VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

MEGHA SURESHA (1BM21CS262)

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 MEGHA SURESHA (1BM21CS262)**,** 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. Sowmya T
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Index

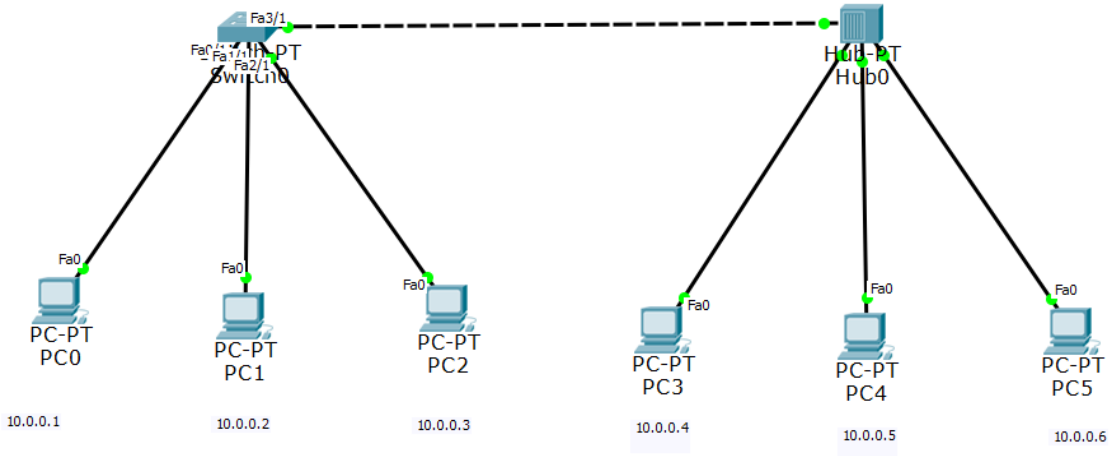
CYCLE 1:

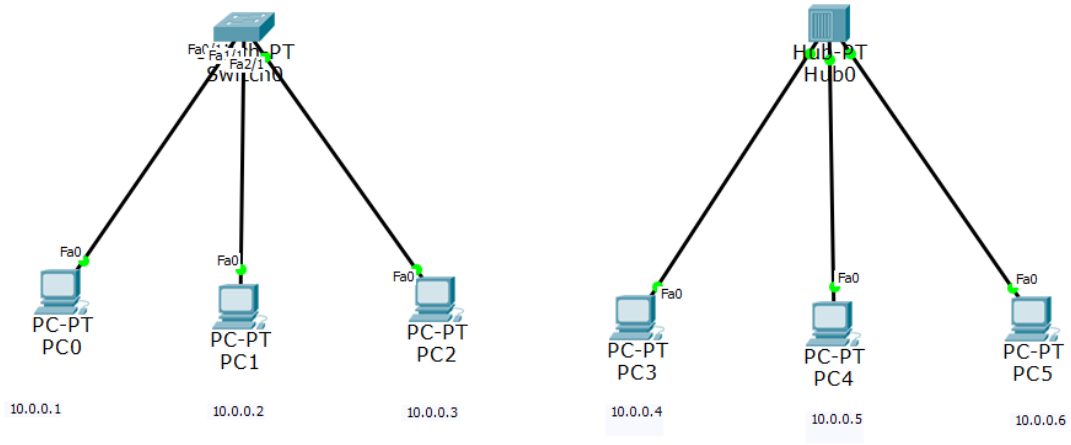
Experiment 1:

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

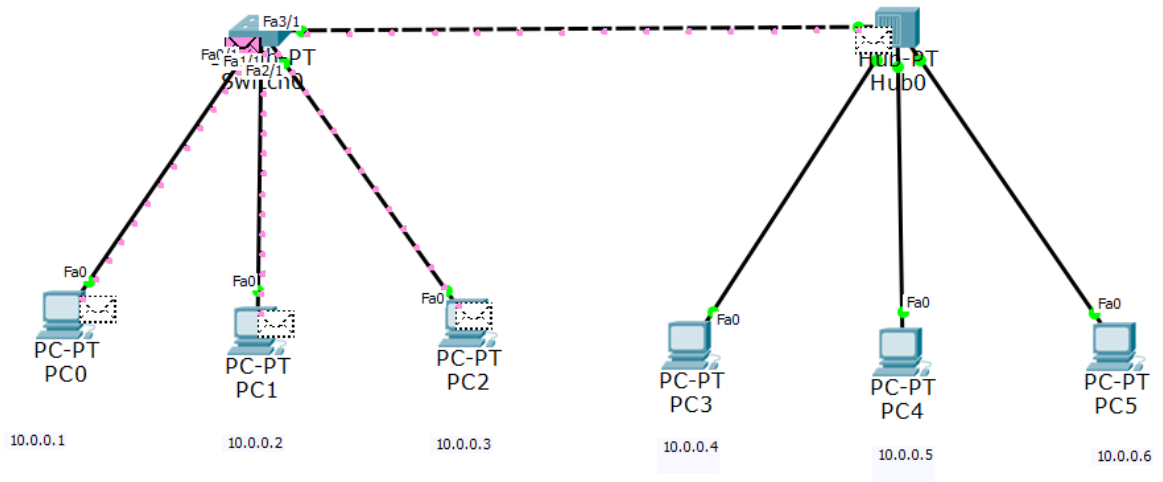
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message

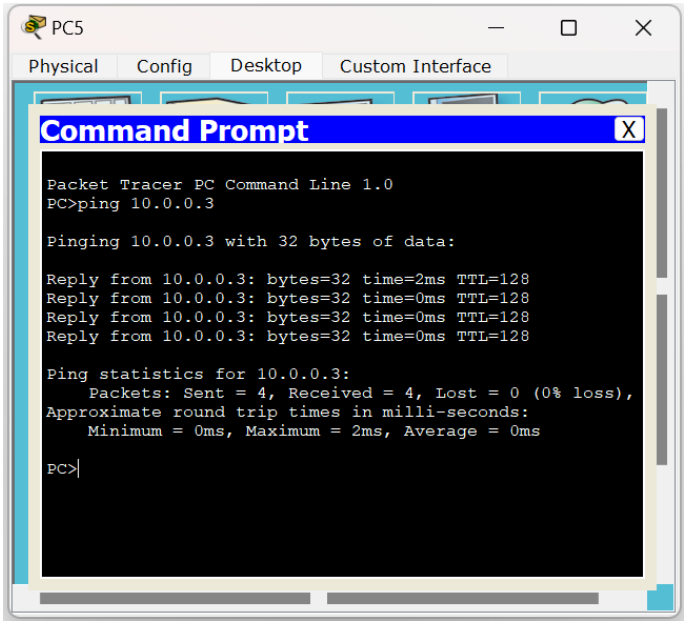
Topology:

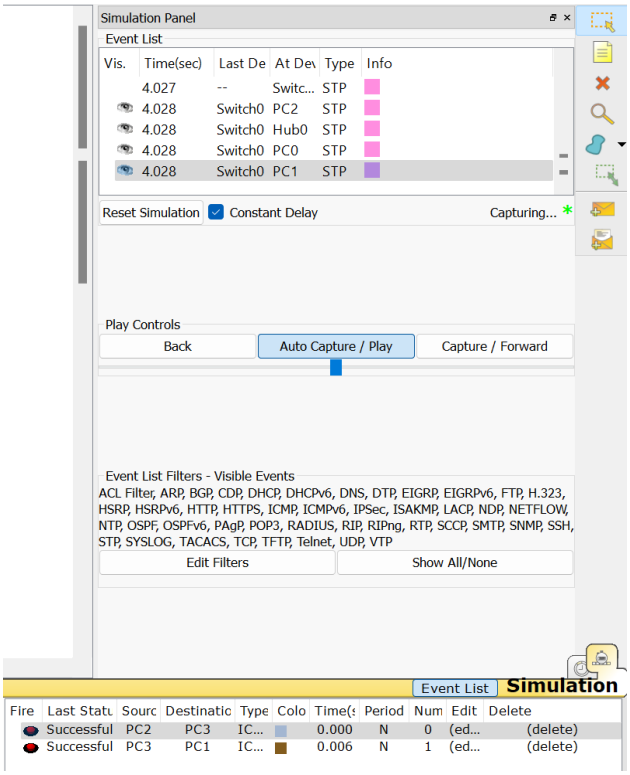




Output:





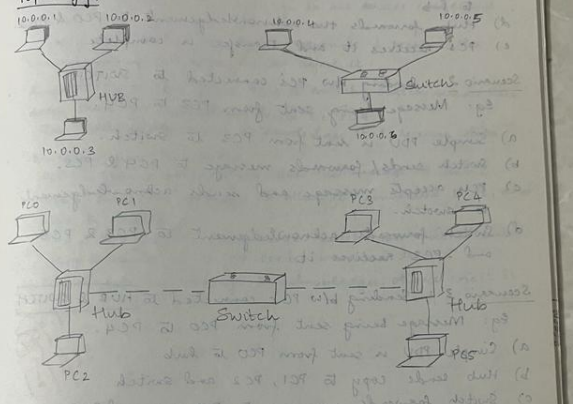


Observation:

EXPERIMENT-1

Aim: Create a topology and simulate sending a simple PDU from source to destination using SIMPLE HUB and SWITCH as connecting domains.

Topology:



Procedure:

1. Add generic hub and connect 3 PC's to it using copper straight through wire.
2. Add a generic switch and connect 3 PC's to it using copper straight through wire.
3. Wait for switch-PC connections to be established.
4. Connect hub & switch using a copper cross-over wire.
5. Set IP addresses for all PC's (end devices) as 10.0.0.1, 10.0.0.2, 10.0.0.3 .. etc.
6. Send simple PDU from one PC to another and wait for simulation to run after clicking auto-capture.

RESULT:

Scenario 1: Sending b/w PC's connected to HUB

Eg: Message being sent from PC1 to PC2.

a) Simple PDU is sent from PC1 to hub.
b) Hub sends copies of the message to PC0 and PC2.
c) PC2 receives message & sends acknowledgement to hub.
d) Hub forwards the acknowledgement to PC0 & PC1.
e) PC1 receives it and transfer is complete.

Scenario 2: Sending b/w PC's connected to SWITCH

Eg: Message being sent from PC3 to PC4.

a) Simple PDU is sent from PC3 to switch.
b) Switch sends/forwards message to PC4 & PC5.
c) PC4 accepts message and sends acknowledgement to switch.
d) Switch forwards acknowledgement to PC3 & PC5 and PC3 receives it.

Scenario 3: Sending b/w PC's connected to HUB & SWITCH

Eg: Message being sent from PC0 to PC4.

a) Simple PDU is sent from PC0 to hub.
b) Hub sends copy to PC1, PC2 and switch.
c) Switch forwards message to PC3, PC4 and PC5.
d) PC4 receives message and sends an acknowledgement to the switch.
e) Switch forwards acknowledgement to PC3, PC5 & hub.
f) Hub sends copy of acknowledgement to PC0, PC1, PC2.
g) PC0 receives acknowledgement, thereby completing transfer of message.

```
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
"
"
"

Ping statistics for 10.0.0.3:
    Packets: sent=4, received=4, lost=0 (0% loss)
Approximate round trip times in milliseconds:
    Minimum=0ms, Maximum=0ms, Average=0ms
```

Result:

a) Hub
~~Simple~~ PDU is sent from ~~PC1 to hub~~.
Hub

Observation:

- In simulation mode, message was sent from source to hub, the hub forwards the message to all devices connected with it.
  Only the destination device responds & receives the message whereas all other devices reject it.
- Simple PDU is sent from PC3 to switch. Switch sends this data only to destination device i.e PC4.
  The destination device acknowledges it & receives data.
- Message from PC0 → PC4
  PC0 is source node which sent the message to hub 1.
  Hub 1 sends the message to other PC's connected to it, and also to switch.
  PC1, PC2 rejects the message. Switch forwards to hub 2.
  Hub 2 sends it to all devices connected to it.
  PC4 accepts the message & acknowledges, others reject it.

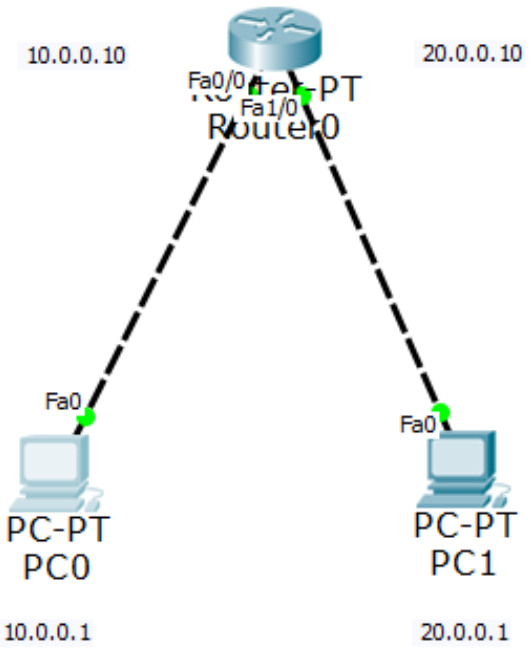
Sgd. T
16/6/23

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:

```
Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

After Configuring Gateway:

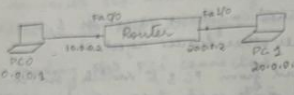
```
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

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

Observation:

EXPERIMENT-2

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

Topology 1:



Procedure:

1. Use generic router and connect 2 end devices to it.
2. Set IP addresses of the PC's as 10.0.0.1 and 20.0.0.2 respectively.
3. In the router → Command line Interface:
   Enter 'no' for continue with configuration dialogue
   Type 'enable'
   Type 'configure terminal'
   router config # interface Fa 0/0
   router (config-if) # ip address 10.0.0.2 255.0.0.0
   router (config-if) # no shutdown
   router (config-if) # exit
4. Repeat step 3 for Fa 1/0 interface.
5. Ping PC 1 from PC0.
6. Set output gateway for PC0 as 10.0.0.2 and 20.0.0.2 for PC1
7. Ping PC1 again from PC0.

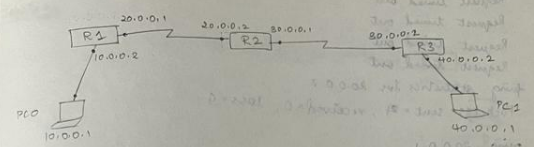
Result:

Before setting default gateway
ping 20.0.0.1
Request timed out
Request timed out
Request timed out
Request timed out
Ping statistics for 20.0.0.2
Packets: Sent = 4, Received = 0, Lost = 4

After setting default gateway
ping 20.0.0.1
Reply from 20.0.0.1 time = 0ms TTL = 128
Reply from " " "
" "
Ping statistics for 10.0.0.2
Packets: Sent = 4, Received = 4, Lost = 0

Three Routers + End devices

Topology 2:



Procedure:

1. Connect 3 generic routers R1, R2 & R3 and 2 end devices.
2. The end devices PC0 connected to R1 and PC1 to R3. R2 is connected to R1 and R3, acting as an intermediate.
3. Set IP addresses of each end device, each device is considered to be a separate network.
4. In CLI of R1, configure the IP address of Fast Ethernet port (Fa 0/0) as 10.0.0.2 by the steps mentioned previously.
5. Similarly, router-router connection is done using serial port. For R1, configure the IP address of se 2/0 as 20.0.0.1.
6. Repeat above 2 steps for R2 and R3. Following are the required IP addresses:

Router 1	Fa 0/0 → 10.0.0.2
	Se 2/0 → 20.0.0.1
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

7. In PC0 and PC1, set IP address of the output gateway as 10.0.0.2 and 40.0.0.2 respectively.
8. Now, ping from PC0 to PC1: PC0 to router 2; PC0 to router 1.
   i.e. IP address 10.0.0.1 to 20.0.0.1
   IP address 10.0.0.1 to 20.0.0.2
   IP address 10.0.0.1 to 30.0.0.1
   IP address 10.0.0.1 to 40.0.0.1
9. Note down the ping responses.

S.P.T.

Result:

i) ping 20.0.0.2

Request timed out
Request timed out
Request timed out
Request timed out

ping statistics for 20.0.0.2
Packets: sent = 4, received = 0, loss = 4

ii) ping 20.0.0.1

Reply from 20.0.0.1 time = 0ms TTL = 128
Reply from 20.0.0.1 " "
Reply from 20.0.0.1 " "
Reply from 20.0.0.1 " "

iii) ping 30.0.0.1

Reply from 30.0.0.1 Destination host unreachable
Reply from 30.0.0.1
Reply from 30.0.0.1
Reply from 30.0.0.1

Ping statistics from 30.0.0.1
Packets: sent = 4, received = 0, lost = 4

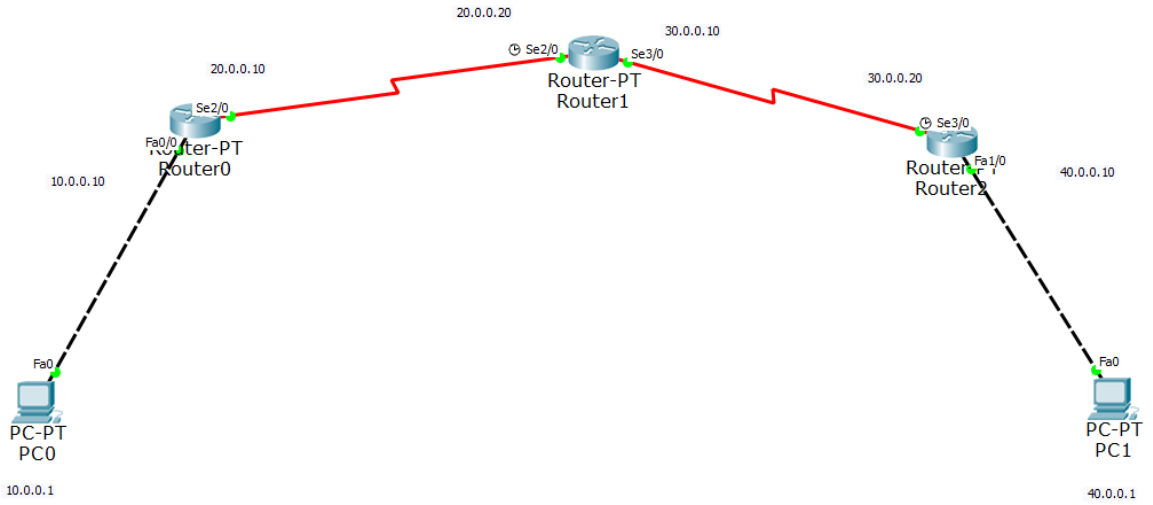
Sri
23/6/23

Experiment 3:

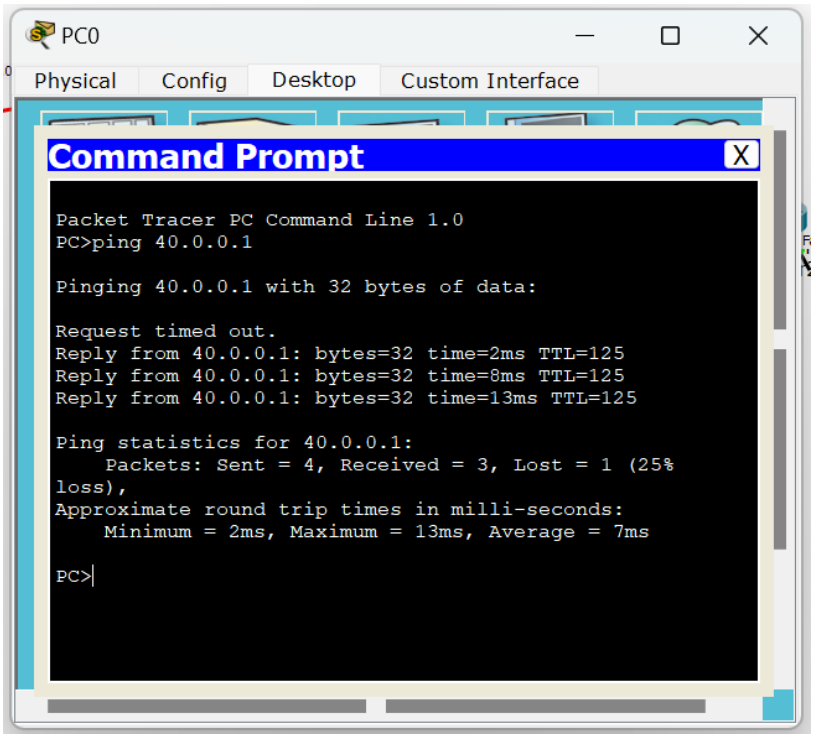
Configure default route, static route to the Router

Aim: Configure default route, static route to the Router

Topology:



Output:

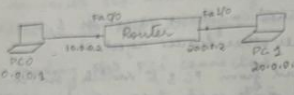


Observation:

EXPERIMENT-2

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

Topology 1:



Procedure:

1. Use generic router and connect 2 end devices to it.
2. Set IP addresses of the PC's as 10.0.0.1 and 20.0.0.2 respectively.
3. In the router → Command line Interface:
   Enter 'no' for continue with configuration dialogue
   Type 'enable'
   Type 'configure terminal'
   router config # interface Fa 0/0
   router (config-if) # ip address 10.0.0.2 255.0.0.0
   router (config-if) # no shutdown
   router (config-if) # exit
4. Repeat step 3 for Fa 1/0 interface.
5. Ping PC 1 from PC0.
6. Set output gateway for PC0 as 10.0.0.2 and 20.0.0.2 for PC1
7. Ping PC1 again from PC0.

