VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

VENU GOPAL V (1BM21CS414)

in partial fulfilment 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 **VENU GOPAL V (1BM21CS414),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment 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.

Lohith J J
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Table Of Contents

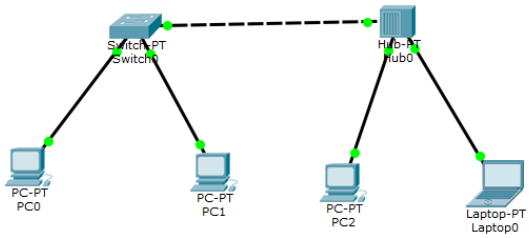
Cycle-1

Experiment 1

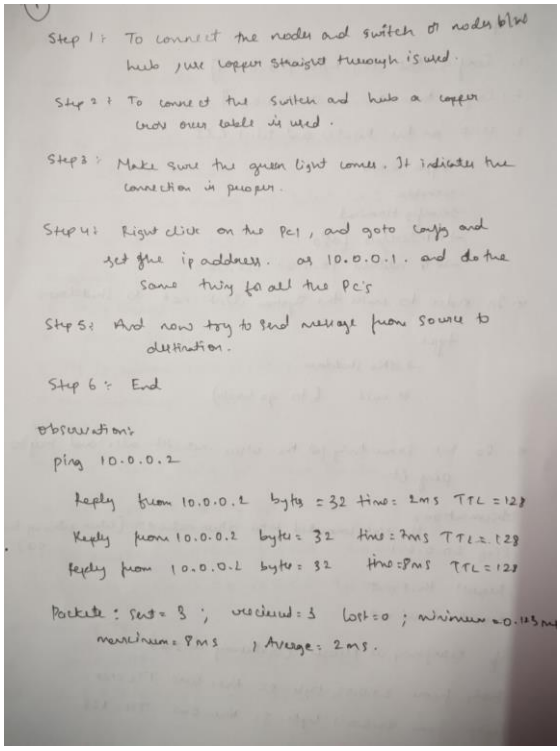
Aim of the program:

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

Topology:



Procedure:



Step 1: To connect the nodes and switch or nodes b/w hub, use copper straight through is used.

Step 2: To connect the switch and hub a copper cross over cable is used.

Step 3: Make sure the green light comes. It indicates the connection is proper.

Step 4: Right click on the PC1, and goto config and set the ip address. as 10.0.0.1. and do the same thing for all the PC's

Step 5: And now try to send message from source to destination.

Step 6: End

Observations

ping 10.0.0.2

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

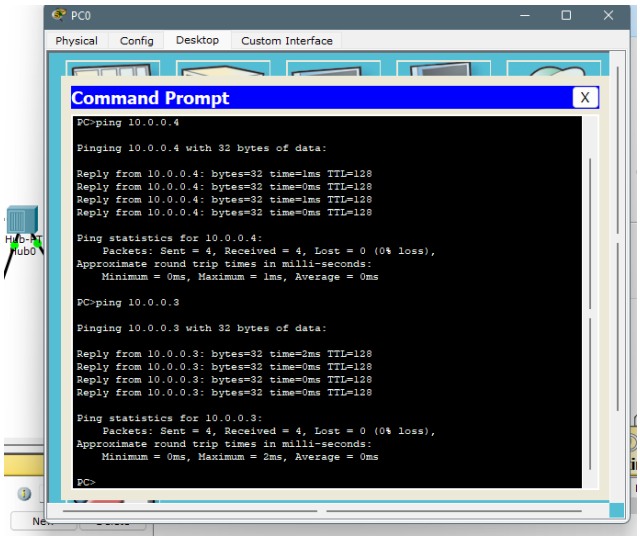
Reply from 10.0.0.2 bytes: 32 time: 7ms TTL = 128

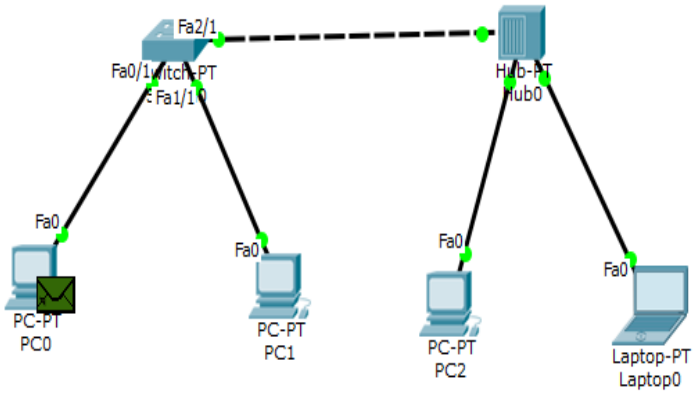
Reply from 10.0.0.2 bytes: 32 time: 8ms TTL = 128

Packets: Sent = 3; received: 3 lost = 0; minimum = 0.125ms

maximum: 8ms, Average: 2ms.

Output:


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
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=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=2ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>

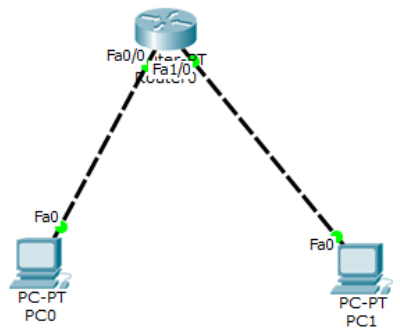

Fa2/1
Fa0/1
Fa1/1
Hub-PT
Hub0
Fa0
PC-PT
PC0
Fa0
PC-PT
PC1
Fa0
PC-PT
PC2
Fa0
Laptop-PT
Laptop0

Experiment 2

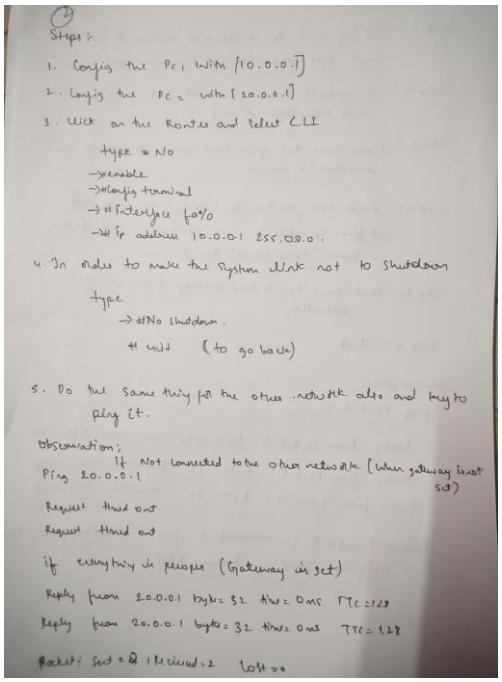
Aim of the program:

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

Topology:



Procedure:



Steps :-

1. Config the PC1 with [10.0.0.1]
2. Config the PC2 with [20.0.0.1]
3. Click on the Router and select CLI

   type = No
   → enable
   → #config terminal
   → # interface fa0/0
   → # IP address 10.0.0.1 255.0.0.0.

