VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

DHIKSHA RATHIS (1BM21CS055)

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 **DHIKSHA RATHIS (1BM21CS055),** 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

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
17.	15/6/2023	Tool Exploration -Wireshark	76-77

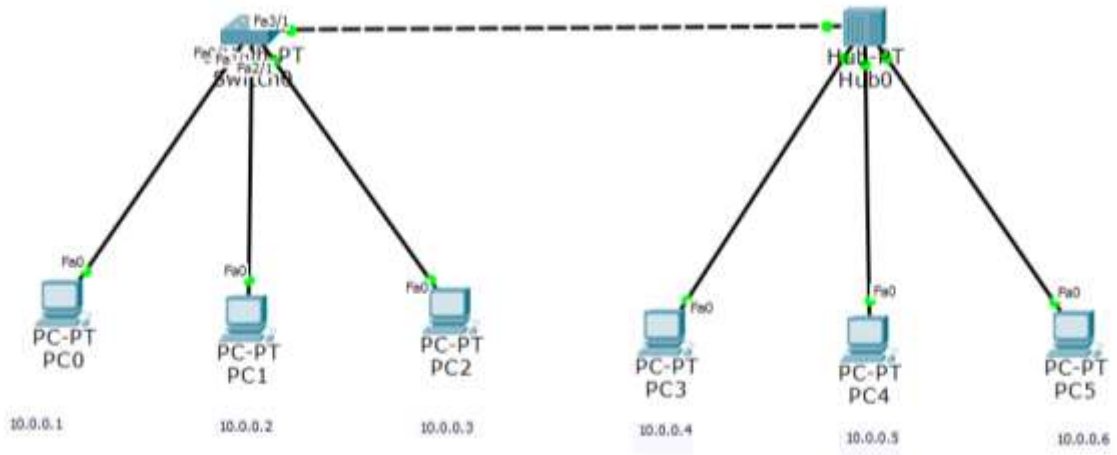
CYCLE 1:

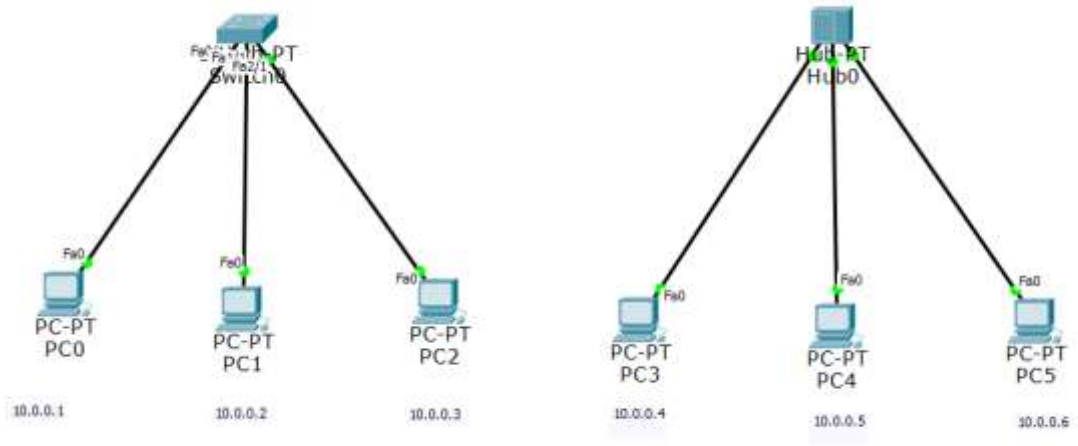
Experiment 1:

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

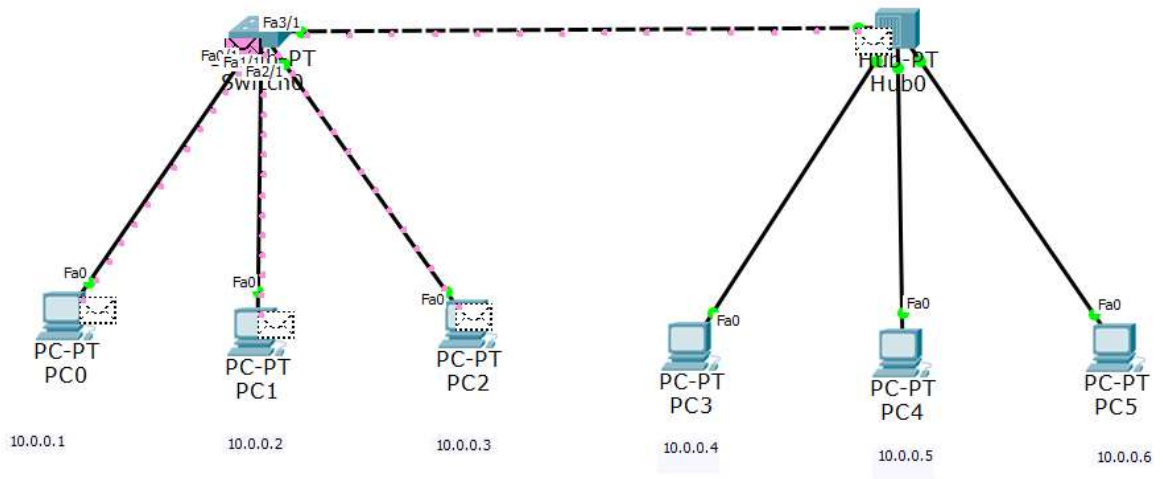
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message

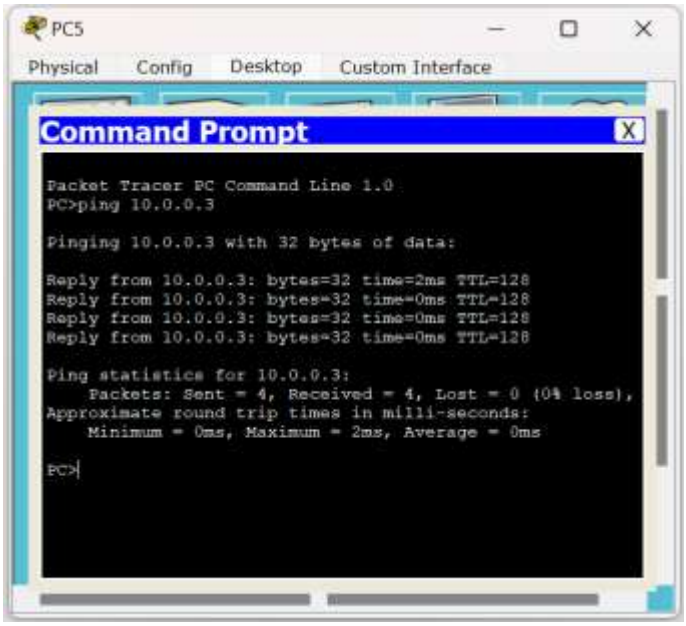
Topology:

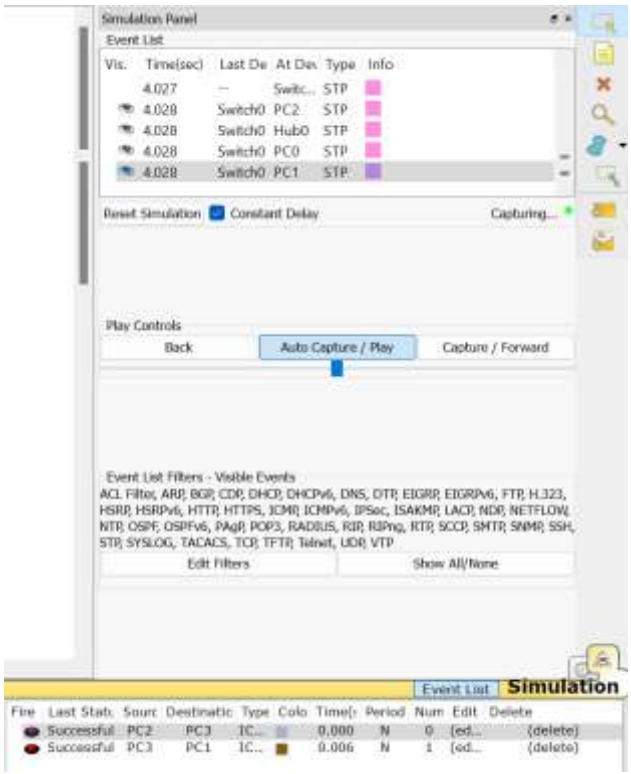




Output:


Fa3/1
Fa2/1
Switch0
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


PC5
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=2ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>


Simulation Panel
Event List
Vis. Time(sec) Last De At Dev Type Info
4.027 -- Switc... STP
4.028 Switch0 PC2 STP
4.028 Switch0 Hub0 STP
4.028 Switch0 PC0 STP
4.028 Switch0 PC1 STP
Reset Simulation
Constant Delay
Capturing...
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
Edit Filters
Show All/None
Event List
Simulation
Fire Last Stat. Sourc Destinatic Type Colo Time(- Period Num Edit Delete
Successful PC2 PC3 IC... 0.000 N 0 (ed... (delete)
Successful PC3 PC1 IC... 0.006 N 1 (ed... (delete)

Observation:

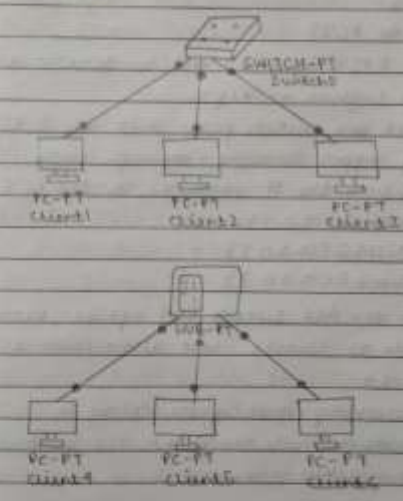
Date 15/6/23
Page 1

Experiment 1

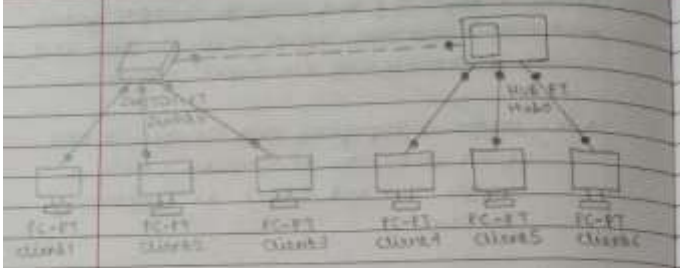
Create a topology and simulate a PDU from source to destination using a single hub and switch as connecting domains

Aim: create a topogoly and simulate a PDU from source to destination using a simple hub and switch as connecting domains

Topology



Date 15/6/23
Page 2



Procedure:

(Switch to PC's)

1) Place 3 PC's (Generic) and to generic switch in the Logical workspace
2) Connect the switch ports with all 3 PC's using copper straight-through wire
3) Configure the IP-addresses of the PC's
   PC 5 (10.0.0.1)
   Client2 (10.0.0.2)
   Client 3 (10.0.0.3)
4) Click the Add simple PDU option, select the source as Client1 and destination as Client 4
5) On clicking Autocapture / play in simulation mode, the packet can be traced in simulation mode

(Hub to PC's)

1) Follow the same procedure as done for switch but replace switch with hub
   Client 4 (10.0.0.4)
   Client 5 (10.0.0.5)
   Client 6 (10.0.0.6)

Now connect the switch and hub and simulate the packet transfer from source to destination

Output:

PC > ping 10.0.0.3

pinging 10.0.0.3 with 32 bytes of data :

Reply from 10.0.0.3 : byte = 32 time = 0 ms TTL = 128
Reply from 10.0.03 : byte = 32 time = 2 ms TTL = 128
Reply from 10.0.0.3 : byte = 32 time = TTL = 128
Reply from 10.0.0.3 : byte = 32 time = TTL = 128

~~Observation~~ :

Ping statistics for 10.0.0.3 :
Packet : Sent = 4 Reached = 4 Lost = 0
Approx. round trip times in milli seconds :
Minimum = 0 ms Maximum = 2 ms
Average = 0 ms

Observation

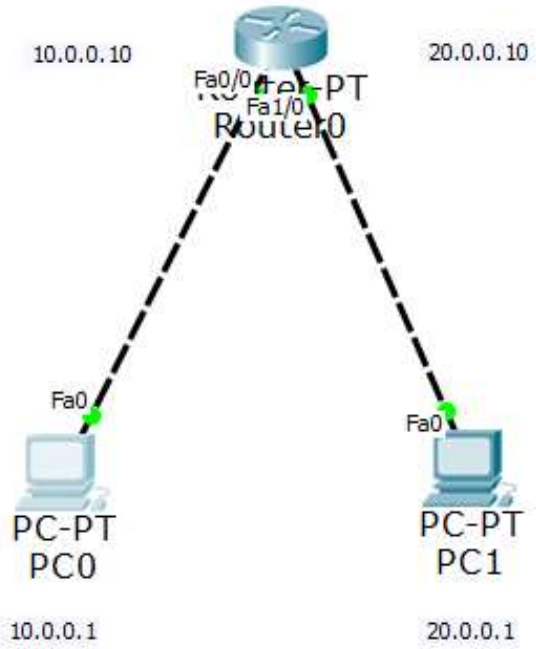
1) Observed that switch transfers the packets to all devices during the 1st iteration and then it records the IP address of the intended destination device & sends the packed to that particular destination
2) Observed that Hub broadcast to packets to all ends devices and which are not intended to devices receive the packets discard to packets and intends device receive the packets and sends back the acknowledgement to the source.

Experiment 2:

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

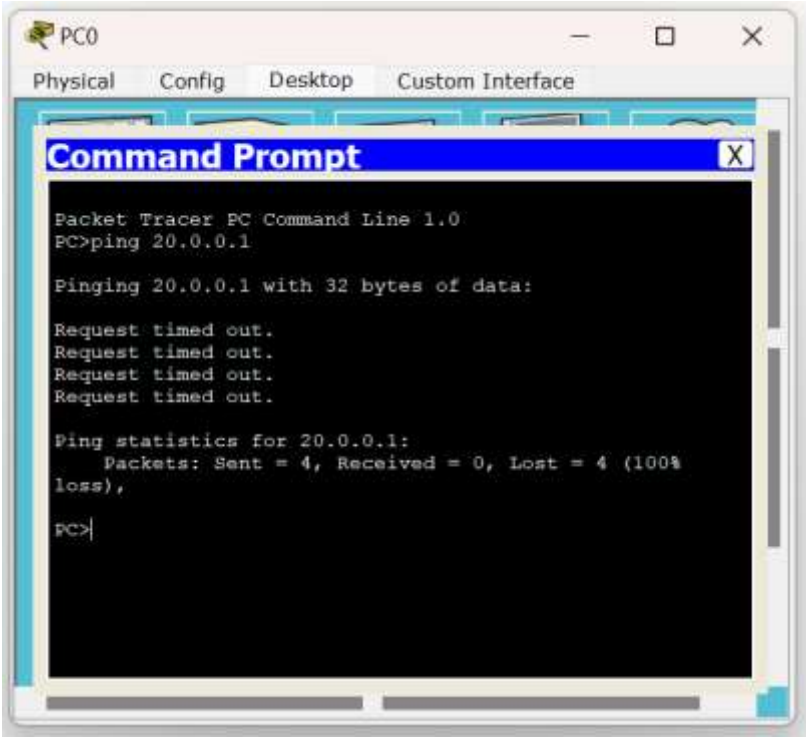
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Topology: (1 router & 2 PC'S)

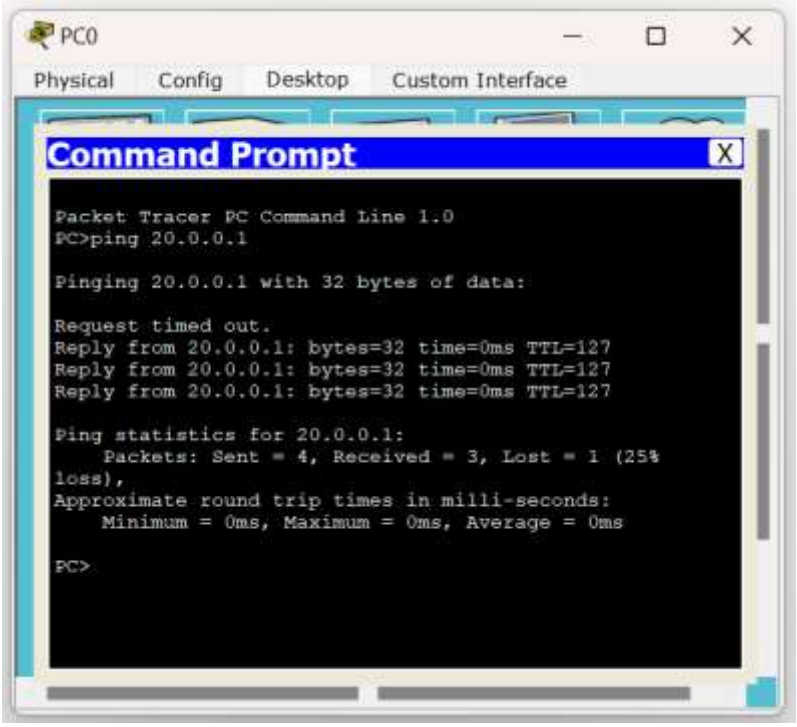


Output:

