

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPUTER NETWORKS LAB

Submitted by

MANOJ HOWALE S (1BM20CS082)

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

October-2022 to Feb-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 LAB COURSE "**COMPUTER NETWORKS**" carried out by **MANOJ HOWALE S (1BM20CS082)**, 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 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Computer Networks - (20CS5PCCON)** work prescribed for the said degree.

Dr. SEEMA PATIL
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

.

Index

Sl. No .	Date	Experiment Title
1	10-11-22	Creating a topology and simulating sending a simple PDU from source to destination using hub and switch as connecting devices.
2	17-11-22	Configuring IP address to Routers in Packet Tracer. Explore the following messages: Ping Responses, Destination unreachable, Request timed out, Reply
3	24-11-22	Configuring default route to the Router
4	01-12-22	Configuring DHCP within a LAN in a packet Tracer
5	08-12-22	Configuring RIP Routing Protocol in Router
6	15-12-22	Demonstration of WEB server and DNS using Packet Tracer
7	29-12-22	Write a program for error detecting code using CRC-CCITT (16-bits).
8	05-01-23	Write a program for distance vector algorithm to find a suitable path for transmission.
9	12-01-23	Implement Dijkstra's algorithm to compute the shortest path for a given topology.
10	12-01-23	Write a program for congestion control using Leaky bucket algorithm.
11	26-01-23	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.
12	26-01-23	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.

10/11/22

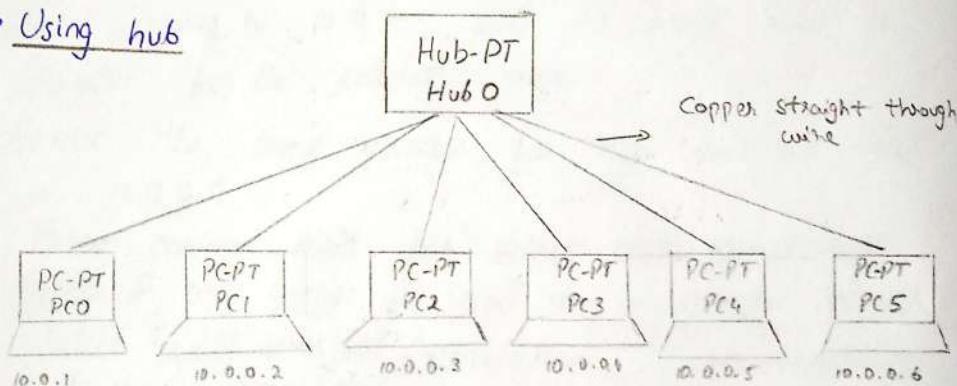
Lab 1

Hubs and Switches

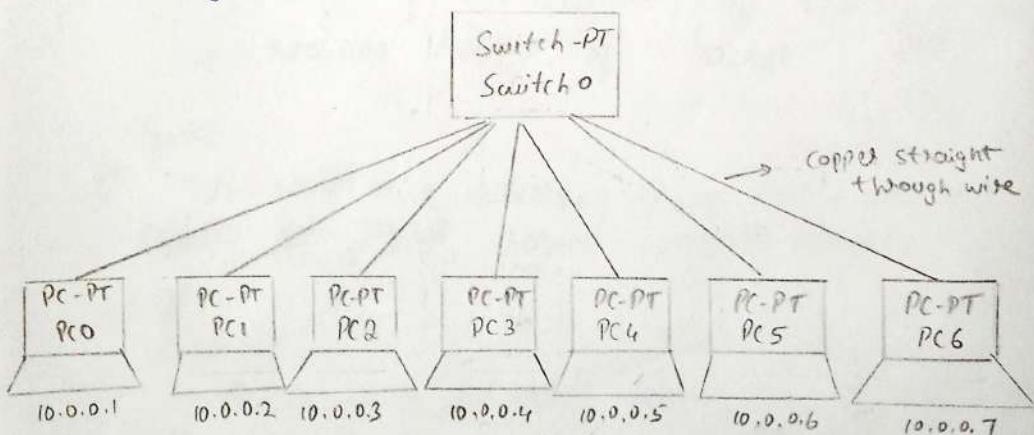
Aim: Creating a topology and simulate sending a simple PDU from source to destination using simple hub and switch as connecting devices.

Topology:

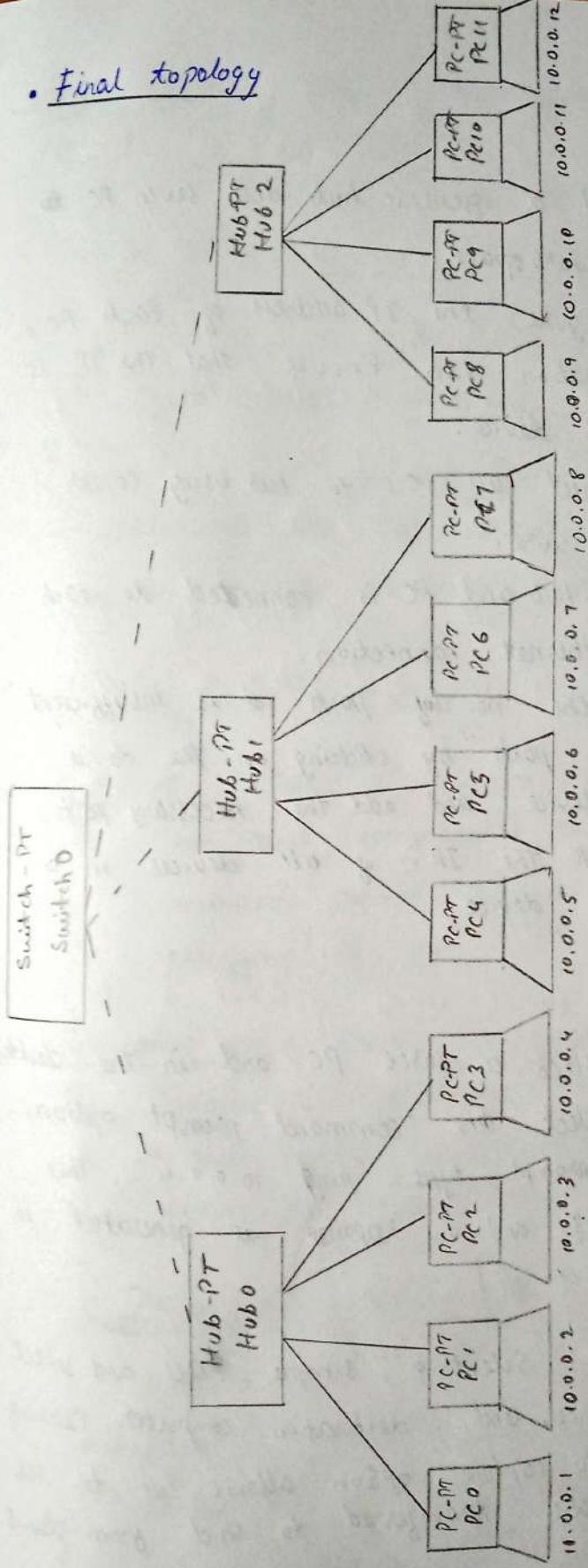
• Using hub



• 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

Working

Real time: Select a source 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 port of the PC in configure and add a note below each PC containing IP address.

Real time mode: Select the PC you want to send the socket 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 PDC by selecting the pair of PC and click on autocapture 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 = 0ms

Reply from 10.0.0.3 : byte = 32 time = 0ms

Ping statistic for 10.0.0.3

Packets sent = 4, received = 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 receives 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 they 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 crossover as they belong to the same network layer but PC and hubs are connected through copper straight through as they 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 an acknowledgement back to the source PC.

Results :

PC > 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

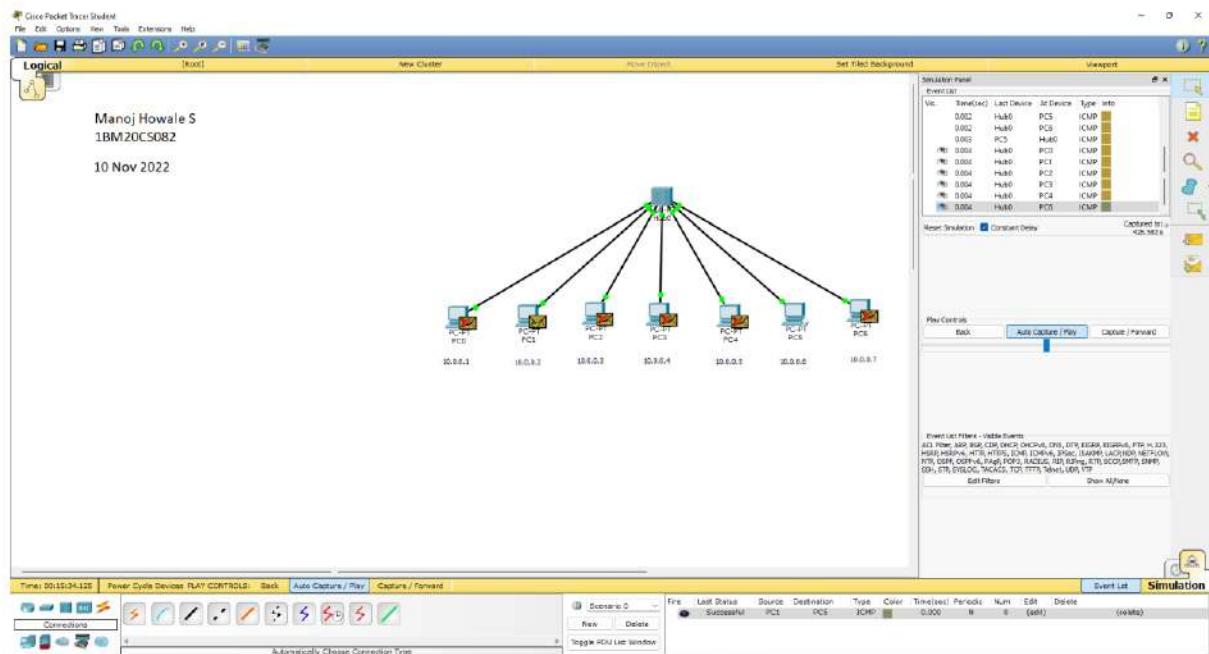
Reply from 10.0.0.5: bytes = 32 time = 1ms TTL = 128

Ping statistics for 10.0.0.7:

