# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"JnanaSangama", Belgaum -590014, Karnataka.**

**LAB REPORT**
**on**

# COMPUTER NETWORKS

***Submitted by***

**ANUSHREE HARRISH(1BM20CS020)**

***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 **ANUSHREE HARRISH(1BM20CS020),** 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. Latha N R
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

# Index

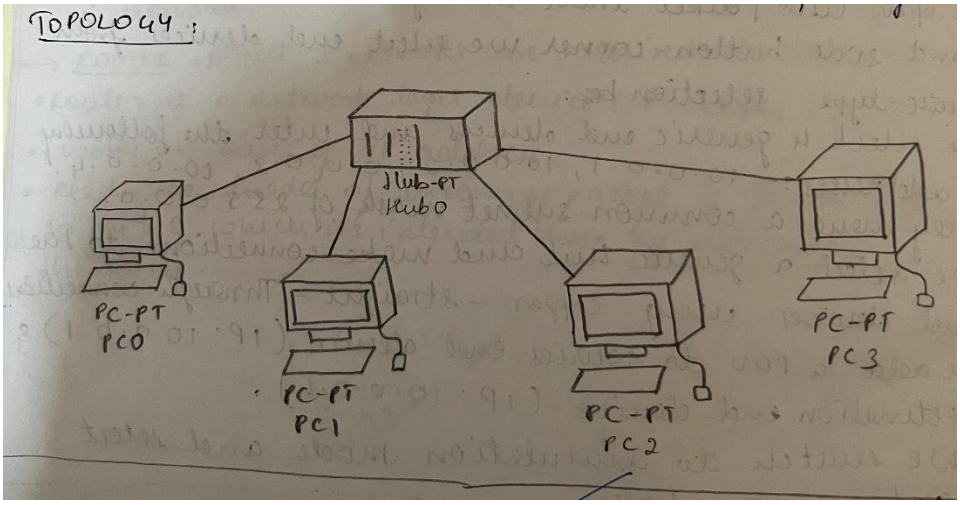
# Cycle-1

# Experiment No-1

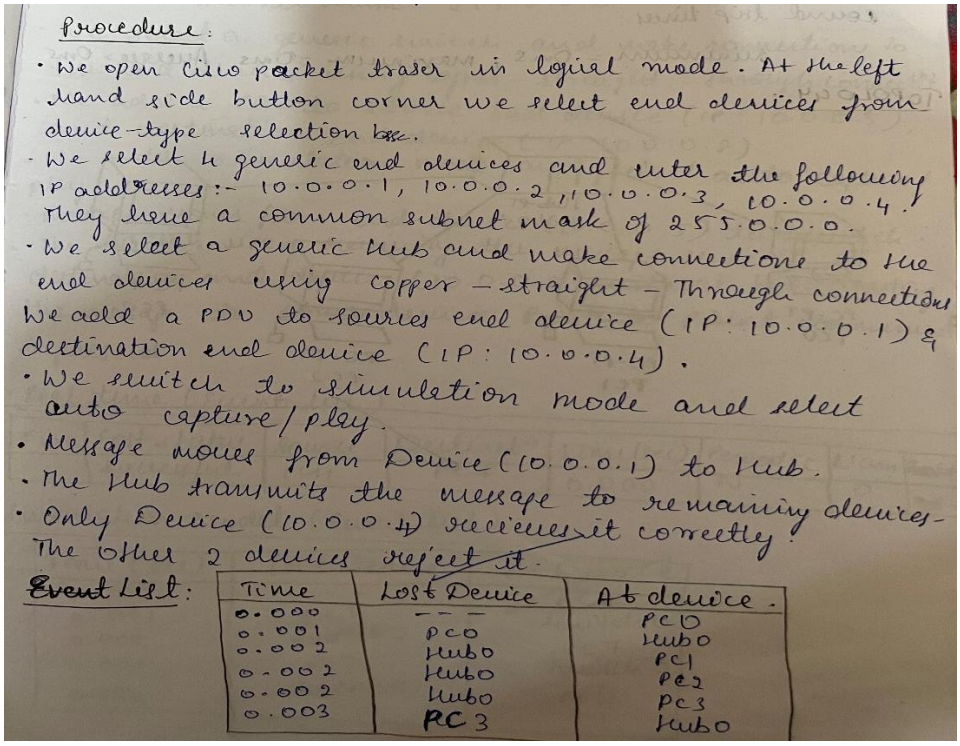
**Aim of the program-** Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.

**USING HUB AS CONNECTING DEVICE:**

**TOPOLOGY:**




**PROCEDURE:**



Procedure:

- We open Cisco packet tracer in logical mode. At the left hand side button corner we select end devices from device-type selection box.
- We select 4 generic end devices and enter the following IP addresses:- 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4. They have a common subnet mask of 255.0.0.0.
- We select a generic hub and make connections to the end devices using copper – straight – Through connections.

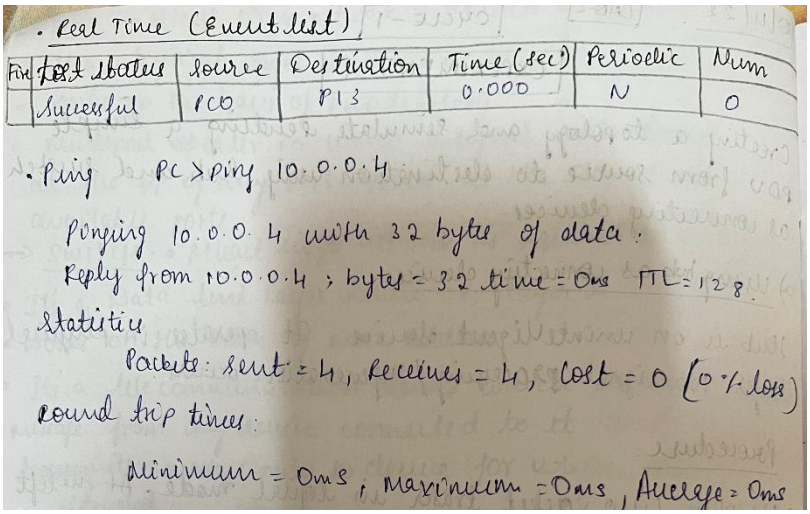
We add a PDU to source end device (IP: 10.0.0.1) & destination end device (IP: 10.0.0.4).

- We switch to simulation mode and select auto capture/play.
- Message moves from Device (10.0.0.1) to Hub.
- The Hub transmits the message to remaining devices.
- Only Device (10.0.0.4) receives it correctly. The other 2 devices reject it.

Event List:

| Time | Last Device | At device |
|---|---|---|
| 0.000 | --- | PC0 |
| 0.001 | PC0 | Hub0 |
| 0.002 | Hub0 | PC1 |
| 0.002 | Hub0 | PC2 |
| 0.002 | Hub0 | PC3 |
| 0.003 | RC 3 | Hub0 |

## Output:



• Real Time (Event list)

| Fire | Last Status | Source | Destination | Time (sec) | Periodic | Num |
|---|---|---|---|---|---|---|
| | Successful | PC0 | P13 | 0.000 | N | 0 |

• Ping PC > ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4 : bytes = 32 time = 0ms TTL = 128

Statistics

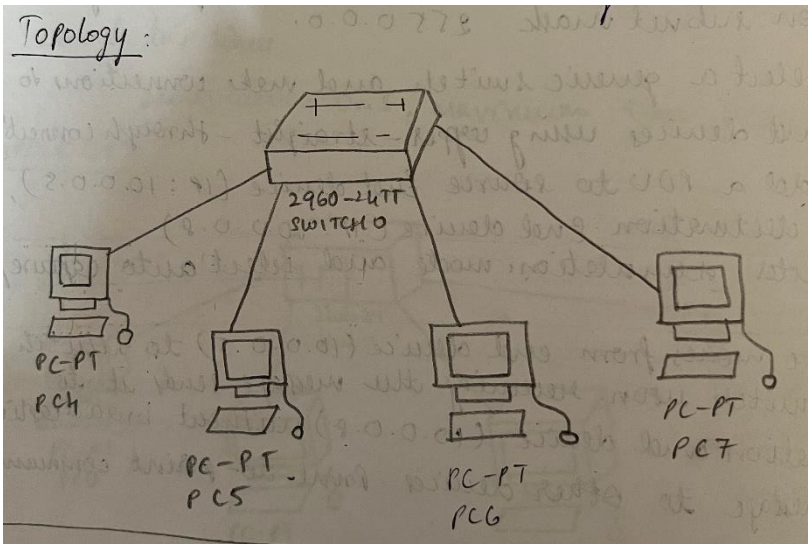
Packets: sent = 4, Received = 4, Lost = 0 (0% loss)

Round trip times:

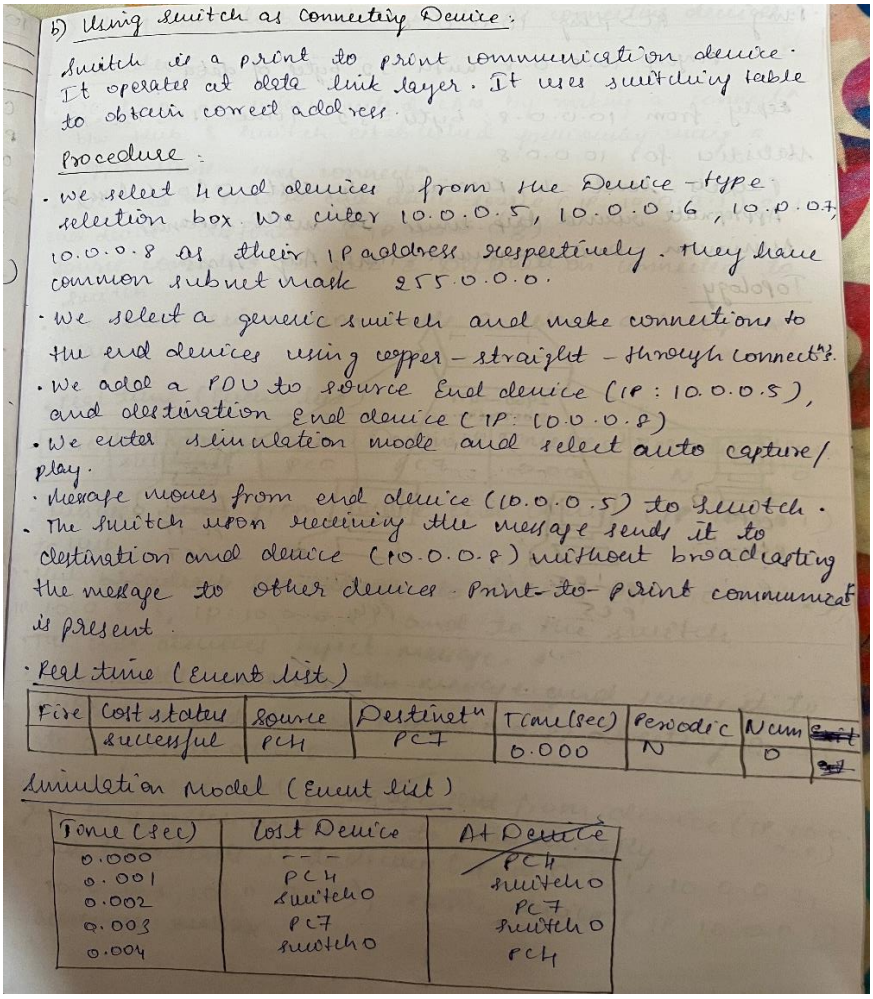
Minimum = 0ms, Maximum = 0ms, Average = 0ms

**USING SWITCH AS CONNECTING DEVICE:**

**TOPOLOGY:**



**PROCEDURE:**



b) Using switch as connecting Device:

Switch is a print to print communication device. It operates at data link layer. It uses switching table to obtain correct address.

