

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

COMPUTER NETWORKS

Submitted by

Vidhaan Appaji (1BM20CS185)

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 “**COMPUTER NETWORKS**” carried out by **Vidhaan Appaji(1BM20CS185)**, who is 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.Nandini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

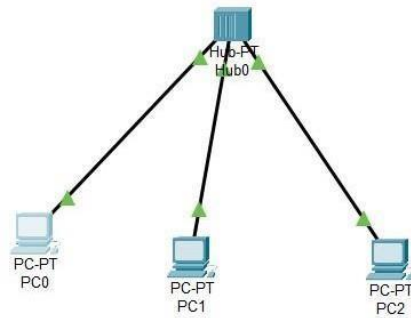
Index

Sl. No.	Date	Experiment Title	Page No.
1		Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.	
2		Configuring IP address to Routers in Packet Tracer. Exploring the following messages: Ping Responses, Destination unreachable, Request timed out, Reply	
3		Configuring default route to the Router	
4		Configuring DHCP within a LAN in a packet Tracer	
5		Configuring default router to router	
6		Configuring RIP Routing Protocol in Routers	
7		Demonstration of WEB server and DNS using Packet Tracer	
8		Write a program for error detecting code using CRC-CCITT (16-bits).	
9		Write a program for distance vector algorithm to find suitable path for transmission.	
10		Implement Dijkstra's algorithm to compute the shortest path for a given topology.	
11		Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.	
12		Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.	

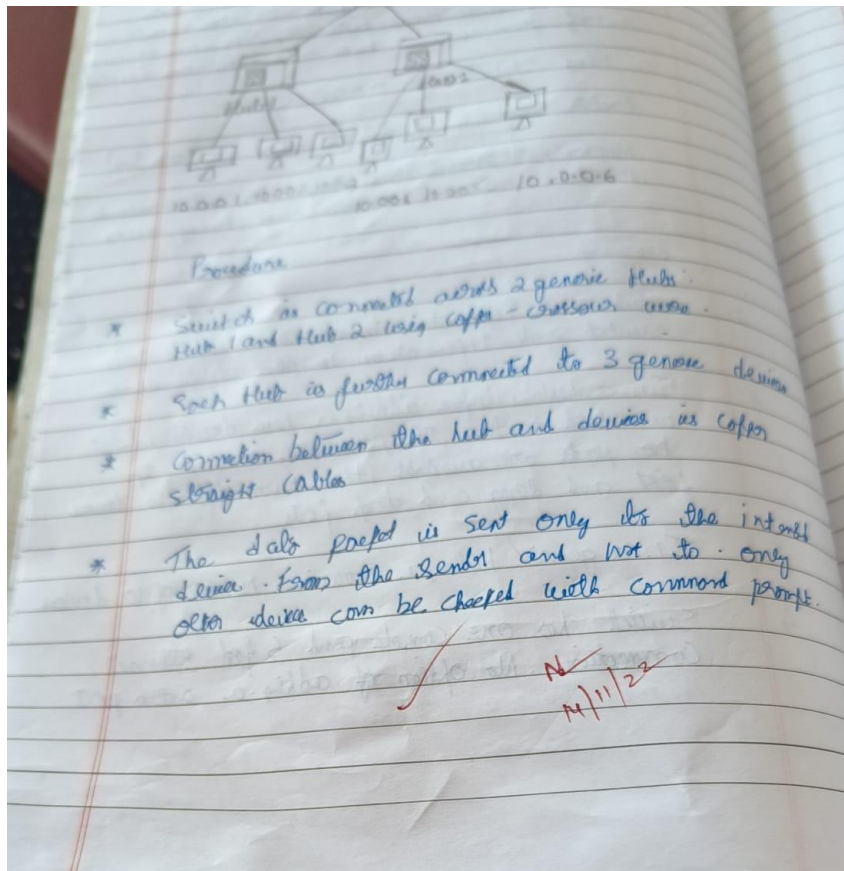
Cycle-1 Experiment No 1 Aim of the program

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

Hub Topology



Procedure



Output

```
Command Prompt

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.1

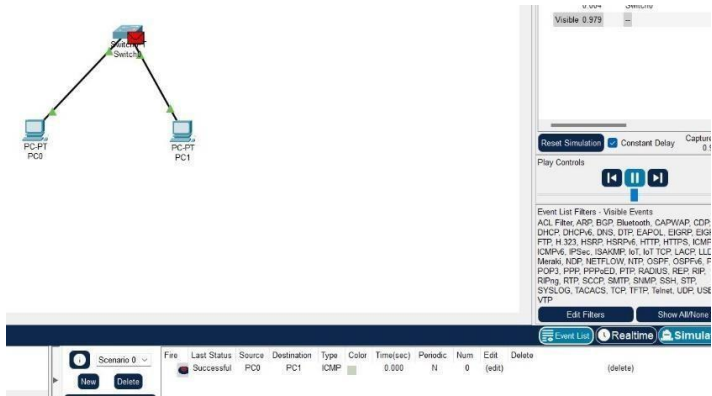
Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=3ms TTL=128
Reply from 10.0.0.1: bytes=32 time=4ms TTL=128
Reply from 10.0.0.1: bytes=32 time<1ms TTL=128
Reply from 10.0.0.1: bytes=32 time=3ms TTL=128

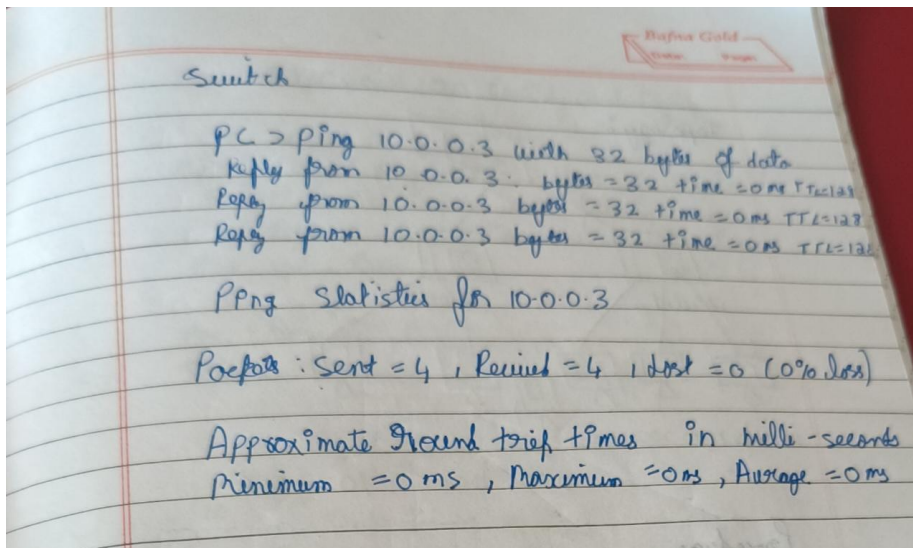
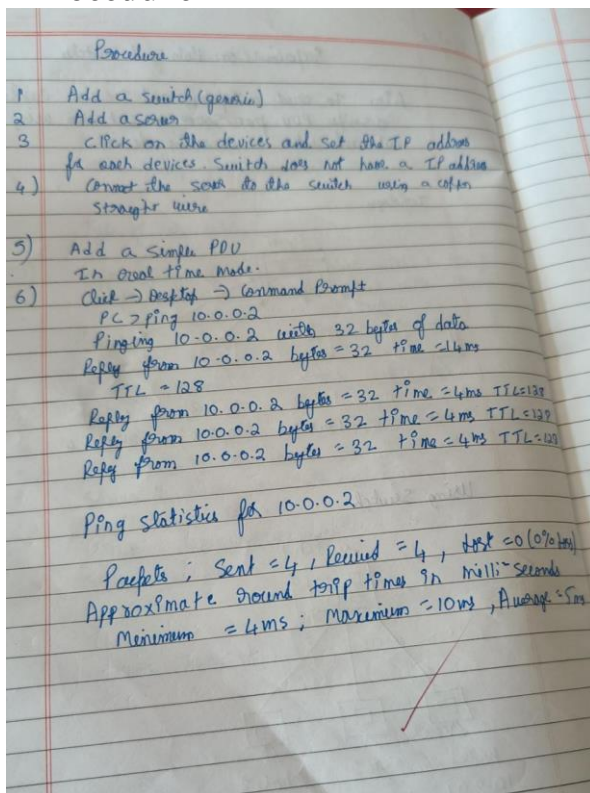
Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 4ms, Average = 2ms

C:\>
```

