**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB REPORT on**

# COMPUTER NETWORKS LAB

***Submitted by***

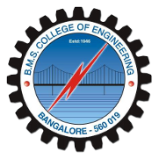
**GYANESH YADAV(1BM20CS202)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**October-2022 to Feb-2023**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

## **Department of Computer Science and Engineering**



### CERTIFICATE

This is to certify that the Lab work entitled “LAB COURSE **“COMPUTER NETWORKS” is carried** out by **GYANESH YADAV(1BM20CS202),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial ulfilment 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. SHYAMALA G** | **Dr. Jyothi S Nayak** |
| Assistant Professor | Professor and Head |
| Department of CSE | Department of CSE |
| BMSCE, Bengaluru | BMSCE, Bengaluru |
|  |  |

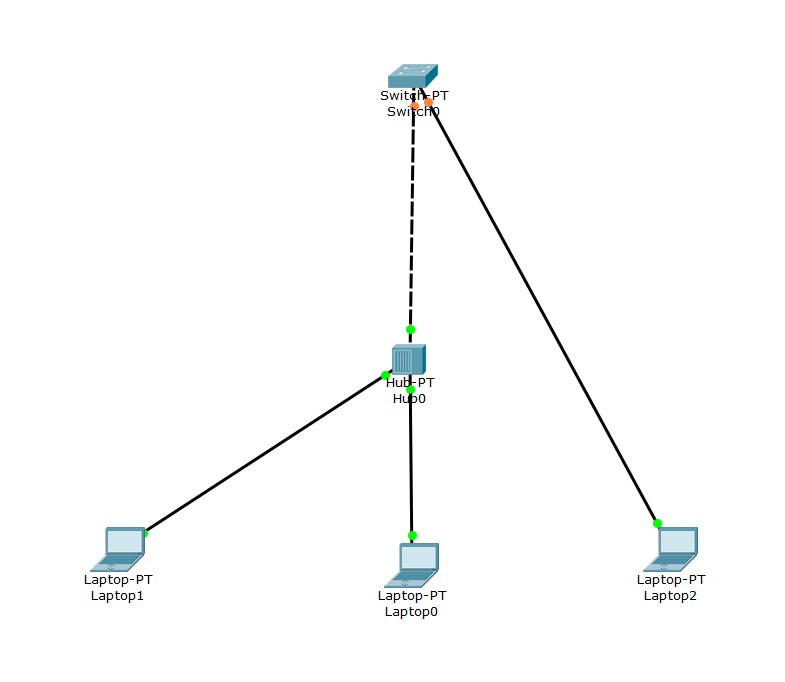
**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **DATE** | **EXPERIMENT** | **Page No.** |
|  |  | **CYCLE 1** | **4-9** |
| **1** | **7/11/22** | **Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.** | **4** |
| **2** | **14/11** | **Configuring IP address to Routers in Packet Tracer.**  **Explore the following messages: Ping Responses,**  **Destination unreachable, Request timed out, Reply** | **5** |
| **3** | **28/11** | **Configuring default route to the Router** | **6** |
| **4** | **12/12** | **Configuring DHCP within a LAN in a packet Tracer** | **7** |
| **5** | **5/12** | **Configuring RIP Routing Protocol in Routers** | **8** |
| **6** | **12/12** | **Demonstration of WEB server and DNS using Packet Tracer** | **9** |
| **II** |  | **CYCLE 2** | **10-27** |
| **1** | **17/12** | **Write a program for error detecting code using CRC-CCITT (16-bits).** | **10-11** |
| **2** | **9/01/23** | **Write a program for distance vector algorithm to find suitable path for transmission.** | **12-16** |
| **3** | **2/1/23** | **Implement Dijkstra’s algorithm to compute the shortest path for a given topology.** | **17-19** |
| **4** | **16/1/23** | **Write a program for congestion control using Leaky bucket algorithm.** | **20-23** |
| **5** | **23/1/23** | **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.** | **24-25** |
| **6** | **23/01/23** | **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.** | **26-27** |