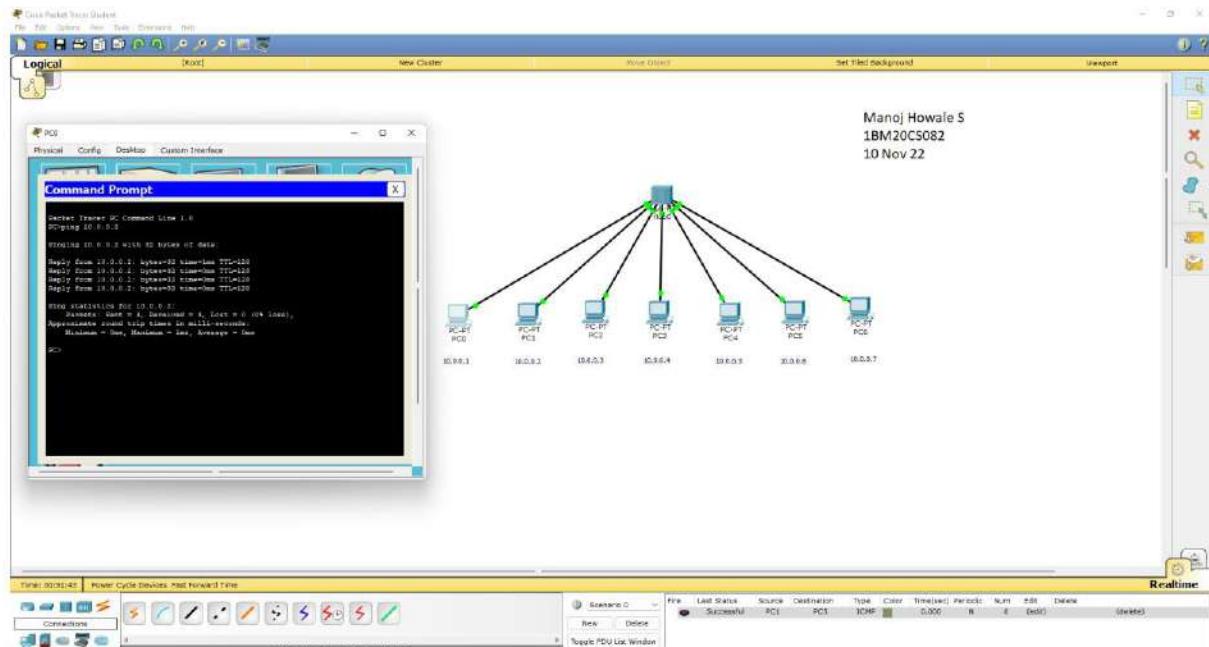
Packet : sent = 4 received = 4, lost = 0 (0% loss)