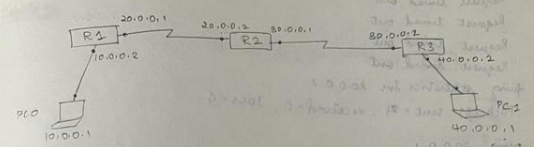
Result:

Before setting default gateway
ping 20.0.0.1
Request timed out
Request timed out
Request timed out
Request timed out
Ping statistics for 20.0.0.2
Packets: Sent = 4, Received = 0, Lost = 4

After setting default gateway
ping 20.0.0.1
Reply from 20.0.0.1 time = 0ms TTL = 128
Reply from " " "
" "
Ping statistics for 10.0.0.2
Packets: Sent = 4, Received = 4, Lost = 0

Three Routers + End devices

Topology 2:



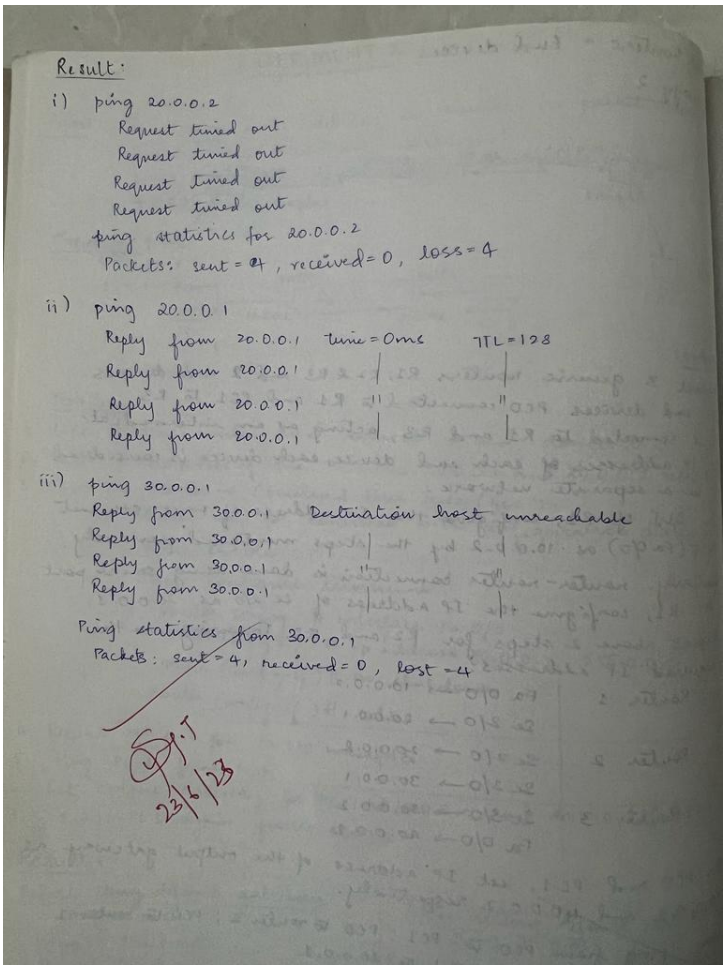
Procedure:

1. Connect 3 generic routers R1, R2 & R3 and 2 end devices.
2. The end devices PC0 connected to R1 and PC1 to R3. R2 is connected to R1 and R3, acting as an intermediate.
3. Set IP addresses of each end device, each device is considered to be a separate network.
4. In CLI of R1, configure the IP address of Fast Ethernet port (Fa 0/0) as 10.0.0.2 by the steps mentioned previously.
5. Similarly, router-router connection is done using serial port. For R1, configure the IP address of se 2/0 as 20.0.0.1.
6. Repeat above 2 steps for R2 and R3. Following are the required IP addresses:

Router 1	Fa 0/0 → 10.0.0.2
	Se 2/0 → 20.0.0.1
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

7. In PC0 and PC1, set IP address of the output gateway as 10.0.0.2 and 40.0.0.2 respectively.
8. Now, ping from PC0 to PC1: PC0 to router 2; PC0 to router 1.
   i.e. IP address 10.0.0.1 to 20.0.0.1
   IP address 10.0.0.1 to 20.0.0.2
   IP address 10.0.0.1 to 30.0.0.1
   IP address 10.0.0.1 to 40.0.0.1
9. Note down the ping responses.

S.P.T.



Result:

i) ping 20.0.0.2
Request timed out
Request timed out
Request timed out
Request timed out
ping statistics for 20.0.0.2
Packets: sent = 4, received = 0, loss = 4

ii) ping 20.0.0.1
Reply from 20.0.0.1 time = 0ms TTL = 128
Reply from 20.0.0.1 " "
Reply from 20.0.0.1 " "
Reply from 20.0.0.1 " "

iii) ping 30.0.0.1
Reply from 30.0.0.1 Destination host unreachable
Reply from 30.0.0.1
Reply from 30.0.0.1
Reply from 30.0.0.1
Ping statistics from 30.0.0.1
Packets: sent = 4, received = 0, lost = 4

23/6/23

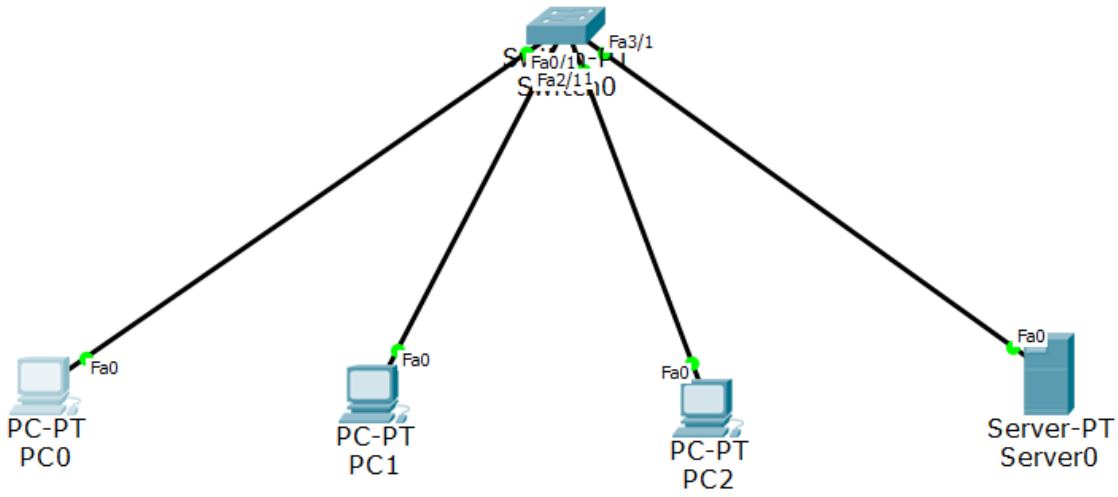
Experiment 4:

Configure DHCP within a LAN and outside LAN.

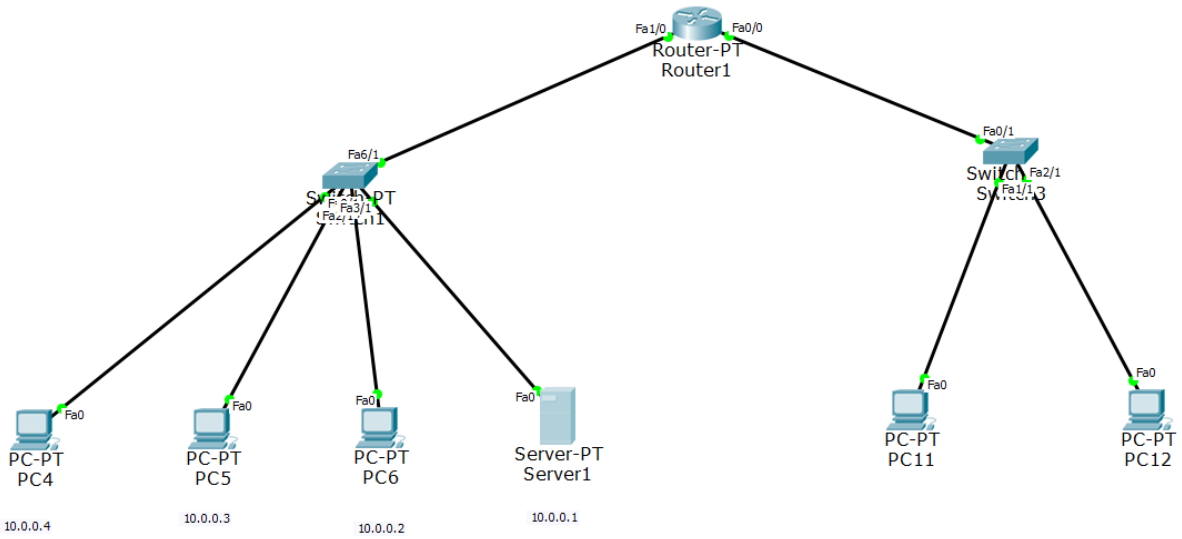
Aim: Configure DHCP within a LAN and outside LAN.

Topology:

(Within a LAN)

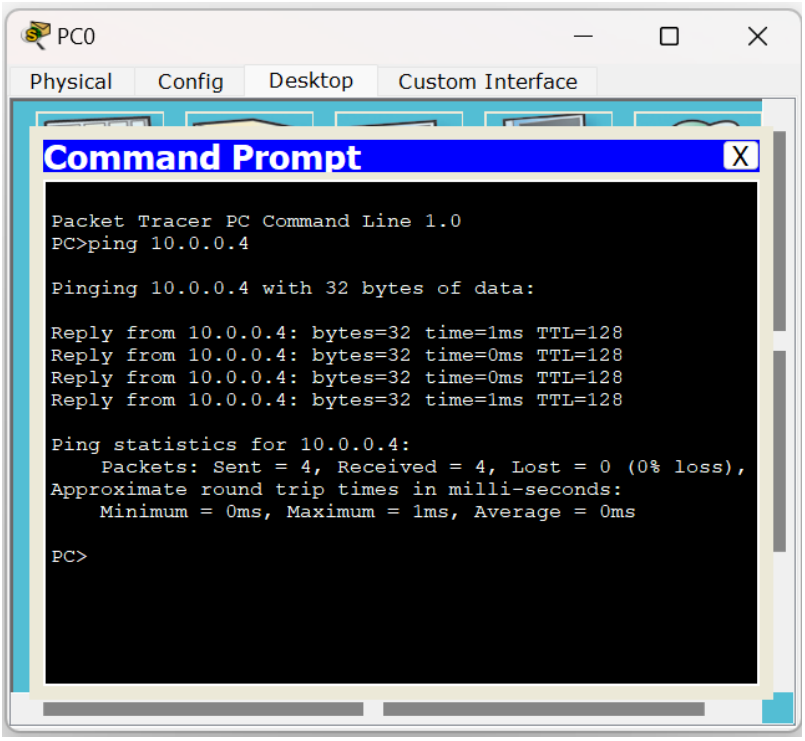


(Outside a LAN)

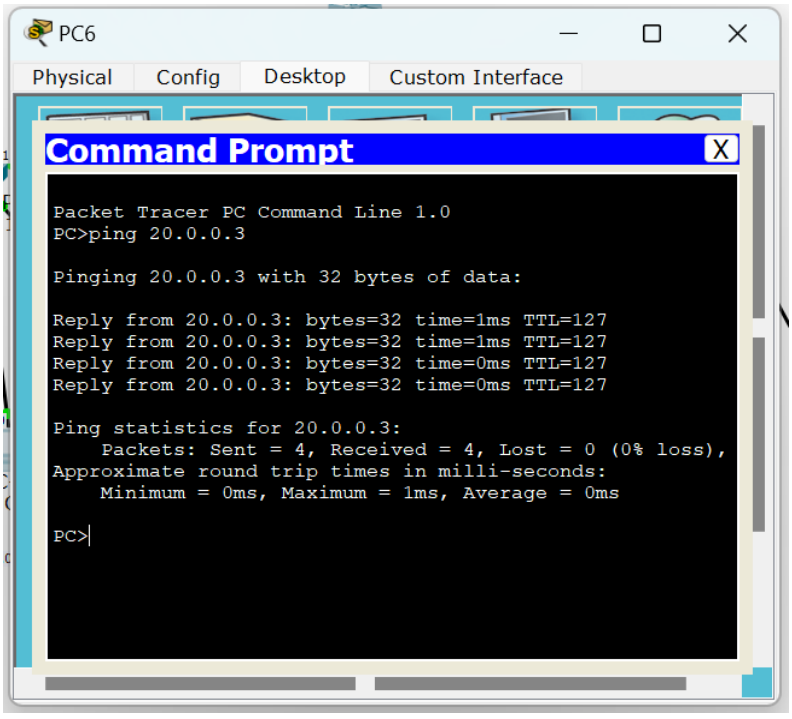


Output:

(Within a LAN)



(Outside a LAN)

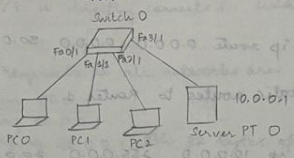


Observation:

EXPERIMENT-4

Aim: Configure DHCP within a LAN and outside LAN

Topology: Within LAN



Procedure:

1. Drag and drop 3 PC's, 1 switch and 1 server.
2. Connect all end devices to switch.
3. Set IP address of saver to say, 10.0.0.1.
4. Click on server services → DHCP.
5. Set start IP address, say 10.0.0.2.
6. Add the configuration.
7. Now click on each PC and go to config and select IP config as DHCP.

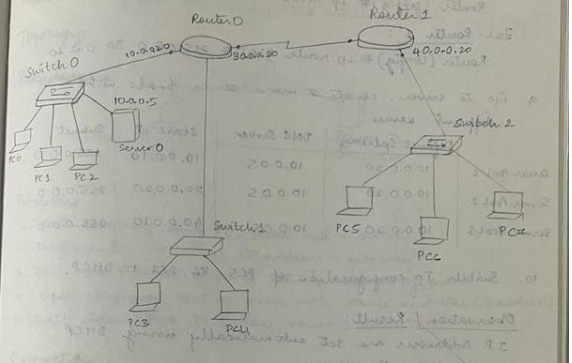
Observation:

1. All the PC's connected to the switch & server get assigned IP addresses starting from 10.0.0.2.
2. PC0 is assigned 10.0.0.2
   PC1 → 10.0.0.3
   PC2 → 10.0.0.4

Result:

All the PC's are assigned IP addresses dynamically/ automatically.

Topology: Outside LAN



Procedure:

1. Repeat procedures done in configuring DHCP within a LAN.
2. Add 2 routers, another set of PC's and switch.
3. Config the ip address of ports Fa4/0 & Fa0/0 of R0.

```
Router > enable
       # config terminal
       # interface Fa4/0
       # ip address 10.0.0.20 255.0.0.0
       # ip helper-address 10.0.0.5
       # no shutdown
       # exit
```

Similarly, do for Fa0/0.

Here, ip helper-address is the address of server0.

4. Switch the ip config in PC5 and PC4 to DHCP.
5. Connect Router 1 to Router 0.
   R1 is connected to Switch 2 and 3 PC's (PC5, PC6, PC7).
6. Config ip address of R1 for interface serial 2/0 and Fa 0/0 with 30.0.0.30 and 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:

   Router (config)# ip route 40.0.0.0 255.0.0.0 20.0.0.30

   For Router 1:

   Router (config)# ip route 10.0.0.0 255.0.0.0 30.0.0.20

9. Go to server, create 2 more server pools with different names.

	Default Gateway	DNS Server	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	40.0.0.10	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.

