VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPUTER NETWORKS

Submitted by

DINESH KUMAR G (1BM20CS043)

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 **DINESH KUMAR G (1BM20CS043),** 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.

Rajeshwari Maldi
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.	1
2		Configuring IP address to Routers in Packet Tracer. Exploring the following messages: Ping Responses, Destination unreachable, Request timed out, Reply	4
3		Configuring default route to the Router	6
4		Configuring DHCP within a LAN in a packet Tracer	9
5		Configuring RIP Routing Protocol in Routers	11
6		Demonstration of WEB server and DNS using Packet Tracer	14
7		Write a program for error detecting code using CRC-CCITT (16-bits).	16
8		Write a program for distance vector algorithm to find suitable path for transmission.	21
9		Implement Dijkstra’s algorithm to compute the shortest path for a given topology.	25
10		Write a program for congestion control using Leaky bucket algorithm	29
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.	31
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.	34

Cycle-1

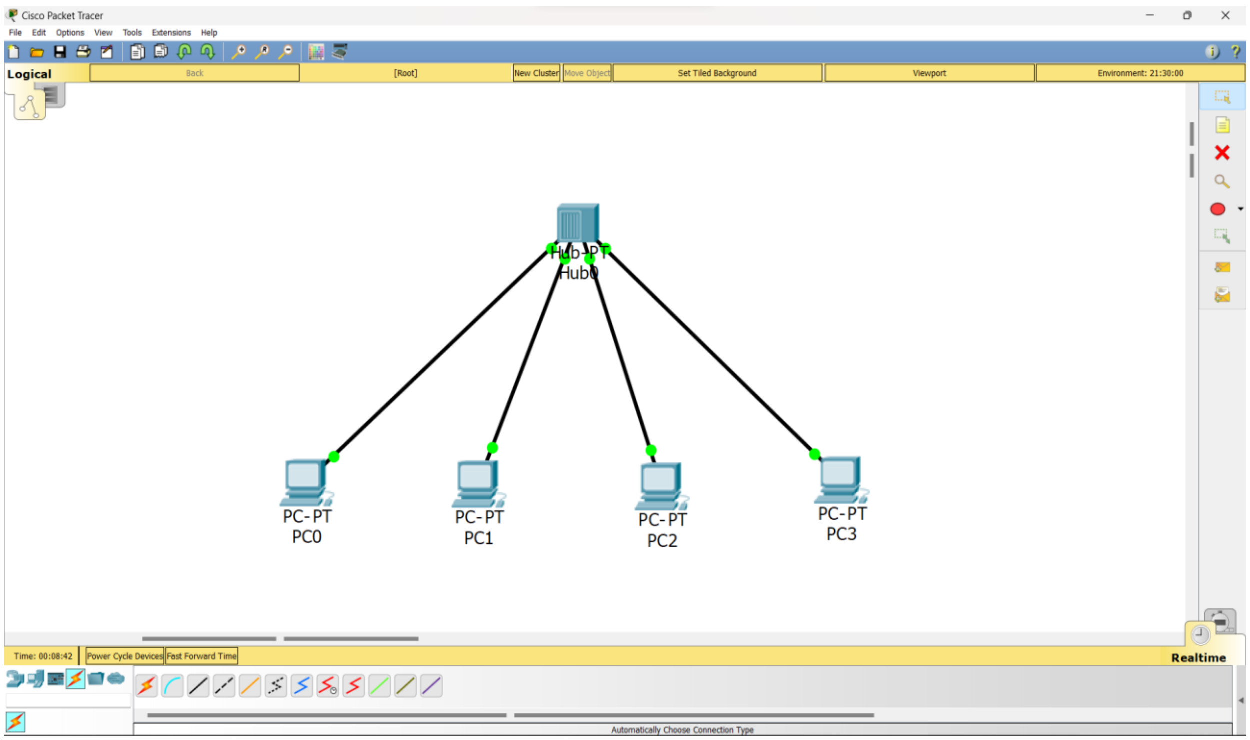
Experiment No 1

Aim

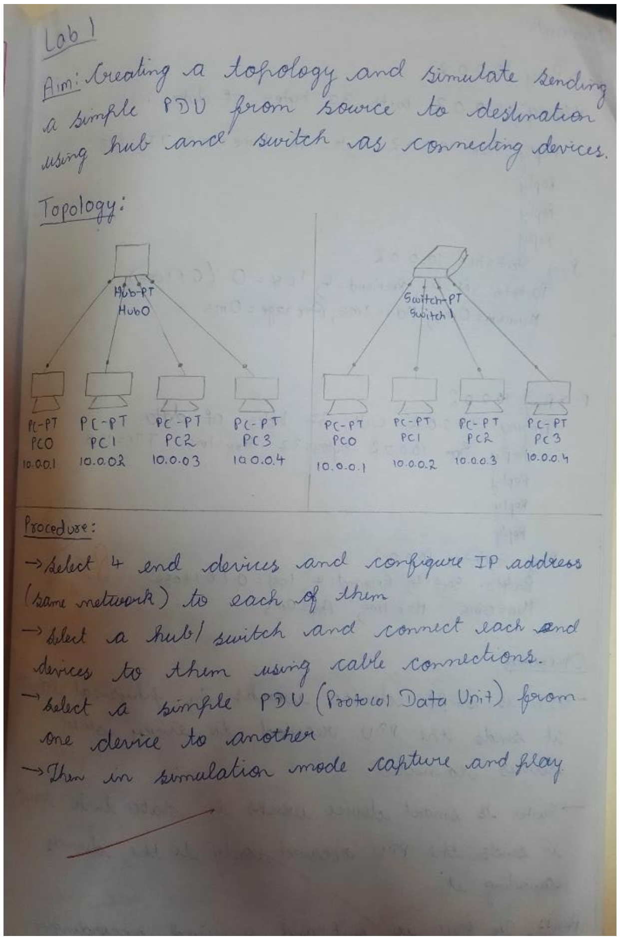
Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.

Hub

Topology



Procedure


Lab 1

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

Topology:

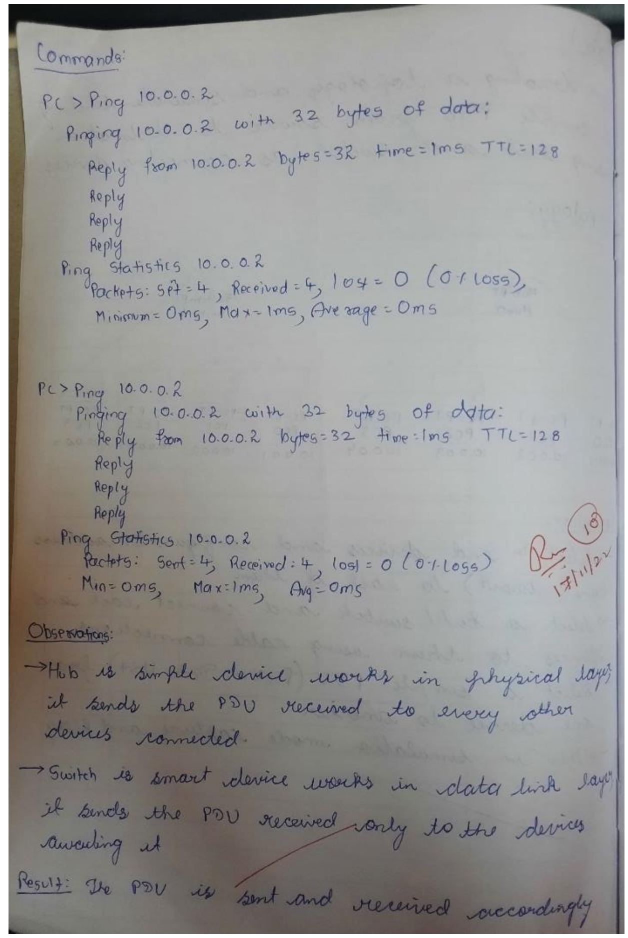
Procedure:

→ Select 4 end devices and configure IP address (same network) to each of them

→ Select a hub/switch and connect each end devices to them using cable connections.

→ Select a simple PDU (Protocol Data Unit) from one device to another

→ Then in simulation mode capture and play


Commands:

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 TTL=128

Reply

Reply

Reply

Ping Statistics 10.0.0.2

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

Minimum=0ms, Max=1ms, Average=0ms

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 TTL=128

Reply

Reply

Reply

Ping Statistics 10.0.0.2

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

Min=0ms, Max=1ms, Avg=0ms

10

17/1/22

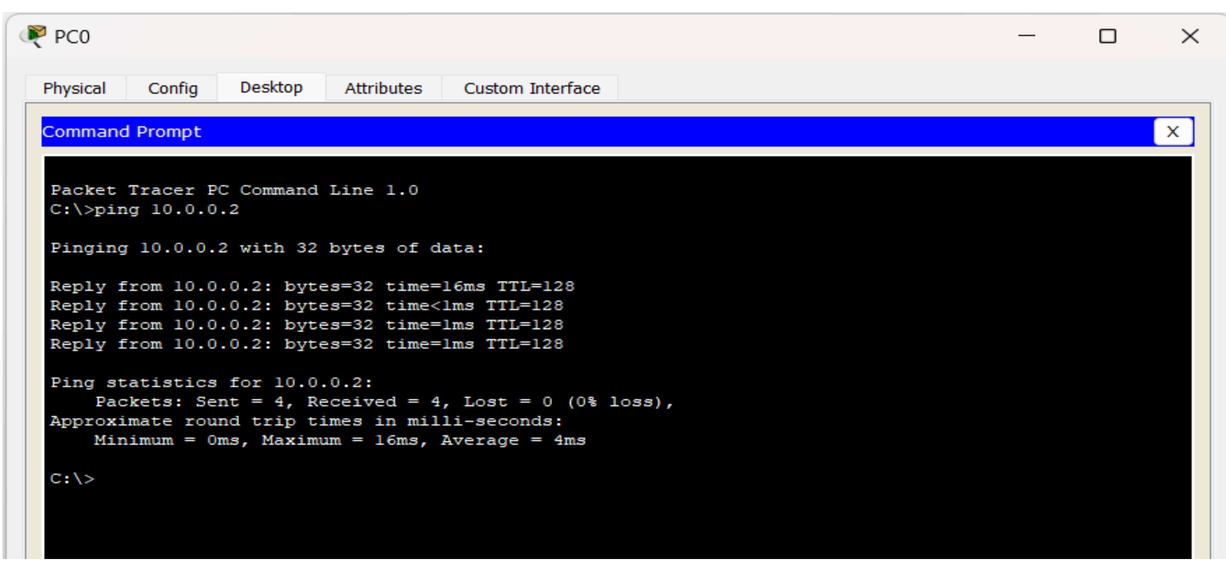
Observations:

→ Hub is simple device works in physical layer it sends the PDU received to every other devices connected.