Procedure:

- We select 4 end devices from the Device-type selection box. We enter 10.0.0.5, 10.0.0.6, 10.0.0.7, 10.0.0.8 as their IP address respectively. They have common subnet mask 255.0.0.0.
- We select a generic switch and make connections to the end devices using copper-straight-through connects.
- We add a PDU to source End device (IP: 10.0.0.5), and destination End device (IP: 10.0.0.8)
- We enter simulation mode and select auto capture/play.
- Message moves from end device (10.0.0.5) to Switch.
- The switch upon receiving the message sends it to destination end device (10.0.0.8) without broadcasting the message to other devices. Print-to-Print communicat is present.

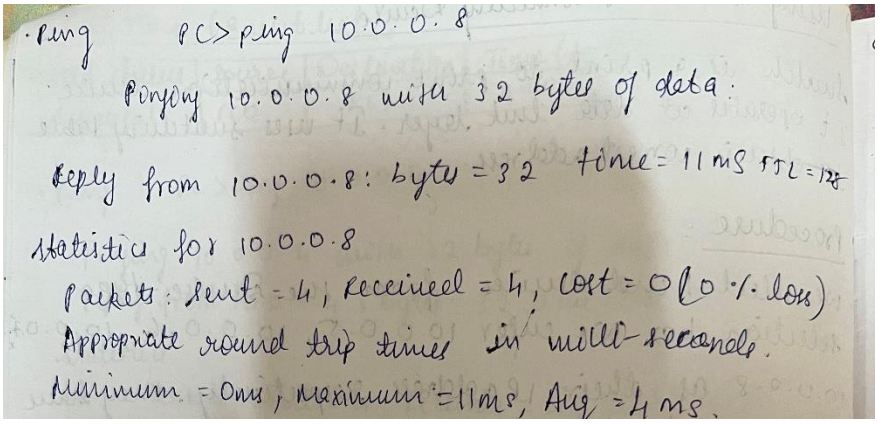
- Real time (Event list)

| Fire | Last status | Source | Destinat" | Time(sec) | Periodic | Num | Edit |
|---|---|---|---|---|---|---|---|
| | successful | PC4 | PC7 | 0.000 | N | 0 | |

Simulation Model (Event list)

| Time (sec) | Last Device | At Device |
|---|---|---|
| 0.000 | - - - | PC4 |
| 0.001 | PC4 | switch0 |
| 0.002 | switch0 | PC7 |
| 0.003 | PC7 | switch0 |
| 0.004 | switch0 | PC4 |

## Output:



- Ping PC> ping 10.0.0.8

Pinging 10.0.0.8 with 32 bytes of data:

Reply from 10.0.0.8: bytes = 32 time = 11ms TTL = 128

Statistics for 10.0.0.8

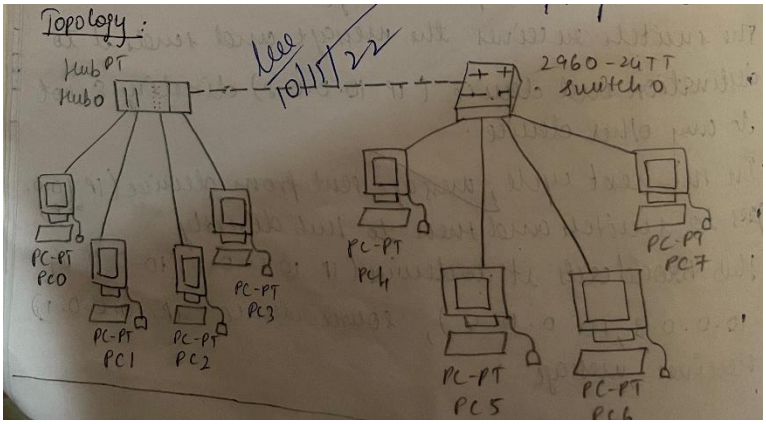
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip times in milli-seconds.

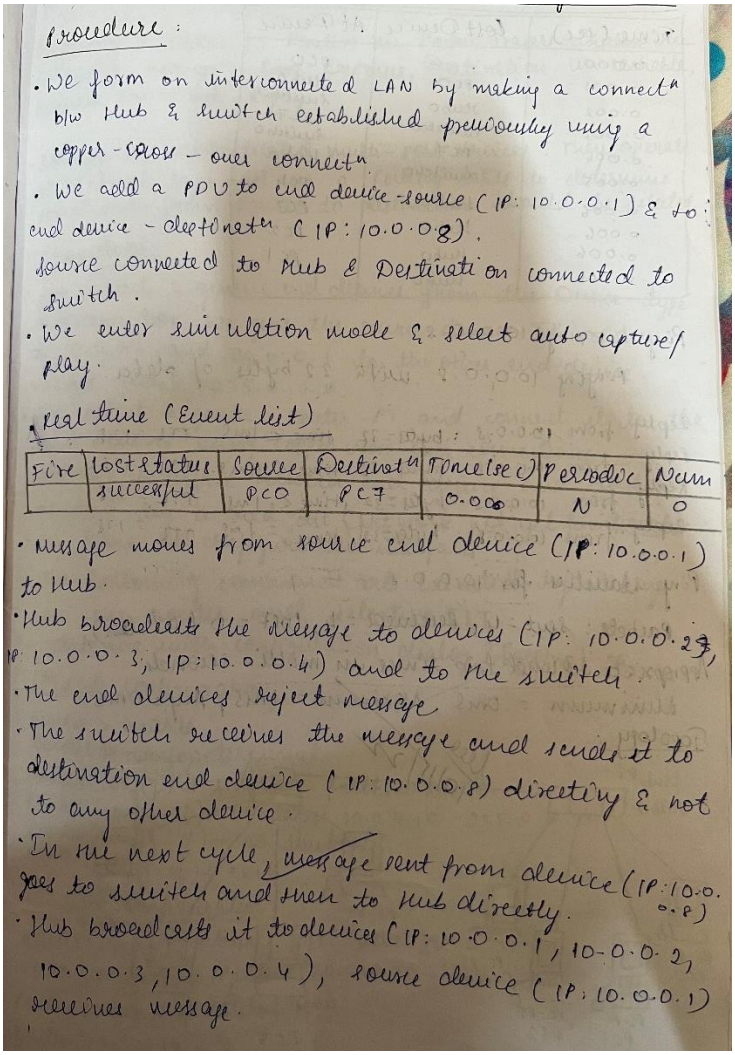
Minimum = 0ms, Maximum = 11ms, Avg = 4 ms.

**USING HUB AND SWITCH AS CONNECTING DEVICE:**

**TOPOLOGY:**




**PROCEDURE:**



Procedure:

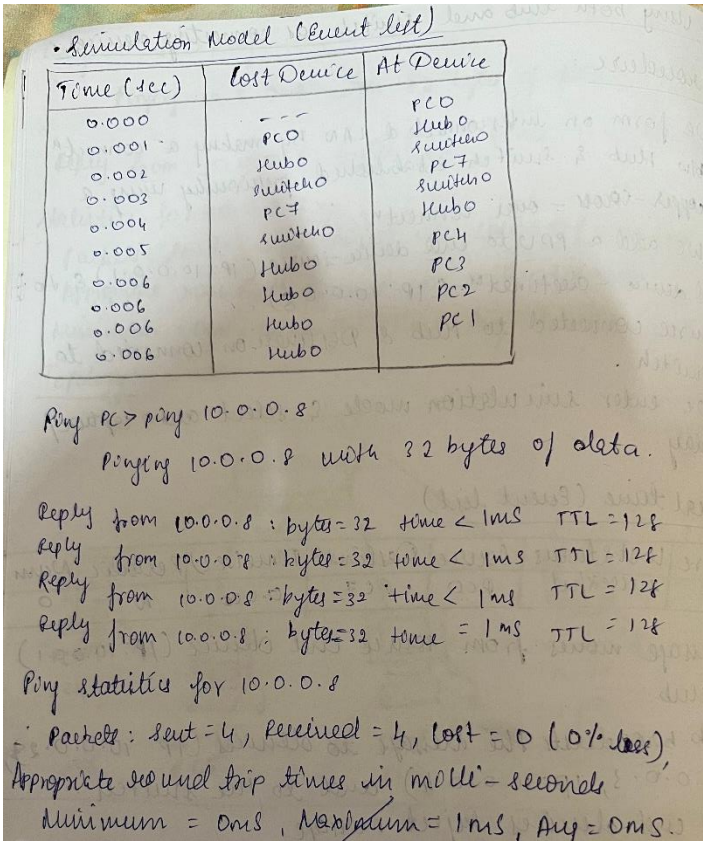
- We form an interconnected LAN by making a connection b/w Hub & switch established previously using a copper-cross-over connection.
- We add a PDU to end device-source (IP: 10.0.0.1) & to end device - destination (IP: 10.0.0.8). Source connected to Hub & Destination connected to switch.
- We enter simulation mode & select auto capture/play.

Real time (Event list)

| Fire | Last Status | Source | Destination | Time(sec) | Periodic | Num |
|---|---|---|---|---|---|---|
| | Successful | PC0 | PC7 | 0.000 | N | 0 |

- Message moves from source end device (IP: 10.0.0.1) to Hub.
- Hub broadcasts the message to devices (IP: 10.0.0.2, IP: 10.0.0.3, IP: 10.0.0.4) and to the switch.
- The end devices reject message
- The switch receives the message and sends it to destination end device (IP: 10.0.0.8) directly & not to any other device.
- In the next cycle, message sent from device (IP: 10.0.0.8) goes to switch and then to Hub directly.
- Hub broadcasts it to devices (IP: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4), source device (IP: 10.0.0.1) receives message.

## Output:



• Simulation Model (Event list)

| Time (sec) | Last Device | At Device |
|---|---|---|
| 0.000 | - - - | PC0 |
| 0.001 | PC0 | Hub0 |
| 0.002 | Hub0 | Switch0 |
| 0.003 | Switch0 | PC7 |
| 0.004 | PC7 | Switch0 |
| 0.005 | Switch0 | Hub0 |
| 0.006 | Hub0 | PC4 |
| 0.006 | Hub0 | PC3 |
| 0.006 | Hub0 | PC2 |
| 0.006 | Hub0 | PC1 |
| 0.006 | Hub0 | |

Ping PC> ping 10.0.0.8

Pinging 10.0.0.8 with 32 bytes of data.