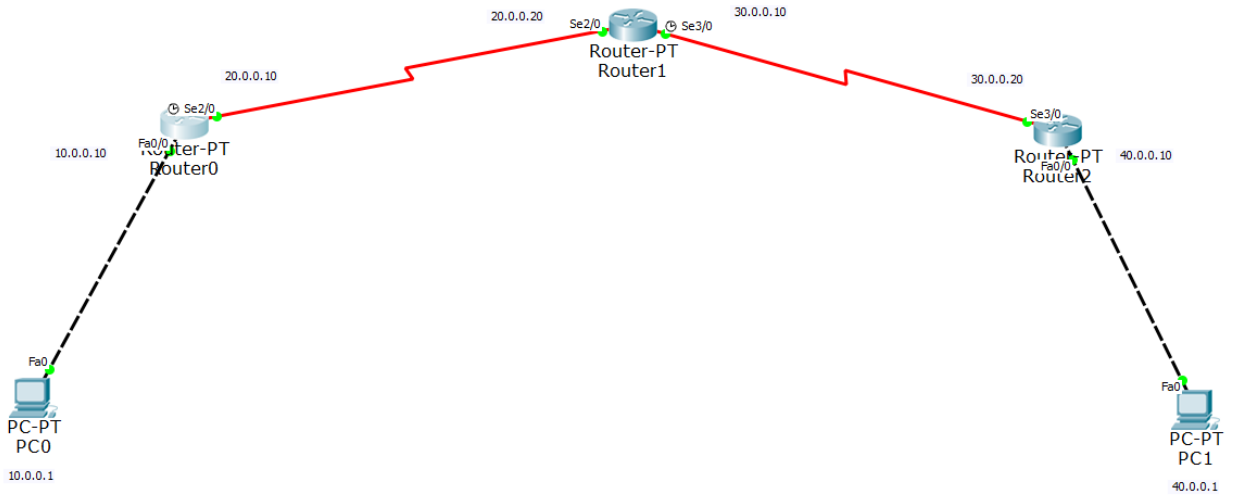
14/12/23

Experiment 5:

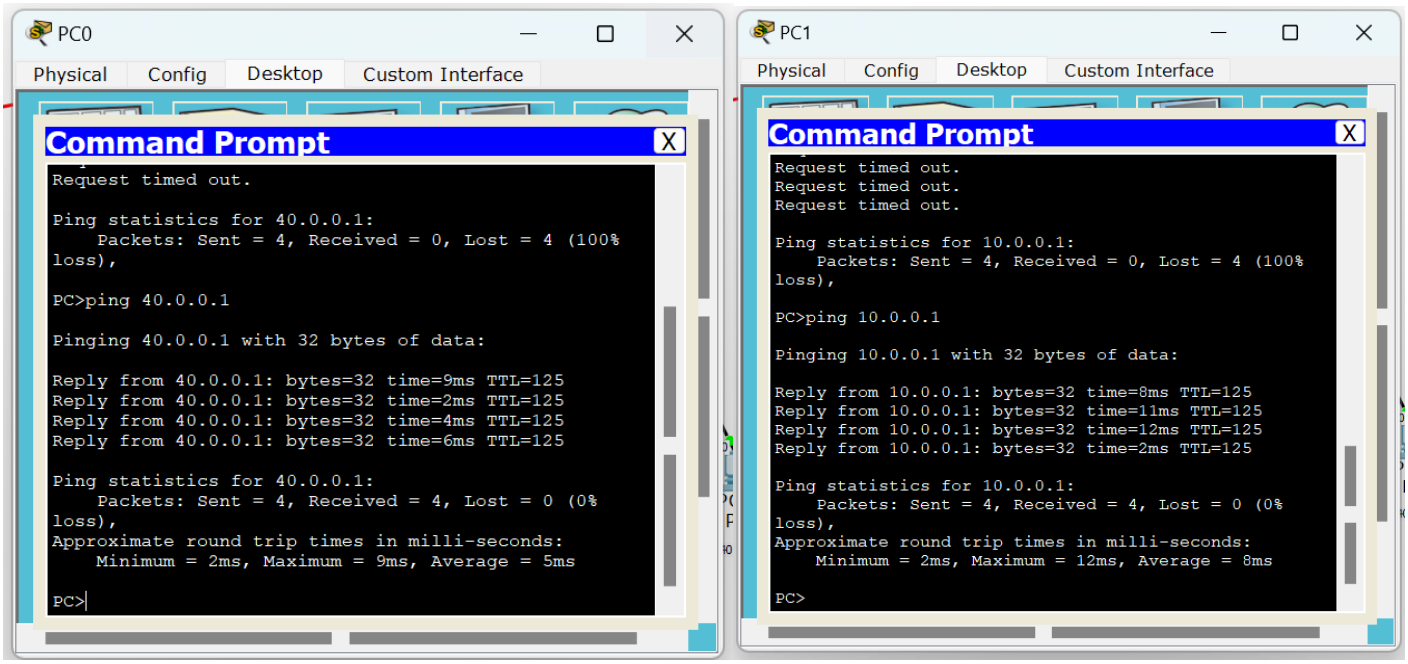
Configure RIP routing Protocol in Routers

Aim: Configure RIP routing Protocol in Routers

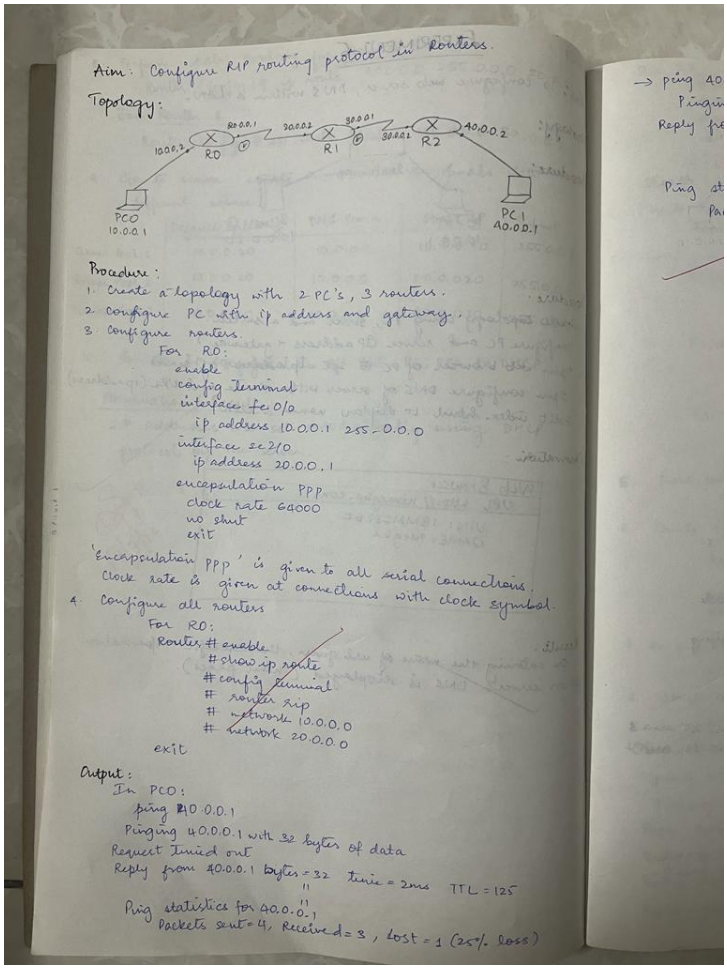
Topology:



Output:



Observation:



Aim: Configure RIP routing protocol in Routers.

Topology:

Procedure:

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

For R0:

```
enable
config terminal
interface fe 0/0
 ip address 10.0.0.1 255.0.0.0
interface se 2/0
 ip address 20.0.0.1
encapsulation ppp
 clock rate 64000
 no shut
 exit
```

'Encapsulation PPP' is given to all serial connections.
Clock rate is given at connections with clock symbol.

4. Configure all routers

For R0:

```
Router# enable
 # show ip route
 # config terminal
 # router rip
 # network 10.0.0.0
 # network 20.0.0.0
exit
```

Output:

In PC0:

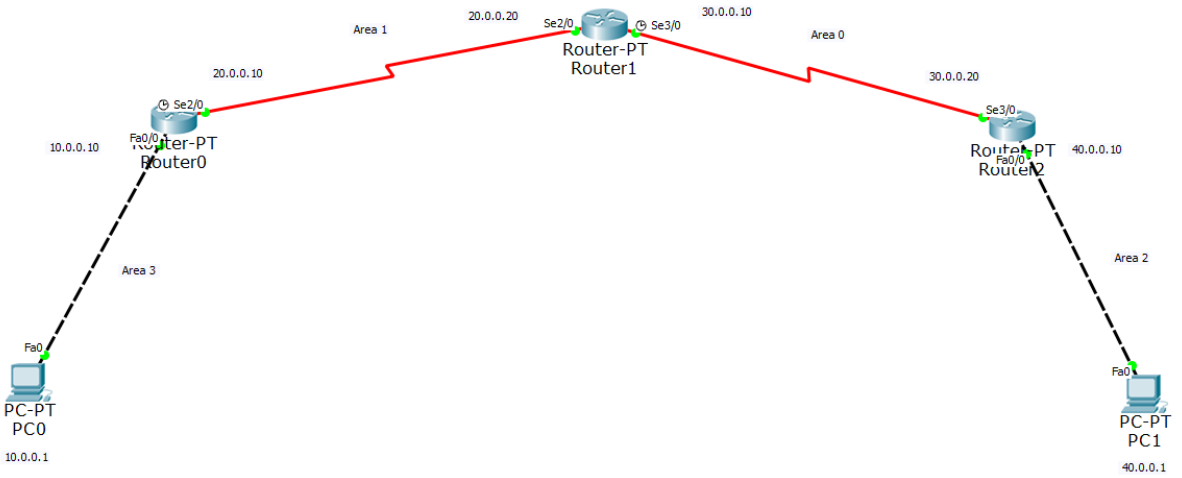
```
ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data
Request Timed out
Reply from 40.0.0.1 bytes = 32 time = 2ms TTL = 125
                  "
Ping statistics for 40.0.0.1
 Packets sent = 4, Received = 3, Lost = 1 (25% loss)
```

Experiment 6:

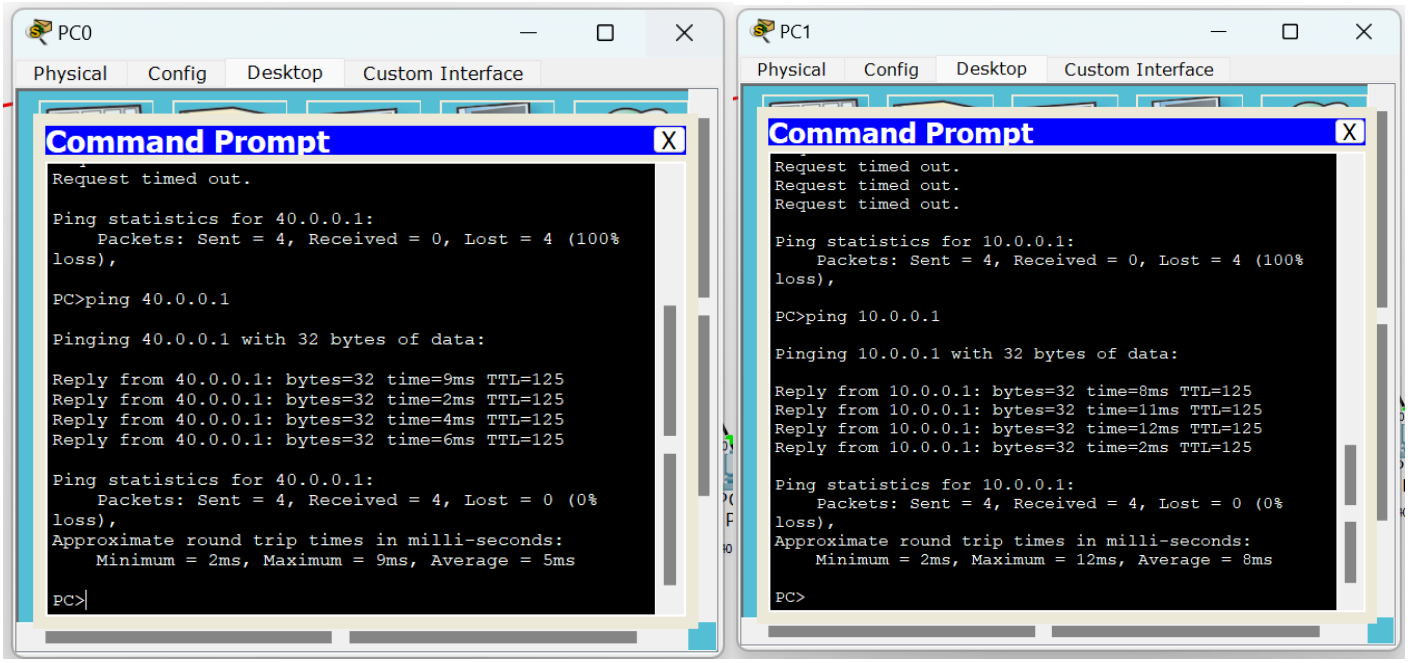
Configure OSPF routing protocol

Aim: Configure OSPF routing protocol

Topology:



Output:



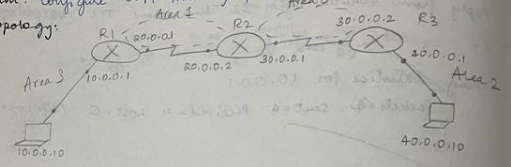
Observation:

11.08.23

EXPERIMENT - 6

Aim: Configure OSPF routing protocol.

Topology:



Procedure:

1. Create topology using 3 routers and 2 PC's.
2. Configure ip address to all interfaces.

In R1,

```
# interface fa/2/0
# ip address 10.0.0.1 255.0.0.0
# noshutdown
# exit

# interface se1/0
  ip address 20.0.0.1  255.0.0.0
  encapsulation ppp
  clock rate 64000
  no shutdown
  exit
```

Encapsulation ppp for all interfaces.
clock rate 64000 for interfaces with clock.

3. 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 R2,

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

In R3,

```
# 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 2
# exit
```

Router id -> identifies router

4. Check routing table of R1.

```
Router # show ip route
```

5. Configure loopback address to routers.

```
R1(config-if) # interface loopback 0
              # ip add 172.16.1.252 255.255.0.0
              # no shutdown

R2(config-if) # interface loopback 0
              # add 172.16.1.253 255.255.0.0
              # no shutdown

R3(config-if) # interface loopback 0
              # ip add 172.16.1.254 255.255.0.0
              # no shutdown.
```

5. Now, check routing table of R3.

```
R3# show ip route
```

6. Create virtual link b/w R1, R2 bcz to connect area 3 to area 0.

R1:

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

R2:

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

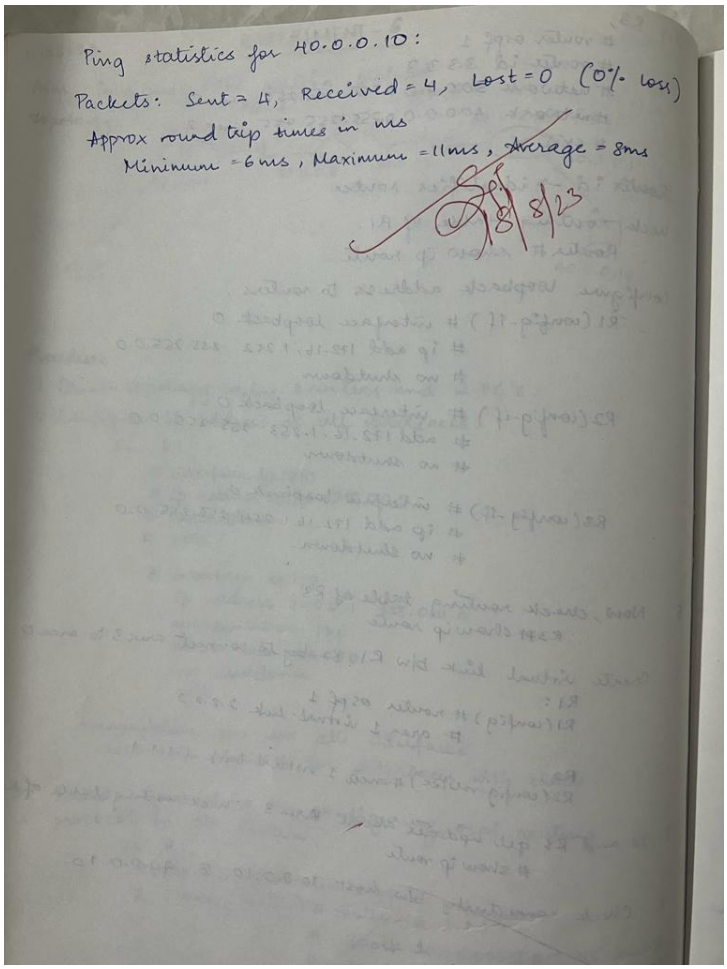
7. R2 and R3 get updates about area 3. Check routing table of R3.

```
# show ip route
```

8. Check connectivity b/w host 10.0.0.10 to 40.0.0.10.

Observation / Result:

```
ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Reply from 40.0.0.10 : bytes = 32 time = 10 ms  TTL = 125
Reply from  "          "          time = 3 ms    "
Reply from  "          "          time = 11 ms   "
Reply from  "          "          time = 6 ms    "
```

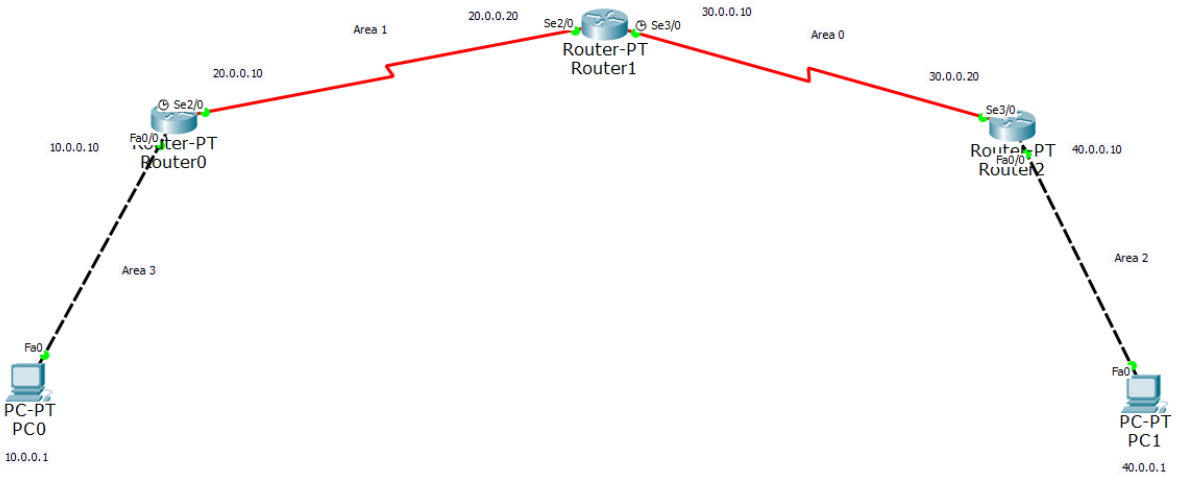


Experiment 7:

Demonstrate the TTL/ Life of a Packet

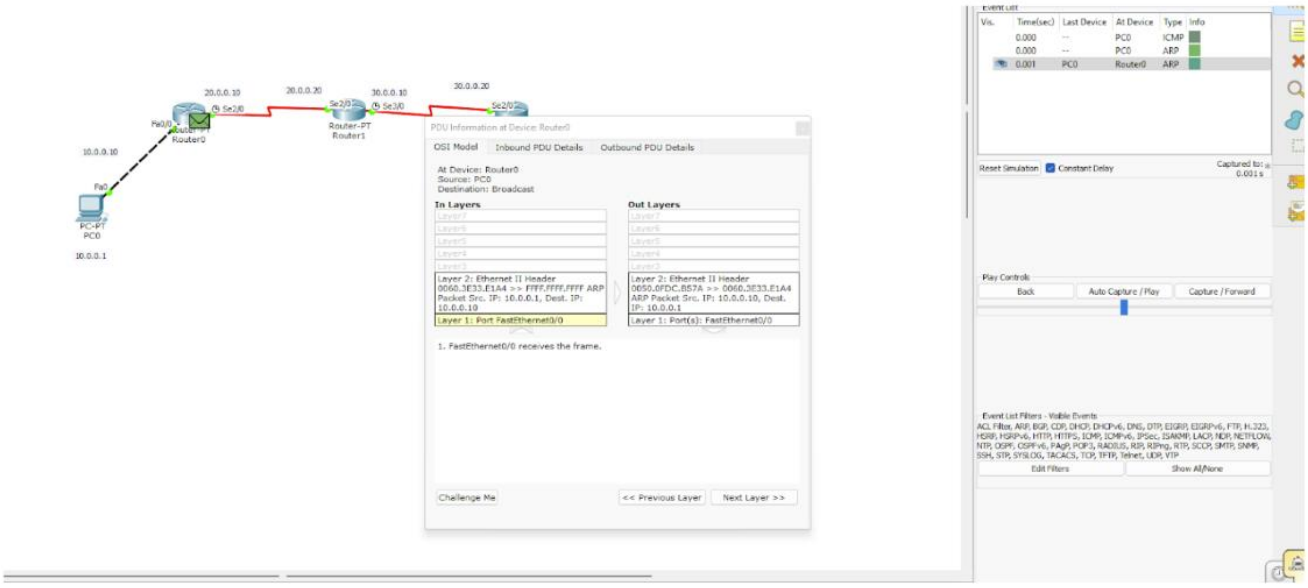
Aim: Demonstrate the TTL/ Life of a Packet

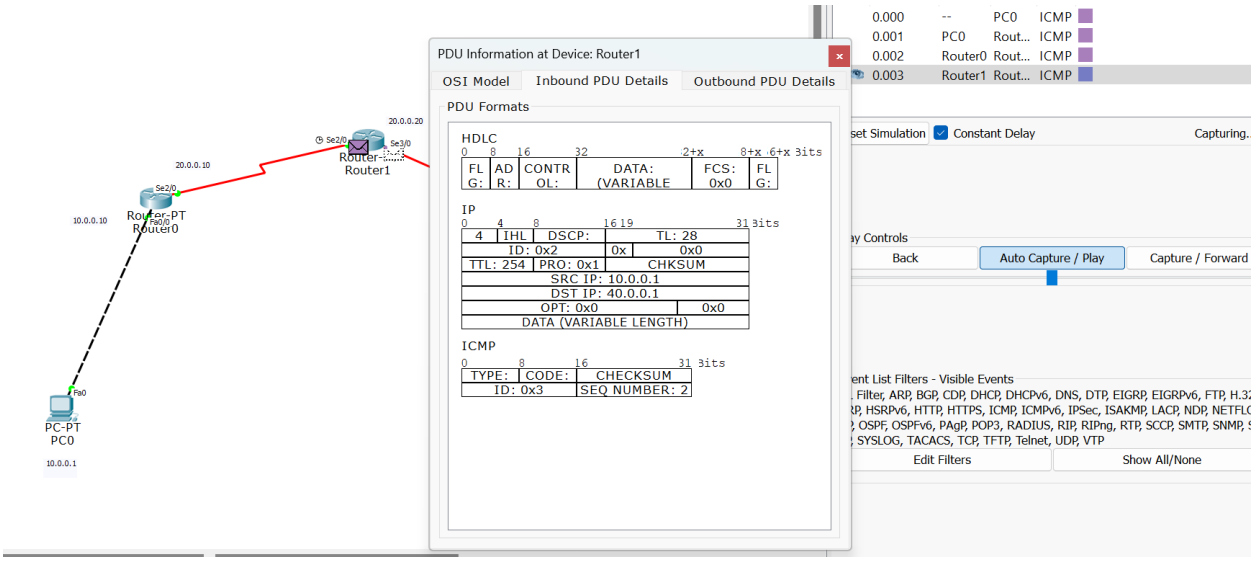
Topology:

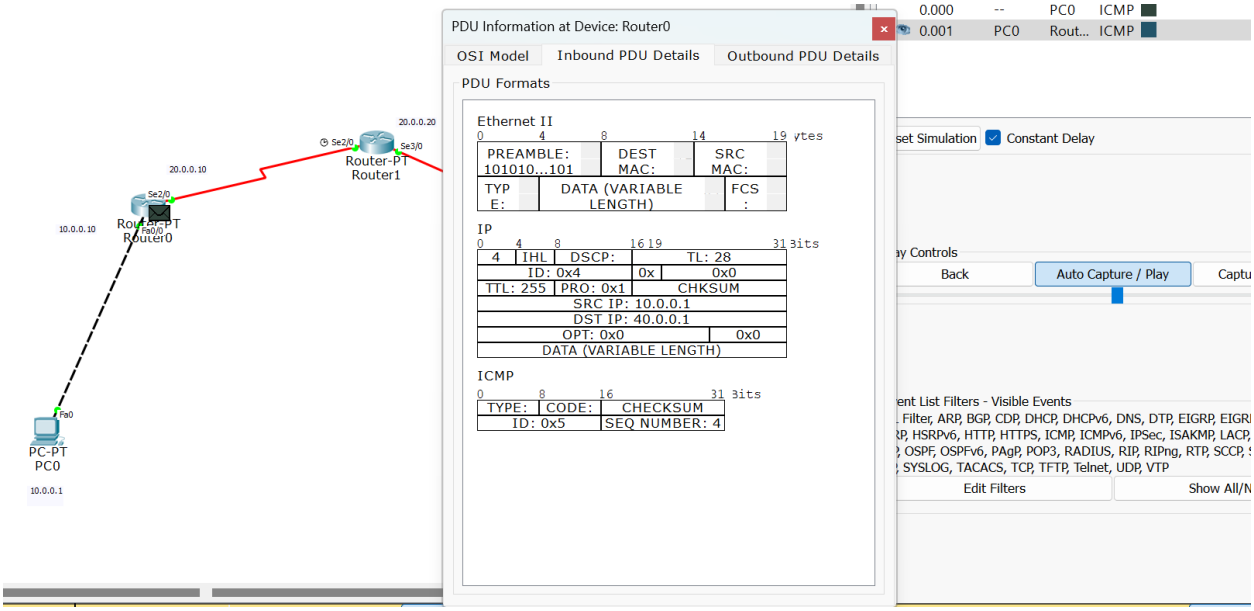


Output:

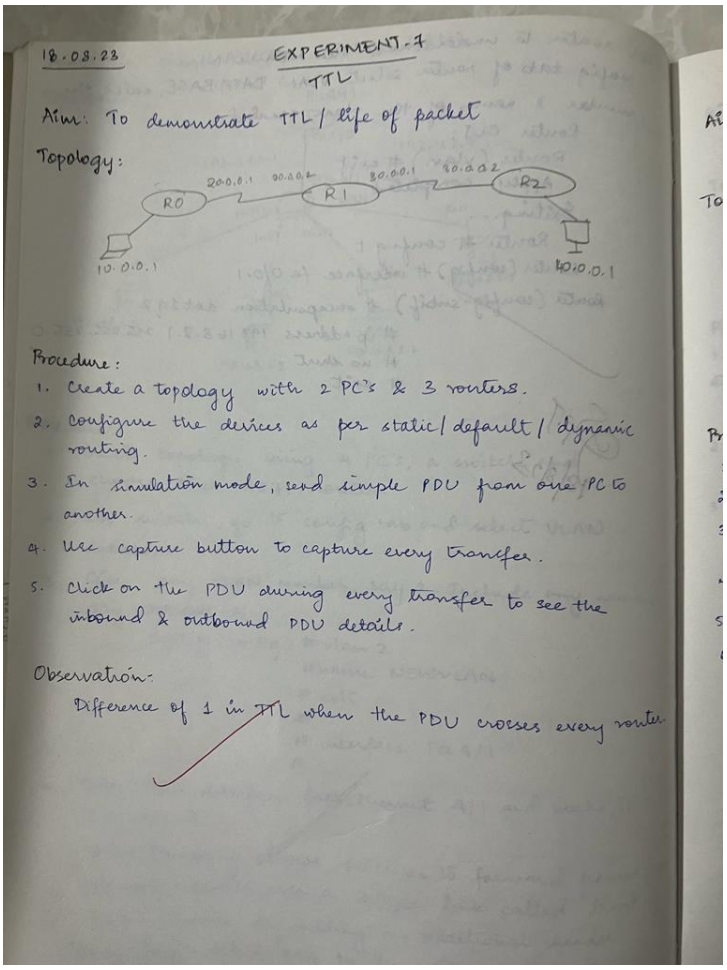
Simple PDU sent from PC0 to PC1 in simulation mode.







Observation:


18.08.23

EXPERIMENT-7

TTL

Aim: To demonstrate TTL / life of packet

Topology:

Procedure:

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

Observation:

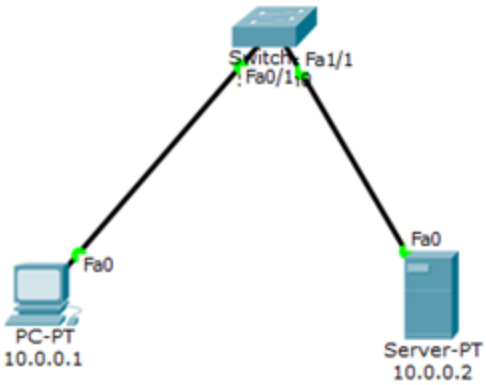
Difference of 1 in TTL when the PDU crosses every router.

Experiment 8:

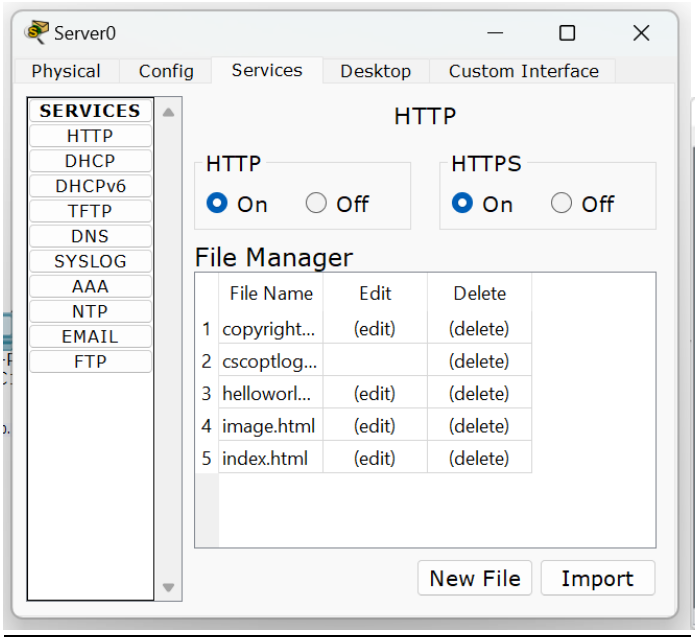
Configure Web Server, DNS within a LAN.

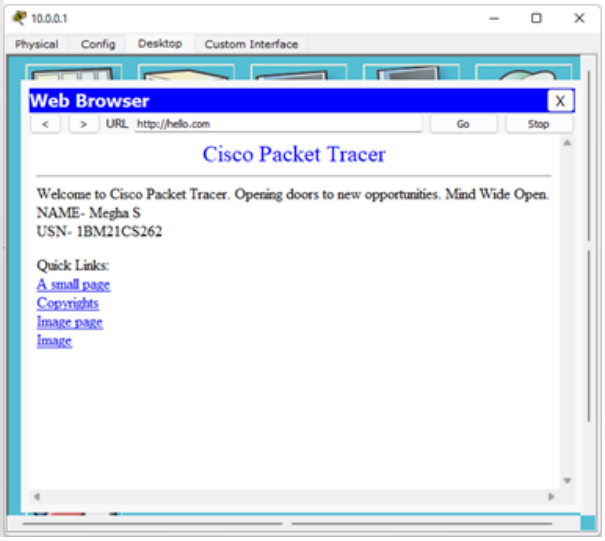
Aim: Configure Web Server, DNS within a LAN.

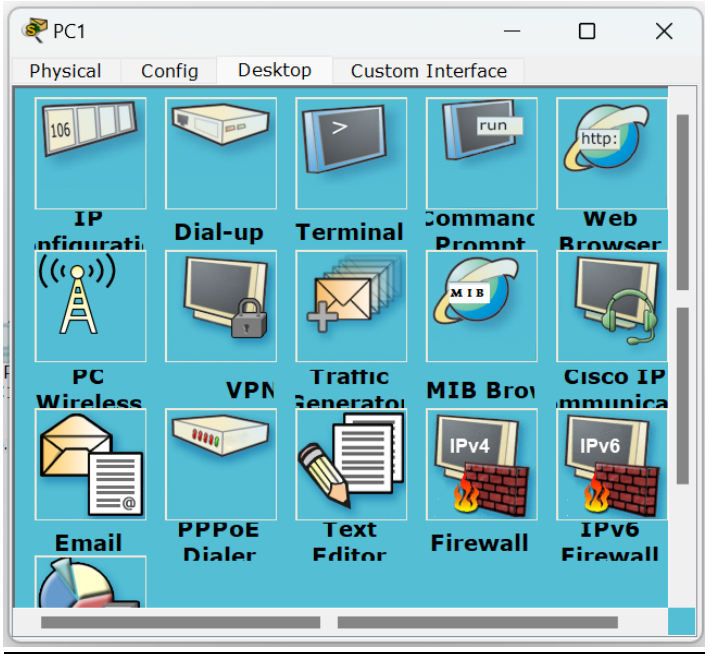
Topology:


Fa1/1
Fa0
Fa0
PC-PT
10.0.0.1
Server-PT
10.0.0.2

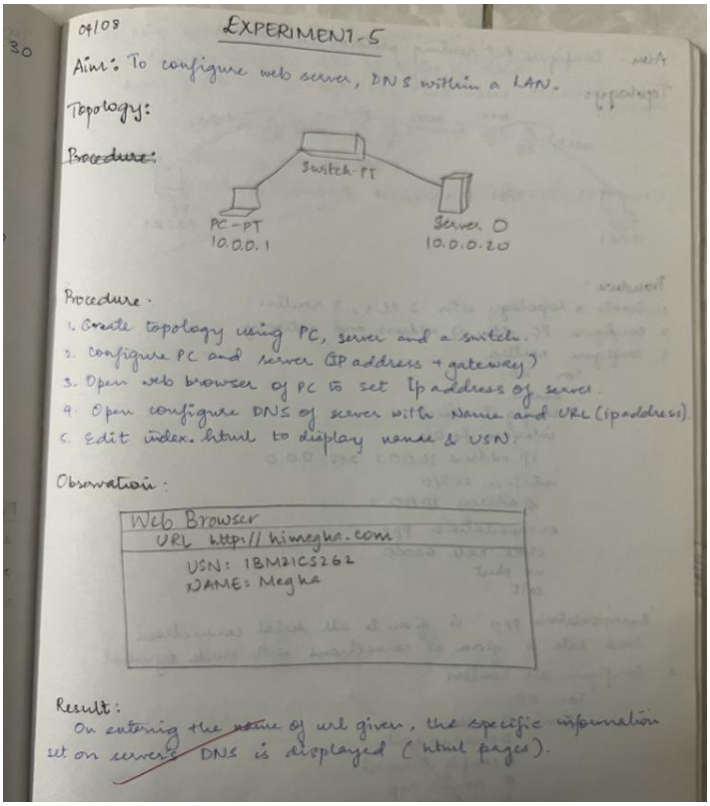
Output:


Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
HTTP
HTTP
On
Off
HTTPS
On
Off
File Manager
File Name
Edit
Delete
1 copyright... (edit) (delete)
2 cscoptlog... (delete)
3 helloworl... (edit) (delete)
4 image.html (edit) (delete)
5 index.html (edit) (delete)
New File
Import


10.0.0.1
Physical
Config
Desktop
Custom Interface
Web Browser
URL http://hello.com
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
NAME- Megha S
USN- 1BM21CS262
Quick Links:
A small page
Copyrights
Image page
Image



Observation:

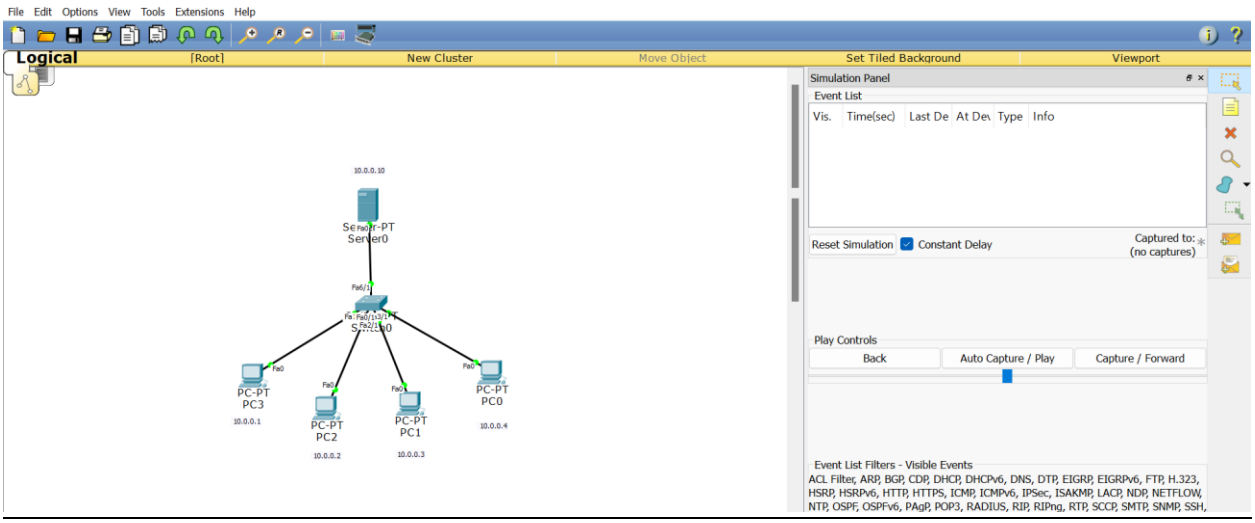


Experiment 9:

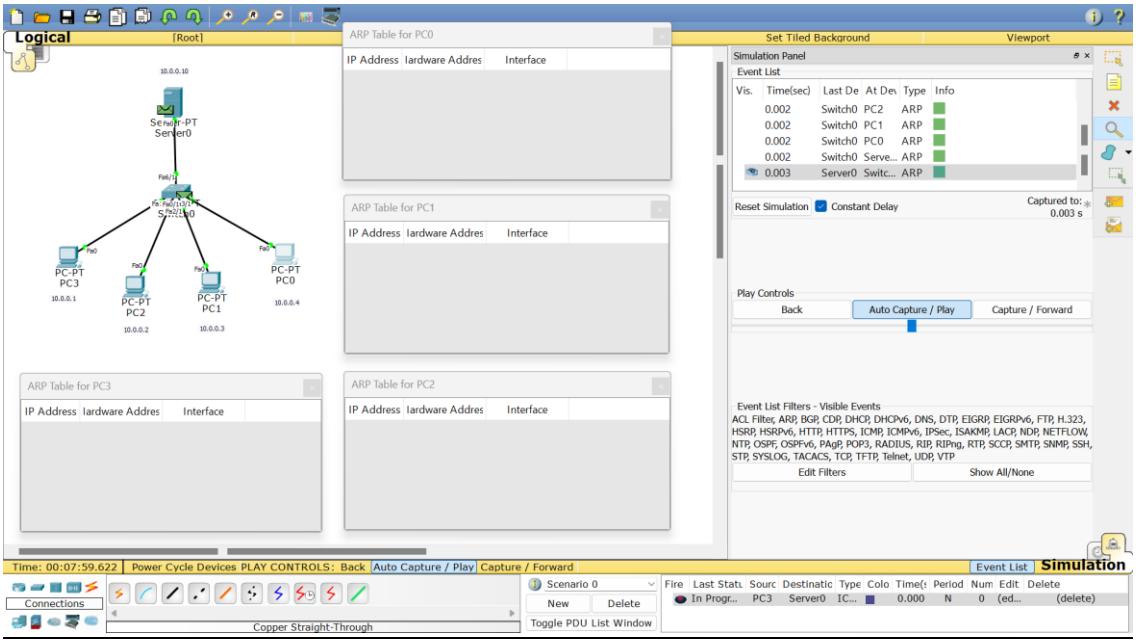
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

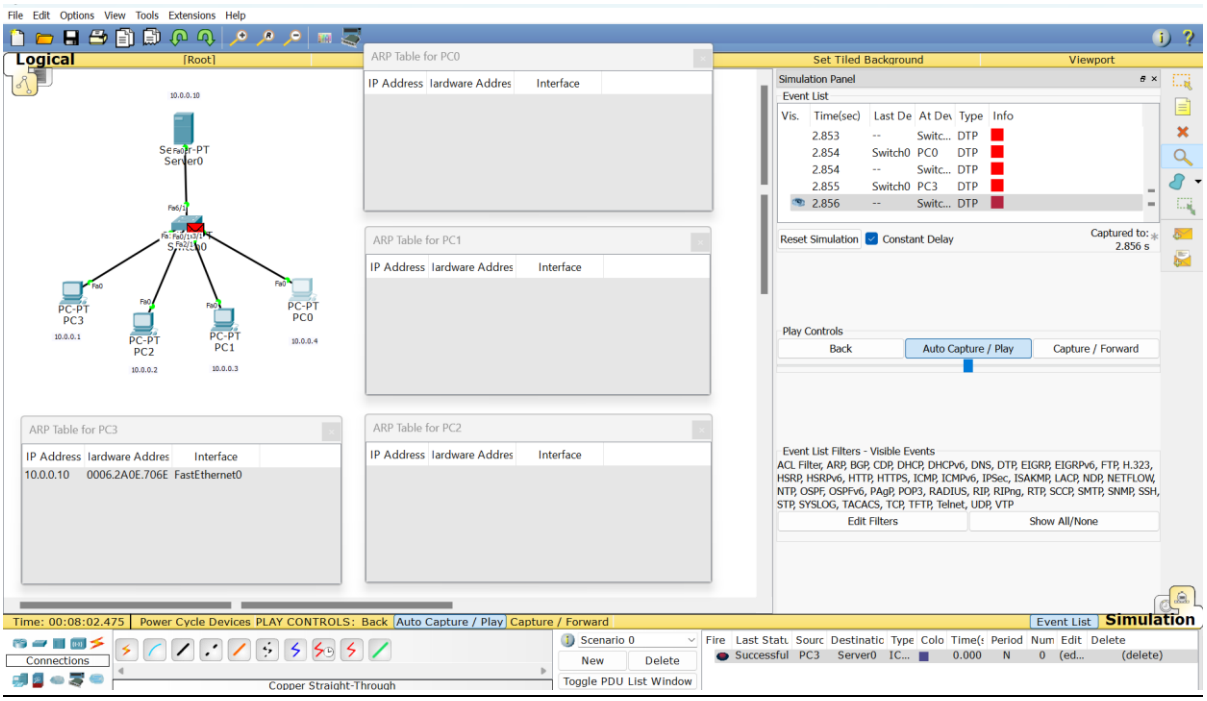
Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

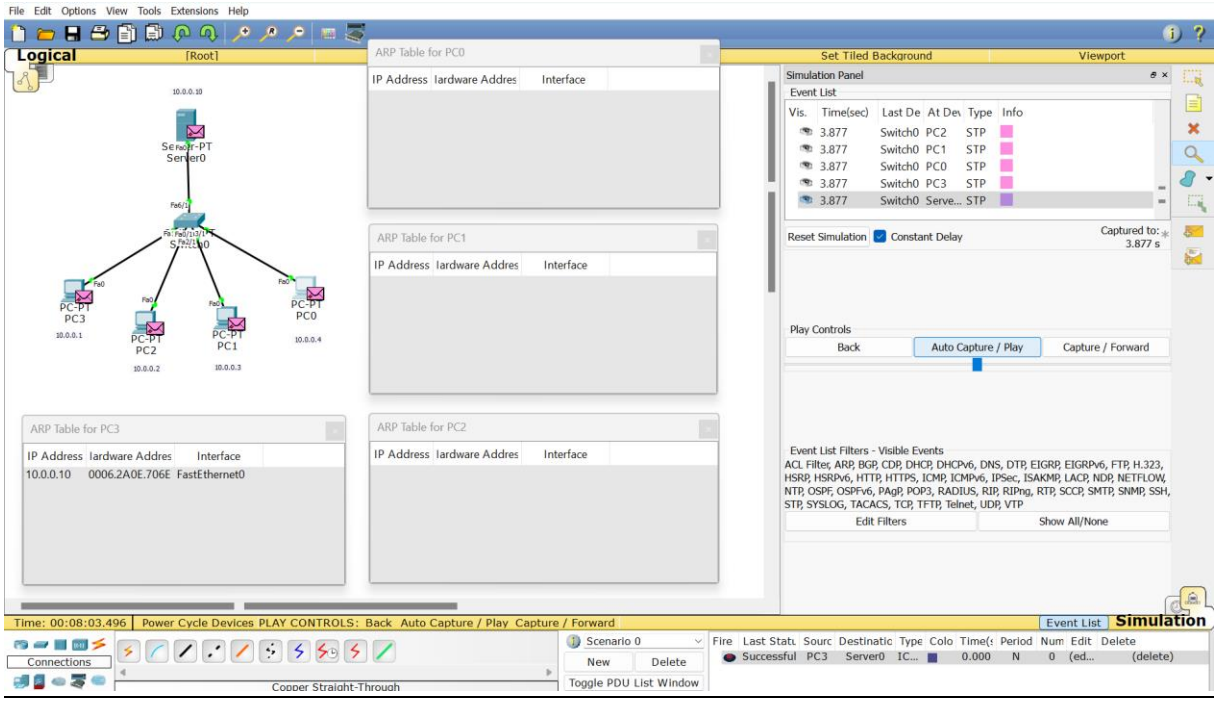
Topology:

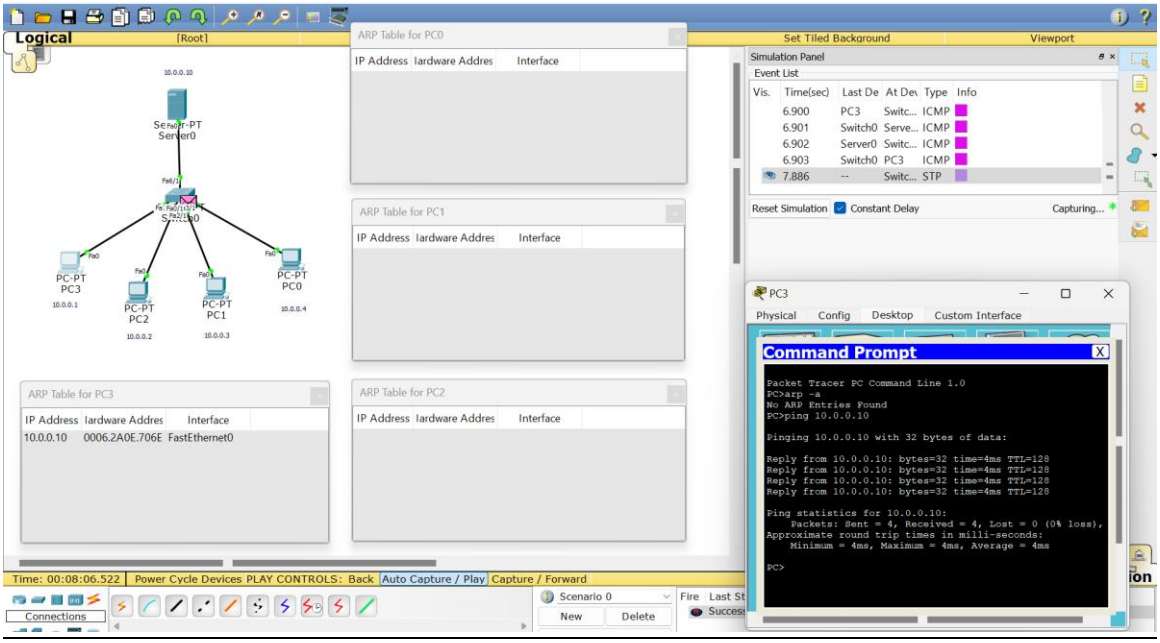

File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event 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,
10.0.0.10
Server-PT Server0
2960-24TT Switch0
PC-PT PC3
10.0.0.1
PC-PT PC2
10.0.0.2
PC-PT PC1
10.0.0.3
PC-PT PC0
10.0.0.4

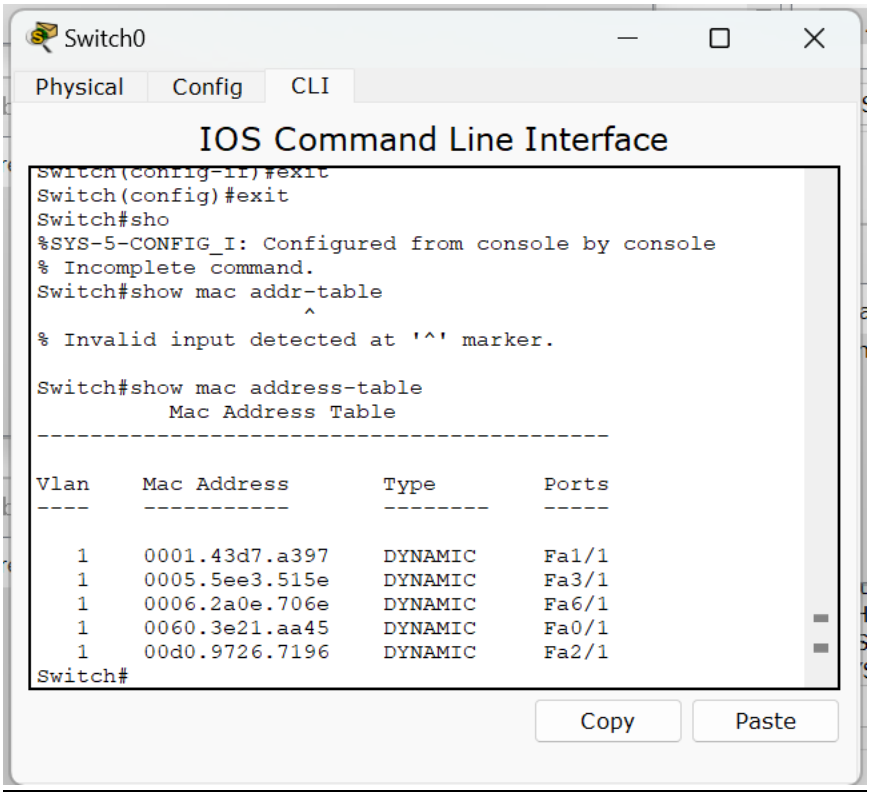
Output:

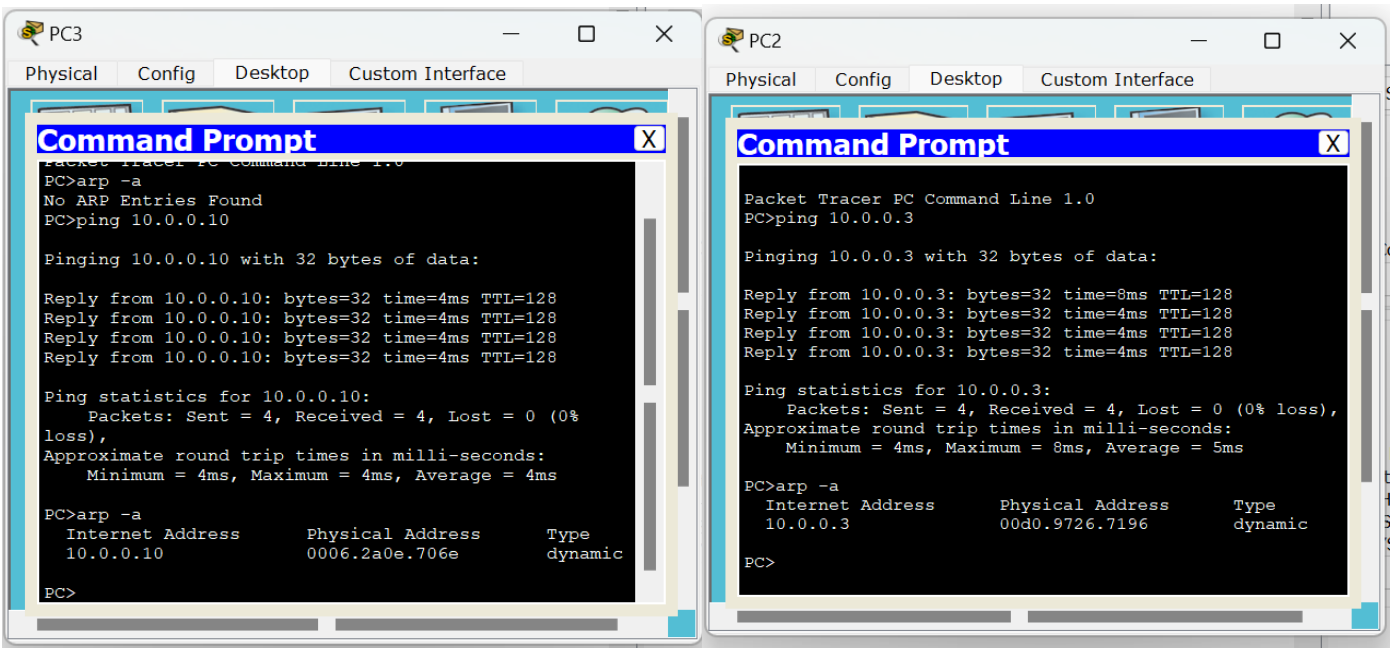

Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
ARP Table for PC1
ARP Table for PC3
ARP Table for PC2
IP Address lardware Addres Interface
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type Info
0.002 Switch0 PC2 ARP
0.002 Switch0 PC1 ARP
0.002 Switch0 PC0 ARP
0.002 Switch0 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 Stat Sourc Destinatic Type Colo Time(Period Num Edit Delete
In Progr... PC3 Server0 IC... 0.000 N 0 (ed... (delete)

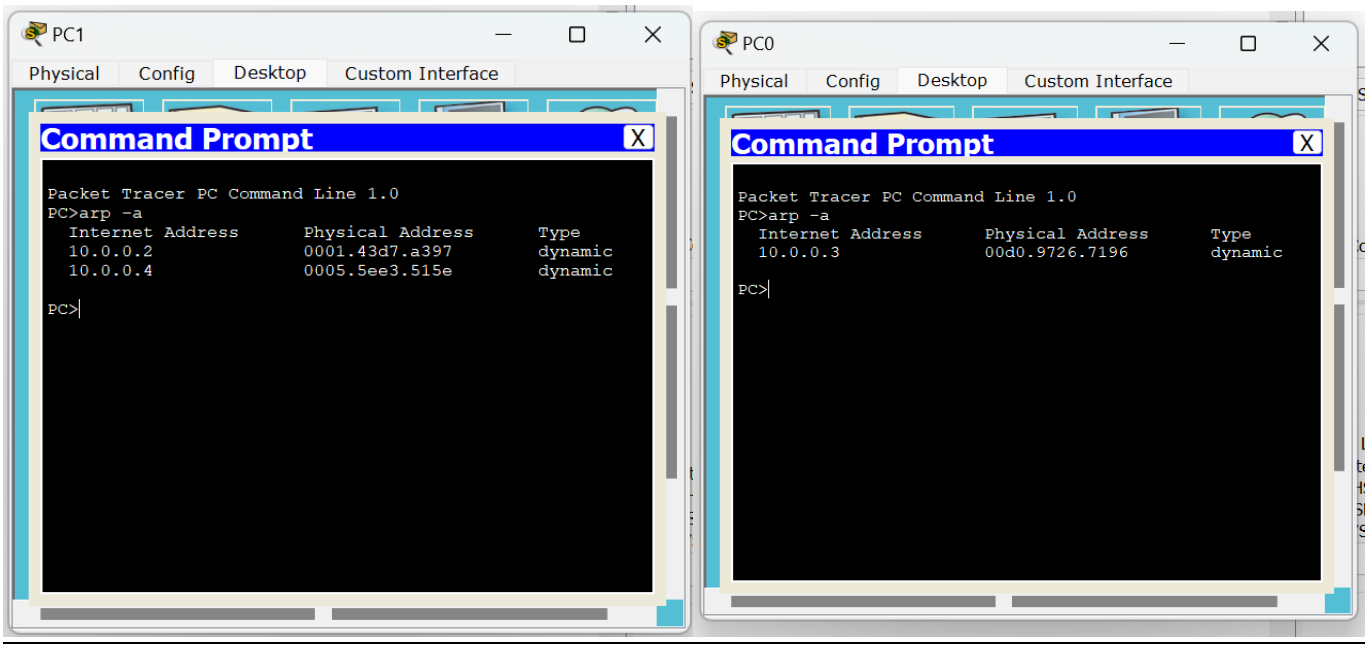

File Edit Options View Tools Extensions Help
Logical
[Root]
Set Tiled Background
Viewport
ARP Table for PC0
IP Address
Hardware Address
Interface
ARP Table for PC1
ARP Table for PC2
ARP Table for PC3
10.0.0.10 0006.2A0E.706E FastEthernet0
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type 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
Hardware Address
Interface
ARP Table for PC1
ARP Table for PC2
ARP Table for PC3
10.0.0.10 0006.2A0E.706E FastEthernet0
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)