## **Cycle-1** Program 1

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

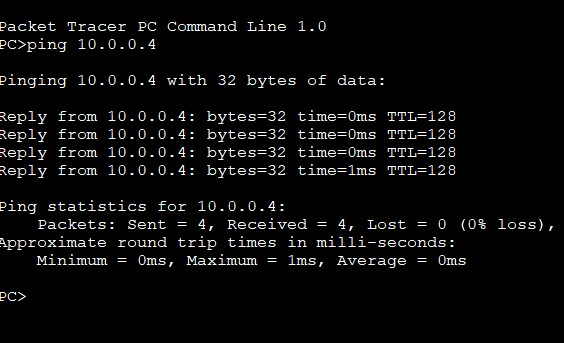
### Topology



### Procedure

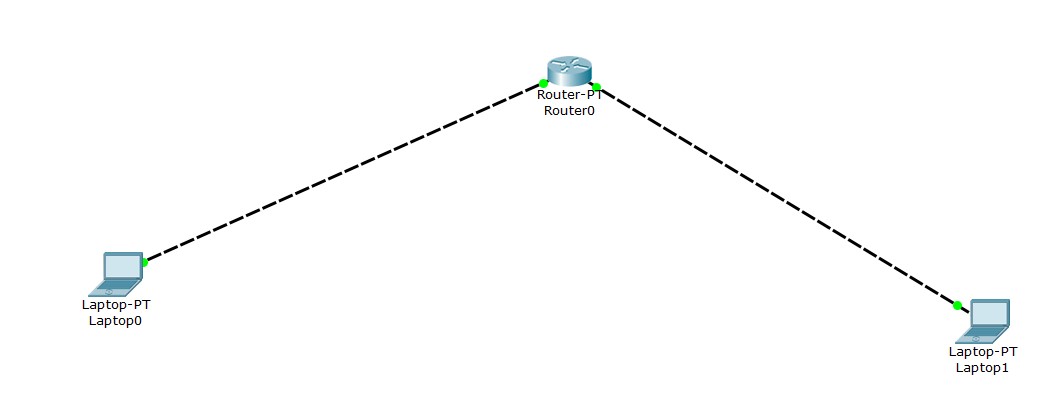
1. We have taken a switch and linked it to four end devices.
2. Link every device with the switch.
3. Provide the IP address to each device.
4. Transfer messages from one device to another and check the Table for Validation.

### Output Screenshot



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

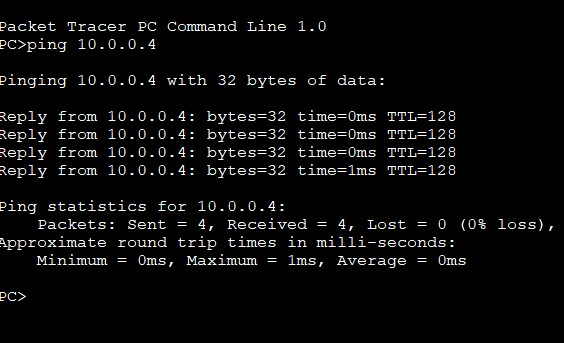
### Topology



### Procedure

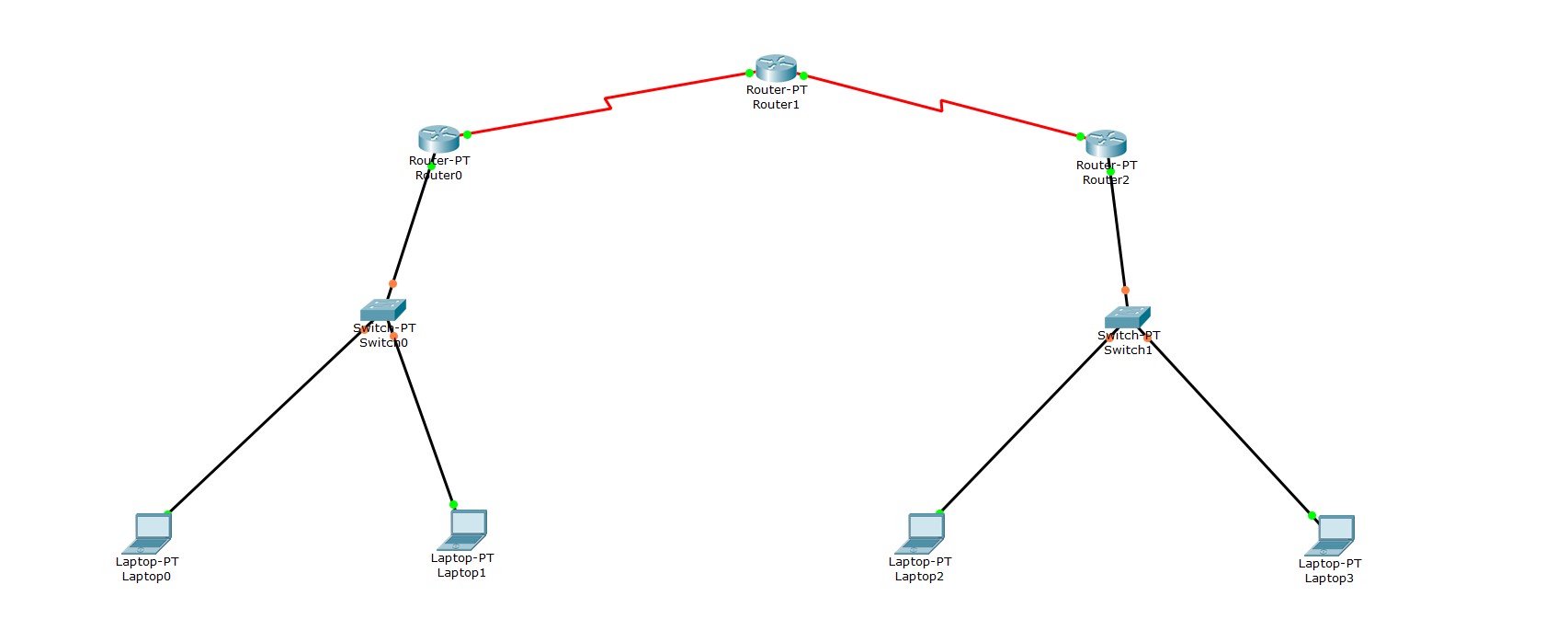
1. Select the router and Open CLI.
2. Press ENTER to start configuring Router1.
3. Type enable to activate the privileged mode.
4. Type config t(configure terminal) to access the configuration menu.
5. Configure interfaces of Router1:

### Output screenshot



**Aim of the program : Configuring default route to the Router**

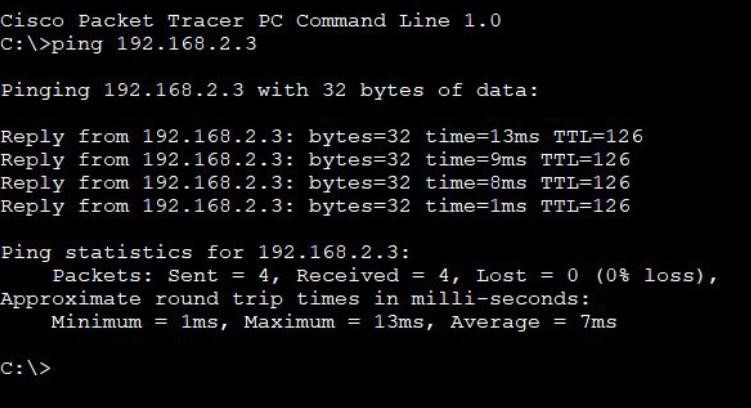
### Topology



### Procedure

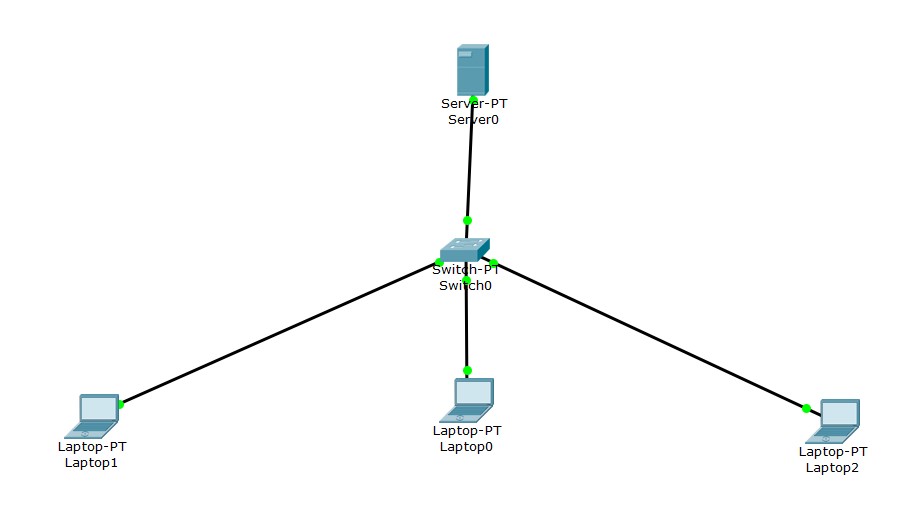
1. First, create a network topology of these given devices listed below in the table.
2. Configuring Hosts (PCs) with IP addresses and Default Gateway using IP Addressing table given below.
3. Configuring the Interfaces (routers) with IP Addresses and Default gateways and assigning the default routes.
4. After configuring all the devices, the red indicator turns into green and the network is live so we can send and receive packets.

### Output Screenshot



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

### Topology



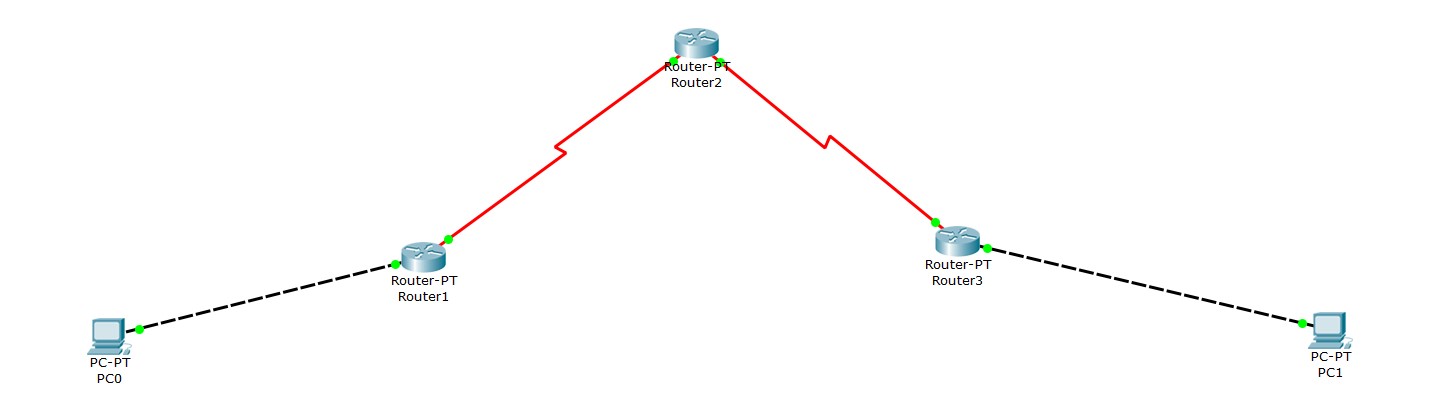
### Procedure

1. First, open the cisco packet tracer desktop and select the devices given below
2. Configure the Server with IPv4 address and Subnet Mask according to the Data given above.
3. Configuring the DHCP server.
4. Configuring Router with IPv4 Address and Subnet Mask.
5. Configuring the PCs and changing the IP configuration.