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

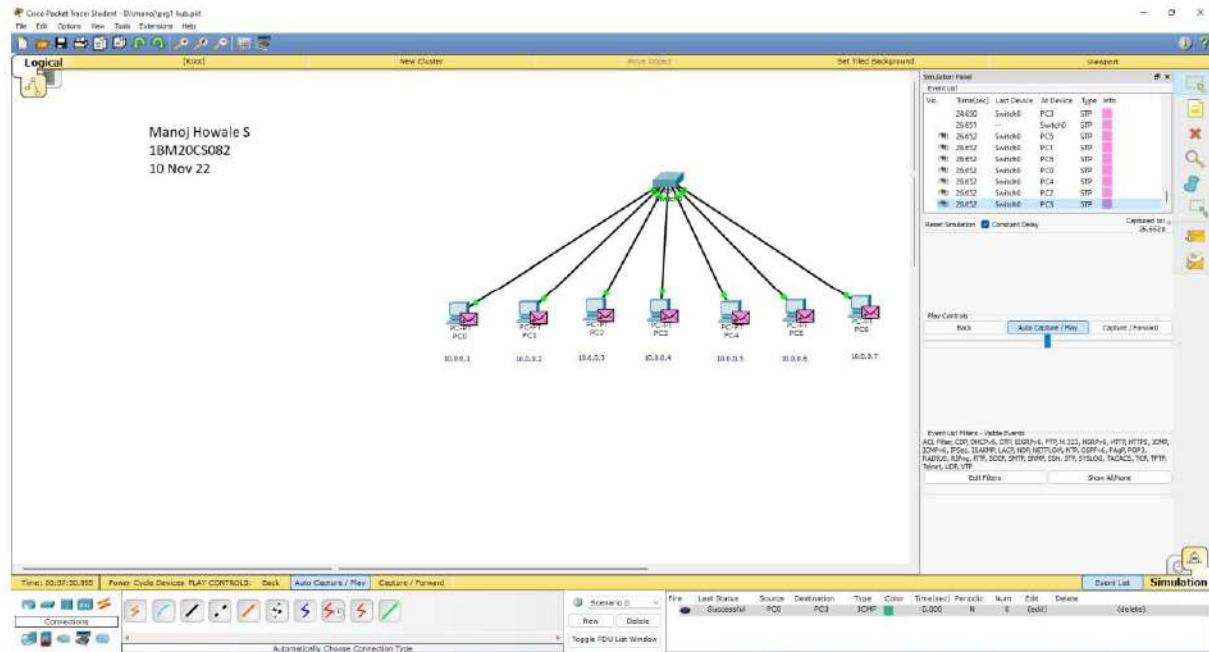
HUBS---SIMULATION



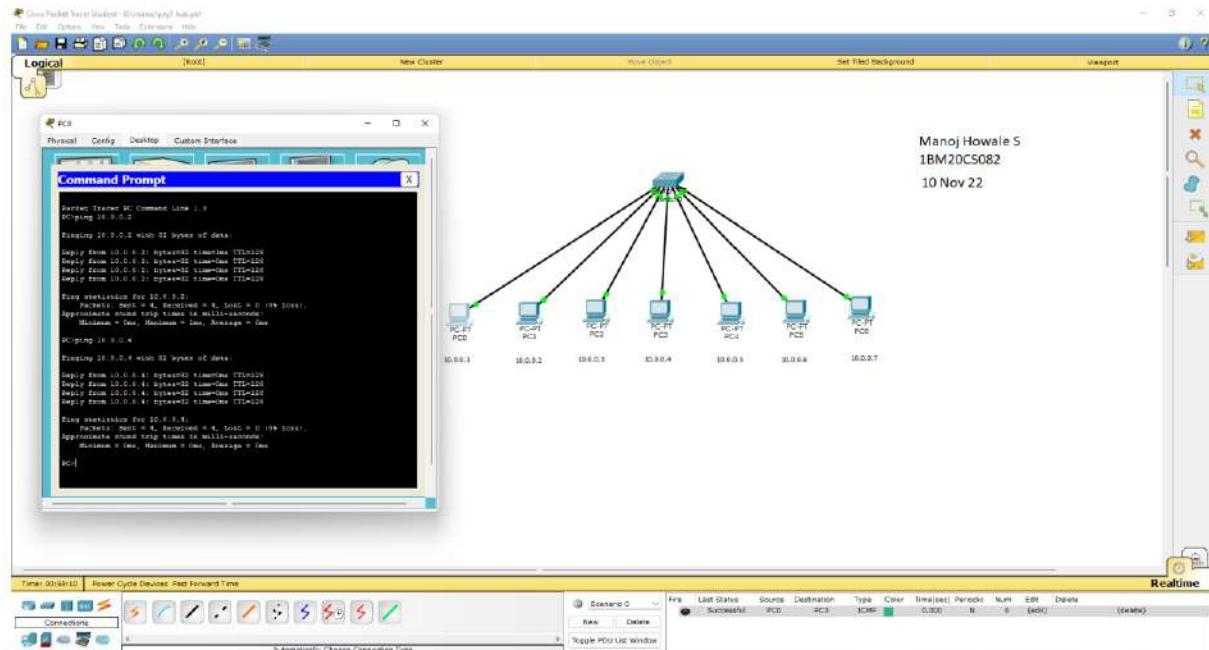
HUBS---REAL TIME



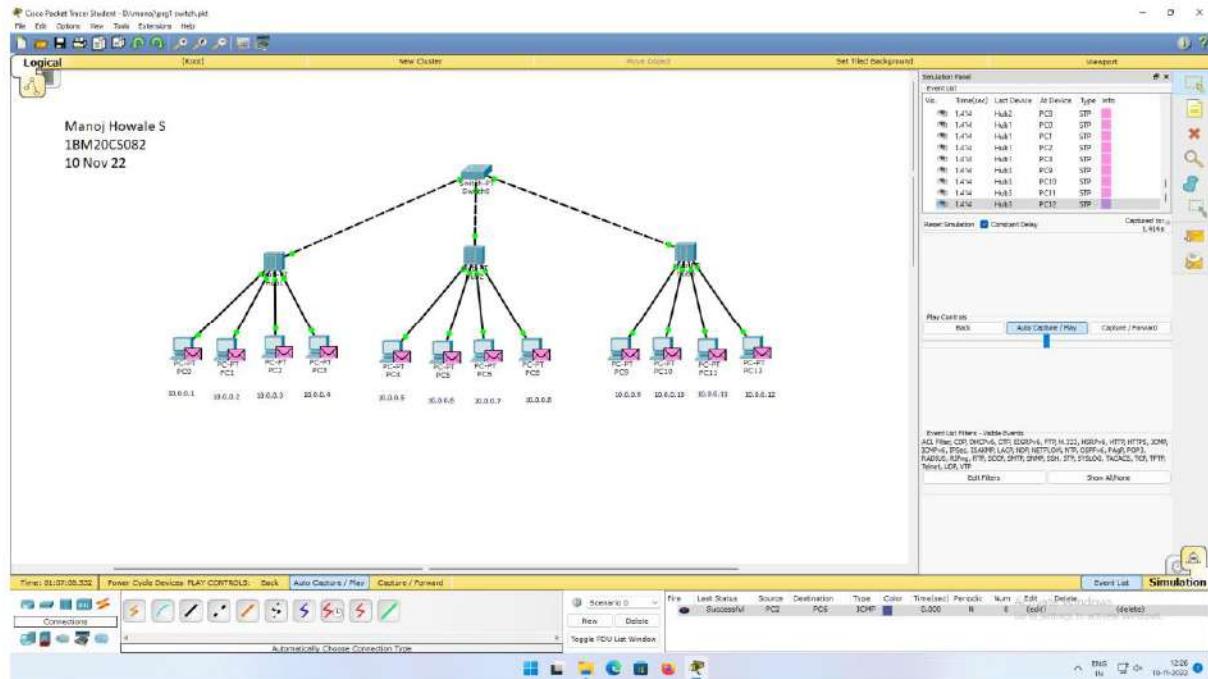
SWITCHES----SIMULATION



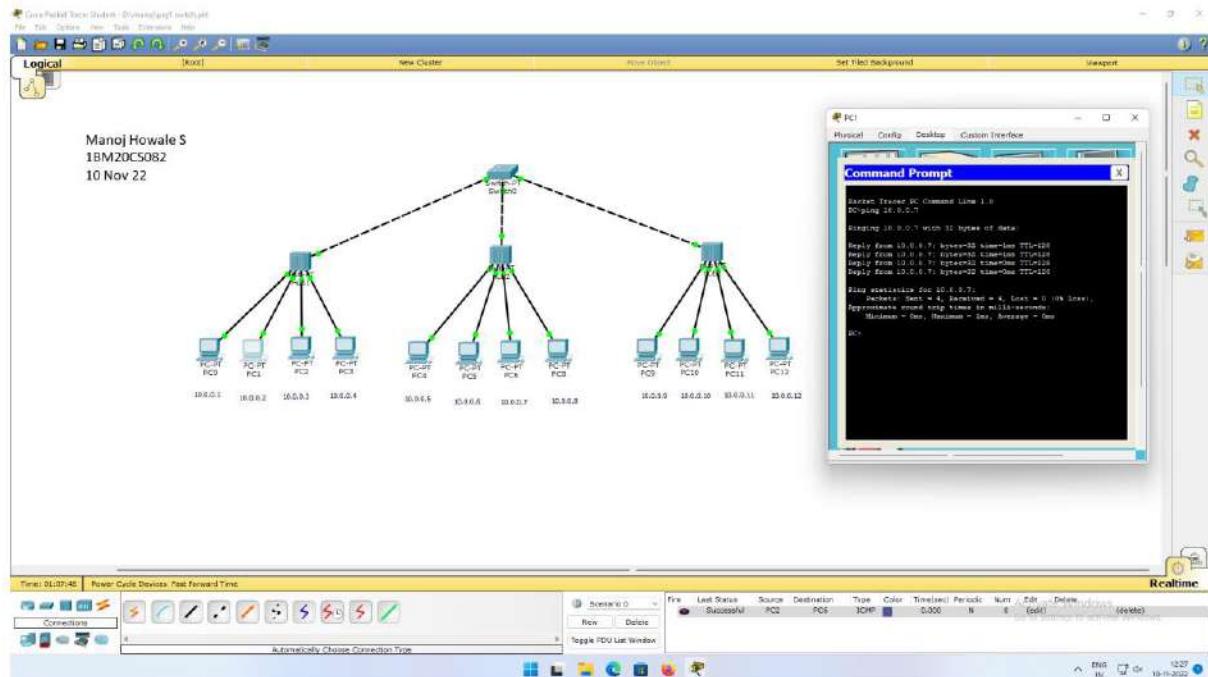
SWITCHES---REAL TIME



FINAL NETWORK----SIMULATION



FINAL NETWORK---REAL TIME



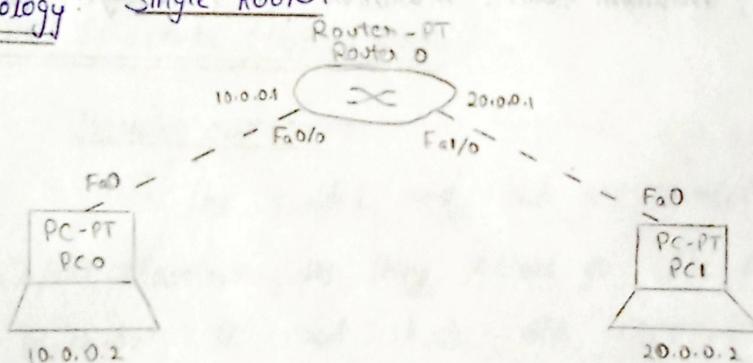
17/11/22

Lab 2

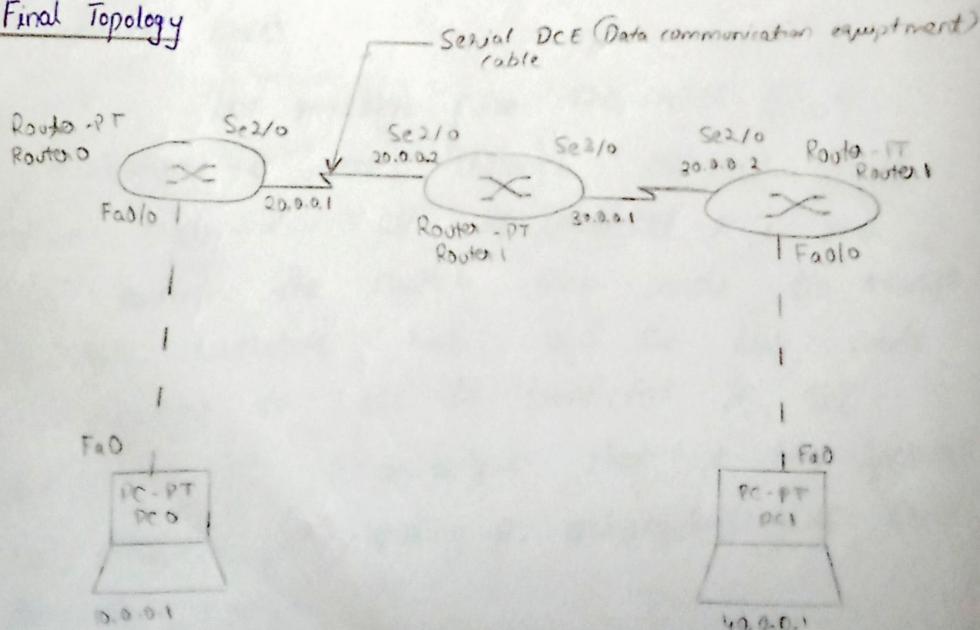
Routers

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 crossover.
- In the Router go to CLI and type the following commands

Router> enable

Router# configure t

Router (config)# interface Fast Ethernet 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-if)# 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, and 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/  
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 Router0 with IP's of Fa0/0 as 40.0.0.10 and Se2/0 as 30.0.0.2.

• Configure Router 1 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 route 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 CLT.

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 packets are received by PC1. If packets are received 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

Ping statistics for 20.0.0.1

packets : sent = 4 , received = 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 ~~route~~ 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 < 1ms TTL = 127

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

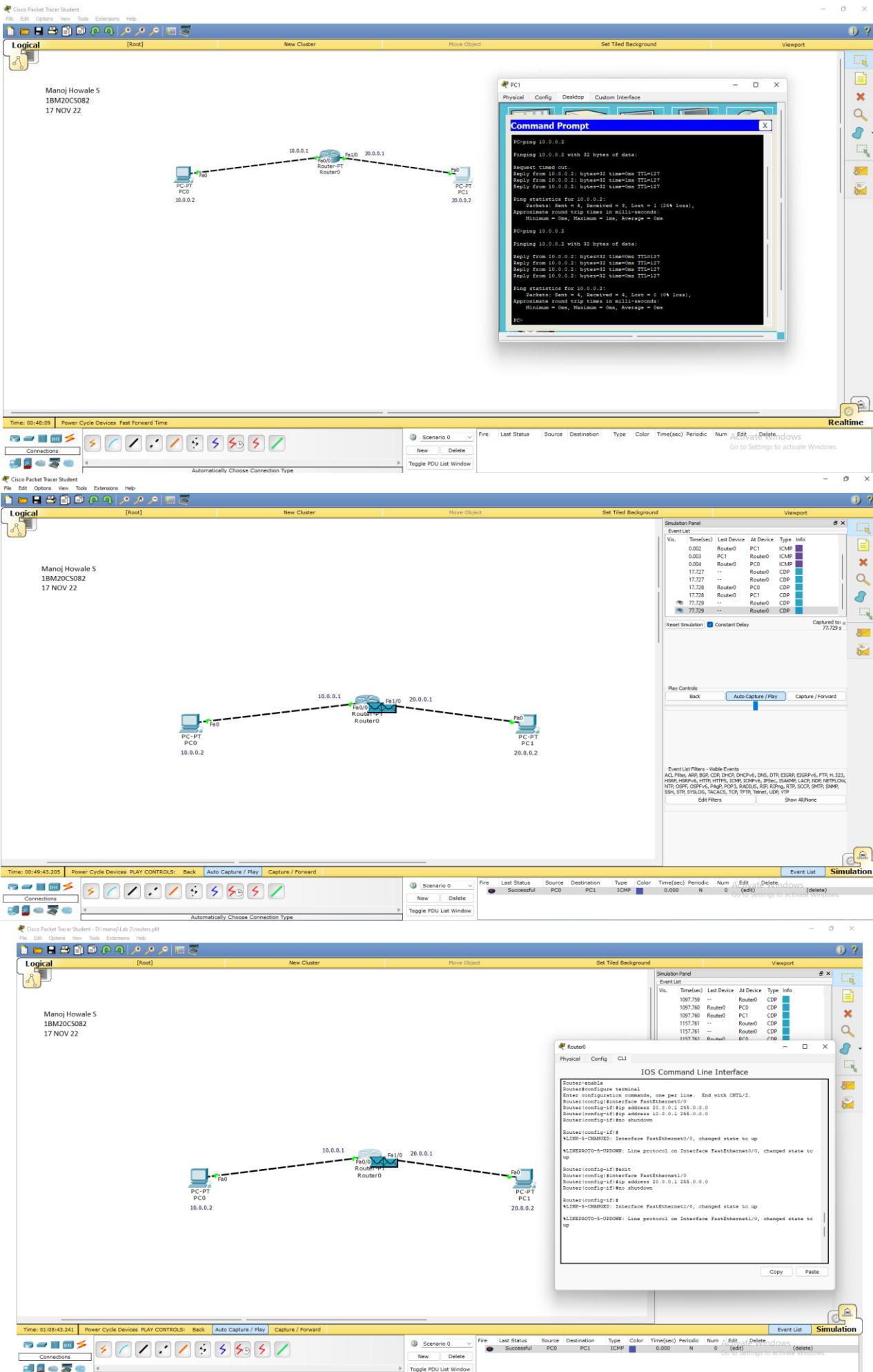
Reply from 40.0.0.1 : bytes = 32 time < 1ms TTL = 127

