf("%s", receivedChecksum);

if (strcmp(receivedChecksum, checksum) == 0) {
    printf("Data is error-free.\n");
} else {
    printf("Data contains errors.\n");
}

return 0;
```

Observation:



Lab 13: Write a program for error detecting code using CRC CCITT (16 bits)

```
#include<stdio.h>
#include<string.h>
void binaryXOR(char *result, const char *a,
               const char *b)
{
    for(int i=0; i<16; i++)
        result[i] = (a[i]==b[i])?'0':'1';
    result[16] = '\0';
}
void CRC(const char* data, int length, char
         *checksum)
{
    char crc[17];
    for(int i=0; i<16; i++)
        crc[i]='0';
    crc[16]='\0';
    for(int i=0; i<length; i++){
        for(int j=0; j<8; j++)
        { char msb = crc[0];
          for(int k=0; k<16; k++)
              crc[k] = crc[k+1];
          crc[15]='0';
          if(msb=='1'){
          char temp[17];
          binaryXOR(temp,crc,"1000100000100001");
          strcpy(crc,temp);
        }
    }
    crc[15] = (data[i]=='1'?'1':'0');
}
```

```
    strcpy(checksum, crc);
}
void main()
{
    char data[100];
    printf("Enter data in binary");
    scanf("%s", data);
    int datalength = strlen(data);
    char checksum[17];
    calculateCRC(data, datalength, checksum);
    char recchecksum[17];
    printf("Enter received CRC:");
    scanf("%s", recchecksum);

if(strcmp(recchecksum, checksum) == 0)
        printf("Data is error-free\n");

else
    printf("Data contains errors\n");

return 0;
}
```

Output:

Enter data in binary: 1100101011100100
Calculated CRC : 1110100101111000
Entered received CRC: 1110100101111000
Data is error free.

Output:

```
Enter data in binary: 10001
Calculated CRC: 0111001001000001
Enter received CRC: 0111001001000001
Data is error-free.
```

```
Enter data in binary: 10011
Calculated CRC: 0111001101000001
Enter received CRC: 1011010101010101
Data contains errors.
```

Experiment 14:

Write a program for congestion control using Leaky bucket algorithm

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

Code:

```
#include<stdio.h>

int main(){
    int incoming, outgoing, buck_size, n, store = 0;
    printf("Enter bucket size:");
    scanf("%d", &buck_size);
    printf("Enter outgoing rate:");
    scanf("%d", &outgoing);
    printf("Enter number of inputs:");
    scanf("%d", &n);

    while (n != 0) {
        printf("Enter the incoming packet size: ");
        scanf("%d", &incoming);
        if (incoming <= (buck_size - store)){
            store += incoming;
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
        } else {
            printf("Dropped %d no of packets\n", incoming - (buck_size - store));
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
```

```
            store = buck_size;
        }
        store = store - outgoing;
        printf("After outgoing %d packets left out of %d in buffer\n", store,
buck_size);
        n--;
    }
}
```

Observation:

Lab 14: Write a program for congestion control using leaky bucket algorithm.

```
#include <stdio.h>
void main()
{
  int psize, bsize, outgoing, empty, choice;
  printf("Enter bucket size ");
  scanf("%d", &bsize);
  empty = bsize;
  printf("Enter the outgoing rate=");
  scanf("%d", &outgoing);

  while(1)
  {
  printf("\n Enter the packet size");
  scanf("%d", &psize);
  if(psize < bsize && psize <= emptyspace)
    {
      emptyspace = emptyspace - psize;
      print("The packet of size %d is
            added & in the bucket!\n",
            psize);
      empty += outgoing;
    }

else
    printf("The packet of size %d is dropped
          due to lack of space in the bucket!\n"

    printf("\n Enter 1 to continue or 0 to
          stop:");
```

```
scanf ("%d", &choice);
if (choice == 0)
    break;
}
}
```

Output:

Enter bucket size : 5000
" outgoing rate : 200

Enter packet size : 300
The packet size 3000 is added and in the bucket
Enter 1 to continue or 0 to stop : 1

Enter packet size : 2000
Added to bucket.

Enter packet size : 1500.
Packet is dropped due to lack of space.

Enter packet : 100
Packet of size 100 is added & in the bucket.

Output:

```
Enter bucket size:1000
Enter outgoing rate:100
Enter number of inputs:3
Enter the incoming packet size: 300
Bucket buffer size 300 out of 1000
After outgoing 200 packets left out of 1000 in buffer
Enter the incoming packet size: 400
Bucket buffer size 600 out of 1000
After outgoing 500 packets left out of 1000 in buffer
Enter the incoming packet size: 1100
Dropped 600 no of packets
Bucket buffer size 500 out of 1000
After outgoing 900 packets left out of 1000 in buffer
```

Experiment 15:

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

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

Code:

ClientTCP.py

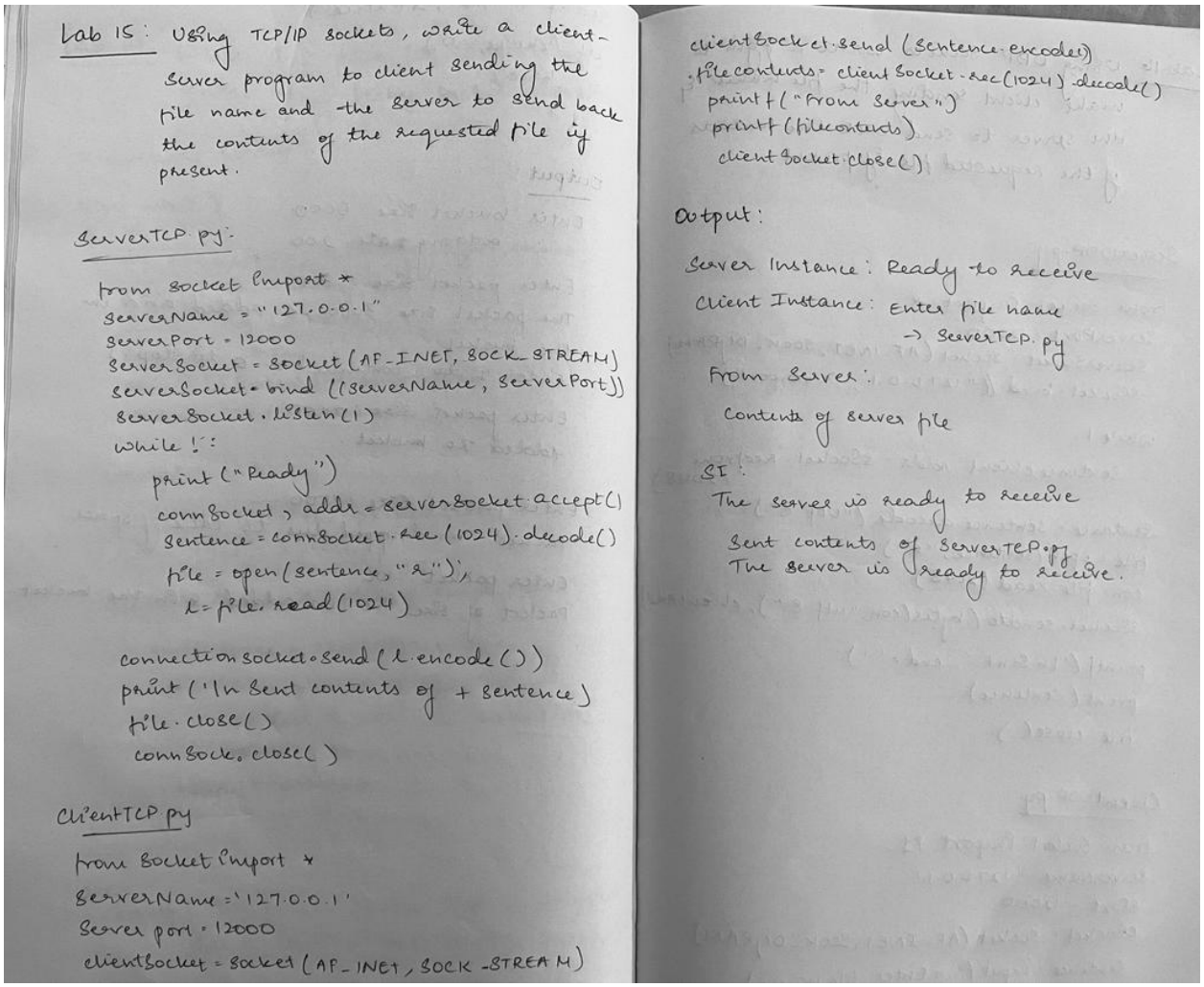
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name:")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ("\nFrom Server:\n")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

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 ("\nSent contents of "+ sentence)
file.close()
connectionSocket.close()
```