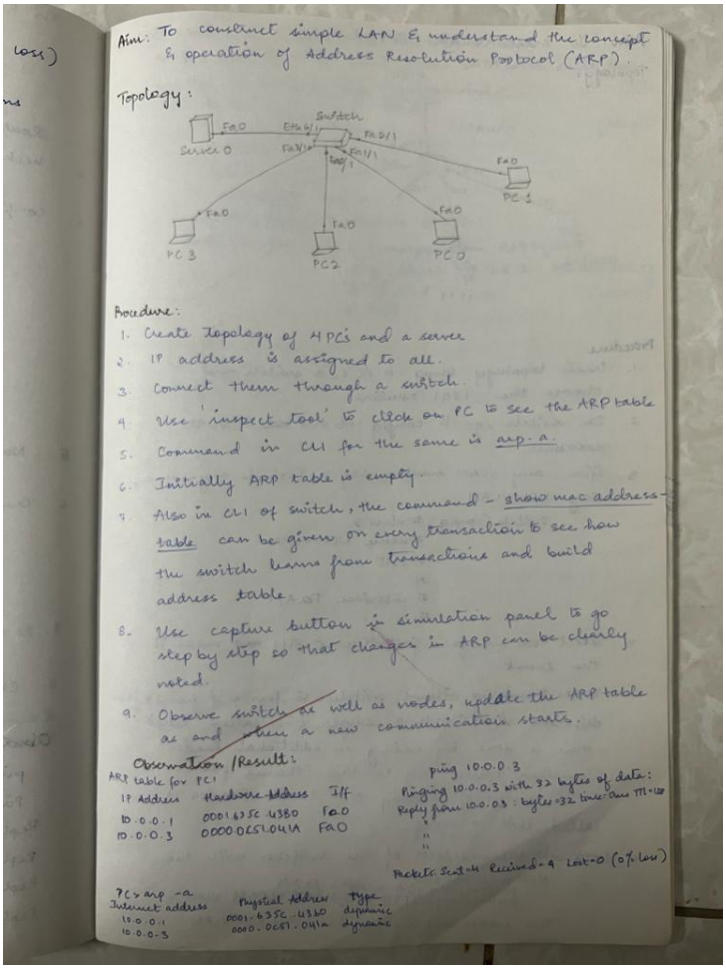

Logical
[Root]
Set Tiled Background
Viewport
Server-PT Server0
PC-PT PC3
PC-PT PC2
PC-PT PC1
PC-PT PC0
ARP Table for PC0
ARP Table for PC1
ARP Table for PC3
ARP Table for PC2
IP Address
Interface
10.0.0.10 0006.2A0E.706E FastEthernet0
Simulation Panel
Event List
Reset Simulation
Constant Delay
PC3
Physical
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
Ping statistics for 10.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Minimum = 4ms, Maximum = 4ms, Average = 4ms
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>


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:

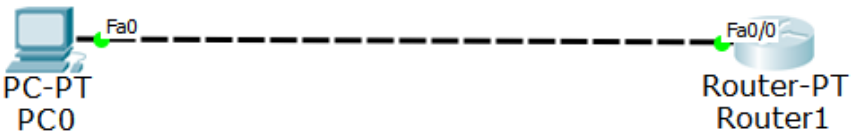


Experiment 10:

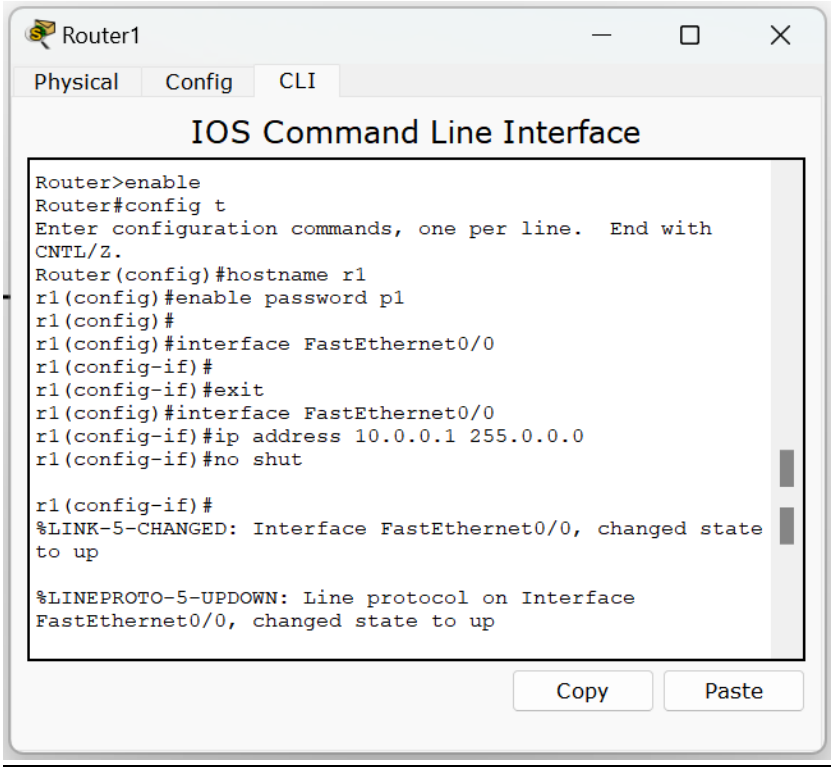
To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.

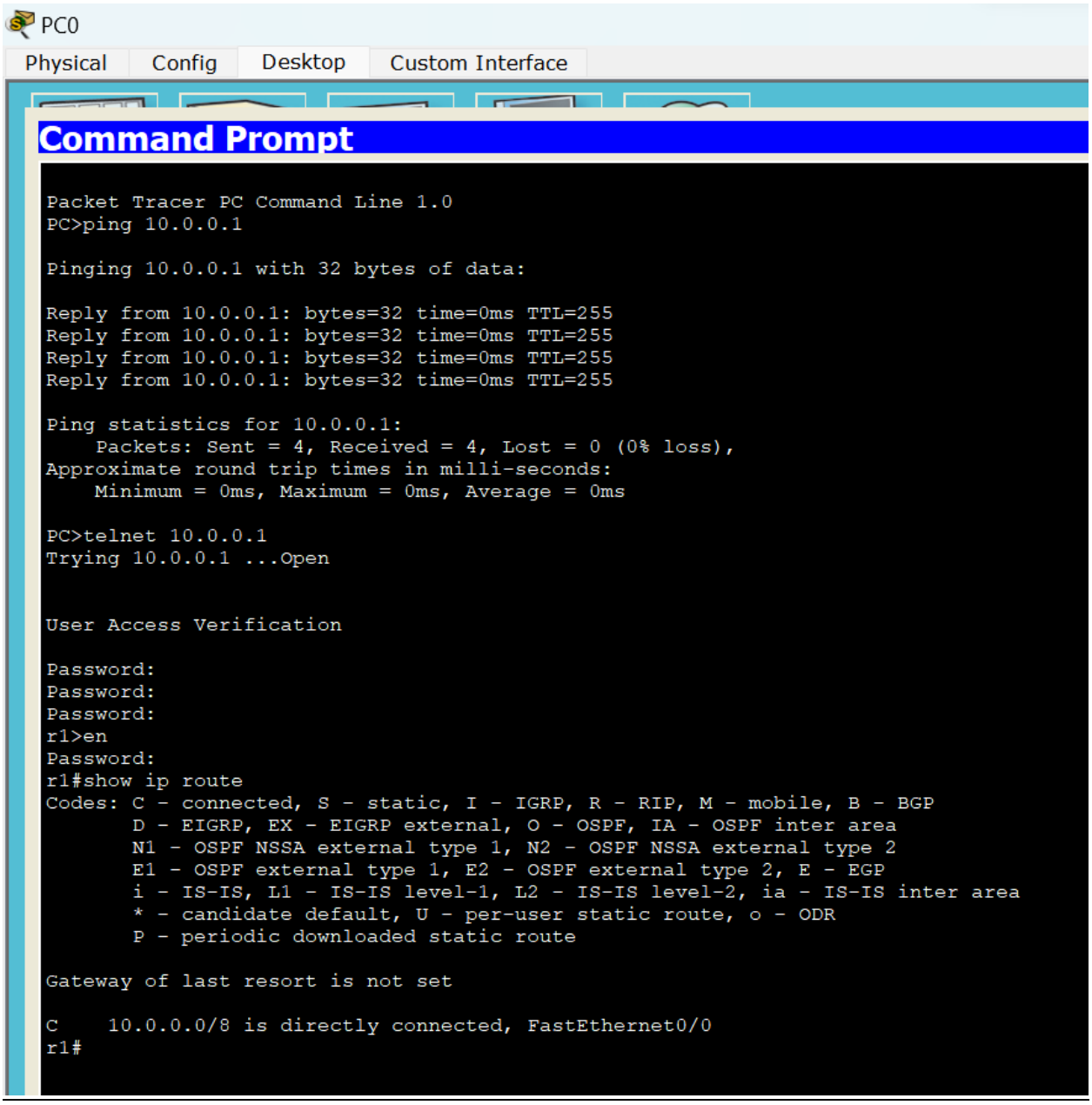
Aim: To understand the operation of TELNET by accessing the router in server room from a PC in the IT office.

Topology:

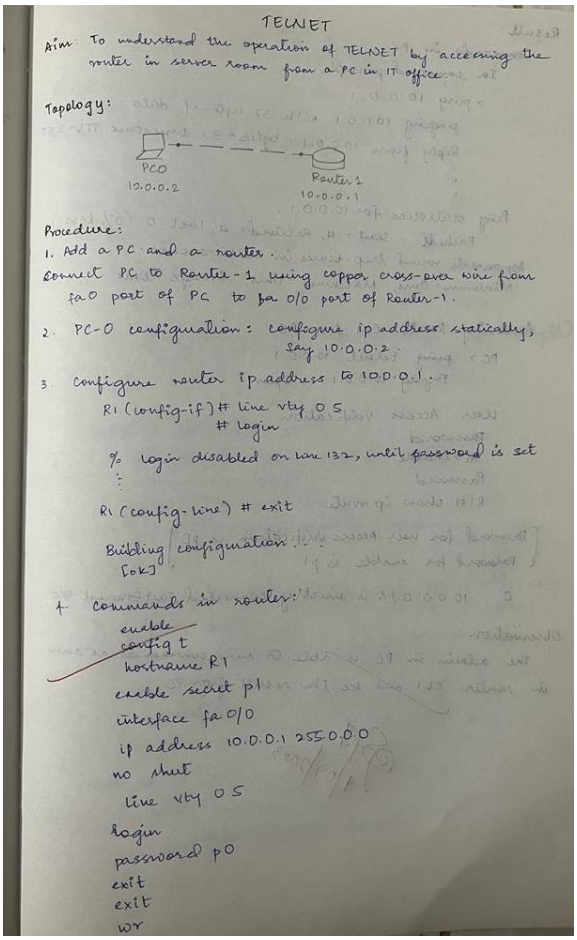

Fa0
Fa0/0
PC-PT
PC0
Router-PT
Router1

Output:


Router1
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#config t
Enter configuration commands, one per line. End with
CNTL/Z.
Router(config)#hostname r1
r1(config)#enable password p1
r1(config)#
r1(config)#interface FastEthernet0/0
r1(config-if)#
r1(config-if)#exit
r1(config)#interface FastEthernet0/0
r1(config-if)#ip address 10.0.0.1 255.0.0.0
r1(config-if)#no shut
r1(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state
to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to up
Copy
Paste


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:



TELNET

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

Topology:

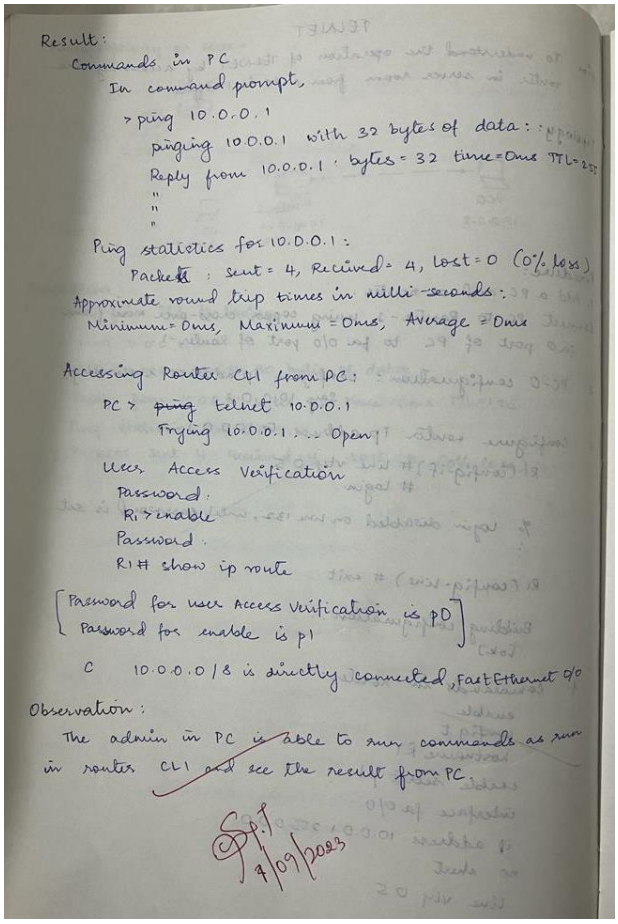
Procedure:

1. Add a PC and a router.
Connect PC to Router-1 using copper cross-over wire from fa0 port of PC to fa 0/0 port of Router-1.
2. PC-0 configuration: configure ip address statically, say 10.0.0.2.
3. Configure router ip address to 10.0.0.1.

```
R1(config-if)# line vty 0 5
              # login
% login disabled on line 132, until password is set
:
R1(config-line) # exit
Building configuration...
[OK]
```

4. Commands in router:

```
enable
config t
hostname R1
enable secret p1
interface fa 0/0
ip address 10.0.0.1 255.0.0.0
no shut
line vty 0 5
login
password p0
exit
exit
wr
```



Result:

Commands in PC

In command prompt,

> 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=2

"

"

"

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

Accessing Router CLI from PC:

PC > ~~ping~~ telnet 10.0.0.1

Trying 10.0.0.1 ... Open

User Access Verification

Password:

R1>enable

Password:

R1# show ip route

[Password for user Access Verification is pD
Password for enable is p1]

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

Observation:

The admin in PC is able to run commands as run in router CLI and see the result from PC.

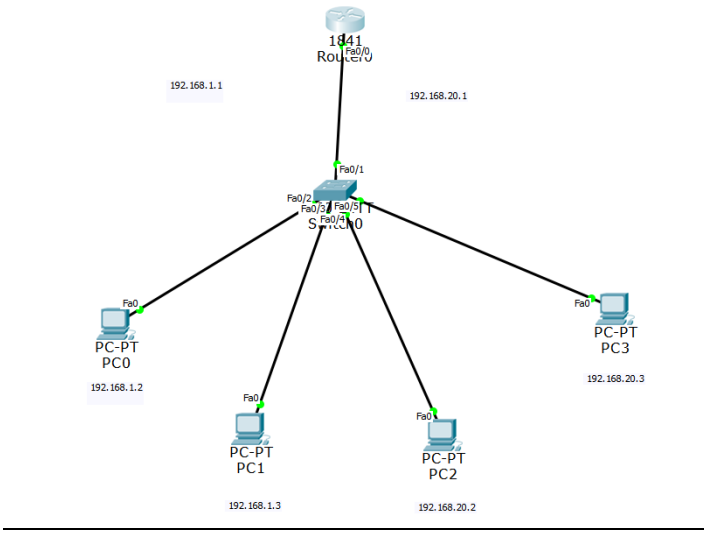
Sp.J
4/09/2023

Experiment 11:

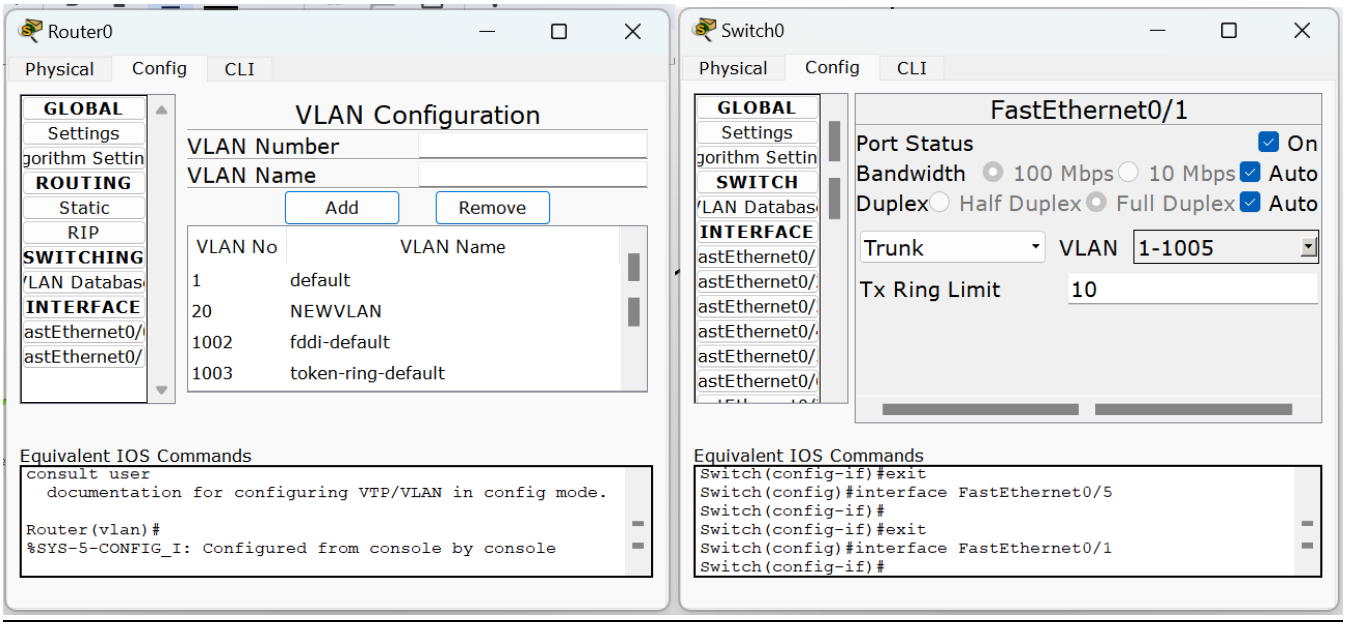
To construct a VLAN and make the PC's communicate among a VLAN

