

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

COMPUTER NETWORKS

Submitted by

GAGAN D (1BM20CS049)

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 **GAGAN D (1BM20CS049)**, 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.

Rekha G S
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

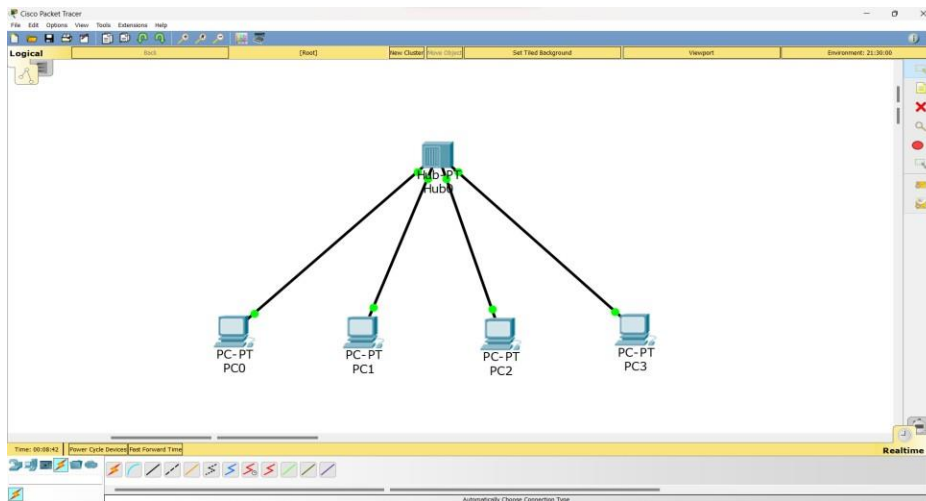
Cycle-1

Experiment No 1

Aim 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

Lab-1

Aim: Configuring building a topology & simulate data sending a simple PDU from source to destination using hub & switch as connecting devices.

Diagram: (Topology)

Hub:

Switch:

Commands: (from PC-0)

```
ping 10.0.0.3
>> Pinging 10.0.0.3 with 32 bytes of data:
>> Reply from 10.0.0.3: bytes=32 time=1ms 0% loss
>>
>> ping statistics for 10.0.0.3:
>>    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
>>    Approx. round trip times in milliseconds:
>>        Minimum=0ms, Maximum=1ms, Average=0ms
```

Procedure:

- select 4 end devices & configure IP addresses (same network) to each of them
- select a hub/switch & connect each end device to them
- send a simple PDU from one device to another

Results:

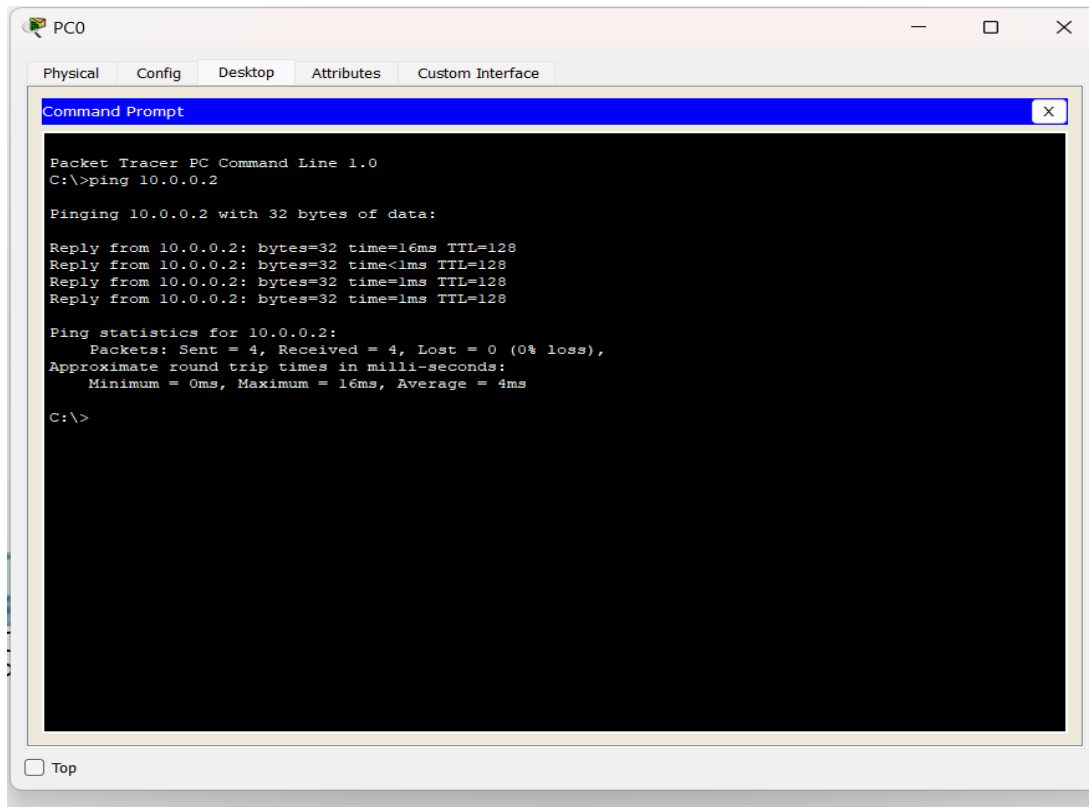
- The PDU is sent & received accordingly

Observation:

- HUB is a simple device which sends the PDU received to every other device connected. Works in physical layer
- Switch is a smart device which sends to PDU received to the device waiting for it. Works in 2 layers.

10/12/11

Output



The screenshot shows a Packet Tracer PC window titled 'PC0'. It has tabs for 'Physical', 'Config', 'Desktop', 'Attributes', and 'Custom Interface'. The 'Desktop' tab is active, displaying a 'Command Prompt' window. The command prompt shows the following output:

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

Pinging 10.0.0.2 with 32 bytes of data:

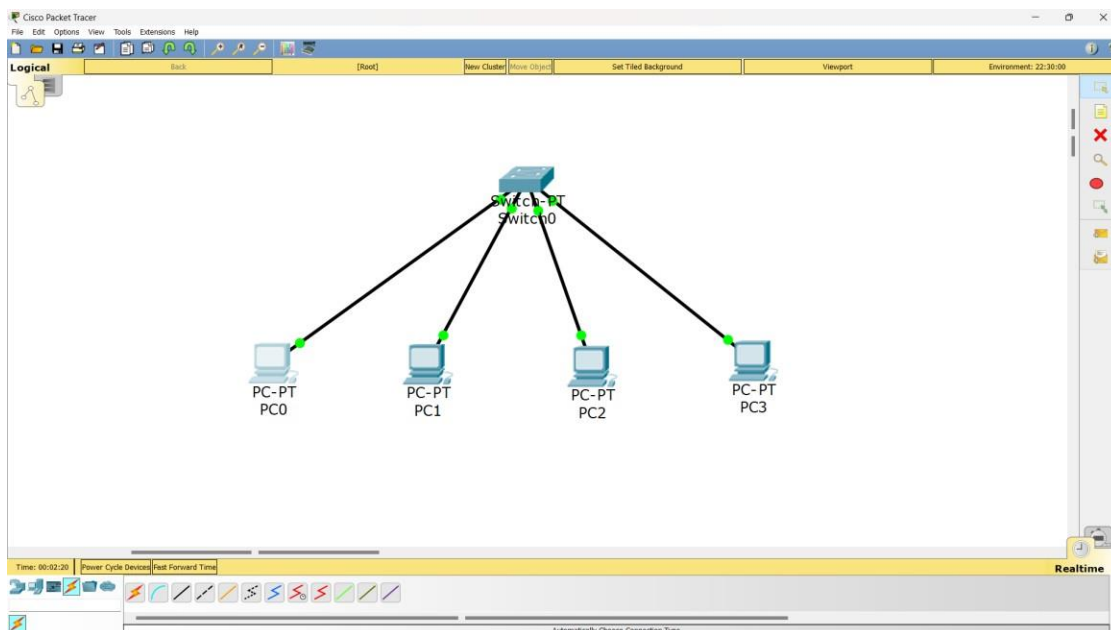
Reply from 10.0.0.2: bytes=32 time=16ms TTL=128
Reply from 10.0.0.2: bytes=32 time<1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128