Before Configuring Gateway:



Before Configuring Gateway:



Observation:

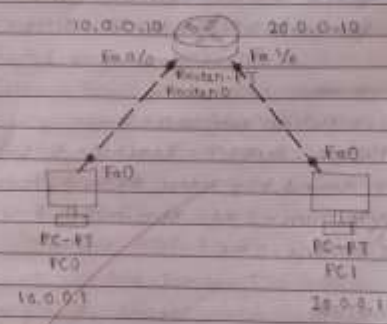
Date 22.6.23
Page 10

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 router in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Topology:



Procedure:

1) Select one Generic Router and two Generic PC-PT's. Connect them with appropriate cables, use copper cross over to connect the devices.
2) Set the IP address of both PC-PT's in the config tab. Along with which, enter gateway in settings option in config tab. PC0 has IP address as 10.0.0.1 and gateway as 10.0.0.10, similarly PC1 has

Date 22.6.23
Page 11

20.0.0.1 and 20.0.0.20 respectively
3) The Router details are configured using the CLI tab (command line interface). First, we need to enable the router and configure terminal.
4) The IP address of both fastethernet 0/0 and fastethernet 1/0 are set as 10.0.0.1 and 20.0.0.1 respectively with its subnet mask as 255.0.0.0
5) The no shut command is used to enable the interface and allow the gateway to communicate which results in green light
6) Ping the PC-PT (20.0.0.1) from PC0, this can be done in the command prompt present in Desktop tab

Output

1) PC > ping 20.0.0.1

Pinging 20.0.0.1 with 32 bits of data:

Request timed out
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=2ms TTL=127

Ping statistics for 20.0.0.1:
Packets: Sent=4, Recieved=3, Lost=1 (25% loss)
Approximate round trip times in milli-seconds:
Minimum=0ms, Maximum=2ms, Average=0ms

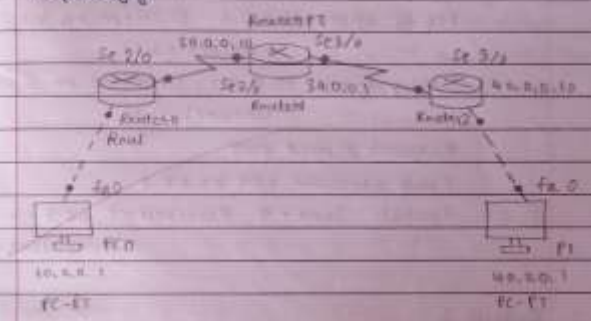
Observation:
Observed that when we ping destination we get allocated 32 bits of data. First 8 bytes are used to learn about the router. Remaining bytes are used for sending packets to destination. Then, again if we ping all bytes are used for message sending and there will be no timed out message.

Experiment 3 Part 1

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

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

Topology:



Procedure

1. Connect 2 PC's and 3 routers using copper cross over cable for PC to router and serial DCE cable to connect router to router.
2. Set IP address of both PC's and their gateway numbers.
3. Now for setting IP address and gateway numbers to routers.
   Select a router and perform following commands. Type enable and enable, followed by typing t. Type interface fast ethernet 0/0 and type the IP address 10.0.0.10 255.0.0.0.

4) The no shut command is used to enable the interface and via communication which results in green light

5) Enter appropriate IP addresses

6) Repeat the process for other routers

Output 1

pc > ping 40.0.0.1

pinging 40.0.0.1 with 32 bytes of data

Reply from 10.0.0.1: Destination host unreachable

Reply from 10.0.0.1: Destination host unreachable

Reply from 10.0.0.1: Destination host unreachable

Request timed out

Ping statistics for 40.0.0.1

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

Output 2

pc > ping 10.0.0.1

pinging with 32 bytes of data

Reply from 10.0.0.1: bytes = 32 time = 2ms TTL = 125

Reply from 10.0.0.1: bytes = 32 time = 3ms TTL = 125

Reply from 10.0.0.1: bytes = 32 time = 2ms TTL = 125

Reply from 10.0.0.1 = bytes = 32 time = 3ms TTL = 125

Ping statistics for 10.0.0.1

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

Approximate round trip times in milliseconds

Minimum = 2ms, maximum = 8ms

Average = 3ms

Observation

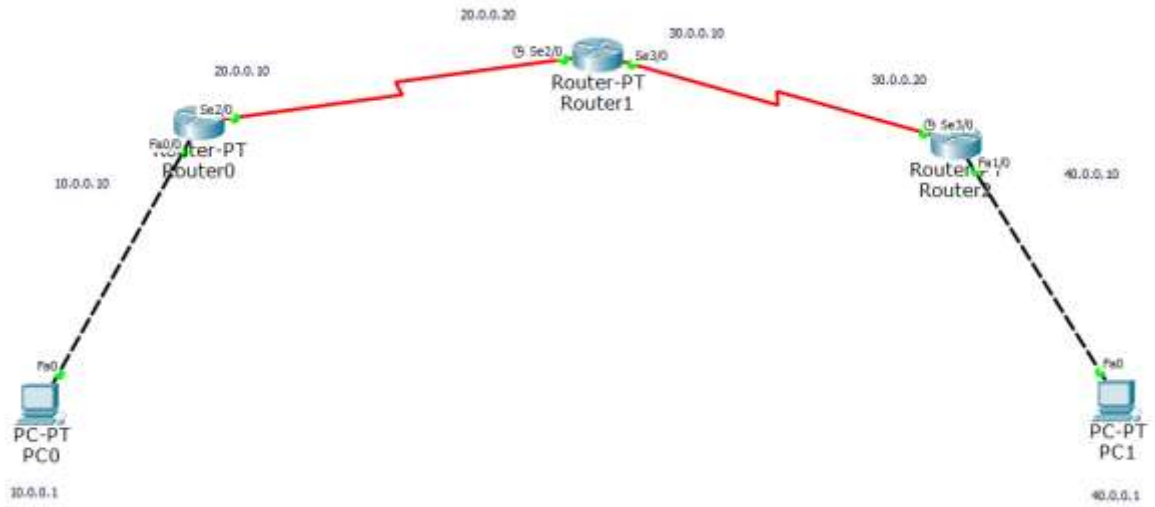
When the routers are not introduced with other two IP addresses and we ping them to get a message output saying host unreachable. Once when we introduce routers with other two IP addresses and we ping, now we get message sent successfully with 0% loss.

Experiment 3:

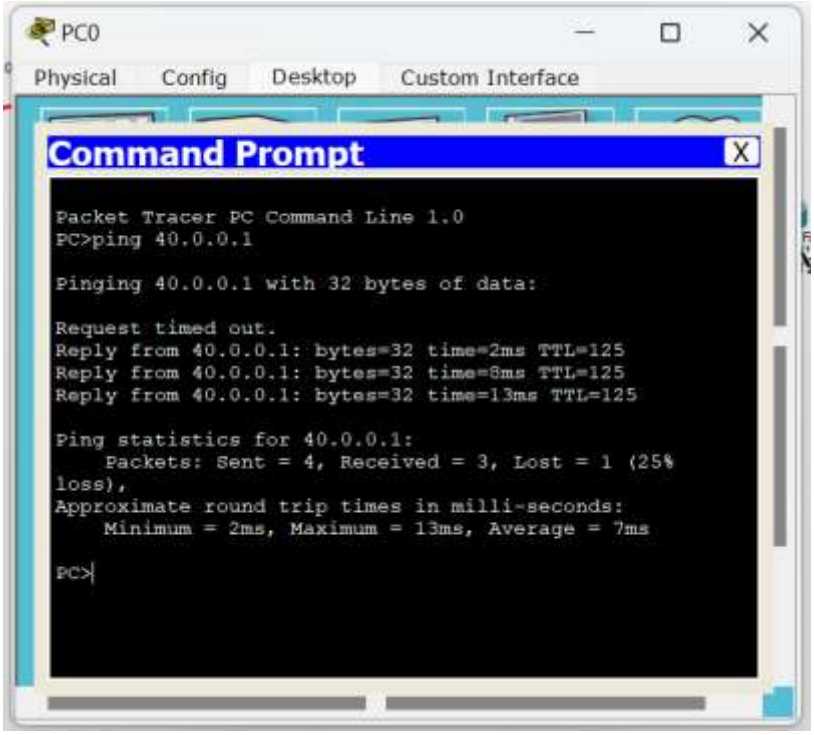
Configure default route, static route to the Router

Aim: Configure default route, static route to the Router

Topology:



Output:



Observation:

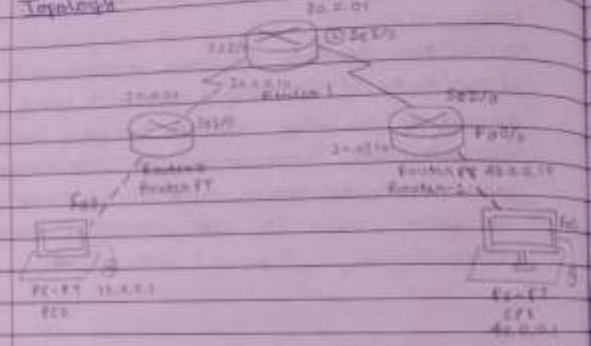
Date 13.7.23
Page 16

Experiment 2

Configure default route, static route to the Router

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

Topology:



Procedure:

1) Select 3 Routers and 2 PC's. Connect them with appropriate cables.
2) Set the IP address of the PC's and gateway to the Routers in settings option in config tab. PC has IP address 10.0.0.1 and 40.0.0.1 while routers have Network IP as 20, 30 and 40
3) The Router is configured using the CLI tab (Command Line Interface). First, we need to enable the router and config terminal
4) Router 0 and Router 2 are configured statically and Router 1 is default static Router.
The Subnet Mask for Default is 0.0.0.0 indicating that any number can sent at any

Date 13.7.23
Page 17

any system in the network.
5) Gateway of last resort is always not set for default routers
6) Ping the PC's 10.0.0.1 and 40.0.0.1 to test the network for effective communication. This can be done in Command Prompt present in Desktop tab.

Output:

1) PC > ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time=123ms
TTL=253

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

Reply from 40.0.0.1: bytes=32 time=3ms
TTL=253

Reply from 40.0.0.1: bytes=32 time=12ms
TTL=253

Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0,
(0% loss)

Approximate round trip times in milli-seconds:
Minimum: 3ms, Maximum = 123ms,
Average = 36ms

Observation

When the routers are not introduced to the IP addresses, they don't know about the networks beyond their scope. We can route the routers statically or by the method of default routing. Default routing allows us to expand the scope and allow the packets to ping successfully.

Static Routing will let the router know what has to be done when a packet is sent. Hence, we get 0% loss which implies that pinging was 100% successful

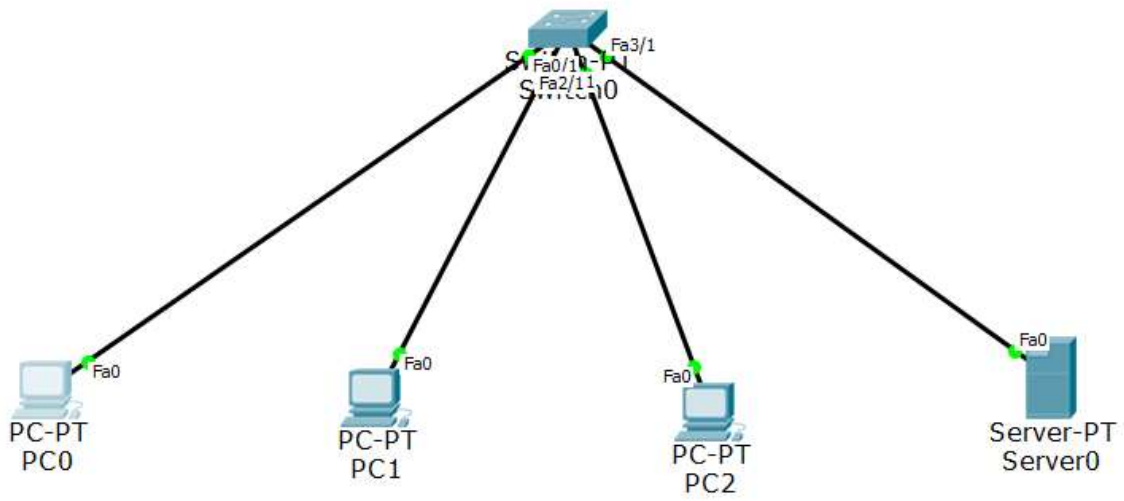
Experiment 4:

Configure DHCP within a LAN and outside LAN.

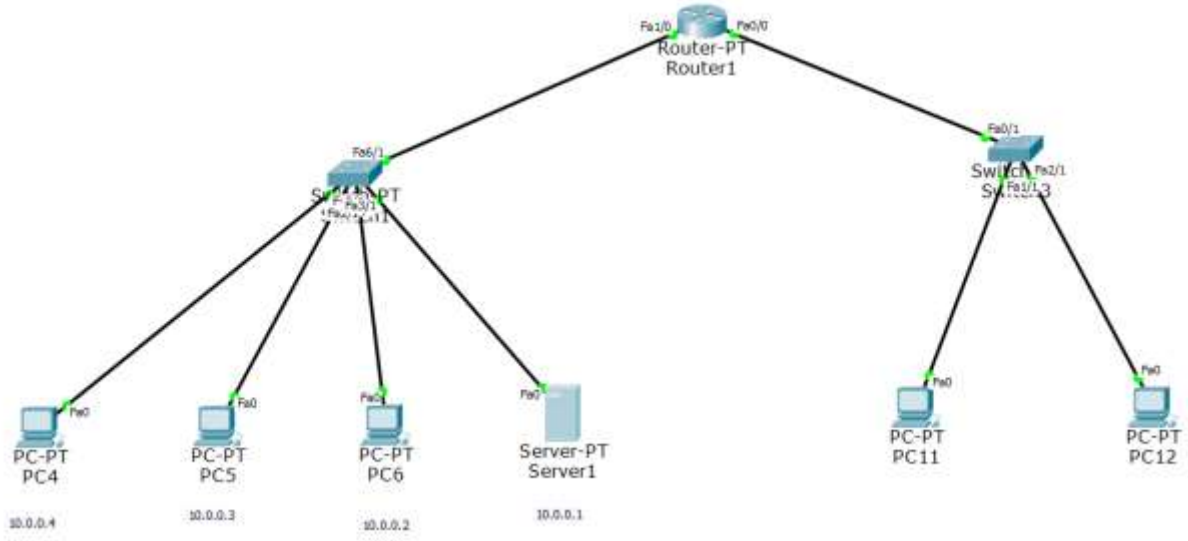
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

(Within a LAN)

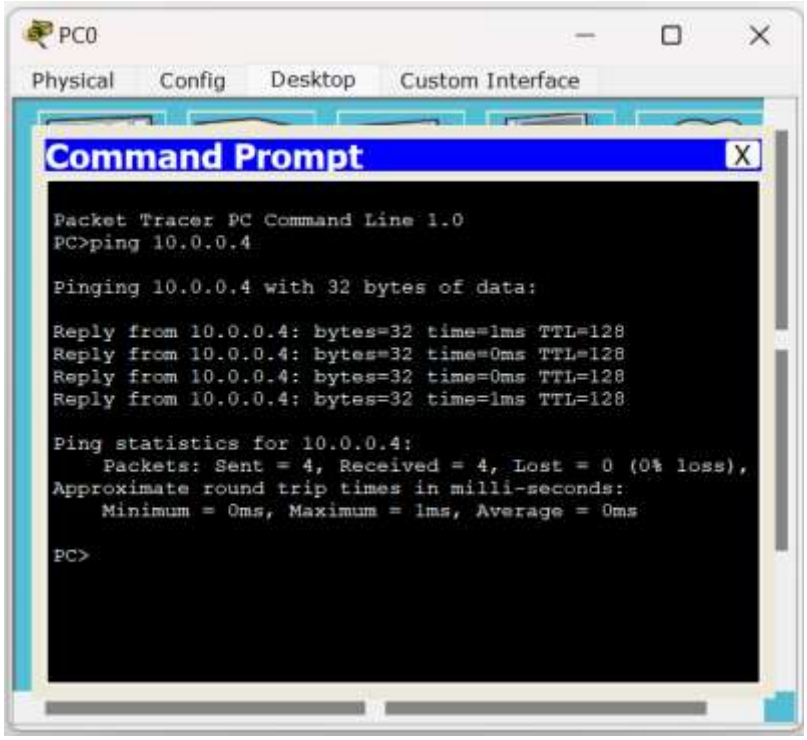


(Outside a LAN)