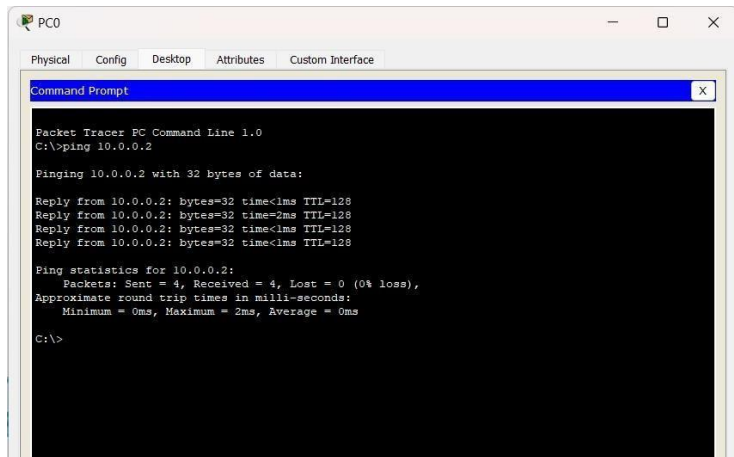
Switch Topology



Procedure

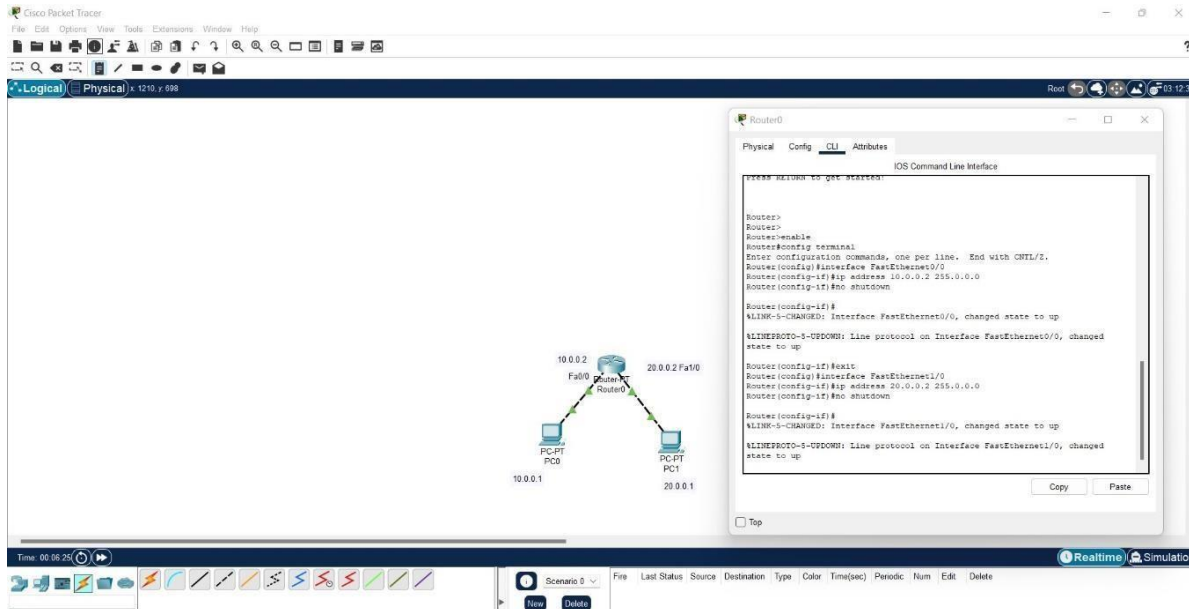


Output



Configuring IP address to Routers in Packet Tracer. Exploring the following messages: Ping Responses, Destination unreachable, Request timed out, Reply.

Topology



Procedure

1. Insert 3 routers and 2 end-devices
2. Set the IP address of the end-devices as shown in the topology
3. Connect the end-devices to the routers using copper cross-over. Connect the routers to each other using serial DCE
- 4) Click on the routers and go to the CLI to note the router id on both the other two routers

5) Run the following commands on the CLI
To configure routers

```
[config? : y/n] n  
Router > enable  
Router # config t  
Router [config] int e0/0/0  
Router # (config-if) ip
```

ip address 10.0.0.10 255.0.0.0

Configure the other routers using the above commands

6. After this, we will implement the static route using some commands. ~~except~~ except, we ~~do~~ let the connection with the end device and router connected to the other routers

