10/11/22

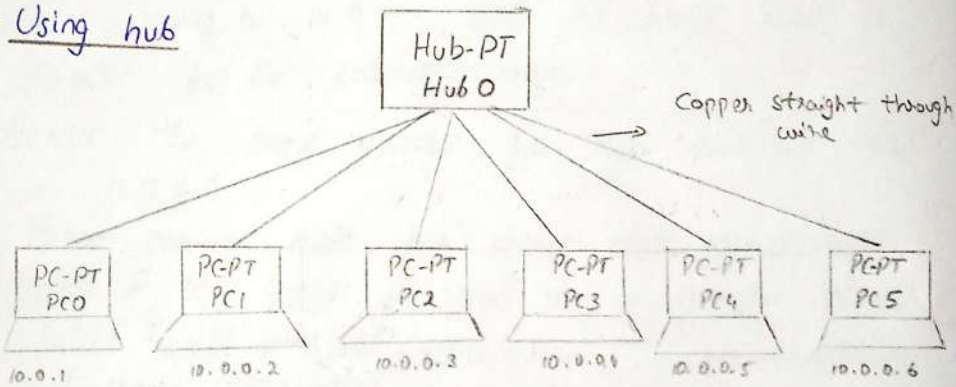
Lab 1

Hubs and Switches

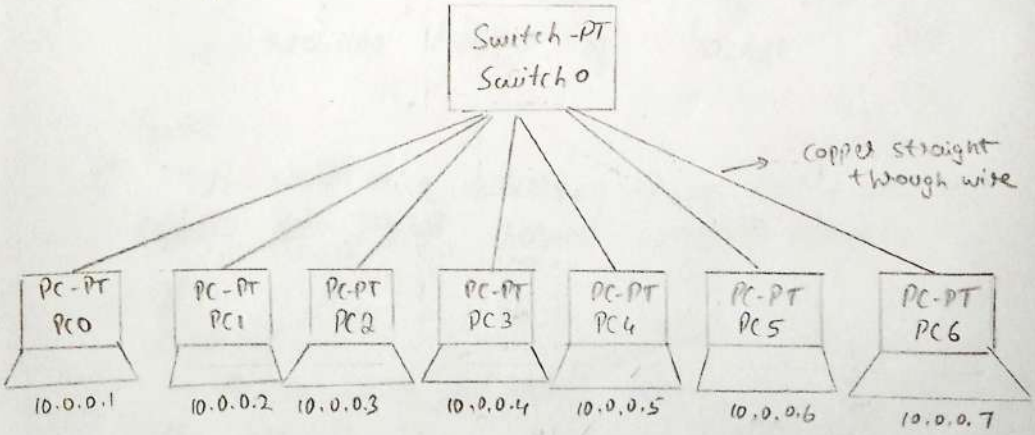
Aim: Creating a topology and simulate sending a simple PDU from source to destination using simple hub and switch as connecting devices.

Topology:

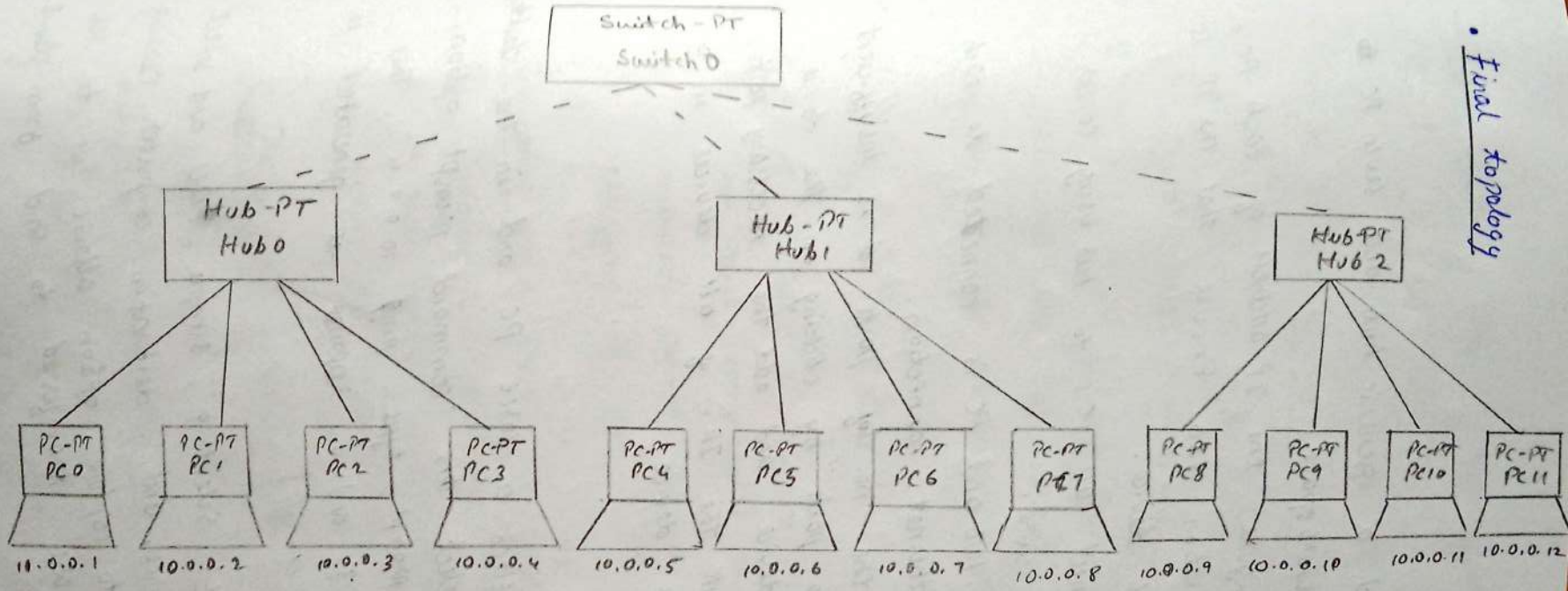
- **Using hub**



- **Using switch**



• Final topology



Procedure:

→ Using hub: • Add a generic hub and seven PC to the workspace.

• Configure the IP address of each PC's in the configuration tab. Ensure that the IP is different for each device.

• Connect all PC's to hub using copper straight through wire.

• The hub and PC is connected to each other's fast ethernet connection.

• If the no. of ports is insufficient then add extra ports by clicking on the device. Turn off the device and add the necessary ports.

• Write the IP's of all devices in a note below the device

Real time: Select a sorce PC and in the desktop tab select the command prompt option. In command prompt type "ping 10.0.0.4". This pings the PC 3 and a response is generated in PC 0.

Simulation time: Select a simple PDU and select a source and destination computer. Clicking on auto capture option allows us to see how packets are transferred to and from device.

→ Using Switch: • Add a generic switch and seven PC's to workspace.

• Configure the IP address of each PC's in the configuration tab. Ensure that IP is different for each device.

• Connect all PC's to the switch using copper straight through.

• If no. of ports are insufficient then add extra ports by clicking on device. Turn off the device and add the necessary ports.

• Write the IP's of all devices in a note below the device.

Real time: Select a source PC and in the desktop tab select command prompt option. In command prompt ping the destination PC by specifying its IP

Simulation time: Select a simple PDU and select a source and destination computer. Clicking on auto capture option allows us to see how packets are transferred.

→ Final structure (Hybrid mode)

• Add a switch, ~~and~~ 3 hubs and 12 PC's to workspace.

• Connect the three hubs to the switch and 4 PC's to each of the hubs using copper cross over and copper straight through respectively.

· Configure the IP of each of the PC in configure and add a note below each PC entering IP addr

Real time mode: Select the PC you want to send the packet from and open its command prompt, specify the destination PC by specifying its IP address. A response is sent by the destination PC to source PC.

Simulation mode: Add a simple PDU by selecting the pair of PC and click on auto capture from right panel.

Observation

HUB:

Learning outcome:

When a source sends a packet in the network the hub receives the packet and sends broadcast over the network, i.e, it sends data to all the end devices in the network and the node whose ip matches with the specified address accepts the packet and acknowledges it, remaining nodes discards / ignores the message.

The communication between hub and end devices is established through copper straight through wire as they belong to different layers.

Result

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 = 2ms

Reply from 10.0.0.3 : byte = 32 time = 0ms

Reply from 10.0.0.3 : byte = 32 time = 0 ms

Reply from 10.0.0.3 : byte = 32 time : 0ms

Ping statistic for 10.0.0.3

Packets sent = 4, recieved = 4, loss = 0

Switch

Learning outcome

When a source device sends a message sends a message to the switch once the connection is established, which takes some time called as learning time, the switch recieves the packet. It initially broadcasts the packet to all connected devices to locate the destination. Once the destination is located the message is sent only to that device.

The connection between the switch and end device is established using copper straight through as the belong to different network layer.

Result:

PC> ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data

Reply from 10.0.0.2: bytes = 32 time = 1ms

Reply from 10.0.0.2: bytes = 32 time = 3ms

Reply from 10.0.0.2: bytes = 32 time = 0ms

Reply from 10.0.0.2: bytes = 32 time = 0ms

Ping statistics for 10.0.0.2

packets sent = 4, Received = 4, lost = 0 (0% loss)

Final Structure (Hybrid model)

Learning outcome:

The switch and hub are connected through copper cross over as they belong to the same network layer but PC and hubs are connected through copper straight through as the belong to different network layer.

The message from the source PC to destination is sent through the hub which then sends to all its connected PC's and the switch. The switch then sends the message to the respective hub and the hub sends the message to all its connected PC. The destination PC acknowledges that it has received the message by sending a acknowledgement back to the source PC.

Results:

PC > ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data

Reply from 10.0.0.5: bytes = 32 time = 1ms TTL = 128

Reply from 10.0.0.5: bytes = 32 time = 1ms TTL = 128

Reply from 10.0.0.5 bytes = 32 time = 1ms TTL = 128

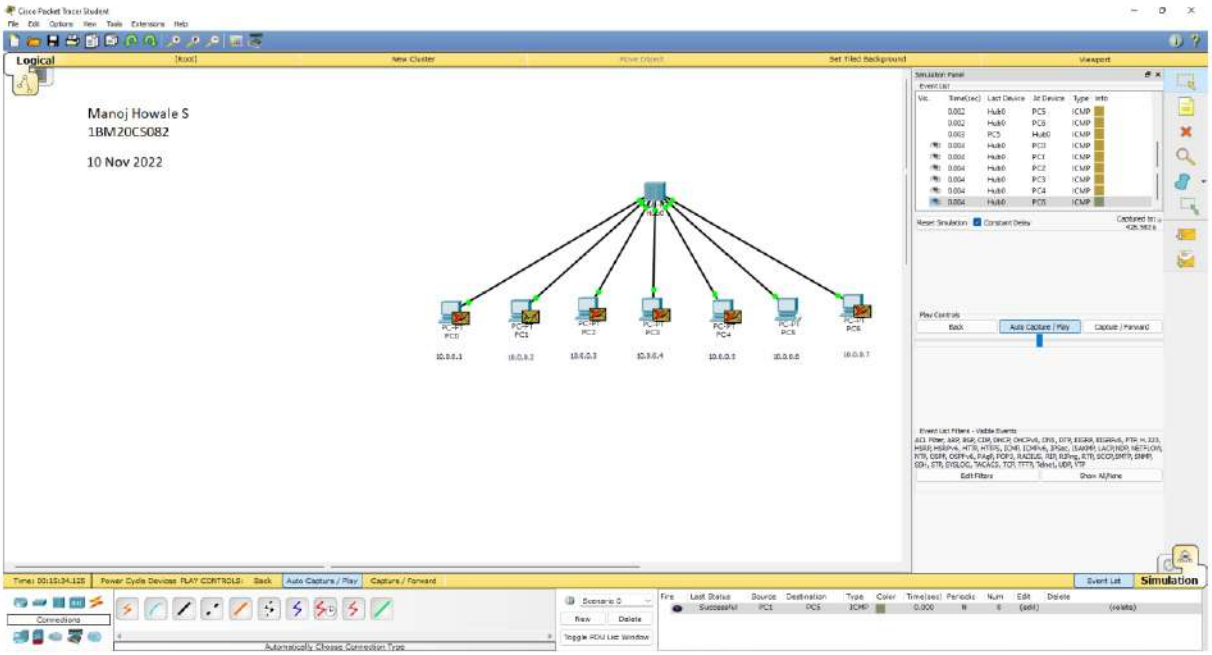
Reply from 10.0.0.5 bytes = 32 time = 1ms TTL = 128

Ping statistics for 10.0.0.7:

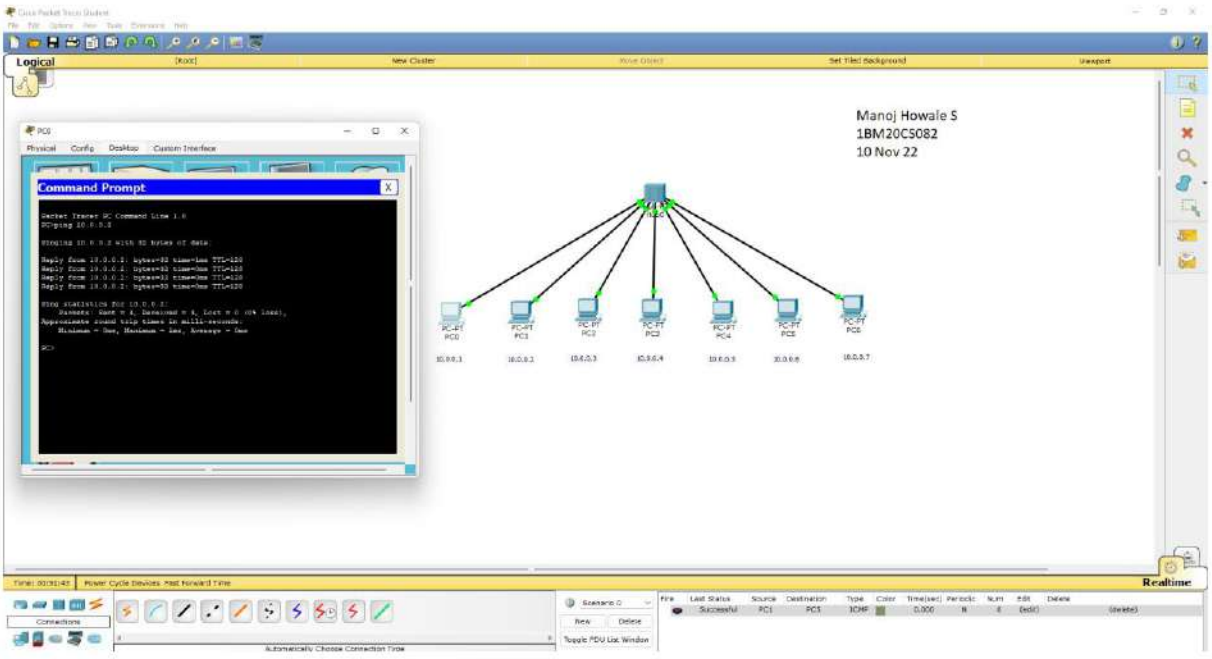
Packet: sent = 4 received = 4, lost = 0 (0% loss)

minimum = 0ms, maximum = 1ms, Average = 0ms

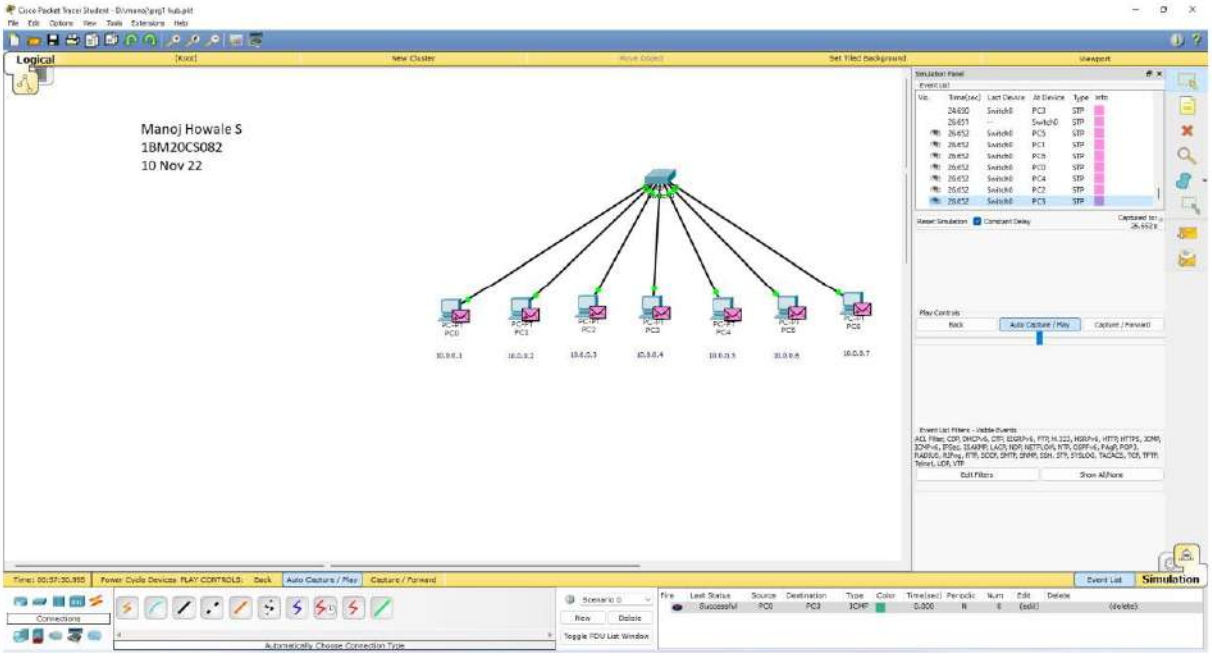
HUBS----SIMULATION



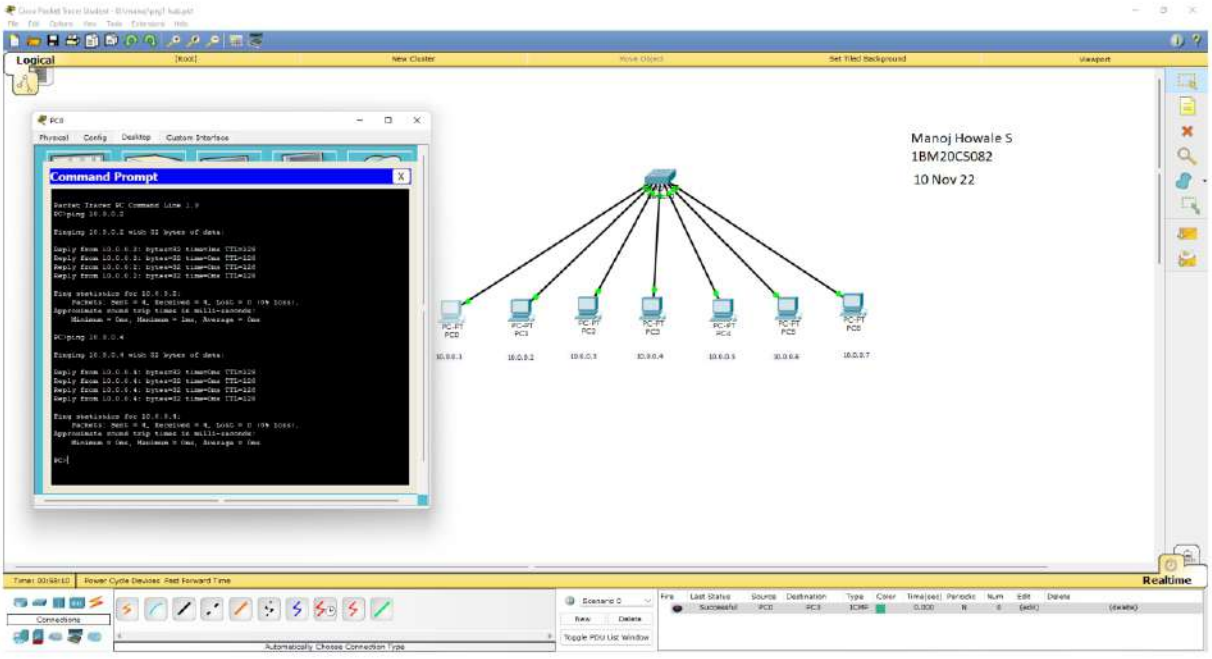
HUBS----REAL TIME



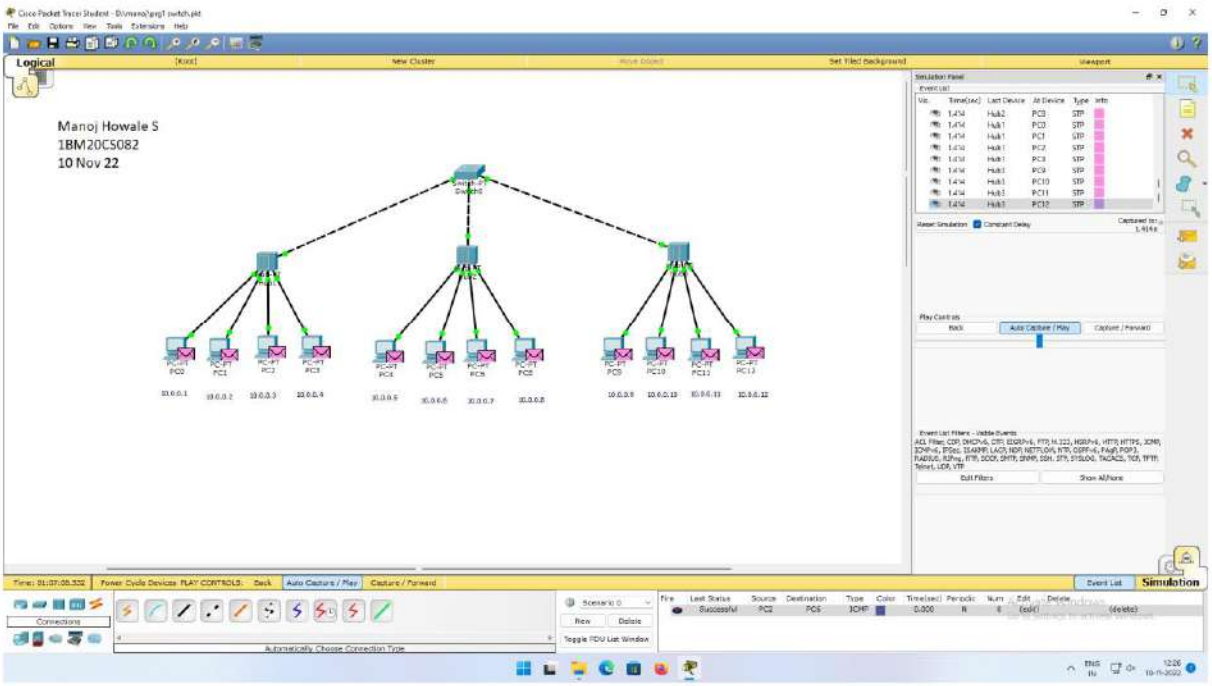
SWITCHES-----SIMULATION



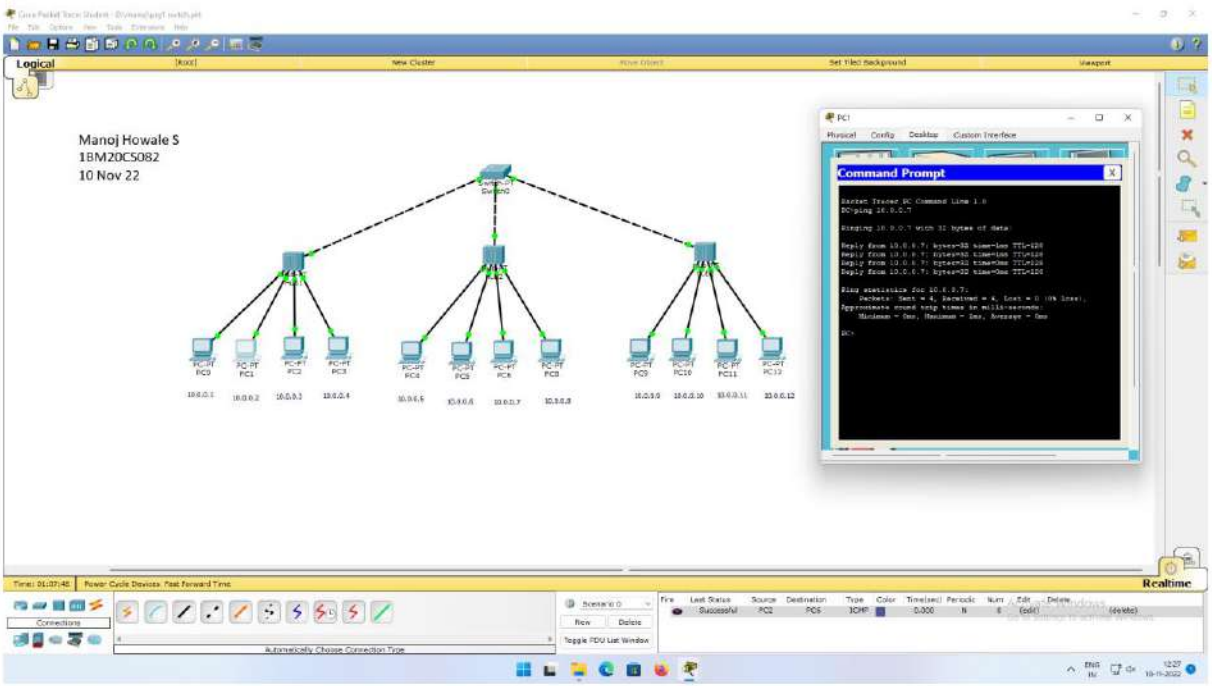
SWITCHES----REAL TIME



FINAL NETWORK-----SIMULATION



FINAL NETWORK----REAL TIME



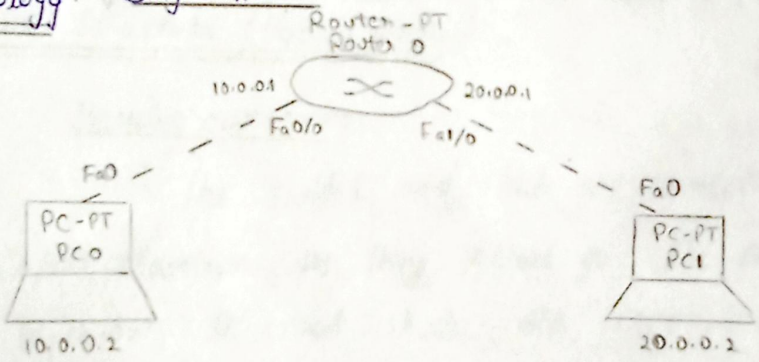
17/11/22

Lab 2

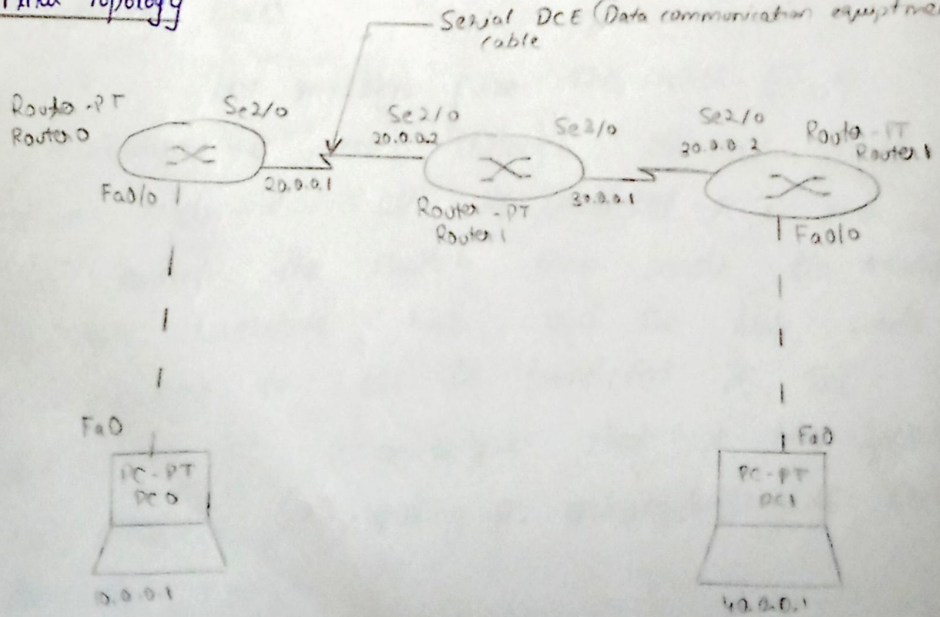
Routers

AIM: Configuring IP address to Routers in packet tracer. Explore the following messages: Ping Responses, Destination unreachable, Request time out, Reply

Topology: Single Router



Final Topology



Procedure:

Single Router

- Add a router and two PC to the workspace
- Configure the IP address of each PC as 10.0.0.2 and 20.0.0.2 respectively and gateway of each of the PC to 10.0.0.1 and 20.0.0.1 respectively. Connect the two PC's to the router using copper cross over.
- In the router go to CLI and type the following commands

```
Router> enable
Router# configure t
Router(config)# interface FastEternet 0/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface Fast Ethernet 1/0
Router(config-if)# ip address 20.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# exit
Router>
```

- After entering these command the lights between PC's and routers are turned green
- Ping PC1 from PC0 from desktop → command prompt.