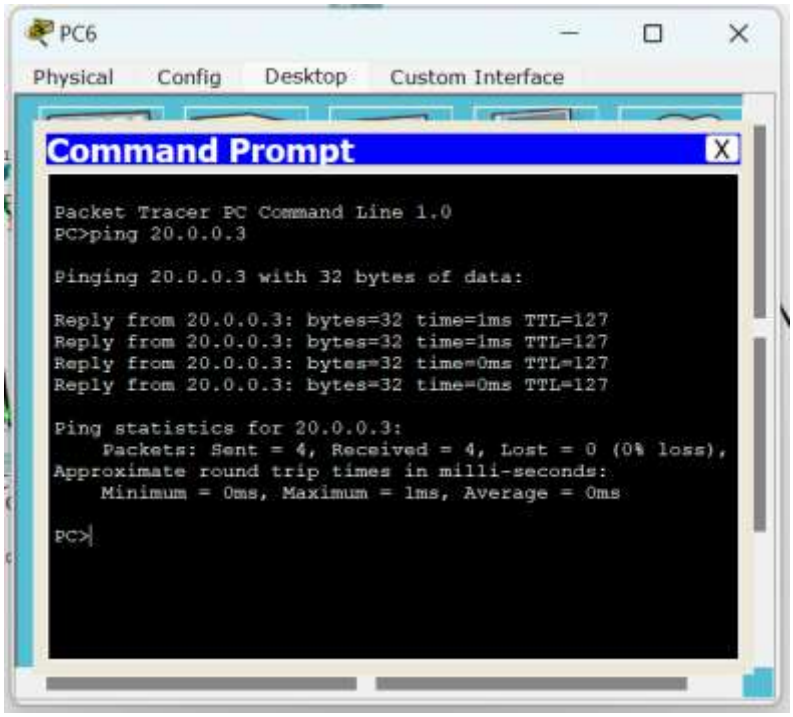


Output:

(Within a LAN)



(Outside a LAN)



<u>Observation:</u>

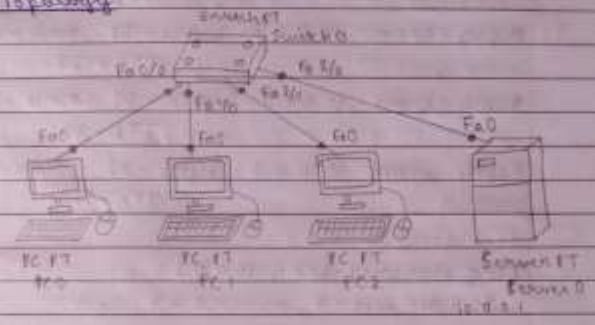
Date: 13.7.23
Page: 19

Experiment 4.1

Configure DHCP within a LAN and outside LAN

Aim: Configure DHCP within a LAN and outside LAN

Topology:



Procedure:

1) Connect 3 PC's and one server to a switch using copper straight through cable
2) Click on server and go to services tab. Select DHCP and turn on the DHCP service
3) Set the IP address of the start IP address as 10.0.0.2 and click on save button
4) Before this, set the IP address of server in config tab under gateway as 10.0.0.1
5) Next click on PC0 and go to desktop tab, here click on IP configuration. Select DHCP here. It will request for an address and successfully gets the DHCP request also sets the IP address.
6) Do steps previously mentioned for other 2 PC's
7) In order to send a packet across, go to command prompt and type ping destination IP address

Date: 13.7.23
Page: 20

Output

PC > ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128

Reply from 10.0.0.3: bytes=32 time=1ms TTL=128

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128

Ping statistics from 10.0.0.3:
Packets: sent=4, received=4, lost=0 (0% loss)

Approximate round trip times in milliseconds:
Minimum=0ms, Maximum=1ms, Average=0ms

Observation

DHCP (Dynamic Host Configuration Protocol) is used to dynamically assign an IP address to any device or node. It is a client server protocol in which servers manage a pool of unique IP addresses and also about client configuration parameters.

DHCP enabled clients sends a request to DHCP server when they want to connect to network. IP configuration information from address pools is how DHCP server responds to the client request.

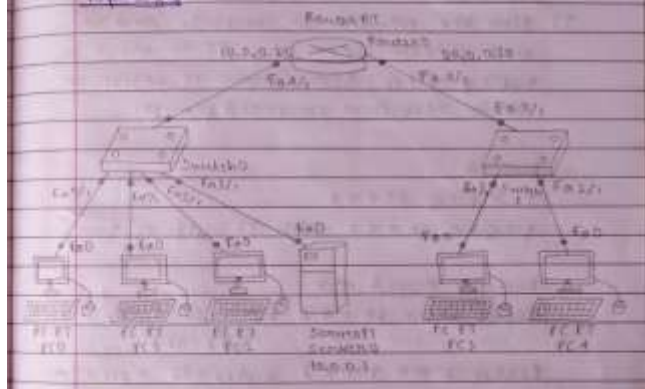
Date 13/7/23
Page 21

Experiment 4.2

Configure DHCP within a LAN and outside LAN

Aim Configure DHCP within a LAN and outside LAN

Topology



Procedure

1) Add a Router, a switch and 2 PC's to the previous experiment Network and connect the router to both switches
2) Set the server IP address of server and with server set the first 3 PC's IP address through DHCP
3) Set the Router static IPs IP address statically by using terminal
4) Go to server and set the gateway as 10.0.0.30 defined by which No ghost in Packet Tr
5) Now go to services and add one more pool name as server Pool 1, start IP address as 20.0.0.2 and default gateway as 20.0.0.30. Then click add and save.
6) Now set the other two PC's IP address by going to IP configuration automatically generate IP address by selecting DHCP.
7) Now the network is complete and can send packets from any PC to other by typing ping destination IP address in their respective command prompt

Output

PC > ping 20.0.0.2

pinging 20.0.0.2 with 32 bytes of data

Request timed out

Reply from 20.0.0.2: bytes=32 time=0ms TTL=127

Reply from 20.0.0.2: bytes=32 time=0ms TTL=127

Reply from 20.0.0.2: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.2

Packets: sent=4, received=3, lost=1 (25% loss)

Approximate round trip times in milliseconds

Minimum=0ms, Maximum=0ms, Average=0ms

Observation

DHCP is used to assign IP addresses dynamically to different devices. To assign continuous IP address we create a server pool where we assign the starting IP address and a default gateway number for PC's under different

Date 13/7/23
Page 23

switches we create a different server pool again and start. This takes care of delivering the packets to correct destination IP address and also sends back the ACK (Acknowledgement) to initial device.

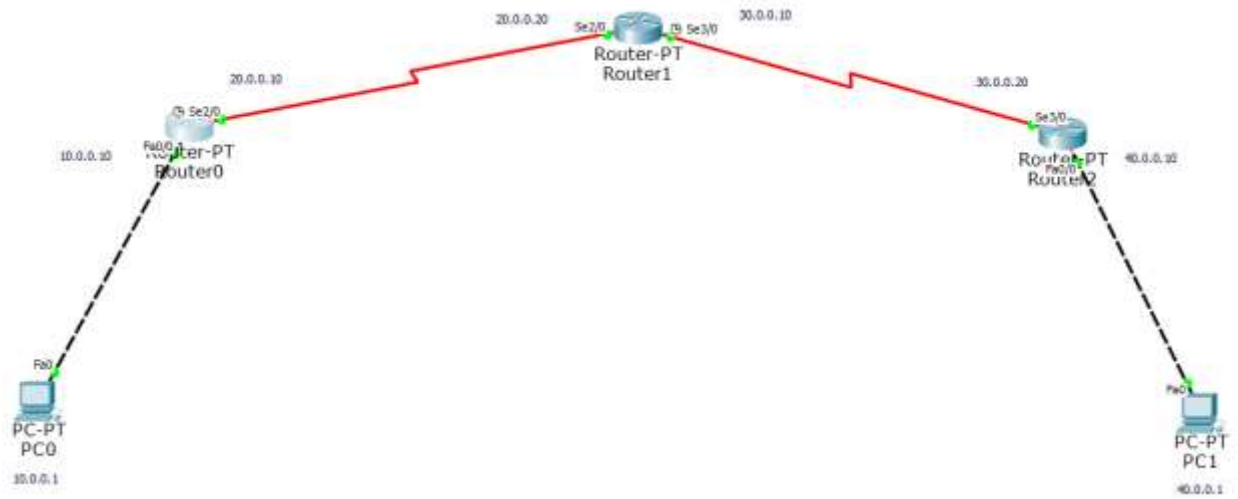
20/7/23

Experiment 5:

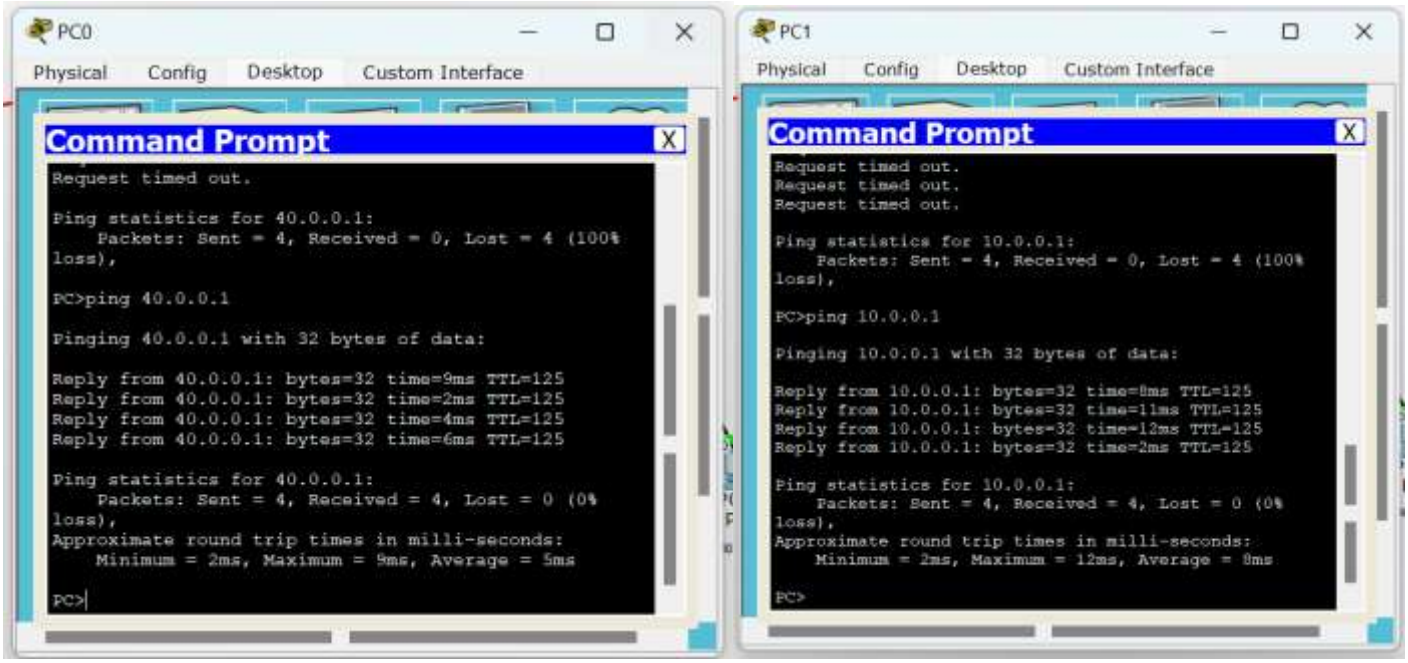
Configure RIP routing Protocol in Routers

Aim: Configure RIP routing Protocol in Routers

Topology:



Output:



Observation:

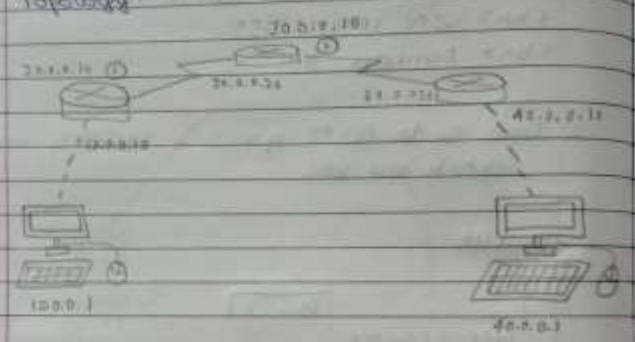
Page 26

Experiment 6

Configure Router Information Protocol in Routers

Aim Configure Router Information Protocol in Routers

Topology



Procedure

1) Create a Network using 3 Routers, 2 PC's and connect them with suitable cables
2) Set IP address, gateway for PC's as 10.0.0.1, 10.0.0.10 and 40.0.0.1, 40.0.0.10
3) Configure Router, execute configuration of terminal and all the other interfaces.
4) Encapsulation PPP and clock rate 64000 is done for routers with clock symbol.
5) Repeat for all Routers and using CLI (Command line interface) to do the Configurations

Ping output

ping 40.0.0.1

pinging 40.0.0.1 with 32 bytes of data

Page 27

Request timed out

Reply from 40.0.0.1 bytes = 32 time = 2ms TTL = 128

Reply from 40.0.0.1 bytes = 32 time = 5ms TTL =

Reply from 40.0.0.1 bytes = 32 time = 10ms TTL

Ping statistics for 40.0.0.1

Packets sent = 4, Received = 3, Lost = 1 (25% Loss)

Approximate round time in milliseconds

Minimum = 5ms Maximum = 10ms Average = 8ms

Observations

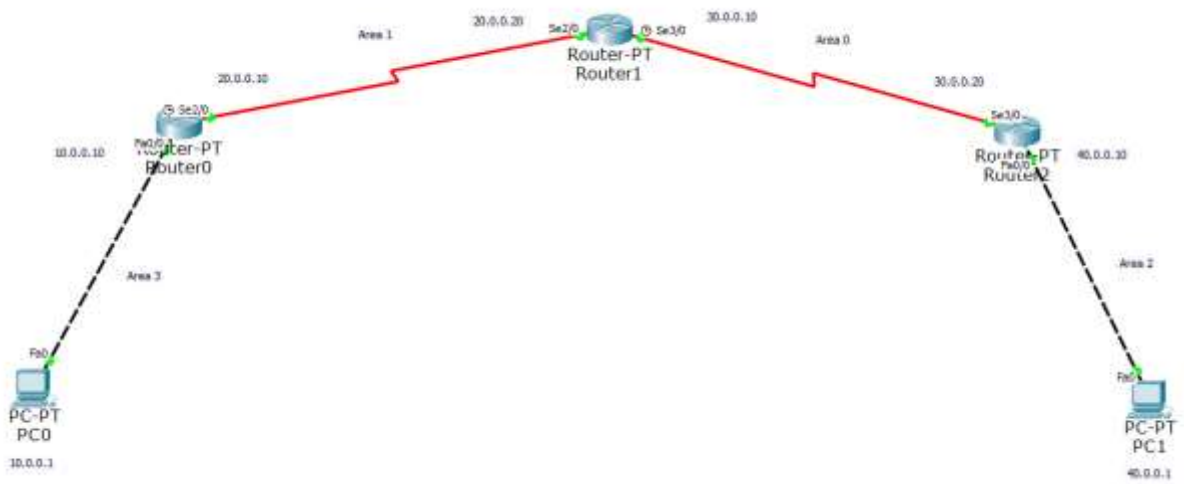
1) RIP uses hop count as a routing metric to find best path between source and destination
2) HOP count are the nodes available between source and destination and the path with least hop count is selected
3) Updates of network are exchanged periodically and that of routing information is always a broadcast
4) Routing tables are sent in updates
5) The Routers always trust sending in updates information which is received from neighbouring routers

Experiment 6:

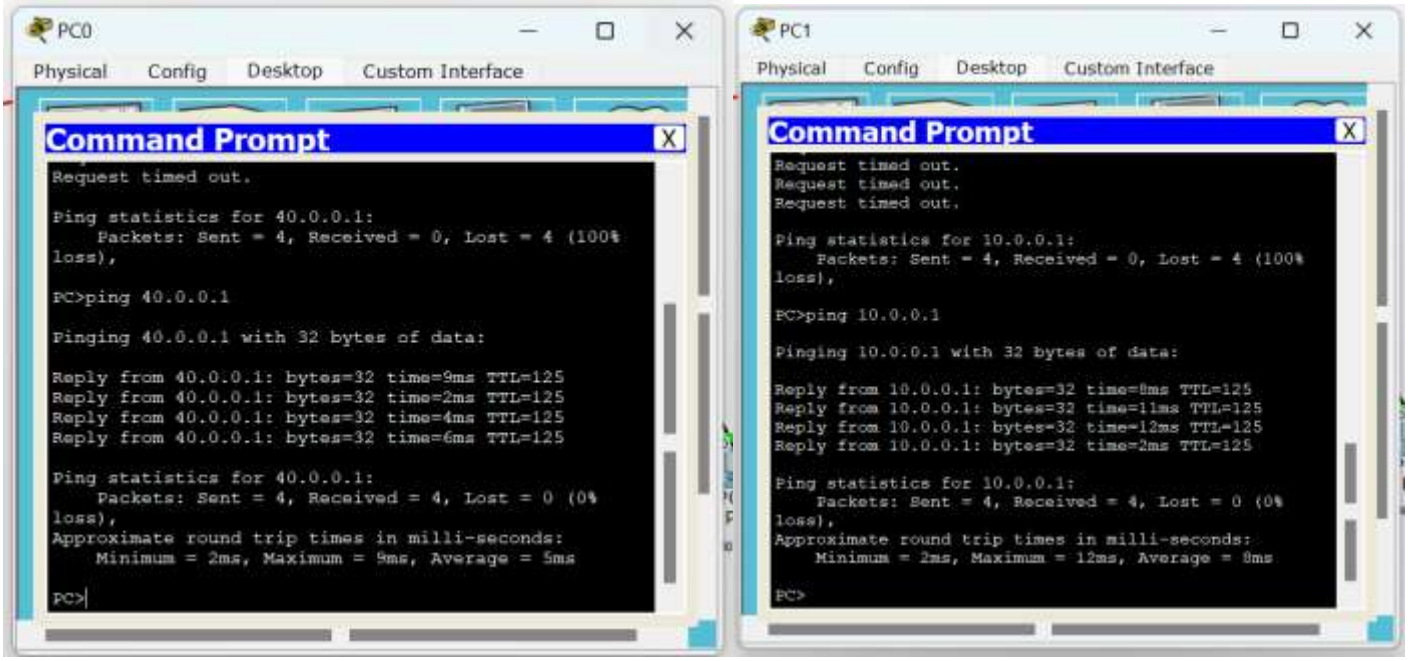
Configure OSPF routing protocol

Aim: Configure OSPF routing protocol

Topology:



Output:



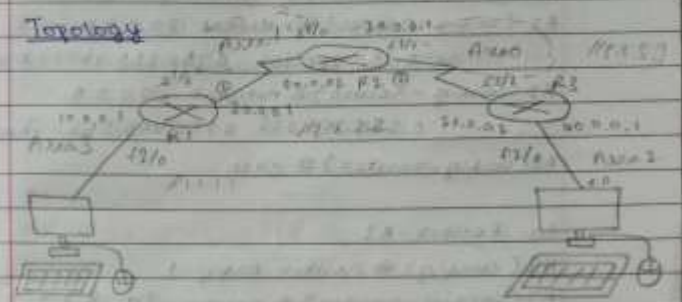
Observation:

Experiment 7

Configure OSPF Routing Protocol

Aim: Configure OSPF (Open Shortest Path First) Routing Protocol.

Topology



Procedure

1) Configure the PC's with IP address and gateway according to the topology seen above
2) Configure each of the Routers according to the IP addresses given in topology
3) Encapsulation ppp and clock rate need to be set as done in Router Information Protocol Experiment
4) Now, enable OSPF protocol.

In Router R1,

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

In Router R2,

```
R2(config)# router ospf 1
R2(config-router)# router-id 2.2.2.2
R2(config-router)# network 20.0.0.0
        0.255.255.255  0.255.255.255  area 1
R2(config-router)# network 30.0.0.0
        0.255.255.255  0.255.255.255  area 0
R2(config-router)# exit
```

In Router R3,

```
R3(config)# router ospf 1
R3(config-router)# router-id 3.3.3.3
R3(config-router)# network 30.0.0.0
        0.255.255.255  0.255.255.255  area 0
R3(config-router)# network 40.0.0.0
        0.255.255.255  0.255.255.255  area 3
R3(config-router)# exit
```

5)

```
R1(config-if)# interface loopback 0
R1(config-if)# ip add 172.16.1.252 255.255.255.0
R1(config-if)# no shut

R2(config-if)# interface loopback 0
R2(config-if)# ip add 172.16.1.253 255.255.255.0
R2(config-if)# no shut

R3(config-if)# interface loopback 0
R3(config-if)# ip add 172.16.1.254 255.255.255.0
R3(config-if)# no shut
```

6) Virtual link between R1 and R2.

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

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

7) Show ip Route is executed

Observation

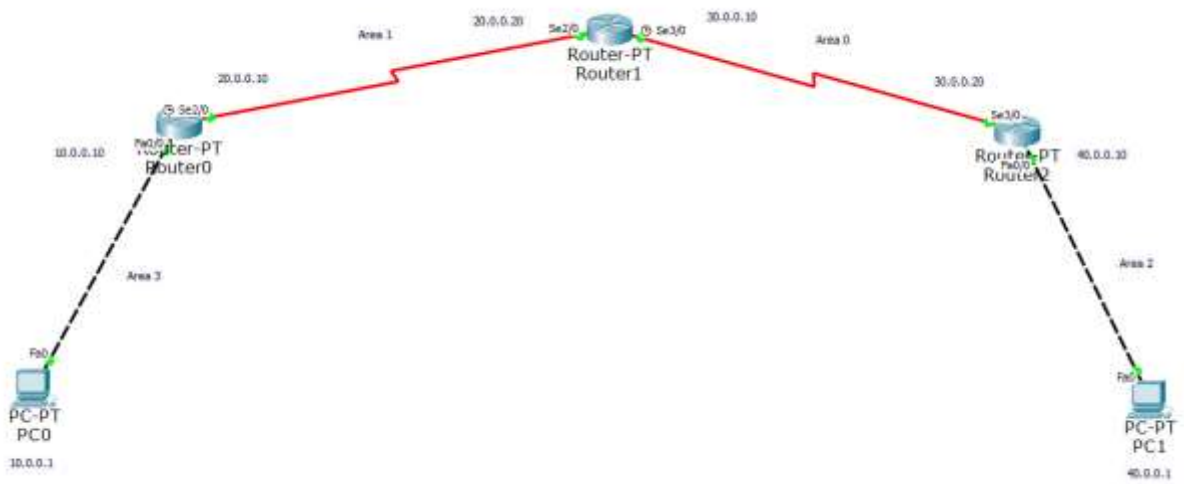
1) OSPF is a link status routing protocol which is used to find the best path between source and destination router using its own SPF algorithm (Shortest Protocol or Priority First)
2) This network is divided into 4 areas where area is the backbone.
3) After we make virtual link between the area which is not connected to backbone, we can ping messages successfully.

Experiment 7:

Demonstrate the TTL/ Life of a Packet

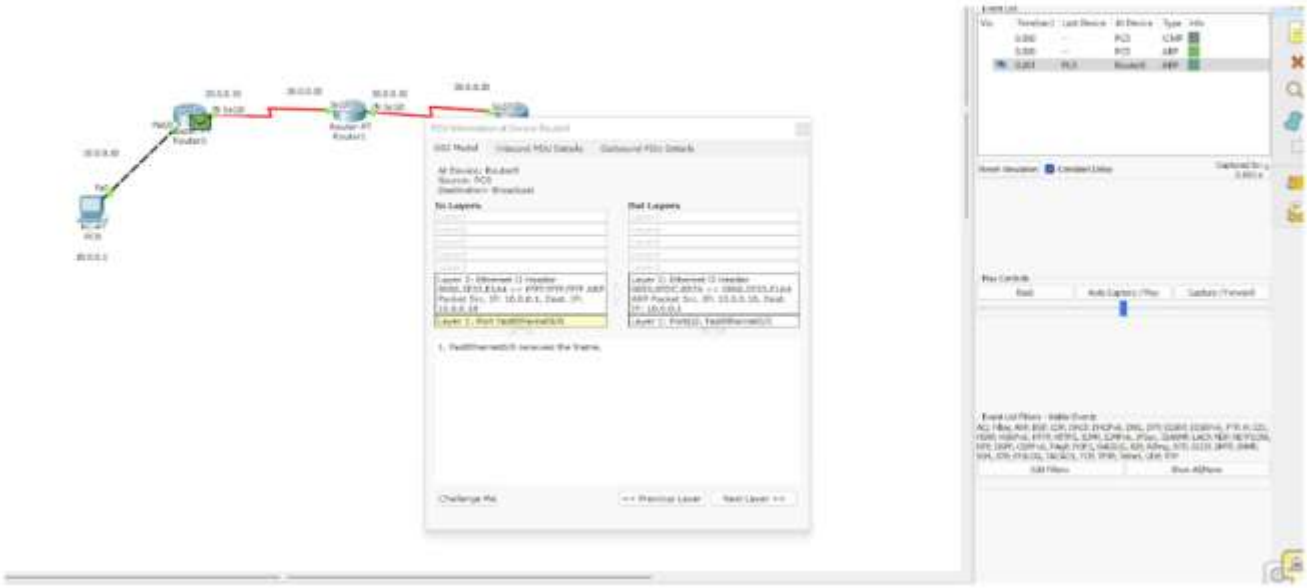
Aim: Demonstrate the TTL/ Life of a Packet

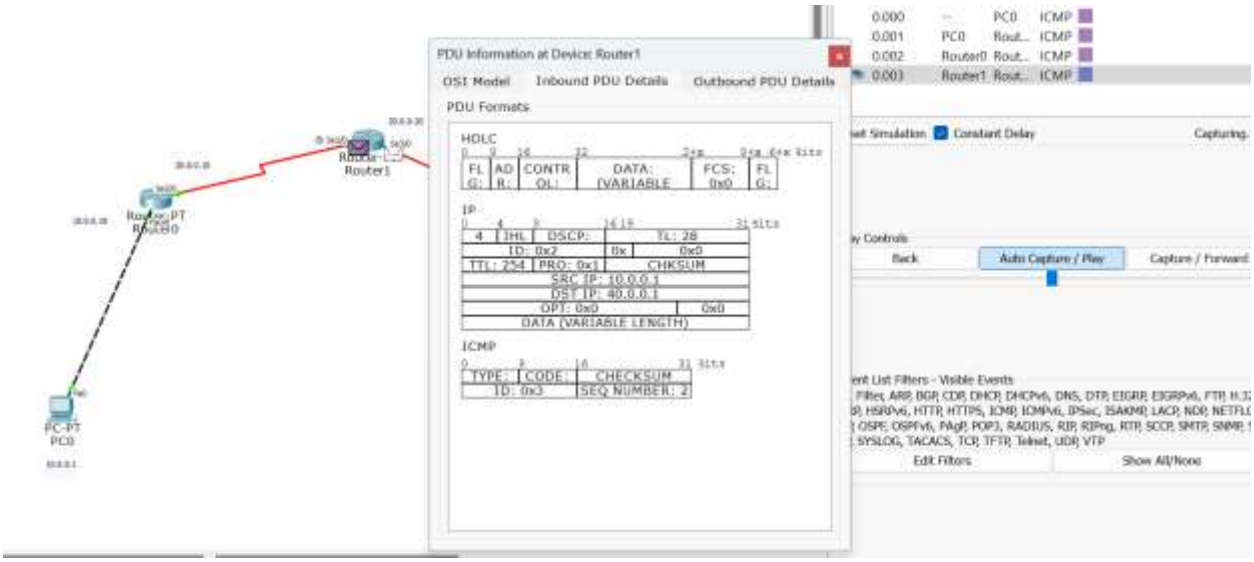
Topology:

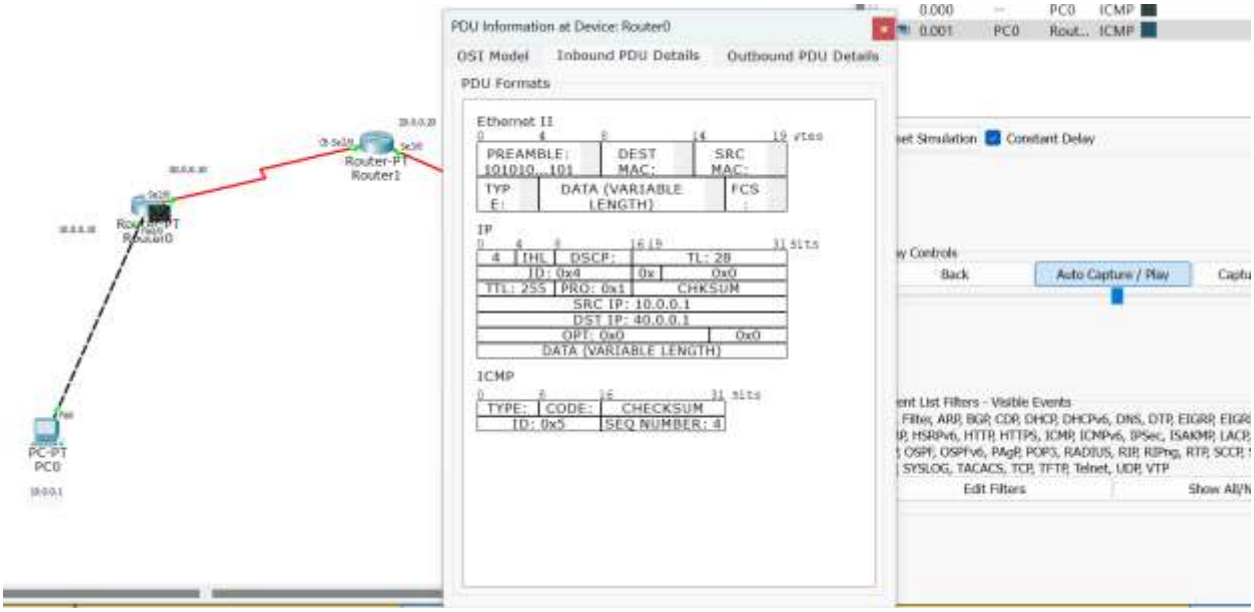


Output:

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
FL G:
AD R:
CONTR OL:
DATA: (VARIABLE
FCS: 0x0
FL G:
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
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
0.000
PC0
ICMP
0.001
PC0
Rout...
ICMP
Constant Delay
Back
Auto Capture / Play
Edit Filters

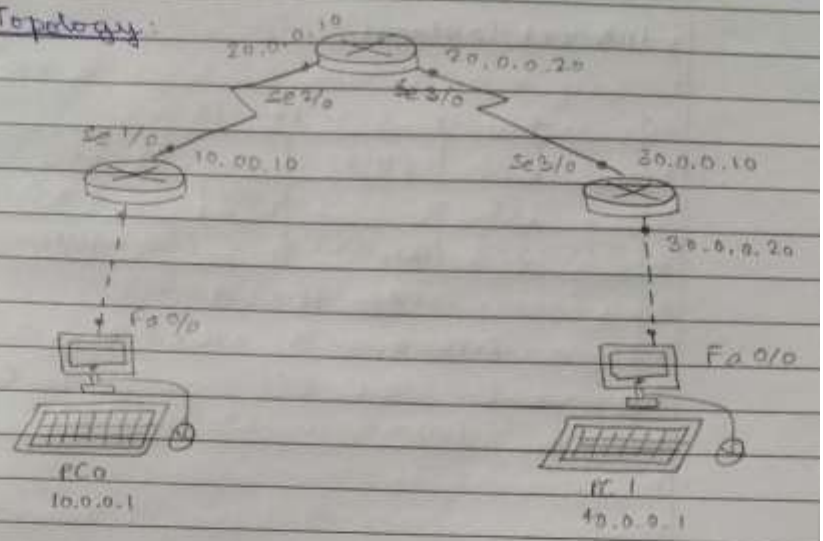
Observation:

Date / /
Page 37

Experiment 10

To Determine TTL of a Packet (Time-to-live)

Topology:



Procedure

- Create a topology as shown above with 3 Routers and 2 PC's
- Set IP gateway for both PC's, configure the router either by static / default routing methods
- In simulation mode on bottom right corner, send PDU (Packet Data unit) from one PC to another
- Use capture buttons for every transfer