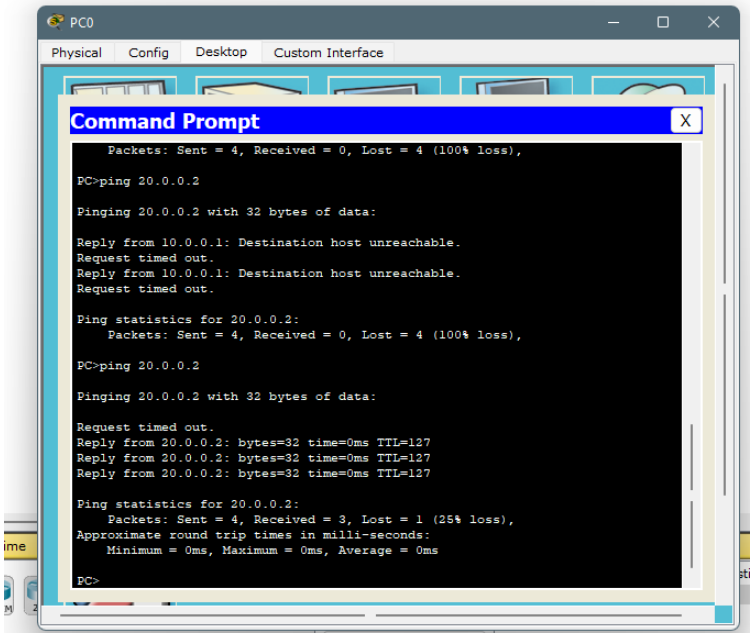
4. In order to make the system link not to shutdown

   type
   → #No shutdown.
   # exit (to go back)

5. Do the same thing for the other network also and try to ping it.

Observation;
If Not connected to the other network (when gateway is not set)
Ping 20.0.0.1
Request timed out
Request timed out
if everything is proper (Gateway is set)
Reply from 20.0.0.1 byte = 32 time = 0ms TTL = 128
Reply from 20.0.0.1 byte = 32 time = 0ms TTL = 128
Packet: Sent = 4 | Received = 2 Lost = 0

Output:

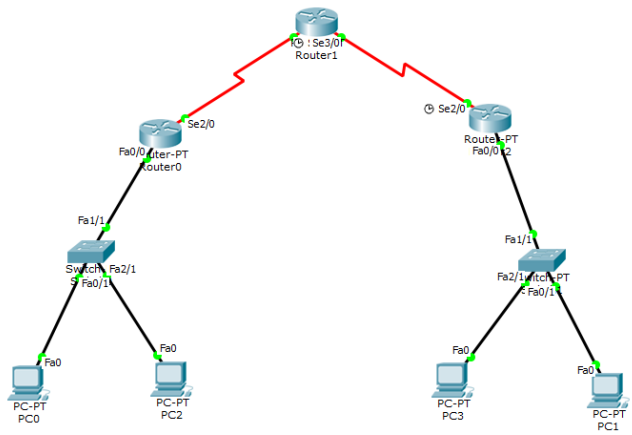

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Reply from 10.0.0.1: Destination host unreachable.
Request timed out.
Reply from 10.0.0.1: Destination host unreachable.
Request timed out.
Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>

Experiment 3

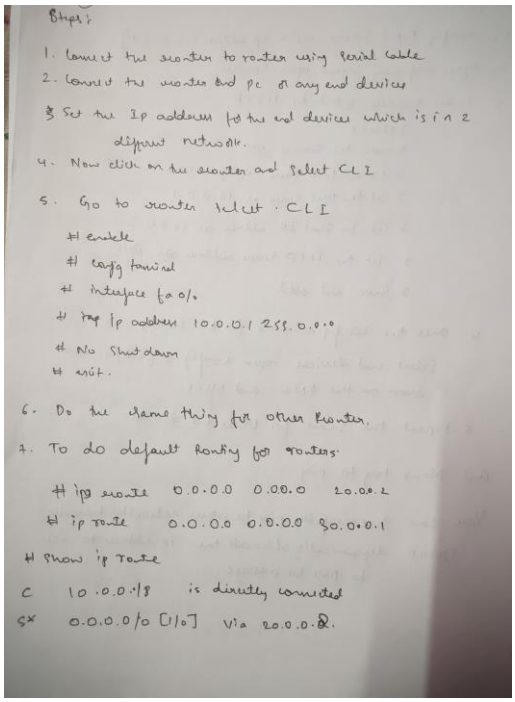
Aim of the program:

Configuring default route to the Router

Topology:



Procedure:



Steps:

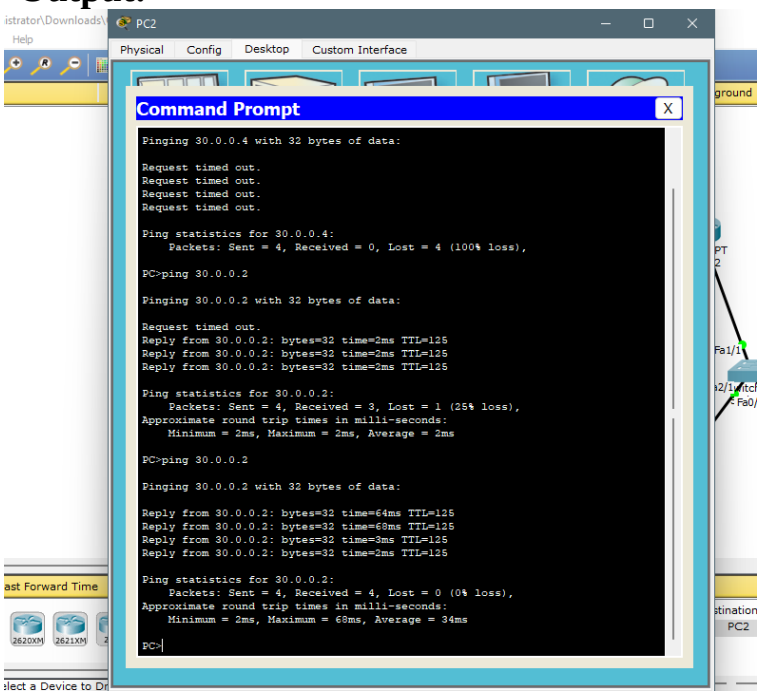
1. Connect the router to router using serial cable
2. Connect the router and pc or any end device
3. Set the Ip address for the end devices which is in a different network.
4. Now click on the router and select CLI
5. Go to router select CLI

```
# enable
# config terminal
# interface fa 0/0
# ip address 10.0.0.1 255.0.0.0
# No shut down
# exit.
```