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 = 16ms, Average = 4ms

C:\>
```

Switch

Topology



Procedure

* Commands: From PC-01

```
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<1ms 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),
>>    Approx. round trip times in milliseconds:
>>        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

* Procedure:

- select 4 end devices & configure IP addresses (same network) to each of them
- select a hub/switch & connect each end device to them
- send a simple PDU from one device to another

* Results:

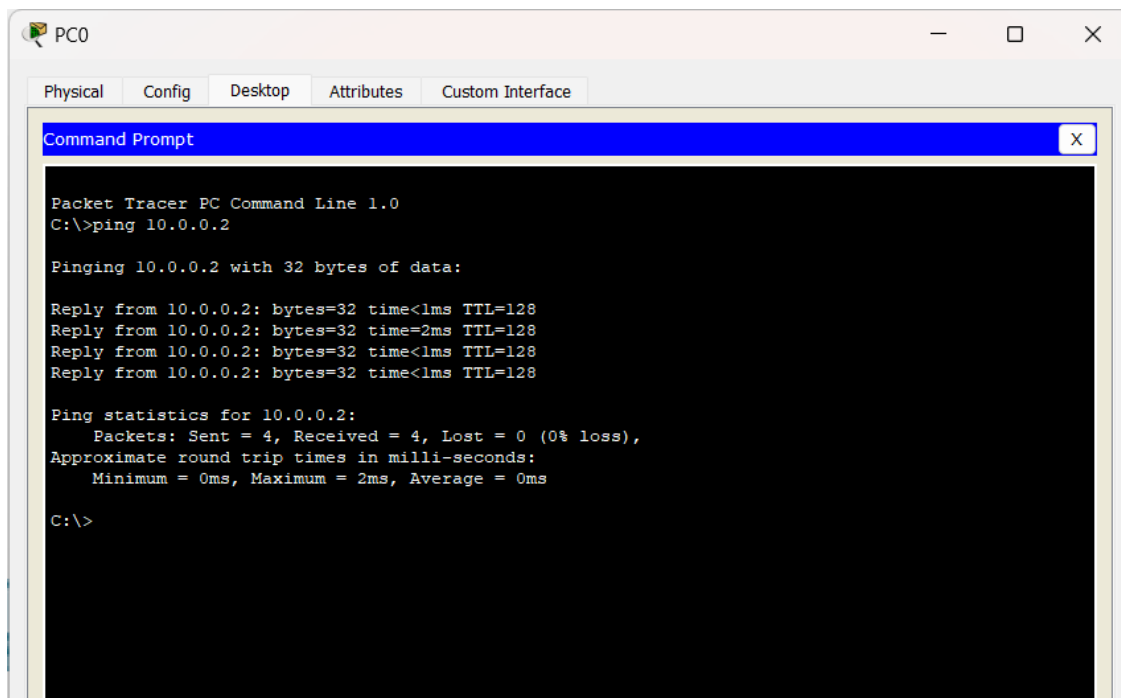
- The PDU is sent & received accordingly

* Observation:

- HUB is a simple device which sends the PDU received to every other device connected. works in physical layer
- Switch is a smart device which sends to PDU received to the device awaiting for it. works in 2 layers.

(10) Rm

Output



```
PC0
Physical Config Desktop Attributes Custom Interface
Command Prompt

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<1ms 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

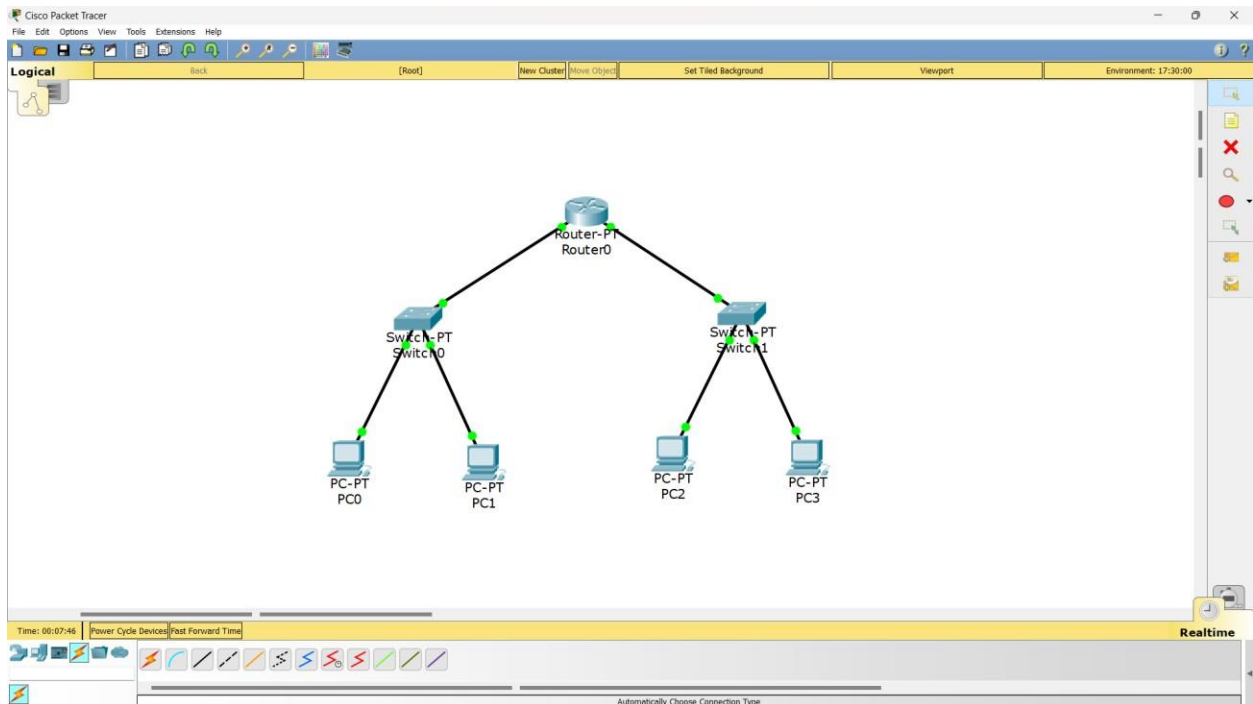
C:\>
```

Experiment No 2

Aim of the program

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

Topology



Procedure

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 10.0.0.10 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

Router(config-if)#exit
Router(config)#
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet1/0
Router(config-if)#ip address 20.0.0.10 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet1/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up

Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet1/0
Router(config-if)#
```

Aim: To Configure IP address to Router in Packet Tracer. Explore following messages: Request, Response, Destination unreachable, Request timed out, Reply.

Topology:

Procedure:

- select two end devices & configure ip address (different network)
- select a router & connect these devices as shown above
- configure the ip for router as per the commands
- configure the gateway for 2 device (gateway = ip address of router for same network)
- send a simple PDU btw 2 devices.

Results:

- The PDU is sent & received appropriately.

Observation:

- when the device is disconnected from the router until it is ip-configured.
- Request times out btw 2 devices occur when the network are isolated to connect the network, also configure the gateways.

Ping: (Request timed out)

Ping statistics for 20.0.0.2

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

Output:

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:\>

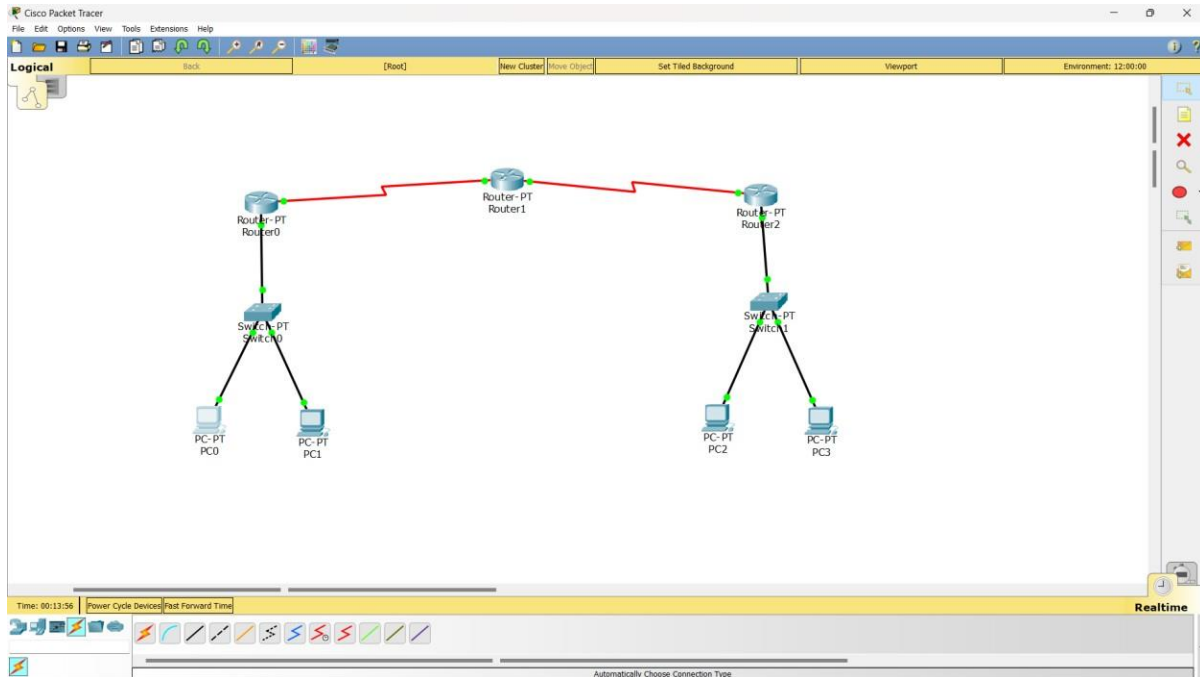
```


Experiment No 3

Aim of the program

Configuring default route to the Router

Topology



Procedure

```

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C      10.0.0.0/8 is directly connected, FastEthernet0/0
C      20.0.0.0/8 is directly connected, Serial2/0

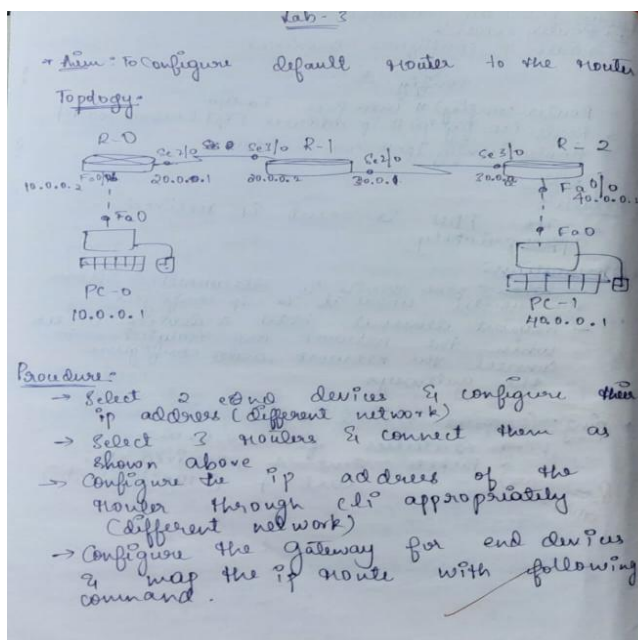
Router#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 20.0.0.2 to network 0.0.0.0

C      10.0.0.0/8 is directly connected, FastEthernet0/0
C      20.0.0.0/8 is directly connected, Serial2/0
S*    0.0.0.0/0 [1/0] via 20.0.0.2

```



Commands:

```

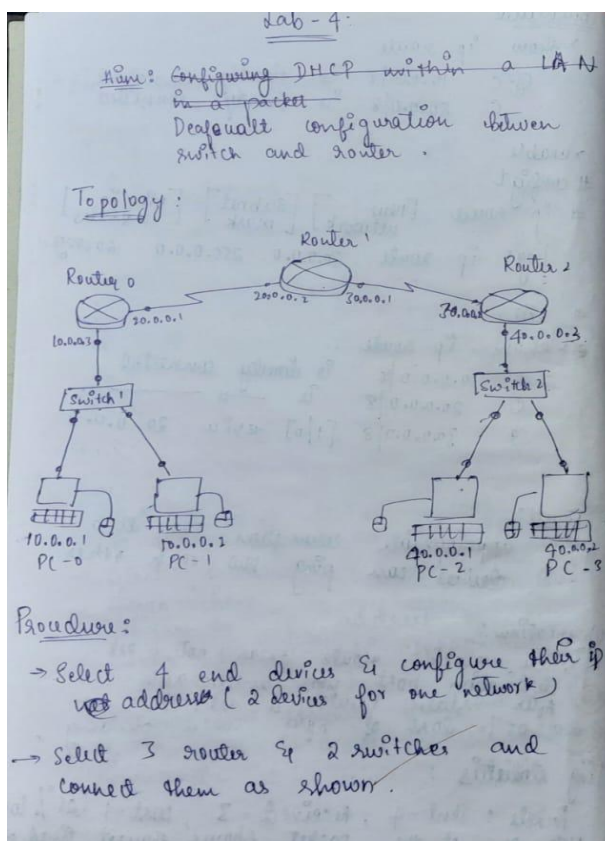
> show ip route
C 10.0.0.0/8 is directly connected
C 20.0.0.0/8 is directly connected

> enable
# config t
# ip route [new network] [subnet] [via ip address]
! eg: ip route 20.0.0.0 255.0.0.0 20.0.0.2

# exit
> show ip route
C 10.0.0.0/8 is directly connected
C 20.0.0.0/8 is directly connected
S 30.0.0.0/8 [1/0] via 20.0.0.2
  
```

* Result:
After appropriate connection, the two end devices can ping two each other.

* Observation:
Since static route was not set, destination host was unreachable. After static route is set, there is a 25% loss of bytes when pinging.
ping statistics
Packets: Sent = 4, Received = 3, Lost = 1 (25%)



→ Configure the IP addresses of routers through CLI appropriately (different networks)

→ Configure the gateway for end devices & map the IP route with following command.

