TPCT's

College of Engineering, Osmanabad

Laboratory Manual

Computer Science And Engineering

For

Third Year Students

Manual Prepared by

R.B.Randive

Author COE, Osmanabad

TPCT's

College of Engineering
Solapur Road, Osmanabad
Department Computer Science And Engineering

Vision of the Department:

To produce trained computer professionals who can successfully meet the demands of academia, IT Industry and research by building a strong teaching and research environment.

Mission of the Department:

To provide industry and research oriented quality education to UG and PG students and train them to apply this knowledge for solving real world problems and make them competitive in the ever-changing and challenging global work environment

College of Engineering

Technical Document

This technical document is a series of Laboratory manuals of Computer Science And Engineering Department and is a certified document of College of engineering, Osmanabad. The care has been taken to make the document error-free. But still if any error is found, kindly bring it to the notice of subject teacher and HOD.

Recommended by,

HOD

Approved by,

Principal

FOREWORD

It is my great pleasure to present this laboratory manual for second year engineering students for the subject of Computer Network II to understand and visualize the basic concepts of networking using different algorithm. Computer Network II cover basic concepts of Networking. This being a core subject, it becomes very essential to have clear theoretical and designing aspects.

This lab manual provides a platform to the students for understanding the basic concepts of Computer Network II. This practical background will help students to gain confidence in qualitative and quantitative approach to Computer Network II.

m

H.O.D
CSE Dept

LABORATORY MANUAL CONTENTS

This manual is intended for the Second Year students of CSE branches in the subject of Computer Network II. This manual typically contains practical/ Lab Sessions related to Computer Network II covering various aspects related to the subject for enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

SUBJECT INDEX:

1. Do’s & Don’ts in Laboratory.

2. Lab Exercises
    1.
    2.
        .
        .
        .
        .

3. Quiz
4. Conduction of viva voce examination
5. Evaluation & marking scheme

1.Dos and Don'ts in Laboratory :-

1. Do not handle any equipment before reading the instructions /Instruction manuals.
2. Read carefully the power ratings of the equipment before it is switched ON, whether ratings 230 V/50 Hz or 115V/60 Hz. For Indian equipment, the power ratings are normally 230V/50Hz. If you have equipment with 115/60 Hz ratings, do not insert power plug, as our normal supply is 230V/50Hz., which will damage the equipment.
3. Observe type of sockets of equipment power to avoid mechanical damage.
4. Do not forcefully place connectors to avoid the damage.
5. Strictly observe the instructions given by the Teacher/ Lab Instructor.

Instruction for Laboratory Teachers:-

1. Submission related to whatever lab work has been completed should be done during the next lab session.
2. Students should be instructed to switch on the power supply after getting the checked by the lab assistant / teacher. After the experiment is over, the students must hand over the circuit board, wires, CRO probe to the lab assistant/teacher.
3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

2.Lab Exercises

1. i) Convert from 32 bit binary IP address to dotted decimal IP address
   ii) Convert a dotted decimal IP form to 32 bit binary IP form
   iii) Convert a dotted decimal IP form ti its class and display the network and host portions
   iv) Convert from IP notation to dotted decimal IP form
   v) Convert from IP subnet dotted IP form to CIDR
2. Obtain routing table at each node using distance vector routing algorithm for a given subnet
3. To simulate the implementing routing protocol using border gateway protocol(BGP)
4. To simulate the open shortest path first routing protocol based on cost assigned to path
5. Implement Dijkstra algorithm to compute the shortest path through a given graph
6. Simulation of sliding window protocol
7. Study of implementation of DHCP
8. Study of implementation of DNS
9. Configure a network topology using packet tracer software
10. Using RSA algorithm encrypt a text data and decrypt the same

Experiment No.1
Conversion

Aim:- . i) Convert from 32 bit binary IP address to dotted decimal IP address

ii) Convert a dotted decimal IP form to 32 bit binary IP form

iii) Convert a dotted decimal IP form ti its class and display the network and host portions

iv) Convert from IP notation to dotted decimal IP form

v) Convert from IP subnet dotted IP form to CIDR

Objective: IP Addressing Internet Architecture IPv4 Addressing **IP address** Classes Subnets and subnet mask Subnets design with IP addressing.

Hardware Requirement: Intel based desktop pc: RAM 512 MB

Software Requirement: Turbo C/Borland C

Theory: IP enables hosts to communicate with each other at the Network layer. In IPv4 each packet contains a source and destination address. That's how the routers on the network knows where the packet is coming from and where they must forward the packet. The IP addresses fields are represented in 32 bits. Routers know how to interpret those numbers, but for a human to understand them would be too difficult. From our point of view, we use what's called a dotted decimal address. A dotted decimal address is the human representation of the binary address. For example, the address **192.168.10.1** is a dotted decimal address. In its binary form, the address is **11000000101010000000101000000001**. IP addresses have 4 octets. For example, 192 is the first octet, 168 the second, 10 the third and 1 the last octet. In its binary form, 11000000 is the first octet, 10101000 the second, 00001010 the third and 00000001 the last octet. Every octet, in its decimal form, can get a value from 0 to 255.

In the binary system there are only 1s and 0s. Depending on their position in the octet, they get different values. Each position is a power of 2. To get the decimal number you have to sum up those number.

1	1	1	1	1	1	1	1
2^7 = 128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1

For example, we know that 10101000 is 168. But let's see if we get the same number after we make the calculations.

2^7 * 1 + 2^6 * 0 + 2^5 * 1 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 0 + 2^0 * 0 = 128 + 32 + 8 = 168

Let's now learn how to convert those numbers from decimal to binary. The decimal to binary conversion is similar to binary to decimal conversion. Keep in mind those powers of 2. When you calculate the binary value, you take those powers of 2 and compare them with your number. If your number is greater, you write down 1 and you subtract that power of 2 from the number. If your number is lower than the power of 2, you write down 0. You continue to make the calculations until you reach 2^0.

Class A addresses were the ones from 0.0.0.0 to 127.255.255.255. A class A network is has a default netmask of 255.0.0.0 allowing for up to 16,777,214 hosts per network (2^24 – 2). However, there's possible to create only 128 network from the whole class A space. Or at least it was, back in the days, when classless routing was not used.

Class B addresses are from 128.0.0.0 to 191.255.255.255. The whole class B was able to create 16,384 networks (2^14) with a maximum number of 65,534 hosts per network (2^16 – 2). The default netmask si 255.255.0.0.

Class C networks were found within the 192.0.0.0 – 223.255.255.255 range. This class allowed for more networks -2,097,150 (2^21) but the maximum hosts per network was only 254 (2^8 – 2).

The class D and the class E address blocks are the same used today for multicasting, respectively the experimental addresses.

Classless interdamain routing (CIDR) is a novel standard in routing intended to slow down the rapid increase of volume in the routing table on the internet .It help to solve the problem of large organization in allocating the IP address by using VLSM (variable length subnet masking) not only does it base itself on VLSM, but it also reduce the amount of values in a global routing table significantly by advertising collection of subnet instead of each subnet individually. Calculating the CIDR subnet is simple.

- **Calculating CIDR subnet.**
- 1. Make the note of CIDR address suffix by (in this content ,the suffix means the part after the"/" in the CIDR address decimal notation).This is crucial in the calculating the subnet address for ex.if address is 222.125.44.20/17 then the suffix no is 17.
- 2. You will need 32 digit write a many one's as the suffix values, and fill in the rest of the 32 bit as zero using the ex. Address, our binary value would be 11111111111111111000000000000000
- 3. Split your 32 bit digit into 4 octet using our ex. This would be done in the following way:11111111.11111111.10000000.00000000
- 4. Convert each octet into decimal values. With our ex.,the result would be 255.255.128.0.

Program:-

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
intreadBinary(char bin[4][8])
{
charbinstring[80];
inti,j;
intnextchar=0;
intlen;
printf("Enter 32 bit IP address in binary form (******** ******** ******** ********) : ");
fflush(stdout);
scanf("%[^\r\n]",binstring);
len=strlen(binstring);
for(i=0;i<4;i++){
if(nextchar>=len) {
printf("Invalid input string(too short)\n");
return 0;
}
for(j=0;j<8;j++)
{
while(nextchar<len&&binstring[nextchar]==' ')
nextchar++;
if(nextchar>=len)
{
printf("invalid input string (too short)\n");
return 0;
}
bin[i][j]=binstring[nextchar];
nextchar++;
if(bin[i][j]!='0' && bin[i][j]!='1'){
printf("Invalid input string(not a binary no)\n");
return 0;
}
}
}
return 1;
}
intreadIPAddress(intipaddr[]){
chardecstring[4][4];
inti;
printf("Enter 32bit IP Address in dotted decimal notation(***.***.***.***) : ");
fflush(stdout);
if(scanf("%[^.].%[^.].%[^.].%[^\r\n]",decstring[0],decstring[1],decstring[2],decstring[3])<4)
{
printf("Invalid input string(wrong ip address format )\n");
return 0;
```