6. Do the same thing for other Router.
7. To do default Routing for routers.

```
# ip route 0.0.0.0 0.0.0.0 20.0.0.2
# ip route 0.0.0.0 0.0.0.0 30.0.0.1
# show ip route
C   10.0.0.18   is directly connected
S*  0.0.0.0/0 [1/0] via 20.0.0.2.
```

Output:

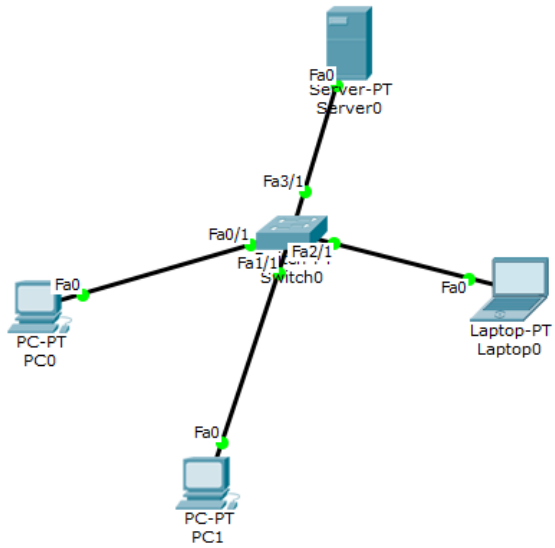

PC2
Physical
Config
Desktop
Custom Interface
Command Prompt
Pinging 30.0.0.4 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 30.0.0.4:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 30.0.0.2
Pinging 30.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 30.0.0.2: bytes=32 time=2ms TTL=125
Reply from 30.0.0.2: bytes=32 time=2ms TTL=125
Reply from 30.0.0.2: bytes=32 time=2ms TTL=125
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 2ms, Average = 2ms
PC>ping 30.0.0.2
Pinging 30.0.0.2 with 32 bytes of data:
Reply from 30.0.0.2: bytes=32 time=64ms TTL=125
Reply from 30.0.0.2: bytes=32 time=68ms TTL=125
Reply from 30.0.0.2: bytes=32 time=3ms TTL=125
Reply from 30.0.0.2: bytes=32 time=2ms TTL=125
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 68ms, Average = 34ms
PC>

Experiment 4

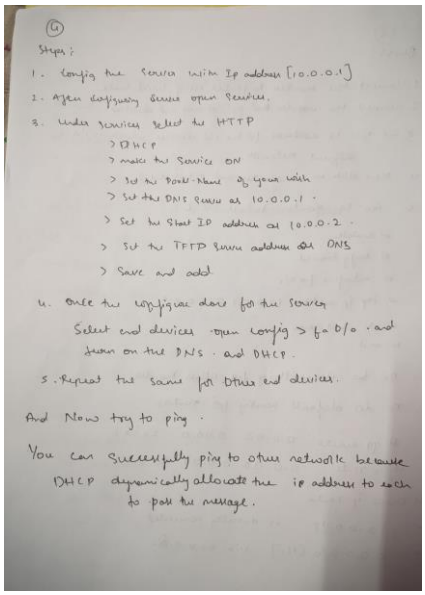
Aim of the program:

Configuring DHCP within a LAN in a packet Tracer

Topology:



Procedure:



(4)

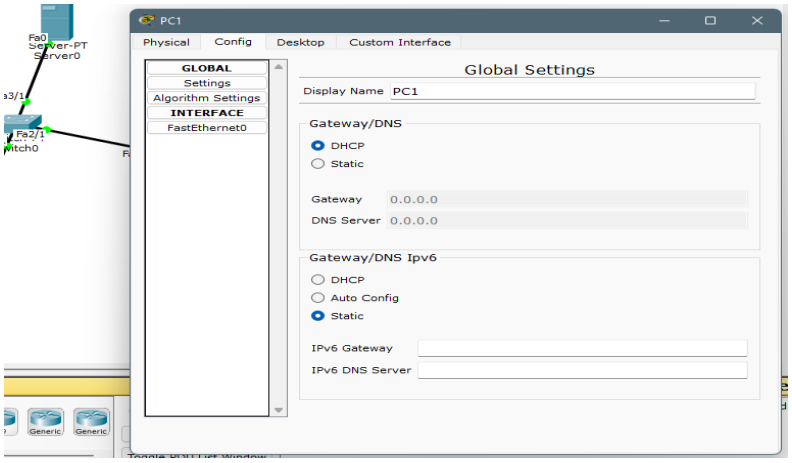
Steps;

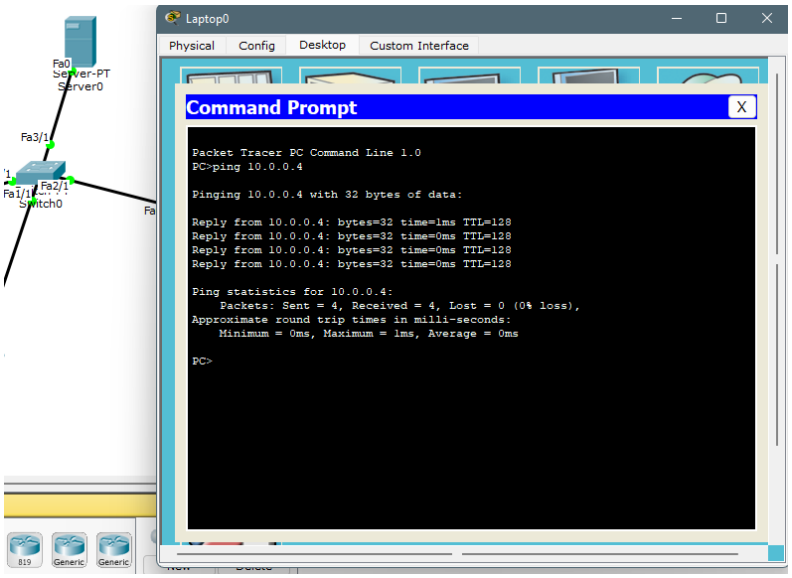
1. Config the Server with IP address [10.0.0.1]
2. After configuring Server open Services.
3. Under Services select the HTTP
   - > DHCP
   - > make the Service ON
   - > Set the Pool-Name as your wish
   - > Set the DNS Server as 10.0.0.1 .
   - > Set the Start IP address as 10.0.0.2 .
   - > Set the TFTP Server address as DNS
   - > Save and add
4. Once the configuration done for the Server
   Select end devices open config > fa 0/0 . and turn on the DNS . and DHCP .
5. Repeat the same for other end devices.