Final topology :

• Add three routers and two PC's to the workspace as shown. Connect the router and PC using a copper cross over cable and two routers using a Serial DCE cable.

• Configure the IP address and gateway of both PC's as 10.0.0.1, 10.0.0.10 and 40.0.0.1, 40.0.0.10 respectively.

• In the router0 go to the CLI and type the commands

```
Router > enable
Router# configure t
Router (config)# interface FastEthernet 0/0
Router (config-if)# ip address 10.0.0.10 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit
Router (config)# interface Serial 2/0
Router (config-if)# ip address 20.0.0.1 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit
Router (config)# exit
Router# exit
Router >
```

• Configure router2 similarly as router 0 with IP's of Fa0/0 as 40.0.0.10 and Se2/0 as 30.0.0.2.

• Configure router1 in CLI with both interface as Se 2/0 and Se 3/0 with IP's 20.0.0.2 and 30.0.0.1

• After performing all these commands all the lights are turned green indicating the circuit is complete and connected.

• The next hop of all the routers need to be configured to complete the connection

In the CLI of Router 0

```
Router (config) # ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router (config)# ip router 40.0.0.0 255.0.0.0 20.0.0.2
Router (config)# exit
```

In the CLI of router 1

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

In the CLI of router 2

```
Router (config) # ip route 10.0.0.0 255.0.0.0 30.0.0.1
Router (config) # ip route 20.0.0.0 255.0.0.0 30.0.0.1
```

• Ping PC0 to PC1 from desktop → command prompt

Observation

Single Router

Learning Outcome: After all the connections are made to the router the lights are red till the router is configured in the CLI.

When PC0 pings PC1 for the first time we get the first packet as request timed out. When pinging for the second time all four packet are recieved by PC1. If we reverse ping from PC1 to PC0 all four packets are recieved as the router has learnt the address of its connected nodes.

Result: Ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data

Request timed out

Reply from 20.0.0.1 : bytes = 32 time<1ms TTL=127

Reply from 20.0.0.1 : bytes = 32 time<1ms TTL=127

Reply from 20.0.0.1 : bytes = 32 time<1ms TTL=127

Ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1 : bytes=32 time<1ms TTL=127

Reply from 20.0.0.1 : bytes=32 time<1ms TTL=127

Reply from 20.0.0.1 : bytes=32 time<1ms TTL=127

Reply from 20.0.0.1 : bytes=32 time<1ms TTL=127

Ping statistics for 20.0.0.1

packets : sent = 4 , recieved = 4, lost = 0 (0% loss)

Final Configuration

Observation : The routers even though being configured in CLI would only know the address of the nodes or routers that are directly connected to. To send a packet to a network that is not connected directly to it requires the need to configure its ip route. After providing the address of next hop of every networks to each router the packets are transferred smoothly.

Result : Ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1 : bytes = 32 time < 1 ms TTL = 127

Reply from 40.0.0.1 : bytes = 32 time < 1 ms TTL = 127

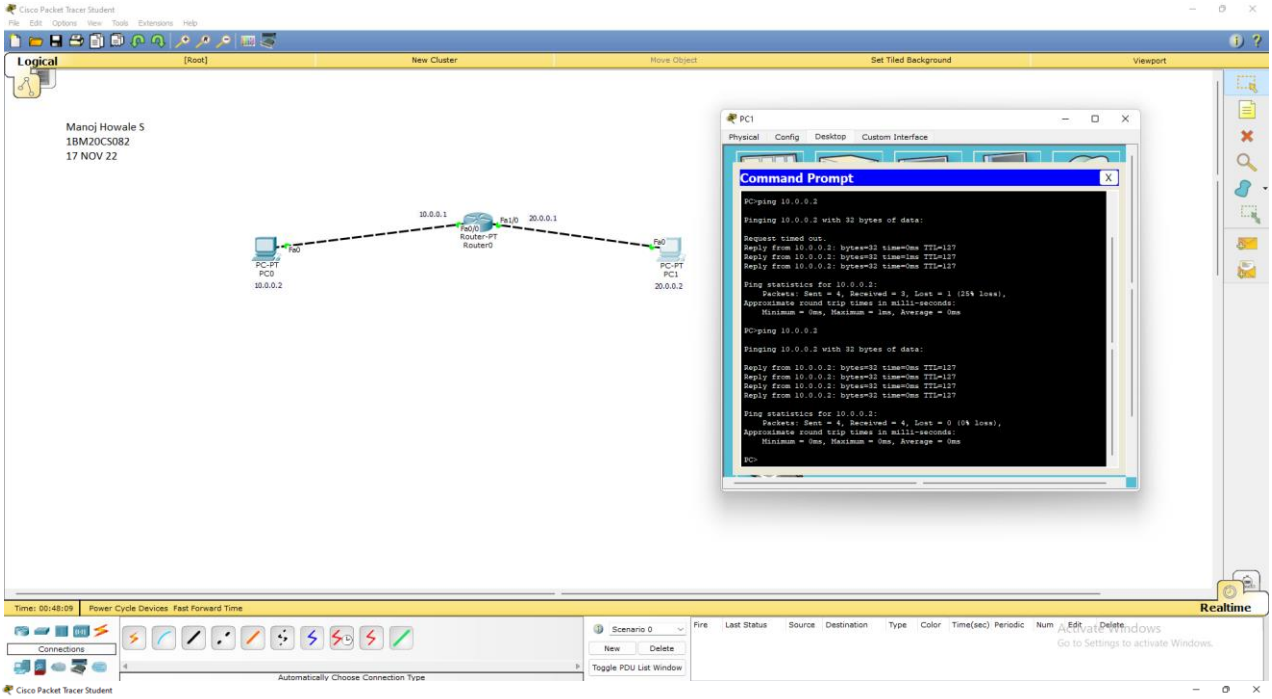
Reply from 40.0.0.1 : bytes = 32 time < 1 ms TTL = 127

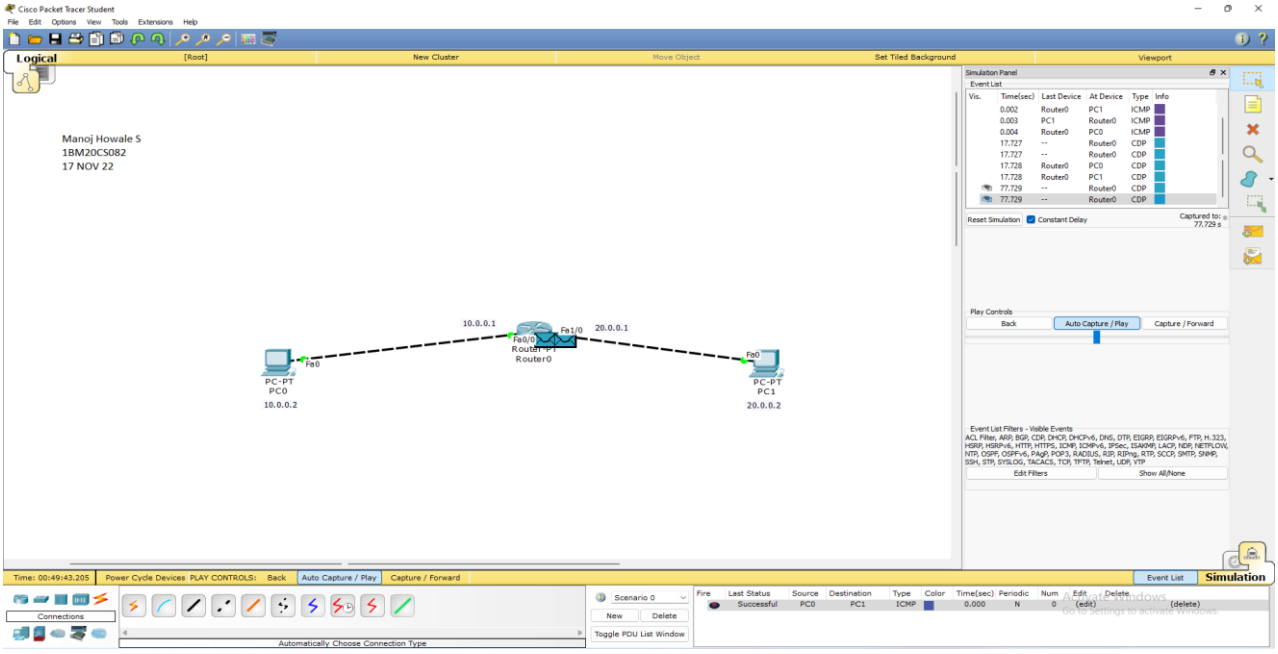
Reply from 40.0.0.1 : bytes = 32 time < 1 ms TTL = 127

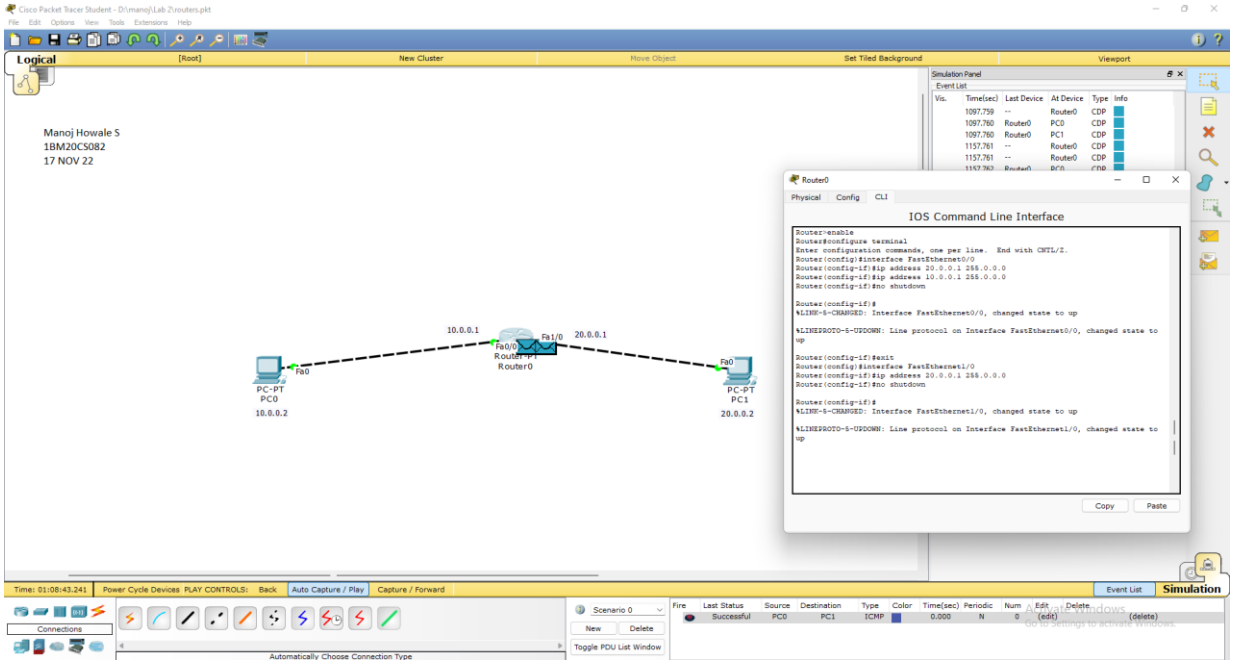
Ping statistics for 40.0.0.1

Packets : sent = 4 , recieved = 4 , lost = 0 (0% loss)