**Output Screenshot**

**Aim of the program : Configuring RIP Routing Protocol in Routers**

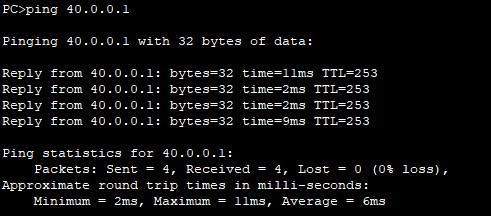
**Topology**



**Procedure**

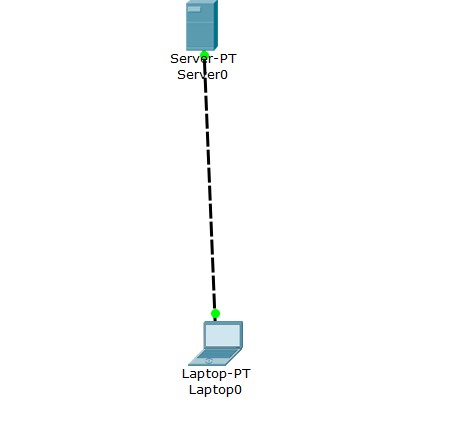
1. Configure the network interfaces. This example shows multiple loopback interface addresses to simulate attached networks. ...
2. Create the RIP group and add the interface. ...
3. Create the routing policy to advertise both direct and RIP-learned routes. ...
4. Apply the routing policy.

**Output Screenshot**



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

### Topology



### Procedure

1. Build the network topology.
2. Configure static IP addresses on the PCs and the server. Server. ...
3. Configure DNS service on the generic server. To do this, click on the server, then Click on Services tab. ...
4. Test domain name – IP resolution.

### Output screenshot



## **Cycle-2** Program 1

### Aim: Write a program for error detecting code using CRC-CCITT (16-bits)

**Code:**

def xor(a, b): result = [] for i in range(1, len(b)): if a[i] == b[i]:

result.append('0') else:

result.append('1') return ''.join(result) def mod2div(dividend, divisor):

pick = len(divisor) tmp = dividend[0 : pick] while pick < len(dividend):

if tmp[0] == '1':

tmp = xor(divisor, tmp) + dividend[pick] else:

tmp = xor('0'\*pick, tmp) + dividend[pick] pick += 1 if tmp[0] == '1':

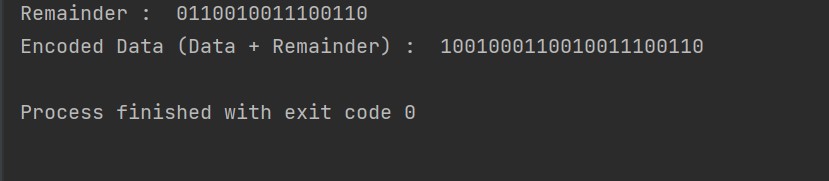
tmp = xor(divisor, tmp) else:

tmp = xor('0'\*pick, tmp) checkword = tmp

return checkword def encodeData(data, key):

l\_key = len(key) appended\_data = data + '0'\*(l\_key-1) remainder = mod2div(appended\_data, key) codeword = data + remainder print("Remainder : ", remainder) print("Encoded Data (Data + Remainder) : ", codeword) data = "100100" key = "10001000000100001" encodeData(data, key)

### Output Screenshot



## Program 2

**Aim: To write a program for a distance vector algorithm to find a suitable path for transmission.**

**Code:**

#include<stdio.h> struct node

{

unsigned dist[20]; unsigned from[20];

}rt[10]; int main()

{

int costmat[20][20]; int nodes,i,j,k,count=0; printf("\nEnter the number of nodes : "); scanf("%d",&nodes);//Enter the nodes

printf("\nEnter the cost matrix :\n"); for(i=0;i<nodes;i++)

{

for(j=0;j<nodes;j++)

{

scanf("%d",&costmat[i][j]); costmat[i][i]=0; rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix

rt[i].from[j]=j;

}

} do

{ count=0;

for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the

direct distance from the node i to k using the cost matrix //and add the distance from k to node j for(j=0;j<nodes;j++) for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{//We calculate the minimum distance

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]

; rt[i].from[j]=k;

count++;

}

}while(count!=0); for(i=0;i<nodes;i++)

{

printf("\n\n For router %d\n",i+1); for(j=0;j<nodes;j++)

{

printf("\t\nnode %d via %d Distance %d

",j+1,rt[i].from[j]+1,rt[i].dist[j]);