Reply from 10.0.0.8 : bytes=32 time<1ms TTL=128
Reply from 10.0.0.8 : bytes=32 time<1ms TTL=128
Reply from 10.0.0.8 : bytes=32 time<1ms TTL=128
Reply from 10.0.0.8 : bytes=32 time=1ms TTL=128

Ping statistics for 10.0.0.8

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

Approximate round trip times in milli-seconds

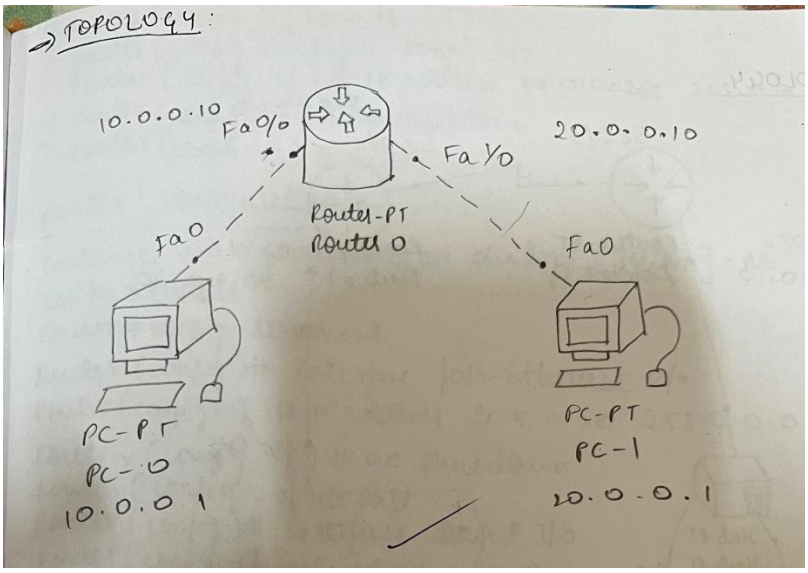
Minimum = 0ms, Maximum = 1ms, Avg = 0ms.

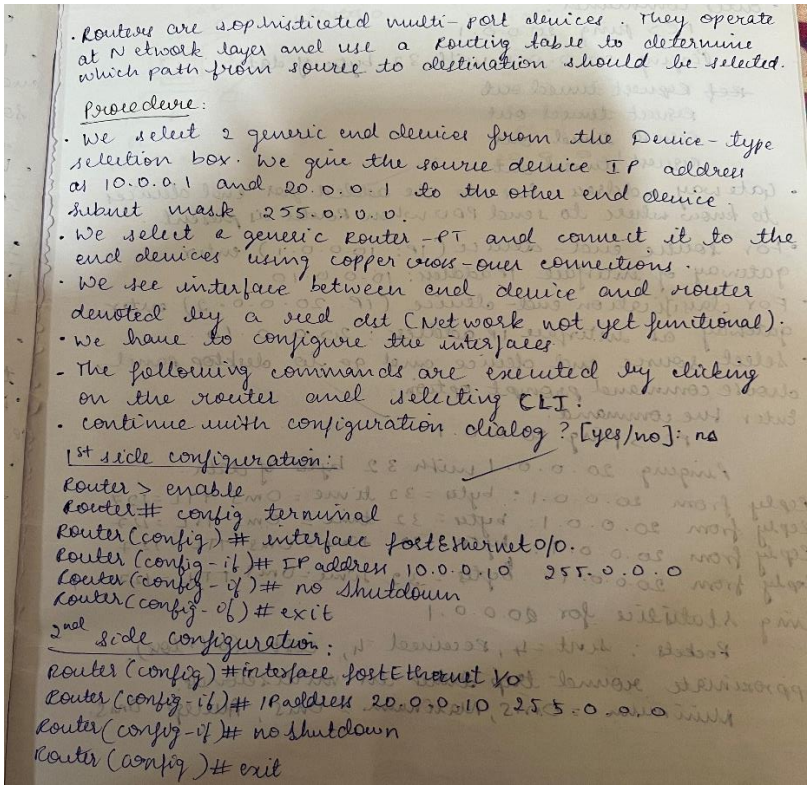
# Experiment No-2

**Aim of the program-** Configuring IP address to Routers in Packet Tracer. Explore the following messages: Ping Responses, Destination unreachable, Request timed out, Reply.

**TOPOLOGY, PROCEDURE AND OUTPUT:**

**PART-1**



- Routers are sophisticated multi-port devices. They operate at N etwork layer and use a routing table to determine which path from source to destination should be selected.

Procedure:

- We select 2 generic end devices from the Device-type selection box. We give the source device IP address as 10.0.0.1 and 20.0.0.1 to the other end device. Subnet mask 255.0.0.0.
- We select a generic Router-PT and connect it to the end devices using copper cross-over connections.
- We see interface between end device and router denoted by a red dot (Network not yet functional).
- We have to configure the interfaces
- The following commands are executed by clicking on the router and selecting CLI:
- Continue with configuration dialog? [yes/no]: no

1st side configuration:

```
Router > enable
Router# config terminal
Router(config)# interface fastEthernet 0/0.
Router(config-if)# IP address 10.0.0.10 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
```

2nd side configuration:

```
Router(config)#interface fastEthernet 1/0
Router(config-if)# IP address 20.0.0.10 255.0.0.0
Router(config-if)# no shutdown
Router(config)# exit
```

- The interfaces (represented by red dot) turn green. This indicates that the network is functional.
- We added PDU's to the end devices.
- We ~~for~~ select the source end devices and go to the command prompt option in Desktop panel.
- Enter command:

PC> ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data.

~~Ref~~ Request timed out
Request timed out
Request timed out
Request timed out

- Gateway address has to be added for end devices to know where to send PDU when Router is present.
- For source end-device (IP: 10.0.0.1) enter gateway of interface IP address: 10.0.0.10
- For destination end-device (IP: 20.0.0.2) enter gateway as interface IP address: 20.0.0.10.
- Select source end device and go to desktop panel, choose command prompt option.
- Enter the command:

PC > ping 20.0.0.1

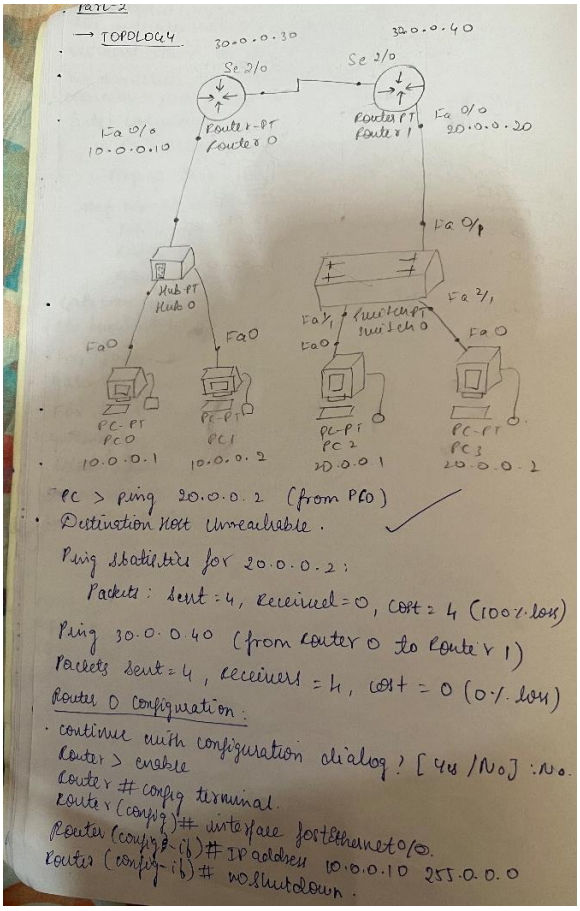
Pinging 20.0.0.1 with 32 bytes of data.

Reply from 20.0.0.1: bytes = 32 time = 0ms TTL = 127
Reply from 20.0.0.1: bytes = 32 time = 0ms TTL = 127
Reply from 20.0.0.1: bytes = 32 time = 0ms TTL = 127
Reply from 20.0.0.1: bytes = 32 time = 0ms TTL = 127

Ping statistics for 20.0.0.1

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

## PART-2




PC > ping 20.0.0.2 (from PC0)

Destination Host Unreachable.

Ping statistics for 20.0.0.2:

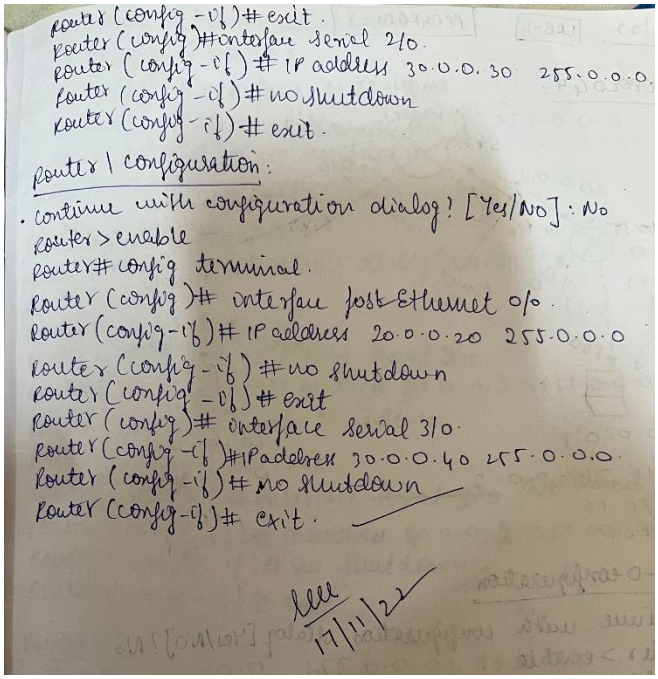
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

Ping 30.0.0.40 (from Router 0 to Router 1)

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

Router 0 configuration:

```
Continue with configuration dialog? [Yes/No]: No.
Router > enable
Router # config terminal.
Router (config)# interface fastEthernet 0/0.
Router (config-if)# IP address 10.0.0.10 255.0.0.0
Router (config-if)# no shutdown.
```



```
Router (config-if)# exit.
Router (config)# interface Serial 2/0.
Router (config-if)# IP address 30.0.0.30 255.0.0.0.
Router (config-if)# no shutdown
Router (config-if)# exit.
```

Router 1 configuration:

```
Continue with configuration dialog? [Yes/No]: No
Router > enable
Router# config terminal.
Router (config)# interface fastEthernet 0/0.
Router (config-if)# IP address 20.0.0.20 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit
Router (config)# interface Serial 3/0.
Router (config-if)# IP address 30.0.0.40 255.0.0.0.
Router (config-if)# no shutdown
Router (config-if)# exit.
```