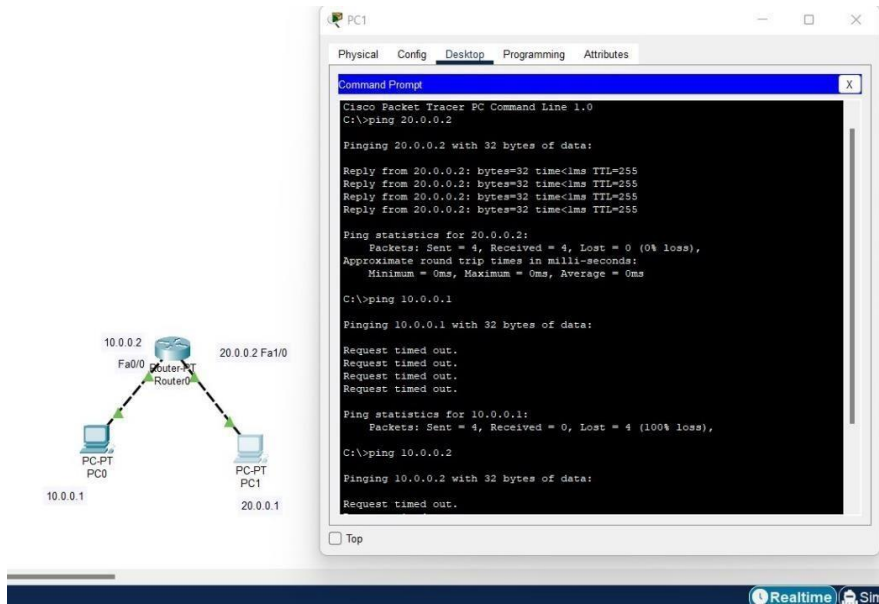
7. The following commands are used to do the step

```
Router # config t  
Router # [config] ip route 10.0.0.0 20.0  
Router # [config] exit  
Show ip route.
```

Observation

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 =

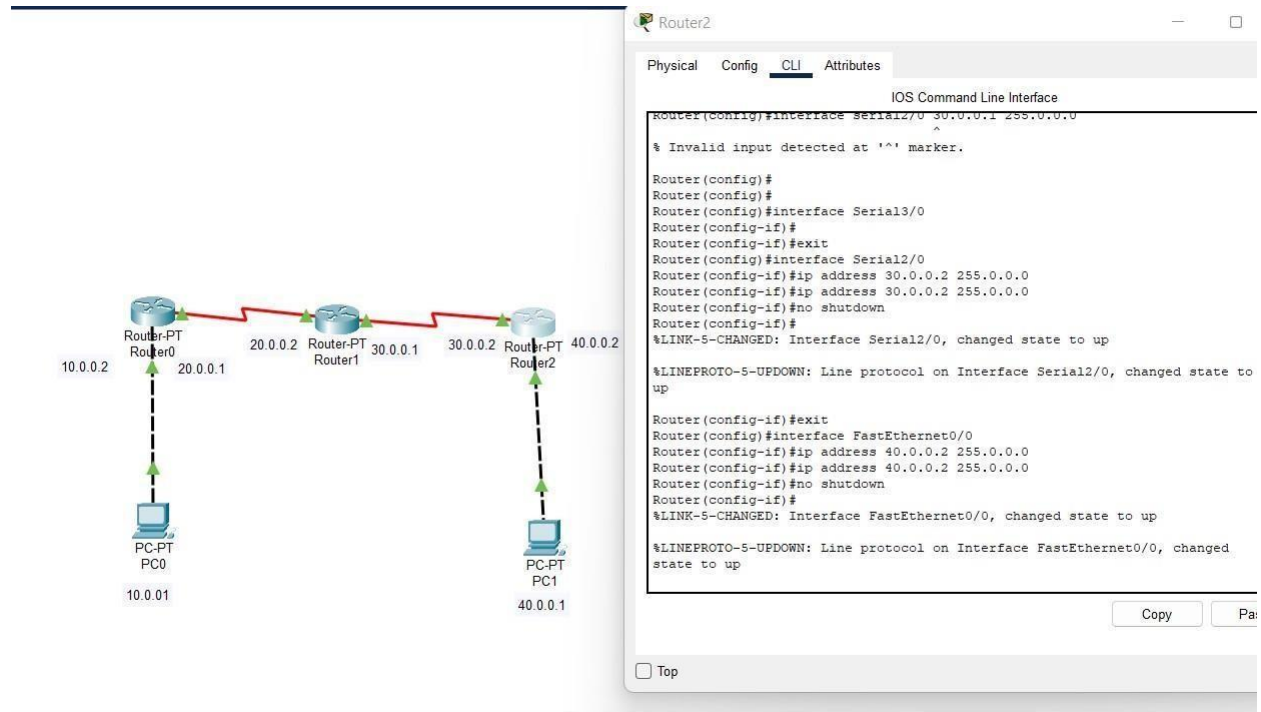
Output



Experiment No 3 Aim of the program

Configuring default route to the Router

Topology



Procedure

Procedure

Add 3 routers, 3 switches, each router is connected to a switch through a copper-straight, 3 routers are connected with each other through Serial DCE

Add 4 PCs connected to the switch and also this for other switches too through copper straight

Assign the IP address as shown in the above topology

Click on the router and go to the (CLI) mode. The router has also the other two routers through default routing.

use the following commands on the CLI

```

Conf # ip 10.0.0.1
Router # config t
Router (config) # int fastethernet 0/10
Router (config) # ip address 10.0.0.10
Router (config) # no shut.
Router (config) # exit
Router (config) # for all the Routers
Router (config) # ip address 20.0.0.10
Router (config) # exit
Router (config) # ip address 20.0.0.10
Router (config) # exit