Commands:

Case-1: For Router with a switch connection [Router-0 & Router-2]

```

> show ip route
C 10.0.0.0/8 is directly connected.
C 20.0.0.0/8 is directly connected.

> enable
# config t
# ip route 0.0.0.0 0.0.0.0 [via ip address]
! eg: ip route 0.0.0.0 0.0.0.0 [20.0.0.1]

# exit
> show ip route
C 10.0.0.0/8 is directly connected
C 20.0.0.0/8 is directly connected
S* 0.0.0.0/0 [1/0] via 20.0.0.2
  
```

Case-2: For Router with Router connection [Router-1]

```

> show ip route
  
```

`C 20.0.0.0/8`
`C 30.0.0.0/8`

> enable
 # config t
 # ip route [new network] [subnet mask] [via ip address]
 [eg: ip route 10.0.0.0/8 255.0.0.0 20.0.0.0]
 for all

exit
 > show ip route
 C 20.0.0.0/8 is directly connected
 C 30.0.0.0/8
 S 10.0.0.0/8 [1/0] via 20.0.0.0
 S 40.0.0.0/8 [1/0] via 30.0.0.0

Result:
 After appropriate connection, the end devices can ping btw each other.

Observation:
 After setting default routes for switched nodes, ping route for the middle router when end devices from one network ping the other network devices, there is no packet loss unlike static route.

Ping statistics:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
 Minimum = 2ms, Maximum = 12ms, Average = 4ms

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

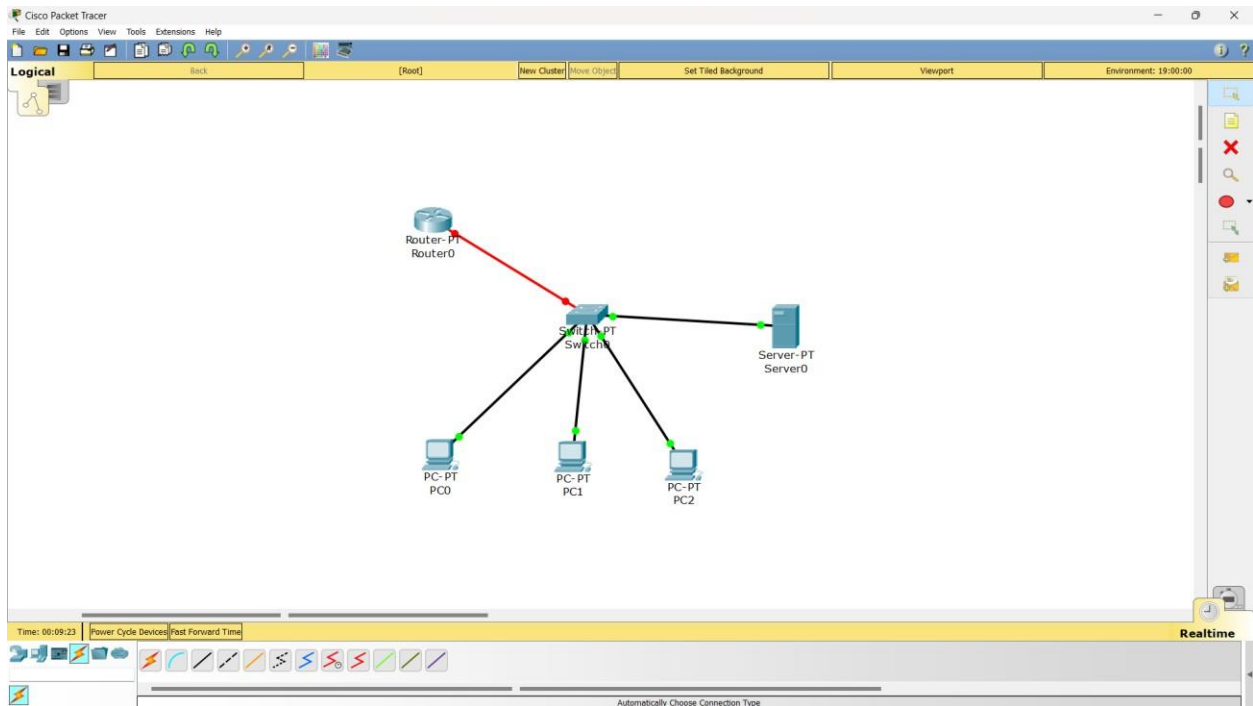
C:\>
  
```

Experiment No 4

Aim of the program

Configuring DHCP within a LAN in a packet Tracer

Topology



Procedure

The screenshot shows the configuration window for Server0, specifically the DHCP service configuration. The 'Services' tab is selected, and the 'DHCP' service is enabled. The configuration details are as follows:

Interface	Service
FastEthernet0	On

Pool Name: serverPool

Default Gateway: 10.0.0.2

DNS Server: 10.0.0.1

Start IP Address : 10.0.0.3

Subnet Mask: 255.0.0.0

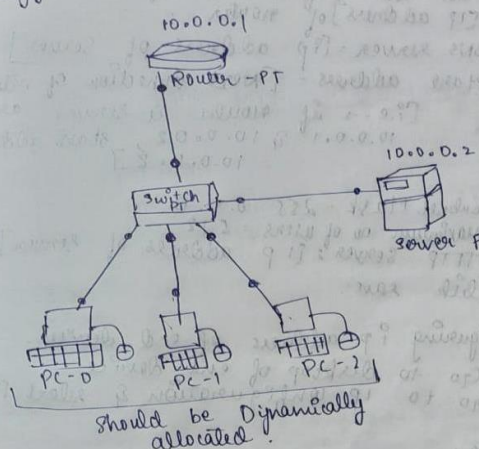
Maximum number of Users : 512

TFTP Server: 10.0.0.1

Buttons: Add, Save, Remove

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max User	TFTP Server
serverPool	10.0.0.2	10.0.0.1	10.0.0.3	255.0.0.0	512	10.0.0.1

Aim: Configuring DHCP within a LAN
Topology:



Procedure:

- Select 3 end devices, do not assign IP address to them.
- Select a switch, a server & a router & connect them as shown.
- Configure the IP address for the server (static) & the Router (through CLI) for same network.

Configuring DHCP of server:

- Go to services in server
- select DHCP & turn ON the service
- Set the default Gateway as the [IP address] of router.

→ DNS server = [IP address of server]
 → start address = [Next connection of network
 (i.e., if Router & server are 10.0.0.1 & 10.0.0.2, start address = 10.0.0.2)]

- Subnet Mask = 255.0.0.0
- Maximum no of users = 512
- TFTP server: [IP address of server]
- click save.

Configuring IP address of end devices:

- Go to desktop of end device
- Go to IP configuration & select DHCP

* Result:
 The IP addresses of end devices are ~~connected~~ allocated dynamically.

* Observations:
 DHCP (Dynamic Host Configuration Protocol) is a client/server protocol that automatically provides an IP (Internet Protocol) with all IP address network as base & other related configuration. DHCP application DORA (Discover, Offer, Request, Acknowledgment)

Output