Output

Internet Protocol

0	4	8	16	19	31
4	IHL	DHCP	TL 28		
10.0x6			0x	0x0	
TTL: 255		PRO: 0x1	Checksum		
Sender IP 10.0.0.1					
Destination IP 40.0.0.1					
Opt 0x0				0x0	
Data (Variable length)					

Observation

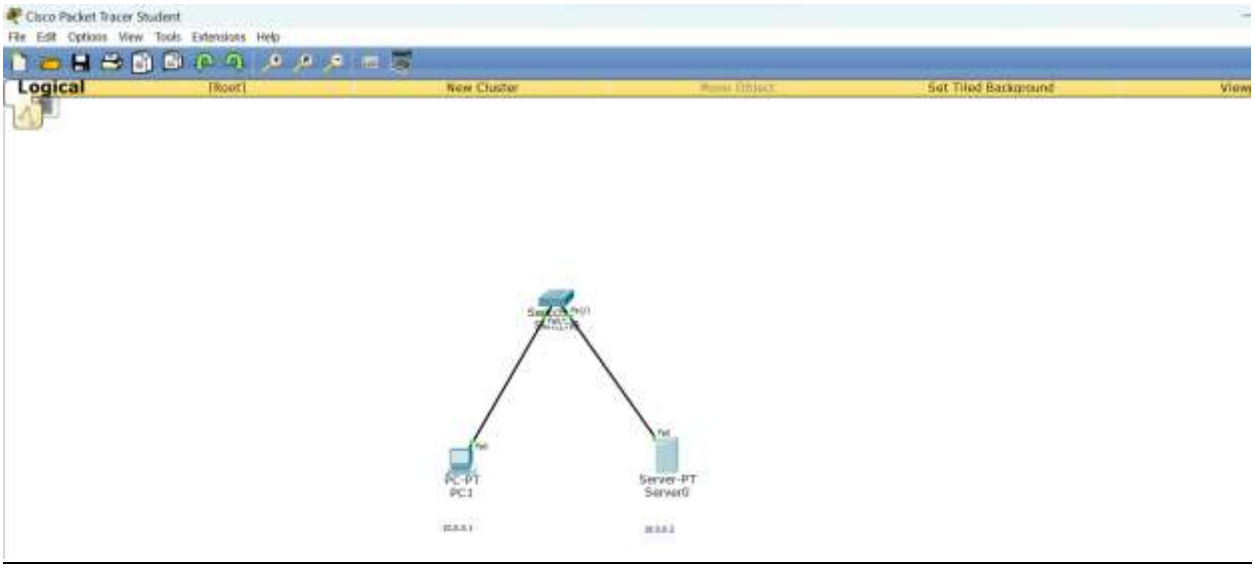
- The number of hops, the packet travels before being discarded is TTL
- It is set by router, reduced by each router along the path
- It reduces TTL by one while forwarding
- When it reaches 0, it discards it, sends an ICMP message

Experiment 8:

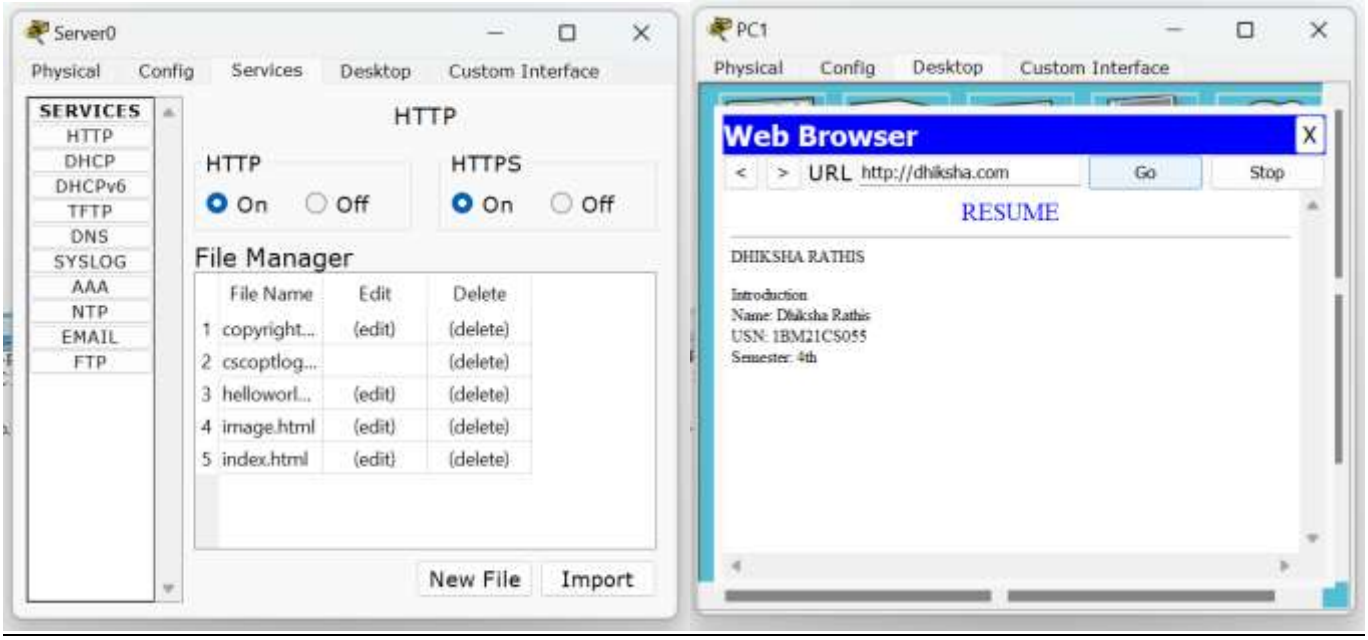
Configure Web Server, DNS within a LAN.

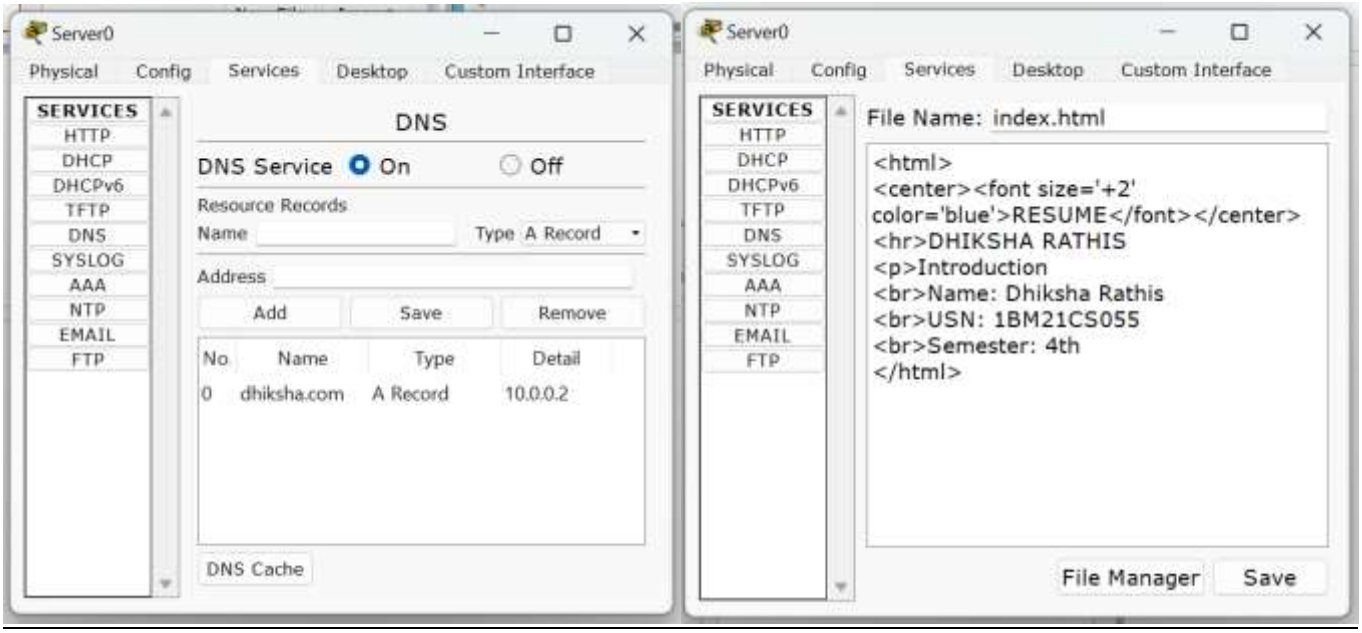
Aim: Configure Web Server, DNS within a LAN.

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
Web
PC
VPN
MIB
Cisco IP
Email
PPPoE
Text
Firewall
IPv6
PC1
Physical
Config
Desktop
Custom Interface
Web Browser
URL http://dhiksha.com
Go
Stop
RESUME
DHIKSHA RATHIS
Introduction
Name: Dhiksha Rathis
USN: 1BM21CS055
Semester: 4th

Observation:

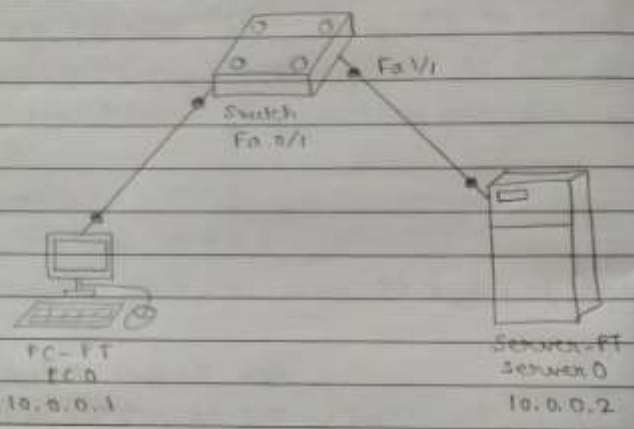
Date / /
Page 24

Experiment 5

Configure Web Server, DNS within a LAN

Aim: Configure Web Server, DNS (Domain Name System) within a LAN.

Topology:



Procedure:

1) Add a Personal Computer PC, a switch and a Server in the logical interface
2) Place a switch PT and connect it to the PC-PT and server using copper straight-through cable
3) Configure the IP address for the PC-PT and server
4) In server, add name, 10.0.0.2 and then add the FastEthernet 1/1 switch
5) Add the URL in PC
6) To make changes go to server → services → HTTP make changes in the index
7) To make changes, turn HTTP on, go to DNS, add name and address

in HTTP, click on edit index

```
<html>
<center><font size='+2' color='green'>
        RESUME </font></center>
<hr> Introduction
<br> Name: Dhiksha Rathis
<br> USN: 1BM21CS055
<br> Semester: 4th
```

8) Then Go to PC-PT open web browser and search the URL

Output:

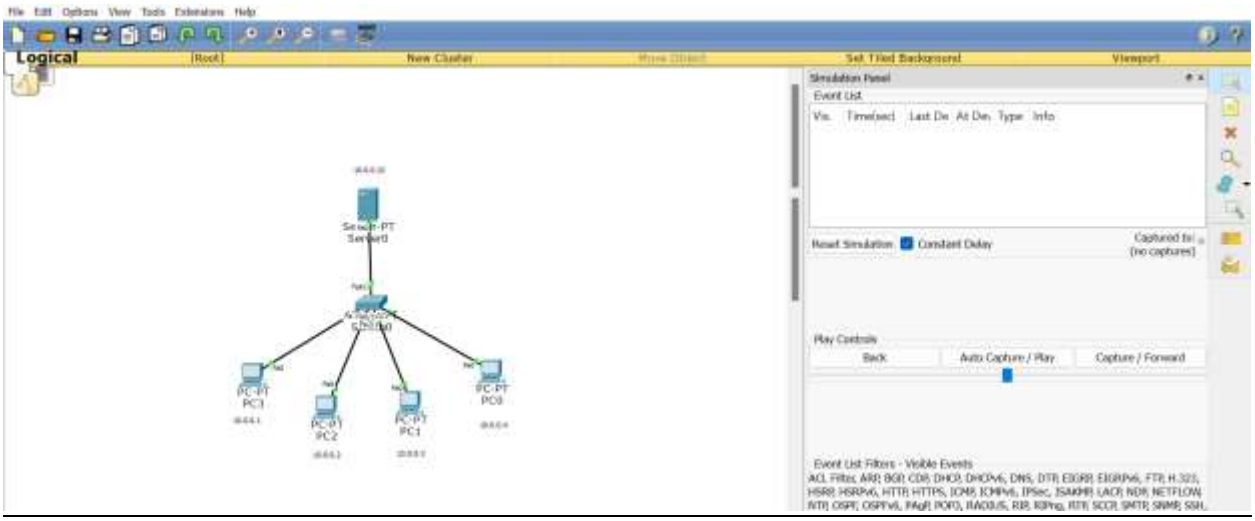
Desktop
WEB BROWSER
URL http://google.com [Go] [STOP]
RESUME
Introduction
Name: Dhiksha Rathis
USN: 1BM21CS055
Semester: 4th

Experiment 9:

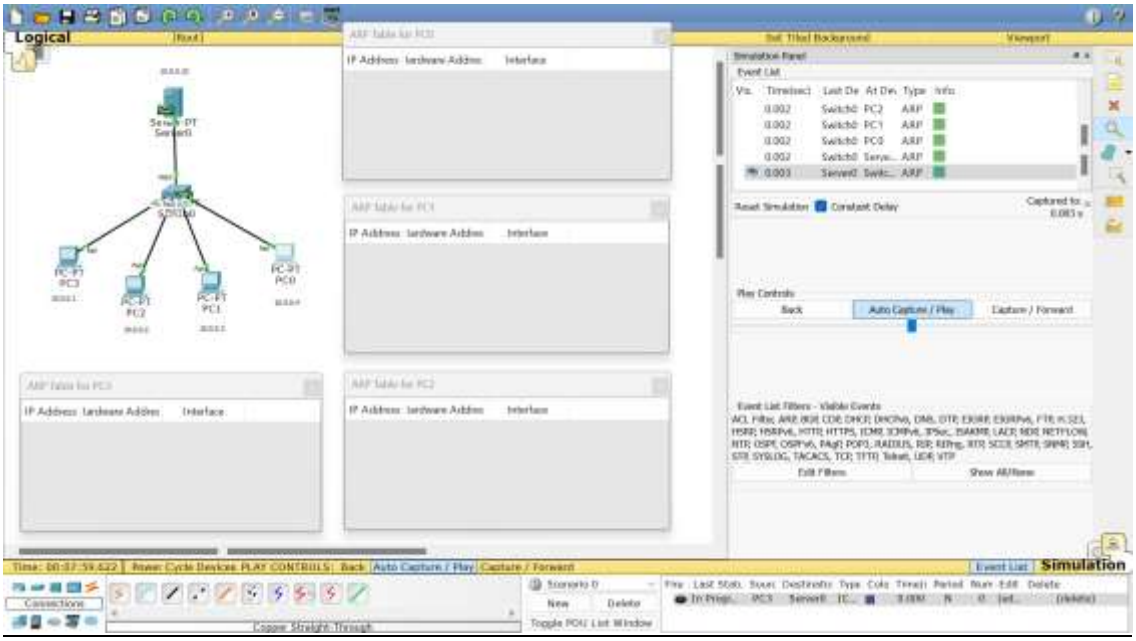
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

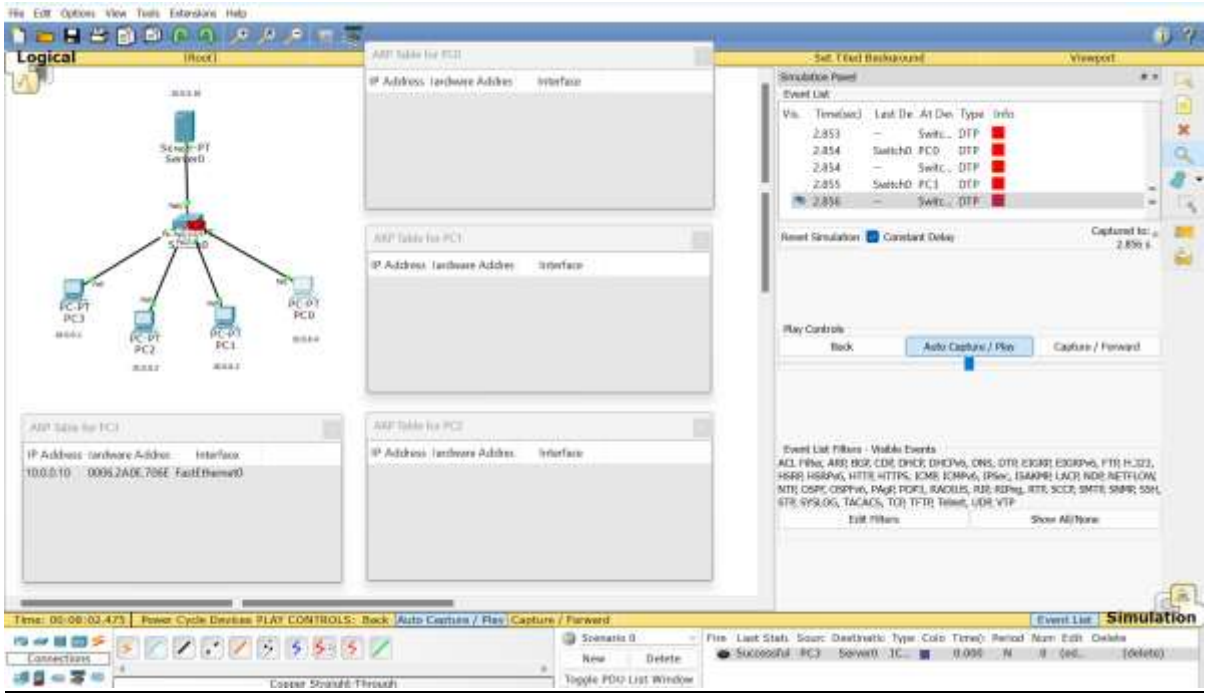
Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