```
}
for(i=0;i<4;i++)
{
ipaddr[i]=atoi(decstring[i]);
if(ipaddr[i]<0 || ipaddr[i]>255)
{
printf("Invalid input string (Incorrect no's for IP address)\n");
return 0;
}
}
return 1;
}
intreadIPSlashAddress(intipaddr[],int *slash)
{
chardecstring[4][4];
charslashstring[3];
inti;
printf("Enter IP address in CIDR notation (***.***.***.***/**) :");
fflush(stdout);
if(scanf("%[^.].%[^.].%[^.].%[^/]/%[^\r\n]",decstring[0],decstring[1],decstring[2],decstring[3],sla
shstring) <5)
{
printf("Invalid input string(wrong IP address format)\n");
return 0;
}
for(i=0;i<4;i++)
{
ipaddr[i]=atoi (decstring[i]);
if(ipaddr[i]<0|| ipaddr [i]>255 )
{
printf("invalid input string (incorrect no for ip address)\n");
return 0;
}
}
*slash =atoi(slashstring);
if(*slash<0|| *slash>32)
{
printf("invalid input string (invalid netmask length)\n");
return 0;
}
return 1;
}
intreadIPMaskAddress(intipaddr[],int mask[])
{
chardecstring[4][4];
charmaskstring[4][4];
inti;
```

```
printf("Enter IP Address /subnet you want converted to CIDR slash
form(***.***.***.***/***.***.***.***.) : ");
fflush(stdout);
if(scanf("%[^.].%[^.].%[^.].%[^/]/[^.].%[^.].%[^.].%[^\r\n]",decstring[0],decstring[1],decstring[2]
,decstring[3],maskstring[0],maskstring[1],maskstring[2],maskstring[3])<8)
{
printf("Invalid input string(wrong ip address/netmass )\n");
return 0;
}
for(i=0;i<4;i++)
{
if(ipaddr[i]<0||ipaddr[i]>255)
{
printf("Invalid input string(Incorrect number for IP address)\n");
return 0;
}
}
for(i=0;i<4;i++)
{
mask[i]=atoi(maskstring[i]);
if(mask[i]<0||mask[i]>255)
{
printf("Invalid input string(Incorrect number for netmask)/n");
return 0;
}
}

return 1;
}
voidconvertBinToDec(char bin [4][8],intdec[])
{
inti,j;
for(i=0;i<4;i++)
 {
dec[i]=0;
for(j=0;j<8;j++)
 {
dec[i]*=2;
if(bin[i][j]=='1')
dec[i]++;
 }
 }
 }
voidconvertDecToBin(intdec[],char bin[4][9])
 {
inti,j;
for(i=0;i<4;i++)
```

```
 {
for(j=7;j>=0;j--)
 {
bin [i][j]=(dec[i]& 1)+'0';
dec[i]/=2;
 }
bin[i][8]=0;
 }
 }
voidshowNetworkHost(intipaddr[],intnumNetwork){
inti;
printf("Network portion is :");
for(i=0;i<=numNetwork;i++)
printf("%d.",ipaddr[i]);
printf("\n");
printf("Host portion is : ");
for(i=numNetwork+1;i<4;i++)
printf(".%d",ipaddr[i]);
printf("\n");
 }
voidconvertCIDRToNetmask(intslash,int mask[])
 {
char bin[4][8];
inti,j;
for(i=0;i<4;i++)
for(j=0;j<8;j++)
{
if(slash>0)
{
bin[i][j]='1';
slash--;
}
else
bin[i][j]='0';
}
convertBinToDec(bin,mask);
}
intconvertNetmaskToCIDR(int mask[],int *slash)
{
char bin[4][9];
inti,j;
charlastChar;
convertDecToBin(mask,bin);
lastChar='1';
*slash=0;
for(i=0;i<4;i++)
for(j=0;j<8;j++)
```

```
{
if(bin[i][j]=='1')
{
if(lastChar=='0')
{
printf("cannot convert netmask to CIDR notation\n");
return 0;
}
(*slash)++;
}
lastChar=bin[i][j];
}
return 1;
}
intmain()
{
intchoice ;
char bin[4][8];
char bin2[4][9];
intipaddr[4];
intmask[4];
int slash;
char temp[80];
do
{
printf("\nSelect from one of the choice \n\n:");
printf("1)Convert from 32 bit binary ip form to decimal ip form \n");
printf("2)Convert from dotted decimal ip form to 32bit binary ip form \n");
printf("3)convert a dotted decimal ip form to its class and display the network and host portions
seprately\n");
printf("4)Convert from Ip/CIDR slash notation to dotted decimal ip form\n");
printf("5)Convert from IP/subnet dotted ip form to CIDR slash form\n");
printf("6)Quit the program\n ");
printf("\nEnter your choice : ");
fflush(stdout);
scanf("%d",&choice );
scanf("%[^\n]",temp);
scanf("%c",temp);
switch(choice)
{
case (1) :
if(!readBinary(bin))
break;
convertBinToDec(bin,ipaddr);
printf("The address converted top decimal is :
%d.%d.%d.%d\n",ipaddr[0],ipaddr[1],ipaddr[2],ipaddr[3]);
break;
```

```
case(2):
if(!readIpAddress(ipaddr))
break;
convertDecToBin(ipaddr,bin2);
printf("The address converted to binary is :%8s %8s %8s %8s
\n",bin2[0],bin2[1],bin2[2],bin2[3]);
break;
case(3):
if(!readIPAddress(ipaddr))
break;
if(ipaddr[0]<128)
{
printf("\nThisaddrress belongs to class A \n");
showNetworkHost(ipaddr,0);
}
else if(ipaddr[0]<192)
{
printf("\nThisaddrress belongs to class B \n");
showNetworkHost(ipaddr,1);
}
else if(ipaddr[0]<224)
{
printf("\nThisaddrress belongs to class C \n");
showNetworkHost(ipaddr,2);
}
else if(ipaddr[0]<240)
{
printf("\nThisaddrress belongs to class D \n");
}
else
printf("\nThisaddrress belongs to class E \n");
break;
case(4):
if(!readIPSlashAddress(ipaddr,&slash))
break;
convertCIDRToNetmask(slash,mask);
printf("IP/Netmask in dotted decimal ip form
:%d.%d.%d.%d./%d.%d.%d.%d\n",ipaddr[0],ipaddr[1],ipaddr[2],ipaddr[3],mask[0],mask[1],mas
k[2],mask[3]);
break;
case (5):
if(!readIPMaskAddress(ipaddr,mask))
break;
if(!convertNetmaskToCIDR(mask,&slash))
break;
printf("Ip/Netmask_length in CIDR notation :
%d.%d.%d.%d/%d",ipaddr[0],ipaddr[1],ipaddr[2],ipaddr[3],slash);
```

```
break;
case(6):
system("clear");
exit(0);
default :
printf("I dontknoe the option %d.\n",choice);
printf("Try Again \n");
break;
   }
   }
while(1);
   }
```

Observations:-

```
:1)Convert from 32 bit binary ip form to decimal ip form
2)Convert from dotted decimal ip form to 32bit binary ip form
3)convert a dotted decimal ip form to its class and display the network and host
 portions seprately
4)Convert from Ip/CIDR slash notation to dotted decimal ip form
5)Convert from IP/subnet dotted ip form to CIDR slash form
6)Quit the program

Enter your choice : 1
Enter 32 bit IP address in binary form (******** ******** ******** ********) : 1
1111111 11001100 00001111 11010010
The address converted top decimal is : 255.204.15.210

Select from one of the choice
```

```
Enter your choice : 2
Enter 32bit IP Address in dotted decimal notation(***.***.***.***) : 173.194.36.
36
The address converted to binary is :10101101 11000010 00100100 00100100

Select from one of the choice

:1)Convert from 32 bit binary ip form to decimal ip form
2)Convert from dotted decimal ip form to 32bit binary ip form
3)convert a dotted decimal ip form to its class and display the network and host
 portions seprately
4)Convert from Ip/CIDR slash notation to dotted decimal ip form
5)Convert from IP/subnet dotted ip form to CIDR slash form
6)Quit the program

Enter your choice :
```

```
Enter your choice : 3
Enter 32bit IP Address in dotted decimal notation(***.***.***.***) : 223.196.6.2
53

This addrress belongs to class C
Network portion is :223.196.6.
Host portion is : .253

Select from one of the choice

:1)Convert from 32 bit binary ip form to decimal ip form
2)Convert from dotted decimal ip form to 32bit binary ip form
3)convert a dotted decimal ip form to its class and display the network and host
 portions seprately
```

```
4)Convert from Ip/CIDR slash notation to dotted decimal ip form
5)Convert from IP/subnet dotted ip form to CIDR slash form
6)Quit the program

Enter your choice : 4
Enter IP address in CIDR notation (***.***.***.***/**) :192.168.18.128/25
IP/Netmask in dotted decimal ip form :192.168.18.128./255.255.255.128

Select from one of the choice

:1)Convert from 32 bit binary ip form to decimal ip form
2)Convert from dotted decimal ip form to 32bit binary ip form
3)convert a dotted decimal ip form to its class and display the network and host
 portions seprately
```

```
4)Convert from Ip/CIDR slash notation to dotted decimal ip form
5)Convert from IP/subnet dotted ip form to CIDR slash form
6)Quit the program