And Now try to ping .

You can successfully ping to other network because DHCP dynamically allocate the ip address to each to pass the message.

Output:


PC1
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
Global Settings
Display Name PC1
Gateway/DNS
DHCP
Static
Gateway 0.0.0.0
DNS Server 0.0.0.0
Gateway/DNS Ipv6
DHCP
Auto Config
Static
IPv6 Gateway
IPv6 DNS Server

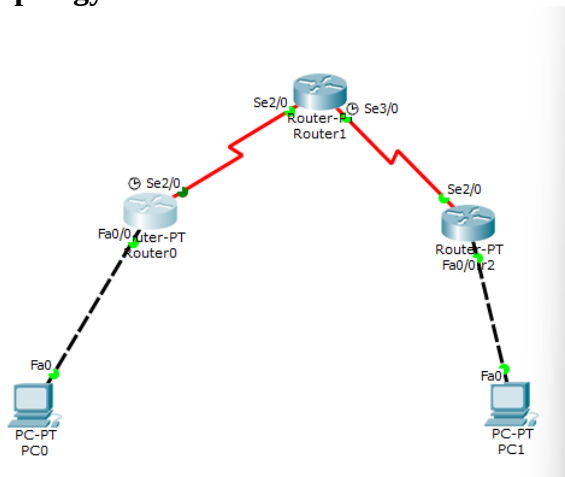

Laptop0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>

Experiment 5

Aim of the program:

Configuring RIP Routing Protocol in Routers

Topology:



Procedure:

(5)

Procedure :-

1) Do the required topology.

2) # enable
config t
interface fa0/0
ip address 10.0.0.1 255.0.0.0.
No shutdown.

3). Repeat the same for all the fastethernet ports.

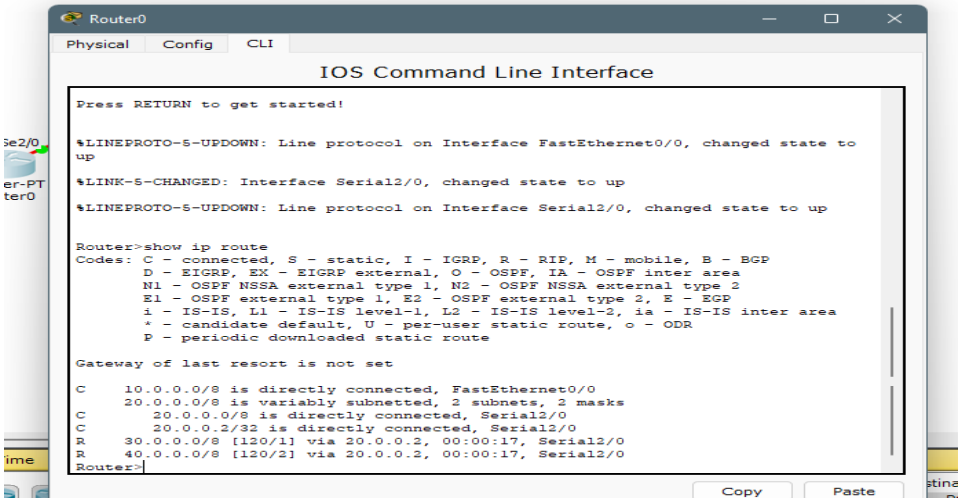
4) Set the gateway.

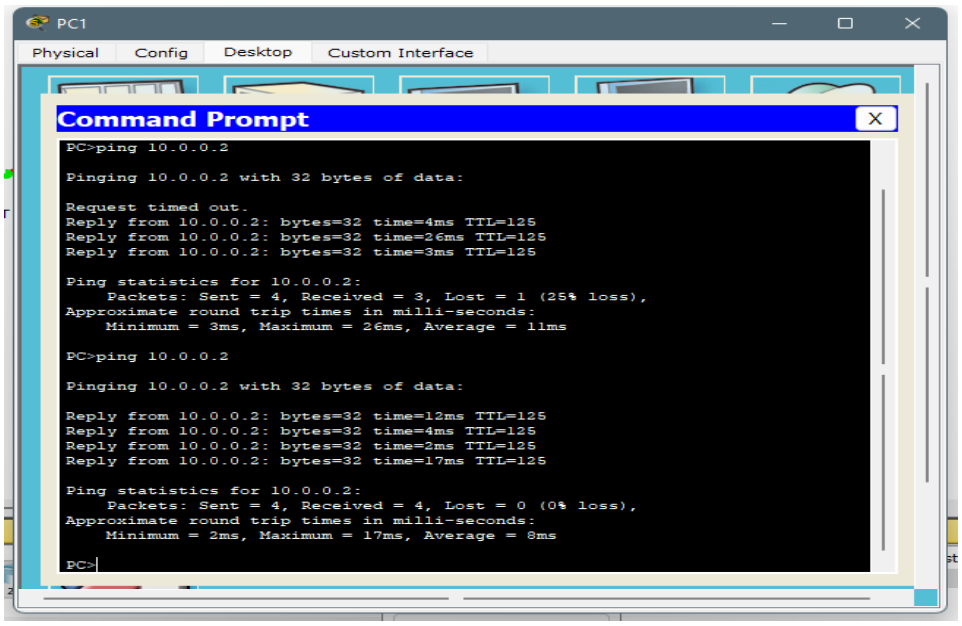
5) At the sender site
interface se2/0
ip address 20.0.0.1 255.0.0.0
encapsulation ppp
clockrate 64000
No shutdown.
exit

6) At the receiver site
encapsulate ppp.
No shutdown.

7) To do RIP routing
router rip
network 10.0.0.0 } adjacent network address.
network 20.0.0.0
exit.