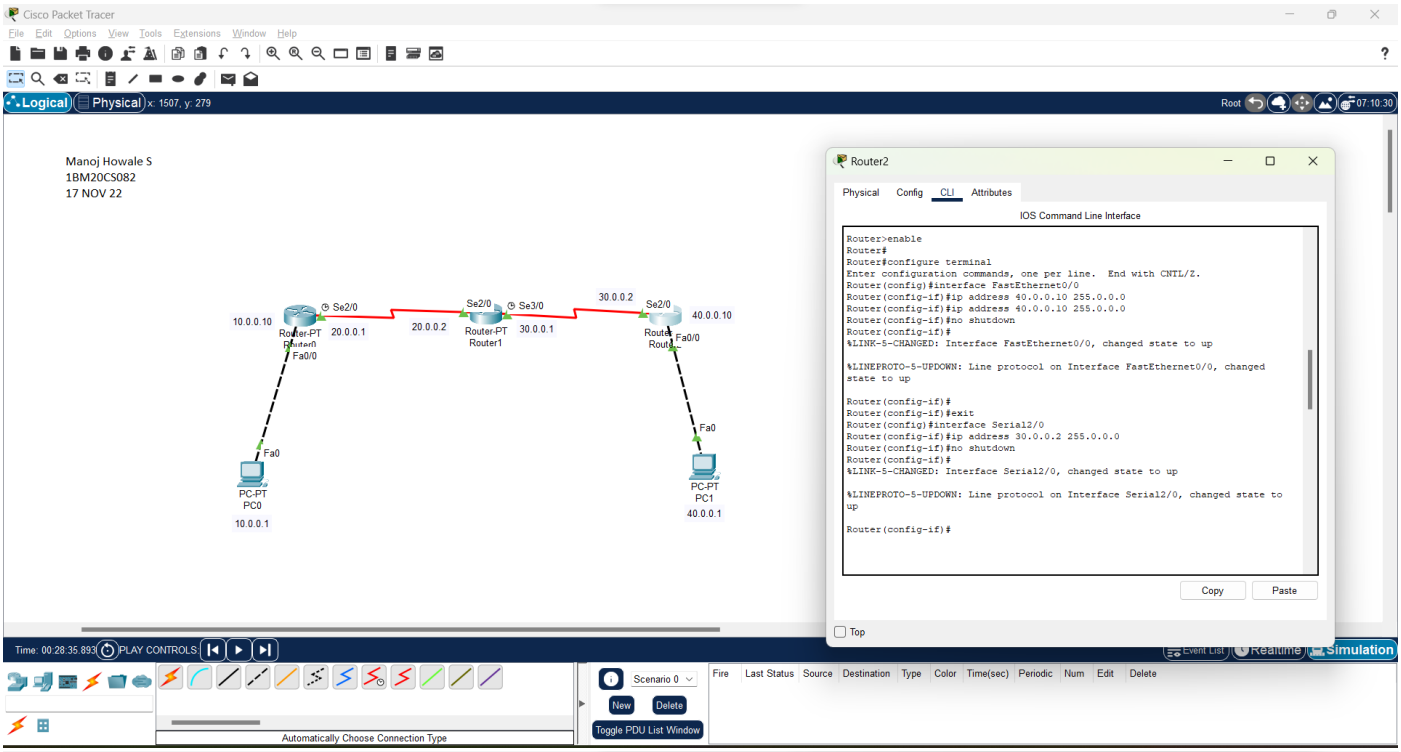
SINGLE ROUTER—REAL TIME

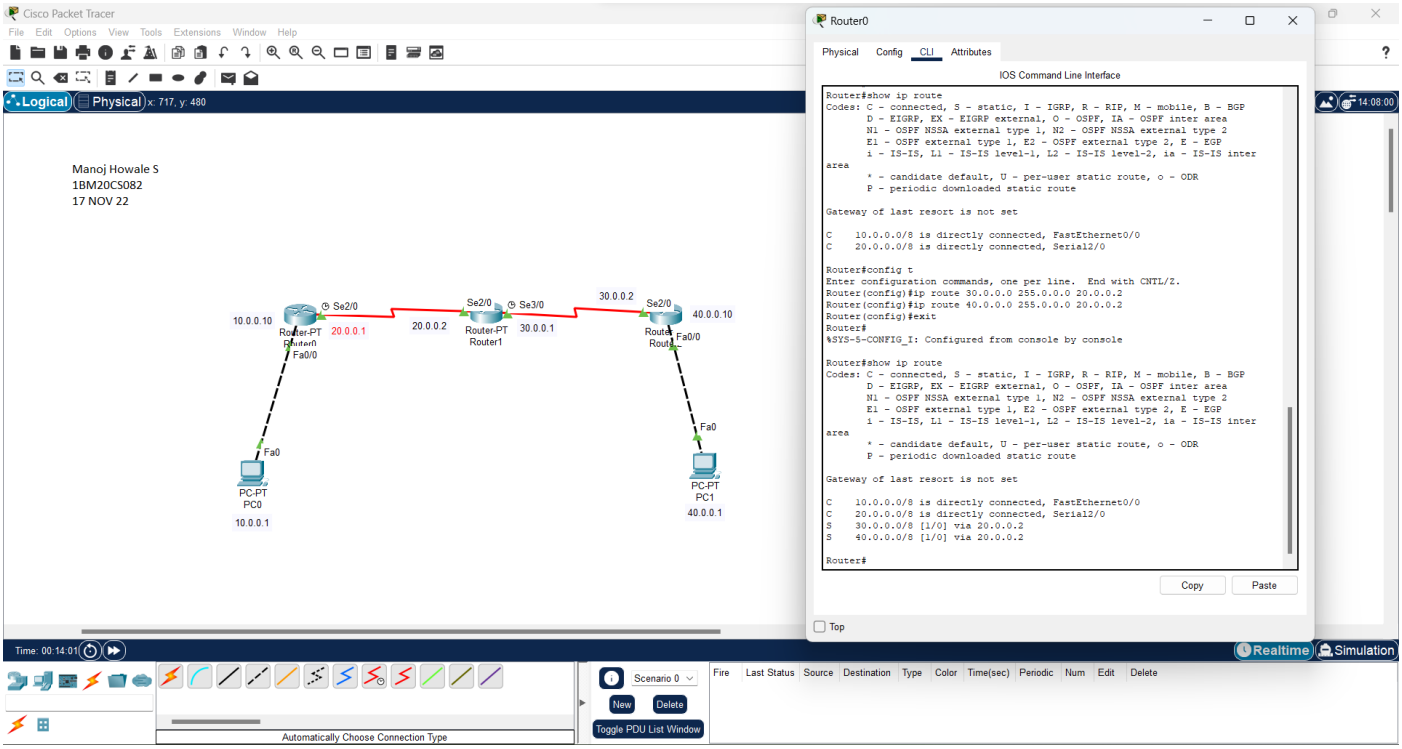


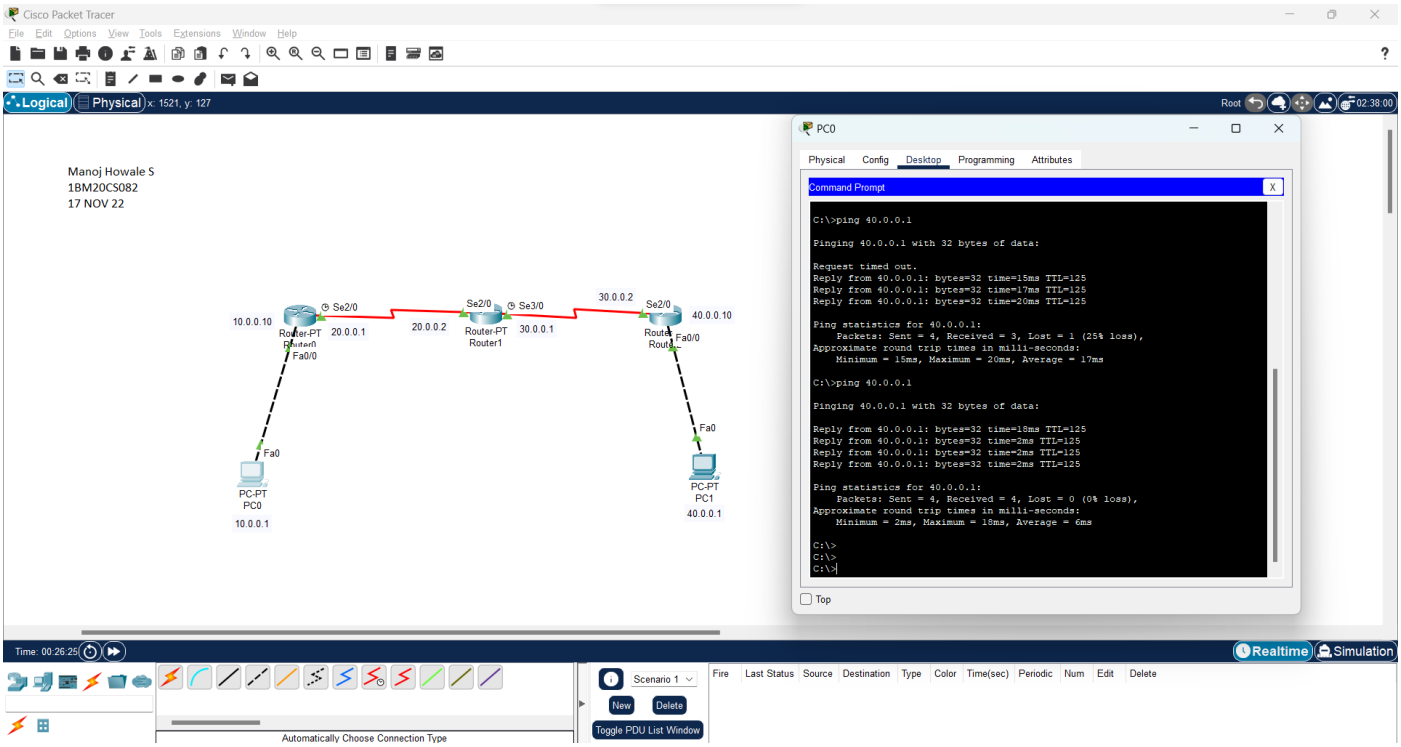


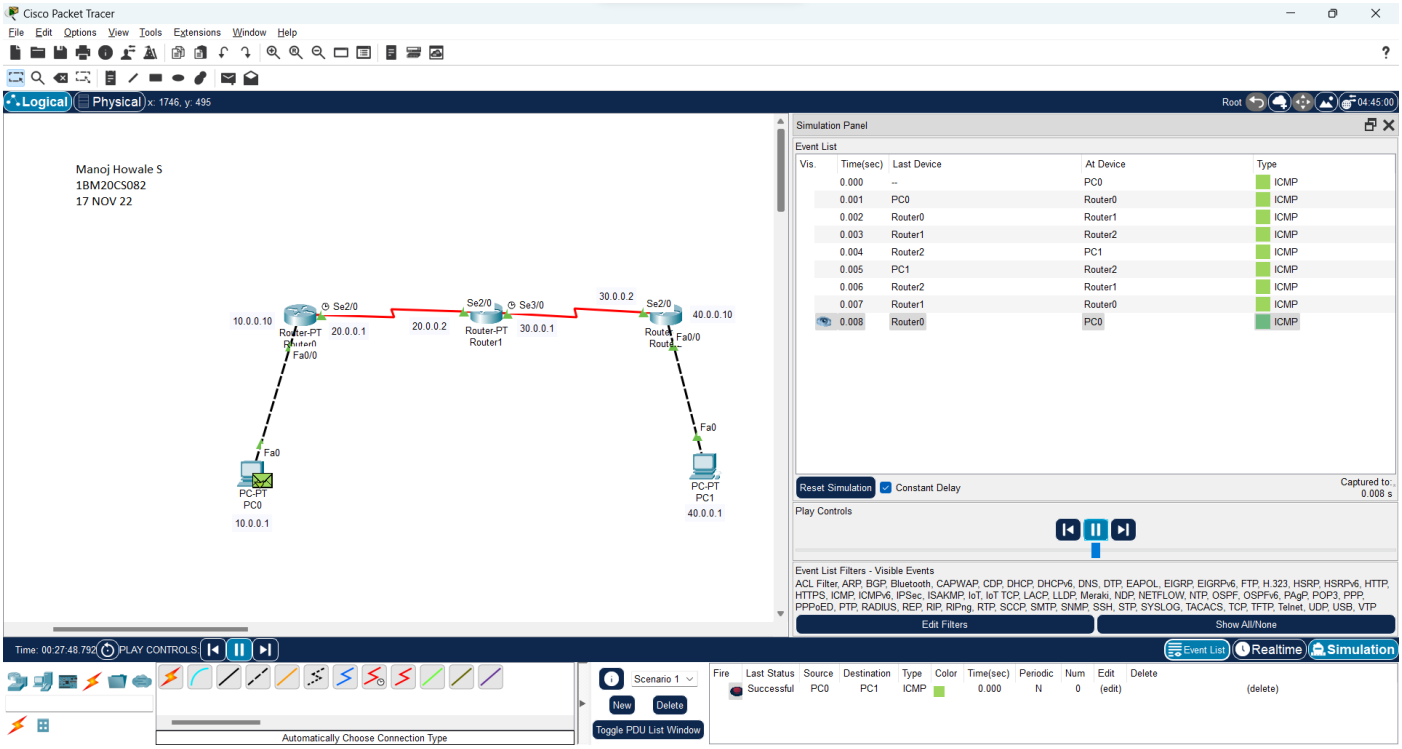


ROUTERS—REAL TIME






Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical Physical x: 1521, y: 127
Root 02:38:00
Manoj Howale S
1BM20CS082
17 NOV 22
10.0.0.10
Se2/0
Router-PT
Router0
Fa0/0
20.0.0.1
Se2/0
Se3/0
20.0.0.2
Router-PT
Router1
30.0.0.1
30.0.0.2
Se2/0
Router
Router2
Fa0/0
40.0.0.10
Fa0
PC-PT
PC0
10.0.0.1
Fa0
PC-PT
PC1
40.0.0.1
PC0
Physical Config Desktop Programming Attributes
Command Prompt
C:\>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=15ms TTL=125
Reply from 40.0.0.1: bytes=32 time=17ms TTL=125
Reply from 40.0.0.1: bytes=32 time=20ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 15ms, Maximum = 20ms, Average = 17ms
C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 40.0.0.1: bytes=32 time=18ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 18ms, Average = 6ms
C:\>
C:\>
C:\>
Top
Time: 00:26:25
Realtime Simulation
Scenario 1
New Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete
Automatically Choose Connection Type


Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical Physical x: 1746, y: 495
Root 04:45:00
Manoj Howale S
1BM20CS082
17 NOV 22
10.0.0.10
Se2/0
Router-PT
Router0
Fa0/0
20.0.0.1
Se2/0
Se3/0
20.0.0.2
Router-PT
Router1
30.0.0.1
30.0.0.2
Se2/0
Router
Router2
Fa0/0
40.0.0.10
Fa0
PC-PT
PC0
10.0.0.1
Fa0
PC-PT
PC1
40.0.0.1
Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type
0.000 -- PC0 ICMP
0.001 PC0 Router0 ICMP
0.002 Router0 Router1 ICMP
0.003 Router1 Router2 ICMP
0.004 Router2 PC1 ICMP
0.005 PC1 Router2 ICMP
0.006 Router2 Router1 ICMP
0.007 Router1 Router0 ICMP
0.008 Router0 PC0 ICMP
Reset Simulation
Constant Delay
Captured to:
0.008 s
Play Controls
Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP
Edit Filters
Show All/None
Time: 00:27:48.792
PLAY CONTROLS:
Event List
Realtime Simulation
Scenario 1
New Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete
Successful PC0 PC1 ICMP 0.000 N 0 (edit) (delete)
Automatically Choose Connection Type