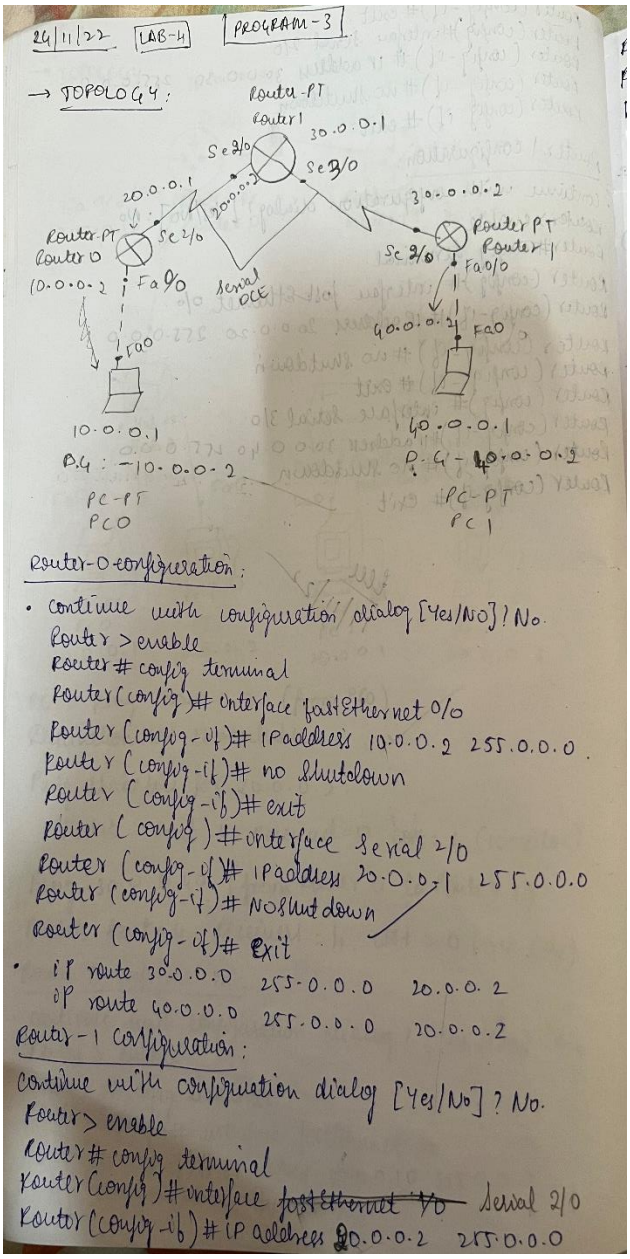
lll
17/11/22

# Experiment No-3

**Aim of the program-**Configuring default route to the Router.

**TOPOLOGY, PROCEDURE AND OUTPUT:**

**PART-1**



Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial 3/0
Router(config-if)# IP address 30.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit

Router-2 configuration:

continue with configuration dialog [yes/no]? no
Router> enable
Router# config terminal
Router(config)# interface serial 2/0
Router(config-if)# IP address 30.0.0.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface ~~serial 3/0~~ fastethernet 0/0
Router(config-if)# IP address 30.0.0.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit

Router 1:

Router> show ip router
C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, serial 2/0

Router> enable
Router# config terminal
Router(config)# ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router(config)# ip router 40.0.0.0 255.0.0.0 20.0.0.2
exit
Router> 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 2:

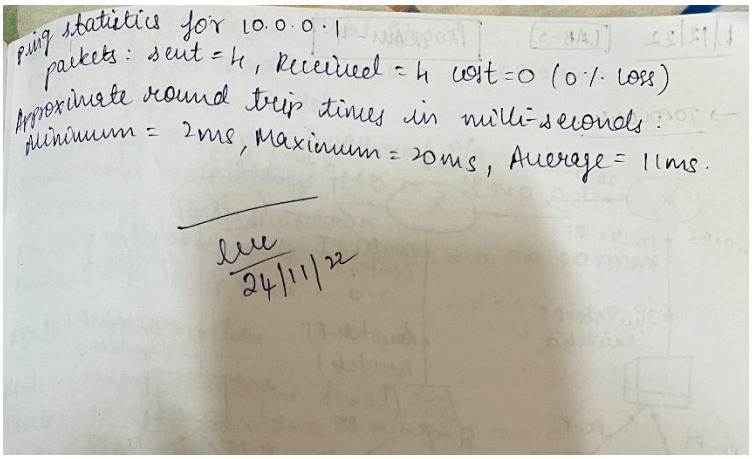
Router(config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router(config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2

Router 3:

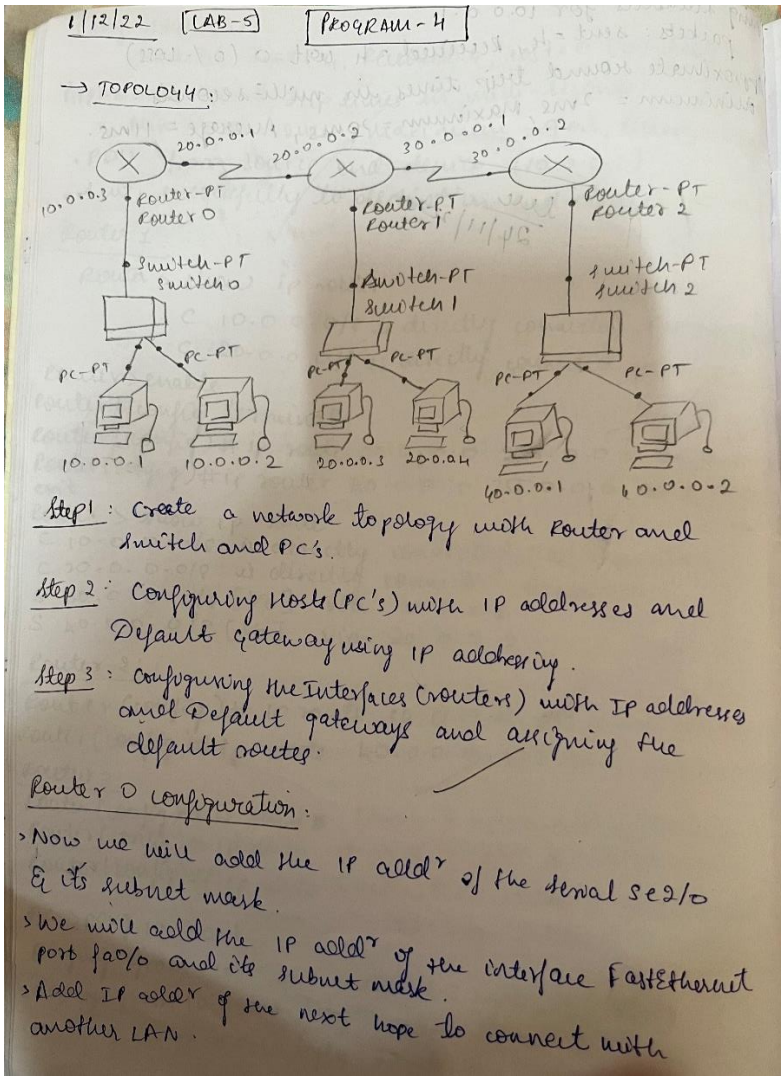
Router(config)# ip route 20.0.0.0 255.0.0.0 30.0.0.1
Router(config)# ip route 10.0.0.0 255.0.0.0 30.0.0.1
Router(config)# ip route Destination n/w subnet mask nethop address.

PC> pong 10.0.0.1

Pinging from 10.0.0.1 with 32 bytes of data
Reply from 10.0.0.1 : bytes = 32 time = 12ms TTL = 125
Reply from 10.0.0.1 : bytes = 32 time = 12ms TTL = 125
Reply from 10.0.0.1 : bytes = 32 time = 20ms TTL = 125


ping statistics for 10.0.0.1
packets: sent = 4, Received = 4 lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 20ms, Average = 11ms.
lee
24/11/22

**PART-2**


1/12/22 [LAB-5] [PROGRAM-4]
→ TOPOLOGY:
20.0.0.1
20.0.0.2
30.0.0.1
30.0.0.2
10.0.0.3
Router-PT Router 0
Router-PT Router 1
Router-PT Router 2
Switch-PT Switch 0
Switch-PT Switch 1
Switch-PT Switch 2
PC-PT
10.0.0.1
10.0.0.2
20.0.0.3
20.0.0.4
40.0.0.1
40.0.0.2
Step1: Create a network topology with Router and Switch and PC's.
Step 2: Configuring Hosts (PC's) with IP addresses and Default gateway using IP addressing.
Step 3: Configuring the Interfaces (routers) with IP addresses and Default gateways and assigning the default routes.
Router 0 configuration:
> Now we will add the IP addr of the serial se2/0 & its subnet mask.
> We will add the IP addr of the interface FastEthernet port fa0/0 and its subnet mask.
> Add IP addr of the next hope to connect with another LAN.

Router(config)#interface serial 2/0
Router(config)#ip address 20.0.0.1 255.0.0.0
Router(config)#No shutdown
Router(config)# interface fastethernet 0/0
Router(config)# ip address 10.0.0.3 255.0.0.0
Router(config-if)#No shutdown
Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2

Router 1 configuration.

Router(config)# interface serial 2/0
Router(config)# ip address 20.0.0.2 255.0.0.0
Router(config-if)# No shutdown
Router(config-if)# exit
Router(config)# interface fastethernet 0/0
Router(config)# ip address 30.0.0.1 255.0.0.0
Router(config-if)# No shutdown
Router(config-if)# exit
Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.1
Router(config)# ip route 0.0.0.0 0.0.0.0 30.0.0.2

Router 2 configuration:

Router(config)# interface serial 3/0
Router(config)# ip address 30.0.0.2 255.0.0.0
Router(config-if)# No shutdown
Router(config-if)# exit
Router(config)# interface fastethernet 1/0
Router(config)# ip address 40.0.0.3 255.0.0.0
Router(config)# exit
Router(config)#ip route 0.0.0.0 30.0.0.1
Router(config)# ip route 0.0.0.0 30.0.0.1

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=255
Reply from 10.0.0.1: bytes=32 time=3ms TTL=255
Reply from 10.0.0.1: bytes=32 time=6ms TTL=255
Reply from 10.0.0.1: bytes=32 time=1ms TTL=255

Packets
Sent=4, Received=4, Lost=0
minimum=2ms, maximum=6ms Average=3ms

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 40.0.0.1

Ping 40.0.0.1 with 32 bytes of data.
Reply from 40.0.0.1: bytes=32 time=1ms TTL=255
Reply from 40.0.0.1: bytes=32 time=6ms TTL=255
Reply from 40.0.0.1: bytes=32 time=2ms TTL=255
Reply from 40.0.0.1: bytes=32 time=2ms TTL=255