Enter your choice : 7
I dont knoe the option 7.
Try Again

Select from one of the choice

:1)Convert from 32 bit binary ip form to decimal ip form
2)Convert from dotted decimal ip form to 32bit binary ip form
3)convert a dotted decimal ip form to its class and display the network and host
 portions seprately
4)Convert from Ip/CIDR slash notation to dotted decimal ip form
5)Convert from IP/subnet dotted ip form to CIDR slash form
6)Quit the program
```

Conclusion: - we implement conversion from

i) 32 bit binary IP address to dotted decimal IP address

ii) A dotted decimal IP form to 32 bit binary IP form

iii) A dotted decimal IP form ti its class and display the network and host portions

iv) From IP notation to dotted decimal IP form

v) From IP subnet dotted IP form to CIDR

Experiment No.2

Distance Vector Routing

Aim:- Obtain Routing table at each node using distance vector routing algorithm for a given subnet.

Objective: Student should be able to understand and implement the program of unicasting.

Hardware Requirement: Intel based desktop pc: RAM 512 MB

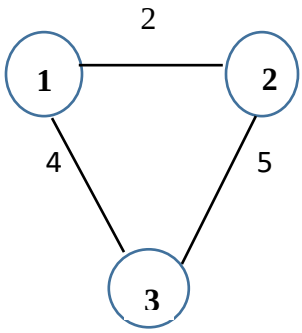
Software Requirement: Turbo C/Borland C

Theory:

Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which considers some other factors like bandwidth and other metrics to reach a destination. Distance Vector routing algorithms are not preferable for complex networks and take longer to coverage.

Node Cost

1	0
2	2
3	4



Node

1	2
2	0
3	5

Node Cost

1	4
2	5
3	0

Routing Table

Algorithm :

Begin
Step 1 : Create struct node unsigned dist[20],,unsigned from[20],rt[10]
Step 2 : Initialize intdmat[20][20],n.i.j.k.count=0
Step 3 : Write “the number of nodes “
Step 4 : Read the number of nodes “n”
Step 5 : Write “The cost metrics : “
Step 6 : initialize i=0
Step 7 : repeat until i<n
Step 8 : increment i
Step 9 : initialize j=0
Step 10 : repeat step (10-16)until j<n
Step 11 : increment j
Step 12 : read dmat[i][j]
Step 13 : initialize dmat[i][i]=0
Step 14 : initialize rt[i].dist[j]=dmat[i][j]
Step 15 : initialize rt[i].from[j]=j
Step 16 : end
Step 17 : start do loop Step (17-33) until
Step 18 : initialize count =0;
Step 19 : initialize i=0
Step 20 : repeat until i<n
Step 21 : increment i
Step 22: initialize j=0
Step 23 : repeat until j<n
Step 24 : increment j
Step 15 : initialize k=0
Step 26 : repeat until k<n
Step 27 : increment k
Step 28 : if repeat Step (28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]
Step 29 : initialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]
Step 30 : initialize rt[i].from[j]=k;
Step 31 : increment count
Step 32 : end if
Step 33 : end do statement
Step 34 : while (count!=0)
Step 35 : initialize i=0
Step 36 : repeat steps (36-44) until i<n
Step 37 : increment i
Step 38 : write ‘State values for router ‘,i+1
Step 39 : initialize j=0
Step 40 : repeat steps (40-43) until j<n
Step 41 : increment j
Step 42 : Write ‘node %d via %d distance % ‘,j+1,rt[i].from[j]+1,rt[i].dist[j]
Step 43 : end
Step 44 : end

End

Program:-

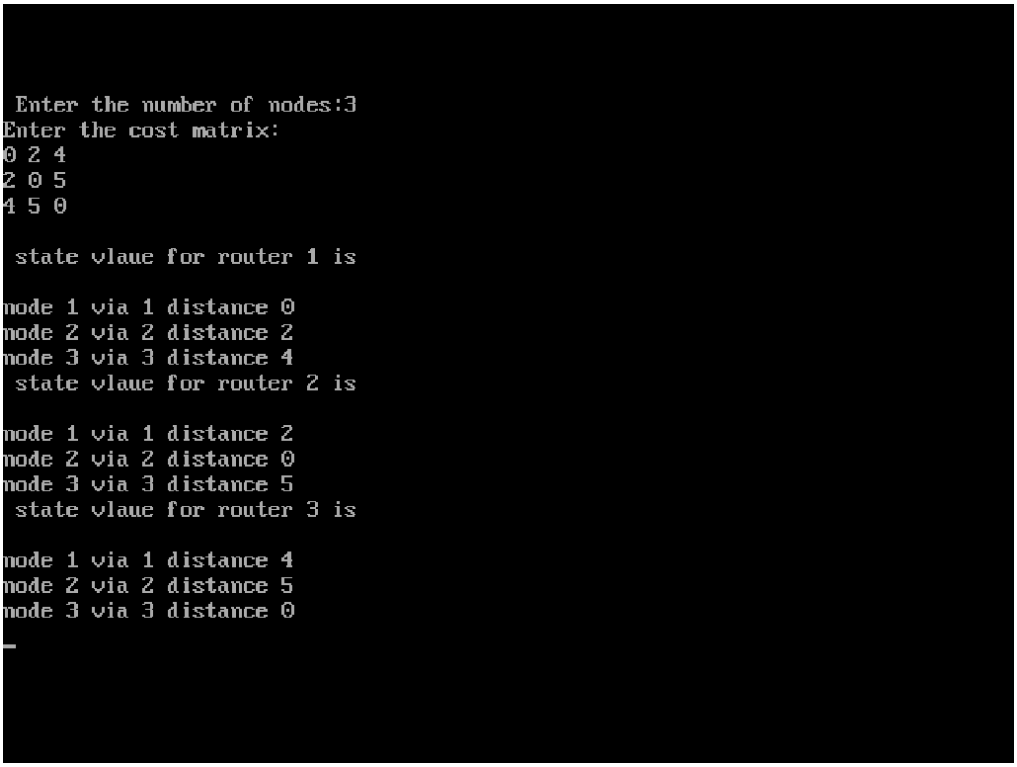
```
#include<stdio.h>
#include<conio.h>
struct node
{
unsigneddist[20];
unsigned from[20];
}
rt[10];
int main()
{
intdmat[20][20];
intn,i,j,k,count=0;
clrscr();
printf("\n Enter the number of nodes:");
scanf("%d", &n);
printf("Enter the cost matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
 {
scanf("%d",&dmat[i][j]);
dmat[i][j]=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 state value for router %d is \n",i+1);
for(j=0;j<n;j++)
```

```
 {
printf("\n node %d via %d Distance %d", j+1);//,rt[i].from[j]+1,rt[i].dist[j]);
 }
 }
printf("\n");