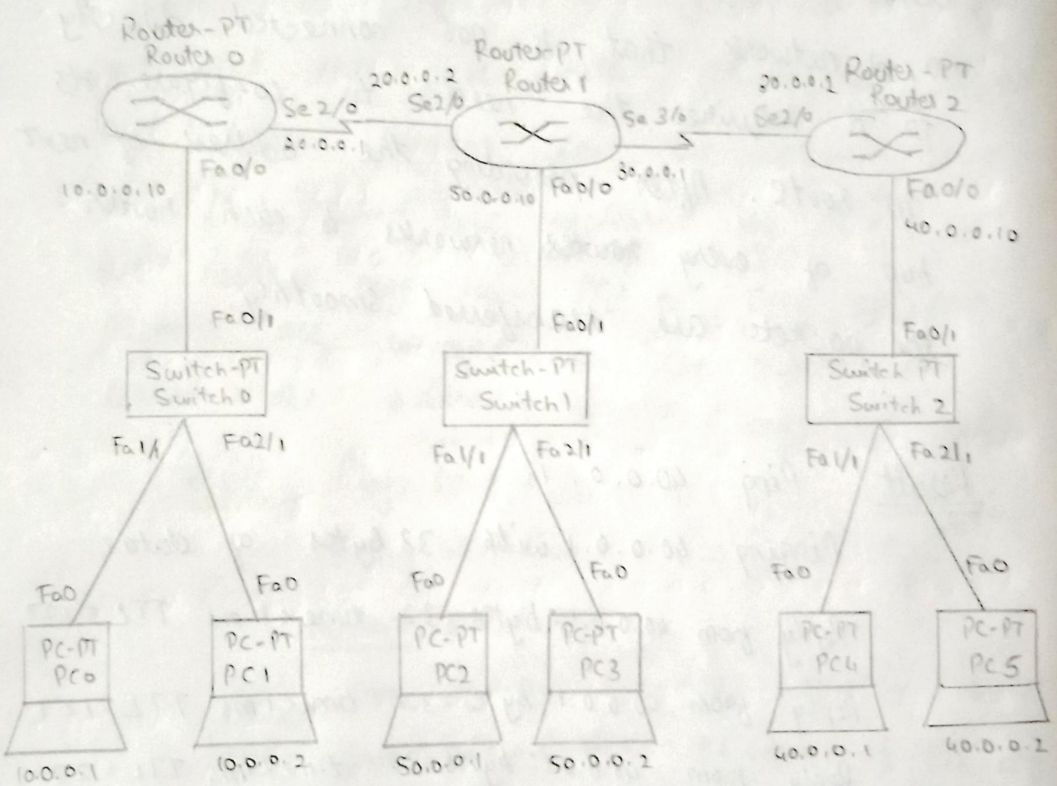
24/11/22

Lab 3

Default Route

Aim: Configuring default route to the routers

Topology:



Procedure:

- Add three routers, three switches and six PC's to the workspace.
- Configure the IP's of each PC's connected under the three switches under the network Id's of 10.0.0.0, 50.0.0.0, 40.0.0.0. All connections between PC - Switch and Switch - Router are made

using copper straight through. The connections between two routers are made using copper serial DCE.

- In the router go to CLI and type the commands

```
Router> enable
Router# configure t
Router(config)# interface Fa 0/0
Router(config-if)# ip address 10.0.0.10 25.0.0.0
Router(config-if)# no shut
Router(config-if)# exit
Router(config)# interface Serial 2/0
Router(config-if)# ip address 20.0.0.1 255.0.0.0
Router(config-if)# no shut
Router(config-if)# exit
Router(config)# exit
Router# exit
Router>
```

- Configure Router 2 and router 1 similarly as router 0.

- After performing all the operations on the routers the lights are all turned green. indicating a complete connection.

- Now ping a PC from that has a different network id compared to the current PC. The results are as shown.

C:\> ping 40.0.0.2

pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: Destination host unreachable
Reply from 40.0.0.2: Destination host unreachable
Reply from 40.0.0.2: Destination host unreachable
Reply from 40.0.0.2: Destination host unreachable.

- To remove this error the static routing of all the routers need to be configured.

- In the CLI of Router 0 set a default route to Router 1

Router (config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2

- In the CLI of Router 2 set a default route to Router 1

Router (config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1

- In the CLI of Router 1 set the next hop of the two network address.

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

- If we repeate the same pinging between two PC's the request is serviced.

Observation:

The routers even though being configured in CLI would only know about the neighbouring routers that it is directly connected to. To send a packet to a network that the router is not directly connected it requires configuration of static routing. After providing default routing and static routing to every network the packets are sent successfully. The packet when pinged for the first time are given a response request timed out.

Result:

PC> Ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Request timed out

Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

PC> Ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

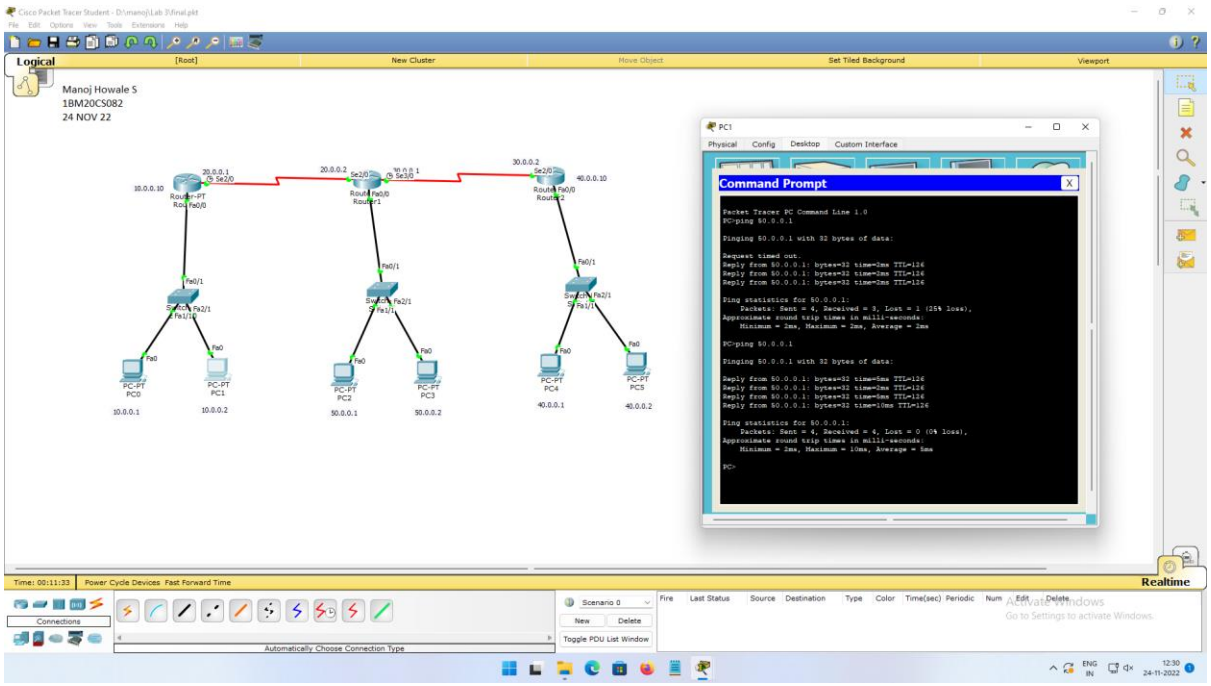
Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

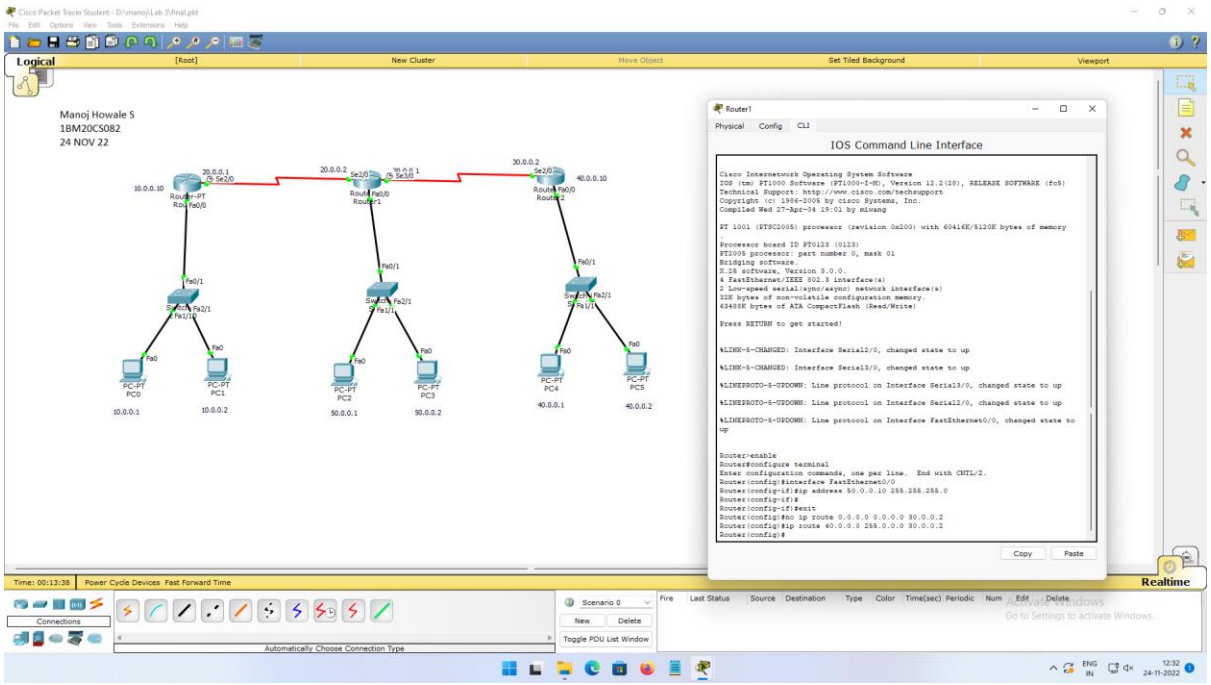
Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