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

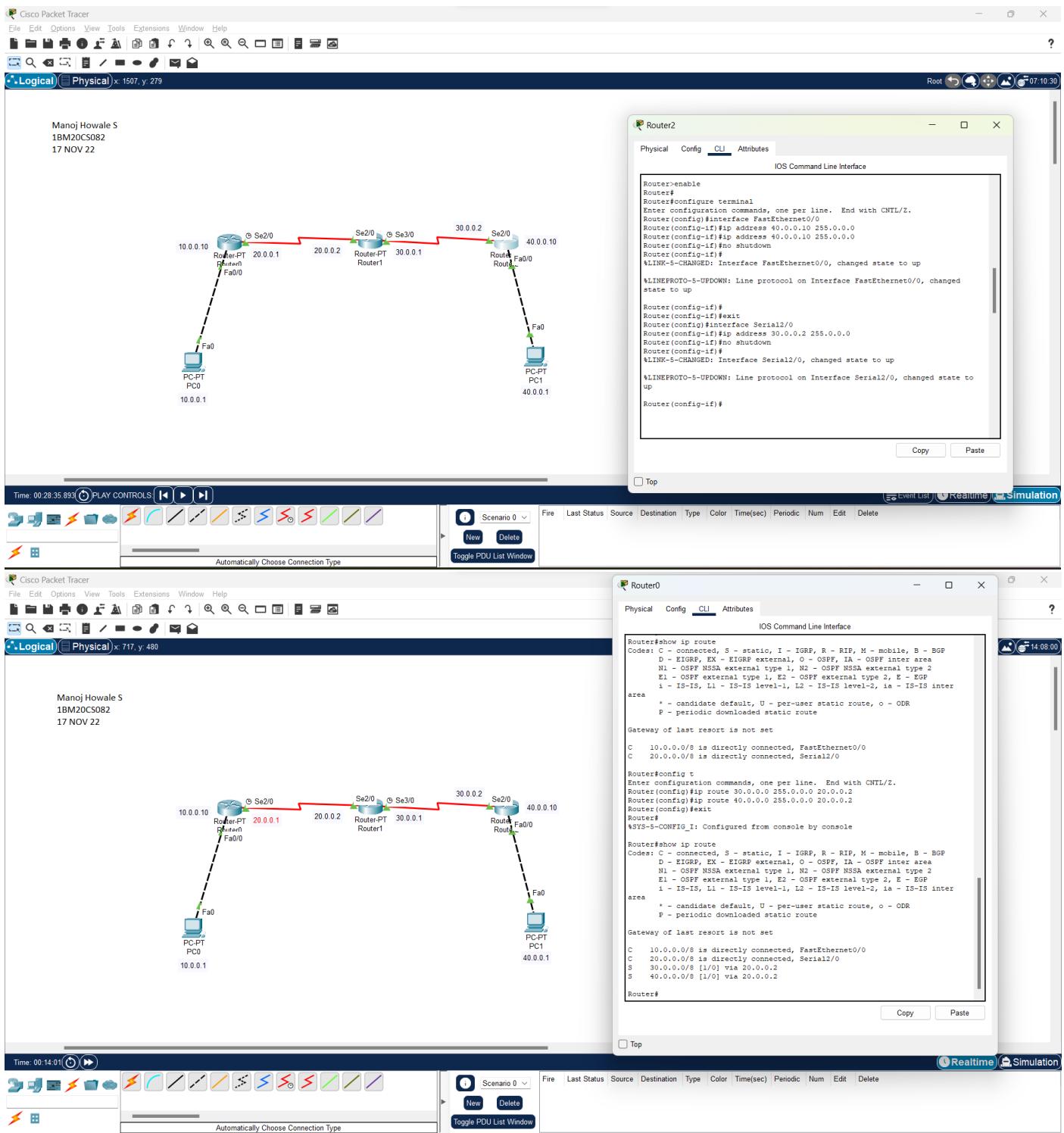
Ping statistics for 40.0.0.1

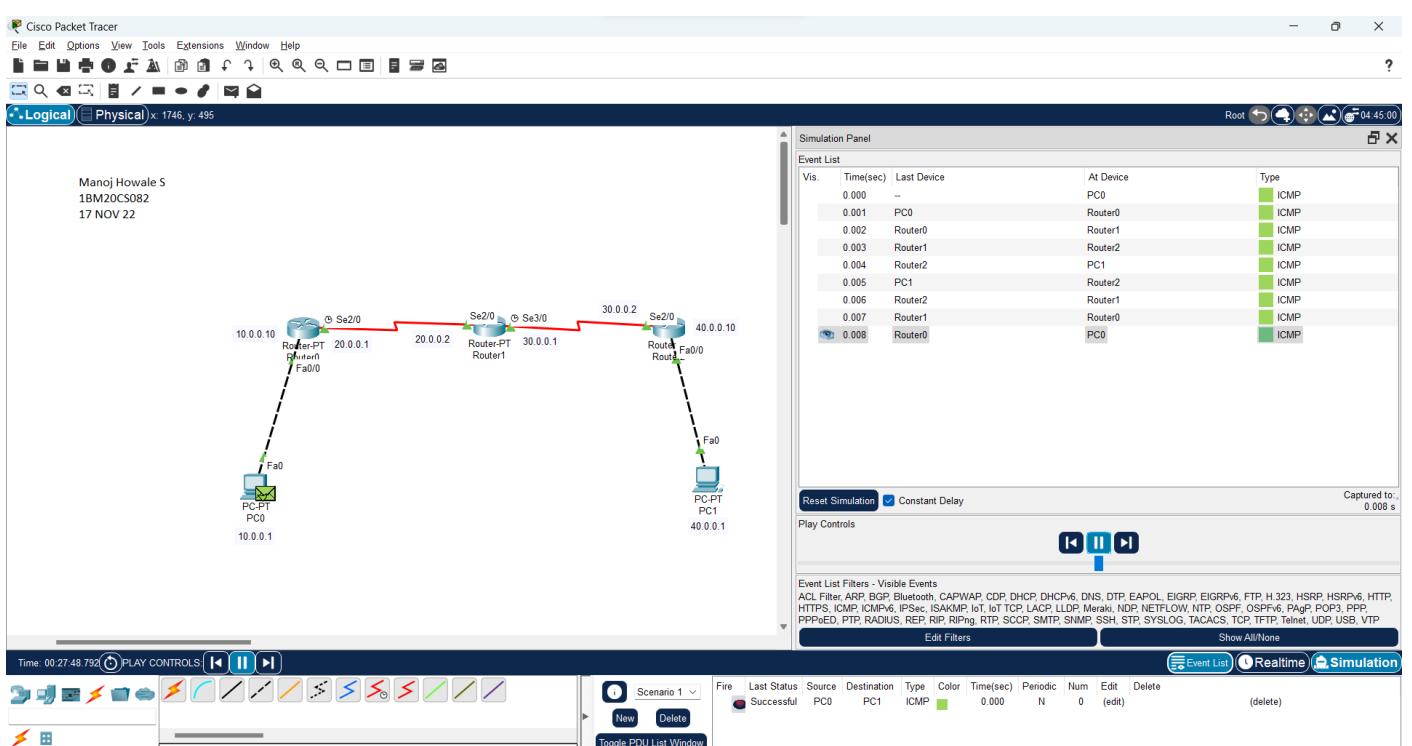
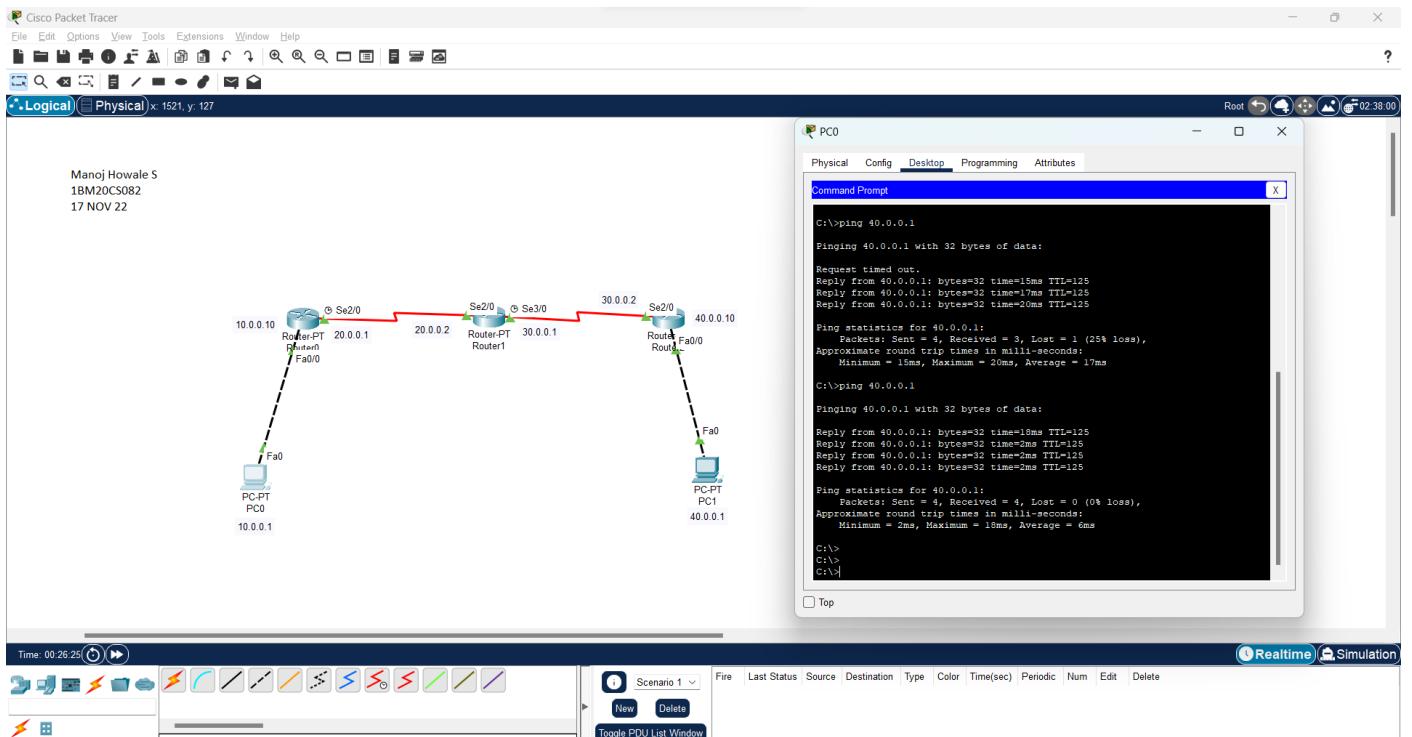
packets : sent = 4 , received = 4 , lost = 0 (0% loss)