→ Switch is smart device works in data link layer it sends the PDU received only to the devices awaiting it

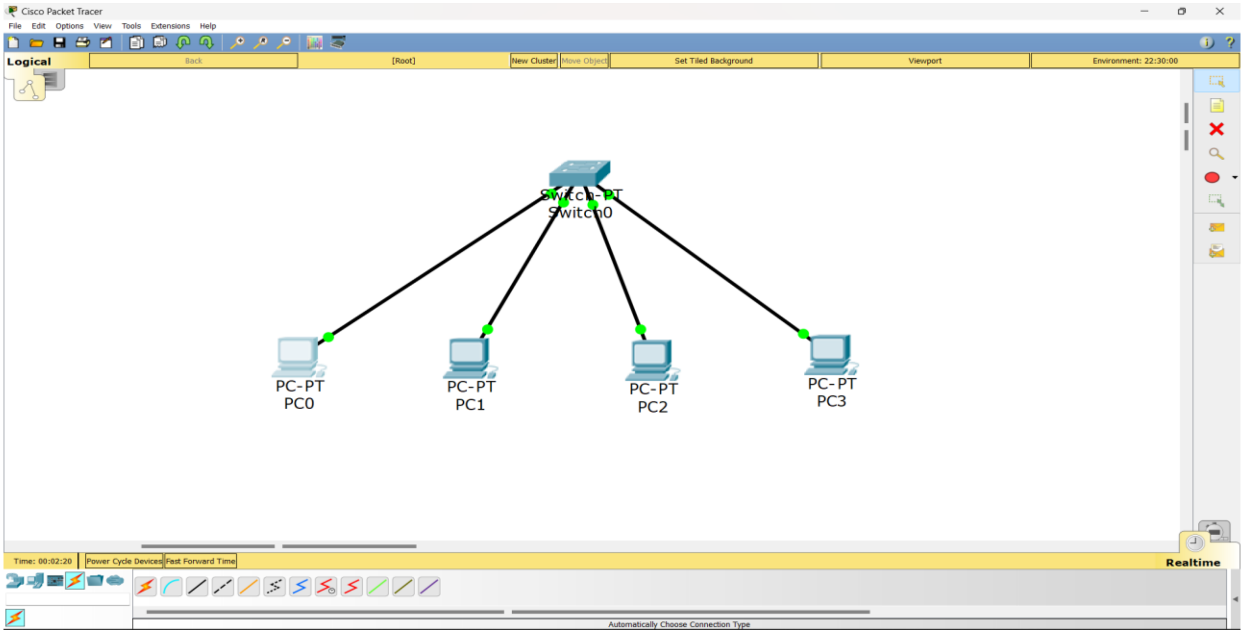
Result: The PDU is sent and received accordingly

Output

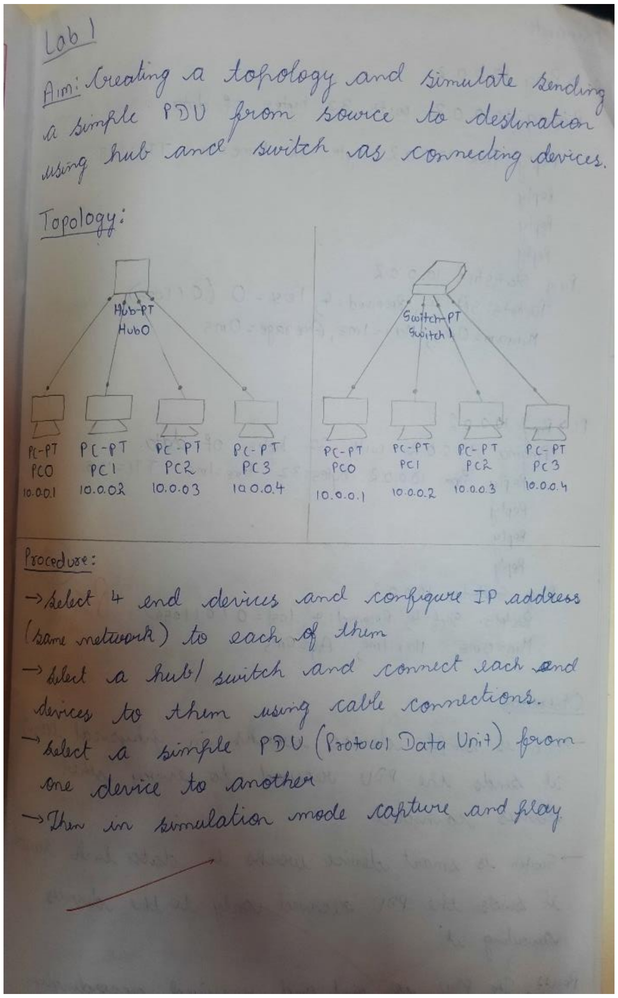


Switch

Topology



Procedure



Lab 1

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

Topology:

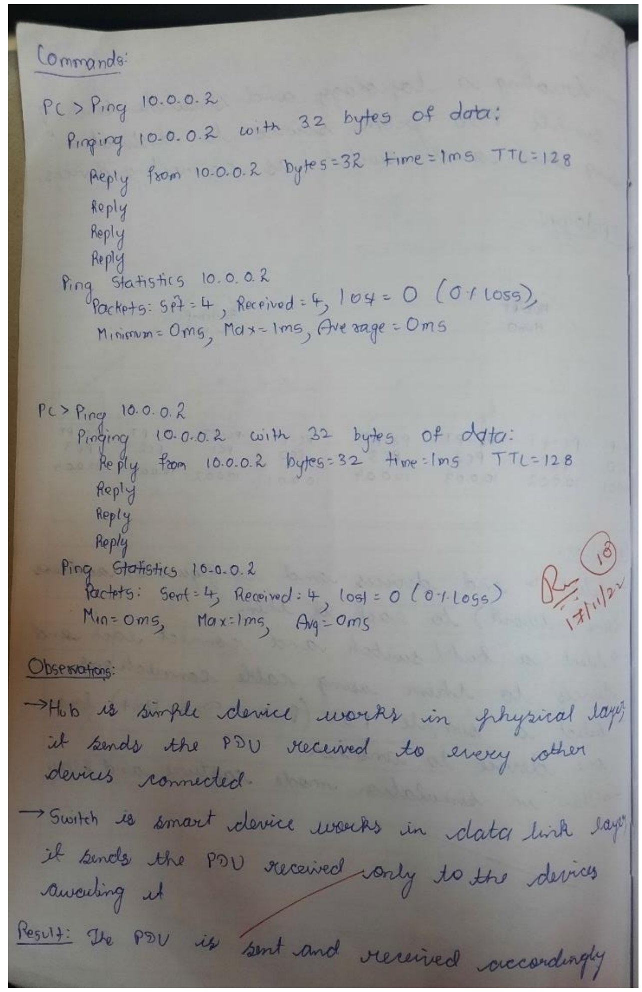
Procedure:

→ Select 4 end devices and configure IP address (same network) to each of them

→ Select a hub/switch and connect each end devices to them using cable connections.

→ Select a simple PDU (Protocol Data Unit) from one device to another

→ Then in simulation mode capture and play



Commands:

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 TTL=128

Reply

Reply

Reply

Ping Statistics 10.0.0.2

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

Minimum = 0ms, Max = 1ms, Average = 0ms

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 TTL=128

Reply

Reply

Reply

Ping Statistics 10.0.0.2

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

Min = 0ms, Max: 1ms, Avg = 0ms

10 17/11/22

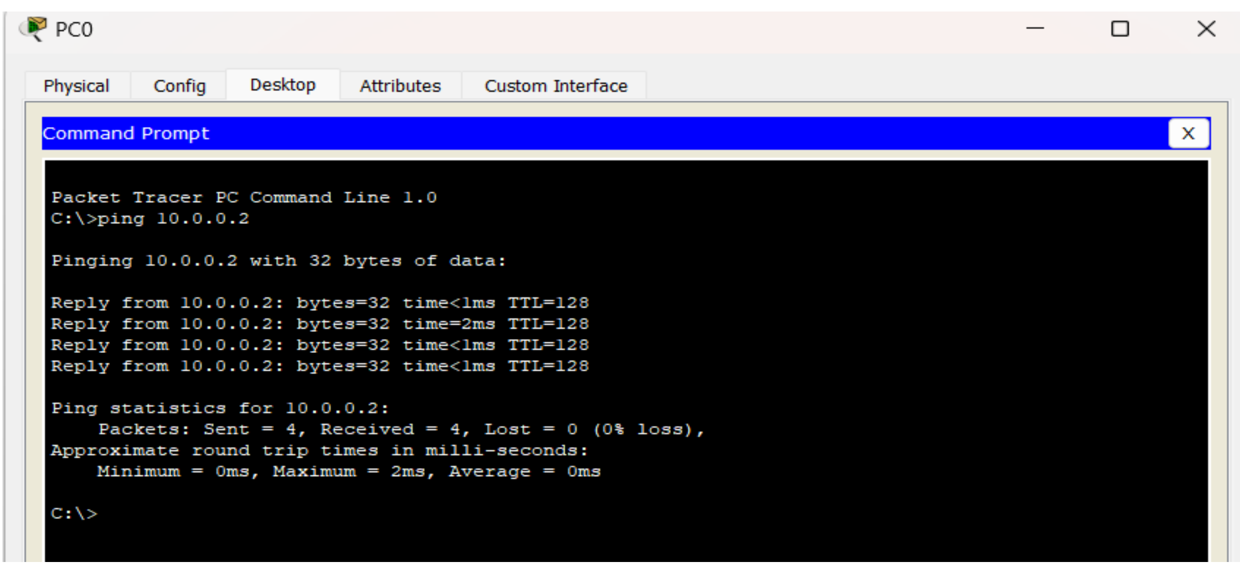
Observations:

→ Hub is simple device works in physical layer it sends the PDU received to every other devices connected.

→ Switch is smart device works in data link layer it sends the PDU received only to the devices awaiting it

Result: The PDU is sent and received accordingly

Output



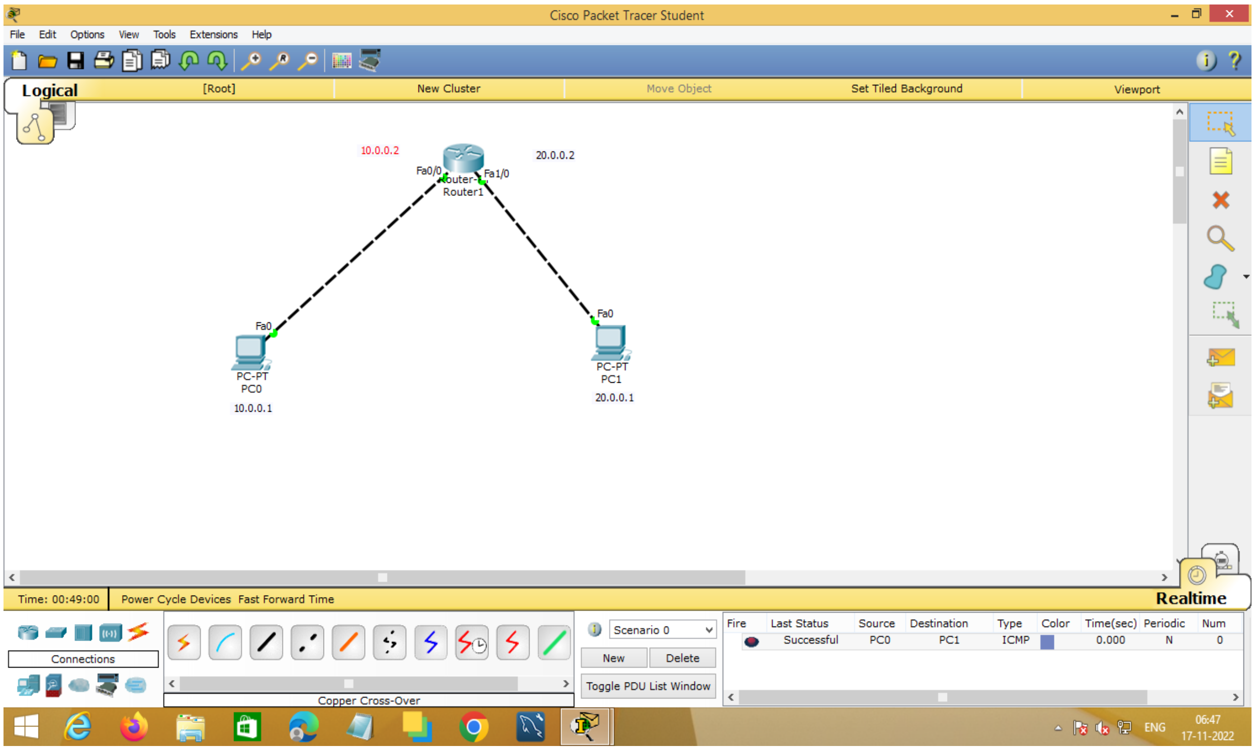

```
Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time<1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=2ms TTL=128
Reply from 10.0.0.2: bytes=32 time<1ms TTL=128
Reply from 10.0.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\>
```

Experiment No 2

Aim

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

Topology

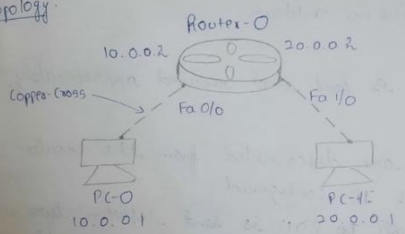


Procedure

Lab 2

Aim: To configure IP address to Routers in Packet tracer. Explore following messages: Ping responses, destination unreachable, Request timed out, Reply.

Topology:



Procedure:

→ Select two end devices and configure IP addresses (different network)

→ Select a router and connect the devices as shown in figure.

→ Configure the IP for router as per the commands.

→ Configure the gateway for two devices (gateway = IP addresses of router for same network)

→ Send a simple PDU between two devices before and after configuring the gateway for the devices.