Output:


Router0
Physical
Config
CLI
IOS Command Line Interface
Press RETURN to get started!
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
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
20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 20.0.0.0/8 is directly connected, Serial2/0
C 20.0.0.2/32 is directly connected, Serial2/0
R 30.0.0.0/8 [120/1] via 20.0.0.2, 00:00:17, Serial2/0
R 40.0.0.0/8 [120/2] via 20.0.0.2, 00:00:17, Serial2/0
Router>
Copy
Paste

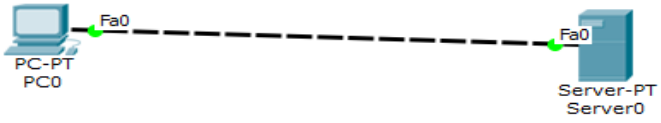

PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 10.0.0.2: bytes=32 time=4ms TTL=125
Reply from 10.0.0.2: bytes=32 time=26ms TTL=125
Reply from 10.0.0.2: bytes=32 time=3ms TTL=125
Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 26ms, Average = 11ms
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=12ms TTL=125
Reply from 10.0.0.2: bytes=32 time=4ms TTL=125
Reply from 10.0.0.2: bytes=32 time=2ms TTL=125
Reply from 10.0.0.2: bytes=32 time=17ms TTL=125
Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 17ms, Average = 8ms
PC>

Experiment 6

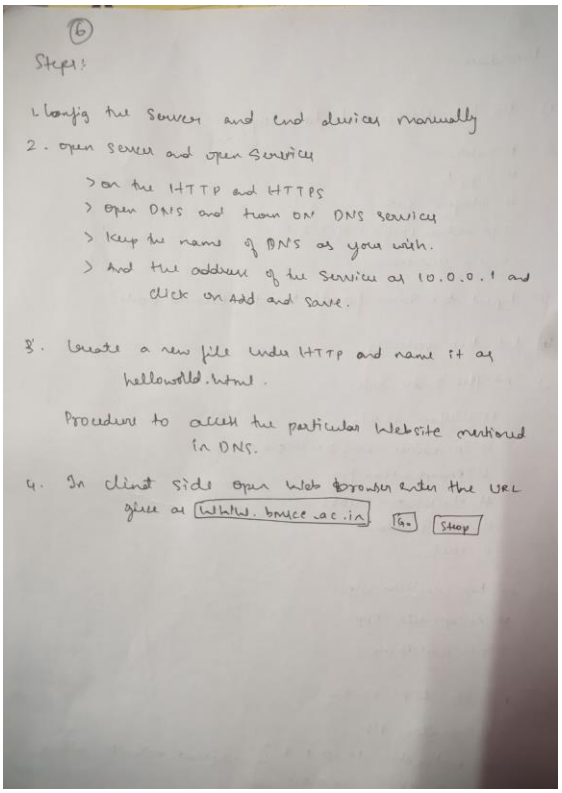
Aim of the program

Demonstration of WEB server and DNS using Packet Tracer

Topology



Procedure:



(6)

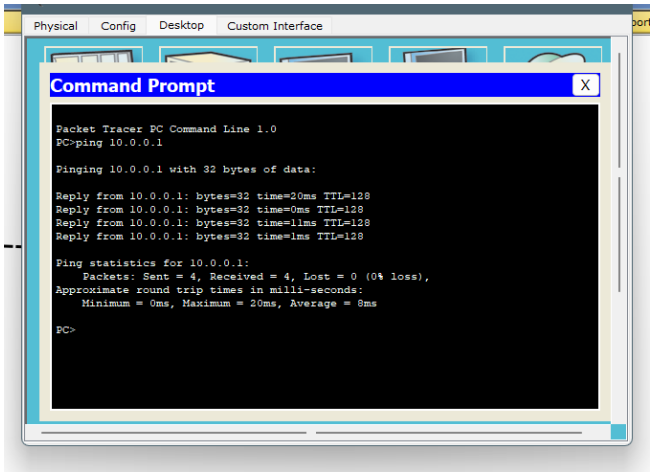
Steps:

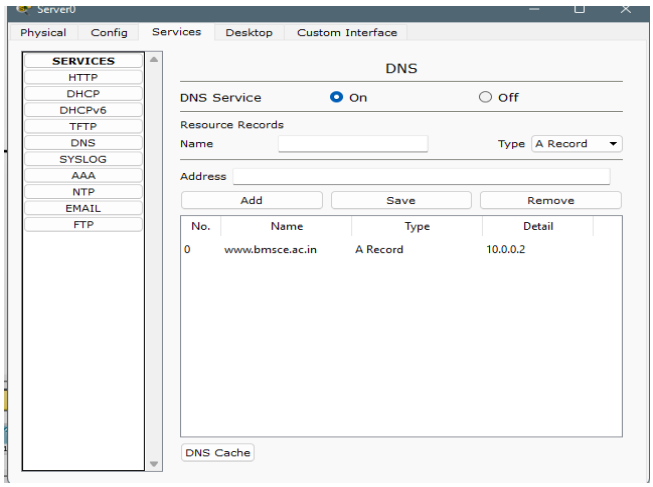
1. Config the server and end devices manually
2. open server and open services
   - > on the HTTP and HTTPS
   - > open DNS and turn ON DNS service
   - > keep the name of DNS as you wish.
   - > And the address of the service as 10.0.0.1 and click on Add and save.
3. Create a new file under HTTP and name it as helloworld.html.

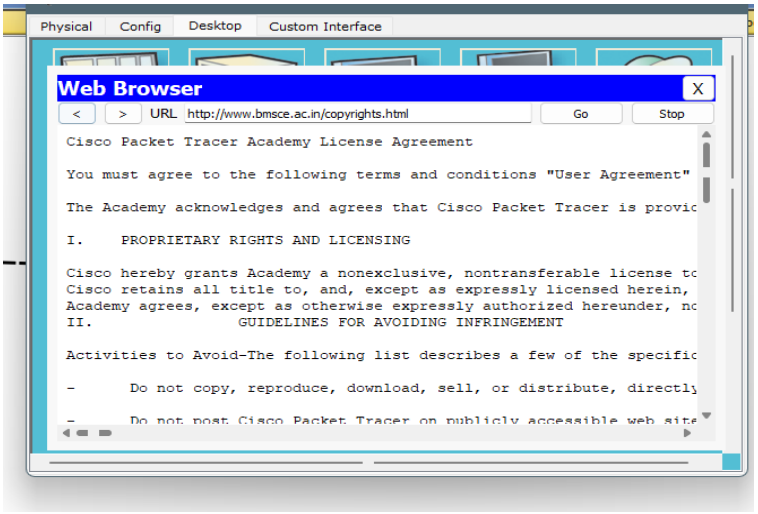
Procedure to access the particular website mentioned in DNS.