}

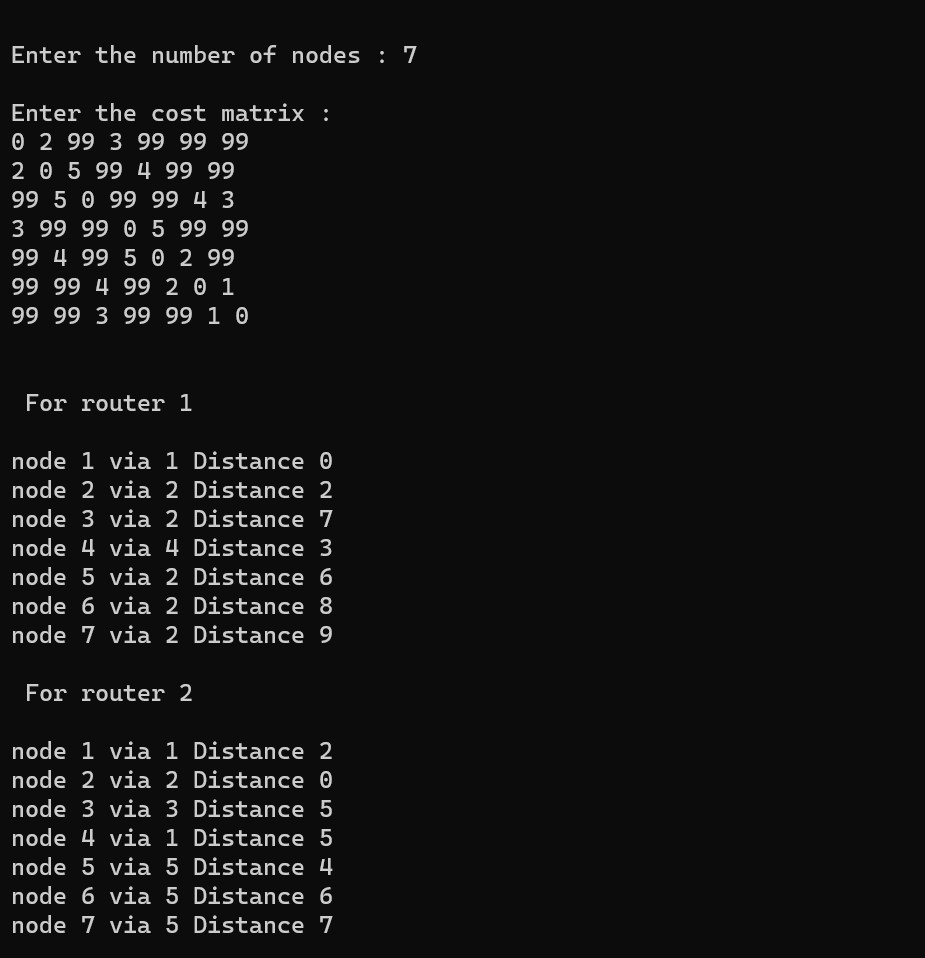
}

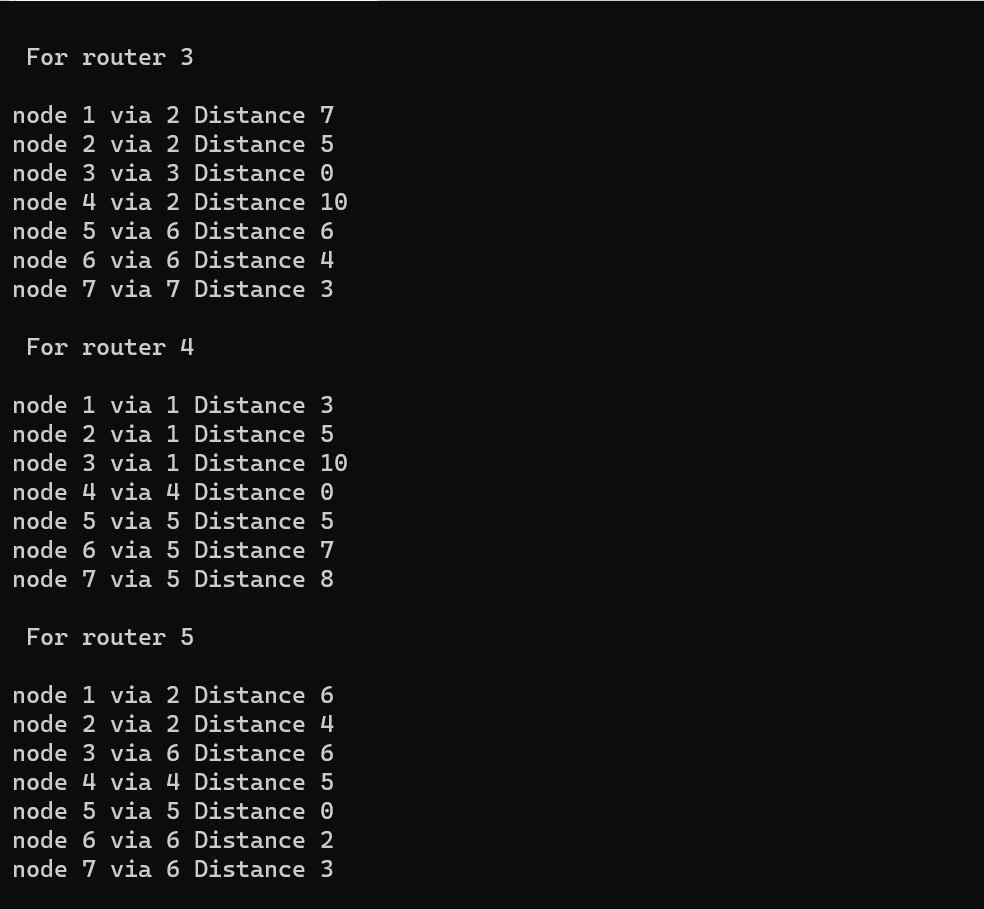
printf("\n\n");