Commands:

Router > enable

> Router # Configure terminal

> Router (config) # interface Fa 0/0

> Router (config-if) # ip address [ip] [subnet mask]

> Router (config-if) # no shutdown

> exit

Results: The PDU is sent and received appropriately.

Observations:

→ The devices are disconnected from the router until it is IP configured

→ When a simple IP PDU is sent between two devices, request time out occurs because the gateways aren't configured, i.e, the two networks are isolated.

→ When the gateways are configured and a simple PDU, it is received appropriately at the other network

Output

PC0

Physical | Config | Desktop | Attributes | Custom Interface

Command Prompt

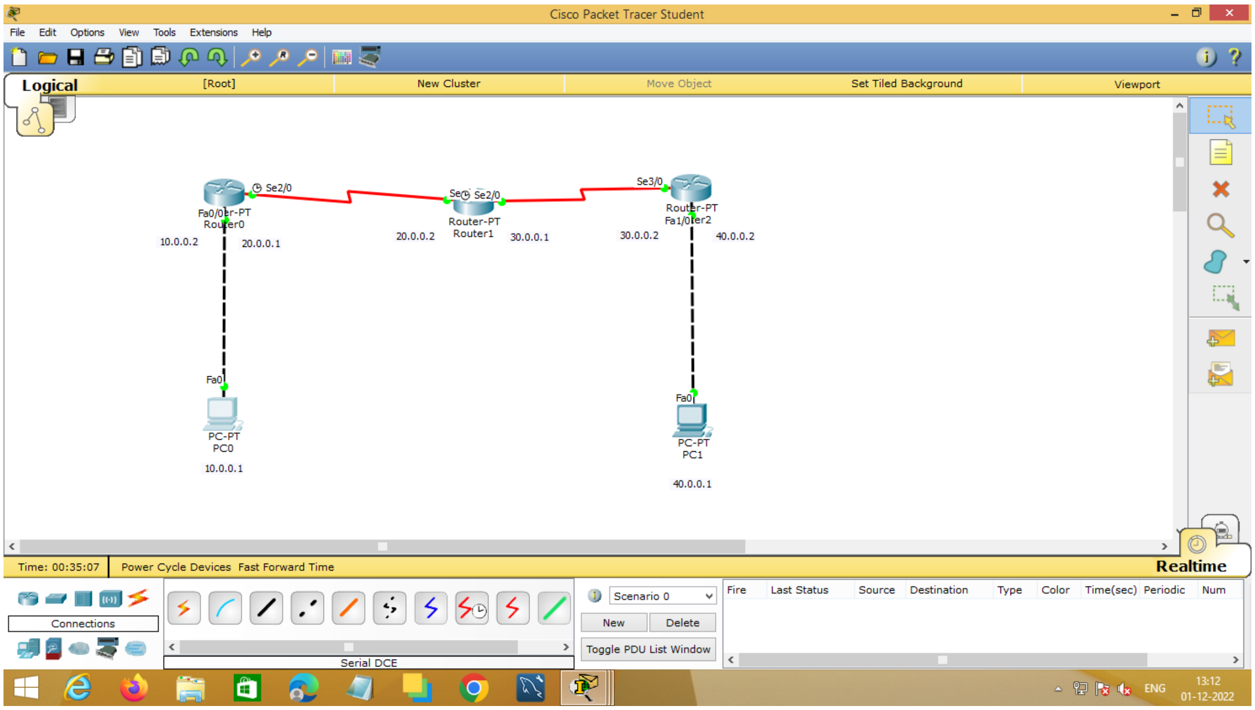
```
Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

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

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

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

C:\>
```

Experiment No 3

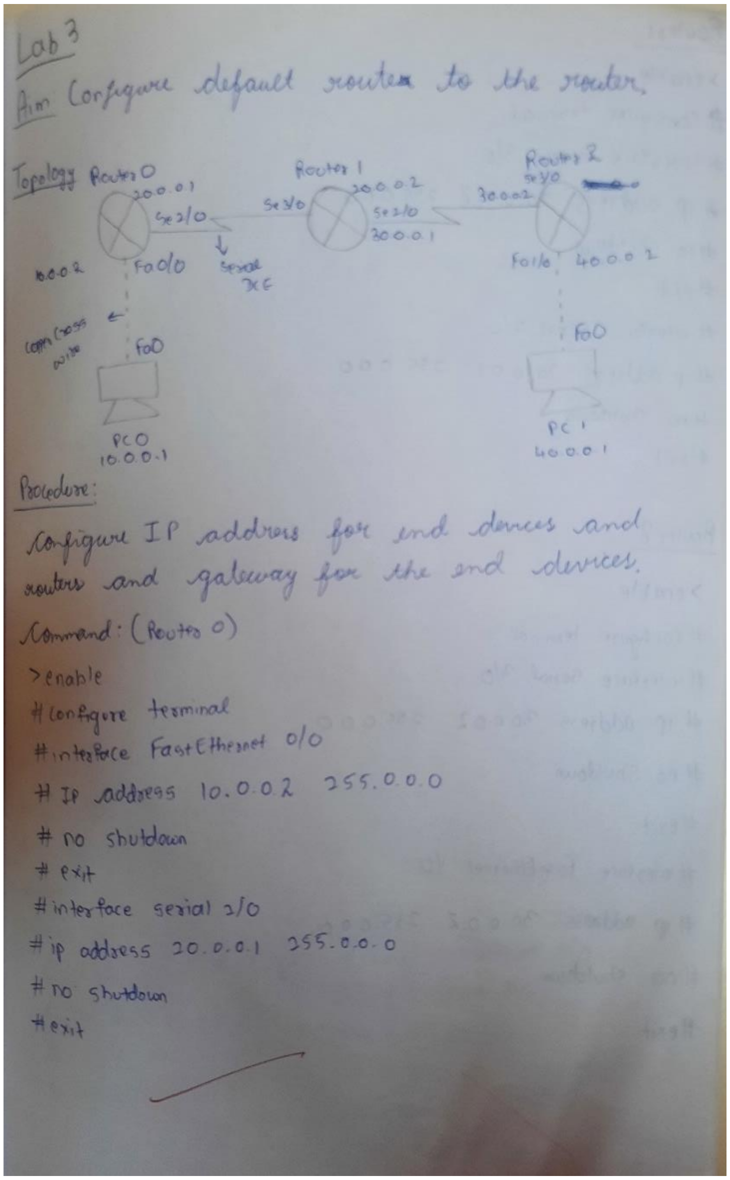
Aim

Configuring default route to the Router

Topology



Procedure



Lab 3

Aim Configure default routes to the router.

Topology Router 0
20.0.0.1
Se 2/0
Fa0/0
Serial DCE
10.0.0.2
copper cross wire
Fa0
PC0
10.0.0.1
Router 1
Se 3/0
20.0.0.2
Se 2/0
30.0.0.1
Router 2
Se 3/0
30.0.0.2
Fa1/0
40.0.0.2
Fa0
PC 1
40.0.0.1

Procedure:

Configure IP address for end devices and routers and gateway for the end devices.

Command: (Router 0)

```
>enable
#configure terminal
#interface FastEthernet 0/0
# IP address 10.0.0.2 255.0.0.0
# no shutdown
# exit
#interface serial 2/0
#ip address 20.0.0.1 255.0.0.0
# no shutdown
#exit
```

Router 1

> enable
configure terminal
interface Serial 3/0
ip address 20.0.0.2 255.0.0.0
no shutdown
exit
interface Serial 2/0
ip address 30.0.0.1 255.0.0.0
no shutdown
exit

Router 2

> enable
configure terminal
interface Serial 3/0
ip address 30.0.0.2 255.0.0.0
no shutdown
exit
interface FastEthernet 1/0
ip address 30.0.0.2 255.0.0.0
no shutdown
exit

Result: (PC0)

PC0 > Ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data

Reply from 10.0.0.2: Destination host unreachable
Reply from 10.0.0.2: Destination host unreachable
Reply from 10.0.0.2: Destination host unreachable
Reply from 10.0.0.2: Destination host unreachable
Reply from 10.0.0.2: Destination host unreachable

Ping Statistics for 40.0.0.1
Packets: Sent = 4, Received = 0, Lost = 4 (100 / Loss)

Command: (Router 0)

> show ip route
Gateway of last router is not set
C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, Serial 2/0

> enable
config t
ip route 30.0.0.0 255.0.0 20.0.0.2
ip route 40.0.0.0 255.0.0.0 20.0.0.2
exit
> show ip route
C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, Serial 2/0
S 30.0.0.0/8 [1/0] via 20.0.0.2
S 40.0.0.0/8 [1/0] via 20.0.0.2

Router 1

> Show ip route
C 20.0.0.0/8 is directly connected, Serial 3/0
C 30.0.0.0/8 is directly connected, Serial 2/0

> enable
config terminal
ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2
exit
> Show ip route
C 20.0.0.0/8 is directly connected, Serial 3/0
C 30.0.0.0/8 is directly connected, Serial 2/0
S 10.0.0.0/8 [1/0] via 20.0.0.1
S 40.0.0.0/8 [1/0] via 30.0.0.2

Router 2

> enable
config t
ip route 10.0.0.0 255.0.0.0 30.0.0.1
ip route 20.0.0.0 255.0.0.0 30.0.0.1
exit
> Show ip route
S 10.0.0.0/8 [1/0] via 30.0.0.1
S 20.0.0.0/8 [1/0] via 30.0.0.1
C 30.0.0.0/8 is directly connected, Serial 3/0
C 40.0.0.0/8 is directly connected, FastEthernet 1/0

Observations

Since the static route was not set, destination host was unreachable before.
Now, we set the static route now,

> 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=22ms TTL=255
Reply from 40.0.0.1; bytes=32 time=22ms TTL=255
Reply from 40.0.0.1; bytes=32 time=22ms TTL=255

Pinging Statistics
Packets: Sent = 4, Received = 3, Lost = 1 (25 / Loss)