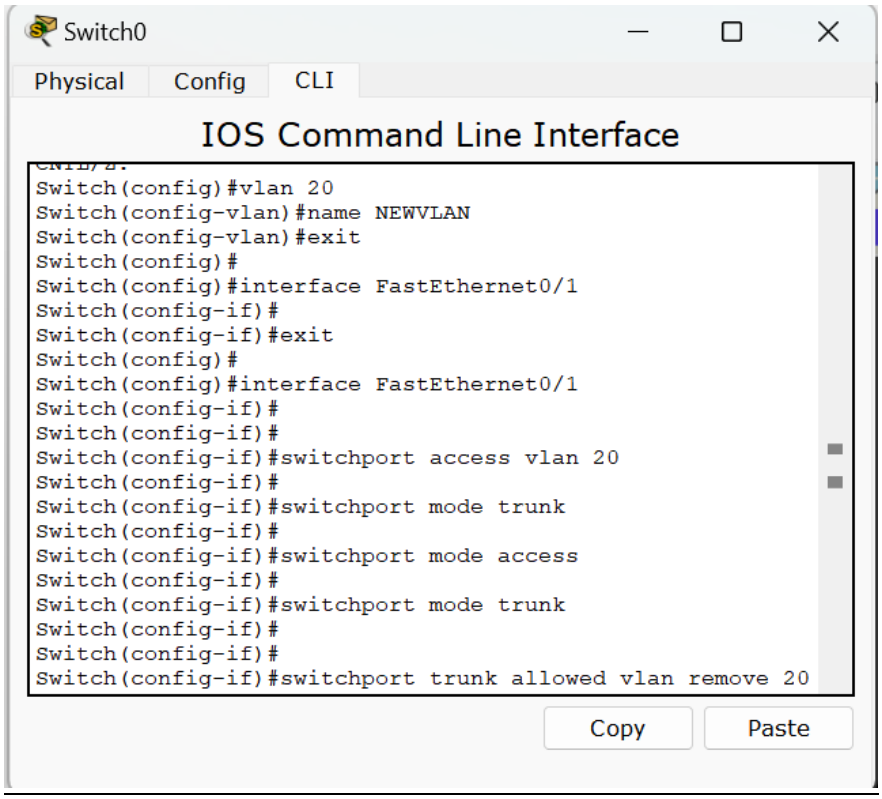
Aim: To construct a VLAN and make the PC's communicate among a VLAN

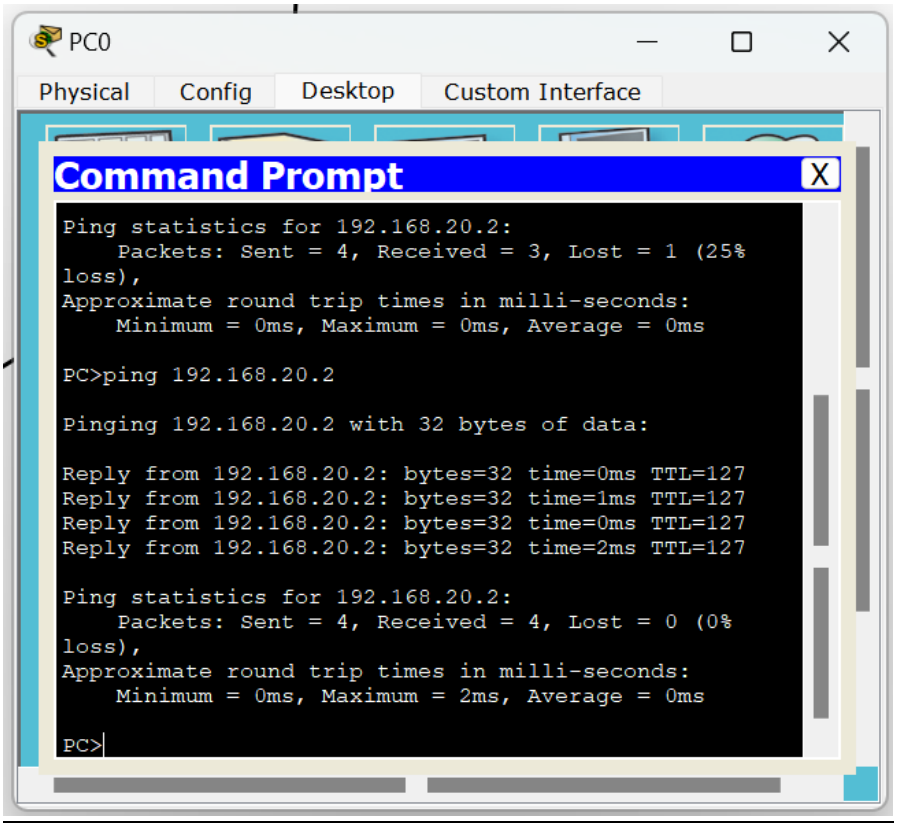
Topology:


1841
Router0
Fa0/0
192.168.1.1
192.168.20.1
Fa0/1
Fa0/2
Fa0/3
Fa0/4
Fa0/5
Switch0
Fa0
PC-PT
PC0
192.168.1.2
PC-PT
PC3
192.168.20.3
PC-PT
PC1
192.168.1.3
PC-PT
PC2
192.168.20.2

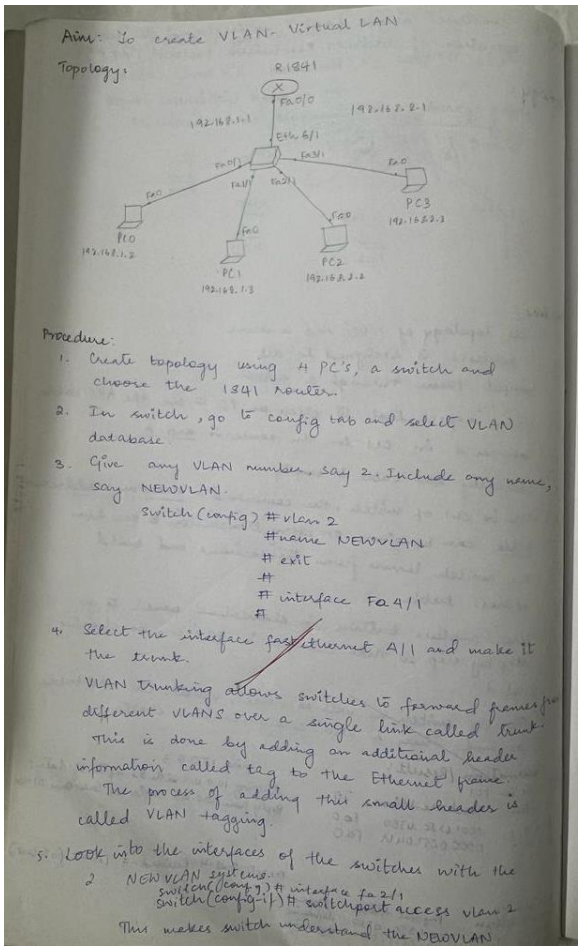
Output:

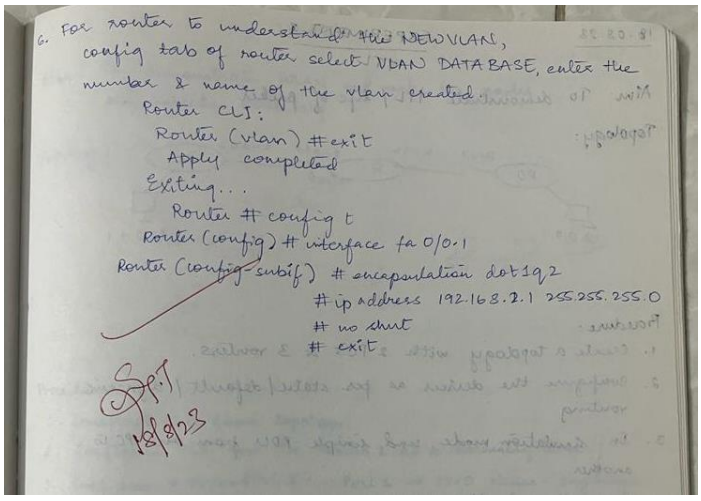

Router0
Physical
Config
CLI
GLOBAL
Settings
gorithm Settin
ROUTING
Static
RIP
SWITCHING
'LAN Databas
INTERFACE
astEthernet0/
astEthernet0/
VLAN Configuration
VLAN Number
VLAN Name
Add
Remove
VLAN No
VLAN Name
1
default
20
NEWVLAN
1002
fddi-default
1003
token-ring-default
Equivalent IOS Commands
consult user
documentation for configuring VTP/VLAN in config mode.
Router(vlan)#
%SYS-5-CONFIG_I: Configured from console by console
Switch0
Physical
Config
CLI
GLOBAL
Settings
gorithm Settin
SWITCH
'LAN Databas
INTERFACE
astEthernet0/
astEthernet0/
astEthernet0/
astEthernet0/
astEthernet0/
astEthernet0/
FastEthernet0/1
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Trunk
VLAN
1-1005
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/5
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/1
Switch(config-if)#


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:





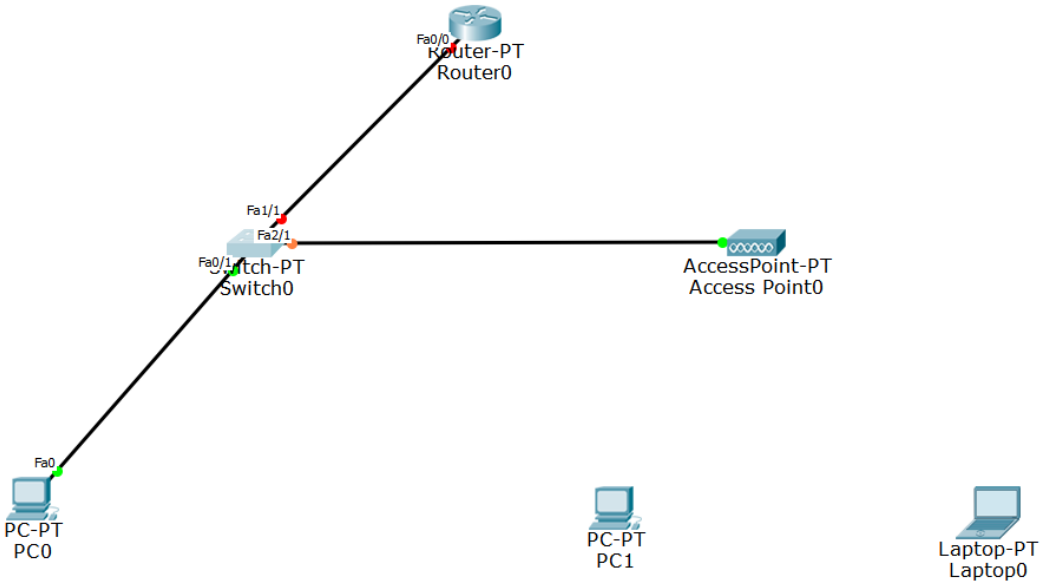
Experiment 12:

To construct a WLAN and make the nodes communicate wirelessly

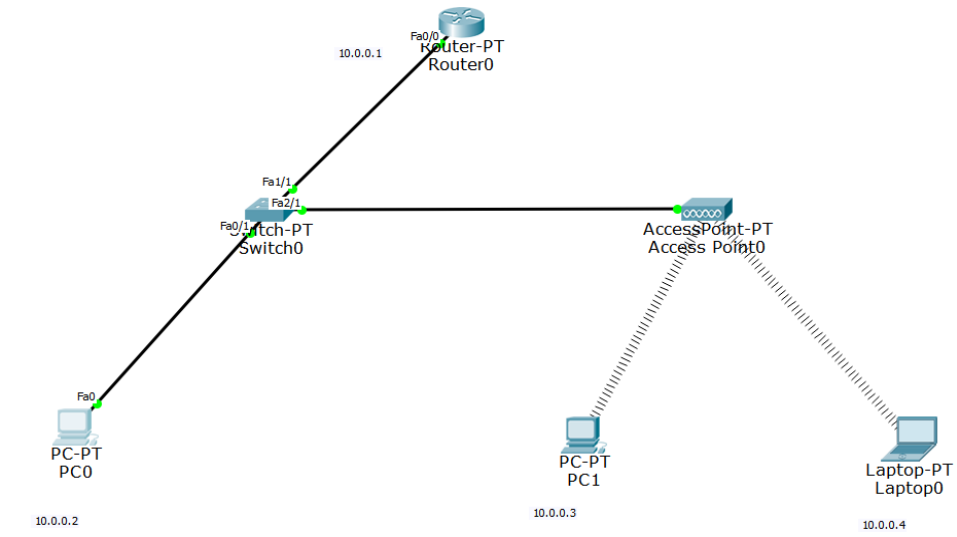
Aim: To construct a WLAN and make the nodes communicate wirelessly.

Topology:

(Initial)

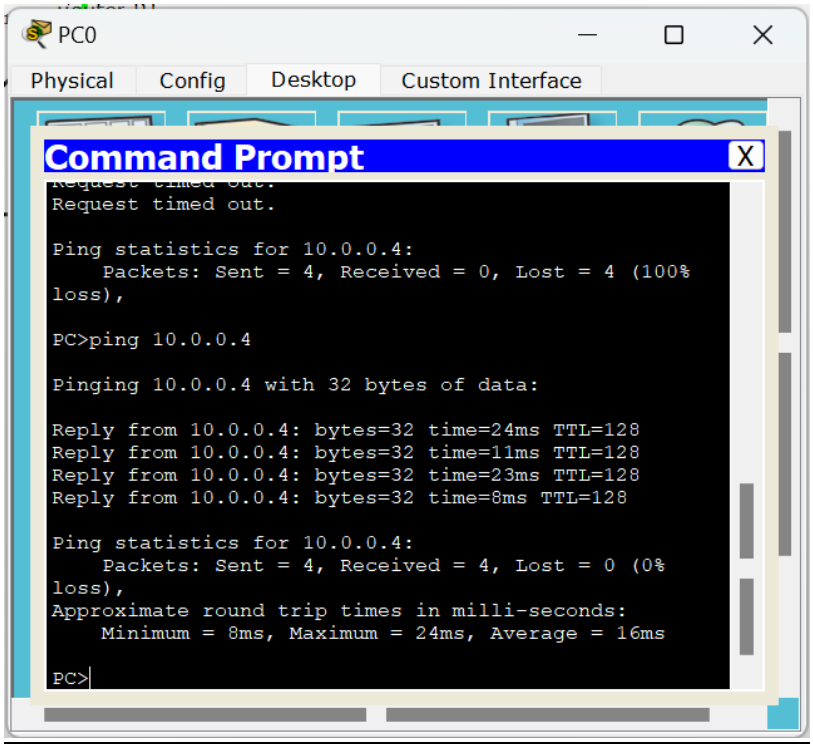


(Final)

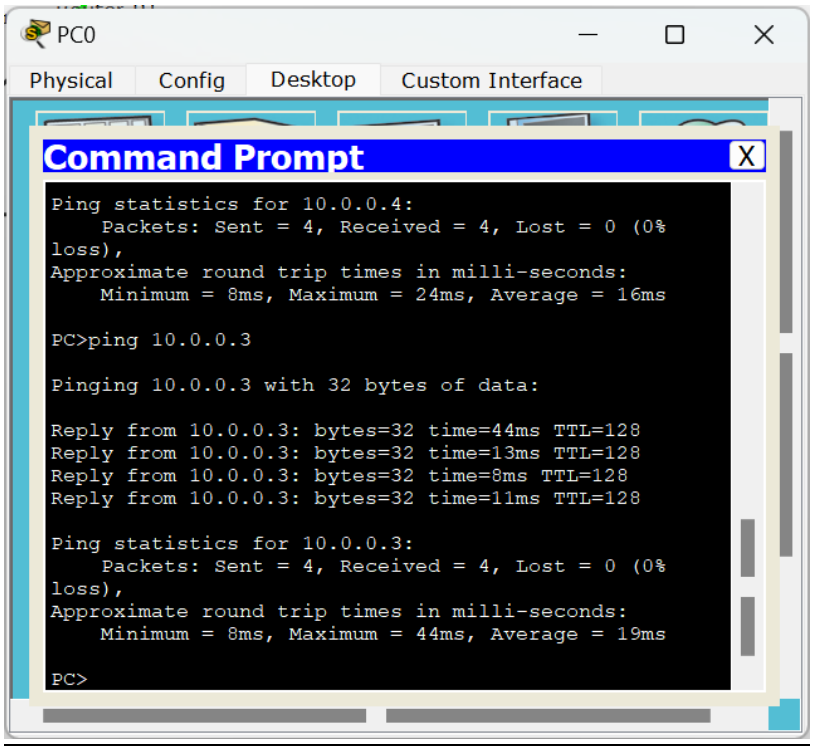


Output:

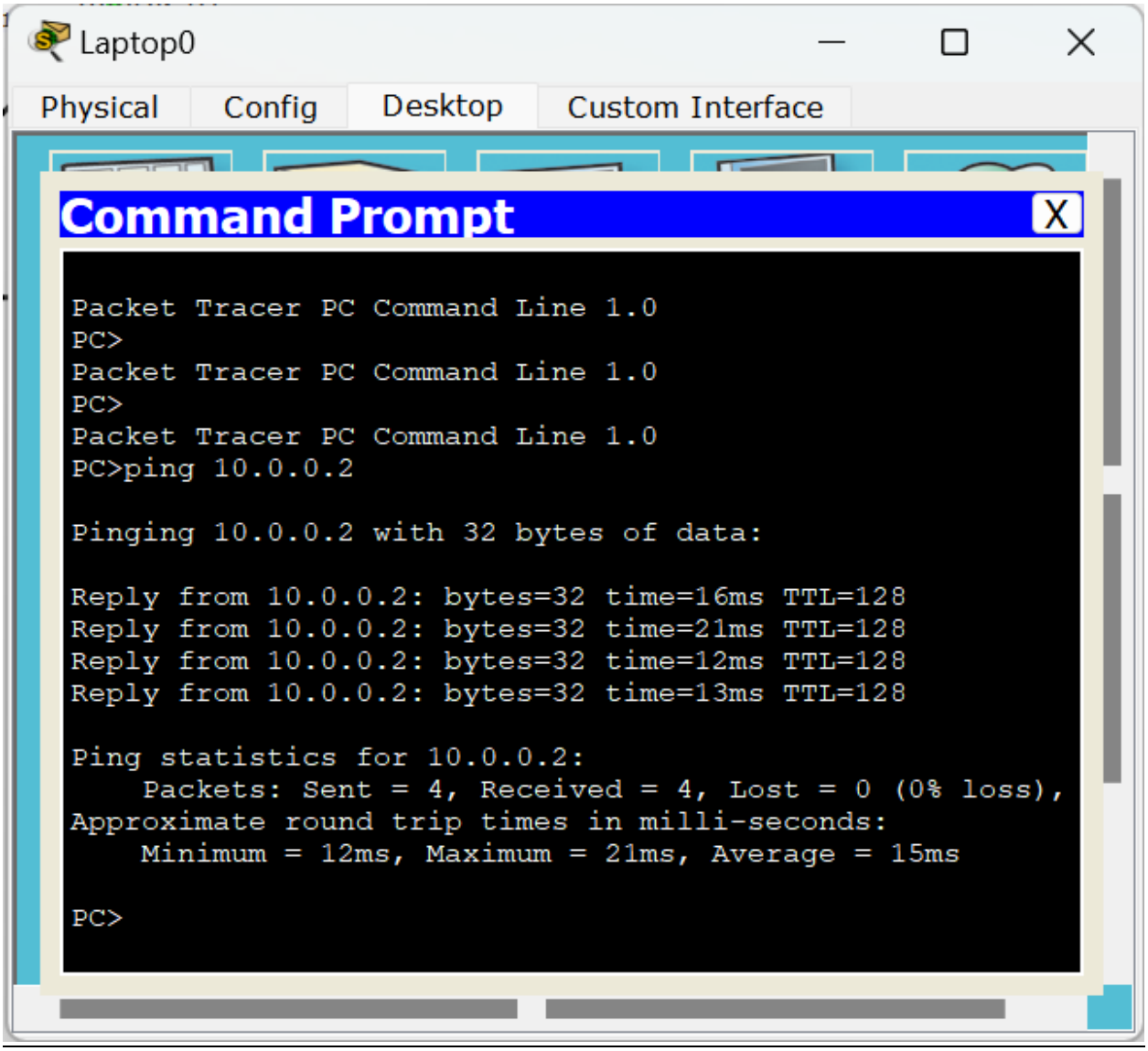
PC0 TO LAPTOP



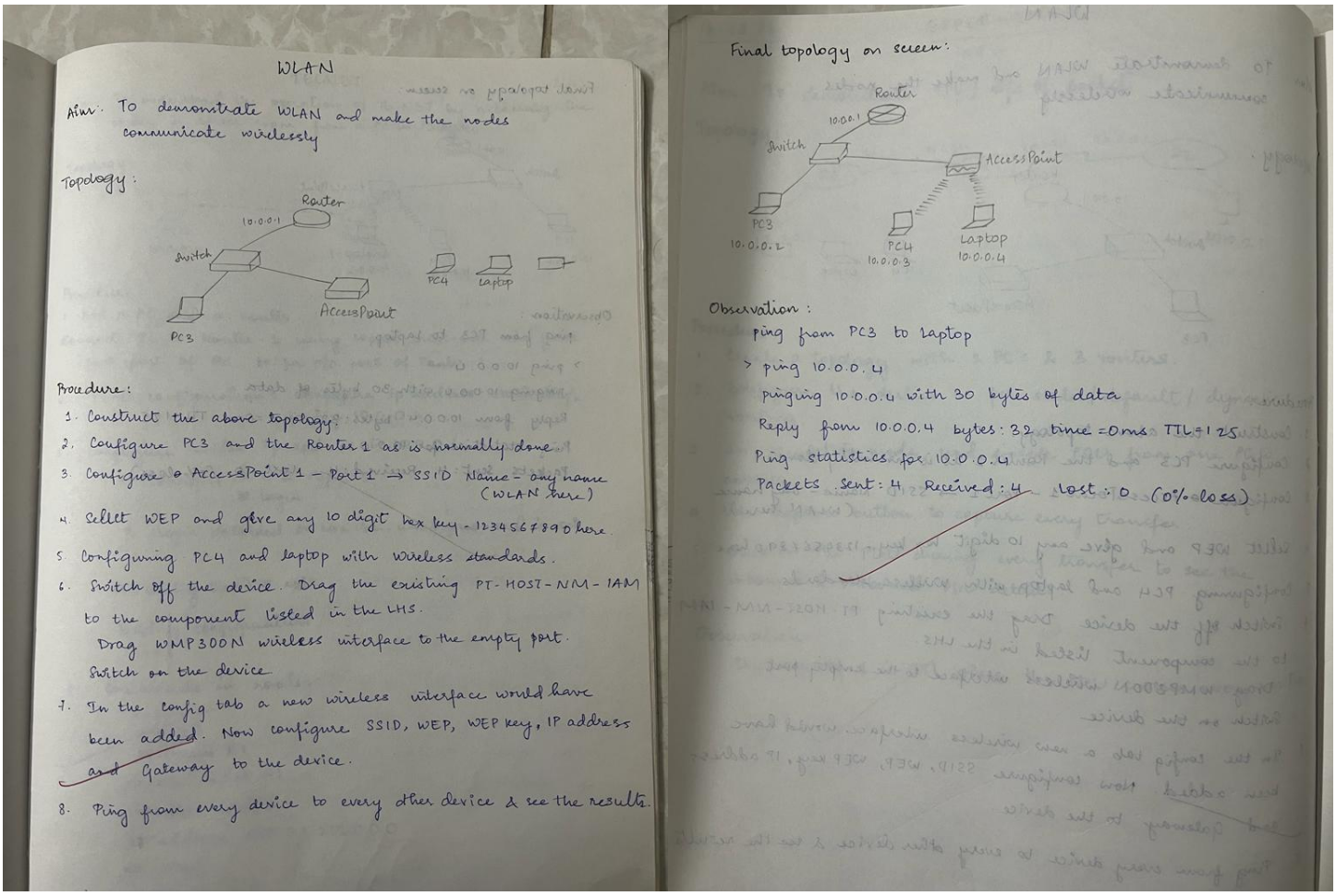
PC0 TO PC1



LAPTOP TO PC0



Observation:



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
```

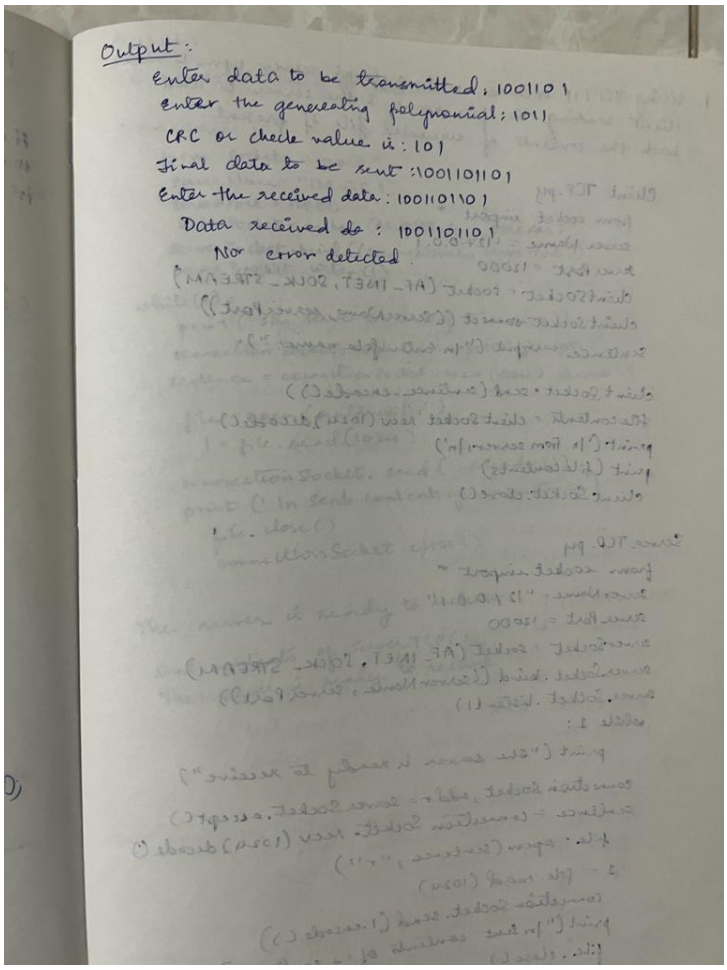
```
    // Verify the received data
    char receivedChecksum[17];
    printf("Enter received CRC: ");
    scanf("%s", receivedChecksum);

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

    return 0;
```

Observation:

Write a pgm for error detecting code using CRC-CCITT (16-bit)

```
#include <stdio.h>
#include <string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length, i, j;
void XOR() {
  for (j=1; j<N; j++)
  check_value[j] = ((check_value[j] == gen_poly[j])?
  '0':'1');
}
void receiver() {
  printf("Enter the received data:");
  scanf("%s", data);
  printf("\n ----\n");
  printf("Data received: %s", data);
  crc();
for(i=0; (i<N-1) && (check_value[i] != '1'); i++);
  if(i<N-1)
    printf("\n Error detected\n");
  else
    printf("\n No error detected\n");
}
void crc() {
    for(i=0; i<N; i++)
        check_value[i] = data[i];
```

```
do {
    if (check_value[0] == '1')
        XOR();

    for (j=0; j<N-1; j++)
        check_value[j] = check_value[j+1];
    check_value[j] = data[i++];
} while (i<=data_length+N-1);
}

int main()
{
    printf("Enter data to be transmitted: ");
    scanf("%s", data);
    printf("\n Enter generating polynomial: ");
    scanf("%s", gen_poly);
    data_length = strlen(data);

    for (i=data_length; i<data_length+N-1; i++)
        data[i] = '0';
    printf("\n - - - - - - \n");
printf(" Data padded with n-1 zeroes : %s", data);
printf("\n - - - - - - ");
    crc();
printf("\n CRC or check value is : %s", check_value);
    for(i=data_length; i<data_length+N-1; i++)
        data[i] = check_value[i-data_length];
    printf("\n - - - - - - ");
    printf("Final data to be sent : %s", data);
    printf("\n - - - - - - \n");
    receiver();
    return 0;
}
```



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

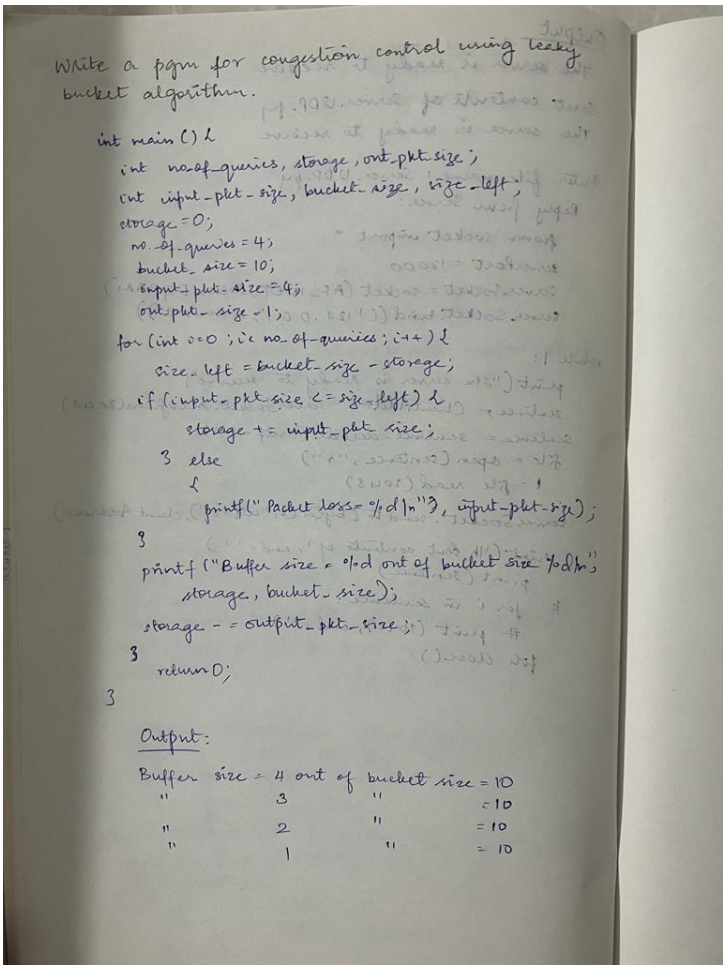
Aim: Write a program for congestion control using Leaky bucket algorithm

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:



Write a pgm for congestion control using leaky bucket algorithm.

```
int main() {
  int no_of_queries, storage, out_pkt_size;
  int input_pkt_size, bucket_size, size_left;
  storage = 0;
   no_of_queries = 4;
   bucket_size = 10;
   input_pkt_size = 4;
   out_pkt_size = 1;
  for (int i=0 ; i< no_of_queries ; i++) {
     size_left = bucket_size - storage;
     if (input_pkt_size <= size_left) {
        storage += input_pkt_size;
     } else
     {
     printf(" Packet loss = %d \n", input_pkt_size);
   }
   printf ("Buffer size = %d out of bucket size %d\n",
      storage, bucket_size);
   storage -= output_pkt_size;
  }
    return 0;
}
```

Output:

```
Buffer size = 4 out of bucket size = 10
  "         3       "          = 10
  "         2       "          = 10
  "         1       "          = 10
```

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:

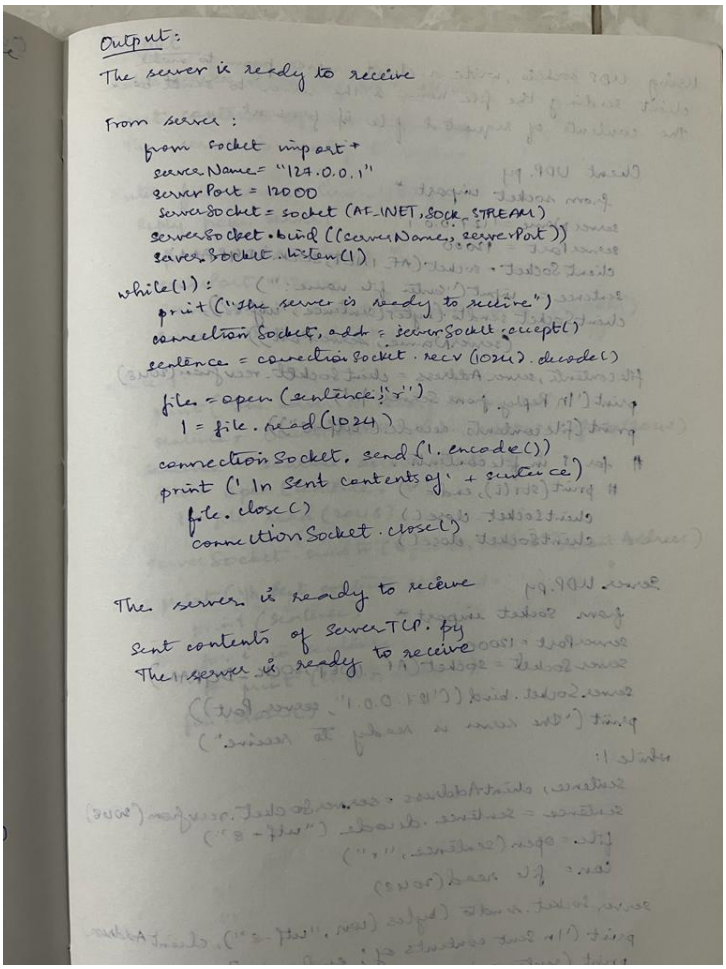
1. Using TCP/IP sockets, write a client-server pgm to make client sending the file name & the server to send back the contents of requested file if present.

Client TCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\n Enter file name: ")

clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print('\n From server:\n')
print(fileContents)
clientSocket.close()
```

Server_TCP.py

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



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:

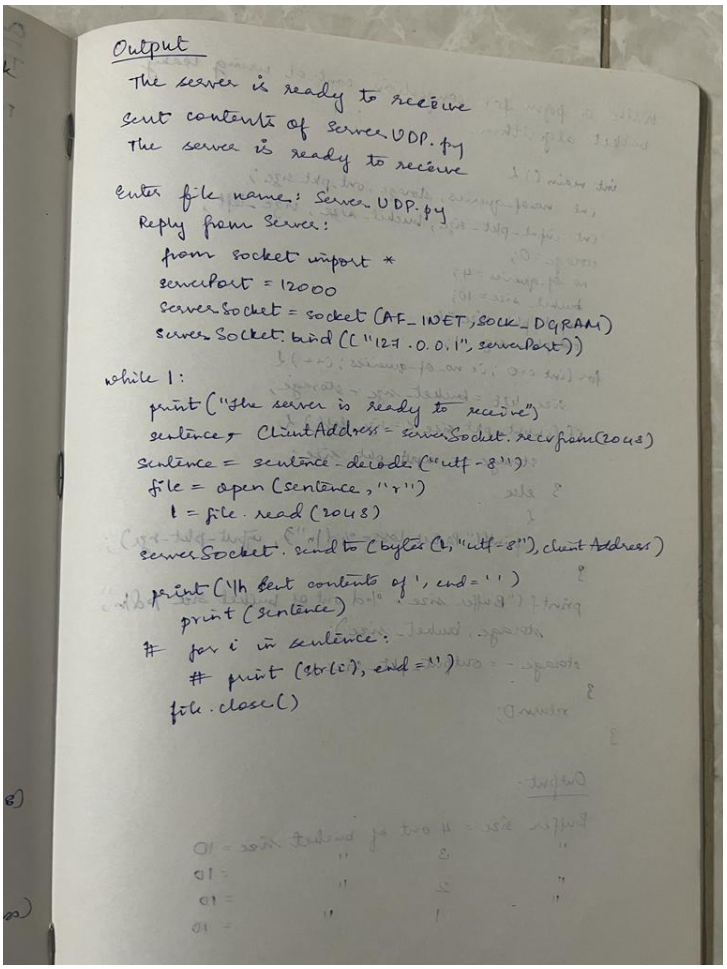
Using UDP sockets, write a client server pgm to make client sending the file name & the server to send back the contents of requested file if present.

Client UDP.py

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

Server UDP.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('\n Sent contents of', end = ' ')
    print(sentence)
    # for i in sentence:
    # print(str(i), end="")
    file.close()
```



Output

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

Enter file name: Server_UDP.py
Reply from Server:

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

while 1:
    print("The server is ready to receive")
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence,"r")
    l = file.read(2048)
    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
    print('Sent contents of ', end=' ')
    print(sentence)
    # for i in sentence:
    #     print(str(i), end="")
    file.close()
```

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
```