4. In client side open web browser enter the URL given as [www.bmsce.ac.in] [Go] [Stop]

Output:


Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=20ms TTL=128
Reply from 10.0.0.1: bytes=32 time=0ms TTL=128
Reply from 10.0.0.1: bytes=32 time=11ms TTL=128
Reply from 10.0.0.1: bytes=32 time=1ms TTL=128
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 20ms, Average = 8ms
PC>


Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
DNS
DNS Service
On
Off
Resource Records
Name
Type
A Record
Address
Add
Save
Remove
No.
Name
Type
Detail
0
www.bmsce.ac.in
A Record
10.0.0.2
DNS Cache


Physical
Config
Desktop
Custom Interface
Web Browser
X
<
>
URL
http://www.bmsce.ac.in/copyrights.html
Go
Stop
Cisco Packet Tracer Academy License Agreement
You must agree to the following terms and conditions "User Agreement"
The Academy acknowledges and agrees that Cisco Packet Tracer is provic
I. PROPRIETARY RIGHTS AND LICENSING
Cisco hereby grants Academy a nonexclusive, nontransferable license tc
Cisco retains all title to, and, except as expressly licensed herein,
Academy agrees, except as otherwise expressly authorized hereunder, nc
II. GUIDELINES FOR AVOIDING INFRINGEMENT
Activities to Avoid-The following list describes a few of the specific
- Do not copy, reproduce, download, sell, or distribute, directly

Cycle-2

Experiment 1

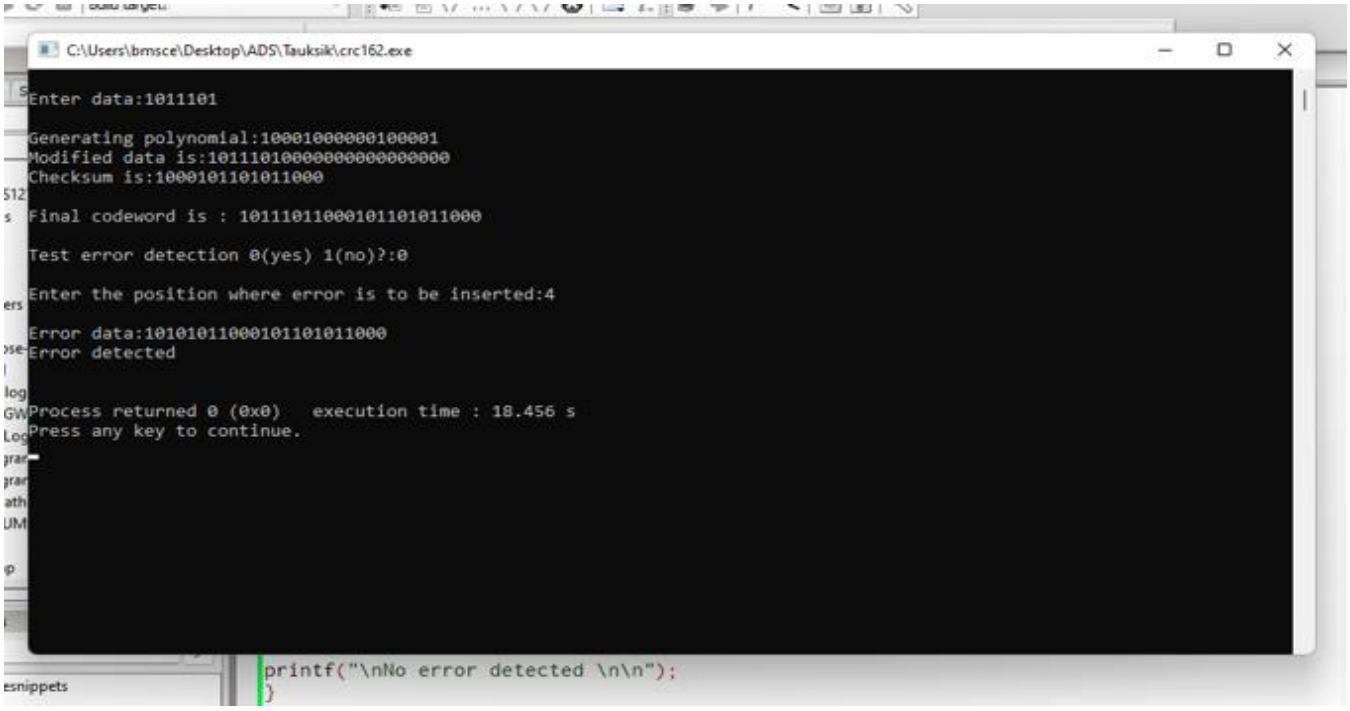
Aim of the Experiment:

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

```
#include <stdio.h>
#include <string.h>
#define N strlen(gen)
char modif[28],checksum[28],gen[28];
int a,e,c,b;
void xor()
{
for(c=1;c<N;c++)
checksum[c]=((checksum[c]==gen[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
checksum[e]=modif[e];
do
{
if(checksum[0]=='1')
xor();
for(c=0;c<N-1;c++)
checksum[c]=checksum[c+1];
checksum[c]=modif[e++];
}while(e<=a+N-1);
}
int main()
{
int flag=0;
```

```
strcpy(gen,"10001000000100001");
printf("\nEnter data:");
scanf("%s",modif);
printf("\nGenerating polynomial:%s\n",gen);
a=strlen(modif);
for(e=a;e<a+N-1;e++)
modif[e]='0';
printf("Modified data is:%s\n",modif);
crc();
printf("Checksum is:%s\n",checksum);
for(e=a;e<a+N-1;e++)
modif[e]=checksum[e-a];
printf("\nFinal codeword is : %s\n",modif);
printf("\nTest error detection 0(yes) 1(no)?:");
scanf("%d",&e);
if(e==0)
{
do{
printf("\nEnter the position where error is to be
    inserted:");
scanf("%d",&e);
}
while(e==0||e>a+N-1);
modif[e-1]=(modif[e-1]=='0')?'1':'0';
printf("\nError data:%s\n",modif);
}
crc();
for(e=0;(e<N-1)&&(checksum[e]!='1');e++);
if(e<N-1)
printf("Error detected\n\n");
else
printf("\nNo error detected \n\n");}
```

Output:


C:\Users\bmsce\Desktop\ADS\Tauksik\crc162.exe
Enter data:1011101
Generating polynomial:10001000000100001
Modified data is:10111010000000000000000
Checksum is:1000101101011000
Final codeword is : 10111011000101101011000
Test error detection 0(yes) 1(no)?:0
Enter the position where error is to be inserted:4
Error data:10101011000101101011000
Error detected
Process returned 0 (0x0) execution time : 18.456 s
Press any key to continue.
printf("\nNo error detected \n\n");
}