10

R
1/12/2

Output

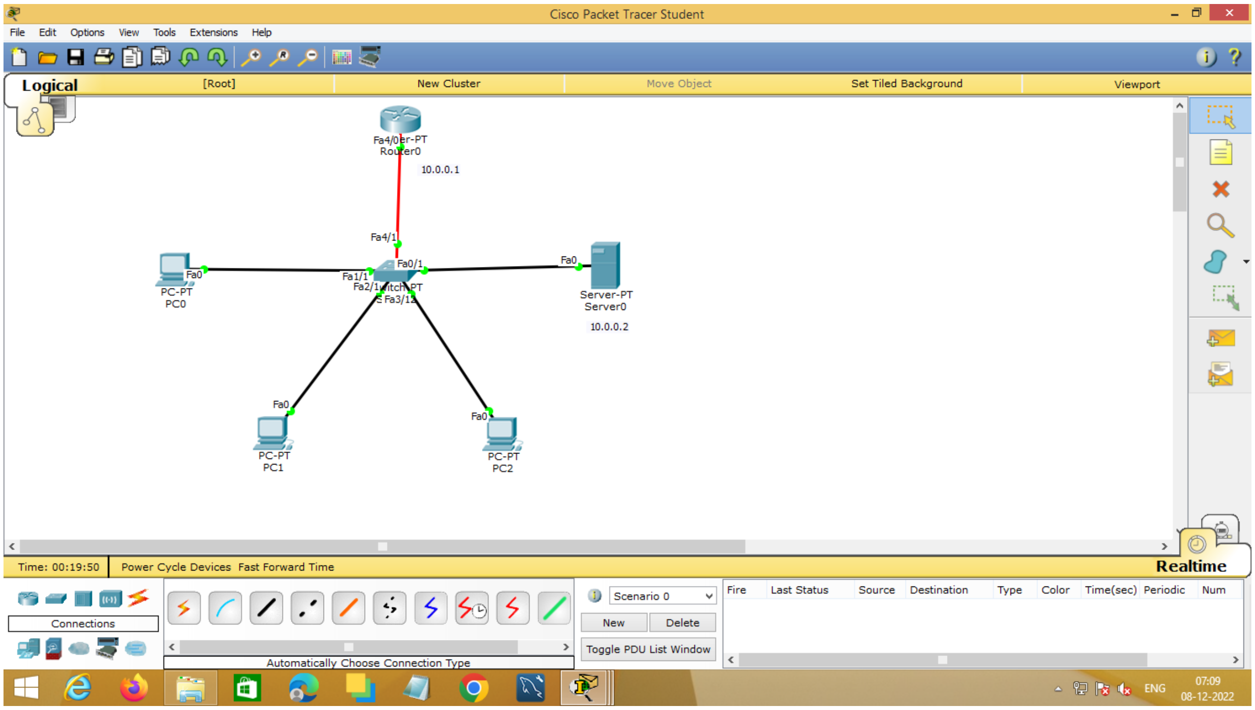
```
Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.1

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

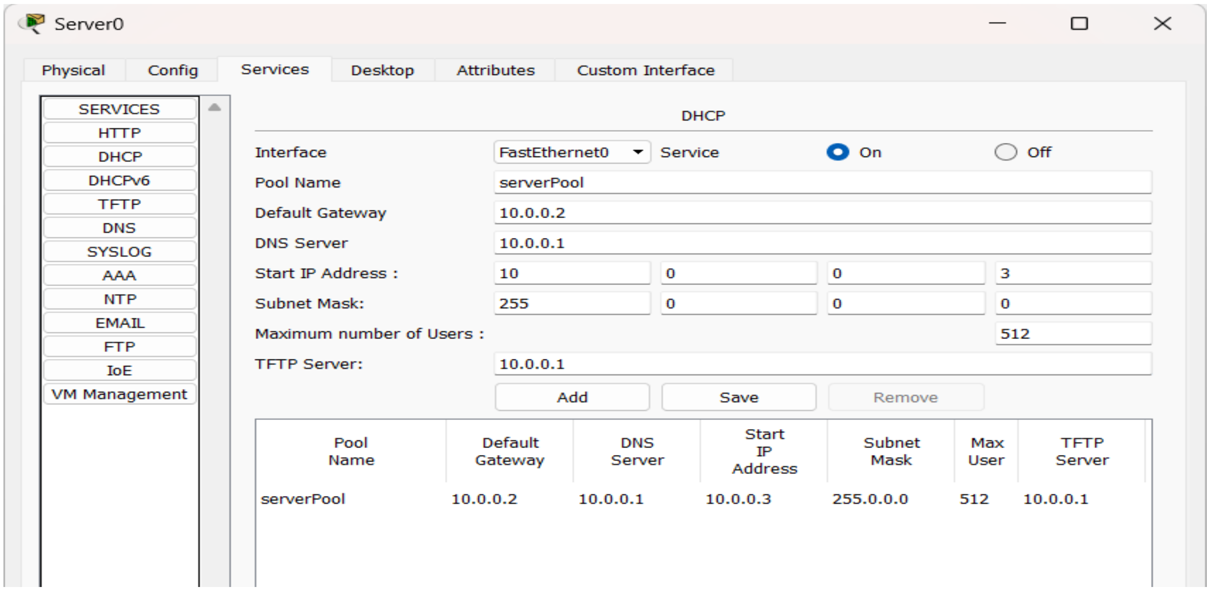
Aim

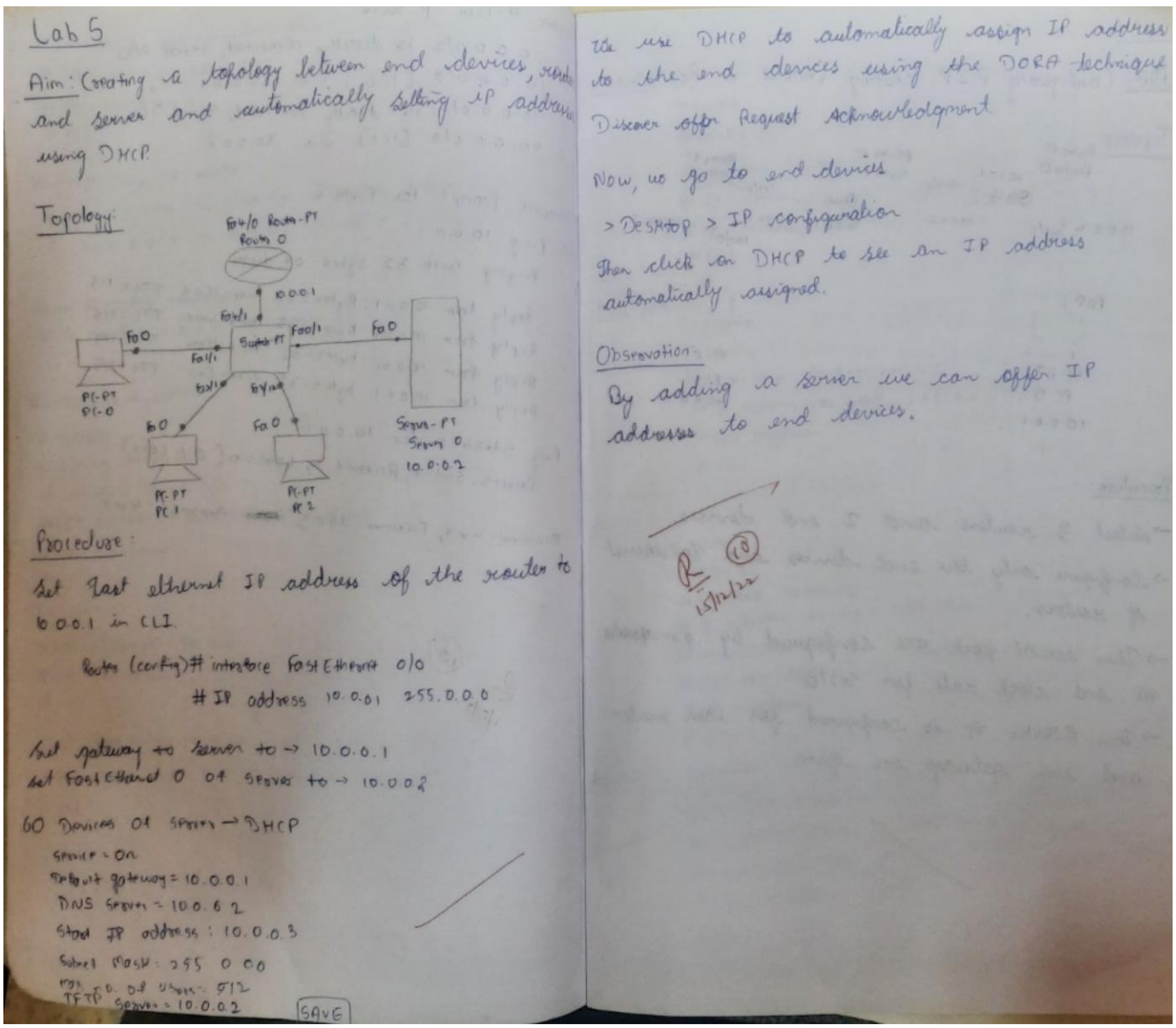
Configuring DHCP within a LAN in a packet Tracer

Topology



Procedure





Lab 5

Aim: Creating a topology between end devices, router and server and automatically setting IP address using DHCP

Topology:

Procedure:

Set fast ethernet IP address of the router to 10.0.0.1 in CLI.

Router (config)# interface FastEthernet 0/0
IP address 10.0.0.1 255.0.0.0

Set gateway to server to → 10.0.0.1
Set FastEthernet 0 of server to → 10.0.0.2

Go Devices Of Server → DHCP
Service = On
Default gateway = 10.0.0.1
DNS Server = 10.0.0.2
Start IP address : 10.0.0.3
Subnet Mask : 255.0.0.0
Max. no. of Users : 512
TFTP Server = 10.0.0.2
SAVE

to use DHCP to automatically assign IP address to the end devices using the DORA technique
Discover offer Request Acknowledgment

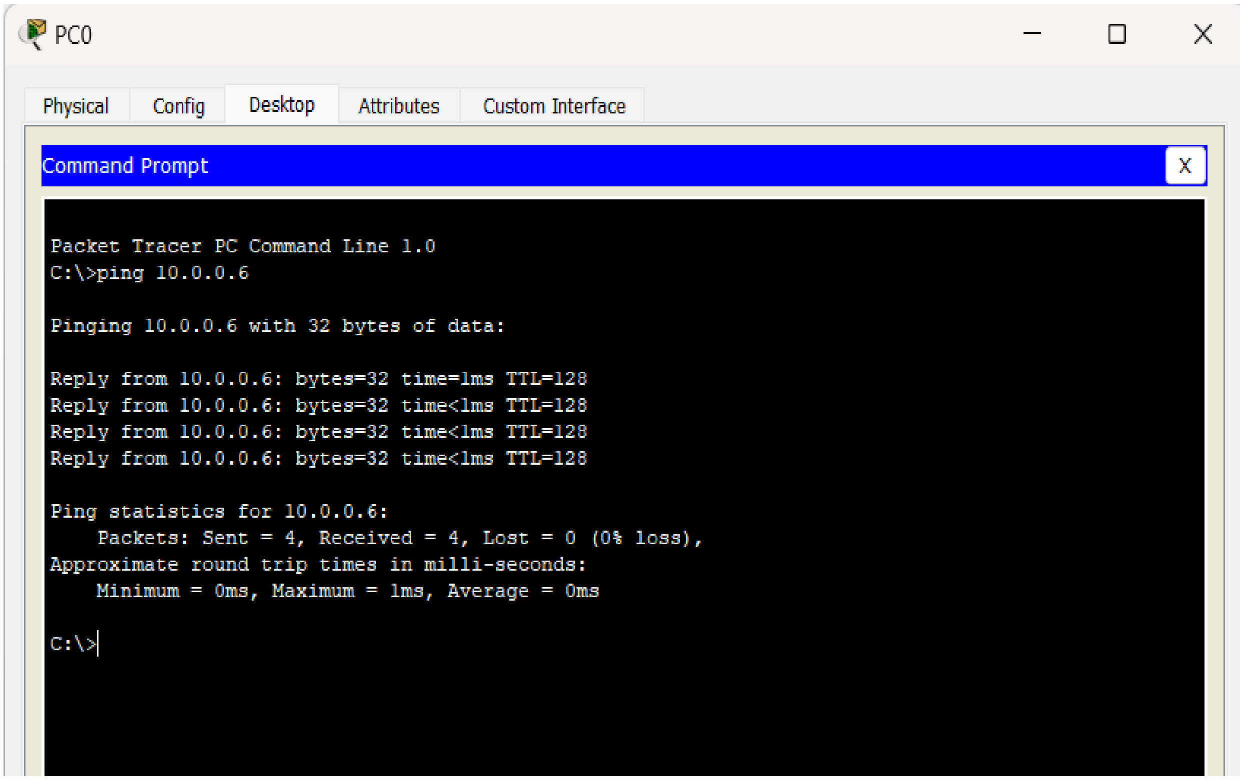
Now, we go to end devices
> Desktop > IP configuration
Then click on DHCP to see an IP address automatically assigned.

Observation

By adding a server we can offer IP addresses to end devices.

10
15/12/22

Output

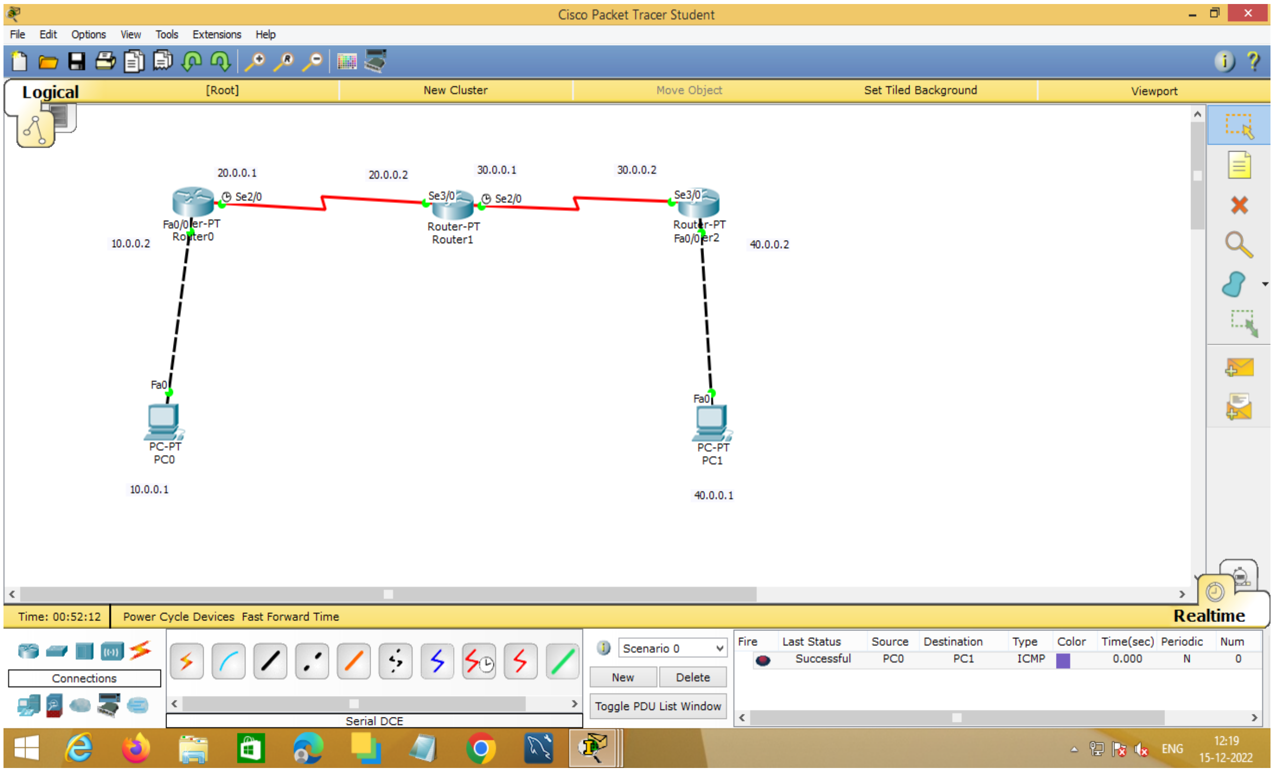

PC0
Physical
Config
Desktop
Attributes
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.6
Pinging 10.0.0.6 with 32 bytes of data:
Reply from 10.0.0.6: bytes=32 time=1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
Reply from 10.0.0.6: bytes=32 time<1ms TTL=128
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

Configuring RIP Routing Protocol in Routers

Topology

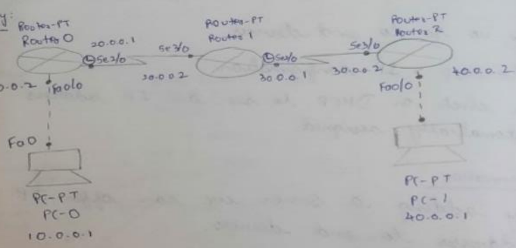


Procedure

Lab 6

Aim: Configuring RIP Routing Protocol in Routers.

Topology:



Procedure:

→ Select 3 routers and 2 end devices.

→ Configure only the end devices and fast ethernet of routers.

→ Then serial port are configured by encapsulation PPP and clock rate for Se2/0.

→ Then #router rip is configured for each router and then gateways are given.

Commands

Router 0