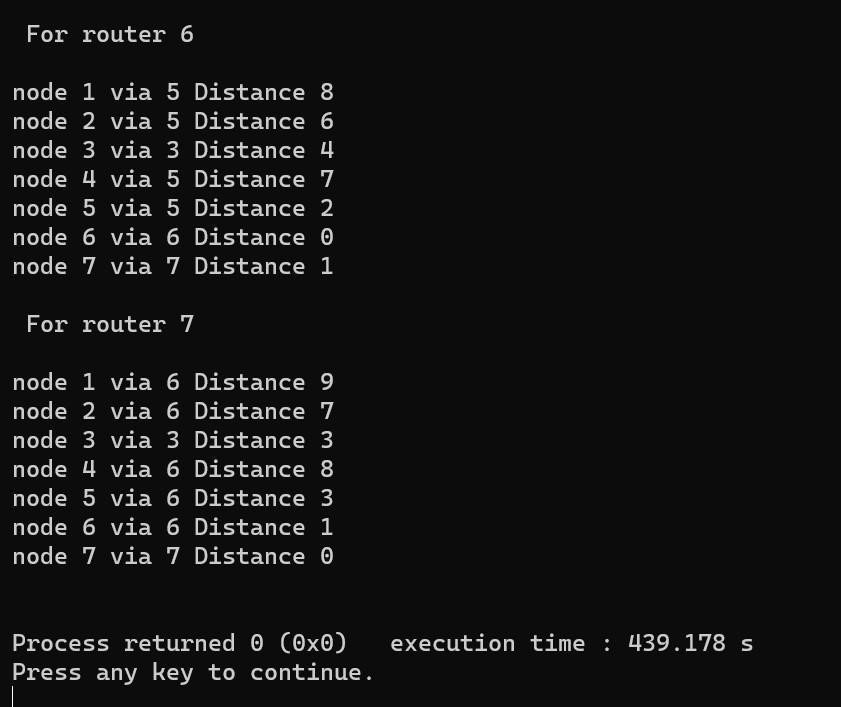
//getch();

}

### Output Screenshot







## Program 3

**Aim: To implement Dijkstra’s algorithm to compute the shortest path for a given topology.**

**Code:**

#include<stdio.h>

void dijkstras(); int c[10][10], n, src;

void main() {

int i,j;

printf("\nEnter the num of vertices: \t");

scanf("%d", &n);

printf("\nEnter the cost matrix: \n"); for(i = 1; i <= n; i++) {

for(j = 1; j <= n; j++) { scanf("%d", &c[i][j]);

}

}

printf("\nEnter the source node: \t"); scanf("%d", &src);

dijkstras();

}

void dijkstras() { int vis[10], dist[10], u, j, count, min;

for(j = 1; j <= n; j++) {

dist[j] = c[src][j];

}

for(j = 1; j <= n; j++) { vis[j] = 0;

}

dist[src] = 0; vis[src] = 1; count = 1; while(count != n) { min = 9999;

for(j = 1; j <= n; j++) { if(dist[j] < min && vis[j] != 1) { min = dist[j]; u = j;

}

}

vis[u] = 1;

count++;

for(j = 1; j <= n; j++) { if(min + c[u][j] < dist[j] && vis[j] != 1) { dist[j] = min + c[u][j];