Reply from 40.0.0.1: bytes = 32 time < 1ms TTL = 127

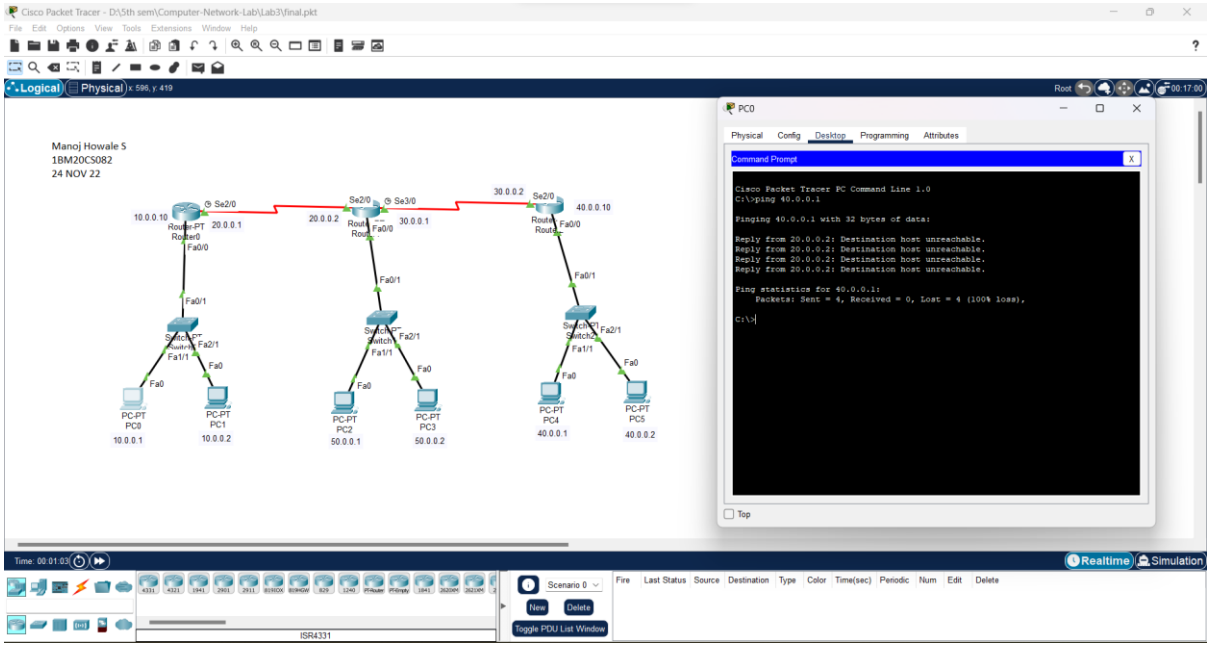
Final Topology -- Realtime



Final Topology – IP route



Final Topology – Destination Host Unrechable



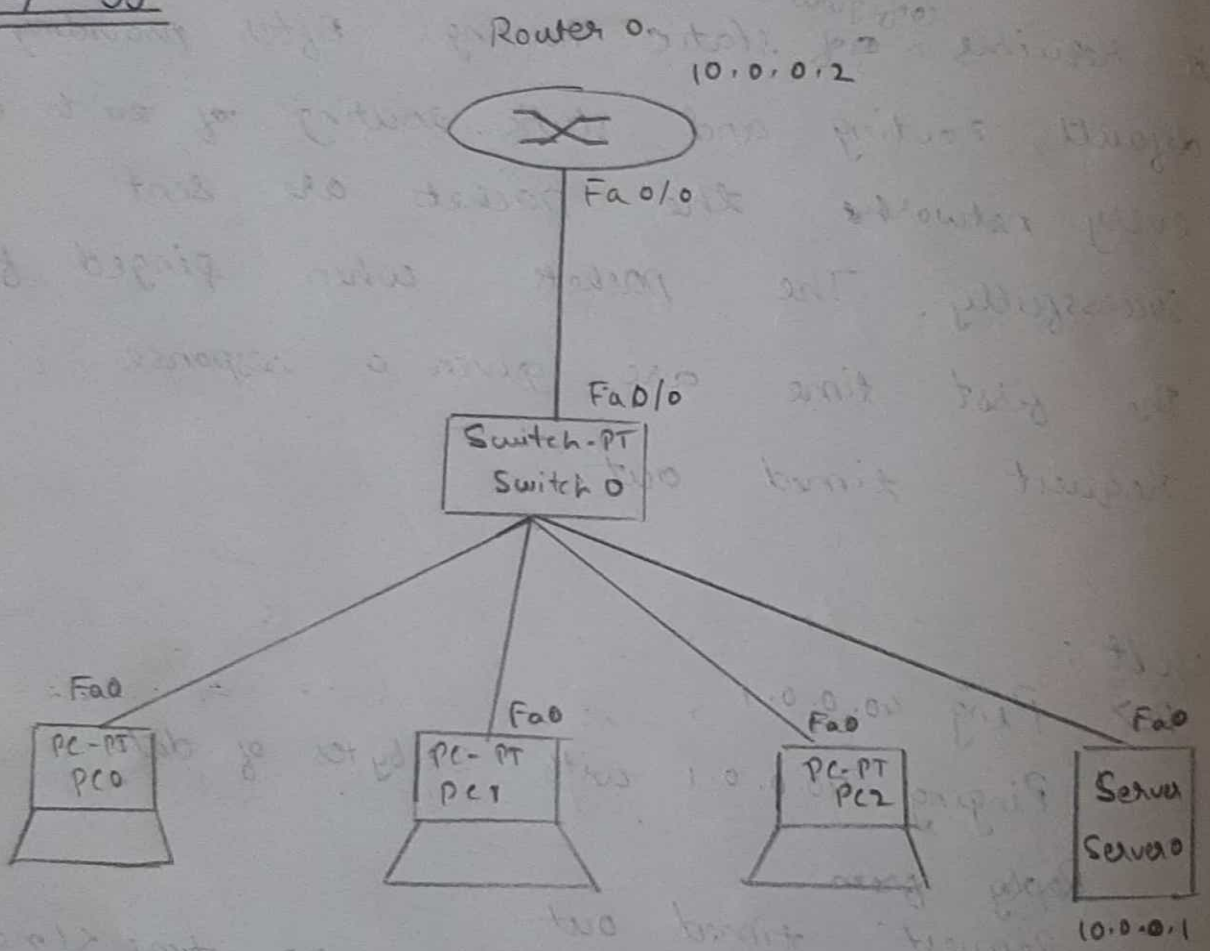
6/12/22

Lab 4

DHCP

Aim: Configuring DHCP within LAN in a Packet tracer

Topology:



Procedure:

- Add 3 PC's, 1 server, 1 switch and 1 router to workspace. Connect PC, server to switch by using copper straight-through wire.
- Configure the server by giving the IP address and gateway.
- Now go to router and open CLI and follow the commands

```
Router > enable
Router # config t
Router (config) # interface FastEthernet 1/0
Router (config-if) # ip address 10.0.0.2 255.0.0.0
Router (config-if) # no shut
Router (config-if) # exit
Router (config) # exit
Router # exit
Router >
```

- Now go to services in server 0 and go to DHCP. Change service from off to on. Now give start IP address 10.0.0.3 and give all servers as 10.0.0.1
- Now open IP configuration in desktop change IP configuration from static to DHCP. Follow the same for all other PC's
- Now ping can be performed.

Learning Outcome:

Server automatically provides IP address for all the PC's Follows below procedure

D - Discover
O - Offer
R - Request
A - Acknowledgement

Result: Ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data

Reply from 10.0.0.5 : bytes = 32 time = 0ms TTL = 12
Reply from 10.0.0.5 : bytes = 32 time = 0ms TTL = 12
Reply from 10.0.0.5 : bytes = 32 time = 0ms TTL = 12
Reply from 10.0.0.5 : bytes = 32 time = 0ms TTL = 12

Ping Statistics for 10.0.0.5

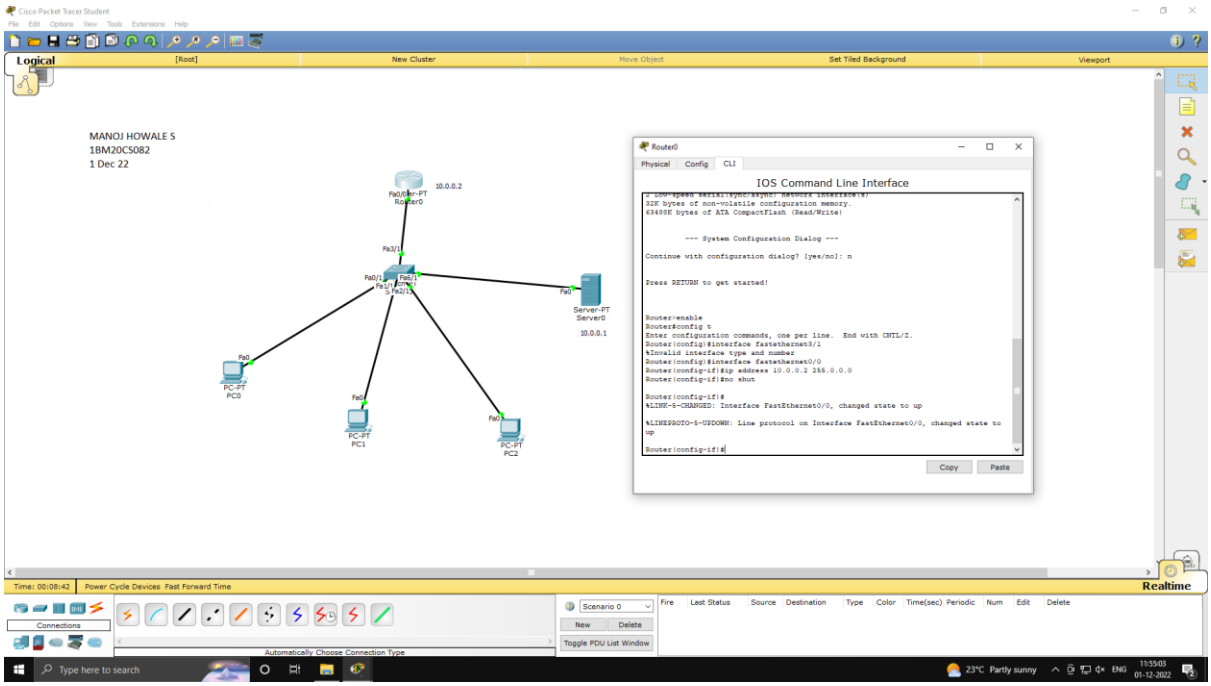
Packets : Sent = 4 , Received = 4, Lost = 0 (0% loss)

Approximate round trip time in milliseconds :

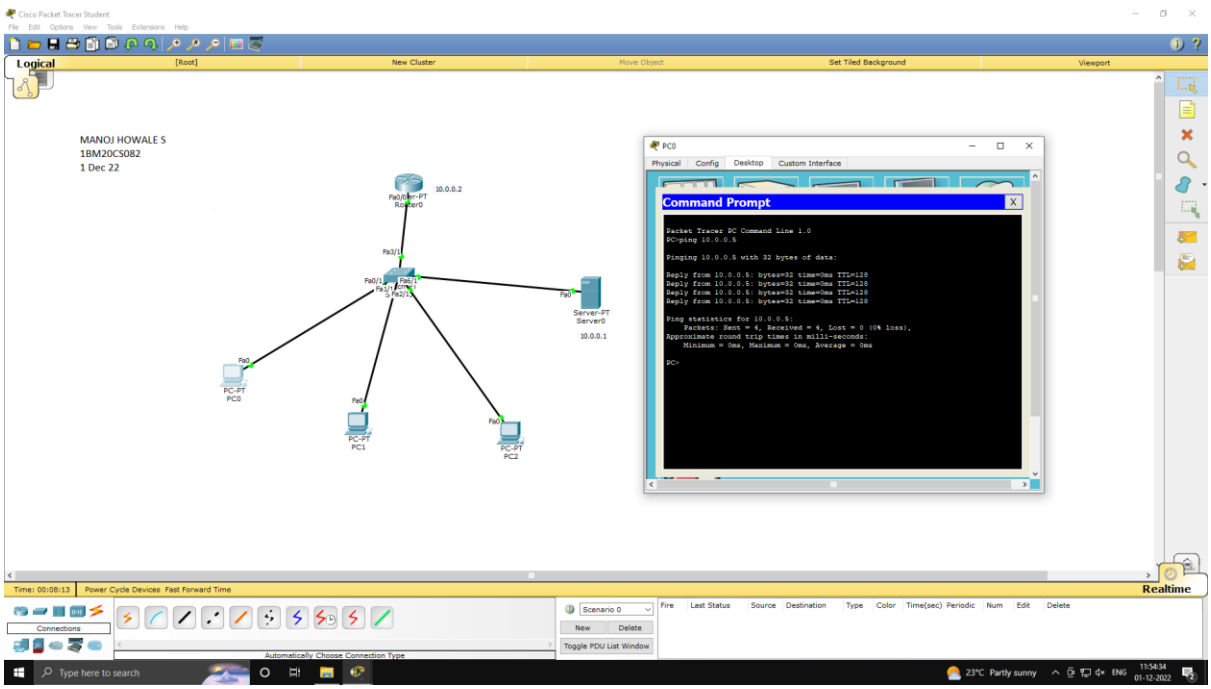
Minimum = 0 ms , Maximum = 0 ms , Average = 0ms

8/12/22

DHCP CLI



DHCP SIMULATION



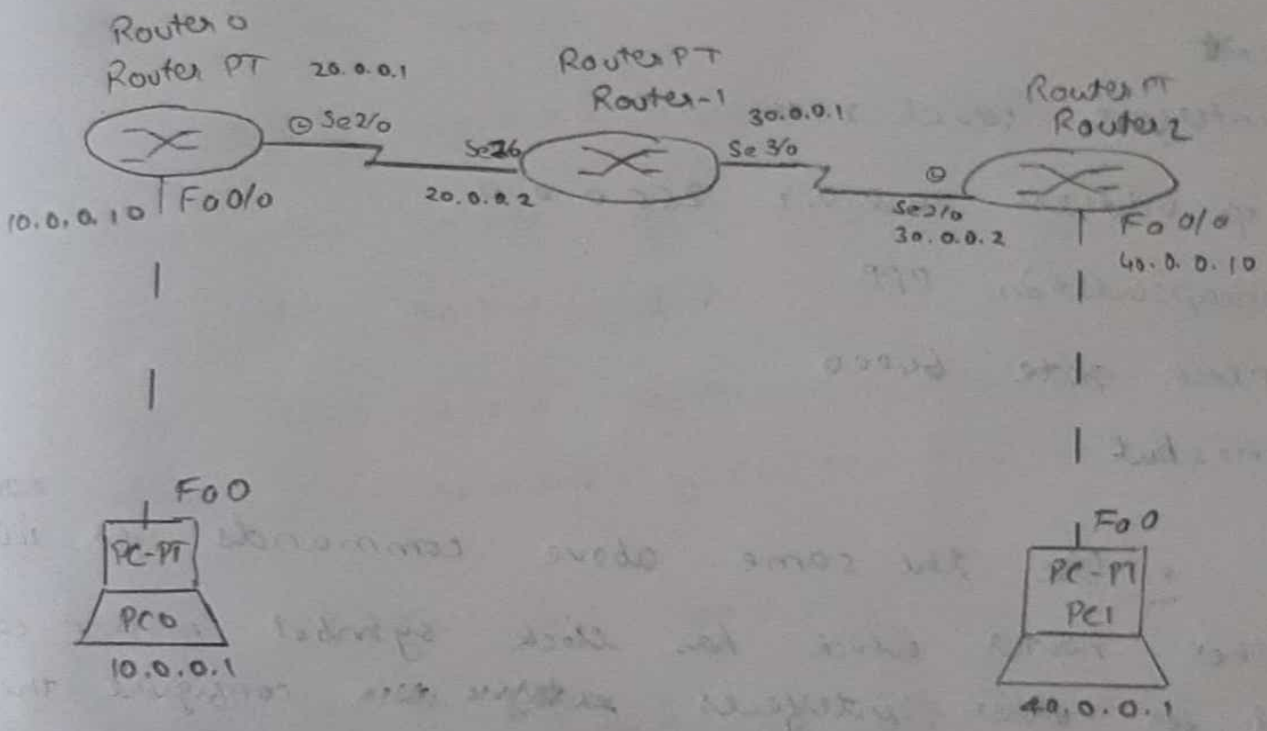
11/2/22

Lab 5

RIP

Aim: Configuring RIP routing Protocal in Routers

Topology:



Procedure:

- Place 3 generic routers, 2 generic PC's and place notes to indicate respective ip addresses.
- Use copper cross over to connect between Router and PC and serial DCE to connet between two routers.
- Set IP address, Subnet and Gateway of the PC's as 10.0.0.1, 255.0.0.0, 10.0.0.10 and 40.0.0.1, 255.0.0.0, 40.0.0.10 respectively.

• For interfacing PC0 and routers 3 and serial DCE of router 3 enter the commands in CLI

> enable
> config-t
> interface fastethernet 0/0
> no shut
> exit
> interface serial 2/0
> ip address 20.0.0.1 255.0.0.0
> encapsulation PPP
> clock rate 64000
> no shut

• Use the same above commands for interfacing another router which has clock symbol in DCE cable and for other interfaces configure the router by entering the same commands except for RIP "clock rate 64000" command.

• Set the RIP protocal when the lights turn green.

> Router RIP
> network 10.0.0.0
> network 20.0.0.0
> exit

• Repeat the commands to all routers with IP address that the router is directly connected to.

Observation:

Learning Outcome:

Instead of using static IP routing for all routers by using RIP protocal routing becomes easy when large number of routers are present.