Experiment 2

Aim of the Experiment

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

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[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 the cost matrix :\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
scanf("%d",&dmat[i][j]);
dmat[i][i]=0;
rt[i].dist[j]=dmat[i][j];
rt[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(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
{
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<n;i++)
{
printf("\n\nState value for router %d is \n",i+1);
for(j=0;j<n;j++)
{
printf("\t\nnode %d via %d Distance %d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n\n");
}
```

Output:

```
C:\Users\bmsce\Desktop\dvr

Enter the number of nodes : 5

Enter the cost matrix :
0 1 2 3 9999
1 0 9999 4 1
2 9999 0 5 8
3 4 5 0 2
9999 1 8 2 0


State value for router 1 is

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

State value for router 2 is

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

State value for router 3 is

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

State value for router 4 is

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

State value for router 5 is

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

Experiment 3

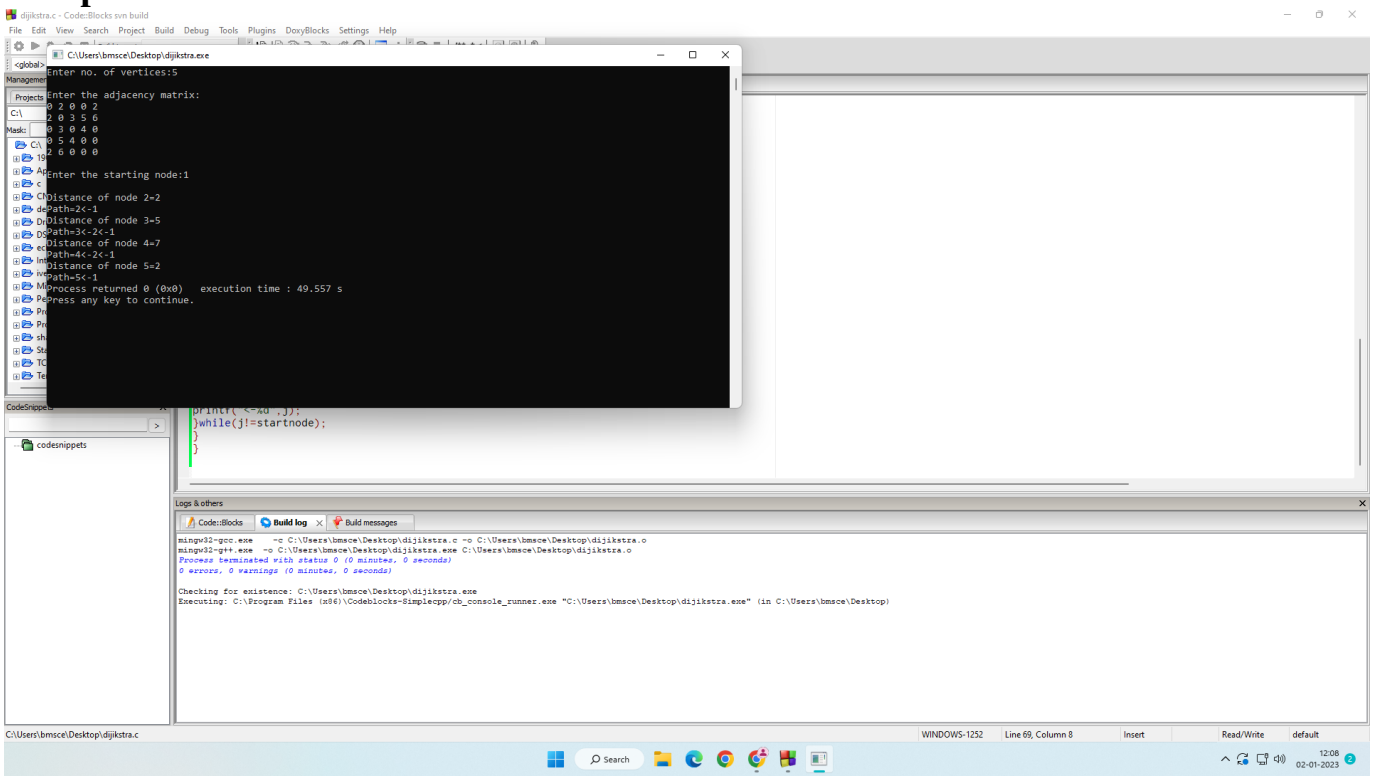
Aim of the Experiment:

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

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=1;i<=n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
distance[startnode]=0;
```

```
visited[startnode]=1;
count=1;
while(count<n-1)
{
mindistance=INFINITY;
//nextnode gives the node at minimum distance
for(i=1;i<=n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
nextnode=i;
}
//check if a better path exists through nextnode
visited[nextnode]=1;
for(i=1;i<=n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}

//print the path and distance of each node
for(i=1;i<=n;i++)
if(i!=startnode)
{
printf("\nDistance of node %d=%d",i,distance[i]);
printf("\nPath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}while(j!=startnode);
}
}
```

Output:


C:\Users\bmsce\Desktop\dijikstra.exe
Enter no. of vertices:5
Enter the adjacency matrix:
0 2 0 0 2
2 0 3 5 6
0 3 0 4 0
0 5 4 0 0
2 6 0 0 0
Enter the starting node:1
Distance of node 2=2
Path=2<-1
Distance of node 3=5
Path=3<-2<-1
Distance of node 4=7
Path=4<-2<-1
Distance of node 5=2
Path=5<-1
Process returned 0 (0x0) execution time : 49.557 s
Press any key to continue.
}while(j!=startnode);
}
}
Logs & others
Code::Blocks
Build log
Build messages
mingw32-gcc.exe -c C:\Users\bmsce\Desktop\dijikstra.c -o C:\Users\bmsce\Desktop\dijikstra.o
mingw32-g++.exe -o C:\Users\bmsce\Desktop\dijikstra.exe C:\Users\bmsce\Desktop\dijikstra.o
Process terminated with status 0 (0 minutes, 0 seconds)
0 errors, 0 warnings (0 minutes, 0 seconds)
Checking for existence: C:\Users\bmsce\Desktop\dijikstra.exe
Executing: C:\Program Files (x86)\Codeblocks-Simplecpp/cb_console_runner.exe "C:\Users\bmsce\Desktop\dijikstra.exe" (in C:\Users\bmsce\Desktop)
C:\Users\bmsce\Desktop\dijikstra.c
WINDOWS-1252
Line 69, Column 8
Insert
Read/Write
default

Experiment 4

Aim of the Experiment:

Write a program for congestion control using Leakybucket algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void bucket(int send);
void bucketoverflow(int send);
int bucketsize=30,bucketmax=60;
int bucketrate=3;
int main()
{
   int i=0;
   while(i<10)
   {
      printf("\n1.Send packets\n2.Nothing to send\n3.Exit\nEnter your choice:\n");
   int ch;
   scanf("%d",&ch);
      switch(ch)
      {
      case 1:
         printf("Enter the packet size required to be sent:\n");
         int send;
         scanf("%d",&send);
         if(send<bucketmax-bucketsize)
         {
            bucket(send);
            printf("Packets sent successfully\n");
         }
         else{
            printf("Error");
            bucketoverflow(send);
         }
         printf("Bucket size = %d\n",bucketsize);
         break;
      case 2:
         printf("No packets sent\n");
         bucketsize-=3;
         printf("Bucket size = %d\n",bucketsize);
         break;
      case 3:
         exit(0);
         break;
      default:
         printf("Invalid option");
      }
      i+=1;
   }
```

```
    return 0;
}
void bucket(int send)
{
bucketsize+=send;
bucketsize-=3;
}
void bucketoverflow(int send)
{
bucketsize-=3;
}
```