```
>enable
#config t
#interface Fastethernet 0/0
#ip address 10.0.0.2 255.0.0.0
#no shutdown
#exit
```

Router 2

```
>enable
#config t
#interface Fastethernet 0/0
#ip address 40.0.0.2 255.0.0.0
#no shutdown
#exit.
```

Router 0

```
>enable
#config t
#interface serial 2/0
#ip address 20.0.0.1 255.0.0.0
#encapsulation PPP
#clock rate 64000
#no shutdown
#exit
```

Router 1

```
>enable
#config t
#interface serial 3/0
#ip address 20.0.0.2 255.0.0.0
#encapsulation PPP
#no shutdown
#exit
#interface serial 2/0
#ip address 30.0.0.1 255.0.0.0
#encapsulation PPP
#clock rate 64000
#no shutdown
#exit
```

Router 2

```
>enable
#config t
#interface serial 3/0
#ip address 30.0.0.2 255.0.0.0
#encapsulation PPP
#no shutdown
#exit
```

Router 0

```
>enable
#config t
#router rip
#network 10.0.0.0
#network 20.0.0.0
#exit
```

Router 1

```
>enable
#config t
#router rip
#network 20.0.0.0
#network 30.0.0.0
#exit
```

Router 2

```
>enable
#config t
#router rip
#network 30.0.0.0
#network 40.0.0.0
#exit.
```

Observations:

PC0

```
>Ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125
                              time=2ms
                              time=2ms
                              time=2ms

Ping statistics for 40.0.0.1:
    Packets: Sent=4, Received=4, Lost=0 (0% Loss)
Approximate round trip in milli-seconds
    Minimum=2ms, Maximum=10ms, Average=4ms
```

Output:

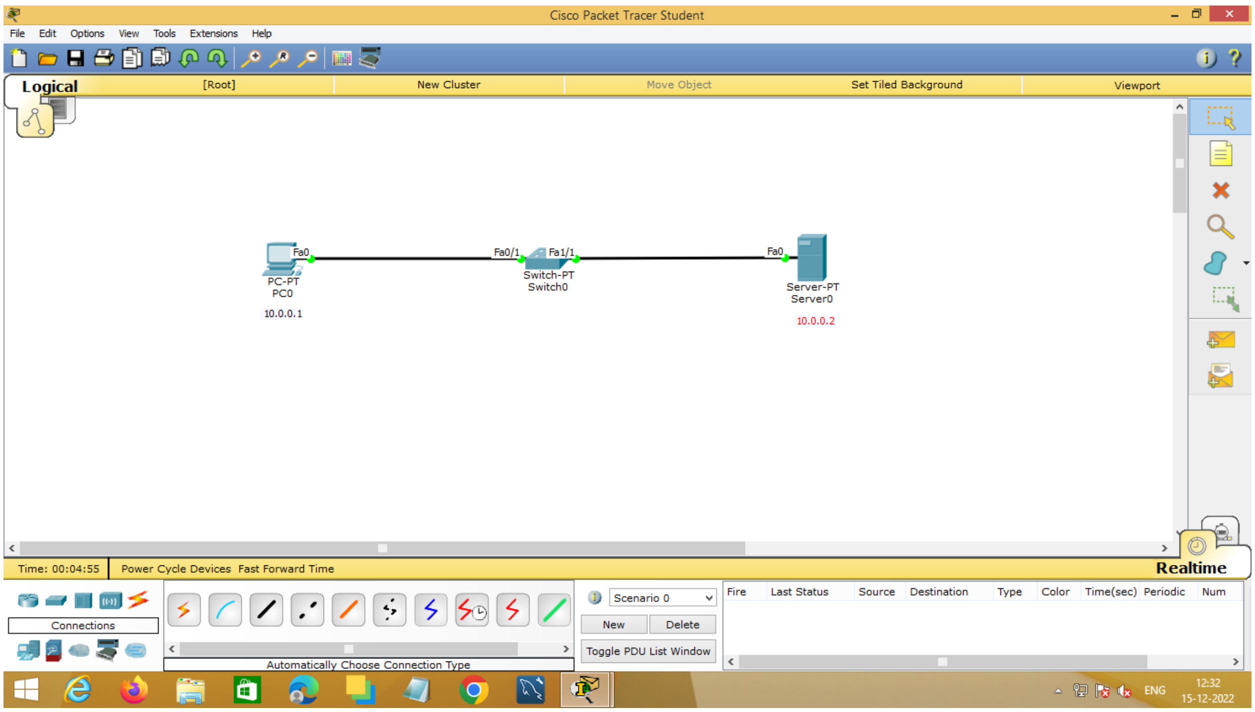
```
C:\>ping 40.0.0.1

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 6

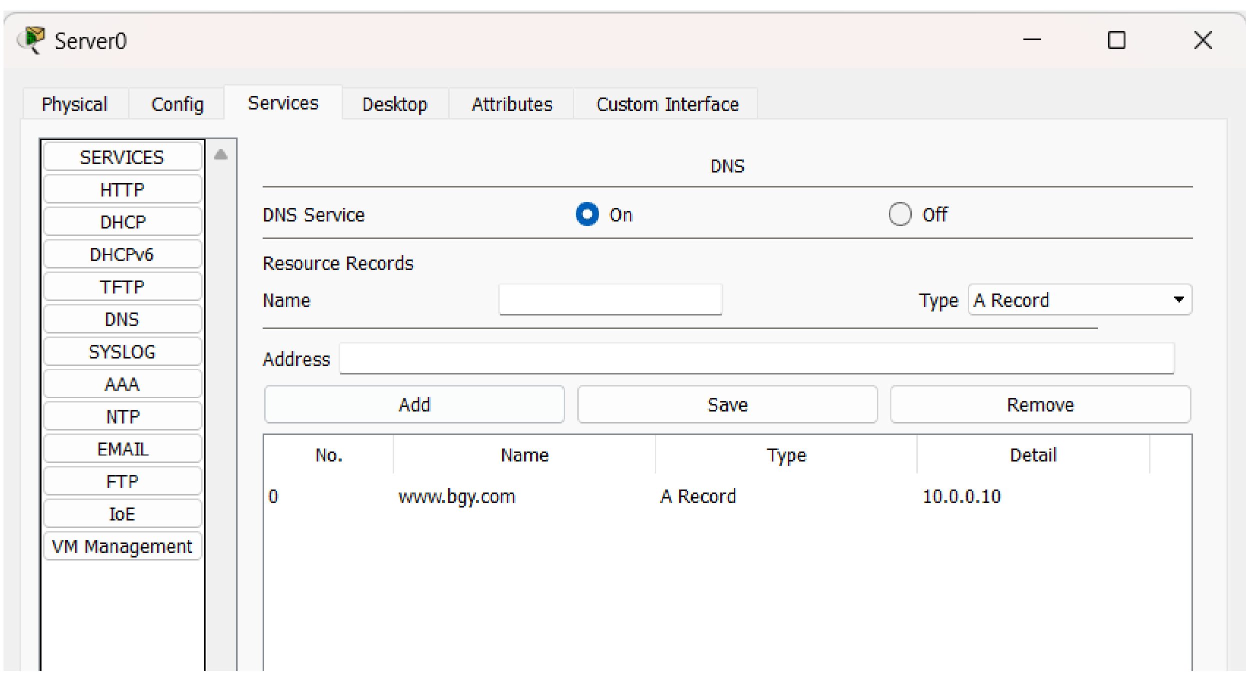
Aim

Demonstration of WEB server and DNS using Packet Tracer

Topology



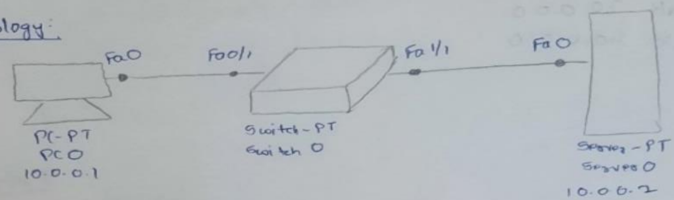
Procedure



Lab 7

Aim: Demonstration of WEB Server and DNS using Packet Tracer

Topology:



Procedure:

→ Connect one end device and one server to a switch

→ Configure the IP address of end device (10.0.0.1) and server (10.0.0.2)

→ Go to services in server and enable HTTP and DNS.

→ Add a domain name (eg. www.abc.com), address name - (Server ip) (10.0.0.2)

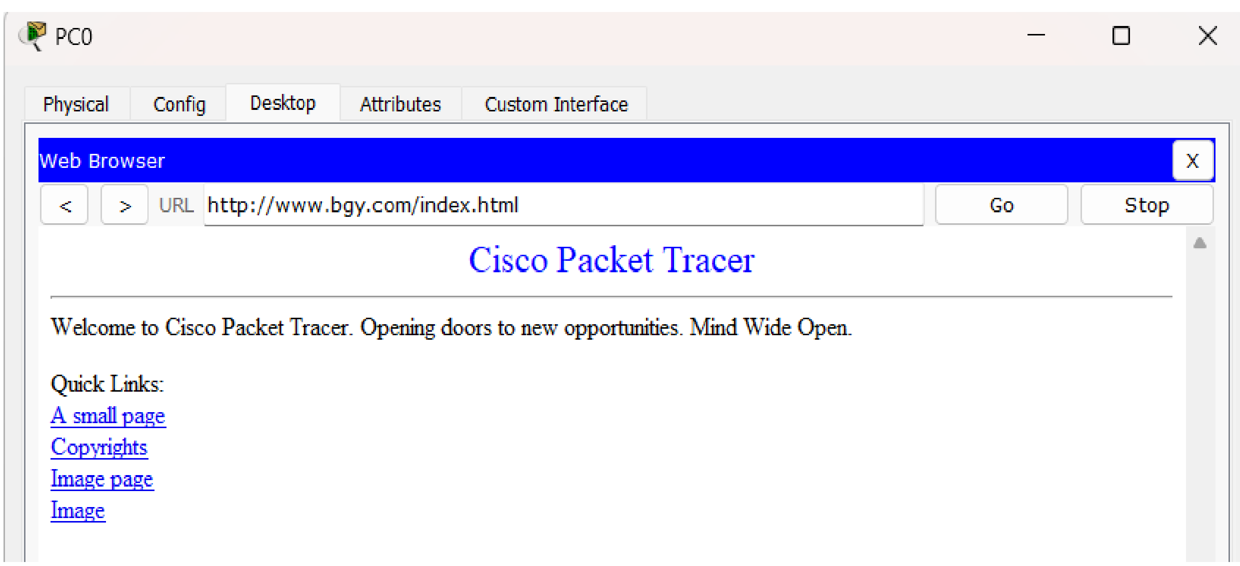
Add and save.