Observation:



Lab 15: Using TCP/IP sockets, write a client-server program to client sending the file name and the server to send back the contents of the requested file if present.

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("Ready")
    connSocket, addr = serverSocket.accept()
    sentence = connSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionsocket.send(l.encode())
    print('\n Sent contents of ' + sentence)
    file.close()
    connSock.close()
```

ClientTCP.py

```
from Socket import *
serverName = '127.0.0.1'
serverport = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print("From Server:")
print(fileContents)
clientSocket.close()
```

Output:

Server Instance: Ready to receive

Client Instance: Enter file name
→ ServerTCP.py

From Server:

Contents of server file

SI:

The server is ready to receive

Sent contents of ServerTCP.py
The server is ready to receive.

Output:

Server.tcp.py - C:/Users/dhiks/Desktop/TFCS Notes/Server.tcp.py (3.11.0)

File Edit Format Run Options Window Help

```
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()
```

Client.tcp.py - C:/Users/dhiks/Desktop/TFCS Notes/Client.tcp.py (3.11.0)

File Edit Format Run Options Window Help

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

```
Enter file name:Server.tcp.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()
```

```
________
The server is ready to receive

Sent contents of Server.tcp.py
The server is ready to receive
```

Experiment 16:

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

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

Code:

ClientUDP.py

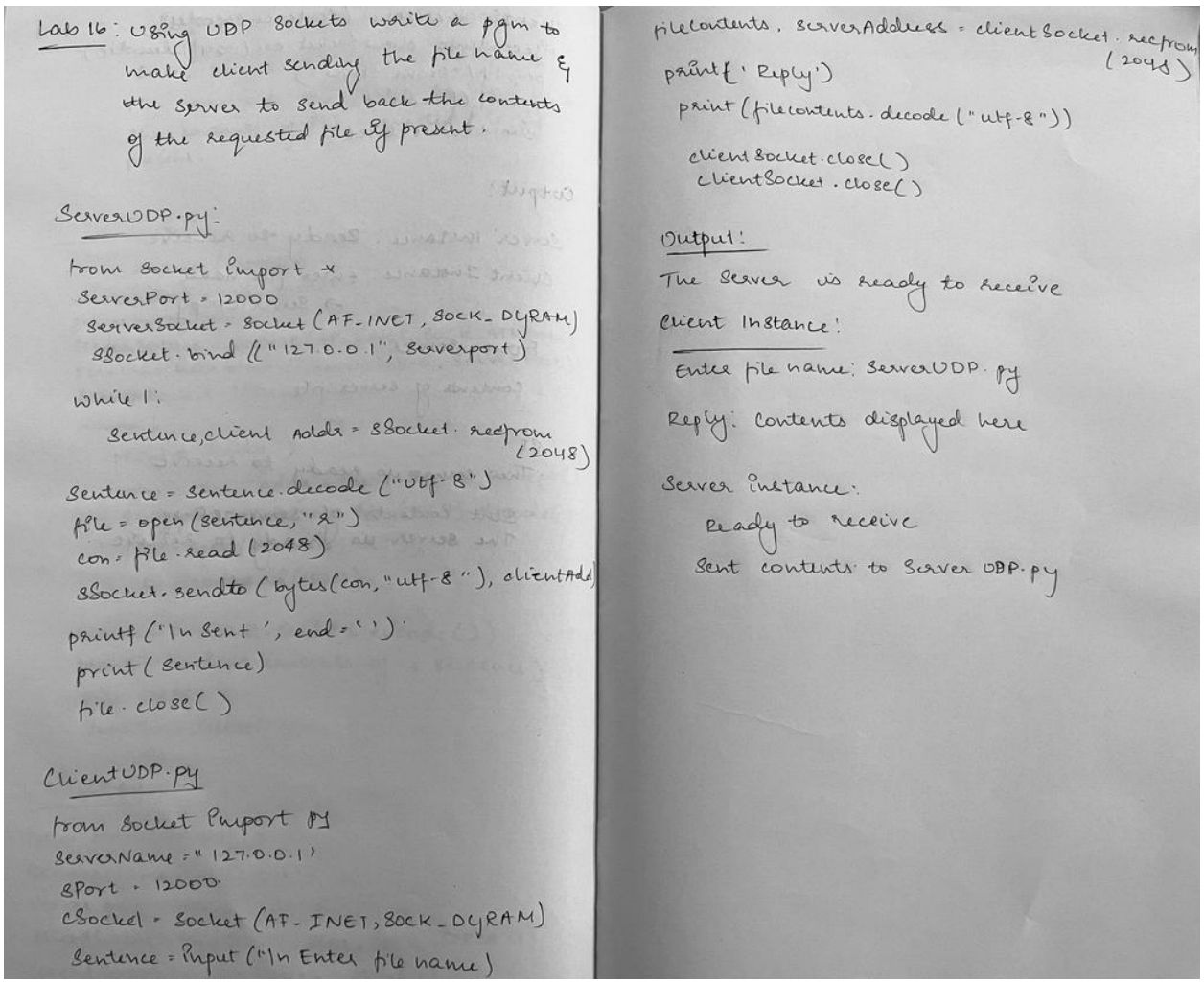
```
from socket import *

serverName = '127.0.0.1'

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(&amp;quot;utf-8&amp;quot;))