getch();
return 0;
}
```

Observations:-



Conclusion: - Hence we studied and implement program evaluate distance vector routing algorithm

Experiment No.3
Border gateway protocol

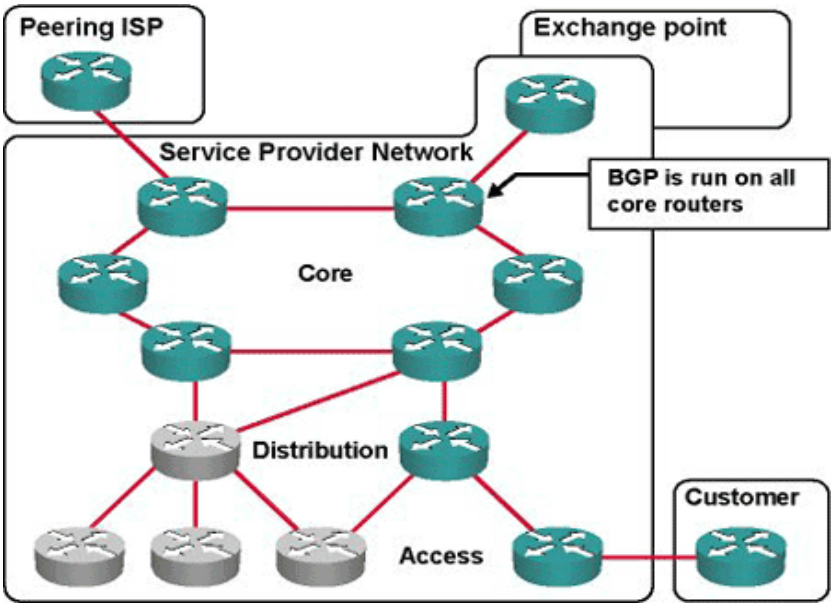
Aim:- To simulate the implementation of routing using protocol border getway

Objective: Student should be able to understand and implement the program of BGP

Hardware Requirement: Intel based desktop pc: RAM 512 MB

Software Requirement: Turbo C/Borland C

Theory: The Border Gateway Protocol (BGP) is the protocol used throughout the Internet to exchange routing information between networks. It is the language spoken by routers on the Internet to determine how packets can be sent from one router to another to reach their final destination. BGP has worked extremely well and continues to the be protocol that makes the Internet work. The challenge with BGP is that the protocol does not directly include security mechanisms and is based largely on trust between network operators that they will secure their systems correctly and not send incorrect data. Mistakes happen, though, and problems could arise if malicious attackers were to try to affect the routing tables used by BGP.



Algorithm:

1]Read the numberof node n.
2]Read the cost matrix for the path from each node to another nade.
3]initialize source to 1 and include 1.
4]compute D of anode which is the distance from source to that corresponding node.

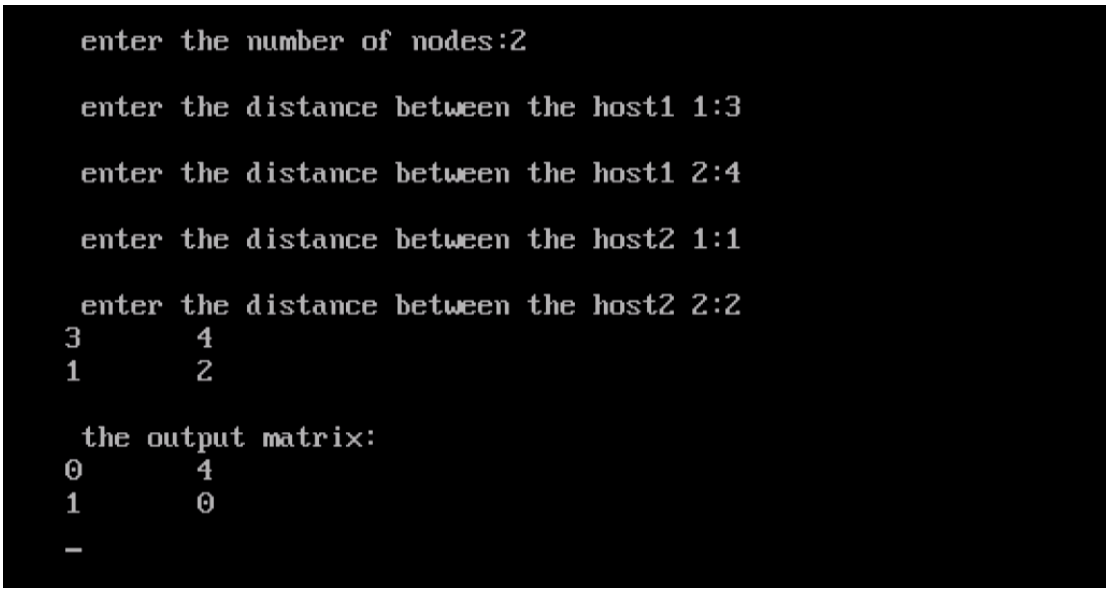
5]repeat step 6 to step 8 for n-1 node.
6]choose the node that has not been include whose distance is minimum and include that node.
7]For every other node not include compare the distance directly from the source with the distance to reach the node using the include node.
8]Take the minimum values as the new distance.
9]Print all the nodes with shortest path cost from source node.

Program :

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n;
int i,j,k;
int a[10][10],b[10][10];
printf("\n enter the number of nodes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("\n enter the distance between the host%d %d:",i+1,j+1);
scanf("%d",&a[i][j]);
}
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}for(k=0;k<n;k++)
{
for(j=0;j<n;j++)
{
for(j=0;j<n;j++)
{
if(a[i][j]>a[i][k]+a[k][j])
{
a[i][j]=a[i][k]+a[k][i];
}
}
}
}
for(i=0;i<n;i++)
```

```
{
for(j=0;j<n;j++)
{
b[i][j]=a[i][j];
if(i==j)
{
b[i][j]=0;
}
}
}
printf("\n the output matrix:\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
getch();
return 0;
}
```

Observations:-



Conclusion: - Hence we studied and implement program evaluate Border Gateway Protocol.

Experiment No.4
Open Shortest Path First

Aim:- To simulate the open shortest path first routing protocol based on cost assigned to path

Objective: Student should be able to understand and implement the program of OSPF.

Hardware Requirement: Intel based desktop pc: RAM 51

Software Requirement: Turbo C/Borland C

Theory:The OSPF (Open Shortest Path First) protocol is one of a family of IP Routing protocols, and is an Interior Gateway Protocol (IGP) for the Internet, used to distribute IP routing information throughout a single Autonomous System (AS) in an IP network.

The OSPF protocol is a link-state routing protocol, which means that the routers exchange topology information with their nearest neighbors. The topology information is flooded throughout the AS, so that every router within the AS has a complete picture of the topology of the AS. This picture is then used to calculate end-to-end paths through the AS, normally using a variant of the Dijkstra algorithm. Therefore, in a link-state routing protocol, the next hop address to which data is forwarded is determined by choosing the best end-to-end path to the eventual destination.

The main advantage of a link state routing protocol like OSPF is that the complete knowledge of topology allows routers to calculate routes that satisfy particular criteria. This can be useful for traffic engineering purposes, where routes can be constrained to meet particular quality of service requirements. The main disadvantage of a link state routing protocol is that it does not scale well as more routers are added to the routing domain. Increasing the number of routers increases the size and frequency of the topology updates, and also the length of time it takes to calculate end-to-end routes. This lack of scalability means that a link state routing protocol is unsuitable for routing across the Internet at large, which is the reason why IGPs only route traffic within a single AS.

Each OSPF router distributes information about its local state (usable interfaces and reachable neighbors, and the cost of using each interface) to other routers using a Link State Advertisement (LSA) message. Each router uses the received messages to build up an identical database that describes the topology of the AS.

From this database, each router calculates its own routing table using a Shortest Path First (SPF) or Dijkstra algorithm. This routing table contains all the destinations the routing protocol knows about, associated with a next hop IP address and outgoing interface.

- The protocol recalculates routes when network topology changes, using the Dijkstra algorithm, and minimises the routing protocol traffic that it generates.

- It provides support for multiple paths of equal cost.
- It provides a multi-level hierarchy (two-level for OSPF) called "area routing," so that information about the topology within a defined area of the AS is hidden from routers outside this area. This enables an additional level of routing protection and a reduction in routing protocol traffic.
- All protocol exchanges can be authenticated so that only trusted routers can join in the routing exchanges for the AS.

Algorithm:

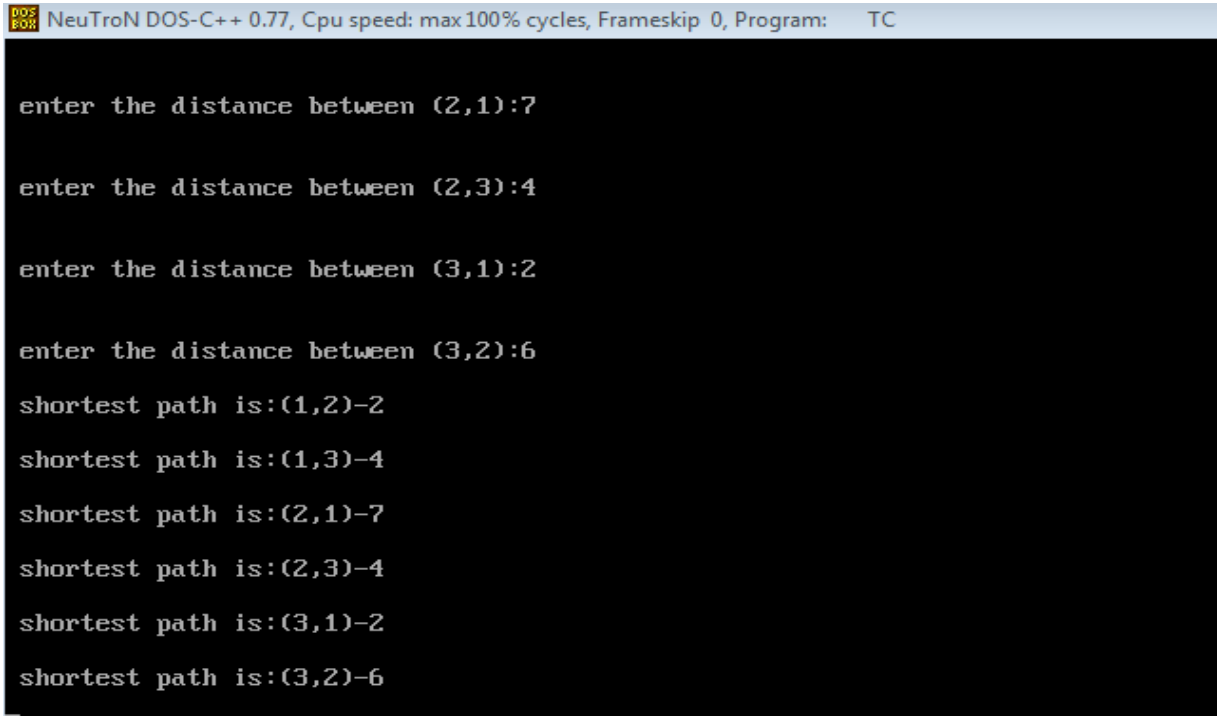
1.Read the number of nodes n
2. Read the cost matrix for the path from each node to another node.
3.Initialize SOURCE to 1 and include 1.
4. compute d of a node which is the distance from source to that corresponding node.
5.Repeat step 6 to step 8 for n-1 node.
6. choose the node that has not been included whose distance is minimum and include that node.
7.for every other node not included compare the distance directly from the source with the distance to reach the node using the newly include node.
8. take the minimum value as the new distance.
9.print all the node with shortest path cost from source code.