→ Go to web browser in end device and search for added domain name.

→ Try to search for domain name/.....html

Result: The added domain name is accessible

Output



Cycle-2

Experiment No 1

Aim

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;
}
```

Observation:

Lab 8: CRC checksum 16-bit program implementation.

divisor(16-bit): 10001000000100001

std polynomial g(x) divisor: $x^{16} + x^{12} + x^5 + 1$ CRC-16-bit

Code:

```
#include<bits/stdc++.h>
#include<string.h>
using namespace std;
int crc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op,ip);
    if(mode)
    {   for(int i=1; i<strlen(poly);i++)
            strcat(op,'0');
    }
    for(int i=0; i<strlen(ip);i++)
    {   if(op[i]=='1')
        {   for(int j=0; j<strlen(poly);j++)
            {   if(op[i+j]==poly[j])
                    op[i+j]='0';
                else
                    op[i+j]='1';
            }
        }
    }
    for(int i=0; i<strlen(op);i++)
    {       if(op[i]=='1')
                return 0;
    }   ~~return~~
        return 1
}
int main()
{
    char ip[50],op[50],recv[50];
    char poly[]="10001000000100001";
    cout<<"Enter ip msg in binary"<<endl;
    cin>>ip;
    crc(ip,op,poly,1);
    cout<<"Transmitted msg:"<<ip<<op+strlen(ip)<<endl;
    cout<<"Enter received msg in binary"<<endl;
    cin>>recv;
    if(crc(recv,op,poly,0))
        cout<<"no error in data"<<endl;
    else
        cout<<"Error occurred"<<endl;
    return 0;
}
```

Output:

Enter msg in binary: 1010101010000000

The transmitted msg is. 10101010100000000110001100011110l

Enter received msg in binary

1010101010000000

No error in data

Output

```
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

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

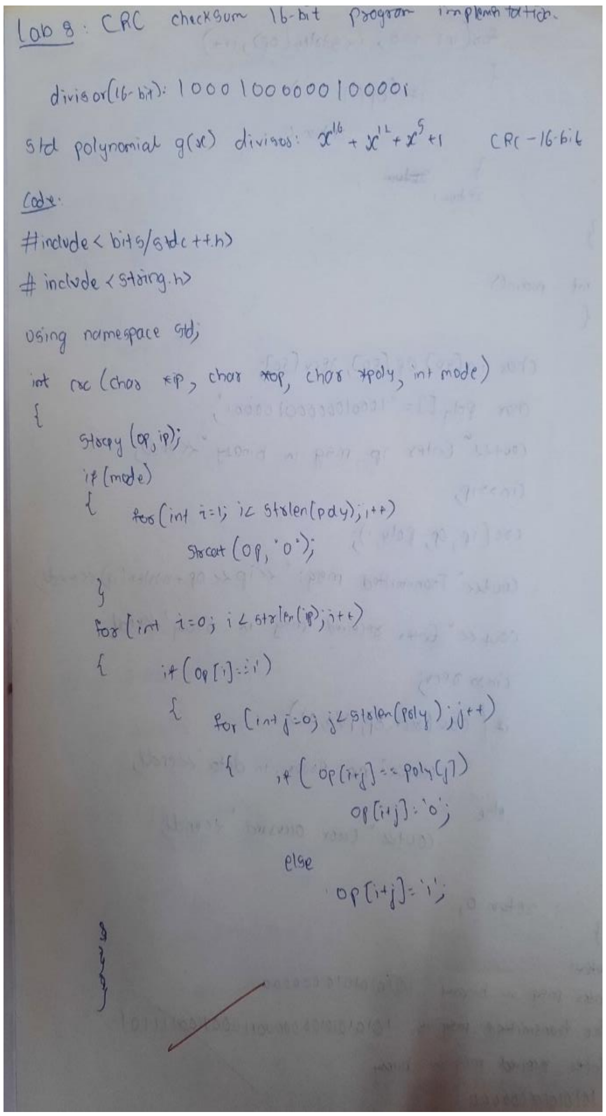
Code

```
#include<stdio.h>


#define INF  99999
#define n 5


void printSolution(int g[n])
{
    printf("Hop count         : ");
    for(int j=0;j<n;j++)
    {
        if(g[j] == INF)
            printf("INF\t");
        else
            printf("%d\t",g[j]);
    }
    printf("\n");
}


void findShortestPath(int dist[][n])
{


    for(int k=0;k<n;k++)
    {
        for(int i=0;i<n;i++)
```

```
        {
          for(int j=0;j<n;j++)
          {
            if(dist[i][j] > dist[i][k] + dist[k][j]
            &&(dist[i][k] != INF && dist[k][j] != INF))
            {
              dist[i][j] = dist[i][k] + dist[k][j];
            }
          }
        }
      }

      char c = 'A';
      for(int i=0; i<n; i++ )
      {
        printf("Router table entries for router %c:\n", c);
        printf("Destination router: A\tB\tC\tD\tE\n");
        printSolution(dist[i]);
        c++;
      }
    }

    int main()
    {
      int graph[][n] = { {0,   1,  1, INF, INF},
                {1,  0,  INF,  INF, INF},
                {1, INF, 0, 1,  1},
                {INF, INF, 1, 0, INF},
```

```
                {INF, INF, 1, INF, 0}};

    findShortestPath(graph);
    return 0;
}
```

Observation:

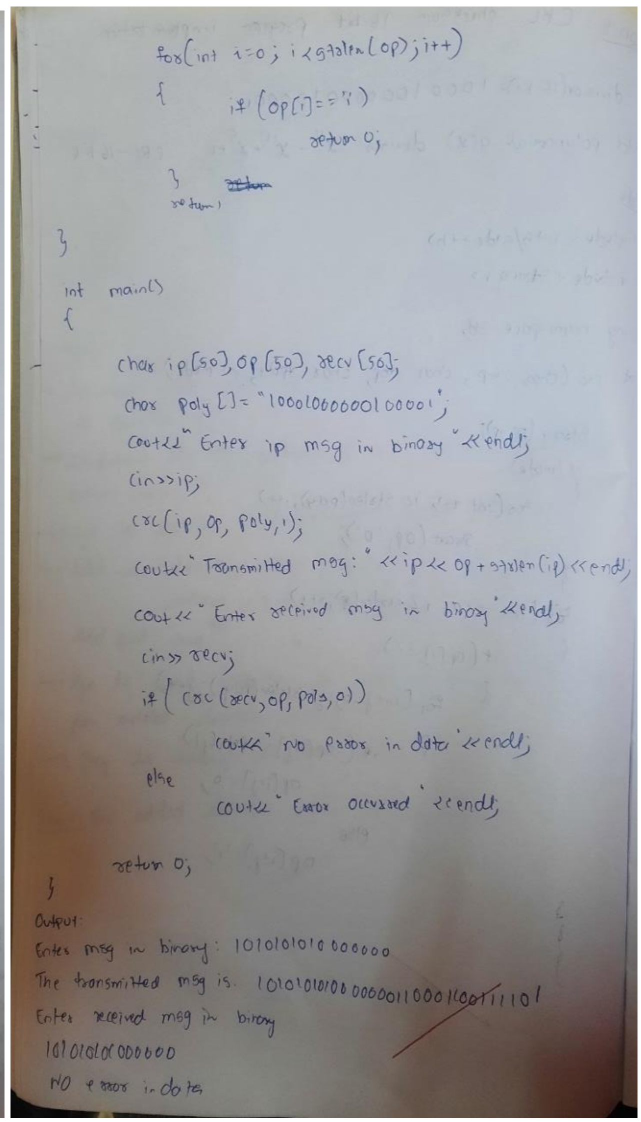

Lab 8: CRC checksum 16-bit program implementation.

divisor(16-bit): 10001000000100001

std polynomial g(x) divisor: $x^{16}+x^{12}+x^{5}+1$ CRC-16-bit

Code:

```
#include<bits/stdc++.h>
#include<string.h>
using namespace std;
int crc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op, ip);
    if(mode)
    {   for(int i=1; i<strlen(poly); i++)
            strcat(op, "0");
    }
    for(int i=0; i<strlen(ip); i++)
    {   if(op[i]=='1')
        {   for(int j=0; j<strlen(poly); j++)
            {   if(op[i+j]==poly[j])
                    op[i+j]='0';
                else
                    op[i+j]='1';
            }
        }
    }
```



```
    for(int i=0; i<strlen(op); i++)
    {   if(op[i]=='1')
            return 0;
    }
    return 1;
}
int main()
{
    char ip[50], op[50], recv[50];
    char poly[]="10001000000100001";
    cout<<"Enter ip msg in binary"<<endl;
    cin>>ip;
    crc(ip, op, poly, 1);
    cout<<"Transmitted msg:"<<ip<<op+strlen(ip)<<endl;
    cout<<"Enter received msg in binary"<<endl;
    cin>>recv;
    if(crc(recv, op, poly, 0))
        cout<<"no error in data"<<endl;
    else
        cout<<"Error occured"<<endl;
    return 0;
}
```

Output:

Enter msg in binary: 1010101010000000

The transmitted msg is. 10101010100000000110001100011101

Enter received msg in binary

1010101010000000

NO error in data

Output:

```
Router table entries for router A:
Destination router: A    B       C       D       E
Hop count         : 0    1       1       2       2
Router table entries for router B:
Destination router: A    B       C       D       E
Hop count         : 1    0       2       3       3
Router table entries for router C:
Destination router: A    B       C       D       E
Hop count         : 1    2       0       1       1
Router table entries for router D:
Destination router: A    B       C       D       E
Hop count         : 2    3       1       0       2
Router table entries for router E:
Destination router: A    B       C       D       E
Hop count         : 2    3       1       2       0


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

Experiment No 3

Aim

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

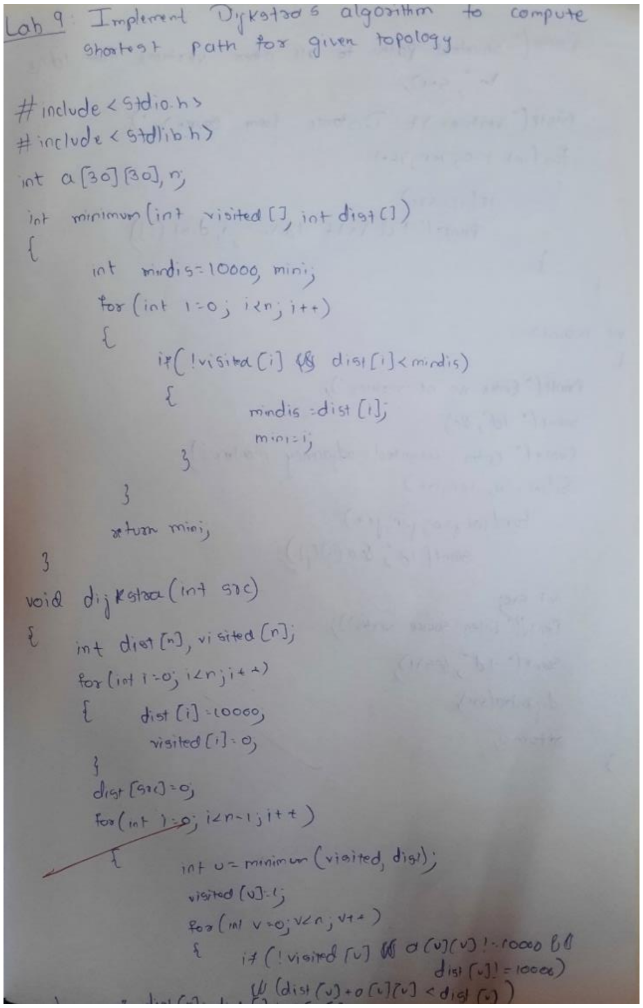
Code

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


void dijkstra(int graph[10][10],int V)
{
    int distance[V], predefine[V], visited[V];
    int startnode, count, min_distance, nextnode, i, j;
    printf("\nEnter the start node: ");
    scanf("%d", &startnode);
    for(i=0; i<V; i++) {
        distance[i] = graph[startnode][i];
        predefine[i] = startnode;
        visited[i] = 0;
    }
    distance[startnode] = 0;
    visited[startnode] = 1;
    count = 1;
    while(count<V-1) {
        min_distance = 99;
        for(i=0; i<V; i++) {
            if(distance[i] < min_distance && visited[i]==0)
            {
                min_distance = distance[i];
```