Ping Statistics

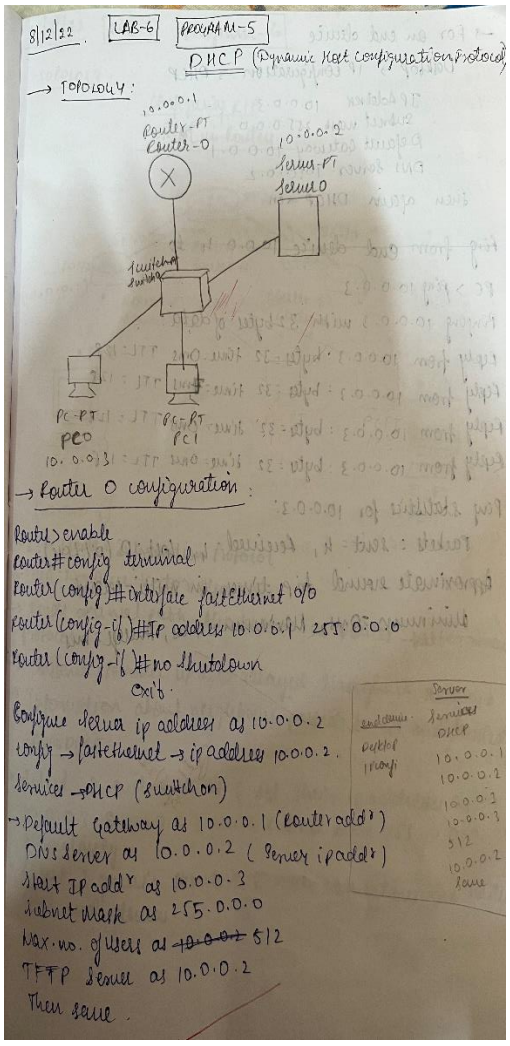
Packets sent=4, Received=4, Lost=0
Approximate round trip time in milliseconds.
minimum=2ms maximum=9ms Average=4ms

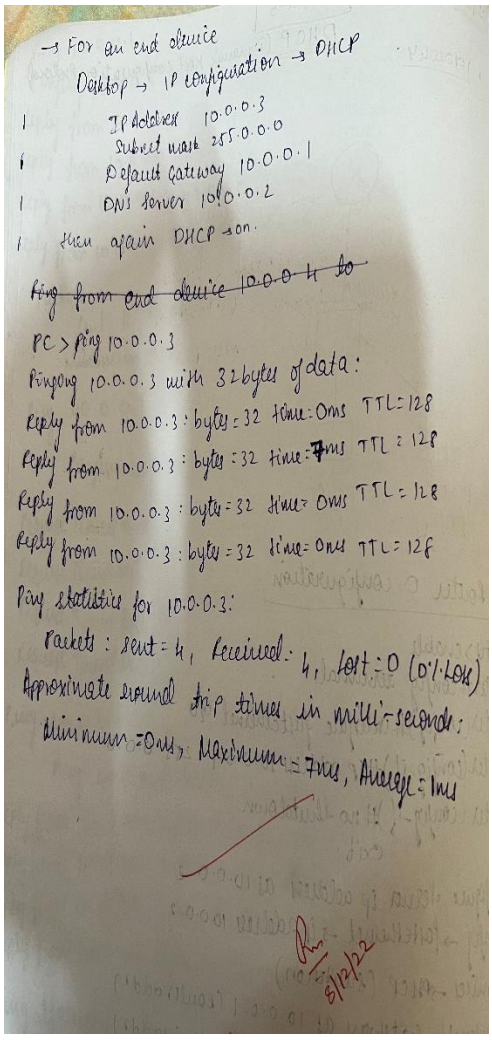
Lee 11/12/22

# Experiment No-4

**Aim of the program-**Configuring DHCP within a LAN in a packet Tracer.

**TOPOLOGY, PROCEDURE AND OUTPUT:**

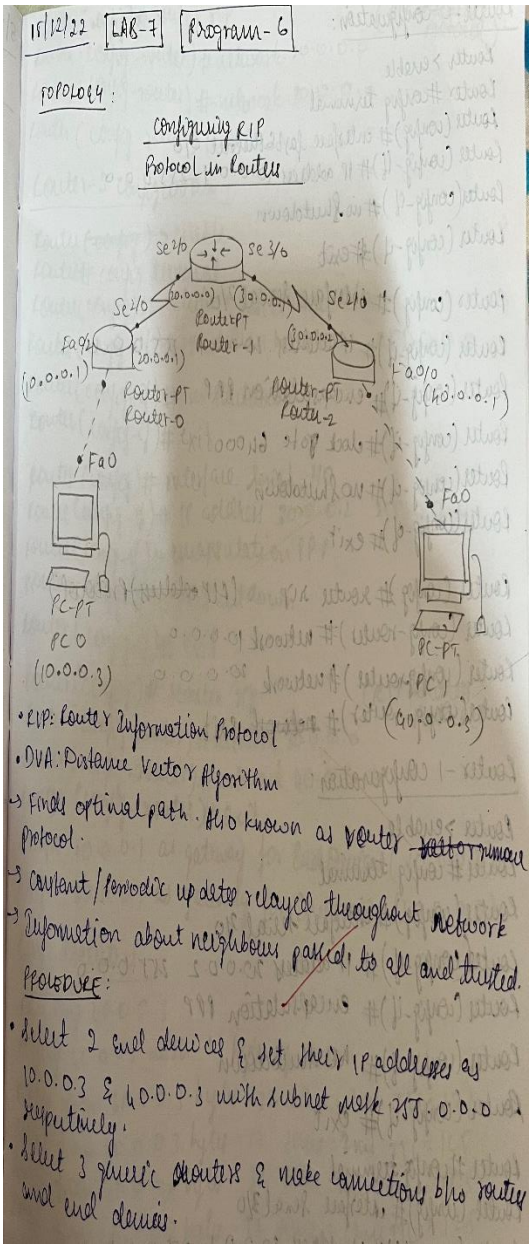


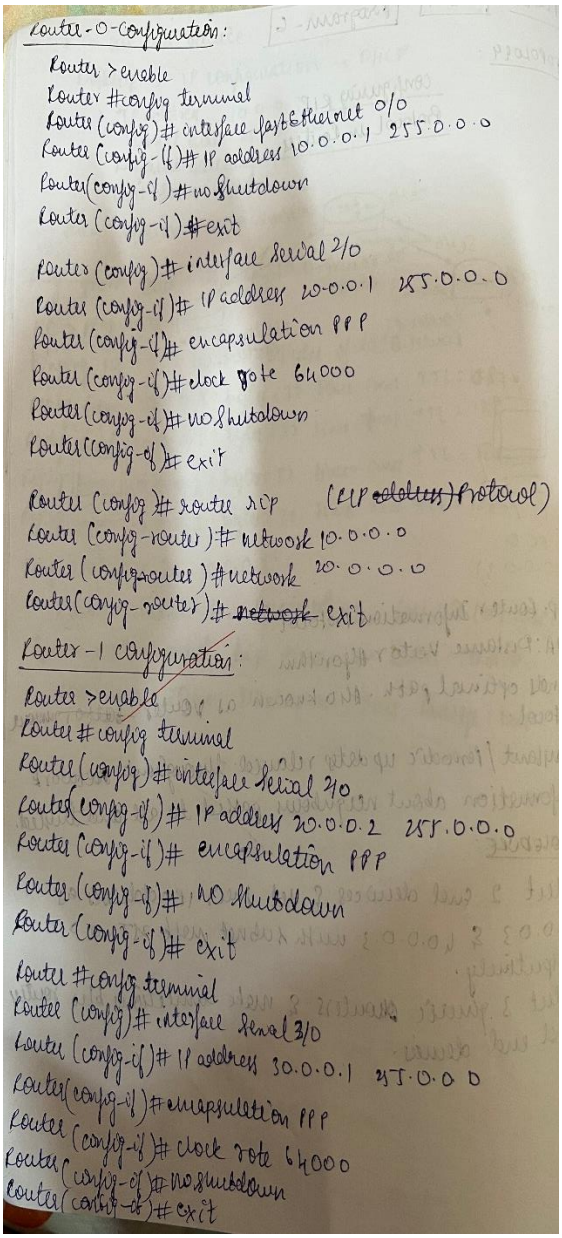



# Experiment No-5

**Aim of the program-** Configuring RIP Routing Protocol in Routers.

**TOPOLOGY, PROCEDURE AND OUTPUT:**



Router(config)# router rip (RIP ~~old test~~ Protocol)
Router(config-router)# network 20.0.0.0
Router(config-router)# network 30.0.0.0
Router(config-router)# exit

Router-2 configuration:

Router ~~(config)~~ > enable
Router# config terminal
Router(config)# interface fastEthernet 0/0
Router(config-if)# ip address 40.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial 2/0
Router(config-if)# ip address 30.0.0.2 255.0.0.0
Router(config-if)# encapsulation PPP
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# router rip (RIP ~~old test~~ Protocol)
Router(config-router)# network 30.0.0.0
Router(config-router)# network 40.0.0.0
Router(config-router)# exit.
Assign 10.0.0.1 as gateway for End Device (10.0.0.3)
Assign 40.0.0.1 as gateway for End Device (40.0.0.3)

Ping Statistics:

PC> Ping 40.0.0.3
Pinging 40.0.0.3 with 32 bytes of data:
Reply from 40.0.0.3 bytes = 32 time = 2ms TTL = 125
Reply from 40.0.0.3 bytes = 32 time = 2ms TTL = 125
Reply from 40.0.0.3 bytes = 32 time = 2ms TTL = 125
Reply from 40.0.0.3 bytes = 32 time = 12ms TTL = 125

Ping statistics for 40.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 12ms, Average = 4ms

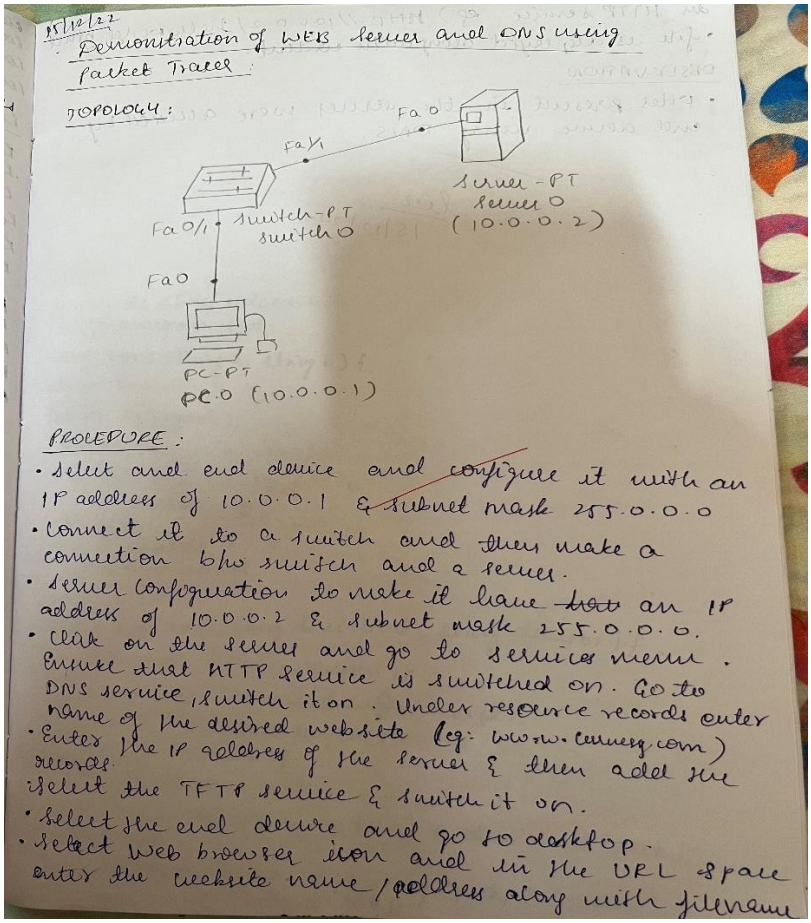
OBSERVATION:

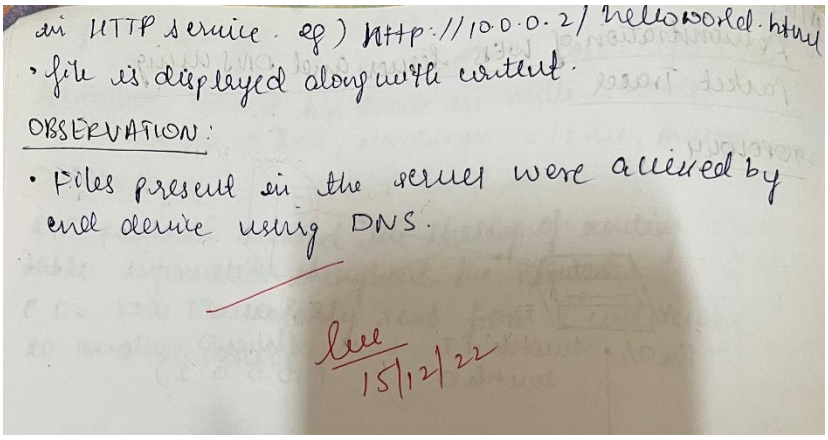
RIP protocol enabled the sharing of routing table information throughout the network.
PDU was successfully sent from 1 end device to another device.