```

5) After this we will implement the default routing using the following commands

For Router 1

```

Router # config t

```

```

Router (config) # int se2/0
Router (config) # ip address 0.0.0.0 0.0.0.0
Router (config) # exit

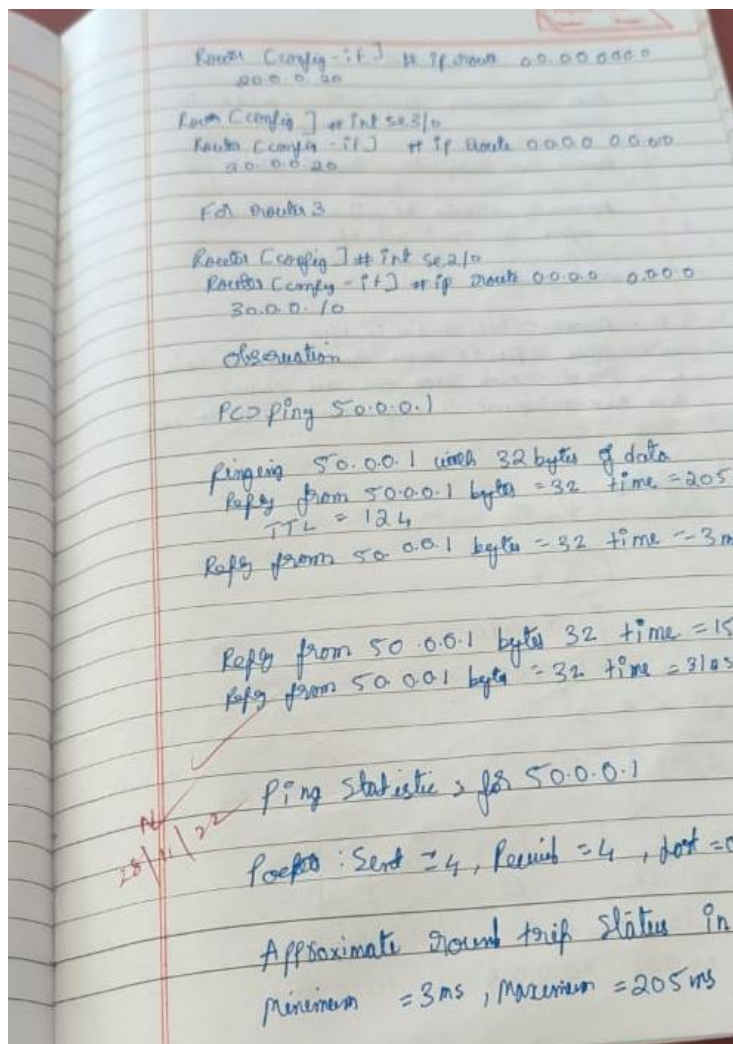
```

For Router 2

```

Router # config t
Router (config) # int se3/0

```

Output

```
Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

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

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

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=10ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125

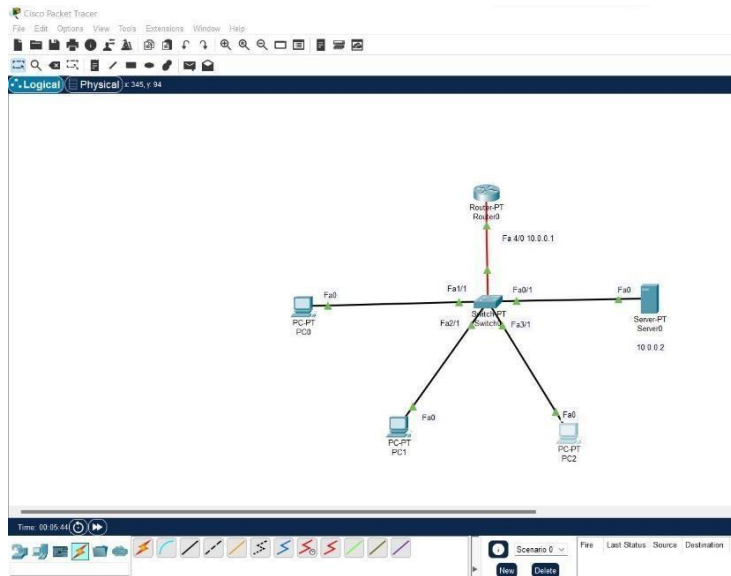
Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 10ms, Maximum = 10ms, Average = 10ms

C:\>
```

Experiment No 4 Aim of the program

Configuring DHCP within a LAN in a packet Tracer

Topology



Procedure

Procedure

- 1) Add a router, switch and server
- 2) Add 3 PCs and server connected to switch through a straight cable, switch is connected to router
- 3) Click on Server → configure the gateway 10.0.0.10 and fast Ethernet 0 to IP 10.0.0.1
- 4) Then click on Desktop → IP configuration make it dynamic. This will make the PC's to request for the dynamic address from server
- 5) Click on each PC → Desktop → IP configuration → dynamic
Repeat this for all PC's
As we change this from static from dynamic we get a message that "DHCP request Successful"

Observation from 10.0.0.2

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 = 128

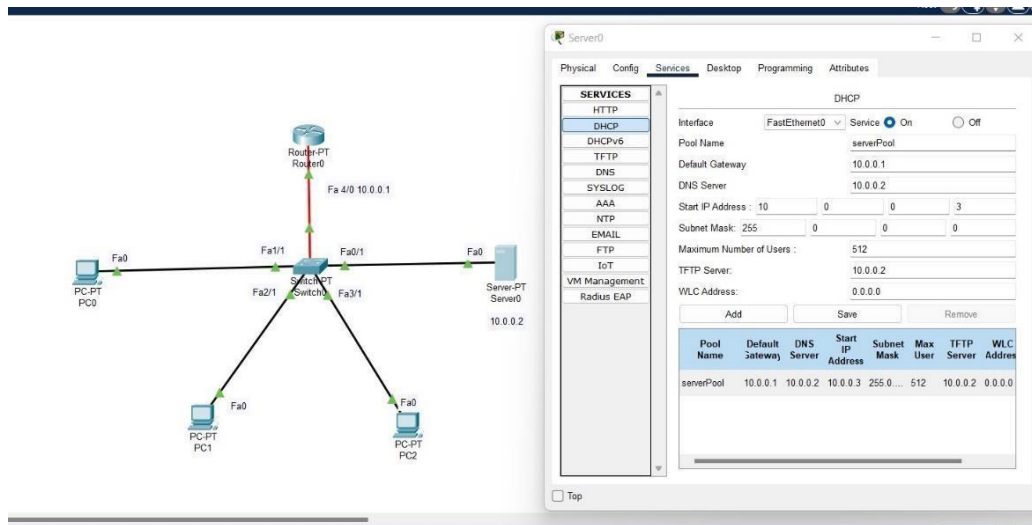
Reply from 10.0.0.1: bytes = 32 time = 1ms
TTL = 128

Reply from 10.0.0.1: bytes = 32 time = 0ms TTL = 128
Reply from 10.0.0.1: bytes = 32 time = 0ms
TTL = 128

Ping Statistics for 10.0.0.1

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milliseconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms

8/12/22



Output

```
PC0
Physical Config Desktop Attributes Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.6

Pinging 10.0.0.6 with 32 bytes of data:

Reply from 10.0.0.6: bytes=32 time=1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128

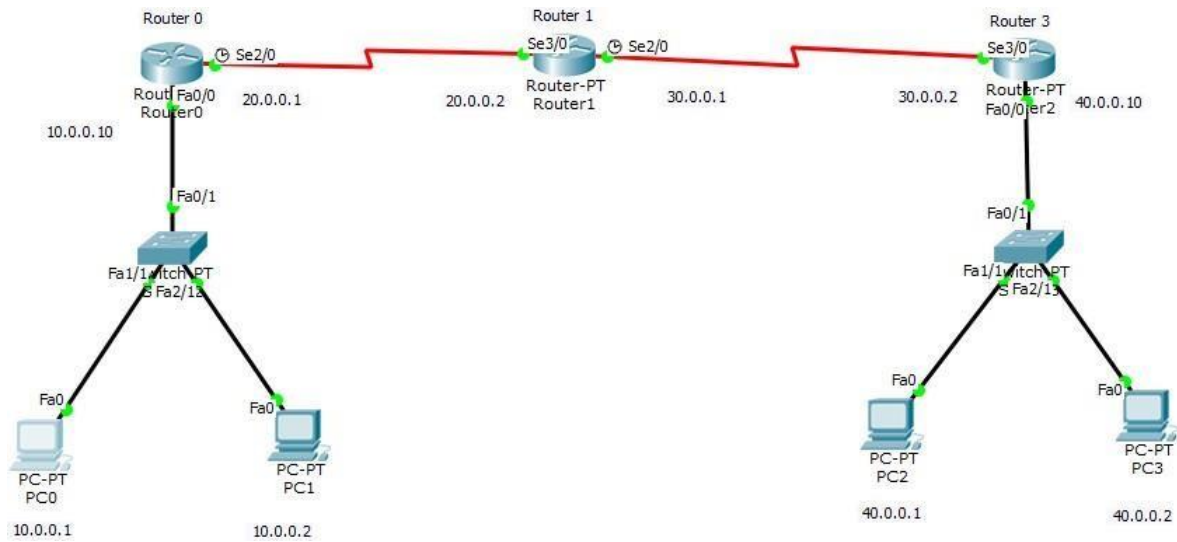
Ping statistics for 10.0.0.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

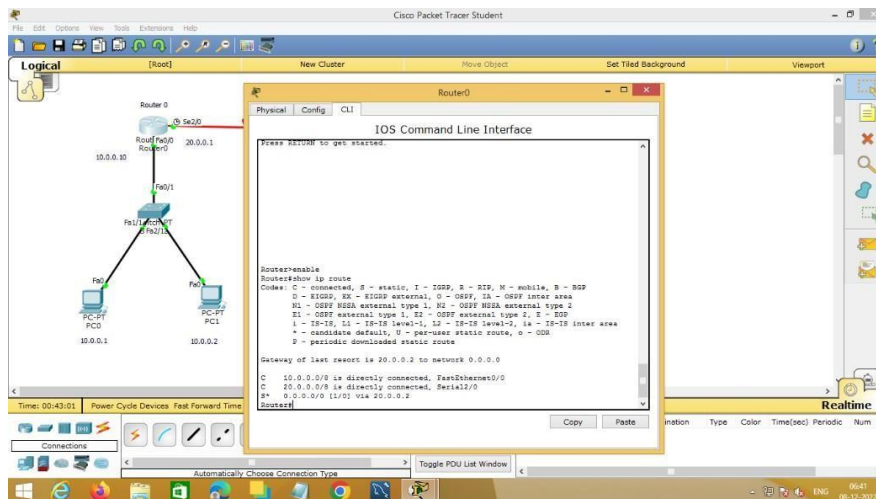
Experiment No 5 Aim of the program

Configuring default router to router

Topology



Procedure



Procedure:

1. Insert 3 routers and 2 end-devices
2. Set the IP address of the end devices as shown in the topology
3. Connect the end devices to the routers using copper cross-over. Connect the routers to each other using serial DCE
- 4) Click on the routers and go to the CLI to note the router learn about the other two routers

5) Write the following commands on the CLI

To configure router

```
[conf # : y/n]
```

```
Router > enable
```

```
Router # conf t
```

```
Router [config] interface fastEthernet 0/0
```

```
Router # (config-if)
```

```
IP address 10.0.0.10 255.0.0.0
```

Configure the other routers using the above commands

6. After this, we will implement the static routing using some commands. ~~except~~ except, we ~~not~~ let the connection with the end device and router connected to the other routers

7. The following commands are used to do the step