Program:

```
#include<stdio.h>
#include<conio.h>
int a[5][5],n,i,j;
void main()
{
voidgetdata();
void shortest();
void display();
clrscr();
printf("\n \n program to find shortest path between two nodes\n");
getdata();
display();
}
voidgetdata()
{
clrscr();
printf("\n\n enter the number of host in the graph\n");
scanf("%d",&n);
printf("\n\n if there is no direct path\n");
printf("\n\n assign the highest distance value 1000\n");
for(i=0;i<n;i++)
{
a[i][i]=0;
for(j=0;j<n;j++)
{
```

```
if(i!=j)
{
printf("\n\n enter the distance between (%d,%d):",i+1,j+1);
scanf("%d",&a[i][j]);
if(a[i][j]==0)
a[i][j]=1000;
}
}
}
}
void shortest()
{
inti,j,k;
for(k=0;k,n;k++)
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(a[i][k]+a[k][j]<a[i][j])
a[i][j]=a[i][k]+a[k][j];
}
}
void display()
{
inti,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(i!=j)
{
printf("\n shortest path is:(%d,%d)-%d\n",i+1,j+1,a[i][j]);
}
getch();
}
```

Observations:-



Conclusion: - Hence we studied and implement program evaluate Open Shortest Path First.

Experiment No.5
Dijkstra algorithm

Aim:- Implement Dijkstra algorithm to compute the shortest path through a given graph.

Objective: Student should be able to understand and implement the program of Dijkstra algorithm.

Hardware Requirement: Intel based desktop pc: RAM 51

Software Requirement: Turbo C/Borland C

Theory: G = (V,E) where V is a set of vertices and E is a set of edges. Dijkstra's algorithm keeps two sets of vertices: S the set of vertices whose shortest paths from the source have already been determined and V-S the remaining vertices. The other data structures needed are: d array of best estimates of shortest path to each vertex pi an array of predecessors for each vertex The basic mode of operation is: Initialize d and pi, Set S to empty, While there are still vertices in V-S, Sort the vertices in V-S according to the current best estimate of their distance from the source, Add u, the closest vertex in V-S, to S, Relax all the vertices still in V-S connected to u Relaxation The relaxation process updates the costs of all the vertices, v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including (u,v) in the path to v.

Algorithm:

Begin
Step1:Declare array path[5][5],min,a[5][5],index,t[5];
Step2:Declare and initialize st=1,ed=5
Step3:Declare variables I,j,stp,p,edp
Step4:print"enter the cost"
Step5:i=1
Step6:repeat step(7to11) until (j<=5)
Step7:j=1
Step8:repeat step (9to10) until(j,=5)
Step9:read a[i][j]
Step10:increment j
Step11:increment i
Step12:print "enter the path"
Step13:read p
Step14:print "enter possible path"
Step15:i=1
Step16:repeat step(7 to 21) until (j<=p)
Step17:j=1
Step18:repeat step (9to 20) until (i<=5)
Step19: read path[i][j]
Step20:increment j

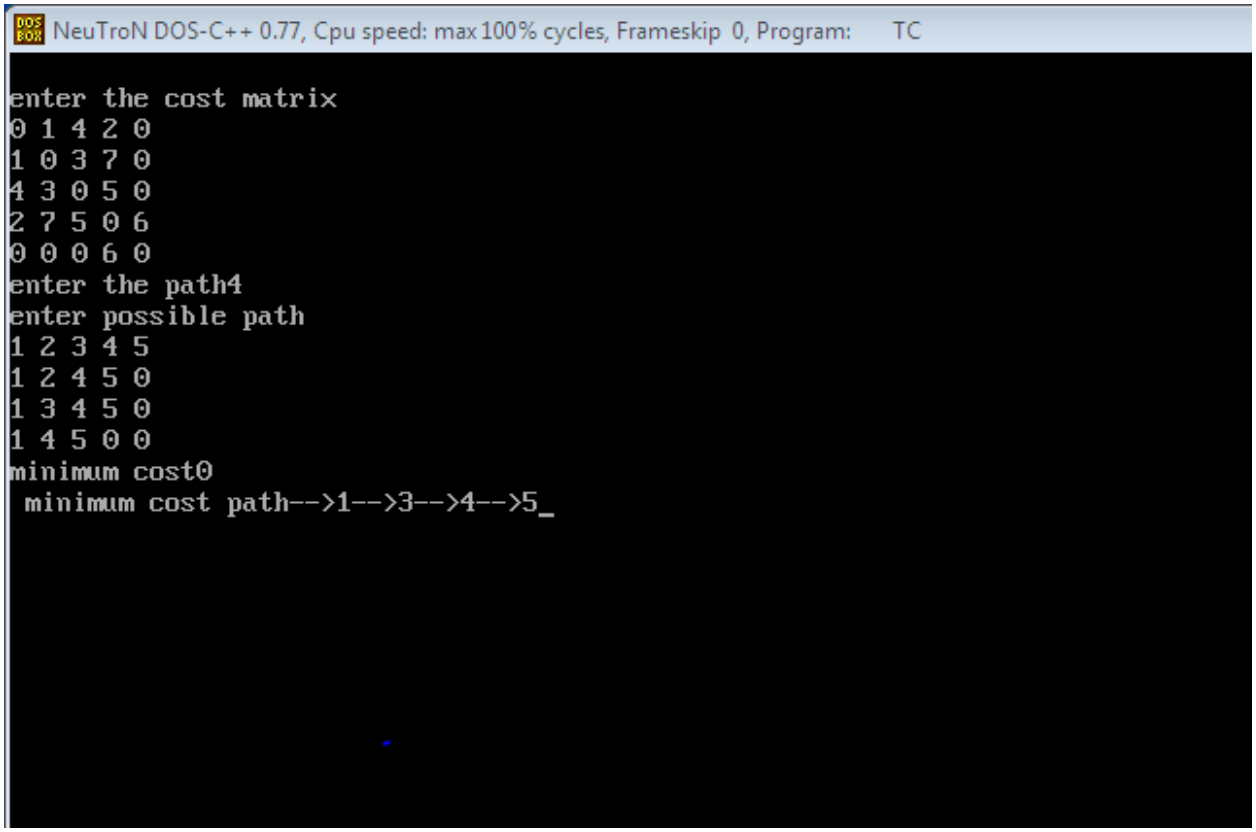
Step21:increment i
Step22:j=1
Step23:repeat step (24 to 34) until (i<=p)
Step24:t[i]=0
Step25:stp=st
Step26:j=1
Step27:repeat step (26 to 34) until (j<=5)
Step28: edp=path[i][i+1]
Step29:t[i]=[ti]+[stp][edp]
Step30:if (edp==ed) then
Step31:break;
Step32:else
Step33:stp=edp
Step34:end if
Step35: min=t[st]
Step36:index=st
Step37: repeat step (38 to 41) until (i<=p)
Step38:min>=t[i]
Step39:min=t[i]
Step40:index=i
Step41:end if
Step42:print " minimum cost" min
Step43:print "minimum cost path"
Step44:repeat step (45 to 48) until (i<=5)
Step45:print "path[index][i]
Step46:if(path[index][i]==ed) then
Step47:break
Step48:end if
End

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
clrscr();
printf("\nenter the cost matrix");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d",&a[i][j]);
printf("enter the path");
scanf("%d",&p);
printf("enter possible path");
for(i=1;i<=p;i++)
for(j=1;j<=5;j++)
```

```
scanf("%d",&path[i][j]);
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=5;j++)
{
edp=path[i][i+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
elsestp=edp;
}
}
min=t[st];
index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];
index=i;
}
}
printf("minimum cost%d",min);
printf("\n minimum cost path");
for(i=1;i<=5;i++)
{
printf("-->%d",path[index][i]);
if(path[index][i]==ed)
break;
}
getch();
}
```

Observations:-



Conclusion: - Hence we studied and implement program evaluate Dijkstra algorithm to compute the shortest path through a given graph.

Experiment No.6
Sliding window protocol

Aim:- Simulation of sliding window protocol.

Objective: Student should be able to understand and implement the program of sliding window protocol.

Hardware Requirement: Intel based desktop pc: RAM 51

Software Requirement: Turbo C/Borland C

Theory:

A **sliding window protocol** is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol (TCP).Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit on the size of the sequence number that can be required.

A transmitter that does not hear an acknowledgment cannot know if the receiver actually received the packet; it may be that it was lost or damaged in transmission. If error detection reveals corruption, the packet will be ignored by the receiver and no acknowledgement will be sent. Similarly, the receiver is usually uncertain about whether its acknowledgements are being received. It may be that an acknowledgment was sent, but was lost or corrupted in the transmission medium. In this case, the receiver must acknowledge the retransmission to prevent the data being continually resent, but must otherwise ignore it.

Program:

SENDER:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
int main()
```

```
{
int sock,bytes_received,connected,true=1,i=1,s,f=0,sin_size;
char send_data[1024],data[1024],c,fr[30]=" ";
struct sockaddr_in server_addr,client_addr;
if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
{
perror("Socket not created");
exite(1);
}
if(setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&true,sizeof(int))==-1)
{
perror("sockopt");
exit(1);
}
server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(17000);
server_addr.sin_addr.s_addr=INADDR_ANY;
if(bind(sock,(struct sockaddr*)&server_addr,sizeof(struct sockaddr))==-1)
{
perror("unable to bind"
);
exit(1);
}
if(listen(sock,5)==-1)
{
perror("listen");
exit(1);
}
fflush(stdout);
sin_size=sizeof(struct sockaddr_in);
connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);
while(strcmp(fr,"exit")!=0)
{
printf("Enter data frame %d:(Enter exit for end):",i);
scanf("%s",fr);
send(connected,fr,strlen(fr),0);
recv(sock,data,1024,0);
if(strlen(data)!=0)
printf("I got an acknowlgment:%s\n",data);
fflush(stdout);
i++;
}
close(sock);
return(0);
}
```