# Experiment No-6

**Aim of the program-** Demonstration of WEB server and DNS using Packet Tracer.

**TOPOLOGY, PROCEDURE AND OUTPUT:**

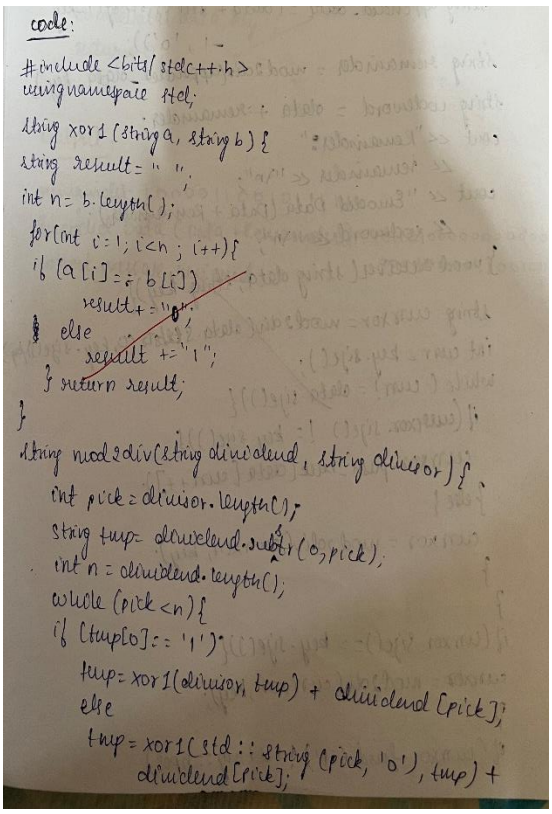


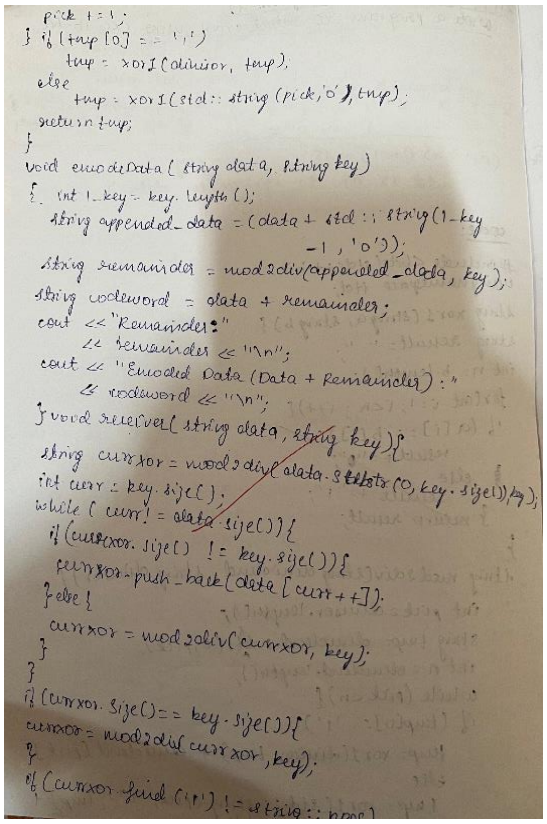
## Cycle-2

## Program-1

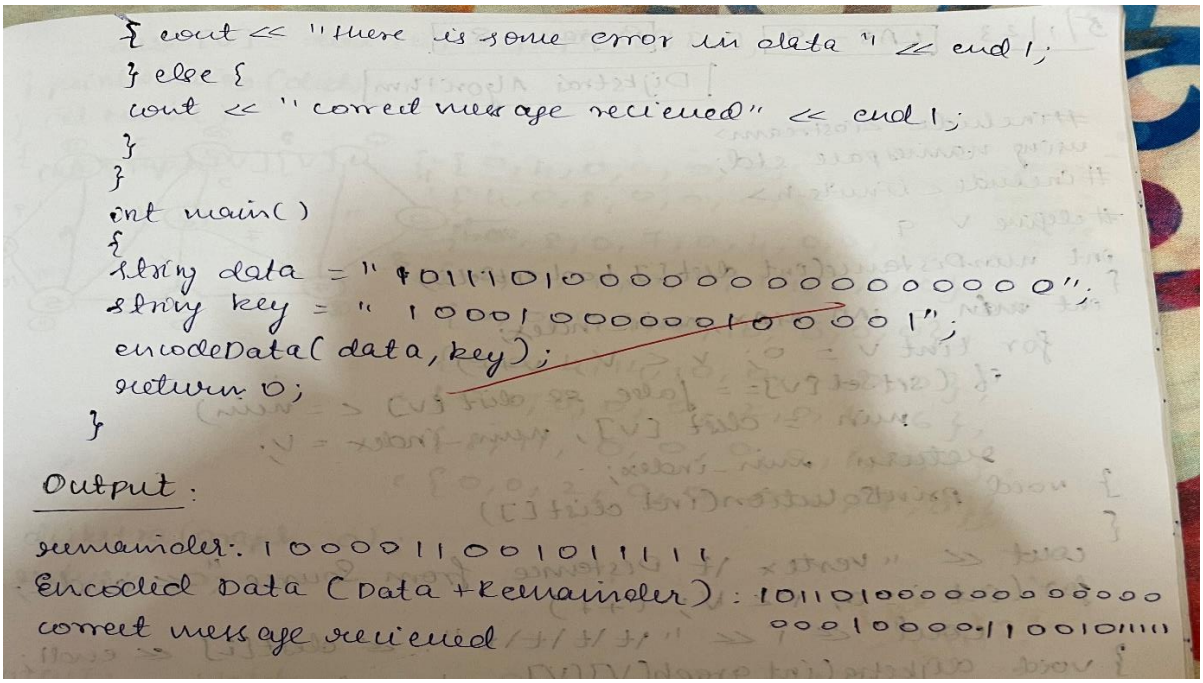
Write a program for error detecting code using CRC-CCITT (16-bits).



```
code:
#include <bits/stdc++.h>
using namespace std;
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++;
} 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";
} 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)
```



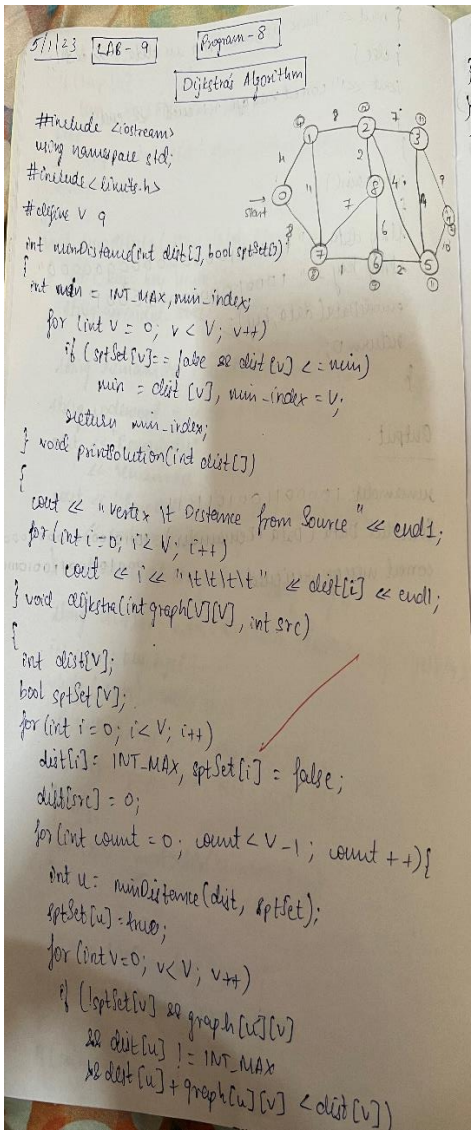
```
{ cout << "there is some error in data " << endl;
} else {
cout << "correct message recieved" << endl;
}
}
int main()
{
string data = "10111010000000000000000";
string key = "10001000000100001";
encodeData(data, key);
return 0;
}
```

Output:

remainder: 1000011001011111
Encoded Data (Data + Remainder): 10111010000000000000
correct message recieved 00010000-1100101111

## Program-2

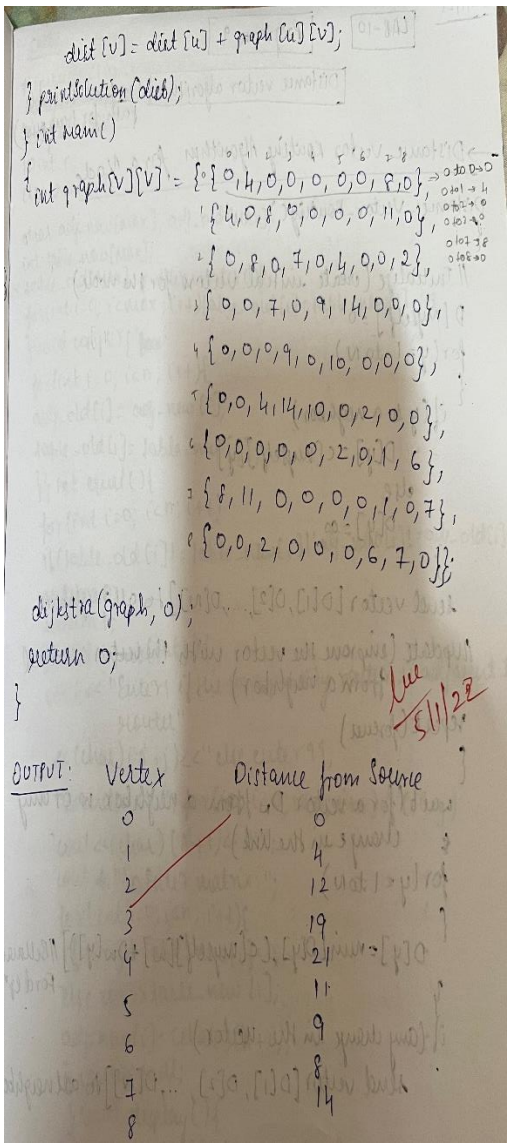
Implement Dijkstra's algorithm to compute the shortest path for a given topology.