Result: Ping 40.0.0.1

pinging 40.0.0.1 with 32 Bytes of data:

Reply from 40.0.0.1 : bytes = 32 time = 14ms TTL = 125

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

Reply from 40.0.0.1 : bytes = 32 time = 14ms TTL = 125

Reply from 40.0.0.1 : bytes = 32 time = 12 ms TTL = 125

Ping statistics for 40.0.0.1:

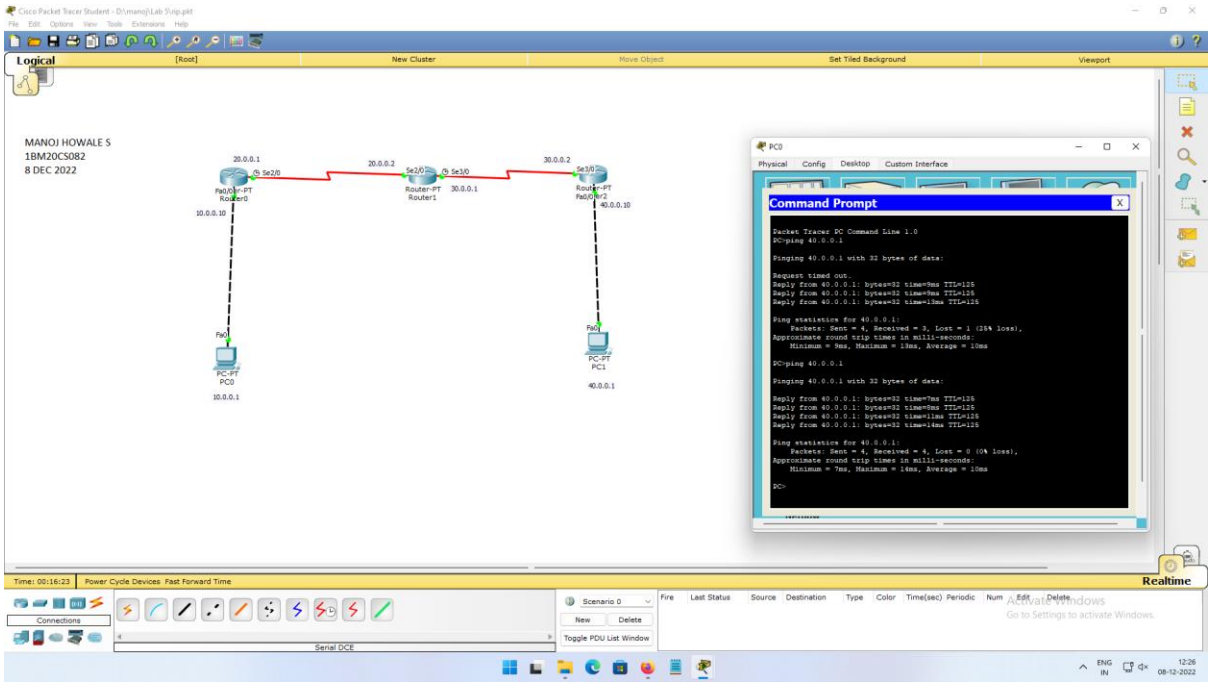
Packets : sent = 4 , Received = 4, Lost = 0 (0% loss)

Approximate round trip times in milliseconds:

minimum = 2ms, maximum = 14 ms, Average = 10 ms

Valid
29-12-2022

RIP OUTPUT


MANOJ HOWALE S
1BM20CS082
8 DEC 2022
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=9ms TTL=125
Reply from 40.0.0.1: bytes=32 time=9ms TTL=125
Reply from 40.0.0.1: bytes=32 time=13ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 9ms, Maximum = 13ms, Average = 10ms
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=7ms TTL=125
Reply from 40.0.0.1: bytes=32 time=8ms TTL=125
Reply from 40.0.0.1: bytes=32 time=13ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 7ms, Maximum = 14ms, Average = 10ms
PC>

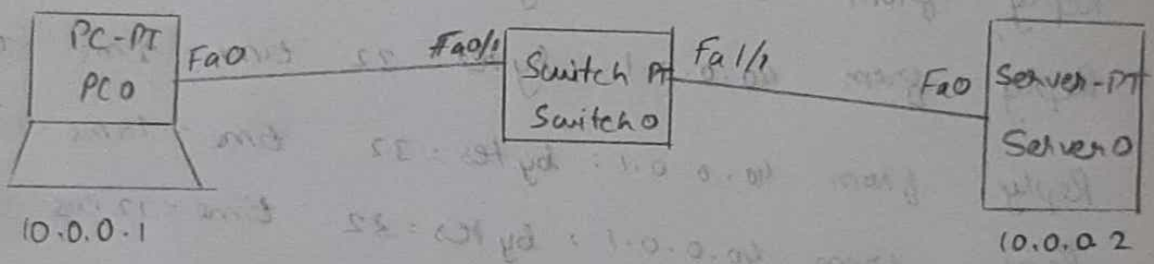
15/12/22

Lab 6

Web Server

Aim: Demonstration of WEB server and DNS using Packet tracer.

Topology:



Procedure:

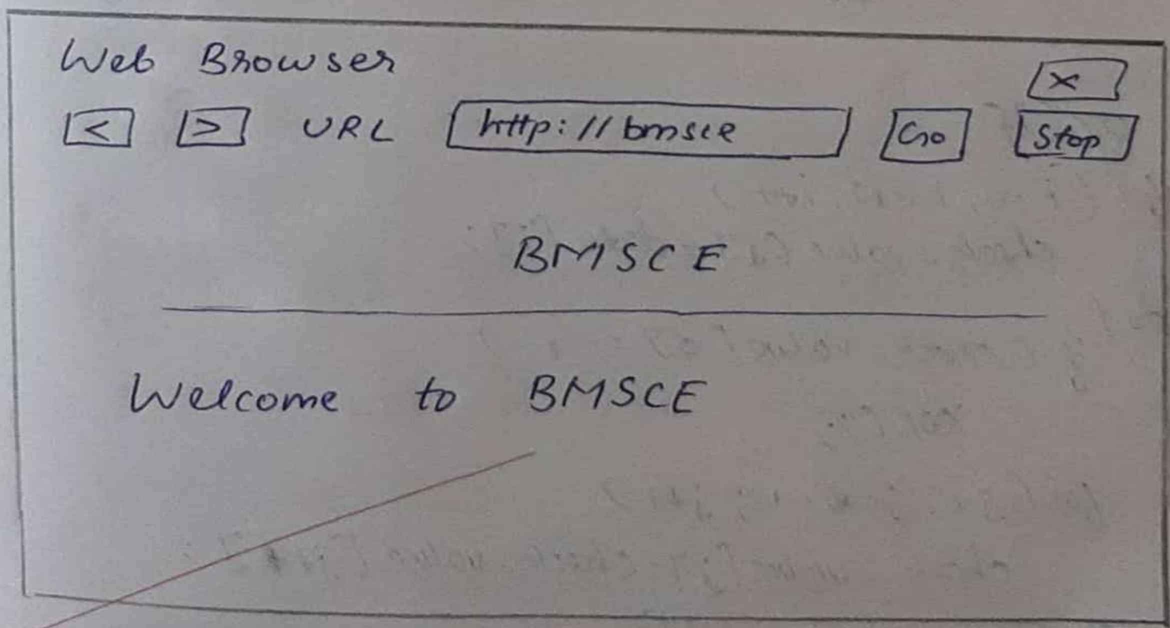
- Add the PC, switch and server in the workspace and connect the components as shown. using copper straight through.
- Configure the IP address of PC as 10.0.0.1 and server as 10.0.0.2
- Now go to the server and in the services tab go to DNS and turn it on.
- Go to HTTP and turn on both http and https. Edit the html files.
- In desktop option go to webservices and enter the IP address of the server.

- Now go to DNS in services tab add a domain name and the ip address of the server in address. Click on add and then save.
- In the desktop option of PC go to webservices and search the domain name entered.
- Now create a cv page and link it to the current website.

Observation:

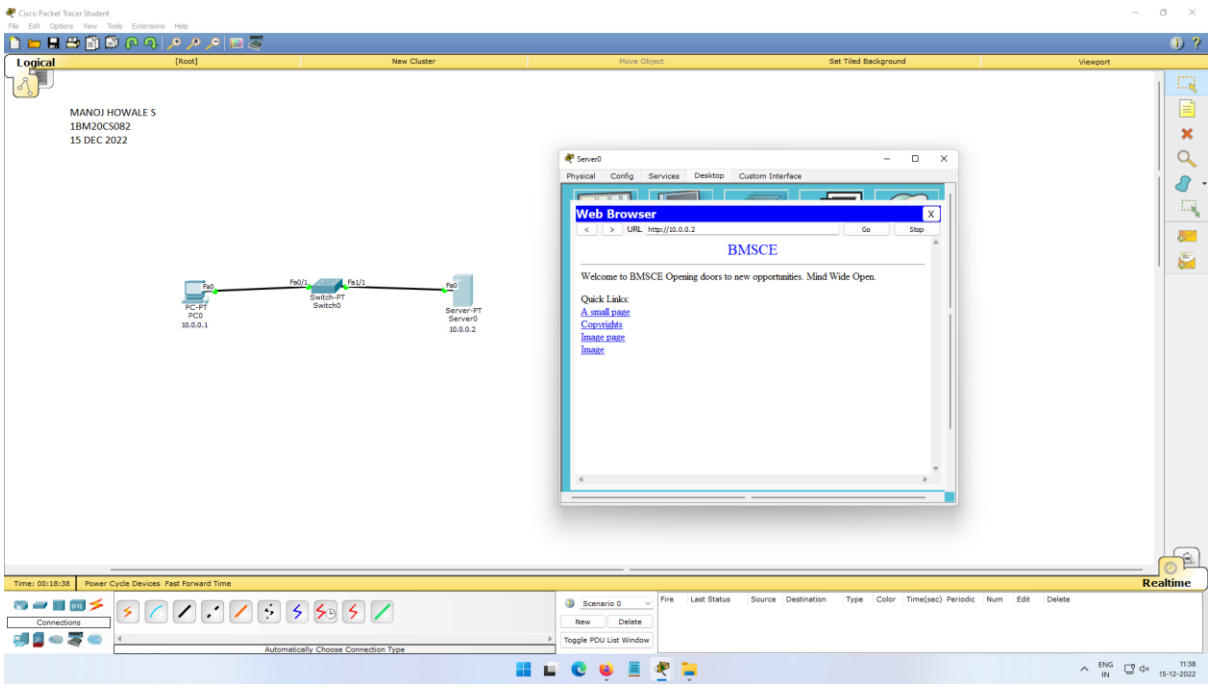
Learning Outcome: Using DNS it is easy to call websites with their names insted of IP address. DNS helps in naming conversion as computers are comfortable with IP address. It maps the name and its corresponding IP address.

Result

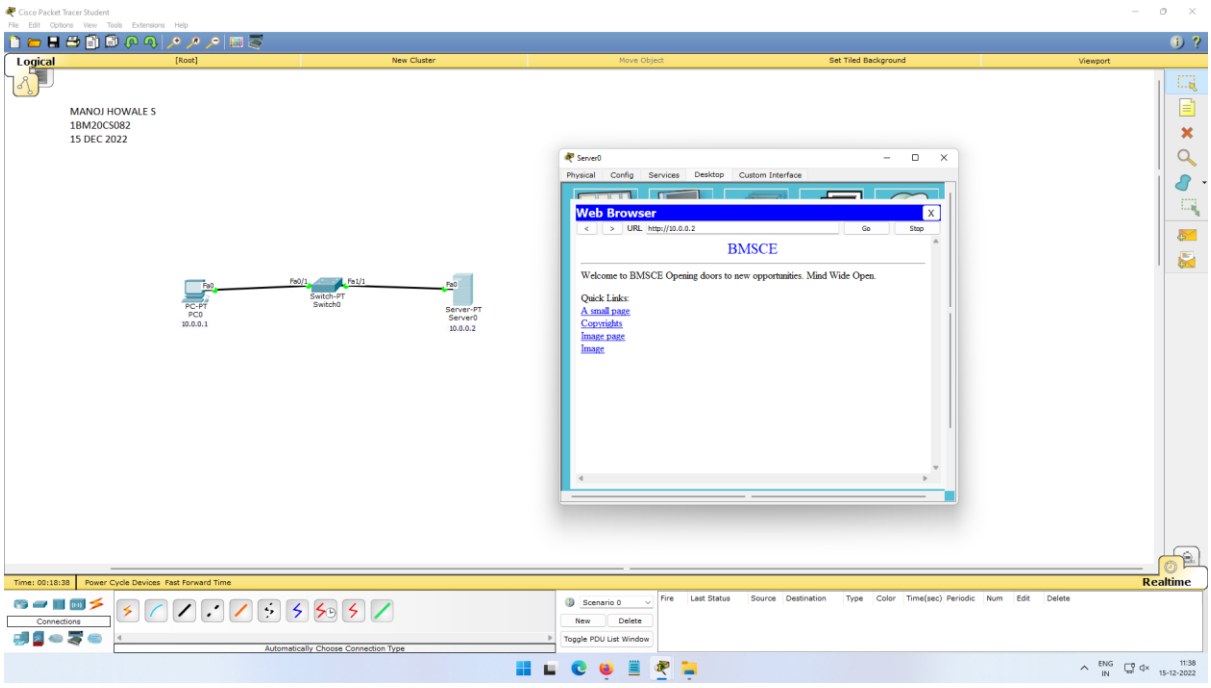


28-12-2021

WEBSITE INTERFACE



RESUME INTERFACE



29/12/22

Lab 7

Error detection using CRC-CCITT (16 bit)

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

```
#include <bits/stdc++.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28], 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 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);
}
```

```
void receiver(){
    printf("Enter the received data");
    scanf("%s", data);
    printf("Data received: %s", data);
    crc();
    for i=0; (i<N-1) && (check_value[i]!='1'); i++);
        if(i<N-1)
            printf("Error detected");
        else
            printf("No error");
    }

int main(){
    printf("Enter data to be sent");
    scanf("%s", data);
    printf("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("Data added with n-1 zeros: %s", data);
    crc();
    printf("CRC or check value is %s", check_value);
    for(i=data_length; i<data_length+N-1, i++)
        data[i] = check_value[i-datalength];
    printf("Final data to send: %s", data);
    receiver();
        return 0;
}
```

Output 1:

Enter data to send: 1000100000001 00001

Enter generator polynomial: 10 111 01

Data added with n-1 zeros: 1000100000001 0000 10 100 0000

CRC or check value is: 010011

Final data to be sent: 1000100000001 0000 10 10011

Data received: 1000100000001 0000 10 10011

No error

Output 2:

Enter data to send: 1000100000001 00001

Enter generator polynomid: 10 111 01

Data added with n-1 zeros: 1000100000001 0000 1000000

CRC or check value is: 010011

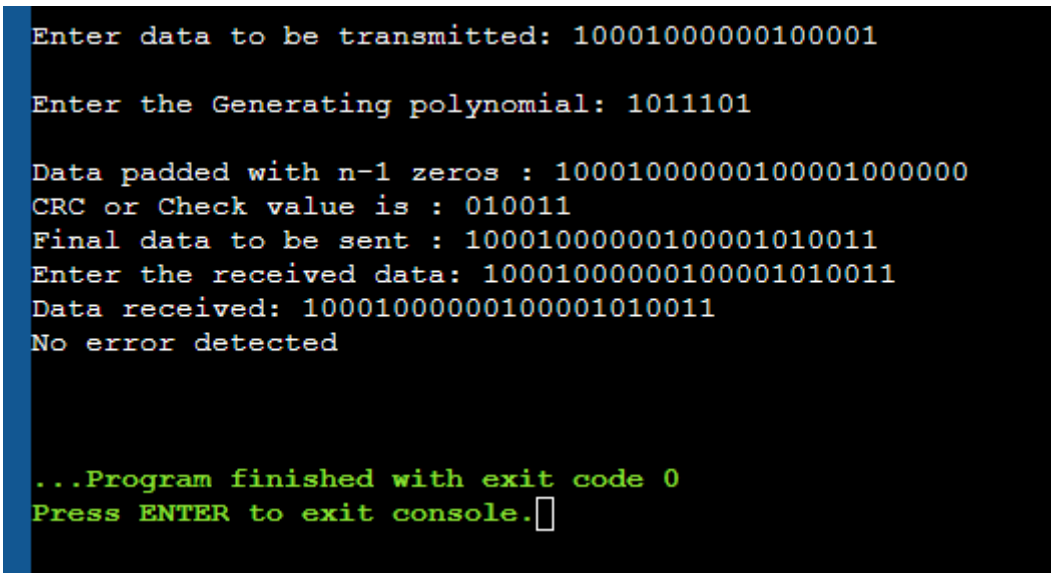
Final data to be sent: 1000100000001 0000 10 10011

Data received: 1000100000001 0000 10 10000

Error detected.