SINGLE ROUTER—REAL TIME



ROUTERS—REAL TIME





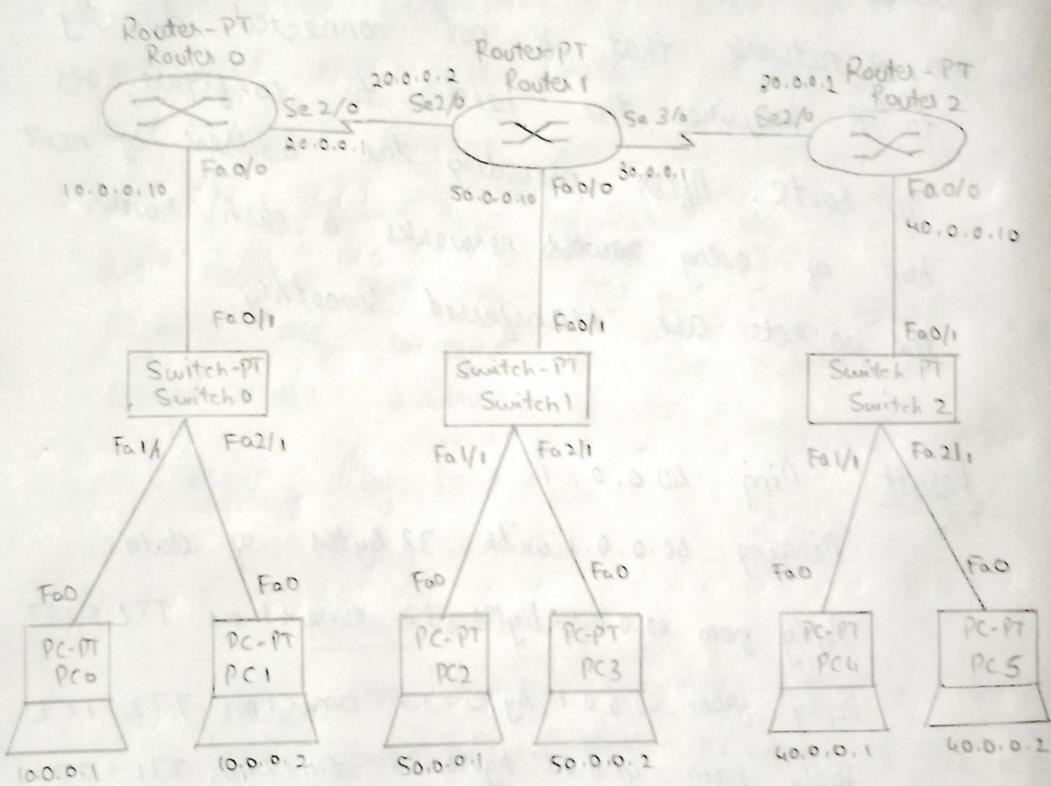
24/11/22

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 one 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 repeat the same ping between two PC's the request is serviced.

Observation :

The routers even though being configured in CLI would only know ~~about~~ about the neighbouring routers ~~and~~ that it is directly connected to. To send a packet to a network that the router is not directly connected it requires ^{configuration} static routing. After providing default routing and static routing ~~of~~ to ~~to~~ every network the packets are sent successfully. The packets 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

Reply from

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

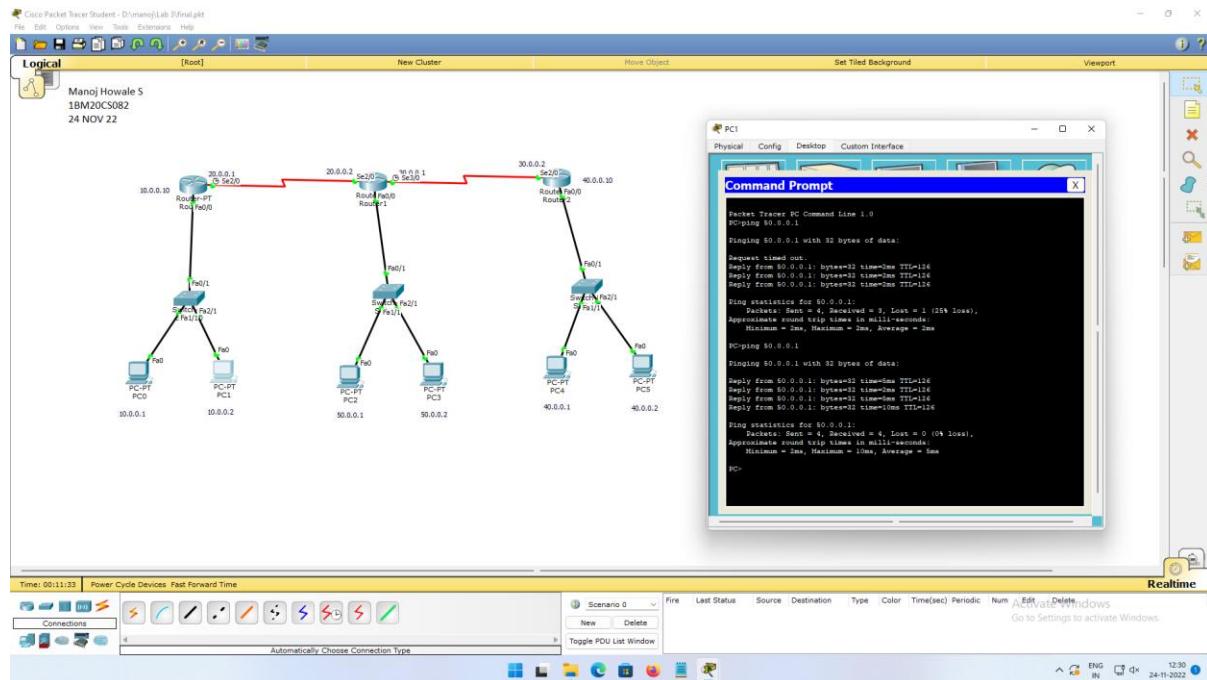
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127