```

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

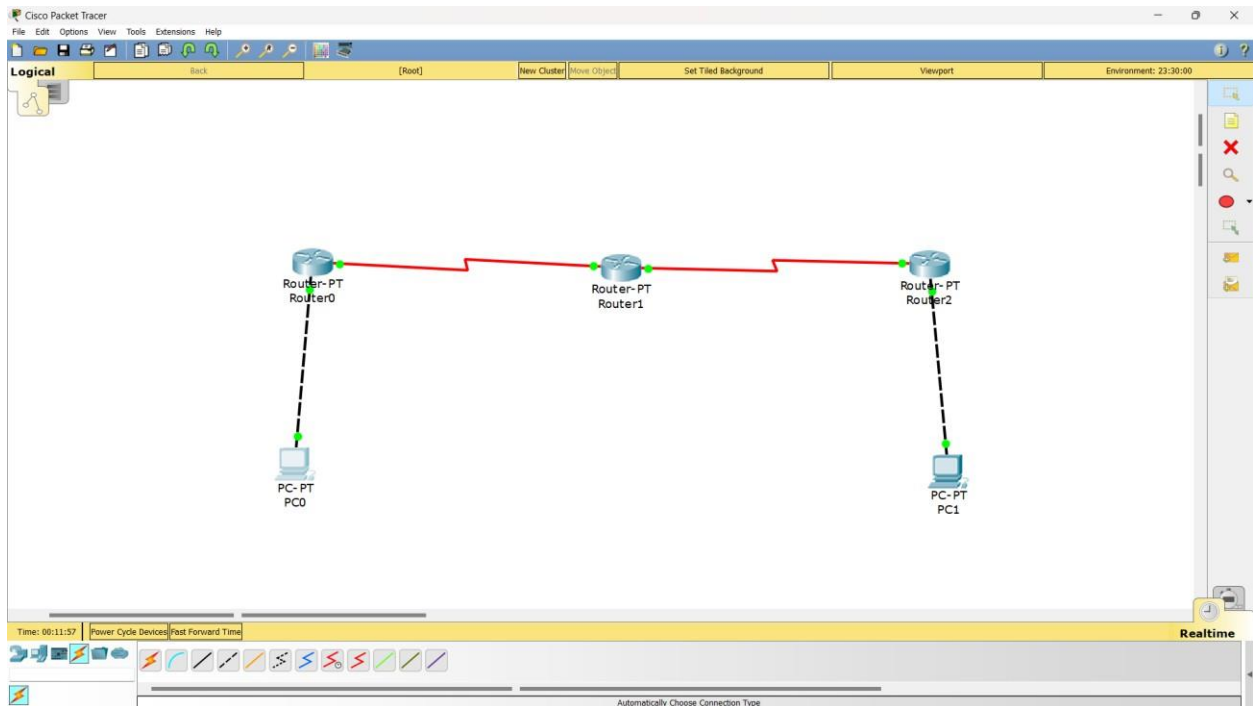
C:\>
    
```

Experiment No 5

Aim of the program

Configuring RIP Routing Protocol in Routers

Topology



Procedure

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 10.0.0.10 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#ip address 30.0.0.1 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#network 30.0.0.0
Router(config-router)#exit
Router(config)#
Router(config)#interface Serial2/0
Router(config-if)#no shutdown

Router(config-if)#

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#ip address 30.0.0.2 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#clock rate 64000
This command applies only to DCE interfaces
Router(config-if)#no shutdown

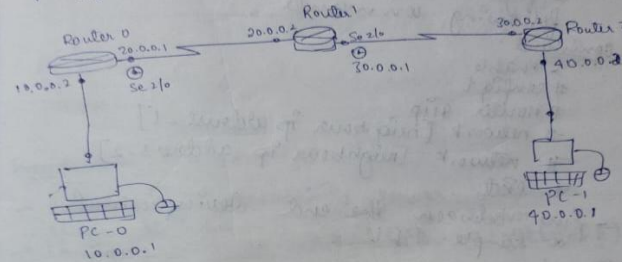
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial3/0
Router(config-if)#ip address 20.0.0.2 255.0.0.0
Router(config-if)#encapsulation ppp
Router(config-if)#clock rate 64000
Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial3/0, changed state to down
Router(config-if)#
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#network 30.0.0.0
Router(config-router)#network 20.0.0.0
Router(config-router)#exit
Router(config)#
%LINK-5-CHANGED: Interface Serial3/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up
```

Aim: Configuring ~~the~~ RIP Routing Protocol in Routers

Topology:



Procedure:

- Select 2 end devices & configure their ip address
- Select 3 routers & connect them as shown above. Configure the Fast ethernet with appropriate cli commands & gateway.
- For the Serial ports, ~~for~~ configure them with the following commands

Commands:

```

> enable
# config t
# interface [port]
# ip address [ip] [subnet mask]
# clock rate 64000 11 only for serial ports with clock symbol
# no shutdown
# exit

```

→ Configure the ~~the~~ ^{RIP} for routers with following commands

Command:

```

> enable
# config t
# router rip
# network [neighbour ip address - 1]
# network [neighbour ip address - 2]
# exit

```

→ ping between the end device & send a simple PDU

Result:

The devices are able to ping each other a simple PDU is passed through

Observation:

RIP is a dynamic routing protocol that uses hop count as a routing metric to find the best path between source & destination network. ~~being~~ The end devices are able to ping with each other [but the first ping had 35% loss]

Ping statistics:

Packets: Sent = 4, Received = 4, Cost = 0.60. Loss
Minimum = 4ms, Maximum = 12ms, Average = 5ms