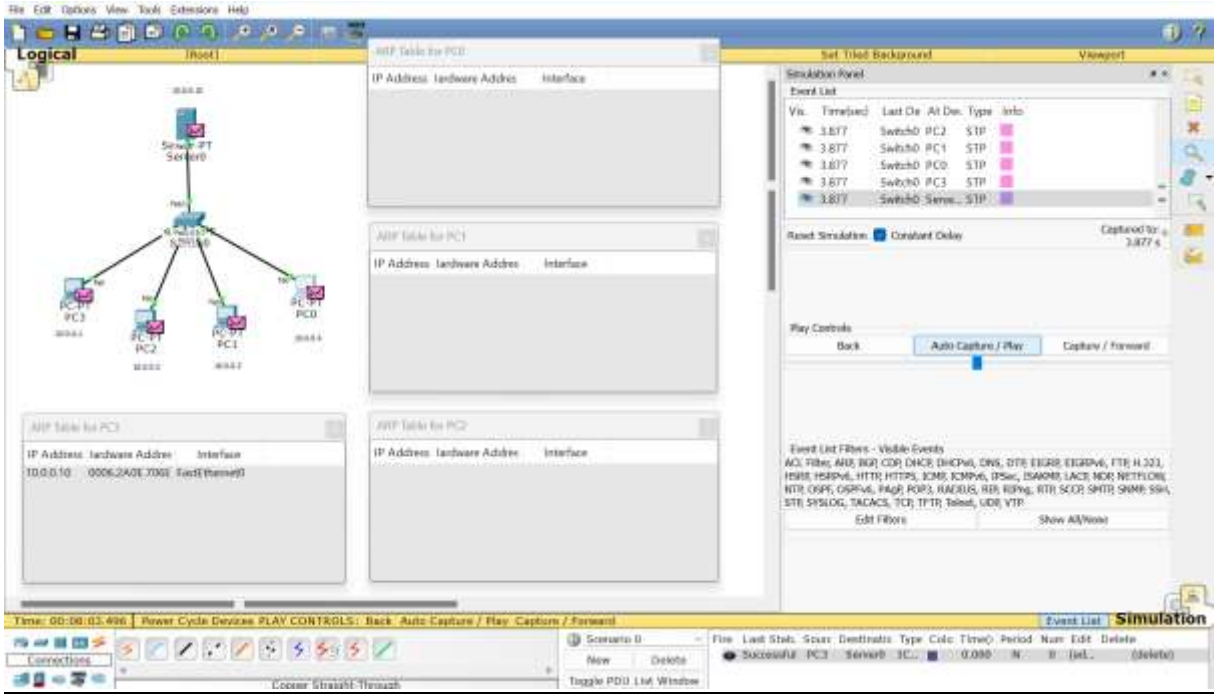
Topology:

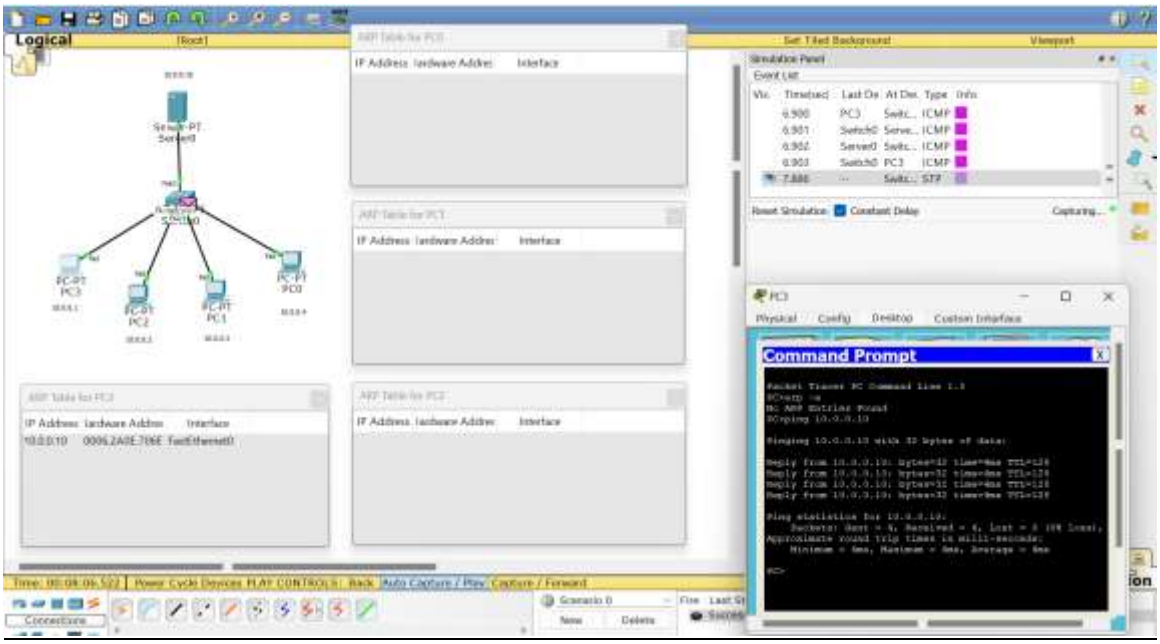


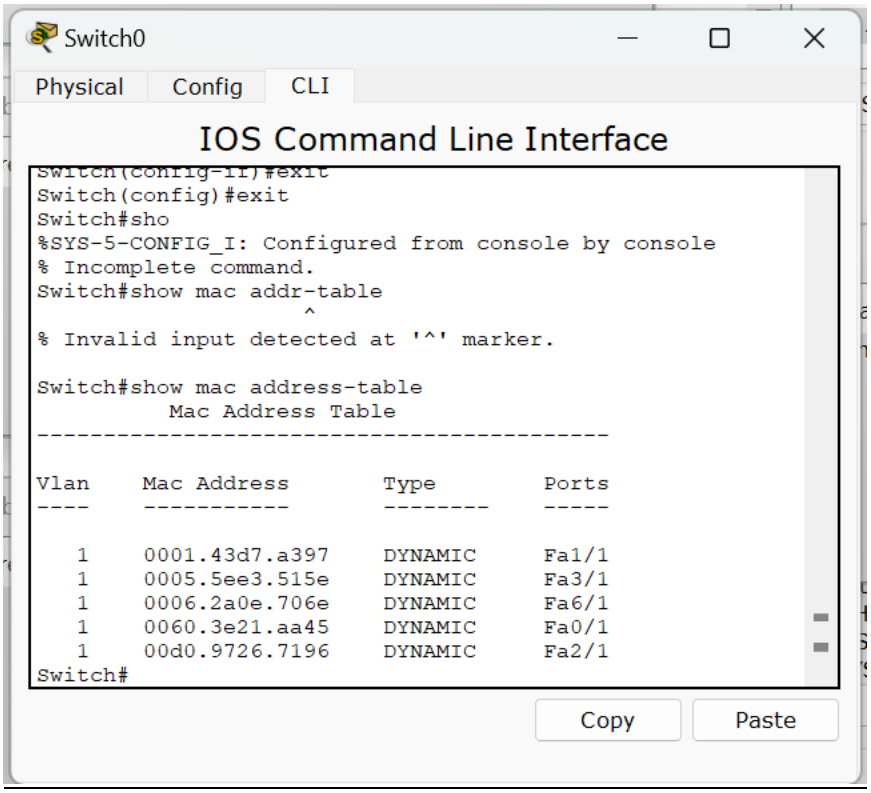
Output:




Logical
Simulation Panel
Event List
Play Controls
Back
Auto Capture / Play
Capture / Forward
Simulation


Logical
Simulation Panel
Event List
Play Controls
Back
Auto Capture / Play
Capture / Forward
Simulation


Logical
Command Prompt


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>
```

PC1

Physical Config Desktop Custom Interface

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>arp -a
  Internet Address      Physical Address      Type
  10.0.0.2              0001.43d7.a397        dynamic
  10.0.0.4              0005.5ee3.515e        dynamic

PC>
```

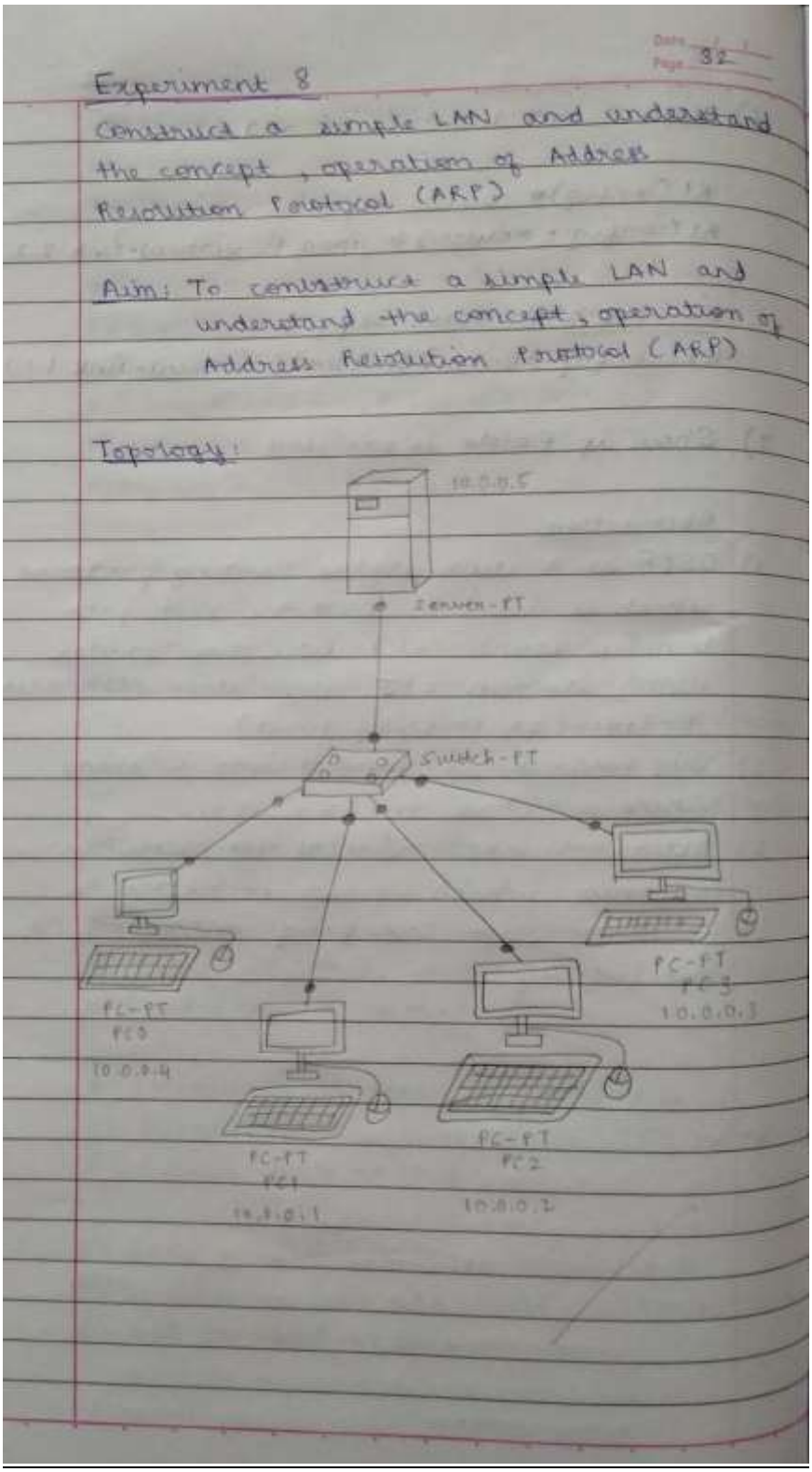
PC0

Physical Config Desktop Custom Interface

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>arp -a
  Internet Address      Physical Address      Type
  10.0.0.3              00d0.9726.7196        dynamic

PC>
```

Observation:



Experiment 8

Construct a simple LAN and understand the concept, operation of Address Resolution Protocol (ARP)

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

Topology:

Procedure:

1) Create a Topology of 4 PC's and a server and IP addresses are assigned to all PC's. Connect them through a switch
2) Use the inspect tool to click on PC to obtain ARP table. Alternatively arp -a in ~~CLI~~ CMD will also do the same.
3) Initially ARP is empty.
4) Also in CLI of switch, the command show mac address-table can be given on every transaction to see how the switch learns from transactions and build the address table
5) Use Capture / Forward in the Simulation Panel to go step by step so the changes ARP can be clearly noted
6) Nodes and switches get updated in ARP table and new connections start.

Command Prompt

PC > arp -a

Internet address	Physical Addresses	Type
10.0.0.2	0005.5e6a.7da2	Dynamic
10.0.0.3	0030.f285.7a19	Dynamic
10.0.0.4	0001.6383.dd62	Dynamic
10.0.0.5	0004.9a42.616c	Dynamic

MAC Address Table

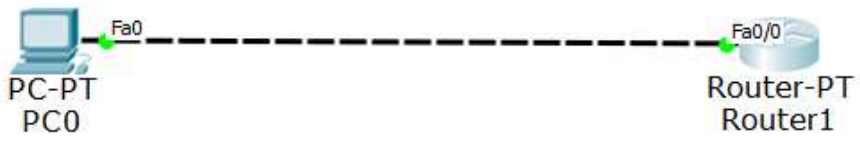
VLAN	Mac Address	Type	Ports
1	0001.6823.6dd2	Dynamic	Fa 0/1
1	0003.e49d.b2d9	Dynamic	Fa1/1
1	0004.9a42.616c	Dynamic	Fa 2/1
1	0005.6e6a.7da2	Dynamic	Eth6/1
1	0003.f286.7a19	Dynamic	Fa 3/1

Experiment 10:

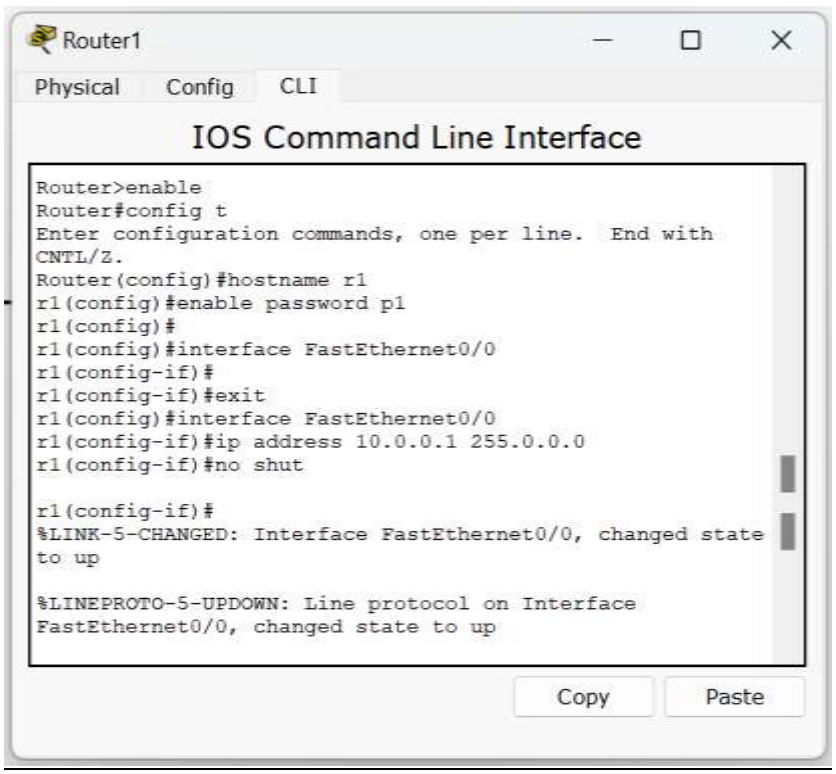
To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.

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#
```