PC> Ping 40.0.0.1

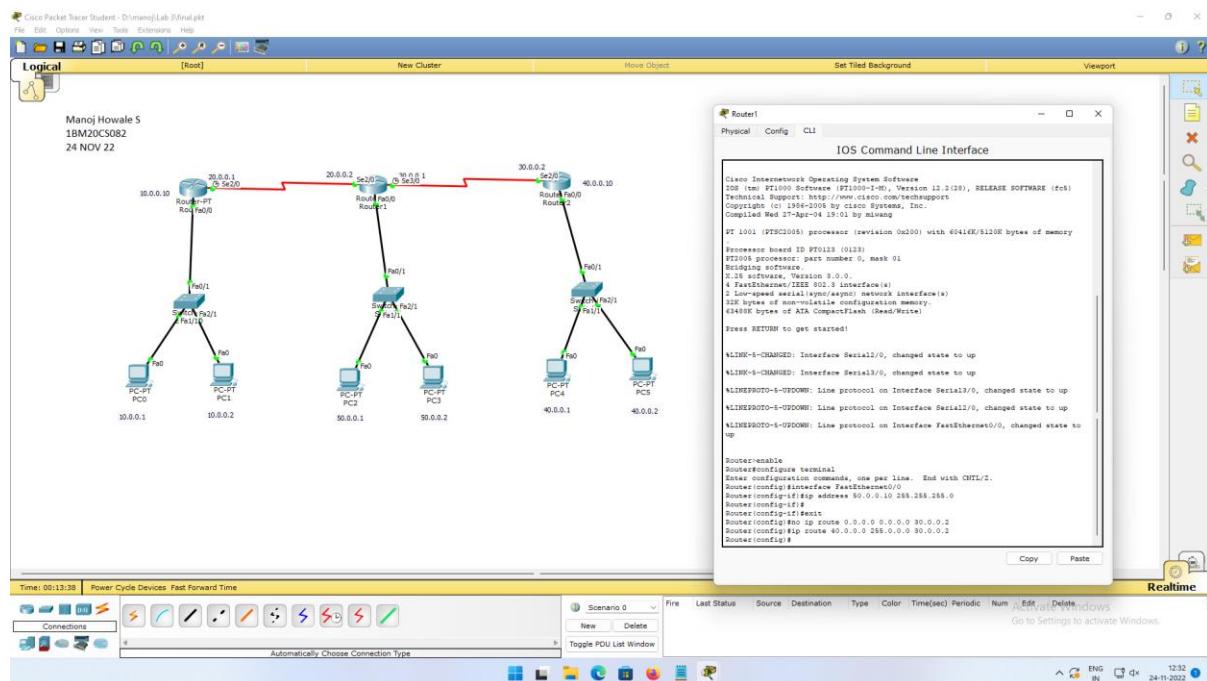
Pinging 40.0.0.1 with 32 bytes of data

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

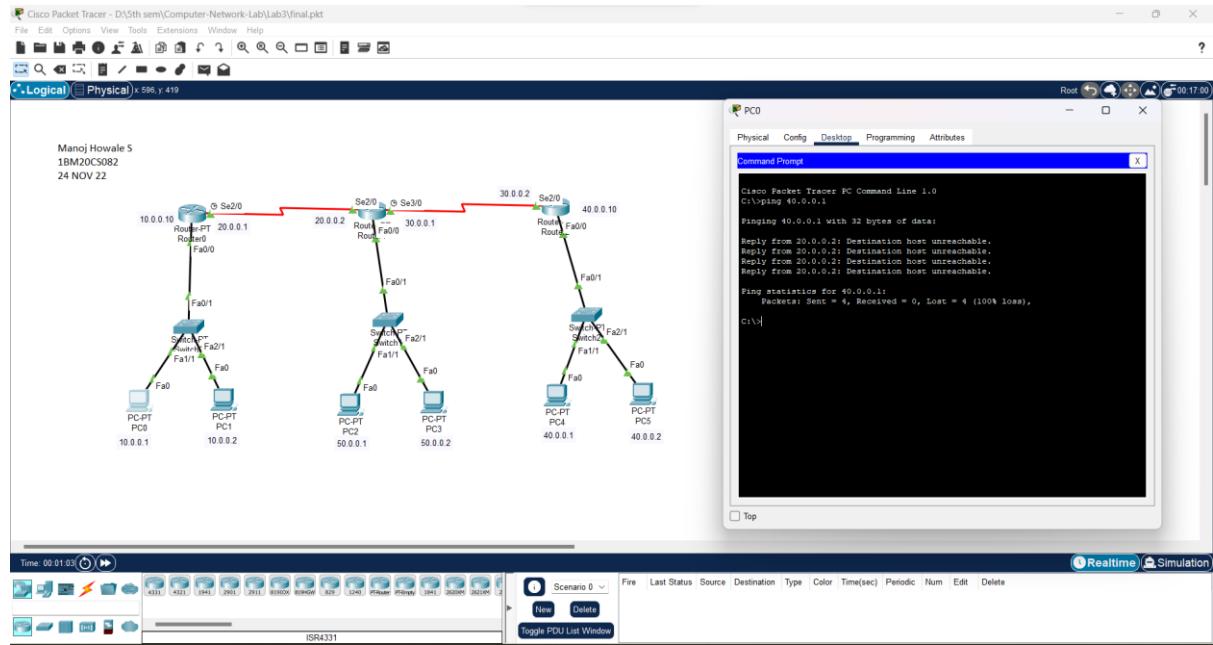
Final Topology -- Realtime



Final Topology – IP route



Final Topology – Destination Host Unreachable



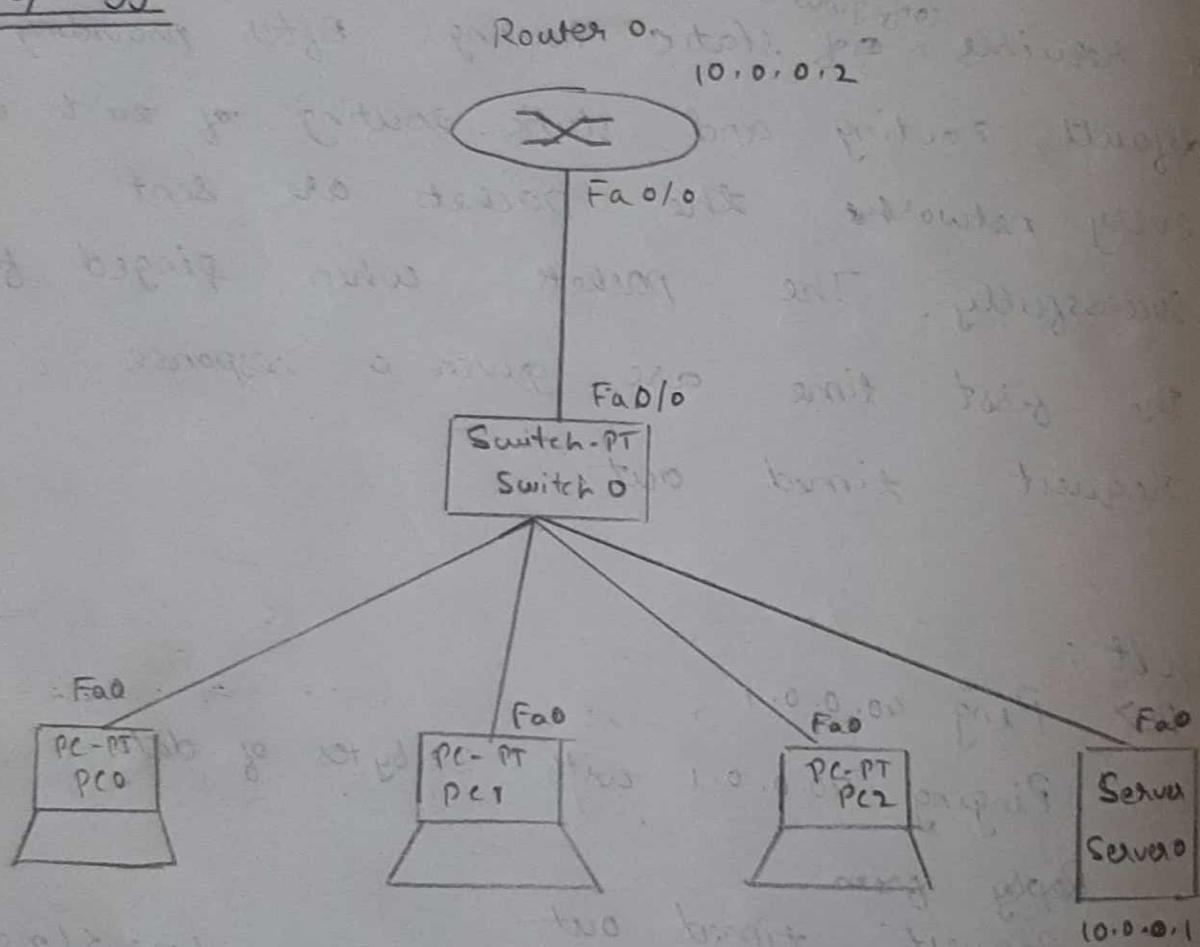
1/12/22

Lab 4

DHCP

Aim: Configuring DHCP within LAN in a Packet tracer using a Router.

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 #  
Router (config) # interface Fast Ethernet 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:

Servers 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 = 128

Reply from 10.0.0.5: bytes = 32 time = 0ms TTL = 128

Reply from 10.0.0.5: bytes = 32 time = 0ms TTL = 128

Reply from 10.0.0.5: bytes = 32 time = 0ms TTL = 128

Ping statistics for 10.0.0.5

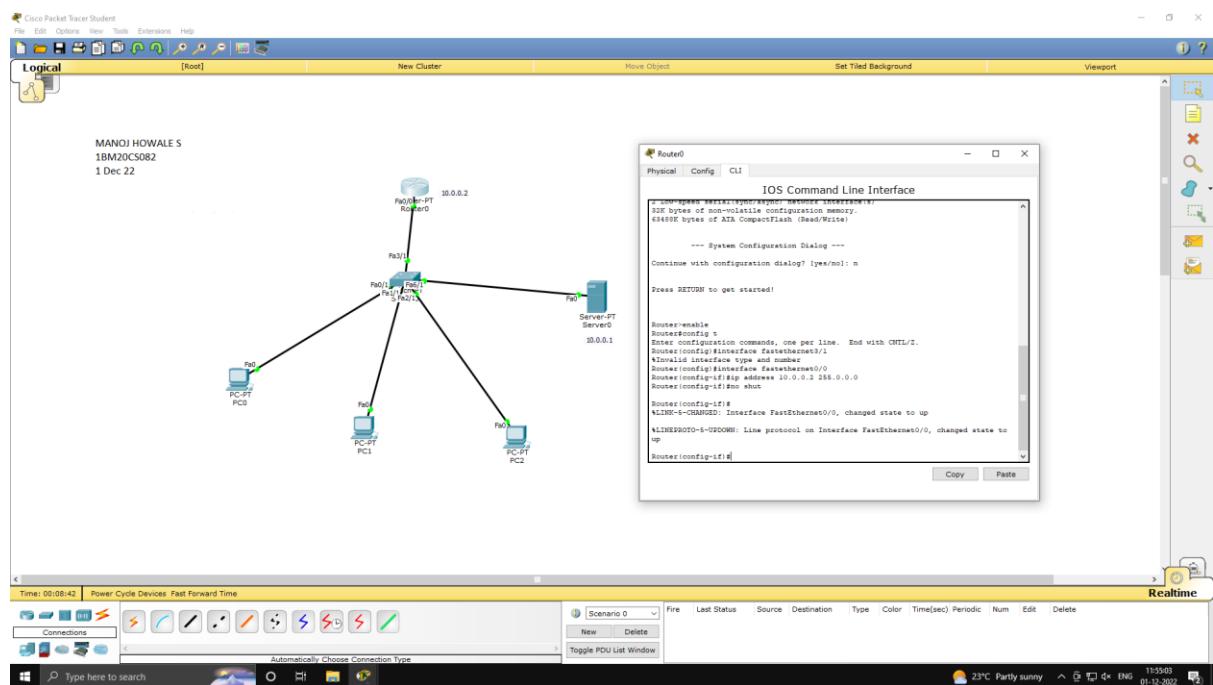
Bytes: Sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip time in milliseconds:

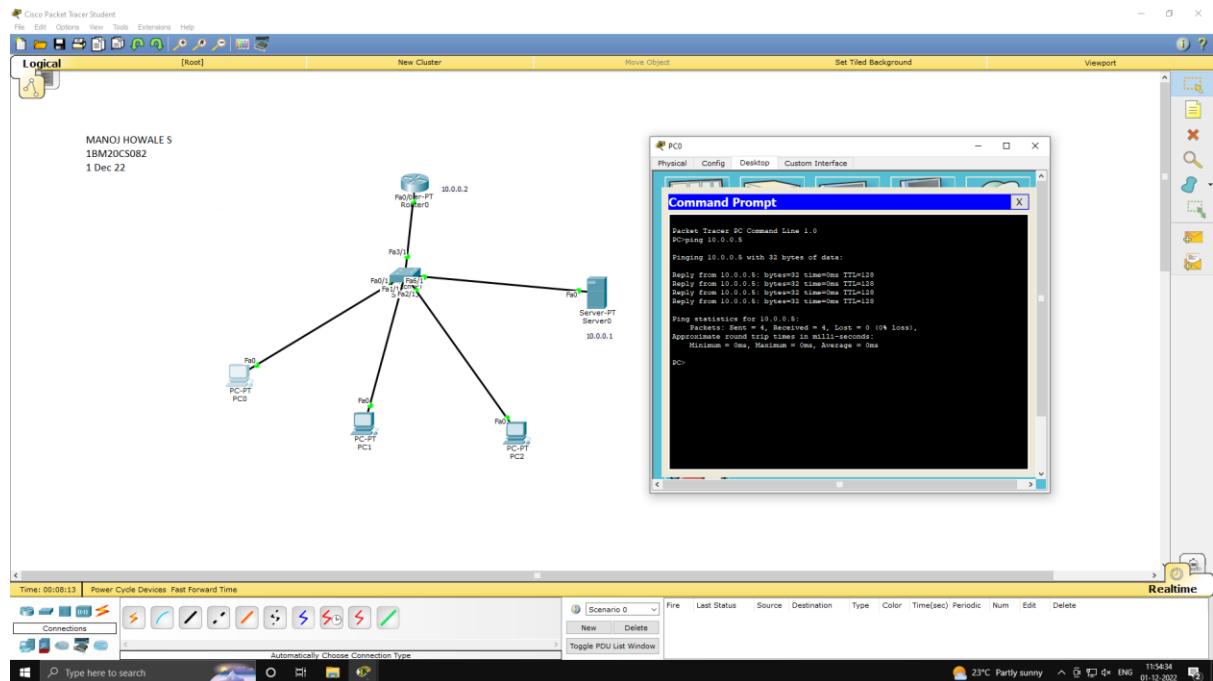
Minimum = 0 ms, Maximum = 0 ms, Average = 0ms