}

}

}

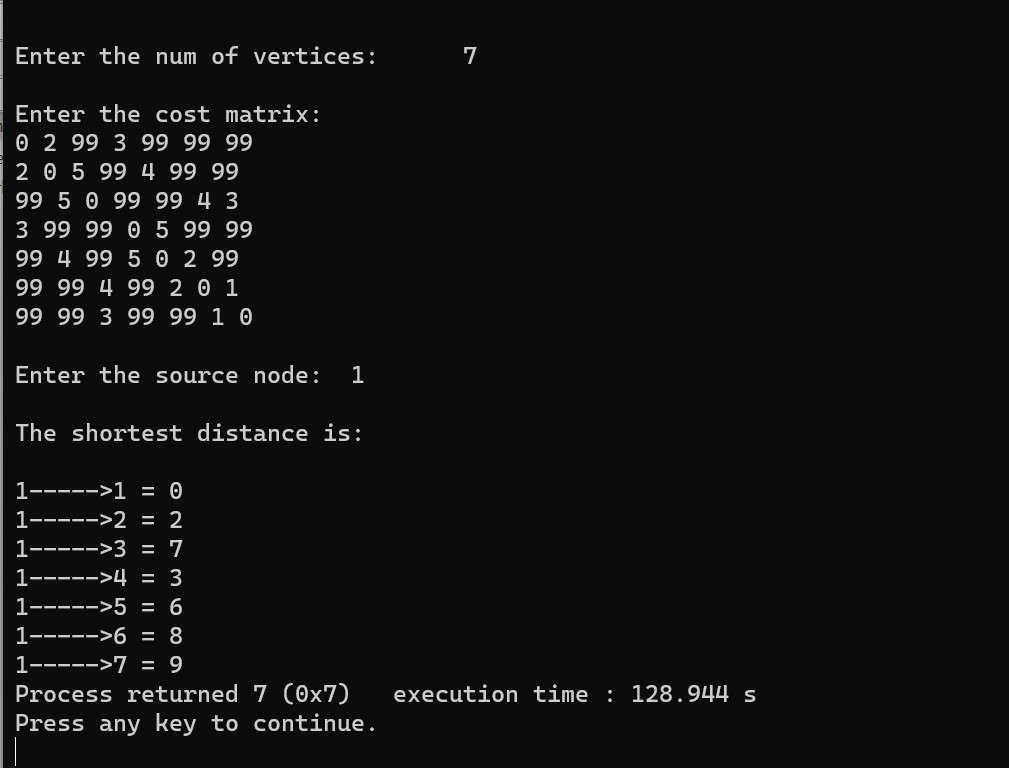
printf("\nThe shortest distance is: \n"); for(j = 1; j <= n; j++) {

printf("\n%d >%d = %d", src, j, dist[j]);

}

}

### Output Screenshot



## Program 4

**Aim: To write a program for congestion control using Leaky bucket algorithm.**

**Code:**

#include<stdio.h>

#include<stdlib.h> #include<conio.h> void main() {

int i,packets[10],content=0,newcontent,time,clk,bcktsize,oprate; for(i=0;i<5;i++)

{ packets[i]=rand()%10;

if(packets[i]==0) --i;

}

printf("\n Enter output rate of the bucket: \n");

scanf("%d",&oprate); printf("\n Enter Bucketsize\n"); scanf("%d",&bcktsize); for(i=0;i<5;++i)

{

if((packets[i]+content)>bcktsize)

{

if(packets[i]>bcktsize)

printf("\n Incoming packet size %d greater than the size of the bucket\n",packets[i]);

else printf("\n bucket size exceeded\n");

}

else {

newcontent=packets[i]; content+=newcontent; printf("\n Incoming Packet : %d\n",newcontent); printf("\n Transmission left : %d\n",content);

time=rand()%10;

printf("\n Next packet will come at %d\n",time); for(clk=0;clk<time && content>0;++clk)

{

printf("\n Left time %d",(time-clk)); if(content)

{

printf("\n Transmitted\n"); if(content<oprate) content=0; else content=content-oprate; printf("\n Bytes remaining : %d\n",content);