5/11/23 LAB - 9 Program - 8

Dijkstra's Algorithm

```
#include <iostream>
using namespace std;
#include <limits.h>
#define V 9
int minDistance(int dist[], bool sptSet[])
{
 int min = INT_MAX, min_index;
  for (int v = 0; v < V; v++)
    if (sptSet[v] == false && dist[v] <= min)
        min = dist[v], min_index = v;
    return min_index;
} void printSolution(int dist[])
{
 cout << "Vertex \t Distance from Source" << endl;
 for (int i = 0; i< V; i++)
    cout << i << "\t\t\t" << dist[i] << endl;
} void dijkstra(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 = minDistance(dist, sptSet);
   sptSet[u] = true;
   for (int v=0; v< V; v++)
    if (!sptSet[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] = {{0,4,0,0,0,0,0,8,0},
                     {4,0,8,0,0,0,0,11,0},
                     {0,8,0,7,0,4,0,0,2},
                     {0,0,7,0,9,14,0,0,0},
                     {0,0,0,9,0,10,0,0,0},
                     {0,0,4,14,10,0,2,0,0},
                     {0,0,0,0,0,2,0,1,6},
                     {8,11,0,0,0,0,1,0,7},
                     {0,0,2,0,0,0,6,7,0}};
 dijkstra(graph, 0);
 return 0;
}
```

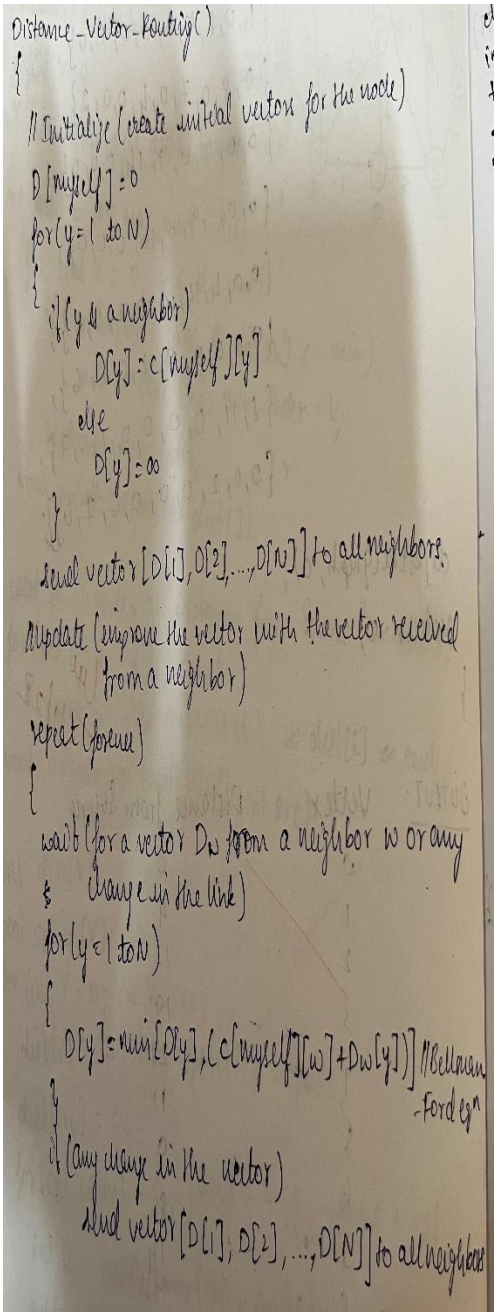
5/11/23

OUTPUT:

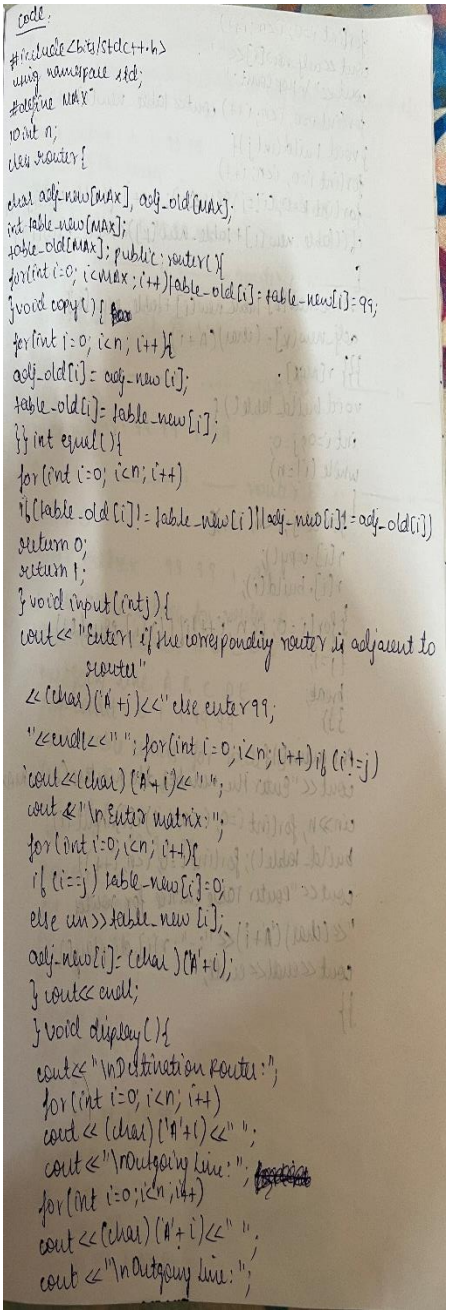
| Vertex | Distance from Source |
|---|---|
| 0 | 0 |
| 1 | 4 |
| 2 | 12 |
| 3 | 19 |
| 4 | 21 |
| 5 | 11 |
| 6 | 9 |
| 7 | 8 |
| 8 | 14 |

## Program-3

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



```
Distance_Vector_Routing()
{
// Initialize (create initial vectors for the node)
D[myself] = 0
for(y = 1 to N)
{
 if(y is a neighbor)
     D[y] = c[myself][y]
  else
     D[y] = ∞
 }
send vector [D[1], D[2], ..., D[N]] to all neighbors

Update (improve the vector with the vector received
        from a neighbor)
repeat (forever)
{
 wait (for a vector Dw from a neighbor w or any
 &  change in the link)
 for(y = 1 to N)
 {
   D[y] = min[D[y], (c[myself][w] + Dw[y])] //Bellman
                                              -Ford eqn
 }
 if (any change in the vector)
    send vector [D[1], D[2], ..., D[N]] to all neighbors
```



```
code:
#include<bits/stdc++.h>
using namespace std;
#define MAX
int n;
class router{
char adj_new[MAX], adj_old[MAX];
int table_new[MAX];
table_old[MAX]; public: router(){
for(int i=0; i<MAX; i++)table_old[i] = table_new[i] = 99;
}void copy(){
for(int i=0; i<n; i++){
adj_old[i] = adj_new[i];
table_old[i] = table_new[i];
}} int equal(){
for(int i=0; i<n; i++)
if(table_old[i] != table_new[i] || adj_new[i] != adj_old[i])
return 0;
return 1;
}void input(int j){
cout<<"Enter 1 if the corresponding router is adjacent to
        router"
<<(char)('A'+j)<<" else enter 99;
"<<endl<<" "; for(int i=0; i<n; i++) if(i!=j)
cout<<(char)('A'+i)<<" ";
cout<<"\n Enter metrix: ";
for(int i=0; i<n; i++){
 if(i==j) table_new[i]=0;
 else cin>>table_new[i];
 adj_new[i] = (char)('A'+i);
} cout<<endl;
}void display(){
 cout<<"\n Destination router:";
 for(int i=0; i<n; i++)
 cout<<(char)('A'+i)<<" ";
 cout<<"\n Outgoing Line: ";
 for(int i=0; i<n; i++)
 cout<<(char)('A'+i)<<" ";
 cout<<"\n Outgoing Line: ";
```

```
for(int i=0; i<n ;i++)
cout<<adj_new[i]<<" ";
cout<<"\nHop count : ";
for(int i=0; i<n; i++) cout<< table_new[i]<<" ";
}void build(int j){
for(int i=0; i<n; i++)
for(int k=0; (i!=j)&&(k<n); k++){if(table_old[i]==99)
if((table_new[i]+table_new[k])<table_new[k])
{
 table_new[k]=table_new[i]+table_new[k];
 adj_new[k]=(char)('A'+i);
}}} r[MAX];
void build_table(){
int i=0,j=0;
while(i!=n)
{
 for(i=j;i<n;i++){
  r[i].copy();
  r[i].build(i);
 } for(i=0; i<n; i++) if(!r[i].equal())
 { j=i;
 break;
 }}
int main(){
cout<<"Enter the number the routers (<="<<MAX<<")";
cin>>n; for(int i=0; i<n;i++) r[i].input(i);
build_table(); for(int i=0; i<n; i++){
cout<<"Router Table entries for router
"<<(char)('A'+i)<<":-"; r[i].display();
cout<<endl<<endl;
}}
```

```
OUTPUT:
Enter the number the routers (<=10): 5
Enter 1 if the corresponding router is adjacent to router A else enter 99:
  B C D E
Enter matrix : 1  1  99 99
Enter 1 of the corres  ''  ____          router B  ''  ____  ''  ____
  A C D E
Enter matrix : 1 99 99 99
Enter 1  ''  ____  ''  ____          router C  ''  ____  ''  ____
  A B D E
Enter matrix : 1  99  1  1
Enter 1  ''  ____  ''  ____          router D  ''  ____  ''  ____
  A B C E
Enter matrix : 99 99  1 99
Enter 1  ''  ____  ''  ____          router E  ''  ____  ''  ____
  A B C D
Enter matrix :  99  99  1  99
Router Table entries for router A :-
Destination ROUTER: A B C D E
Outgoing line: A B C D E
Hop count : 0 1 1 99 99
Router Table entries for router B :-
DR : A B C D E
OL : A B C D E
HC : 1 0 99 99 99
Router C :-
DR : A B C D E
OL : A B C D E
HC : 1 99 0 1 1
Router D :-
DR : A B C D E
OL : A B C D E
HC : 99 99 1 0 99
Router E :-
DR : A B C D E
OL : A B C D E
HC : 99 99 1 99 0
```

## Program-5

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.