29-12-2021

FINAL OUTPUT



5/11/23

Lab 8

Leaky Bucket Algorithm

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

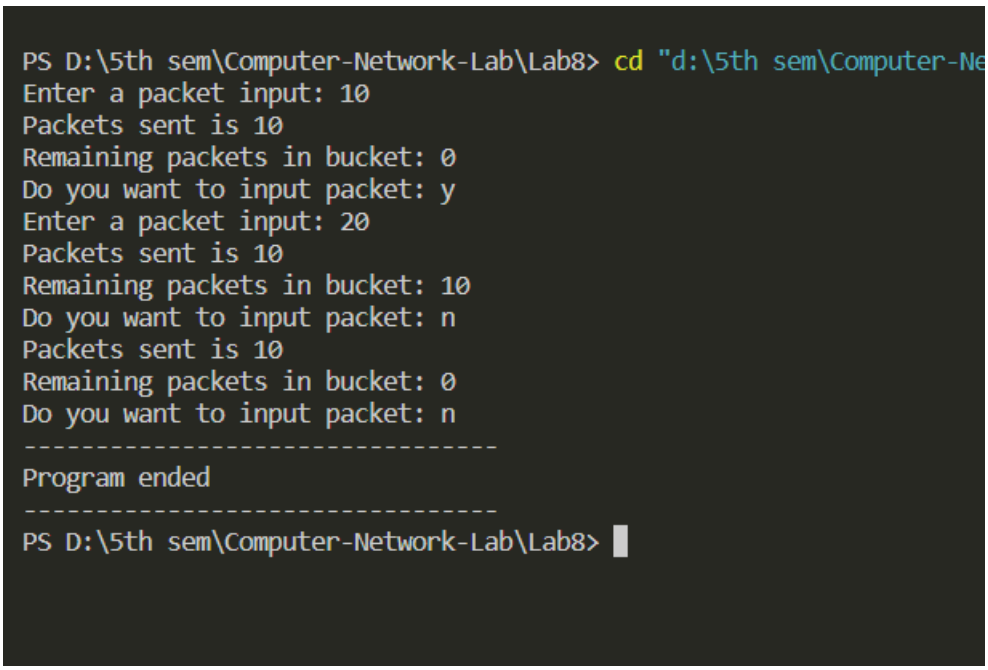
```
#include <stdio.h>
#include <stdlib.h>
int main(){
   int input=0, i=0, bucket limit = 100, op=1
   int size=50, current=0, inpw, out=10;
   char a='y';
   do{
      if(current<=size){
      if (current<=size && a=='y'){
          cout << "Enter packet input";
          cin >> input;
          current += input;
      }
      if (current>=10){
        current = current-10;
        cout << "Packets sent : 10\n");

      }
      else{
          cout << "Packet sent is "<<current<<endl;
          current=0;
      }
      cout << "Remaining packets in bucket: "<<current<<endl;
      cout << "Do you want to input packet;
      cin>>a
   }
   } while (current>0 || a=='y');
```

```
    cout << " End ";
    return 0;
}
```

Output

```
Enter packet input : 20
Packets sent : 10
  Remaining packets = 10
Do you want to input packet : y
Enter a packet input : 10
Packets sent : 10
Remaining packets : 10
Do you want to input packet : n
Packets sent : 10
Remaining packets : 0
Do you want to input packet : n
End.
```

FINAL OUTPUT



Lab 9

Bellmann Ford Alogrithm

Aim: Find suitable path for transmission using Bellmann ford algorithm.

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

int Bellmon Ford (int G[20][20], int V, int E, int edge
 [20][20]) {
int i, u, v, k, distance [20], parent [20], s, flag=1;
for (i=0; i<V; i++) {
    distance [i] = 1000;
    parent [i] = -1; }
    printf ("Enter size ");
    scanf ("%d", &s);
    distance [s-1] = 0;
    for (i=0; i<V-1; i++) {
        for (k=0; k<E; k++) {
            u=edge [k][0];
            v = edge [k][1];
            if (distance [u] + G[u][v] < distance [v])
                distance [v] = distance [u] + G[u][v];
                parent [v] = u;
        }
}
```

```
for (k=0; k<E; k++ ){

    u = edge [k][0]
    v = edge [k][1]
   if (distance[u] + G[u][v] < distance[v] )

       flag = 0
   }

   if (flag)
     for(i=0; i<v; i++)
       printf ( " Vector %d → cost = %d parent = %d \n "
       i+1, distance[i], parent[i]+1 );
       return flag ;
   }

int main (){

   int v, edge [20][2], G[20][20], i, j, k = 0;
   printf ( " Enter no. of vertices " );
   scanf ( " %d ", &v );
   printf ( " Enter graph in matrix form : \n " );
   for (i=0; i<v ; i++)
       for (j=0; j<v; j++ ) {
           scanf ( " %d ", & G[i][j]);
           if ( G[i][j] != 0)
           edge [k][0] = i ;
           edge [k+1][1] = j ;

   }
```

```
if (Bellman_ford(G, V, E, edge))
    printf ("\n No negative weight cycle \n");
else
    printf ("\n Negative weight cycle exists \n");
    return 0;
}
```

Output

```
Enter no. of vertices : 5
Enter graph in matrix:
0  6  0  7  0
0  0  5  8  -4
0  -2 0  0  0
0  0  -3
```

FINAL OUTPUT

```
PS C:\Users\Manoj\OneDrive\Desktop> cd "c:\Users\Manoj\OneDrive\Desktop\" ; if ($?) { g++ Unti
tled-1.cpp -o Untitled-1 } ; if ($?) { .\Untitled-1 }
Enter the number the routers(<10): 5
Enter 1 if the corresponding router is adjacent to routerA else enter 99:
 B C D E
Enter matrix:1 1 99 99

Enter 1 if the corresponding router is adjacent to routerB else enter 99:
 A C D E
Enter matrix:99 99 1 99

Enter 1 if the corresponding router is adjacent to routerE else enter 99:
 A B C D
Enter matrix:99 99 1 99

Router Table entries for router A:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 0 1 1 99 99

Router Table entries for router B:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 1 0 99 99 99

Router Table entries for router C:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 1 99 0 1 1

Router Table entries for router D:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 99 99 1 0 99

Router Table entries for router E:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 99 99 1 99 0
```

12/1/23

Lab 9

Aim : Implement Dijkstras algorithm to find shortest path for given topology

```
#include <stdio.h>
#include <conio.h>
#define INFINITY 999
#define MAX 10

void dijkstra int G[MAX][MAX], int n, int startnode);
int node(){

    int G[MAX][MAX], i, j, n, u;
    printf("Enter no. of vertices");
    scanf("%d", &n);
    printf("\n Enter adjacency matrix: \n");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d", &G[i][j]);
    printf("Enter starting node:");
    scanf("%d", &n);
    dijkstra(G, n, u);
    return 0;
}
```

```
void dijkstra(int G[MAX][MAX], int n, int startnode){
int cost[MAX][MAX], distance[MAX], pred[MAX];
int visited[MAX], count, mindistance, nextnode, i, j;
for(i=0; i<n; i++)
  for(j=0; j<n; j++)
    if(G[i][j]==0)
      cost[i][j]=INFINITY;
    else
      cost[i][j]=G[i][j];
for(i=0; i<n; i++){
  distance[i]=cost[startnode][i];
  pred[i]=startnode;
  visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1){
mindistance=INFINITY;
for(i=0; i<n; i++){
  if(distance[i]<mindistance && visited[i]){
    mindistance=distance[i];
    nextnode=i;
  }
visited[nextnode]=1;
for(i=0; i<n; i++)
    if(!visited[i]){
      if(mindistance+nextnode[i])<distance[i]){
      distance[i]=mindistance+cost[nextnode][i];
      pred[i]=nextnode;
      }
```

```
count++;
for (i=0; i<n; i++)
        if (i != startnode) {
        printf("\n distance of node %d = %d", i, distance[i]);
        j=i;
        do {
         j = pred[j];
          printf("%d", j);
        }
        while (j != startnode);
    }
}
```

Output:

Enter no. of vertices: 4

Enter adjacency matrix

	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	1
3	1	0	1	0

Enter starting node : 1

Distance of 0 = 1

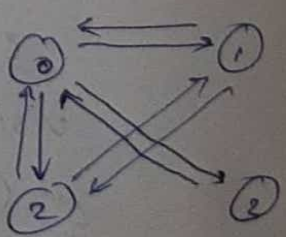
Path = 0 ← 1

Distance of 2 = 1

path = 2 ← 1

Distane of 3 = 2

path = 3 ← 0 ← 1



FINAL OUTPUT

```
PS C:\Users\Manoj> cd "d:\5th sem\Computer-Network-Lab\Lab9\DIJIKSTRA\" ; if ($?) { g++ dijiks
tra.cpp -o dijikstra } ; if ($?) { .\dijikstra }
Enter the graph
0 9 2 5
9 0 6 8
2 6 0 0
5 8 0 0
Vertex          Distance from Source
0               0
1               8
2               2
3               5
PS D:\5th sem\Computer-Network-Lab\Lab9\DIJIKSTRA>
```

Lab 10

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

client.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 filename: ")
clientSocket.send(Sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print(filecontents)
clientsocket.close()
```

Server.py

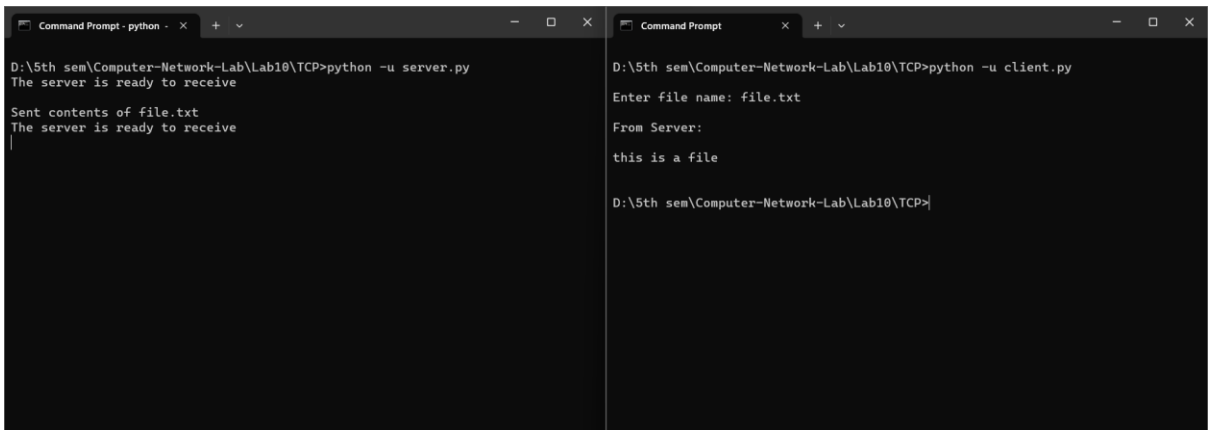
```
from socket import *
servername = '127.0.0.1'
serverport = 12000
serversocket = socket(AF_INET, SOCK_STREAM)
serversocket.Bind((Servername, serverport))
serversocket.listen(1)
while(1):
    print("Server is ready to recieve")
```

```
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:

```
Enter filename : Server TCP.py
From Server:
  connection socket, addr = serversocket.accept
These are the contents of file.
```

FINAL OUTPUT



Lab 10

Using UDP sockets write client server program to make client sending filename and server to send back content of requested file if present.

```
from Socket import *
Server name = '127.0.0.1'
Server port = 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 (" Reply from server )
print (" File contents .decode("utf-8" ))
clientSocket.close().
client.socket.close()
```

```
from socket import *
server port = 12000
serverSocket = socket (AF_INET, SOCK-DGRAM)
serverSocket.bind(('127.0.0.1', serverport ))
print (" Server is ready to recieve")
while 1:
    sentence, clientaddress = Serversocket.recvfrom (2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, 'r')
```

```
l = file.read(2048)
Server_socket.sendto(Bytes(l, "utf-8"), clientaddress)
print('\n Send content of ', end=' ')
print(sentence)
file.close()
```

Output

```
Enter filename: file.txt
Reply from server:
This is a file.
```

FINAL OUTPUT