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

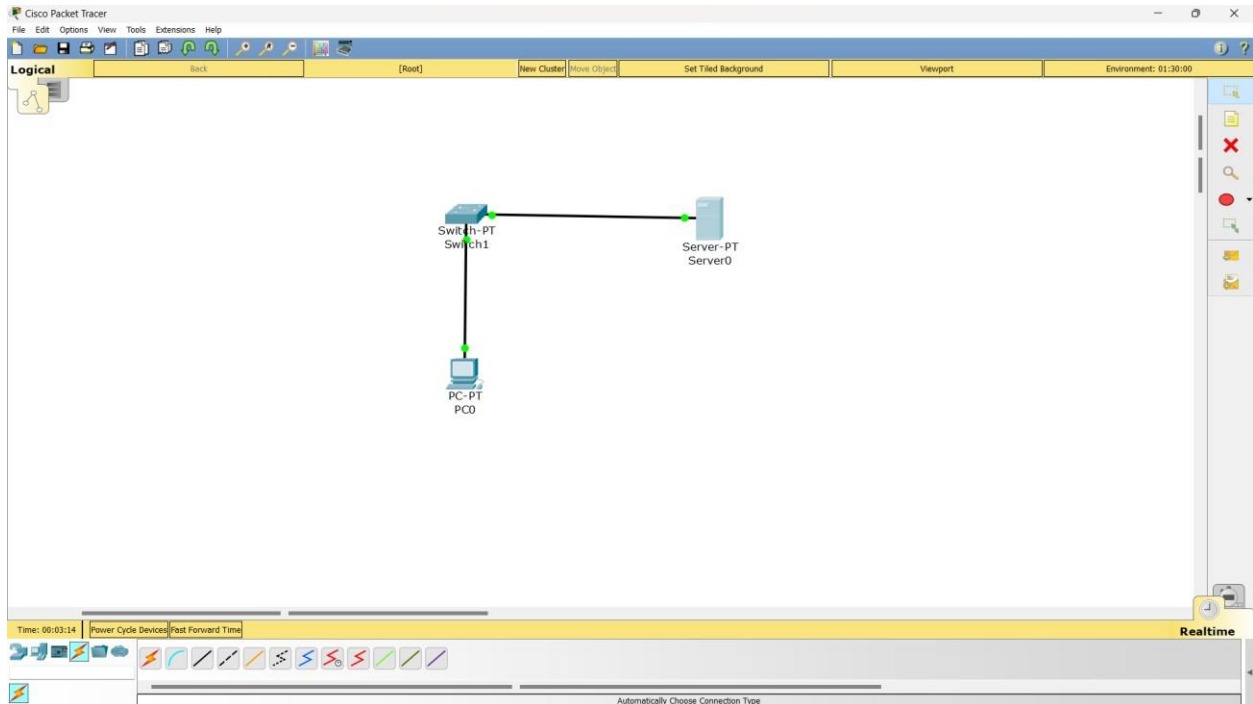
C:\>
```


Experiment No 6

Aim of the program

Demonstration of WEB server and DNS using Packet Tracer

Topology



Procedure

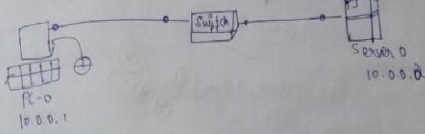
The screenshot shows the configuration window for 'Server0' in Packet Tracer. The 'Services' tab is selected. On the left, a list of services includes HTTP, DHCP, DHCPv6, TFTP, DNS, SYSLOG, AAA, NTP, EMAIL, FTP, IoE, and VM Management. The 'DNS' service is selected, and its status is 'On'. The 'Resource Records' section shows a table with one record:

No.	Name	Type	Detail
0	www.bgy.com	A Record	10.0.0.10

Lab-6

Aim: Demonstration of WEB server & DNS.

Topology:



Procedure:

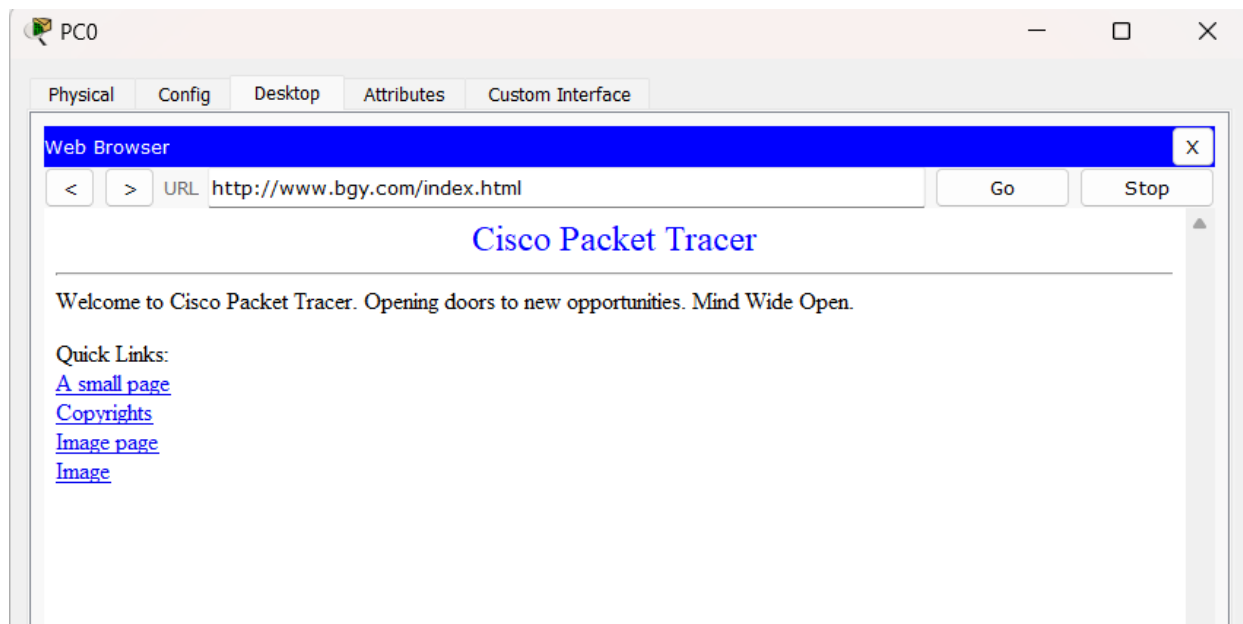
- Select a end device & server & connect them through a switch as shown.
- Configure the ip address (same network)
- Go to services in server
- enable HTTP & DNS
- Add a domain name eg. www.xyz.com
Address = [Server ip] eg. 10.0.0.2
- Add & save.
- Go to Web browser in end device & search for the added Domain name
- Try to go to search for domain name.html

Result:
The added domain name is accessible

Observation:
Consider an HTTP file name helloworld in server. when going through ~~http~~ www.xyz.com/helloworld.html in web browser, the changes done to helloworld.html is reflected in the browser

R
15/12/22

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(1_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()
{

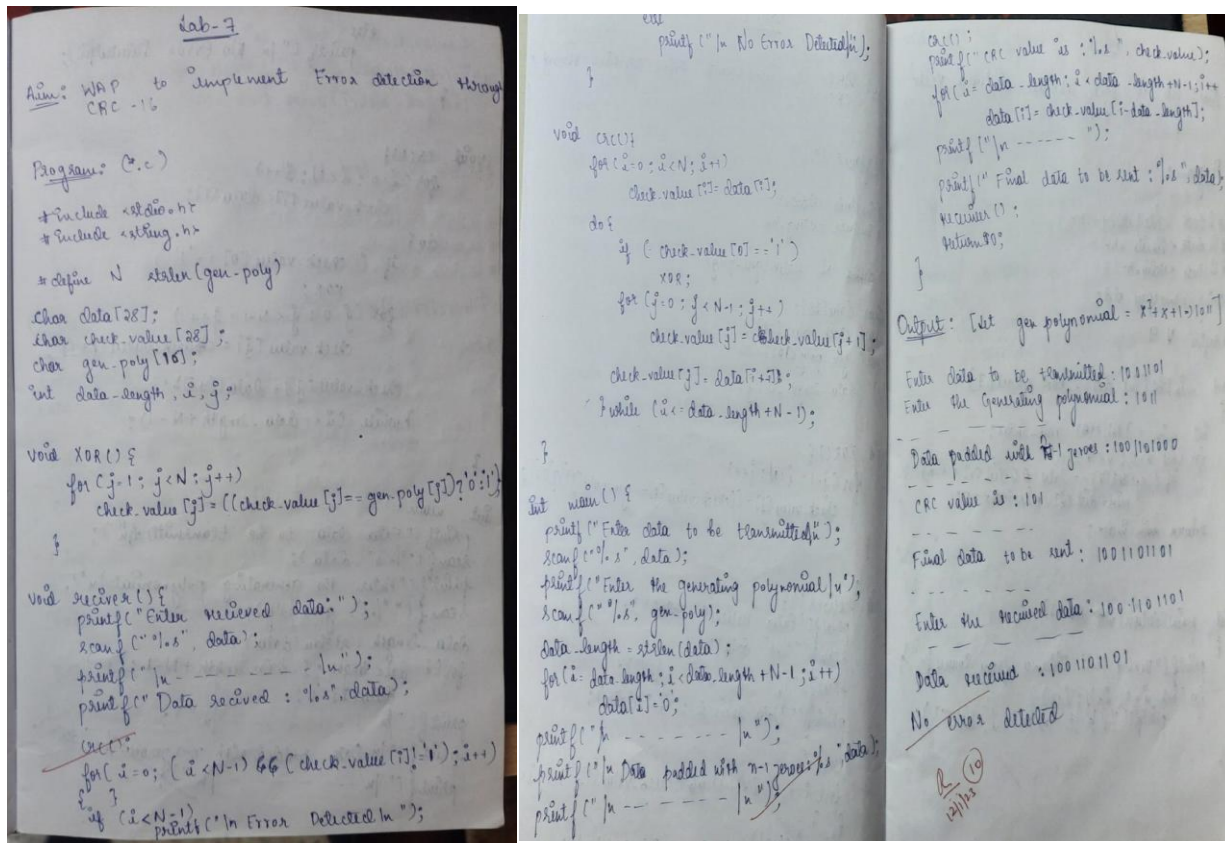
    string data = "1011101";
    string key = "1000100000001";

    encodeData(data, key);

    return 0;
}

```

Observation:



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 of the Experiment

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 - 9

Aim: WAP for distance vector Algorithm to find suitable path for transmission.