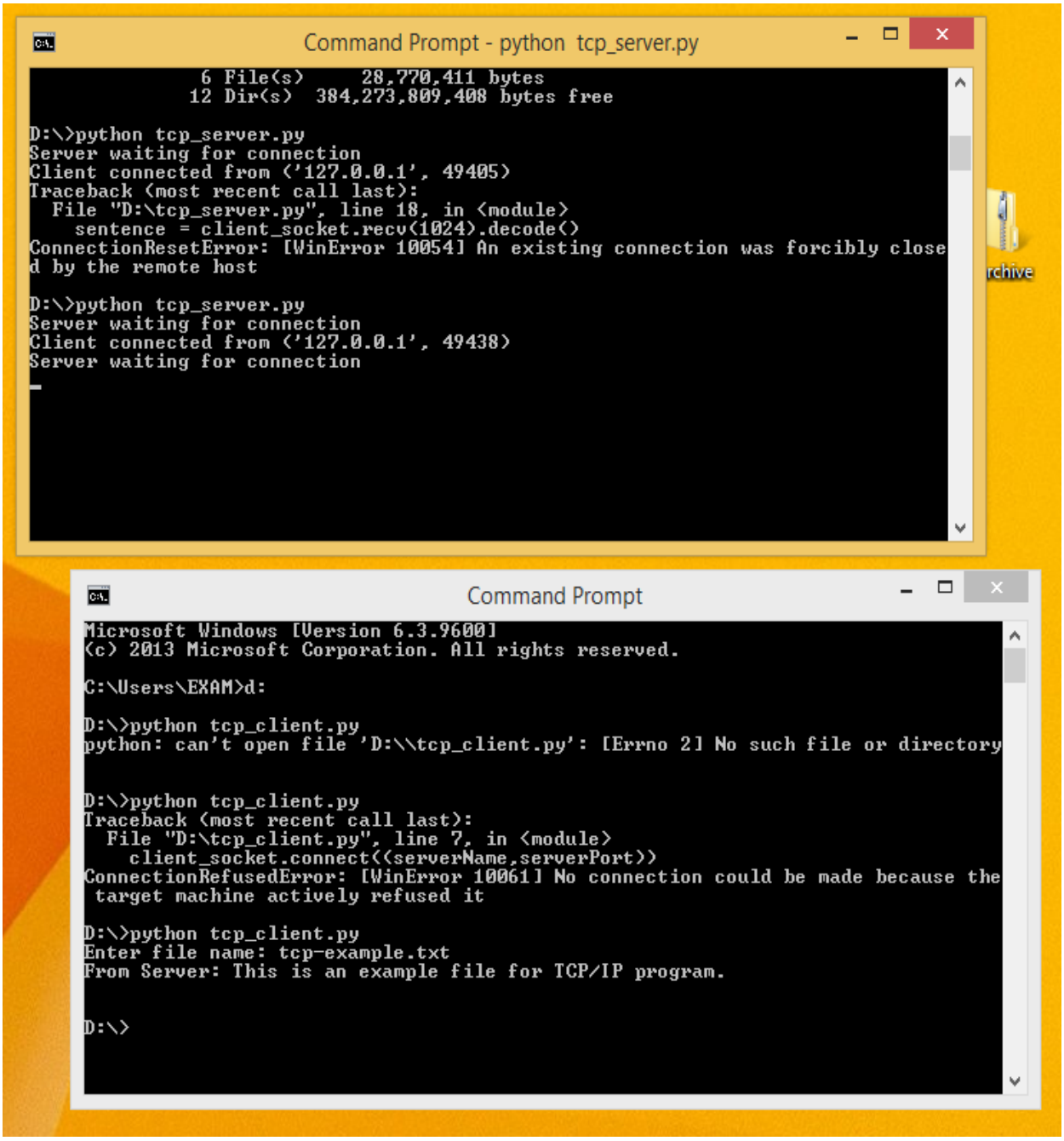
**CODE:**

**SERVER**

```
import socket

serverName = '127.0.0.1'
serverPort = 12345
#create
server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

#bind
server_socket.bind((serverName, serverPort ))

#listen
server_socket.listen(5)

while True:
    print("Server waiting for connection")
    client_socket, addr = server_socket.accept()
    print("Client connected from",addr)
    sentence = client_socket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)

    client_socket.send(l.encode())
    file.close()
    client_socket.close()
```

**CLIENT**

```
import socket

serverName = '127.0.0.1'
serverPort = 12345

client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client_socket.connect((serverName,serverPort))
sentence = input("Enter file name: ")

client_socket.send(sentence.encode())
filecontents = client_socket.recv(1024).decode()
print ('From Server:', filecontents)
client_socket.close()
```


Command Prompt - python tcp_server.py
6 File(s) 28,770,411 bytes
12 Dir(s) 384,273,809,408 bytes free
D:\>python tcp_server.py
Server waiting for connection
Client connected from ('127.0.0.1', 49405)
Traceback (most recent call last):
File "D:\tcp_server.py", line 18, in <module>
sentence = client_socket.recv(1024).decode()
ConnectionResetError: [WinError 10054] An existing connection was forcibly close
d by the remote host
D:\>python tcp_server.py
Server waiting for connection
Client connected from ('127.0.0.1', 49438)
Server waiting for connection
rchive
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\EXAM>d:
D:\>python tcp_client.py
python: can't open file 'D:\\tcp_client.py': [Errno 2] No such file or directory
D:\>python tcp_client.py
Traceback (most recent call last):
File "D:\tcp_client.py", line 7, in <module>
client_socket.connect((serverName,serverPort))
ConnectionRefusedError: [WinError 10061] No connection could be made because the
target machine actively refused it
D:\>python tcp_client.py
Enter file name: tcp-example.txt
From Server: This is an example file for TCP/IP program.
D:\>

## Program-6

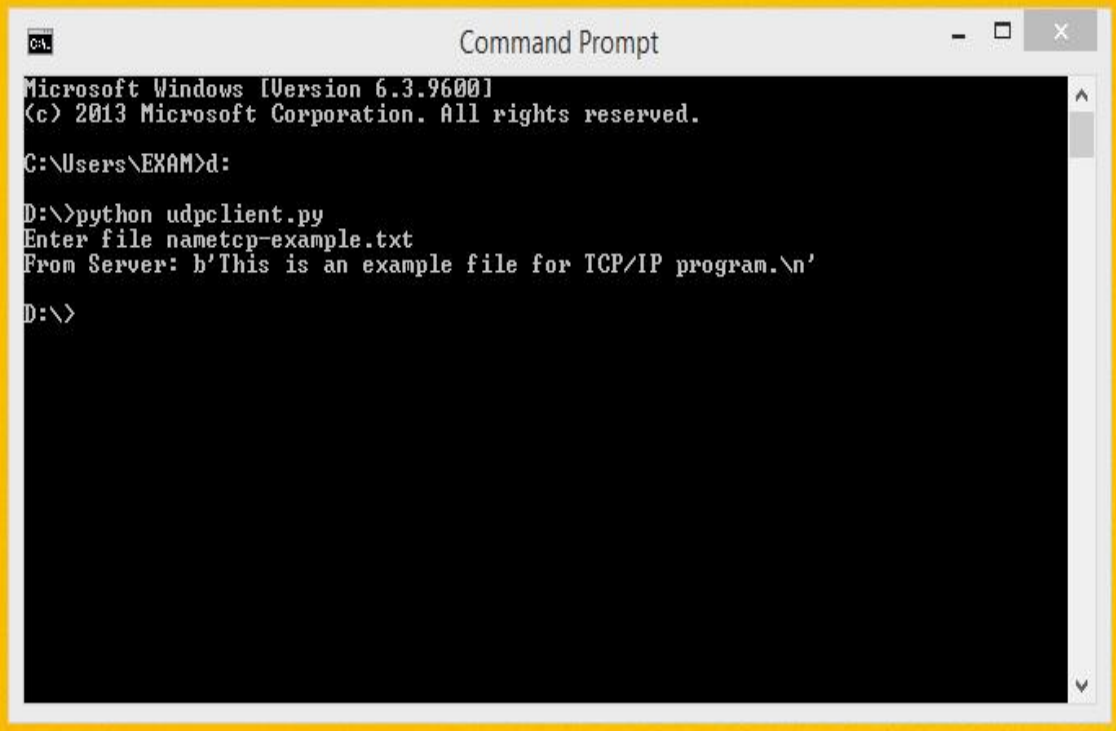
Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

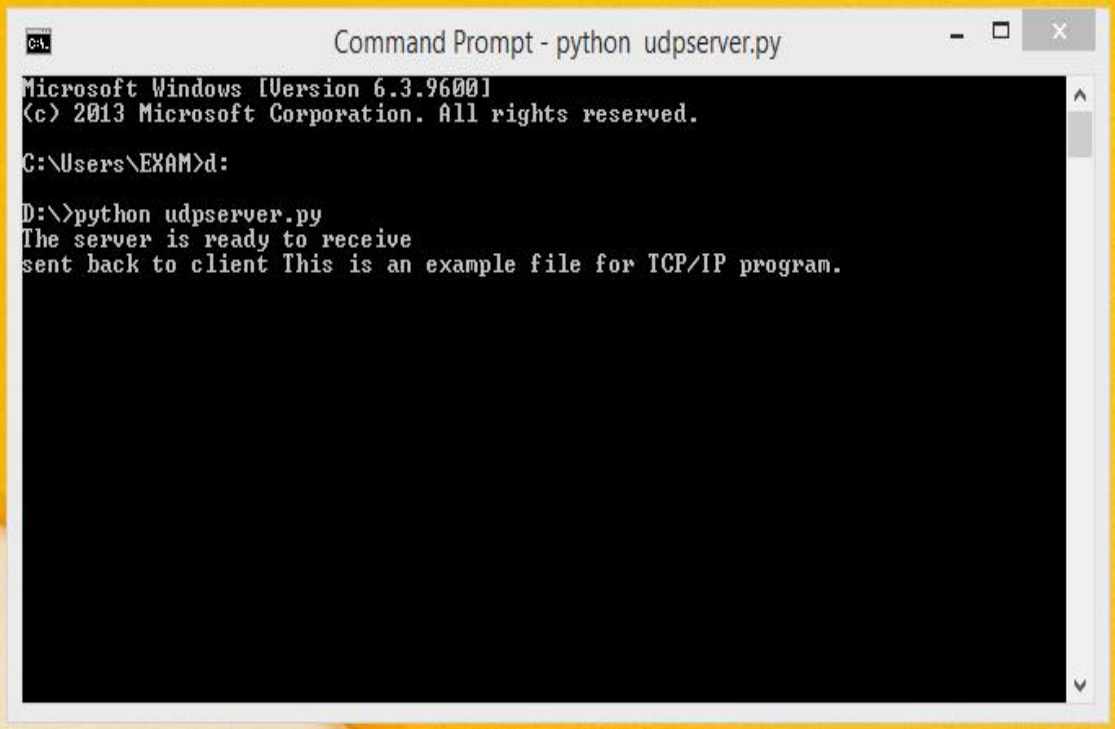
**SERVER**

```
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()
```


Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\EXAM>d:
D:\>python udpclient.py
Enter file nametcp-example.txt
From Server: b'This is an example file for TCP/IP program.\n'
D:\>


Command Prompt - python udpserver.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\EXAM>d:
D:\>python udpserver.py
The server is ready to receive
sent back to client This is an example file for TCP/IP program.