Output:

```
1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
1
Enter the packet size required to be sent:
20
Packets sent successfully
Bucket size = 47

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
1
Enter the packet size required to be sent:
12
Packets sent successfully
Bucket size = 56

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
2
No packet sent
Bucket size = 53

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
2
No packet sent
Bucket size = 50

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
2
No packet sent
Bucket size = 47

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
2
No packet sent
Bucket size = 44

1.Send packets
2.Nothing to send
3.Exit
Enter your choice:
3

Process returned 0 (0x0)   execution time : 21.198 s
Press any key to continue.
```

Experiment 5

Aim of the Experiment:

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

Server:

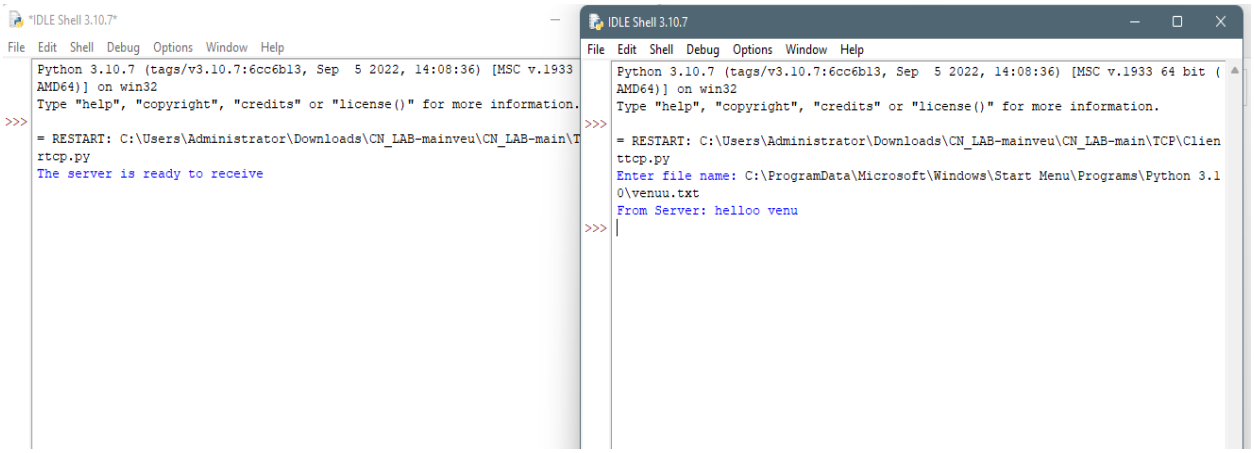
```
from socket import *
serverName = “Venu”
serverPort = 12530
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
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()
```

Output:



Experiment 6

Aim of the Experiment:

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.

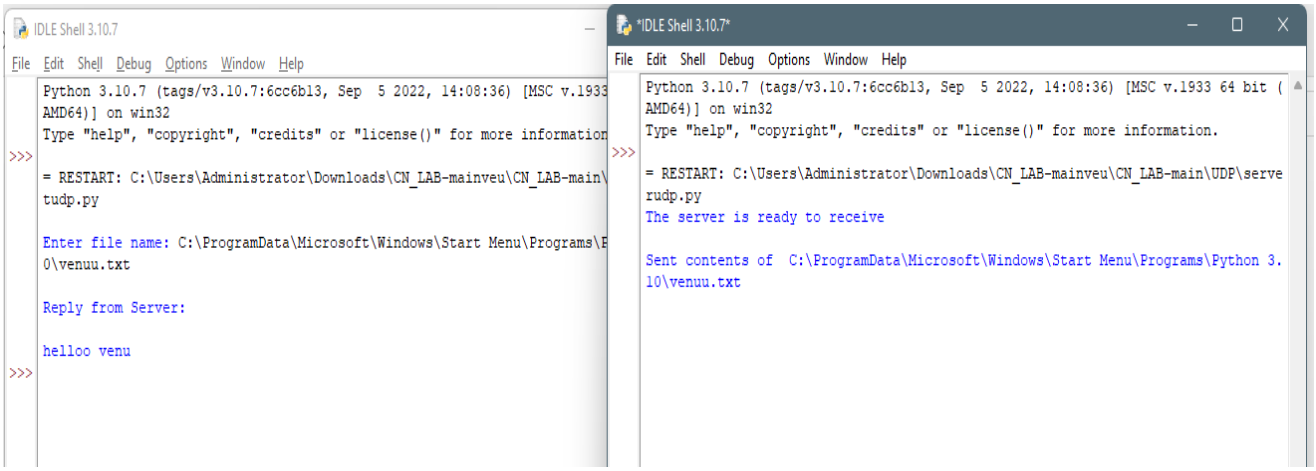
Server:

```
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()
```

Output:


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information
>>>
= RESTART: C:\Users\Administrator\Downloads\CN_LAB-mainveu\CN_LAB-main\
tudp.py
Enter file name: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\P
0\venuu.txt
Reply from Server:
helloo venu
>>>
IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Administrator\Downloads\CN_LAB-mainveu\CN_LAB-main\UDP\serve
rudp.py
The server is ready to receive
Sent contents of C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Python 3.
10\venuu.txt