```
Router # config t
```

```
Router # [config] ip route 10.0.0.0 20.0.0.0
```

```
Router # [config] exit
```

```
Show IP route
```

Observation

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 =

TTL = 125

Reply from 40.0.0.1: bytes = 32 time = 20ms
TTL = 125

Reply from 40.0.0.1: bytes = 32 time = 20ms

Reply from 40.0.0.1: bytes = 32 time = 4ms TTL = 20

Ping statistics for 40.0.0.1

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

Approximate round trip time in milli-seconds -

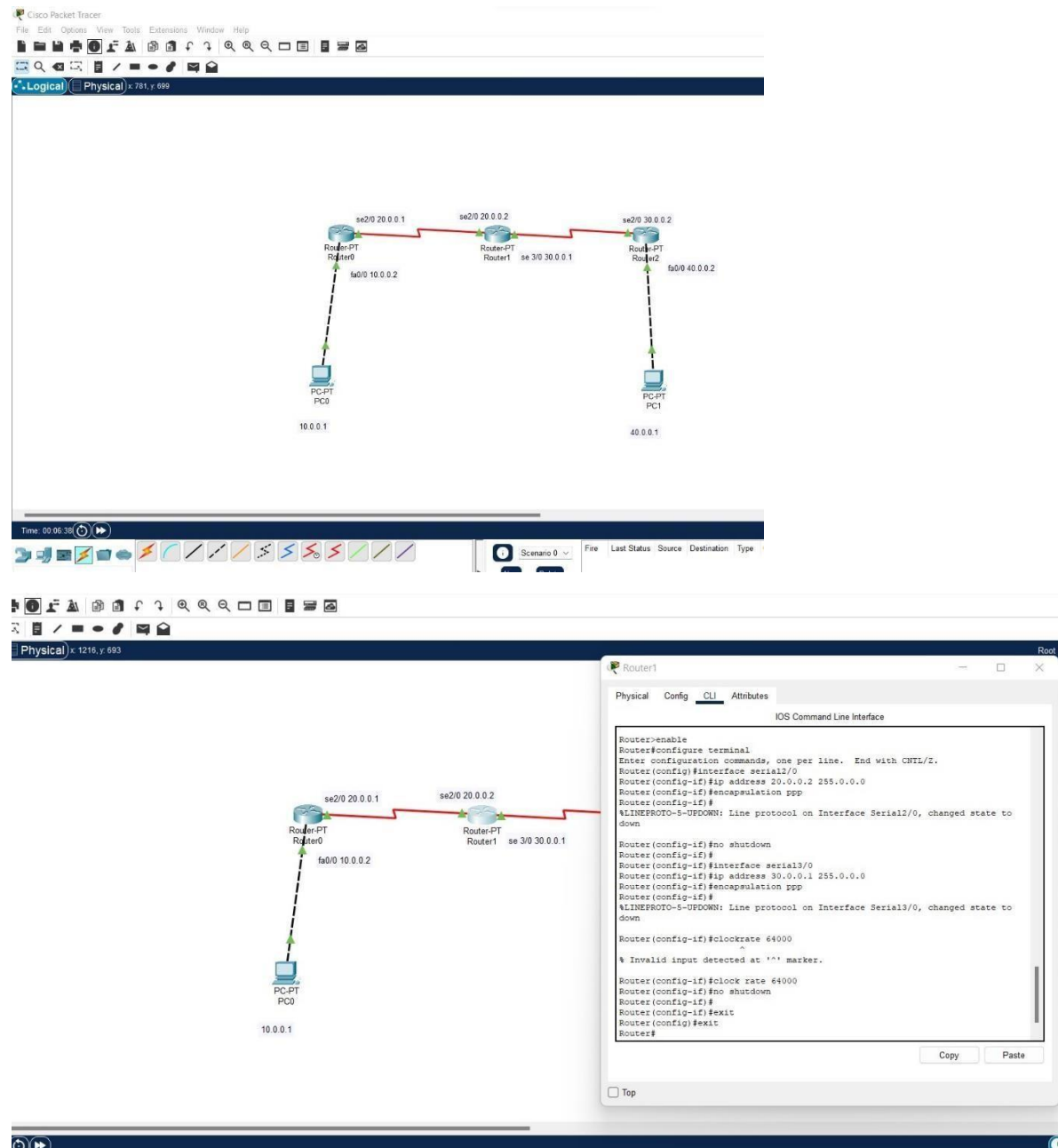
Minimum = 2ms, Maximum = 20ms, Average = 7ms

~~28/11/22~~

Experiment No 6 Aim of the program

Configuring RIP Routing Protocol in Routers

Topology



Procedure

RIP - Routing Information Protocol

CIBR - classless Inter-Domain Routing

PPP - Point to Point Protocol

- 1- Insert 3 routers and 2 PC's, connect the routers through serial DCE cable and PC's with routers through copper cross-over

The IP addresses for the PC's were set to 10.0.0.1 and 40.0.0.1 for PC1 & PC2

4)

After the basic configuration
was done, the P1F has to be set up.

For router 1

config t

int se2/0

encapsulation PPP

clock rate 64000

no shut

router ospf

network 20.0.0.0

exit

5)

PC1 was ordered and 40.0.0.1 ping

Observation

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 = 18ms

TTL = 125

Reply from 40.0.0.1 bytes = 32 time = 12ms

TTL = 125

Reply from 40.0.0.1 bytes = 32 time = 14ms

TTL = 125

Reply from 40.0.0.1 bytes = 32 time =

TTL = 125

Packet sent = 4, Recd = 6, TTL = 0
LO = 100
Approximate Round-trip time in ms
Minimum = 12 ms Maximum = 12 ms Avg = 12 ms

Learning

RIP - Distance information Passed in of 2 types

RIP-1 and RIP-2 where RIP-2 supports
intra domain routing & RIP-1 doesn't

With knowledge of RIP, terminals can ping
each other without taking the router, since
the routers individually

12/12/22

Output

```
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=4ms TTL=125
Reply from 40.0.0.1: bytes=32 time=3ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125

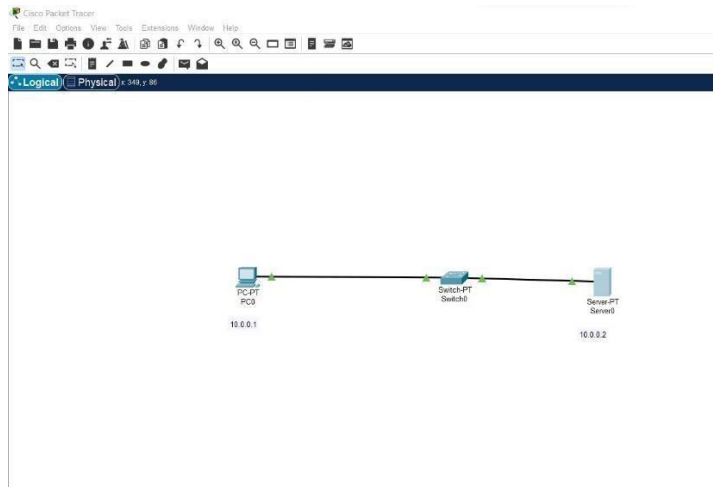
Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 4ms, Average = 3ms