RECEIVER:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
int main()
{
int sock,bytes_received,i=1;
char receive[30];
struct hostent*host;
struct sockaddr_in server_addr;
host=gethostbyname("127.0.0.1");
if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
{
perror("socket not created");
exit(1);
}
printf("socket created");
server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(17000);
server_addr.sin_addr=*((struct in_addr*)host->h_addr);
bzero(&(serve_addr.sin_zero),8);
if(connect(sock,(struct sockaddr*)&server_addr,sizeof(struct_sockadder))==-1)
{
prror("connect");
exit(1);
}
while(1)
{
bytes_receiver=recv(sock,receive,20,0);
receive[bytes_received]='\0';
if(strcmp(receive,"exit")==0||strcmp(receive,"exit")==0)
{
close(sock);
break;
}
else
{
if(strlen(receive)<10)
{
```

```
printf("\n frame %d data %s received \n",i,receive);
send(0,receive,strlen(receive),0);
}
else
{
send(0,"negative",10,0);
}
i++;
}
}
close(sock);
return(0);
}
```

Observations:-

SENDER:

```
$ CC sender.c
$ ./a.out
        Enter Data Frame 1:(Enter exit for end):teju
        Enter Data Frame 2:(Enter exit for end):cse
        Enter Data Frame 3:(Enter exit for end):exit
$
```

RECEIVER:

```
$ CC receiver.c
$ ./a.out
          Socket created
           Frame 1 data teju received
            Frame 2 data cse received
$
```

Conclusion: - Hence we studied and implement program for sliding window protocol.

Experiment No.7
DHCP

Aim:- Study of DHCP.

Objective: Student should be able to understand Dynamic Host Configuration Protocol.

Theory:

What is DHCP?

Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration information such as the subnet mask and default gateway. RFCs 2131 and 2132 define DHCP as an Internet Engineering Task Force (IETF) standard based on Bootstrap Protocol (BOOTP), a protocol with which DHCP shares many implementation details. DHCP allows hosts to obtain necessary TCP/IP configuration information from a DHCP server.
The Microsoft Windows Server 2003 operating system includes a DHCP Server service, which is an optional networking component. All Windows-based clients include the DHCP client as part of TCP/IP, including Windows Server 2003, Microsoft Windows XP, Windows 2000, Windows NT 4.0, Windows Millennium Edition (Windows Me), and Windows 98.

Benefits of DHCP

In Windows Server 2003, the DHCP Server service provides the following benefits:

- **Reliable IP address configuration.** DHCP minimizes configuration errors caused by manual IP address configuration, such as typographical errors, or address conflicts caused by the assignment of an IP address to more than one computer at the same time.
- **Reduced network administration.** DHCP includes the following features to reduce network administration:
    - Centralized and automated TCP/IP configuration.
    - The ability to define TCP/IP configurations from a central location.
    - The ability to assign a full range of additional TCP/IP configuration values by means of DHCP options.
    - The efficient handling of IP address changes for clients that must be updated frequently, such as those for portable computers that move to different locations on a wireless network.
    - The forwarding of initial DHCP messages by using a DHCP relay agent, thus eliminating the need to have a DHCP server on every subnet.

Why use DHCP

Every device on a TCP/IP-based network must have a unique unicast IP address to access the network and its resources. Without DHCP, IP addresses must be configured manually for new

computers or computers that are moved from one subnet to another, and manually reclaimed for computers that are removed from the network.
DHCP enables this entire process to be automated and managed centrally. The DHCP server maintains a pool of IP addresses and leases an address to any DHCP-enabled client when it starts up on the network. Because the IP addresses are dynamic (leased) rather than static (permanently assigned), addresses no longer in use are automatically returned to the pool for reallocation.
The network administrator establishes DHCP servers that maintain TCP/IP configuration information and provide address configuration to DHCP-enabled clients in the form of a lease offer. The DHCP server stores the configuration information in a database, which includes:

- Valid TCP/IP configuration parameters for all clients on the network.
- Valid IP addresses, maintained in a pool for assignment to clients, as well as excluded addresses.
- Reserved IP addresses associated with particular DHCP clients. This allows consistent assignment of a single IP address to a single DHCP client.
- The lease duration, or the length of time for which the IP address can be used before a lease renewal is required.

A DHCP-enabled client, upon accepting a lease offer, receives:

- A valid IP address for the subnet to which it is connecting.
- Requested DHCP options, which are additional parameters that a DHCP server is configured to assign to clients. Some examples of DHCP options are Router (default gateway), DNS Servers, and DNS Domain Name. Terms and Definitions.

The following table lists common terms associated with DHCP.
DHCP Terms and Definitions

Term	Definition
DHCP server	A computer running the DHCP Server service that holds information about available IP addresses and related configuration information as defined by the DHCP administrator and responds to requests from DHCP clients.
DHCP client	A computer that gets its IP configuration information by using DHCP.
Scope	A range of IP addresses that are available to be leased to DHCP clients by the DHCP Server service.

Subnetting	The process of partitioning a single TCP/IP network into a number of separate network segments called subnets.
DHCP option	Configuration parameters that a DHCP server assigns to clients. Most DHCP options are predefined, based on optional parameters defined in Request for Comments (RFC) 2132, although extended options can be added by vendors or users.
Option class	An additional set of options that can be provided to a DHCP client based on its computer class membership. The administrator can use option classes to submanage option values provided to DHCP clients. There are two types of options classes supported by a DHCP server running Windows Server 2003: vendor classes and user classes.
Lease	The length of time for which a DHCP client can use a DHCP-assigned IP address configuration.
Reservation	A specific IP address within a scope permanently set aside for leased use by a specific DHCP client. Client reservations are made in the DHCP database using the DHCP snap-in and are based on a unique client device identifier for each reserved entry.
Exclusion/exclusion range	One or more IP addresses within a DHCP scope that are not allocated by the DHCP Server service. Exclusions ensure that the specified IP addresses will not be offered to clients by the DHCP server as part of the general address pool.
DHCP relay agent	Either a host or an IP router that listens for DHCP client messages being broadcast on a subnet and then forwards those DHCP messages directly to a configured DHCP server. The DHCP server sends DHCP response messages directly back to the DHCP relay agent, which then forwards them to the DHCP client. The DHCP administrator uses DHCP relay agents to centralize DHCP servers, avoiding the need for a DHCP server on each subnet. Also referred to as a *BOOTP relay agent*.
Unauthorized DHCP server	A DHCP server that has not explicitly been authorized. Sometimes referred to as a *rogue DHCP server*. In a Windows Server 2003 domain environment, the DHCP Server service on an unauthorized server running Windows Server 2003 fails to initialize. The administrator must explicitly authorize all DHCP servers running Windows Server 2003 that operate in an Active Directory service domain environment. At initialization time, the DHCP Server service in Windows Server 2003 checks for authorization and stops itself if the server detects that it is in a

	domain environment and the server has not been explicitly authorized.
Automatic Private IP Addressing (APIPA)	A TCP/IP feature in Windows XP and Windows Server 2003 that automatically configures a unique IP address from the range 169.254.0.1 through 169.254.255.254 with a subnet mask of 255.255.0.0 when the TCP/IP protocol is configured for automatic addressing, the **Automatic private IP address** alternate configuration setting is selected, and a DHCP server is not available. The APIPA range of IP addresses is reserved by the Internet Assigned Numbers Authority (IANA) for use on a single subnet, and IP addresses within this range are not used on the Internet.
Superscope	A configuration that allows a DHCP server to provide leases from more than one scope to clients on a single physical network segment.
Multicast IP addresses	Multicast IP addresses allow multiple clients to receive data that is sent to a single IP address, enabling point-to-multipoint communication. This type of transmission is often used for streaming media transmissions, such as video conferencing.
Multicast Scope	A range of multicast IP addresses that can be assigned to DHCP clients. A multicast scope allows dynamic allocation of multicast IP addresses for use on the network by using the MADCAP protocol, as defined in RFC 2730.
BOOTP	An older protocol with similar functionality; DHCP is based on BOOTP. BOOTP is an established protocol standard used for configuring IP hosts. BOOTP was originally designed to enable boot configuration for diskless workstations. Most DHCP servers, including those running Windows Server 2003, can be configured to respond to both BOOTP requests and DHCP requests.

- DHCP Architecture
- DHCP Protocols
- DHCP Processes and Interactions

DHCP Architecture

The DHCP architecture consists of DHCP clients, DHCP servers, and DHCP relay agents on a network. The clients interact with servers using DHCP messages in a DHCP conversation to obtain and renew IP address leases.

DHCP Client Functionality

A **DHCP client** is any network-enabled device that supports the ability to communicate with a DHCP server in compliance with RFC 2131, for the purpose of obtaining dynamic leased IP configuration and related optional information.
DHCP provides support for client computers running any of the following Microsoft operating systems:

- Windows NT version 4.0
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows 98
- Windows Millennium Edition

Automatic IP Configuration
DHCP supports Automatic Private IP Addressing (APIPA), which enables computers running Windows 2000, Windows XP, and Windows Server 2003 to configure an IP address and subnet mask if a DHCP server is unavailable at system startup and the **Automatic private IP address** Alternate Configuration setting is selected. This feature is useful for clients on small private networks, such as a small-business office or a home office.
The DHCP Client service on a computer running Windows XP and Windows Server 2003 uses the following process to auto-configure the client:

1. The DHCP client attempts to locate a DHCP server and obtain an IP address and configuration.
2. If a DHCP server cannot be found or does not respond after one minute, the DHCP client checks the settings on the **Alternate Configuration** tab of the properties of the TCP/IP protocol.

   If **Automatic private IP address** is selected, the DHCP client auto-configures its IP address and subnet mask by using a selected address from the Microsoft-reserved Class B network, 169.254.0.0, with the subnet mask 255.255.0.0. The DHCP client tests for an address conflict to ensure that the IP address is not in use on the network. If a conflict is found, the client selects another IP address. The client retries auto-configuration up to 10 times.

   If **User Configured** is selected, the DHCP client configures a static IP address configuration. The DHCP client tests for an address conflict to ensure that the IP address is not already in use on the network. If a conflict is found, the DHCP client indicates the error condition to the user.
3. When the DHCP client succeeds in self-selecting an address, it configures its network interface with the IP address. The client then continues to check for a DHCP server in the background every five minutes. If a DHCP server responds, the DHCP client abandons its

self-selected IP address and uses the address offered by the DHCP server (and any other DHCP option information that the server provides) to update its IP configuration settings.

If the DHCP client obtained a lease from a DHCP server on a previous occasion, and the lease is still valid (not expired) at system startup, the client tries to renew its lease. If, during the renewal attempt, the client fails to locate any DHCP server, it attempts to ping the default gateway listed in the lease, and proceeds in one of the following ways:

- If the ping is successful, the DHCP client assumes that it is still located on the same network where it obtained its current lease, and continues to use the lease as long as the lease is still valid. By default the client then attempts, in the background, to renew its lease when 50 percent of its assigned lease time has expired.

- If the ping fails, the DHCP client assumes that it has been moved to a network where a DHCP server is not available. The client then auto-configures its IP address by using the settings on the **Alternate Configuration** tab. When the client is auto-configured, it attempts to locate a DHCP server and obtain a lease every five minutes.

Interactions between Client and Server

DHCP servers and DHCP clients communicate through a series of DHCP messages. To obtain a lease, the DHCP client initiates a conversation with a DHCP server using a series of these DHCP messages.

DHCP Messages

The following list includes the eight types of messages that can be sent between DHCP clients and servers. For more information about the structure and specifics of each of these packets, see "DHCP Message Format" later in this section.

DHCP Discover

Broadcast by a DHCP client when it first attempts to connect to the network. The DHCP Discover message requests IP address information from a DHCP server.

DHCP Offer

Broadcast by each DHCP server that receives the client DHCP Discover message and has an IP address configuration to offer to the client. The DHCP Offer message contains an unleased IP address and additional TCP/IP configuration information, such as the subnet mask and default gateway. More than one DHCP server can respond with a DHCP Offer message. The client accepts the best offer, which for a Windows DHCP client is the first DHCP Offer message that it receives.

DHCP Request

Broadcast by a DHCP client after it selects a DHCP Offer. The DHCP Request message contains the IP address from the DHCP Offer that it selected. If the client is renewing or rebinding to a previous lease, this packet might be unicast directly to the server.