N
8/12/22

DHCP CLI



DHCP SIMULATION



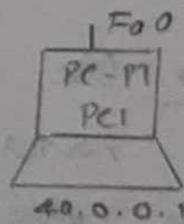
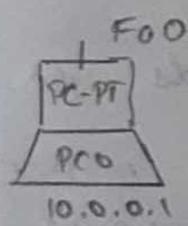
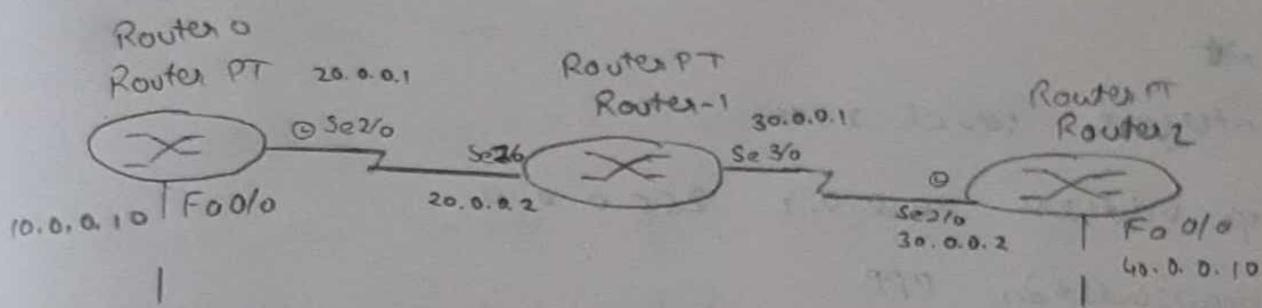
1/2/22

Lab 5

RIP

Aim: Configuring RIP Routing Protocol 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 connect 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 PC^o 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 ~~integrate~~ configure the router by entering the same commands except for encapsulation PPP "clock rate 64000" command.
- Set the RIP protocol 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 protocol 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 = 12ms TTL = 125

Ping statistics for 40.0.0.1:

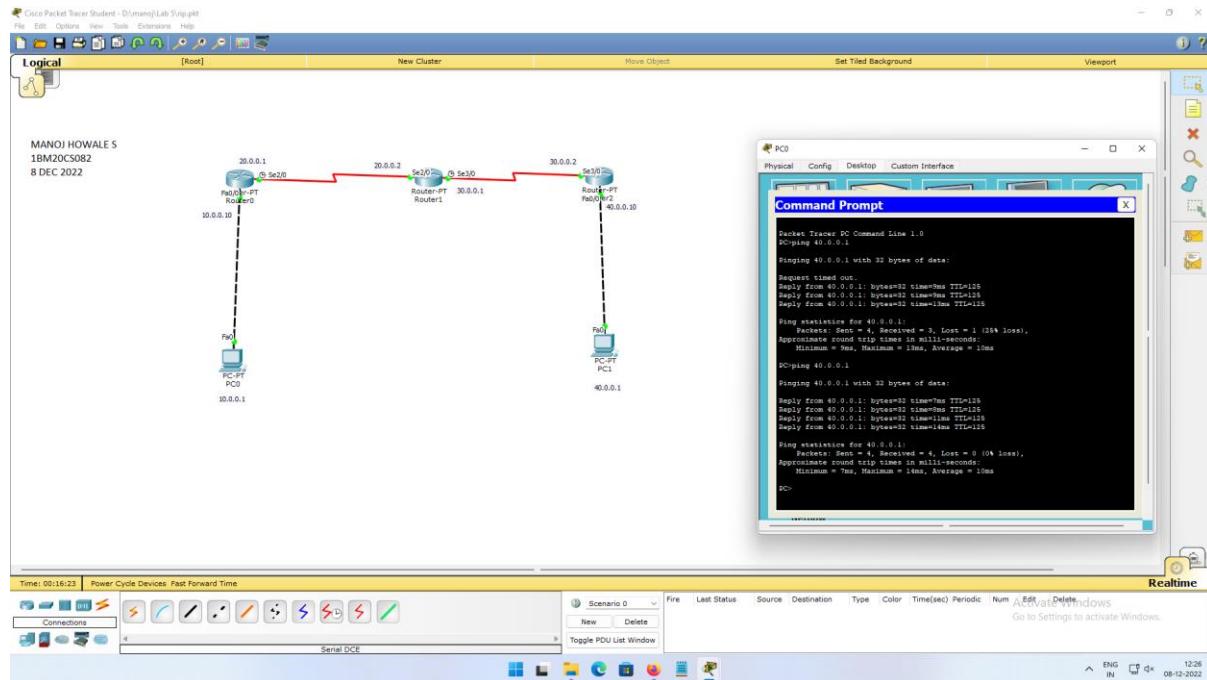
packets : sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip times in milliseconds:

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

Wali
29-12-2022

RIP OUTPUT



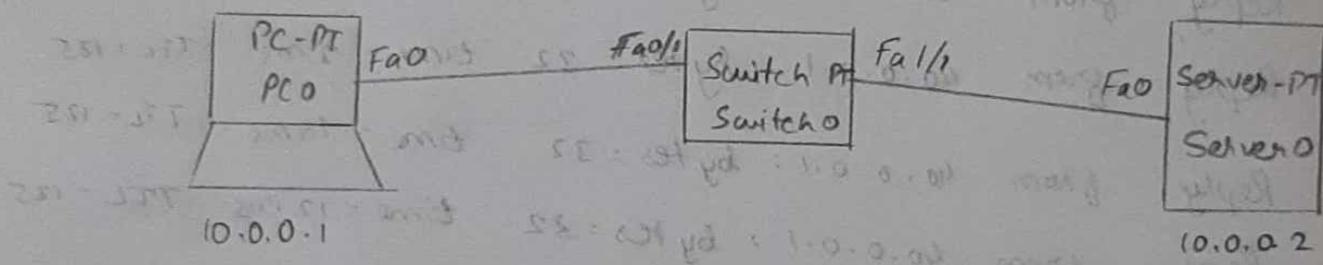
15/12/22

Lab 6

Web Server

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

Topology:



Procedure :

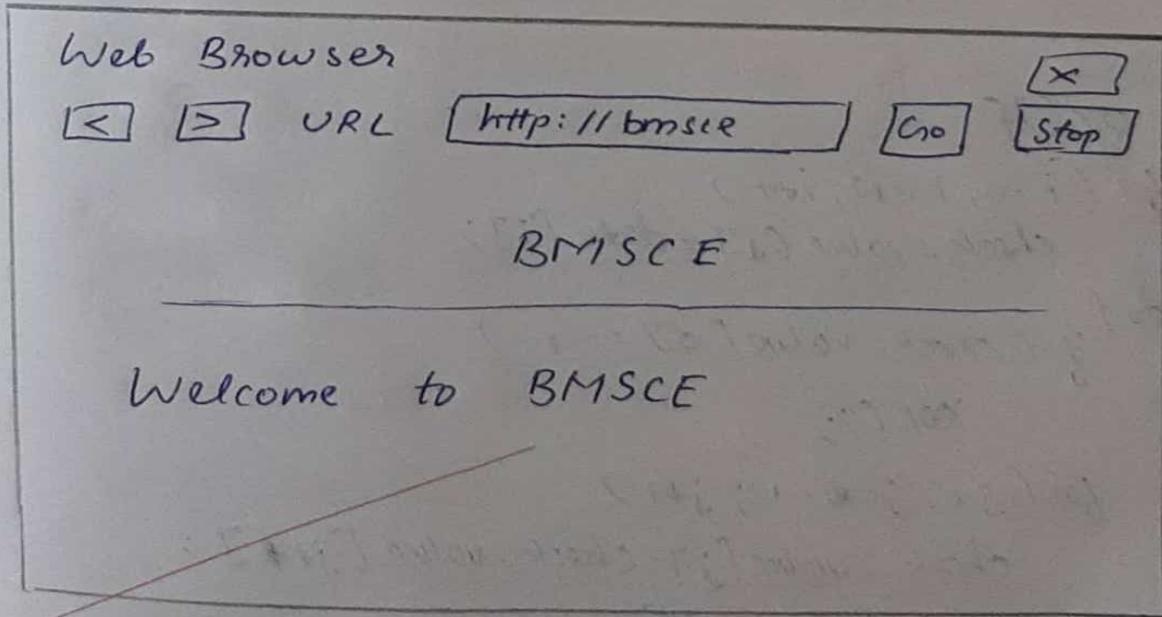
- Add the PC, switch and server in the workspace and connect the components as shown using copper straight through.
- Configure the IP address of PC as 10.0.0.1 and server as 10.0.0.2
- Now go to the server and in the services tab go to DNS and turn it on.
- Go to HTTP and turn on both http and https. 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} current website a .ev page and link it to the

Observation:

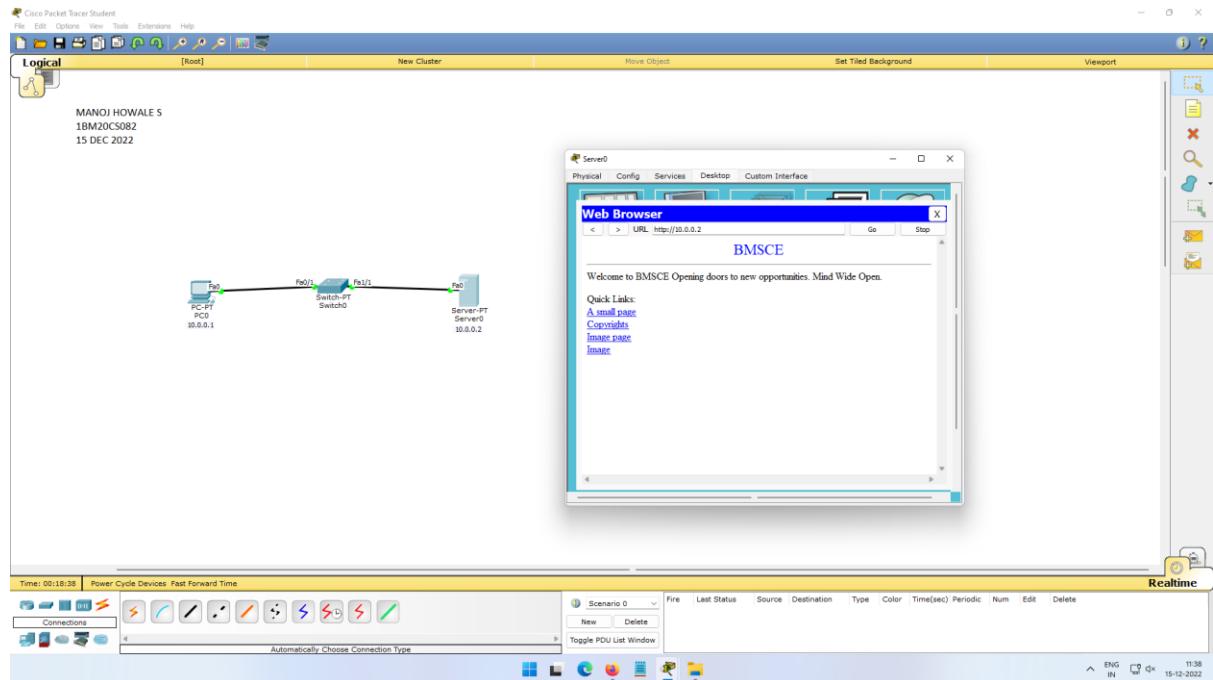
Learning Outcome: Using DNS it is easy to call websites with their names instead of IP address. DNS helps in naming conversion as computers are comfortable with IP address. It maps the name and its corresponding IP address.

Result

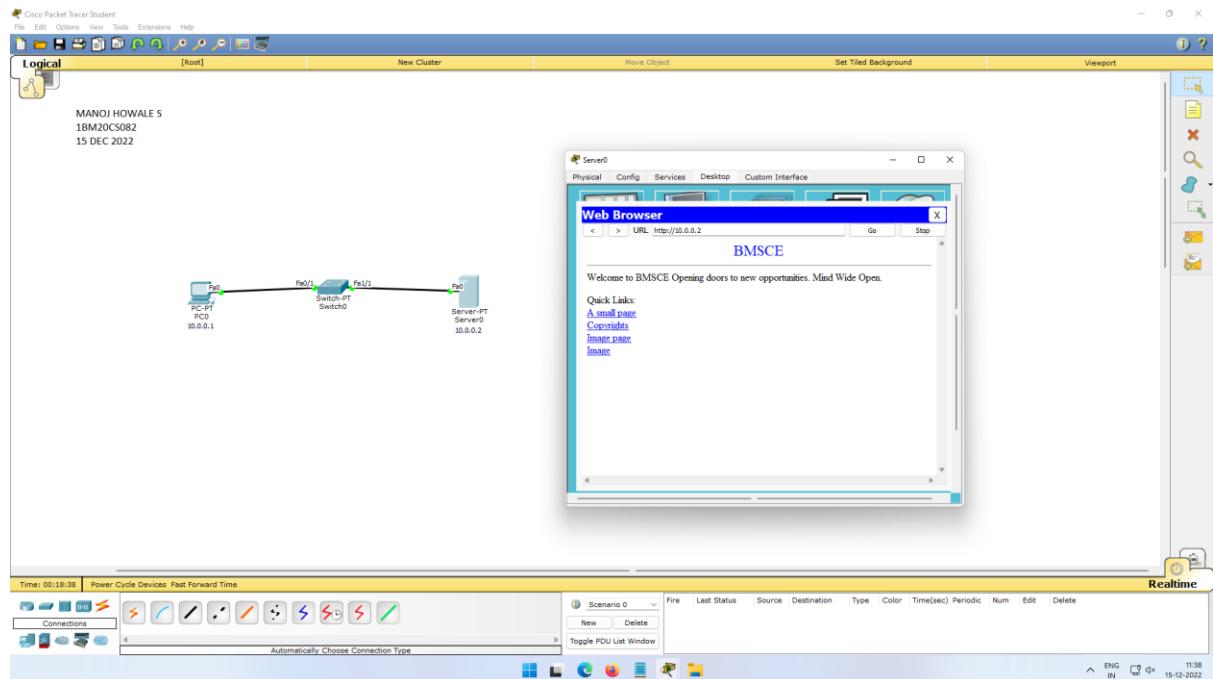


New
url

WEBSITE INTERFACE



RESUME INTERFACE



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 std::cin(gen-poly)
char data[28];
char check-value[28], genpoly[10];
int data-length, i, j;
void XOR() {
    for (j = 1; j < N; j++)
        check-value[j] = ((check-value[j] == genpoly[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 + j];
    } 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-data_length];
    printf ("Final data to send: %s", data);
    receiver();
    return 0;
}

```

Output 1:

Enter data to send: 10001000000100001

Enter generator polynomial: 1011101

Data added with $n-1$ zeros: 10001000000100001000000

CRC or check value is: 010011

Final data to be sent: 10001000000100001010011

Data received: 10001000000100001010011

No error ✓

Output 2:

Enter data to send: 10001000000100001

Enter generator polynomial: 1011101

Data added with $n-1$ zeros: 1000100000010000100000000

CRC or check value is: 010011

Final data to be sent: 10001000000100001010011

Data received: 10001000000100001010000

Error detected. ✓

~~WAN
29/12/2022~~

FINAL OUTPUT

```
Enter data to be transmitted: 10001000000100001
Enter the Generating polynomial: 1011101
Data padded with n-1 zeros : 10001000000100001000000
CRC or Check value is : 010011
Final data to be sent : 10001000000100001010011
Enter the received data: 10001000000100001010011
Data received: 10001000000100001010011
No error detected

...Program finished with exit code 0
Press ENTER to exit console.[]
```

5/1/23

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,
        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 is 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

```
PS D:\5th sem\Computer-Network-Lab\Lab8> cd "d:\5th sem\Computer-Ne
Enter a packet input: 10
Packets sent is 10
Remaining packets in bucket: 0
Do you want to input packet: y
Enter a packet input: 20
Packets sent is 10
Remaining packets in bucket: 10
Do you want to input packet: n
Packets sent is 10
Remaining packets in bucket: 0
Do you want to input packet: n
-----
Program ended
-----
PS D:\5th sem\Computer-Network-Lab\Lab8> █
```

Lab 9

Bellmann Ford Algorithm

Aim: find suitable path for transmission using Bellmann Ford algorithm.

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

int Bellman 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, k, 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 ($?) { g++ Untitled-1.cpp -o Untitled-1 } ; if ($?) { .\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
```

21/1/23

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("\nEnter 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 <= distance[i]) {
                    if (mindistance <= nextnode[i]) && distance[i] > mindistance + cost[nextnode][i]) {
                        distance[i] = mindistance + cost[nextnode][i];
                        pred[i] = nextnode;
                    }
                }
            }
        }
    }
}

```

```

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

```

Output:

Enter no. of vertices: 4

Enter adjacency matrix

.	.	.	.
.	0	1	1
,	1	0	1
1	1	1	0
,	1	0	1

Enter starting node : 1

Distance of 0 = 1

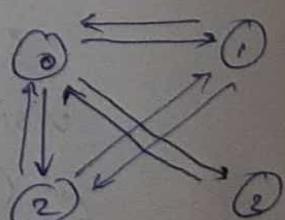
Path = 0 ← 1

Distance of 2 = 1

path = 2 ← 1

Distance of 3 = 2

path = 3 ← 0 ← 1



FINAL OUTPUT

```
PS C:\Users\Manoj> cd "d:\5th sem\Computer-Network-Lab\Lab9\DIJIKSTRA\" ; if ($?) { g++ dijikstra.cpp -o dijikstra } ; if ($?) { .\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('Enter filename: ')
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(2048).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 receive")
```

```
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file = open(sentence, 'r')
l = file.read(1024)
connectionSocket.send(l.encode())
print('Content of ' + sentence)
file.close()
connectionSocket.close()
```

Output :

Enter filename : Server TCP.py

From Server :

connectionSocket, addr = serverSocket.accept()

These are the contents of file.

FINAL OUTPUT

The image shows two separate Command Prompt windows side-by-side. Both windows have a title bar labeled "Command Prompt - python".

Left Window (Server Side):

```
D:\5th sem\Computer-Network-Lab\Lab10\TCP>python -u server.py
The server is ready to receive
Sent contents of file.txt
The server is ready to receive
|
```

Right Window (Client Side):

```
D:\5th sem\Computer-Network-Lab\Lab10\TCP>python -u client.py
Enter file name: file.txt
From Server:
this is a file
D:\5th sem\Computer-Network-Lab\Lab10\TCP>
```

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(sentence.encode("UTF-8"),  
(servername, serverport))
```

```
filecontents, serveraddress = clientSocket.recvfrom(2048)
```

```
print("Reply from server")
```

```
print(filecontents.decode("utf-8"))
```

```
clientSocket.close()
```

```
clientSocket.close()
```

```
from socket import *
```

```
serverPort = 12000
```

```
serverSocket = socket(AF_INET, SOCK_DGRAM)
```

```
serverSocket.bind(("127.0.0.1", serverPort))
```

```
print("Server is ready to receive")
```

```
while 1:
```

```
    sentence, clientAddress = serverSocket.recvfrom(2048)
```

```
    sentence = sentence.decode("utf-8")
```

```
    file = open(sentence, "r")
```

```
l = file.read(2048)
serverSocket.sendto(l, clientAddress)
print('sent content of file.')
print(sentence)
file.close()
```

Output

Enter filename: file.txt

Reply from server:

~~This is a file.~~

FINAL OUTPUT

The image shows two separate Command Prompt windows side-by-side. Both windows have a title bar labeled "Command Prompt".

Left Window (Server Side):

```
D:\5th sem\Computer-Network-Lab\Lab10\UDP>python -u server.py
The server is ready to receive
Sent contents of  file.txt
```

Right Window (Client Side):

```
D:\5th sem\Computer-Network-Lab\Lab10\UDP>python -u client.py
Enter file name: file.txt
Reply from Server:
this is a file
D:\5th sem\Computer-Network-Lab\Lab10\UDP>
```