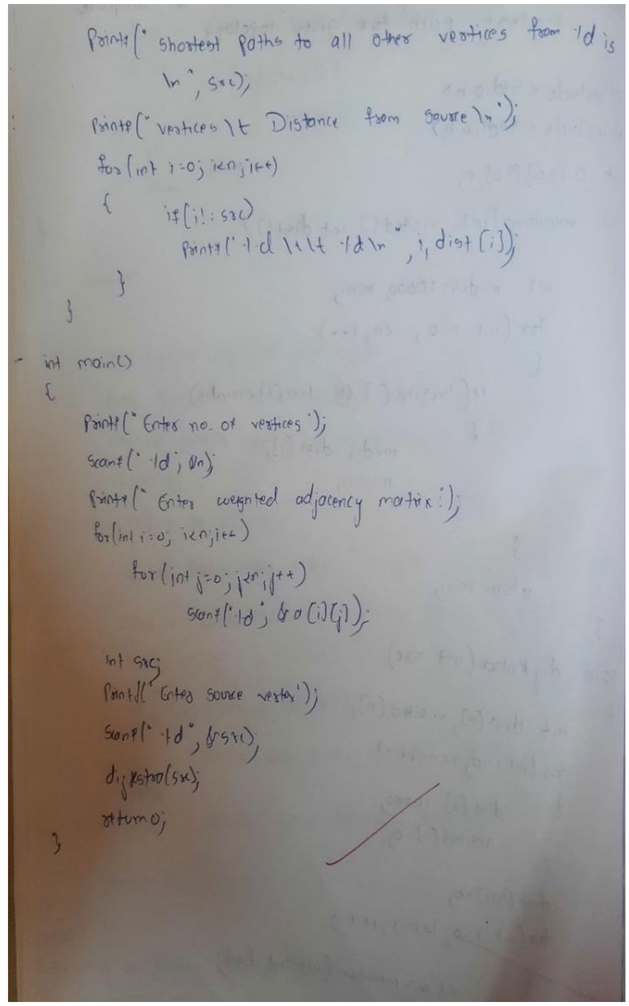
```
                nextnode = i;
            }
        }
        visited[nextnode] = 1;
        for(i=0;i<V;i++)
        {
            if(visited[i] == 0)
            {
                if((min_distance + graph[nextnode][i]) < distance[i])
                {
                    distance[i] = min_distance + graph[nextnode][i];
                    predefine[i] = nextnode;
                }
            }
        }
        count = count + 1;
    }
    for(i=0;i<V;i++) {
        if(i!=startnode) {
            printf("\nDistance of node %d = %d", i, distance[i]);
            printf("\nPath = %d",i);
            j = i;
            do
            {
                j = predefine[j];
                printf(" <- %d",j);
            } while (j != startnode);
        }
```

```
    }
}
int main()
{
    int i, j;
    int V;
    printf("Enter the number of vertices: ");
    scanf("%d", &V);
    int graph[V][V];
    printf("\nEnter the cost/weight matrix: \n");
    for(i=0; i<V; i++) {
        for(j=0;j<V;j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    dijkstra(graph, V);
    return 0;
}
```

Observation:



```
Lab 9: Implement Dijkstra's algorithm to compute
       shortest path for given topology

#include<stdio.h>
#include<stdlib.h>
int a[30][30],n;
int minimum(int visited[], int dist[])
{
    int mindis=10000, mini;
    for(int i=0; i<n; i++)
    {
        if(!visited[i] && dist[i]<mindis)
        {
            mindis=dist[i];
            mini=i;
        }
    }
    return mini;
}
void dijkstra(int src)
{   int dist[n], visited[n];
    for(int i=0; i<n; i++)
    {   dist[i]=10000;
        visited[i]=0;
    }
    dist[src]=0;
    for(int i=0; i<n-1; i++)
    {   int u=minimum(visited, dist);
        visited[u]=1;
        for(int v=0; v<n; v++)
        {   if(!visited[v] && a[u][v]!=10000 &&
                  dist[u]!=10000
               && (dist[u]+a[u][v] < dist[v]))
```



```
    printf("shortest paths to all other vertices from %d is
          \n", src);
    printf("vertices \t Distance from source \n");
    for(int i=0; i<n; i++)
    {   if(i!=src)
            printf("%d \t\t %d \n", i, dist[i]);
    }
}
int main()
{
    printf("Enter no. of vertices");
    scanf("%d", &n);
    printf("Enter weighted adjacency matrix:");
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            scanf("%d", &a[i][j]);
    int src;
    printf("Enter source vertex");
    scanf("%d", &src);
    dijkstra(src);
    return 0;
}
```

Output:

```
Enter the number of vertices: 5

Enter the cost/weight matrix:
0 10 99 5 7
10 0 1 2 99
99 1 0 9 4
5 2 9 0 99
7 99 4 99 0

Enter the start node: 0

Distance of node 1 = 5
Path = 1 <- 4 <- 3 <- 0
Distance of node 2 = 5
Path = 2 <- 4 <- 3 <- 0
Distance of node 3 = 5
Path = 3 <- 0
Distance of node 4 = 5
Path = 4 <- 3 <- 0

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

Experiment No 4

Aim

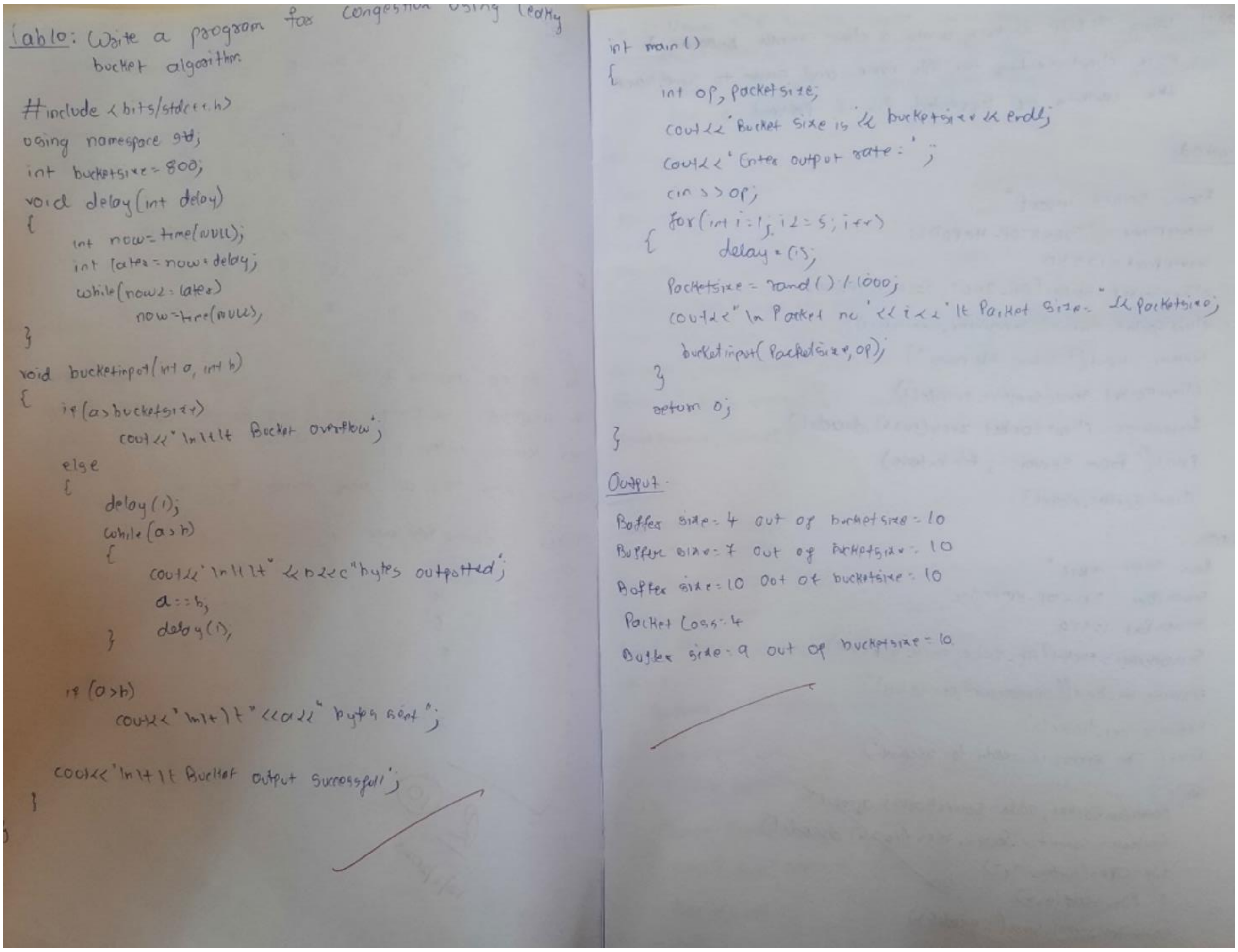
Write a program for congestion control using Leaky bucket algorithm

Code

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
        int no_of_queries, storage, output_pkt_size;
        int input_pkt_size, bucket_size, size_left;
        storage = 0;
        no_of_queries = 4;
        bucket_size = 10;
        input_pkt_size = 4;
        output_pkt_size = 1;
        for (int i = 0; i < no_of_queries; i++) //
        {
                size_left = bucket_size - storage;
                if (input_pkt_size <= size_left) {
                        // update storage
                        storage += input_pkt_size;
                }
                else {
                        printf("Packet loss = %d\n", input_pkt_size);
                }
                printf("Buffer size= %d out of bucket size= %d\n",
                        storage, bucket_size);
```

```
            storage -= output_pkt_size;
      }
      return 0;}
```

Observation:


Lab10: Write a program for congestion using leaky bucket algorithm

```
#include <bits/stdc++.h>
using namespace std;
int bucketsize = 800;
void delay(int delay)
{
    int now=time(NULL);
    int later=now+delay;
    while(now<=later)
        now=time(NULL);
}
void bucketinput(int a, int b)
{
    if(a>bucketsize)
        cout<<"\n\t\t Bucket overflow";
    else
    {
        delay(1);
        while(a>b)
        {
            cout<<"\n\t\t"<<b<<"bytes outputted";
            a-=b;
            delay(1);
        }
    if(a>b)
        cout<<"\n\t\t"<<a<<" bytes sent";
    cout<<"\n\t\t Bucket output successful";
}
int main()
{
    int op, packetsize;
    cout<<"Bucket size is "<< bucketsize << endl;
    cout<<"Enter output rate : ";
    cin>>op;
    for(int i=1; i<=5; i++)
    {
        delay(1);
        packetsize = rand() % 1000;
        cout<<"\n Packet no "<<i<<"\t Packet size= "<<packetsize;
        bucketinput(packetsize, op);
    }
    return 0;
}
```

Output

Buffer size= 4 out of bucketsize= 10
Buffer size= 7 out of bucketsize= 10
Buffer size= 10 out of bucketsize= 10
Packet loss= 4
Buffer size= 9 out of bucketsize= 10

Output:

```
Buffer size= 4 out of bucket size= 10
Buffer size= 7 out of bucket size= 10
Buffer size= 10 out of bucket size= 10
Packet loss = 4
Buffer size= 9 out of bucket size= 10
```