clientSocket.close()
clientSocket.close()
```

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)
        file.close()
```

Observation:



Lab 16: Using UDP Sockets write a pgm to make client sending the file name & the server to send back the contents of the requested file if present.

ServerUDP.py:

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

while 1:
    sentence, clientAddr = sSocket.recvfrom
                                (2048)
sentence = sentence.decode("utf-8")
file = open(sentence, "r")
con = file.read(2048)
sSocket.sendto(bytes(con, "utf-8"), clientAdd
printf("\n Sent ", end=' ')
print(sentence)
file.close()
```

ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
sPort = 12000
cSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name")
fileContents, serverAddress = clientSocket.recvfrom
                                (2048)
print(' Reply')
print(fileContents.decode("utf-8"))

clientSocket.close()
clientSocket.close()
```

Output:

The Server is ready to receive

Client Instance:

Enter file name: ServerUDP.py

Reply: Contents displayed here

Server instance:

Ready to receive

Sent contents to ServerUDP.py

Output:

Client.udp.py - C:/Users/dhiks/Desktop/TFCS Notes/Client.udp.py (3.11.0)

File Edit Format Run Options Window Help

```
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'))
clientSocket.close()
clientSocket.close()
```

Server.udp.py - C:/Users/dhiks/Desktop/TFCS Notes/Server.udp.py (3.11.0)

File Edit Format Run Options Window Help

```
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)
        file.close()
```

```
Enter file nameServer.udp.py

Reply from Server:

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)
        file.close()
```

```
The server is ready to receive

Sent contents of Server.udp.py
```