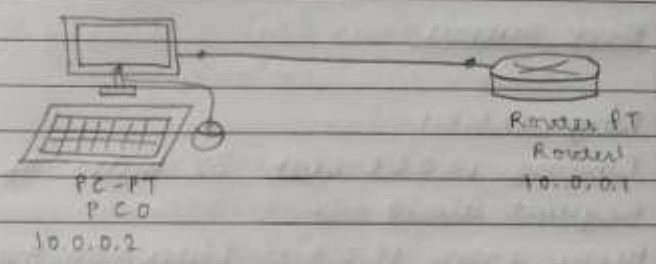
Observation:

Experiment 12

Date / /
Page 41

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

Topology



Procedure

- Create a topology as shown
- Configure IP address and
- Configure router

Enable
Config T
Hostname r1
Enable secret p1
Interface fastethernet 0/0
IP address fastethernet ~~0/0~~ 10.0.0.1 255.0.0.0
No shut
line vty 0 5
login
password p0
Exit
Exit
wr

Output

Password for user access verification is p0
Password for enable is p1
Accessing router CLI from PC
Show ip route

Ping Output

Ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data
Request timed out
Reply from 10.0.0.1 bytes = 32, time = 0 TTL=255
Reply from 10.0.0.1 bytes = 32, time = 0, TTL=255
Reply from 10.0.0.1 bytes = 32, time = 0, TTL=255

Ping Statistics for 10.0.0.1
Typing 10.0.0.1 wpm
User access verification
Password p0
p1 > enable
Password p1
r1 # show ip route
C 10.0.0.0 /8 is directly connected
Fast ethernet 0/0

Observation

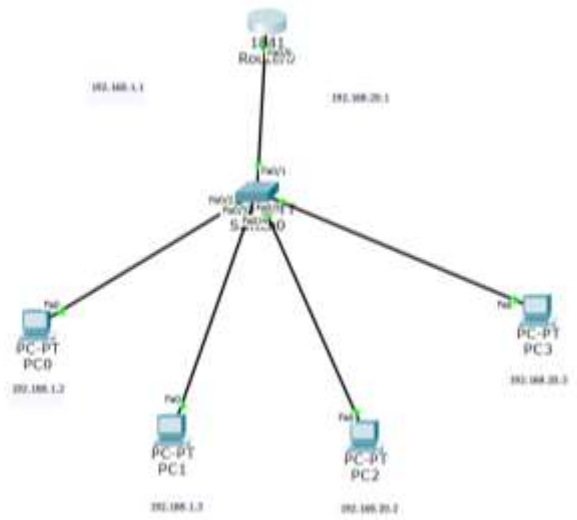
- TELNET is a type of protocol which enables one computer to connect to local computer
- Its used as a std TCP/IP protocol for virtual terminal services provided by ISP

Experiment 11:

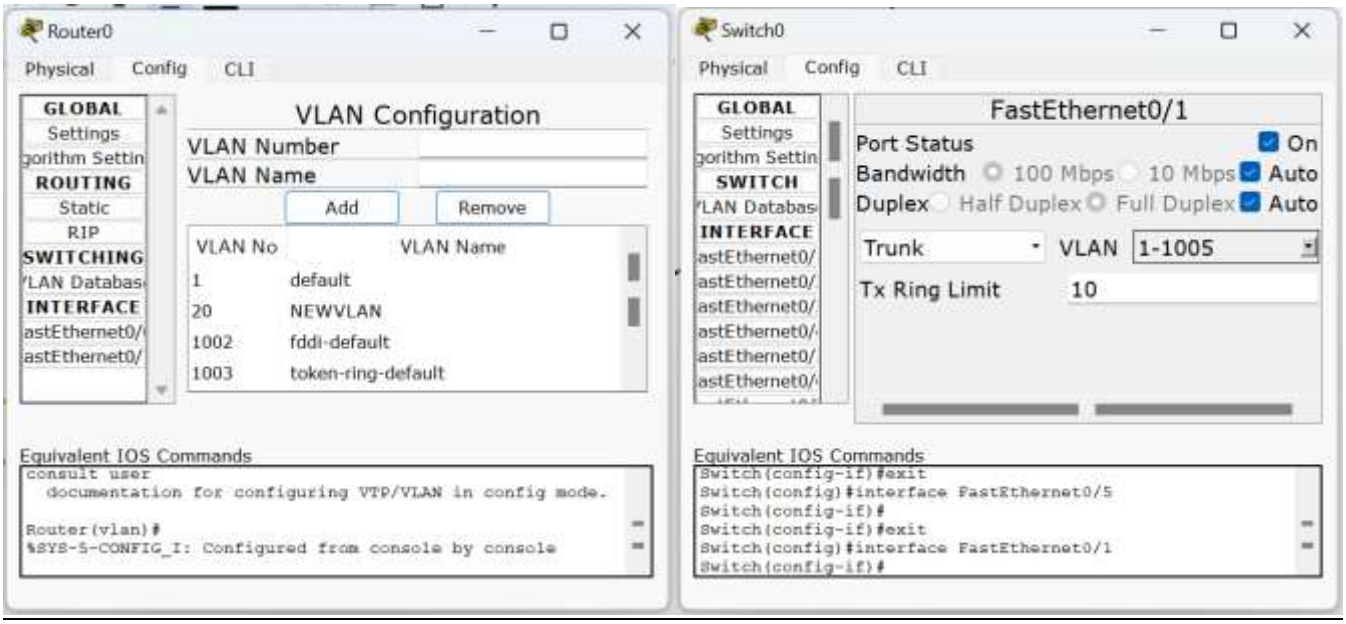
To construct a VLAN and make the PC's communicate among a VLAN

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

```
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>
```

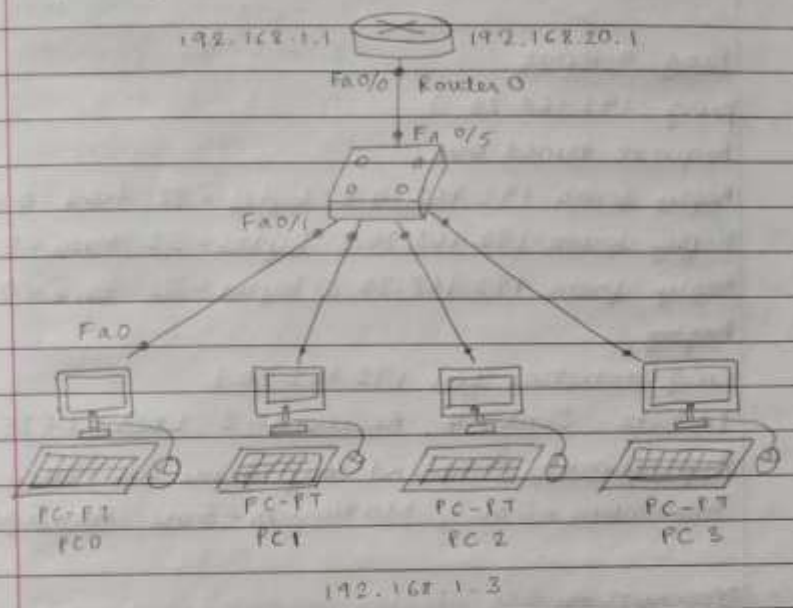
Observation:

Date / /
Page 35

Experiment 9

To construct a VLAN, make a PC communicate among VLAN

Topology



Procedure

- Create a topology as shown above by choosing 1841 router and 2960-24 TT switch
- Set IP of router and PC successfully class C type addresses are used
- In switch, go to config tab, select VLAN database. give a VLAN no. and name
- In config tab of router, type the following

config T
interface fa 0/0
IP address 192.168.1.1 255.255.255.0
No shut
Exit
config T
interface fa 0/0.1

encapsulation dot1q2
IP address 192.168.20.1 255.255.255.0
No Shut
Exit

Ping output
ping 192.168.20.3
Request timed out
Reply from 192.168.20.3: bytes = 32 time = 0 ms TTL 127
Reply from 192.168.20.3: bytes = 32 time = 5ms TTL 127
Reply from 192.168.20.3: bytes = 32 time = 0 ms TTL 127
~~Reply~~
Ping statistics for 192.168.20.3
Packets Sent = 4, Received = 3, Lost = 1 (25% Lm
Approximate round trip time in ms
Minimum = 0 ms, Maximum = 5 ms, Average = 1m

Observation

- We can have one device on one LAN and another on VLAN connected to same switch only to bear broadcast traffic
- VLAN's used subnets / class C address
- Inter VLAN routing gives flexible tool to divide networks which have o Potential to embrace security & performance.

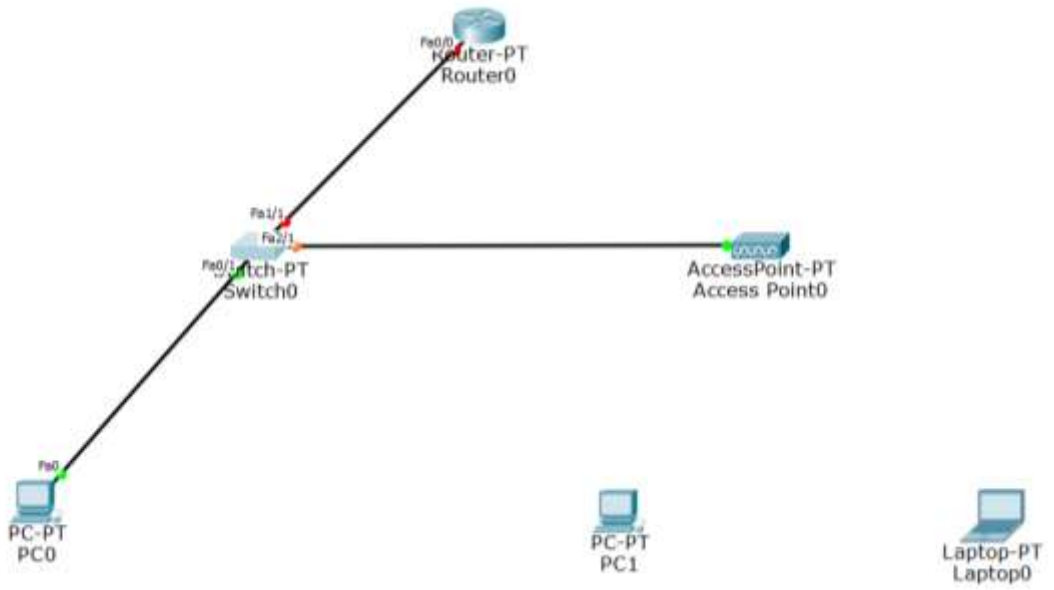
Experiment 12:

To construct a WLAN and make the nodes communicate wirelessly

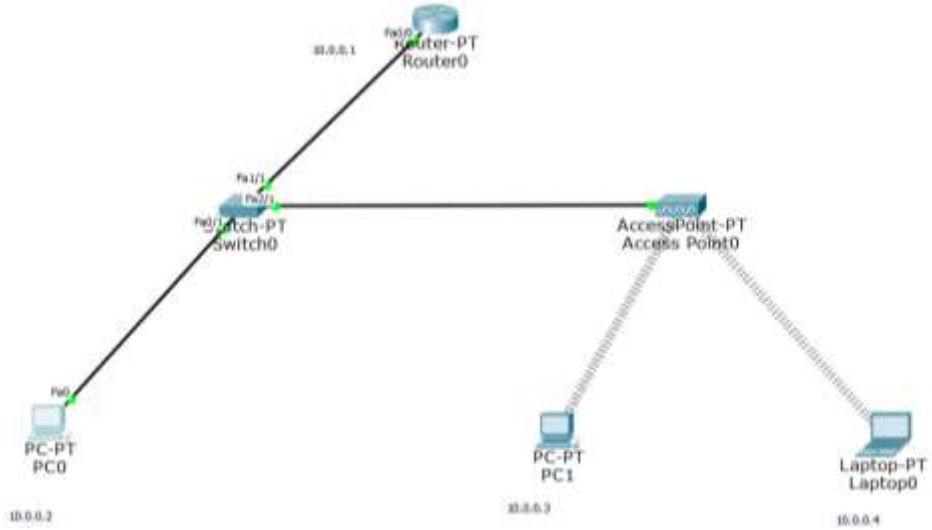
Aim: To construct a WLAN and make the nodes communicate wirelessly.

Topology:

(Initial)

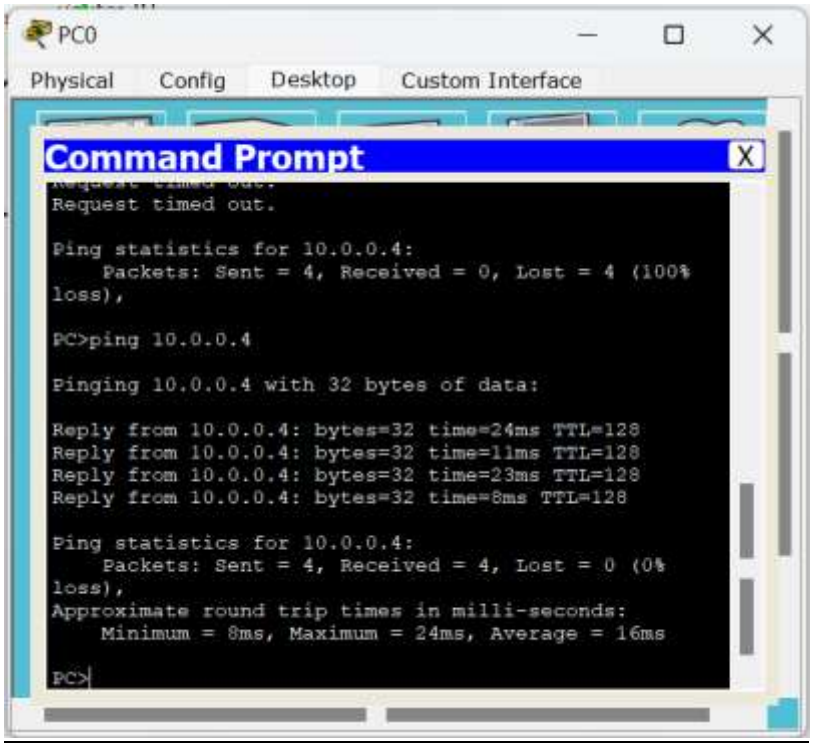


(Final)

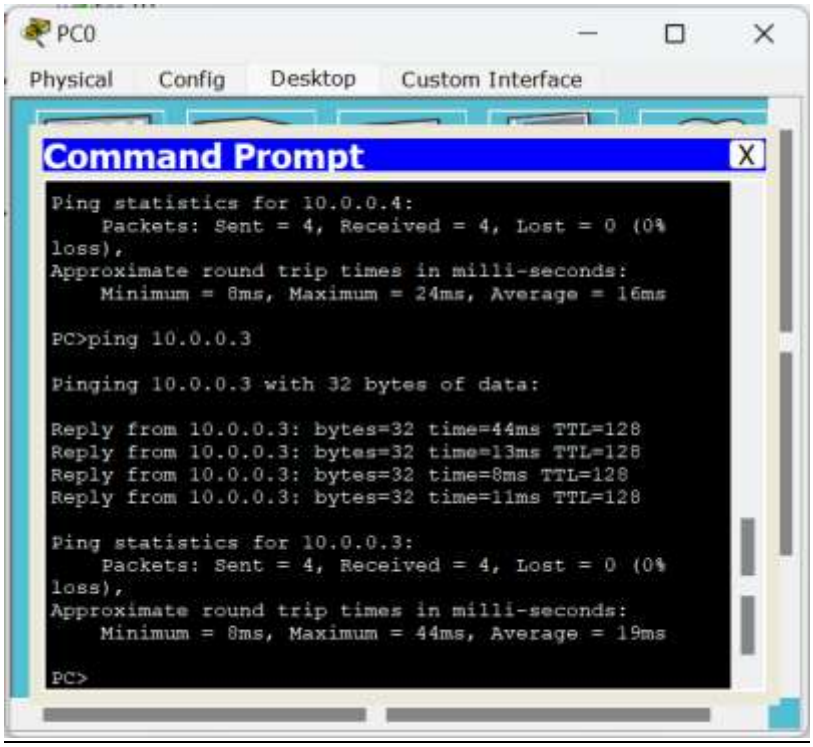


Output:

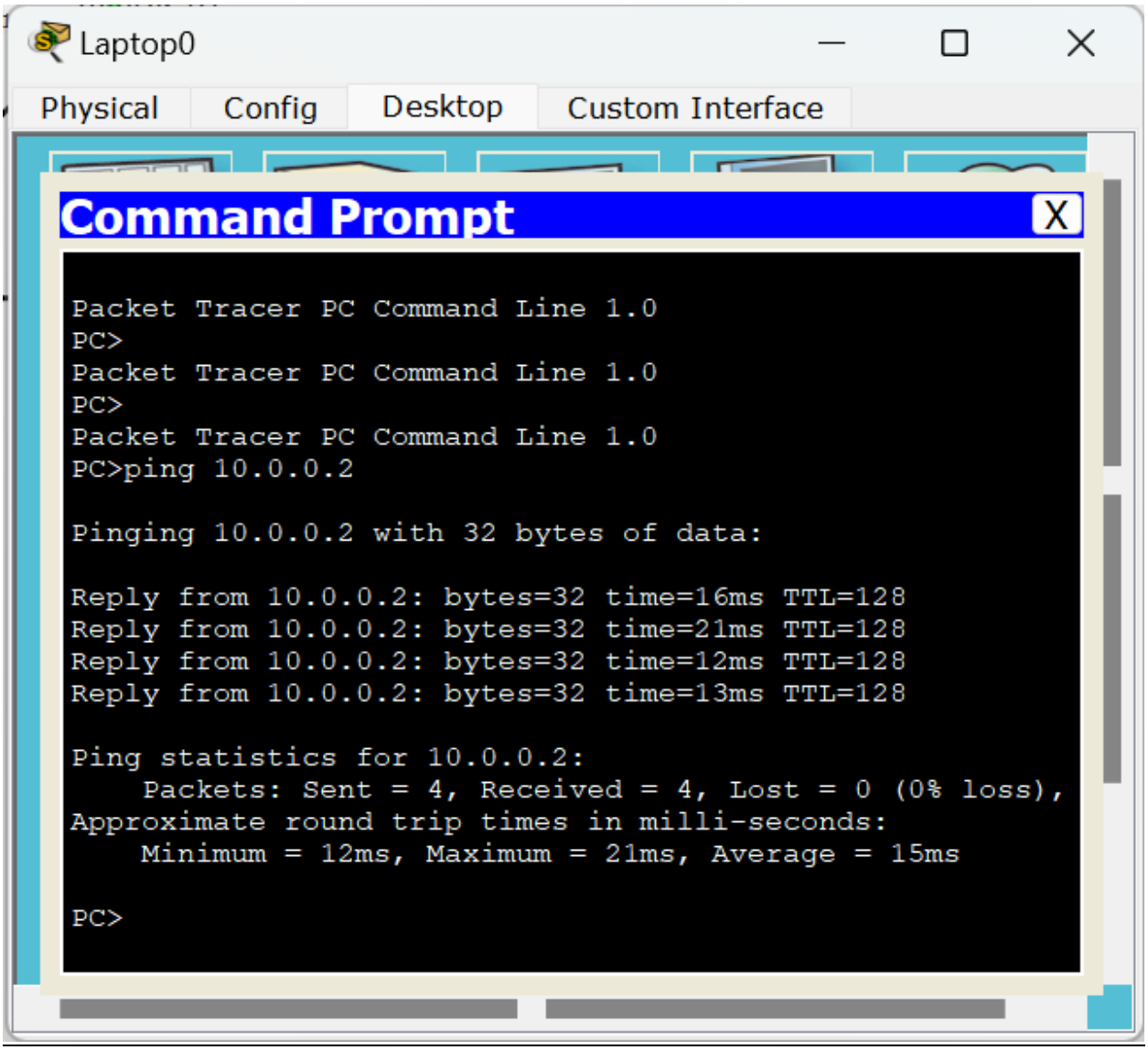
PC0 TO LAPTOP



PC0 TO PC1



LAPTOP TO PC0



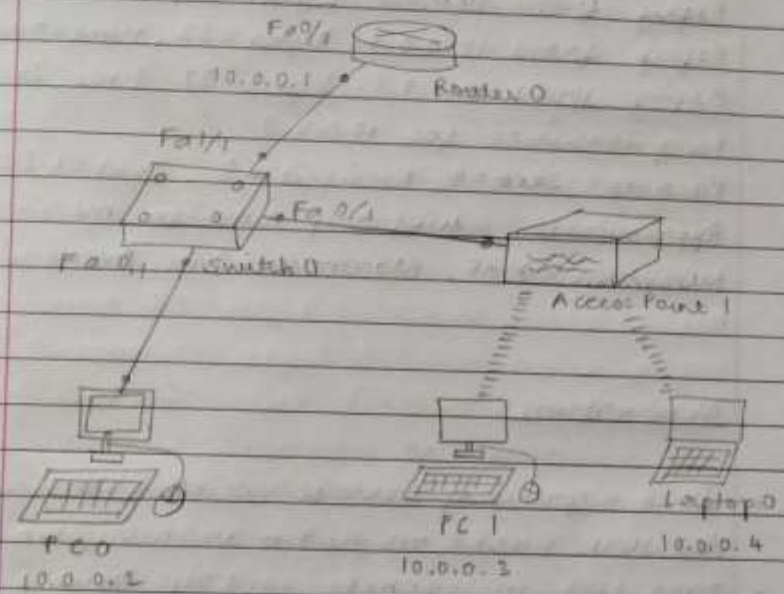
Observation:

Experiment 11

Date / /
Page 39

To construct a WLAN, make the nodes communicate wirelessly.

Topology:



Procedure

- Construct the above topology configure the PC 0 and Router 0 as usual
- Configure access point 1 port SSID
- Select WEP and provide any 10 digit hex key - 1234567890
- Configure PC 1 and Laptop with wireless standard
- Switch off device, drag existing PC-HOST-NM-LAN to component listed in LHS.
- Ping from every device to others

Output

ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of Data
Request timed out
Reply from 10.0.0.3 : bytes = 32, time = 0 ms, TTL=128
Reply from 10.0.0.3: bytes = 32, time = 0 ms, TTL=128
Reply from 10.0.0.3: bytes = 32, time 2ms, TTL=128
Ping statistics for 10.0.0.3
Packets: sent = 4, Received = 3, Lost = 1 (25% Loss)
Approximate round trip time in ms
Minimum = 0 ms, Maximum = 1ms, Average = 0 ms

Observation

- It is a group of devices which forms a network based on Radio transmission
- Data sent in packets contain layers with label instructions MAC Addresses
- Access Point in base station which serves as hub to other stations
- We can connect to multiple devices wirelessly to transmit data.

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, "10001000000100001"); // 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:

Date 17/8/23
Page 43

CYCLE - 2

Experiment 13

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

Code:

```
#include <stdio.h>
#include <string.h>
void binaryXOR(char* result, const const* a,
        const char* b)
{
    for(int i=0; i<16; i++)
        result[i]=(a[i]==b[i])?'0':'1';
    result[16]='\0';
}
void calculateCRC(const char*data, int len,
        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(j=0, j<8, j++)
            char msb=crc[0];
            for(int k=0; k<16; k++)
                crc[k]=crc[k+1];
            crc[16]='0';
            if(msb=='1')
                char temp[17];
            binaryXOR(temp, crc, '10001000001
                        100001');
            strcpy(crc, temp);
```

```
        crc[15] = (data[i] == '1') ? '1' : '0';
    }
    strcpy (checksum, crc);
}
int main ()
{
    char data[100];
    printf (" Enter data in binary");
    scanf ("%s", data);
    int dlen = strlen(data);
    char checksum[17];
    calculateCRC ( data, dlen, checksum);
    printf ("Calculated CRC %s\n", checksum);

    char received[17];
    printf (" Enter received CRC");
    scanf ("%s", received );

    if (strcmp (received, checksum) == 0)
        printf (" Data is error free\n");
    else
        printf (" Data contains errors");

    return 0;
}
```

Output

→ Enter Data in binary: 10001
Calculated CRC: 0111001001000001
Enter received CRC: 1011100100111100
Data Contains errors.

→ Enter Data in binary:
Calculated CRC: 0111001001000001
Enter received CRC: 0111001001000001
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:

Date 17/8/23
Page 46

Experiment 14

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

Code:

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
# define c 50

void main()
{
    int ~~maxlim~~=10;
    int bnckCap = 0, opRate = 5;
    while ( lim <=20)
    {   int Packet,
        printf ("\n Enter new Packet Size");
        scanf ("%d", &Packet);
    if (Packet < c)
    {   buckCap+= Packet,
        printf ("bucket Capacity currently");
        buckCap -= OpRate;
        printf ("bucket Capacity after output: %d
                    buckCap);
        timlim ++;
    }
    else if ( Packet > c || (Packet + buckCap)>c)
    {
        printf ("\n New Packet can't be added
                to bucket");
        buckCap - = OpRate,
        printf ("\n Bucket capacity after
            output : %d", buckCap);
    }
```

```
    else if( buckcap<0)
    {
        buckcap=0;
        printf ("\n Bucket capacity after output:
                %d ", buckcap);
        timelimit++;
        exit(0);
    }
  }
}
```

Output

Enter new packet size : 4000

Enter outgoing rate : 200

Enter packet size : 3000

Packet size 3000 is added and in bucket

Enter 1 to continue or 0 to stop : 1

Enter packet size : 2000

Packet size 2000 is added and in bucket

Enter 1 to continue and 0 to stop : 0

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:

Date 27/8/23
Page 48

Experiment 15

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

Code:

Client.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")
print (filecontents)
clientSocket.close()
```

Server.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 ("server is ready to receive ")
    connSock, addr = serversocket.accept()
    sentence = connSock.recv(1024).decode()
    fileOpen(sentence, "r")
```

```
        l = file.read(1024)
        connsock.send(l.encode())
        print("\n Send contents of" + sentence)
        file.close()
        connectionSocket.close()
```

Output

Servertcp.py :-
The server is ready to receive

Send contents of servertcp.py
The server is ready to receive

Clienttcp.py
Enter file name servertcp.py

From server

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket.bind((ServerName, ServerPort))
serverSocket.listen(1)
while(1)
    print("Server is ready to receive")
    conSock, addr = serversocket.accept()
    file = open(sentence, "r")
    print("Send contents of" + sentence)
    file.close
    connectionSocket.close()
```

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(&quot;utf-8&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:

Date / /
Page 50

Experiment 16

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

Code:

Client.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket ((Name, Port))
sentence = input ("Enter file name:")
clientSocket.sendto (bytes( sentence, "utf-8"),
        (serverName, Port))
filecontents, server = clientAd . recvfrom (2048)
print ("Reply from Server")
print ( filecontents.decode ("utf-8"))
clientSocket.close()
client socket close()
```

ServerUDP.py

```
from socket import *
SName = "127.0.0.1"
SPort = 12000
SSock bind ("127.0.0.1", sPort ))
print ("The server is ready to receive"
while (1):
    sen, clientaddr = serverSock . recvfrom (2048)
    sen = sen decode ("utf-8")
    file = open (sen, "r")
    con = file . read (2048)
    serverSock . sendto (bytes (con, "utf-8"),
```

```
clientAddr )
print ('\n Sent contents of ', end = ' ')
print (serv)
file.close()
```

Output

ServerUDP.py
The server is ready to receive

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

ClientUDP.py
Enter file name : serverUDP.py

Reply from Server

" Contents of 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()
```

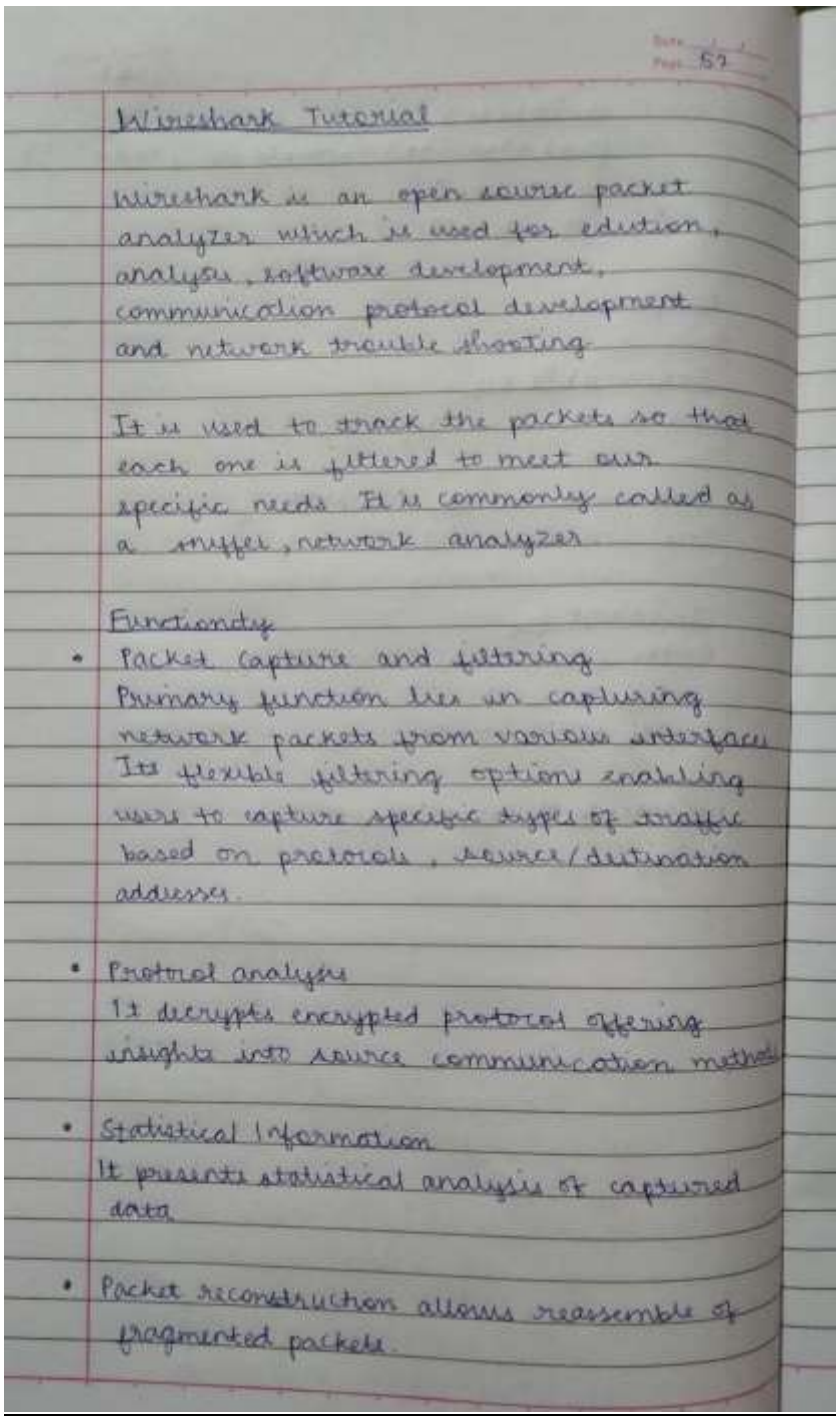
```
The server is ready to receive

Sent contents of Server.udp.py
```

Experiment 17:

Tool Exploration -Wireshark

Aim: Tool Exploration -Wireshark

Observation:



Wireshark Tutorial

Wireshark is an open source packet analyzer which is used for education, analysis, software development, communication protocol development and network trouble shooting

It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network analyzer

Functionality

- Packet Capture and filtering
  Primary function lies in capturing network packets from various interfaces. Its flexible filtering options enabling users to capture specific types of traffic based on protocols, source/destination addresses.

- Protocol analysis
  It decrypts encrypted protocol offering insights into source communication methods

- Statistical Information
  It presents statistical analysis of captured data

- Packet reconstruction allows reassemble of fragmented packets.

Procedure

1) In the First window, select ethernet
2) Filter TCP or any require protocol
3) Click on it, new window opens
4) Dropdown: Transmission control Protocol
   Sor Port: 62 148, Dst Port: 443,
   Seq: 2 Ack: 65 Len - 0
5) This is available in the previous window in the left split of screen.
6) Clicking on dropdown of it, clicking on any of the counterpart in right split side of screen.
7) In command prompt, type > ip config

Result

Windows IP configuration
Ethernet adapter Ethernet
Connection - specific DNS suffix:
Link - local IPv8 address . . Fe80 :: be78 :
f609 ; ed25 : e3291B
IPv4 address . . . . 10.129 . 2.83
Subnet Mast 255.255.0.0
Default Gateway . . . 10 . 129.0.11