Experiment No 5

Aim

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

Code

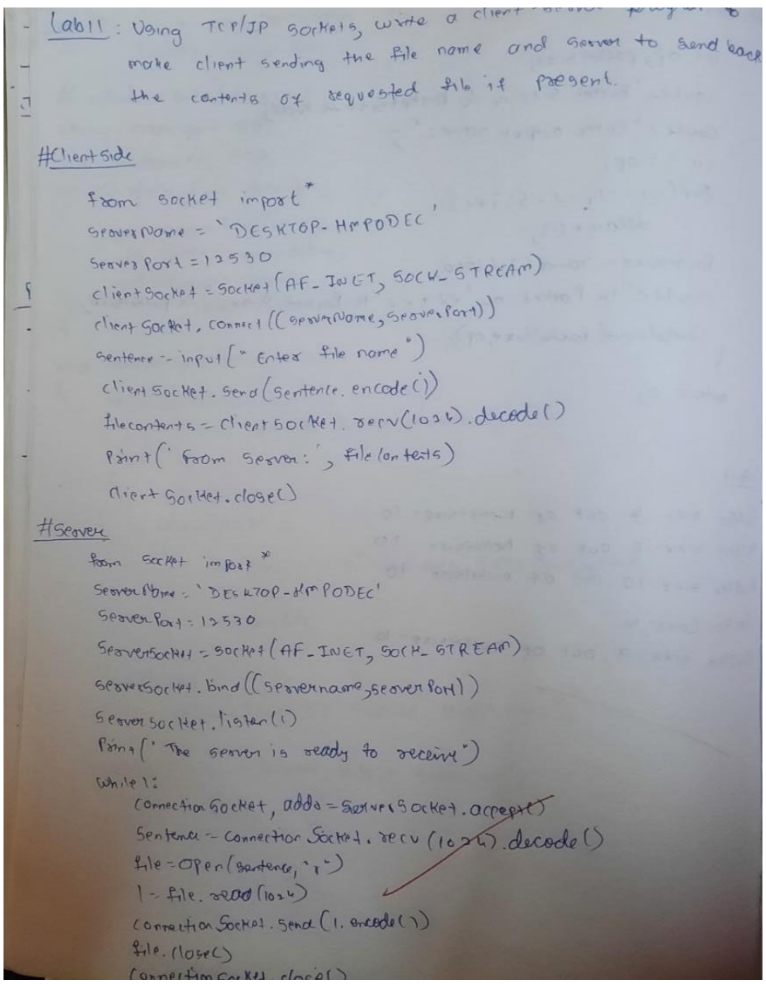
Server:

```
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()
```

Observation:



Lab11: Using TCP/IP sockets, write a client-server program to make client sending the file name and server to send back the contents of requested file if present.

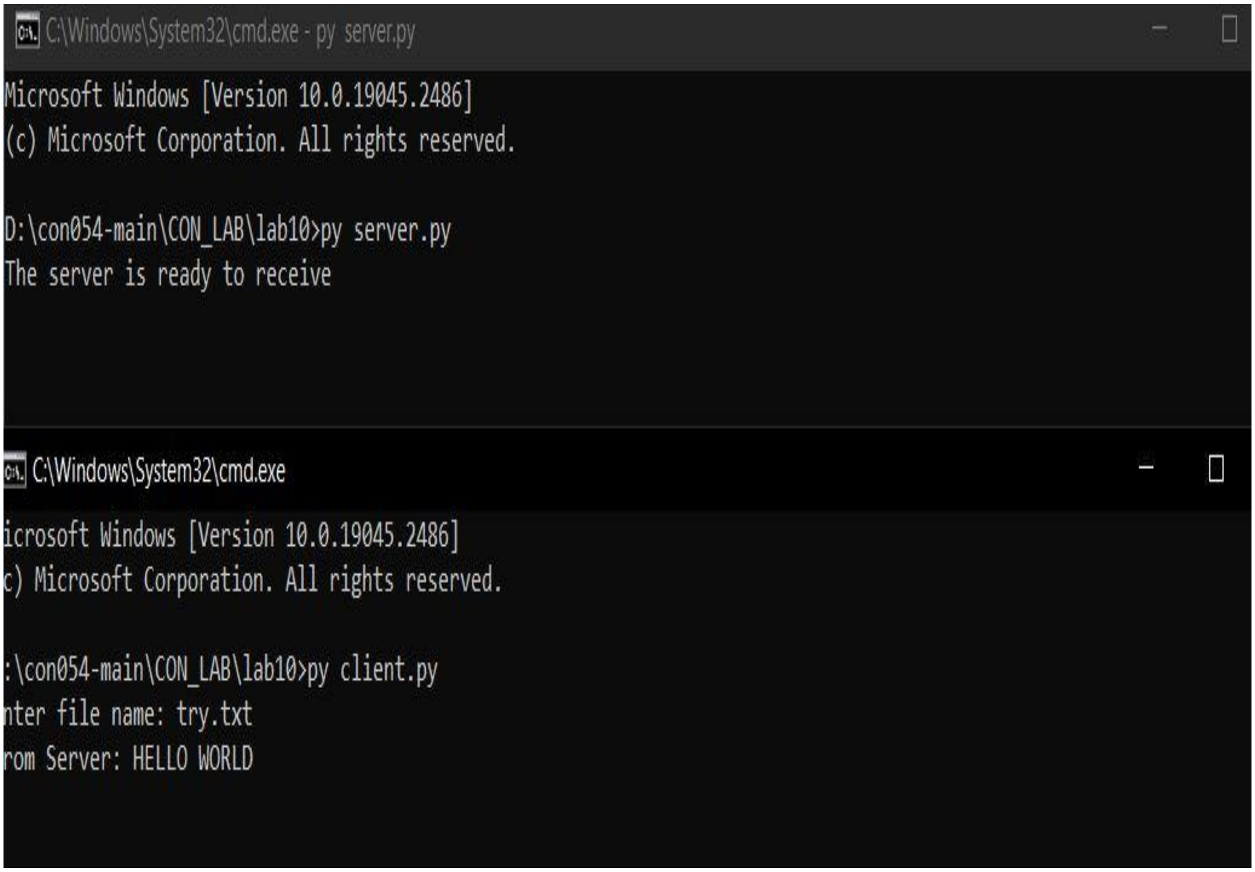
#Client Side

```
from socket import *
serverName = 'DESKTOP-HMPODEC'
Server Port = 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()
```

#Server

```
from socket import *
ServerName = 'DESKTOP-HMPODEC'
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()
    file = open(sentence, "r")
    l = file.read(1024)
    ConnectionSocket.send(l.encode())
    file.close()
    ConnectionSocket.close()
```

Output


C:\Windows\System32\cmd.exe - py server.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py server.py
The server is ready to receive
C:\Windows\System32\cmd.exe
icrosoft Windows [Version 10.0.19045.2486]
c) Microsoft Corporation. All rights reserved.
:\con054-main\CON_LAB\lab10>py client.py
nter file name: try.txt
rom Server: HELLO WORLD

Experiment No 6

Aim

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

Code

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:

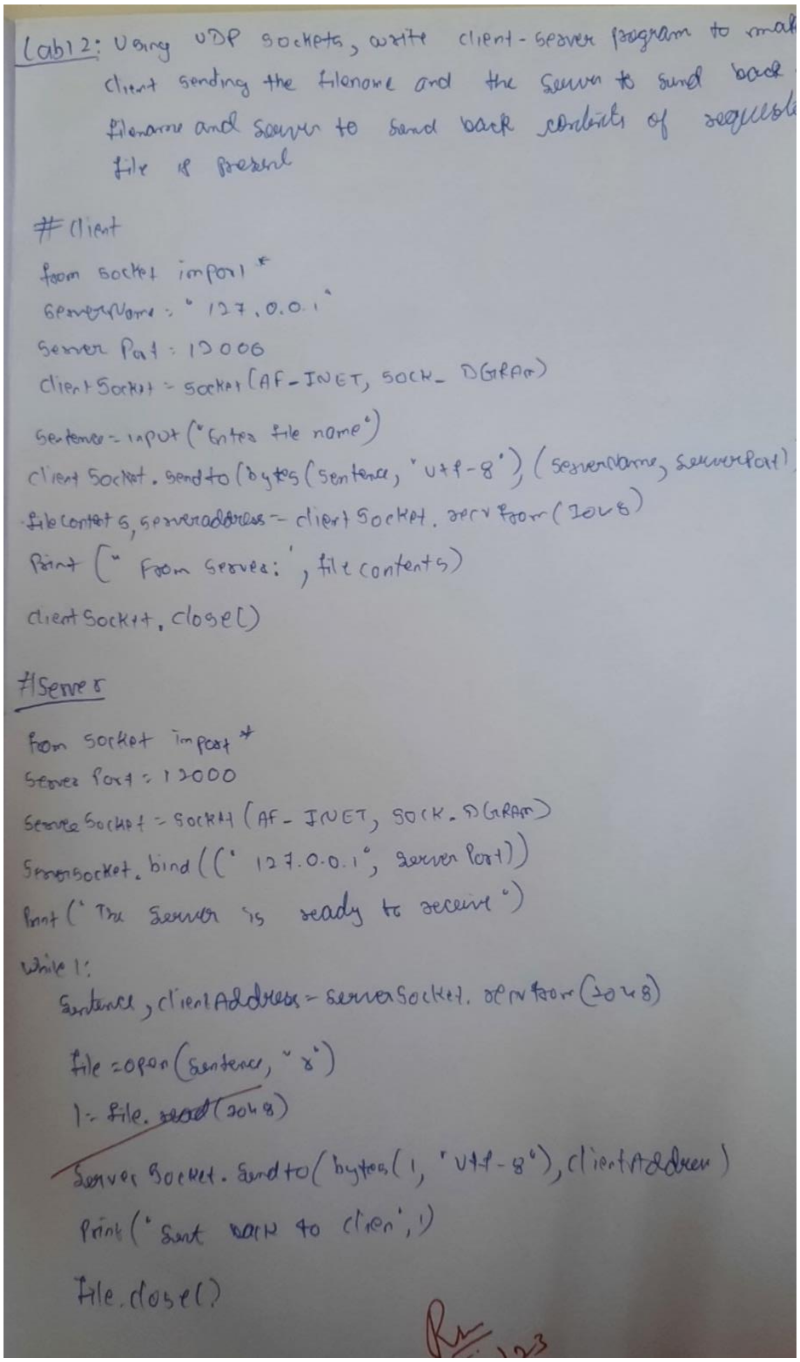
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
```

filecontents,serverAddress = clientSocket.recvfrom(2048)

print ('From Server:', filecontents)

clientSocket.close()

Observation:



Lab12: Using UDP sockets, write client-server program to make client sending the filename and the server to send back filename and server to send back contents of requested file if present

#client

from socket import *
serverName = "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 ("From Server:", filecontents)
clientSocket.close()

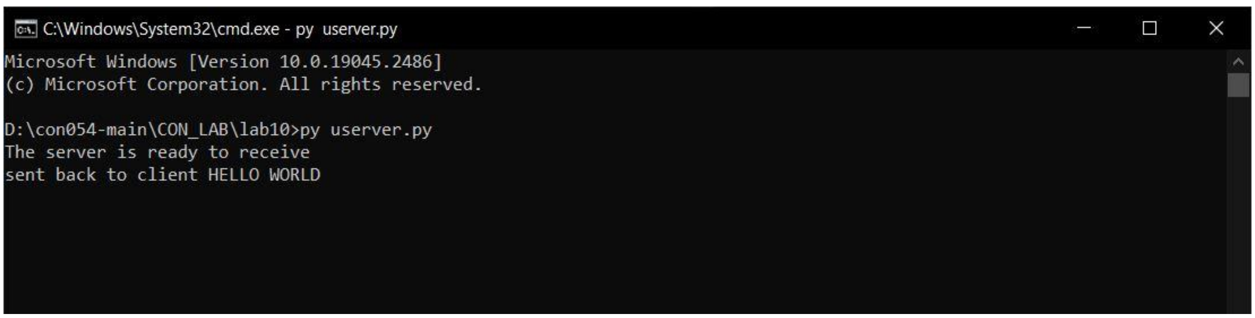
#Server

from socket import *
Server Port = 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()

Output


Select C:\Windows\System32\cmd.exe - py userver.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py userver.py
The server is ready to receive


C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py uclient.py
Enter file name: try.txt
From Server: b'HELLO WORLD\n\n'
D:\con054-main\CON_LAB\lab10>


C:\Windows\System32\cmd.exe - py userver.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.
D:\con054-main\CON_LAB\lab10>py userver.py
The server is ready to receive
sent back to client HELLO WORLD