Program:

```
#include <stdio.h>
struct node {
    unsigned dist[20];
    unsigned from[20];
} nt[10];

int main()
{
    int dmat[20][20];
    int n, i, j, k, count = 0;
    printf("\nEnter the number of nodes: ");
    scanf("%d", &n);
    printf("\nEnter cost matrix: ");
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            scanf("%d", &dmat[i][j]);
    dmat[i][i] = 0;
    nt[i].dist[j] = dmat[i][j];
    nt[i].from[j] = j;
    do {
        count = 0;
        for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                for(k = 0; k < n; k++)
                    if (nt[i].dist[j] > nt[i].dist[k] + nt[k].dist[j])
                        nt[i].dist[j] = nt[i].dist[k] + nt[k].dist[j];
                        nt[i].from[j] = k;
                        count++;
    } while(count != 0);

    for(i = 0; i < n; i++)
        printf("%d\t", nt[i].dist[i]);
    printf("\n");
}
```

Output:

Enter number of nodes: 4

Enter cost matrix

```
0 3 5 99
3 0 99 1
5 4 0 2
99 99 99 0
```

State value for route 1

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

State value for route 2

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

State value for route 3

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

State value for route 4

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

RVC
4/1/23

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 of the Experiment

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-8
Ques: WAP to find shortest distance vertex for transmission.

Program:

```
#include <bits/stdc++.h>
using namespace std;
#define V 9

int minDist(int dist[], bool sptset[])
{
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (!sptset[v] && dist[v] < min)
            min = dist[v], min_index = v;
    return min_index;
}

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

void dij(int graph[V][V], int src)
{
    int dist[V];
    bool sptset[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptset[i] = false;
    dist[src] = 0;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minDist(dist, sptset);
        sptset[u] = true;
        for (int v = 0; v < V; v++)
            if (!sptset[v] && graph[u][v] && 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];
    dij(graph, 0);
    return 0;
}
```

Ex: Enter the graph.

0	0	0	0	0	0	8	0
6	0	8	0	0	0	0	13
0	8	0	7	0	6	0	0
0	0	7	0	7	14	0	0
0	0	0	7	0	10	0	0
5	0	6	0	1	0	0	0
0	0	0	0	0	2	0	1
8	13	0	0	0	0	1	0
0	0	2	0	0	0	6	7

Vertex	Distance from Source
0	0
1	6
2	14
3	21
4	21
5	11
6	9
7	8
8	15

Q. 10
10/10

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 of the Experiment

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);
    }
}
```



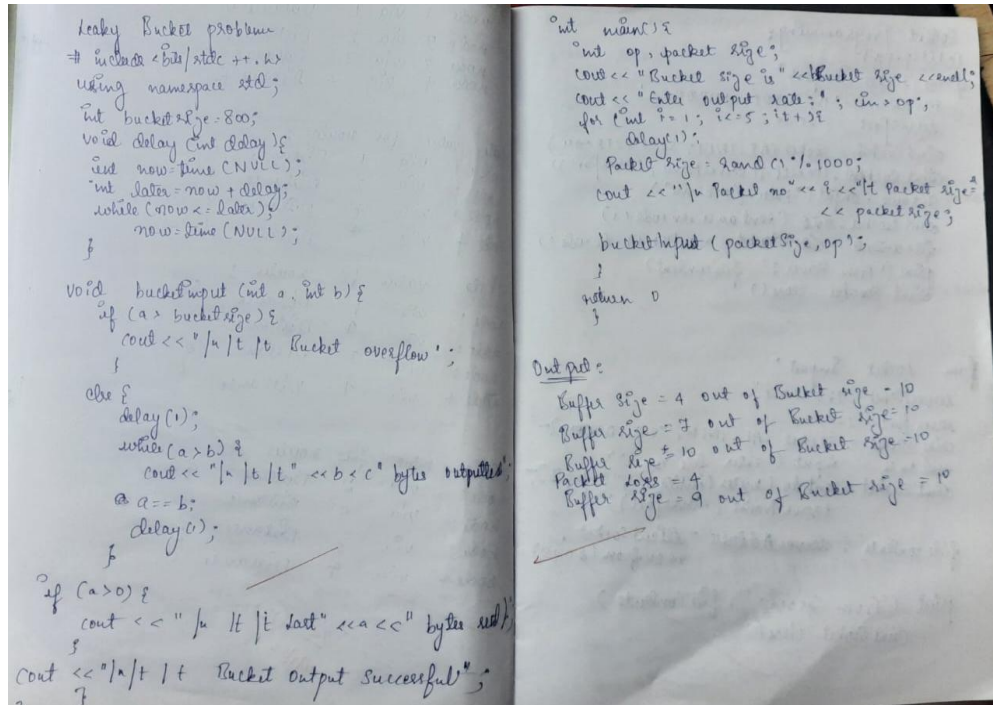
```

        storage -= output_pkt_size;
    }

    return 0;
}

```

Observation:



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 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"

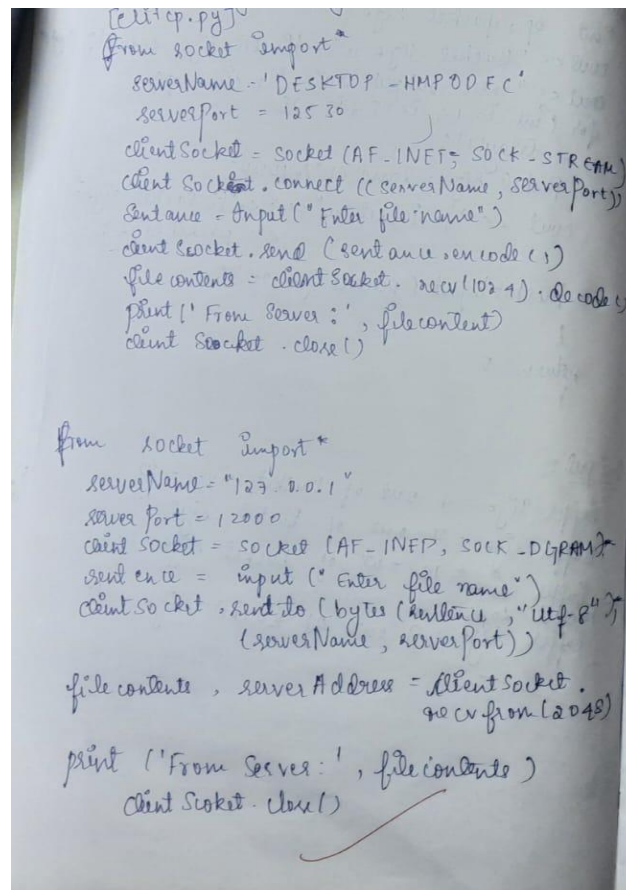
        connectionSocket.send(message.encode())

    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()
```

Observation:



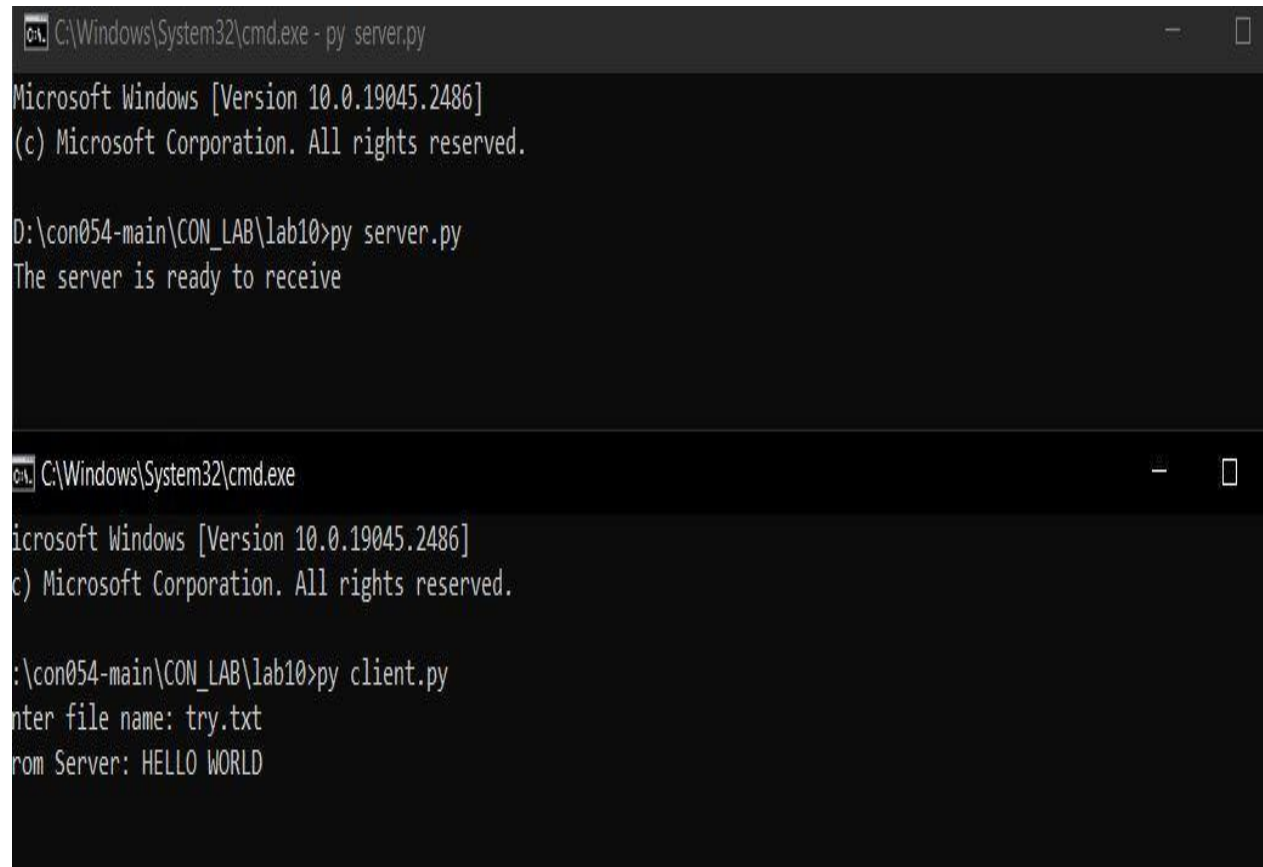
The image shows two handwritten Python code snippets on a piece of paper. The top snippet is for a client, and the bottom snippet is for a server. Both use the socket module to establish a connection and exchange data.

```
[client.py]  
from socket import *  
serverName = 'DESKTOP-HMP0DFC'  
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()
```



```
from socket import *  
serverName = "192.0.0.1"  
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()
```

Output



The image displays two separate Windows command prompt windows. The top window, titled 'C:\Windows\System32\cmd.exe - py server.py', shows the execution of a server script. It displays the Windows version (10.0.19045.2486) and copyright information, followed by the command 'py server.py' and the output 'The server is ready to receive'. The bottom window, titled 'C:\Windows\System32\cmd.exe', shows the execution of a client script. It displays the same Windows version and copyright information, followed by the command 'py client.py'. The client script prompts for a file name, which is entered as 'try.txt', and then outputs 'From Server: HELLO WORLD'.

```
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
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

D:\con054-main\CON_LAB\lab10>py client.py
Enter file name: try.txt
From Server: HELLO WORLD
```

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)

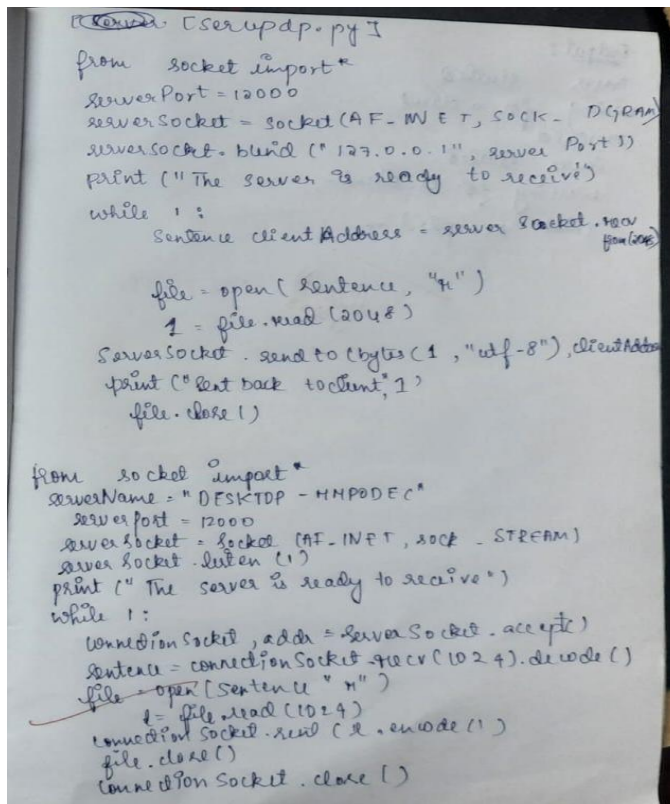
sentence = input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
```

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

```
print ('From Server:', filecontents)
```

```
clientSocket.close()
```

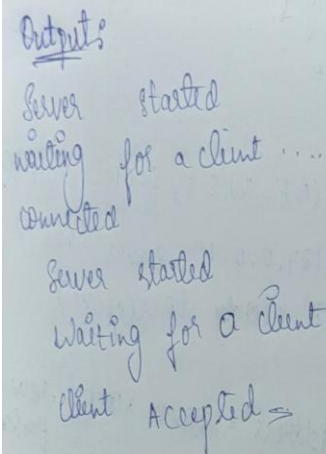
Observation:



Handwritten code for `server.py`:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    file = open(sentence, "w")
    data = file.read(2048)
    serverSocket.sendto(bytes(data, "utf-8"), clientAddress)
    print("Sent back to client")
    file.close()
```

```
from socket import *
serverName = "DFSKTDP - HMPDEC"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "w")
    data = file.read(1024)
    connectionSocket.send(data.encode())
    file.close()
    connectionSocket.close()
```



Handwritten output notes:

Output:

Server started
waiting for a client ...
connected

Server started
waiting for a client ...
Client Accepted =

Run
11/2/23

Output



Screenshot of a Windows command prompt window showing the execution of the `server.py` script:

```
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
```