C:\>
```

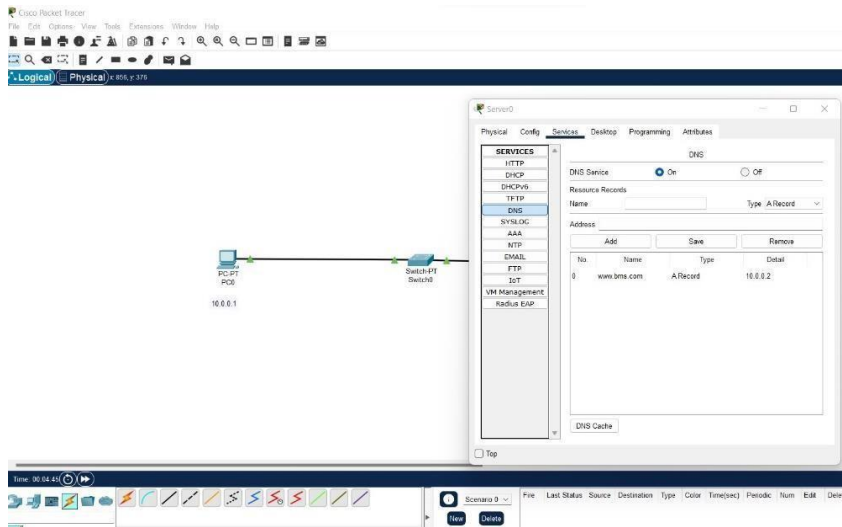
Experiment No 7 Aim of the program

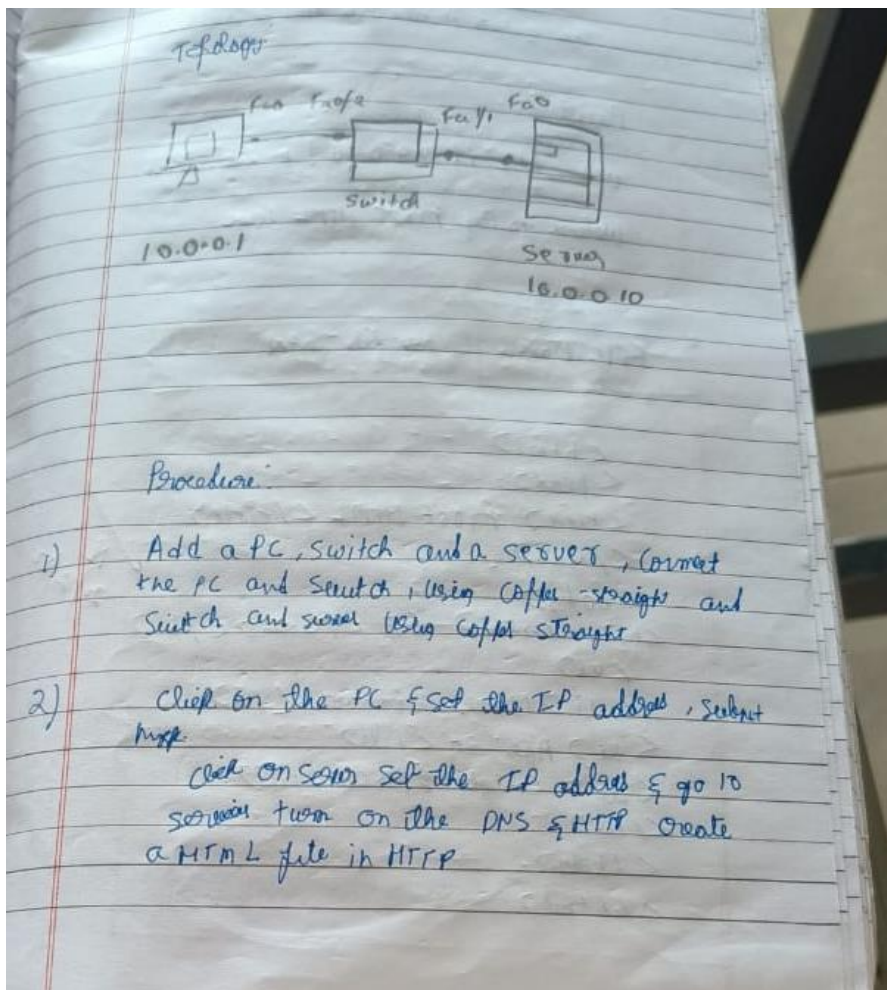
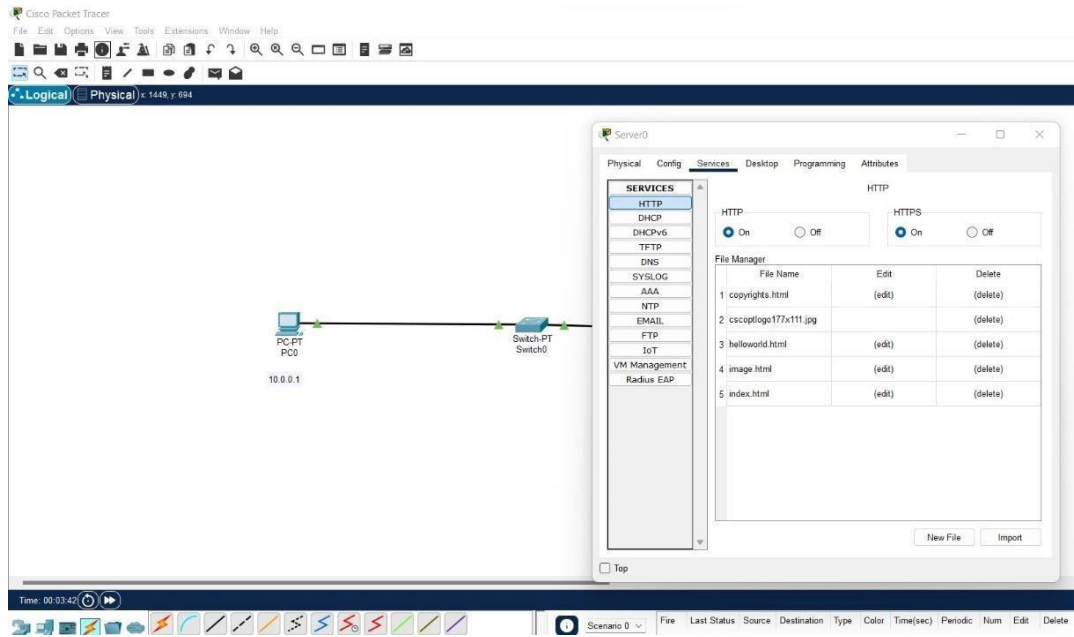
Demonstration of WEB server and DNS using Packet Tracer

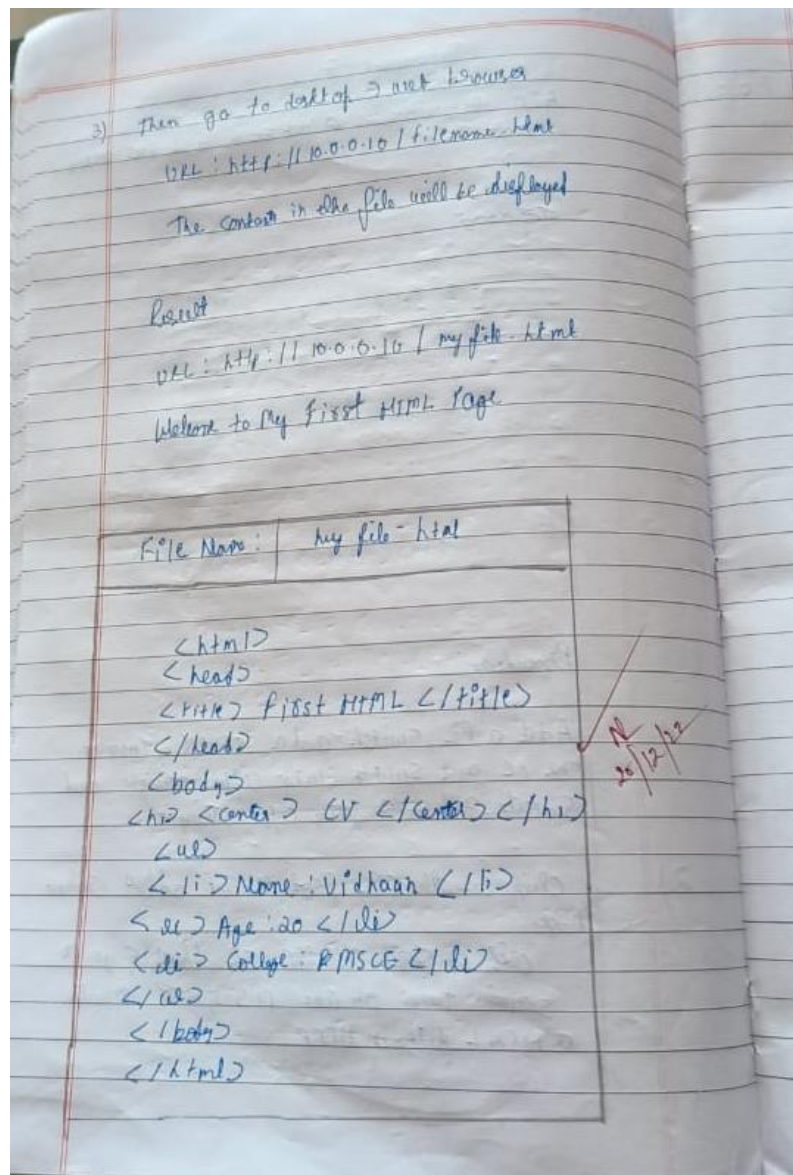
Topology



Procedure







Output



Cycle-2 Experiment No 1 Aim of the Experiment

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

Code