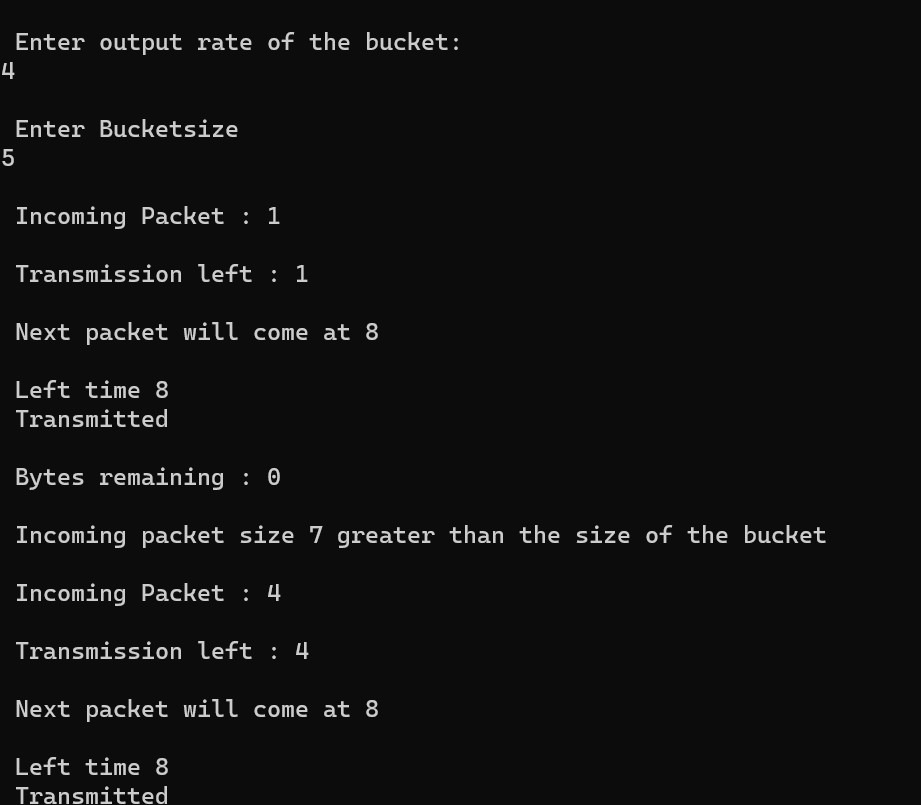
}

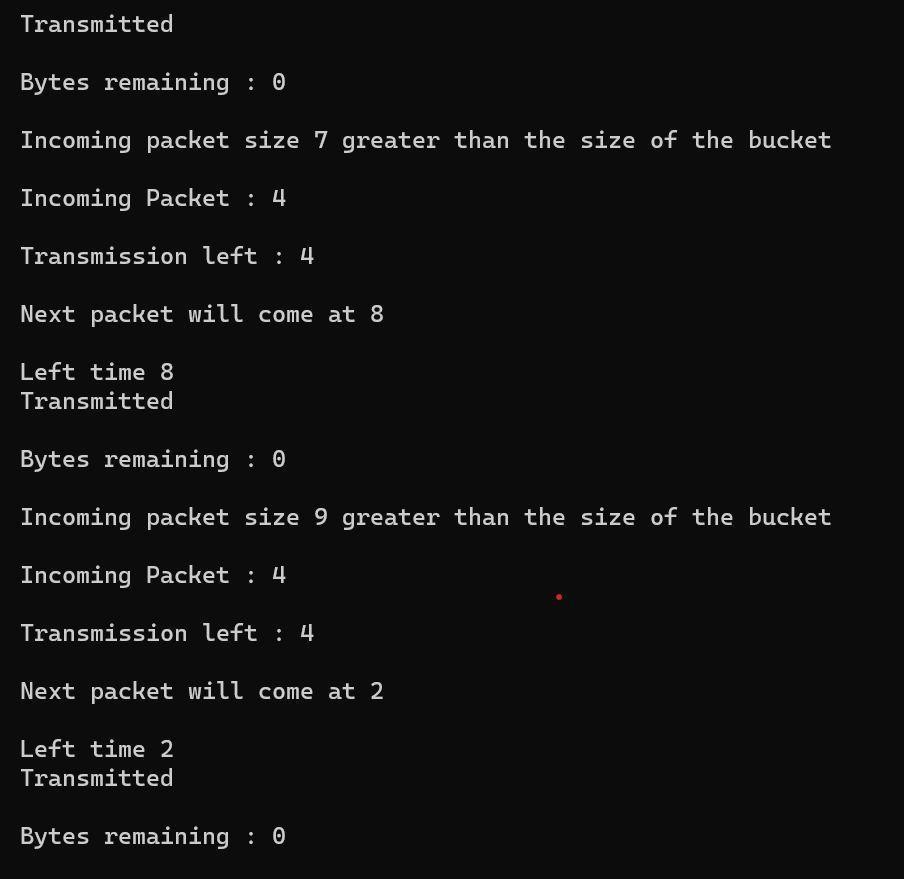
else printf("\n No packets to send\n"); }

}

}

### Output Screenshot





## Program 5

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

**Code:**

**servertcp.py** from socket import \*

serverName='LAPTOP-HATRKFO6

' serverPort = 12530 serverSocket = socket(AF\_INET,SOCK\_STREAM) serverSocket.bind((serverName,serverPort)) serverSocket.listen(1) print ("The server is ready to receive") while 1:

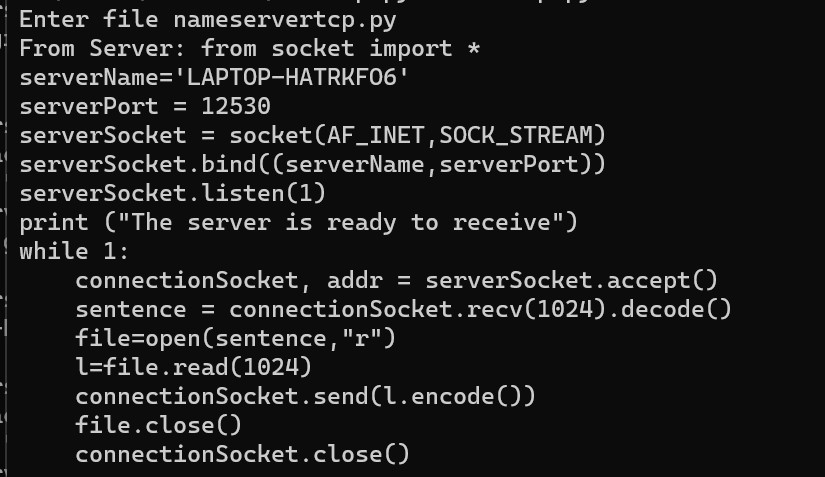
connectionSocket, addr = serverSocket.accept() sentence = connectionSocket.recv(1024).decode() file=open(sentence,"r") l=file.read(1024) connectionSocket.send(l.encode()) file.close() connectionSocket.close()

**clienttcp.py** from socket import \* serverName = 'LAPTOP-HATRKFO6'

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()

### Output Screenshot





## Program 6

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

**Code:**

**serverudp.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,"r") l=file.read(2048) serverSocket.sendto(bytes(l,"utf-8"),clientAddress) print("sent back to client",l) file.close() **clientudp.py** 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()

### Output Screenshot