```
#include<bits/stdc++.h> using
namespace std; void receiver(string
data, string key);

string xor1(string a, string b)
{

    string result = "";

    int n = b.length();

    for(int i = 1; i < n; i++)
    {
        if (a[i] == b[i])
            result += "0";
        else
            result += "1";
    }
    return result;
}

string mod2div(string dividend, string divisor)
{
    int pick = divisor.length();
```



```

string tmp = dividend.substr(0, pick);

int n = dividend.length();

while (pick < n)
{
    if (tmp[0] == '1')                tmp =
xor1(divisor, tmp) + dividend[pick];
    else
        tmp = xor1(std::string(pick, '0'), tmp) +
            dividend[pick];

    pick += 1;
}

if (tmp[0] == '1')
tmp = xor1(divisor, tmp);
else
    tmp = xor1(std::string(pick, '0'), tmp);

return tmp;
}

void encodeData(string data, string key)
{
    int l_key = key.length();

    string appended_data = (data + std::string(l_key - 1, '0'));

    string remainder = mod2div(appended_data, key);

```



```

        string codeword = data + remainder;
    cout << "Remainder : "
           << remainder << "\n";
        cout << "Encoded Data (Data + Remainder) : "
           << codeword << "\n";
        receiver(codeword, key);
    }
    void receiver(string data, string key)
    {
        string currxor = mod2div(data.substr(0, key.size()), key);
        int curr = key.size();    while (curr != data.size())
        {
            if (currxor.size() != key.size())
            {
                currxor.push_back(data[curr++]);
            }
            else
            {
                currxor = mod2div(currxor, key);
            }
        }
        if (currxor.size() == key.size())
        {
            currxor = mod2div(currxor, key);
        }
        if (currxor.find('1') != string::npos)
        {
            cout << "there is some error in data" << endl;
        }
        else
    }

```

```

        {
            cout << "correct message recieved" << endl;
        }
    } int
    main()
    {

        string data = "1011101";
        string key = "100010000001";

        encodeData(data, key);

        return 0;
    }

```

```

Remainder : 10001011000
Encoded Data (Data + Remainder) :101110110001011000
correct message recieved

...Program finished with exit code 0
Press ENTER to exit console.

```

Experiment No 2 Aim of the Experiment

Write a program for distance vector algorithm to find suitable path for transmission.

Code

```
#include<stdio.h>

struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];

int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j]; //initialise the distance equal
to cost matrix
            rt[i].from[j]=j;
        }
    }

    do
    {
        count=0;
        for(i=0;i<nodes;i++)//We choose arbitrary vertex k and we
calculate the direct distance from the node i to k using the cost
matrix

        //and add the distance from k to node j
        for(j=0;j<nodes;j++)
        for(k=0;k<nodes;k++)
            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
            { //We calculate the minimum distance
                rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
            }
        }
    } while(count<nodes);
}
```

```

        rt[i].from[j]=k;
        count++;
    }
}while(count!=0);
for(i=0;i<nodes;i++)
{
    printf("\n\n For router %d\n",i+1);
    for(j=0;j<nodes;j++)
    {
        printf("\t\nnode %d via %d Distance %d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
    }
}
printf("\n\n");
getch();
}

```

Output

```

Enter the cost matrix :
0 1 5 6
1 0 3 4
5 3 0 2
6 4 2 0

For router 1
node 1 via 1 Distance 0
node 2 via 2 Distance 1
node 3 via 2 Distance 4
node 4 via 2 Distance 5

For router 2
node 1 via 1 Distance 1
node 2 via 2 Distance 0
node 3 via 3 Distance 3
node 4 via 4 Distance 4

For router 3
node 1 via 2 Distance 4
node 2 via 2 Distance 3
node 3 via 3 Distance 0
node 4 via 4 Distance 2

For router 4
node 1 via 2 Distance 5
node 2 via 2 Distance 4
node 3 via 3 Distance 2
node 4 via 4 Distance 0

...Program finished with exit code 0
Press ENTER to exit console.

```

Experiment No 3 Aim of the Experiment

Implement Dijkstra's algorithm to compute the shortest path for a given topology.

Code

```
#include<bits/stdc++.h>

#include <limits.h>
#include <stdio.h>
using namespace std;

#define V 5

int minDistance(int dist[], bool Test[])
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if ( Test[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

void printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool Test[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, Test[i] = false;
```

```

dist[src] = 0;

for (int count = 0; count < V - 1; count++) {

    int u = minDistance(dist, Test);

    Test[u] = true;

    for (int v = 0; v < V; v++)

        if (!Test[v] && graph[u][v] && dist[u] != INT_MAX
            && dist[u] + graph[u][v] < dist[v])
            dist[v] = dist[u] + graph[u][v];
}

printSolution(dist);
}

int main()
{

    int graph[V][V] ;
    cout<<"Enter the graph "<<endl;
    for(int i = 0; i<V; i++)
    {
        for(int j = 0; j<V; j++)
            cin>>graph[i][j];
    }

    dijkstra(graph, 0);

    return 0;
}

```

Output

```
Enter the graph
0 1 4 0 5
1 0 3 6 0
4 3 0 0 6
0 6 0 0 10
5 0 6 10 0
Vertex          Distance from Source
0               0
1               1
2               4
3               7
4               5
```

Experiment No 4 Aim of the Experiment

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

Code

Server:

```
from socket import *
serverName = ''
serverPort = 12530
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    try:
        file = open(sentence, "r")
        l = file.read(1024)
        connectionSocket.send(l.encode())
        file.close()
    except Exception as e:
        message = "No such file exist"
        connectionSocket.send(message.encode())
    connectionSocket.close()
```

Client: from socket import *

```
serverName = '192.168.1.104'
serverPort = 12530
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Enter file name")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)
clientSocket.close()
```


Experiment No 5 Aim of the Experiment

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

Code

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)

    file=open(sentence,"r")    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
print("sent back to client",l)    file.close() Client:
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 ('From Server:',
filecontents)

clientSocket.close()
```