DHCP Ack

Broadcast by a DHCP server to a DHCP client acknowledging the DHCP Request message. At this time, the server also forwards any options. Upon receipt of the DHC PAck, the client can use the leased IP address to participate in the TCP/IP network and complete its system startup. This message is typically broadcast, because the DHCP client does not officially have an IP address that it can use at this point. If the DHCPAck is in response to a DHCPInform, then the message is unicast directly to the host that sent the DHCPInform message.

DHCPNack

Broadcast by a DHCP server to a DHCP client denying the client's DHCPRequest message. This might occur if the requested address is incorrect because the client moved to a new subnet or because the DHCP client's lease has expired and cannot be renewed.

DHCPDecline

Broadcast by a DHCP client to a DHCP server, informing the server that the offered IP address is declined because it appears to be in use by another computer.

DHCPRelease

Sent by a DHCP client to a DHCP server, relinquishing an IP address and canceling the remaining lease. This is unicast to the server that provided the lease.

DHCPInform

Sent from a DHCP client to a DHCP server, asking only for additional local configuration parameters; the client already has a configured IP address. This message type is also used by DHCP servers running Windows Server 2003 to detect unauthorized DHCP servers.

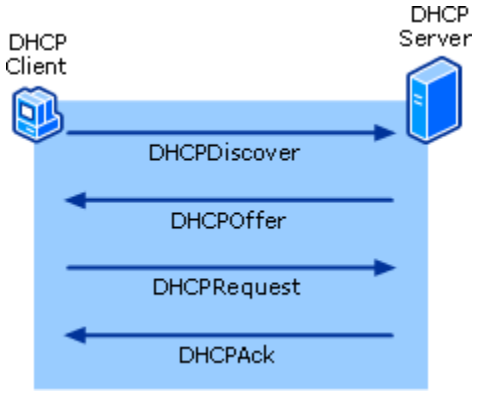
DHCP Lease Process

A DHCP-enabled client obtains a lease for an IP address from a DHCP server. Before the lease expires, the DHCP client must renew the lease or obtain a new lease. Leases are retained in the DHCP server database for a period of time after expiration. By default, this grace period is four hours and cleanup occurs once an hour for a DHCP server running Windows Server 2003. This protects a clients lease in case the client and server are in different time zones, the internal clocks of the client and server computers are not synchronized, or the client is off the network when the lease expires.

Obtaining a New Lease

A DHCP client initiates a conversation with a DHCP server when it is seeking a new lease, renewing a lease, rebinding, or restarting. The DHCP conversation consists of a series of DHCP messages passed between the DHCP client and DHCP servers. The following figure shows an overview of this process when the DHCP server and DHCP client are on the same subnet.

DHCP Lease Process Overview



1. The DHCP client requests an IP address by broadcasting a DHCPDiscover message to the local subnet.

2. The client is offered an address when a DHCP server responds with a DHCPOffer message containing an IP address and configuration information for lease to the client. If no DHCP server responds to the client request, the client sends DHCPDiscover messages at intervals of 0, 4, 8, 16, and 32 seconds, plus a random interval of between -1 second and 1 second. If there is no response from a DHCP server after one minute, the client can proceed in one of two ways:

- If the client is using the Automatic Private IP Addressing (APIPA) alternate configuration, the client self-configures an IP address for its interface.
- If the client does not support alternate configuration, such as APIPA, or if IP auto-configuration has been disabled, the client network initialization fails.

In both cases, the client begins a new cycle of DHCPDiscover messages in the background every five minutes, using the same intervals as before (0, 4, 8, 16, and 32 seconds), until it receives a DHCPOffer message from a DHCP server.

3. The client indicates acceptance of the offer by selecting the offered address and broadcasting a DHCPRequest message in response.
4. The client is assigned the address and the DHCP server broadcasts a DHCPAck message in response, finalizing the terms of the lease.

Conclusion: - Hence we studied DHCP in detail.

Experiment No.8
DNS

Aim:- Study of DNS.

Objective: Student should be able to understand Domain Name System.

Theory:

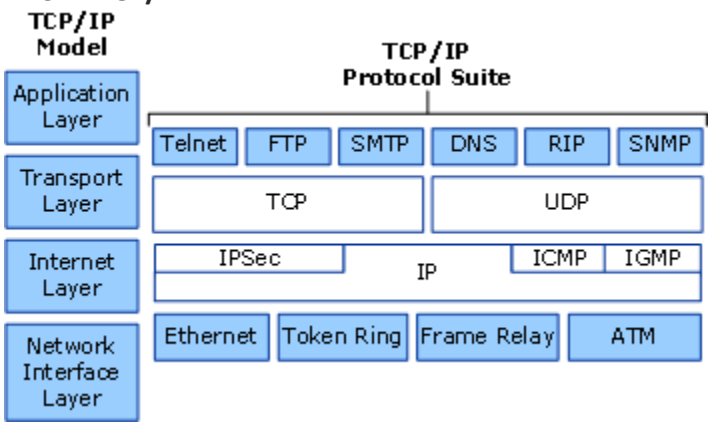
- Technologies That Use DNS
- Related Information

Domain Name System (DNS) is one of the industry-standard suite of protocols that comprise TCP/IP. Microsoft Windows Server 2003. DNS is implemented using two software components: the DNS server and the DNS client (or resolver). Both components are run as background service applications.

Network resources are identified by numeric IP addresses, but these IP addresses are difficult for network users to remember. The DNS database contains records that map user-friendly alphanumeric names for network resources to the IP address used by those resources for communication. In this way, DNS acts as a mnemonic device, making network resources easier to remember for network users.

The Windows Server 2003 DNS Server and Client services use the DNS protocol that is included in the TCP/IP protocol suite. DNS is part of the application layer of the TCP/IP reference model.

DNS in TCP/IP



Technologies That Use DNS

DNS and Active Directory

Windows Server 2003 Active Directory directory service uses DNS as its domain controller location mechanism. When any of the principal Active Directory operations is performed, such as authentication, updating, or searching, Windows Server 2003 computers use DNS to locate Active Directory domain controllers and these domain controllers use DNS to locate each other. For example, when a network user with an Active Directory user account logs in to an Active Directory domain, the user's computer uses DNS to locate a domain controller for the Active Directory domain to which the user wants to log in.

DNS and WINS

The earlier method of name resolution for a Windows network was Windows Internet Name Service (WINS). DNS is different than WINS in that DNS is a hierarchical namespace and WINS is a flat namespace. Down-level clients and applications that rely on NetBIOS names continue to use WINS for name resolution. Since Windows Server 2003 DNS is WINS-aware, a combination of both DNS and WINS can be used in a mixed environment to achieve maximum efficiency in locating various network services and resources.

DNS and DHCP

For Windows Server 2003 DNS, the DHCP service provides default support to register and update information for legacy DHCP clients in DNS zones. Legacy clients typically include other Microsoft TCP/IP client computers that were released prior to Windows 2000. The Windows Server 2003 DNS-DHCP integration enables a DHCP client that is unable to dynamically update DNS resource records directly to have this information updated in DNS forward and reverse lookup zones by the DHCP server.

Domain Name System (DNS) is the default name resolution service used in a Microsoft Windows Server 2003 network. DNS is part of the Windows Server 2003 TCP/IP protocol suite and all TCP/IP network connections are, by default, configured with the IP address of at least one DNS server in order to perform name resolution on the network. Windows Server 2003 components that require name resolution will attempt to use this DNS server before attempting to use the previous default Windows name resolution service, Windows Internet Name Service (WINS).

Typically, Windows Server 2003 DNS is deployed in support of Active Directory directory service. In this environment, DNS namespaces mirror the Active Directory forests and domains used by an organization. Network hosts and services are configured with DNS names so that they can be located in the network, and they are also configured with DNS servers that resolve the names of Active Directory domain controllers.

Windows Server 2003 DNS is also commonly deployed as a non-Active Directory, or standard, Domain Name System solution, for the purposes of hosting the Internet presence of an organization, for example.

DNS Architecture

DNS architecture is a hierarchical distributed database and an associated set of protocols that define:

- A mechanism for querying and updating the database.
- A mechanism for replicating the information in the database among servers.
- A schema of the database.

DNS originated in the early days of the Internet when the Internet was a small network established by the United States Department of Defense for research purposes. The host names of the computers in this network were managed through the use of a single HOSTS file located on a centrally administered server. Each site that needed to resolve host names on the network downloaded this file. As the number of hosts on the Internet grew, the traffic generated by the update process increased, as well as the size of the HOSTS file. The need for a new system, which would offer features such as scalability, decentralized administration, support for various data types, became more and more obvious.

DNS Domain Names

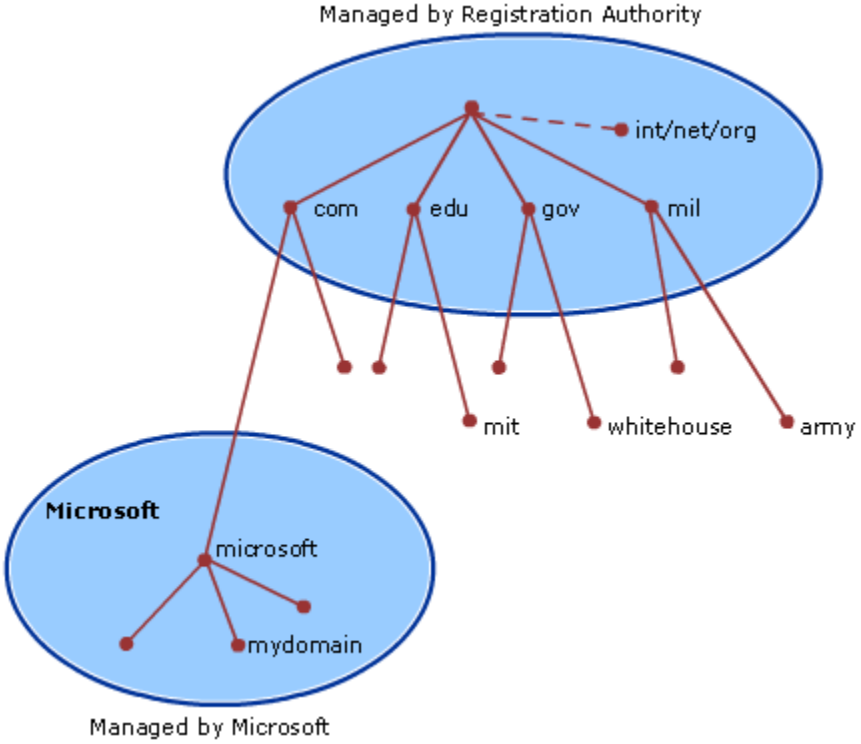
The Domain Name System is implemented as a hierarchical and distributed database containing various types of data, including host names and domain names. The names in a DNS database form a hierarchical tree structure called the domain namespace. Domain names consist of individual labels separated by dots, for example: mydomain.microsoft.com.

A Fully Qualified Domain Name (FQDN) uniquely identifies the hosts position within the DNS hierarchical tree by specifying a list of names separated by dots in the path from the referenced host to the root. The next figure shows an example of a DNS tree with a host called my domain within the microsoft.com. domain. The FQDN for the host would be mydomain.microsoft.com**.**

Understanding the DNS Domain Namespace

The DNS domain namespace, as shown in the following figure, is based on the concept of a tree of named domains. Each level of the tree can represent either a branch or a leaf of the tree. A branch is a level where more than one name is used to identify a collection of named resources. A leaf represents a single name used once at that level to indicate a specific resource.

DNS Domain Name Hierarchy



The previous figure shows how Microsoft is assigned authority by the Internet root servers for its own part of the DNS domain namespace tree on the Internet. DNS clients and servers use queries as the fundamental method of resolving names in the tree to specific types of resource information. This information is provided by DNS servers in query responses to DNS clients, who then extract the information and pass it to a requesting program for resolving the queried name. In the process of resolving a name, keep in mind that DNS servers often function as DNS clients, querying other servers in order to fully resolve a queried name.

How the DNS Domain Namespace Is Organized

Any DNS domain name used in the tree is technically a domain. Most DNS discussions, however, identify names in one of five ways, based on the level and the way a name is commonly used. For example, the DNS domain name registered to Microsoft (microsoft.com.) is known as a second-level domain. This is because the name has two parts (known as labels) that indicate it is located

two levels below the root or top of the tree. Most DNS domain names have two or more labels, each of which indicates a new level in the tree. Periods are used in names to separate labels.
The five categories used to describe DNS domain names by their function in the namespace are described in the following table, along with an example of each name type.

Types of DNS Domain Names

Name Type	Description	Example
Root domain	This is the top of the tree, representing an unnamed level; it is sometimes shown as two empty quotation marks (""), indicating a null value. When used in a DNS domain name, it is stated by a trailing period (.) to designate that the name is located at the root or highest level of the domain hierarchy. In this instance, the DNS domain name is considered to be complete and points to an exact location in the tree of names. Names stated this way are called fully qualified domain names (FQDNs).	A single period (.) or a period used at the end of a name, such as “example.microsoft.com.”
Top level domain	A name used to indicate a country/region or the type of organization using a name.	““.com”, which indicates a name registered to a business for commercial use on the Internet.
Second level domain	Variable-length names registered to an individual or organization for use on the Internet. These names are always based upon an appropriate top-level domain, depending on the type of organization or geographic location where a name is used.	““microsoft.com. ”, which is the second-level domain name registered to Microsoft by the Internet DNS domain name registrar.
Subdomain	Additional names that an organization can create that are derived from the registered second-level domain name. These include names added to grow the DNS tree of names in an organization and divide it into departments or geographic locations.	““example.microsoft.com. ”, which is a fictitious subdomain assigned by Microsoft for use in documentation example names.

Host or resource name	Names that represent a leaf in the DNS tree of names and identify a specific resource. Typically, the leftmost label of a DNS domain name identifies a specific computer on the network. For example, if a name at this level is used in a host (A) RR, it is used to look up the IP address of computer based on its host name.	““host-a.example.microsoft.com.”, where the first label (“host-a”) is

Conclusion: - Hence we studied DNS in detail.

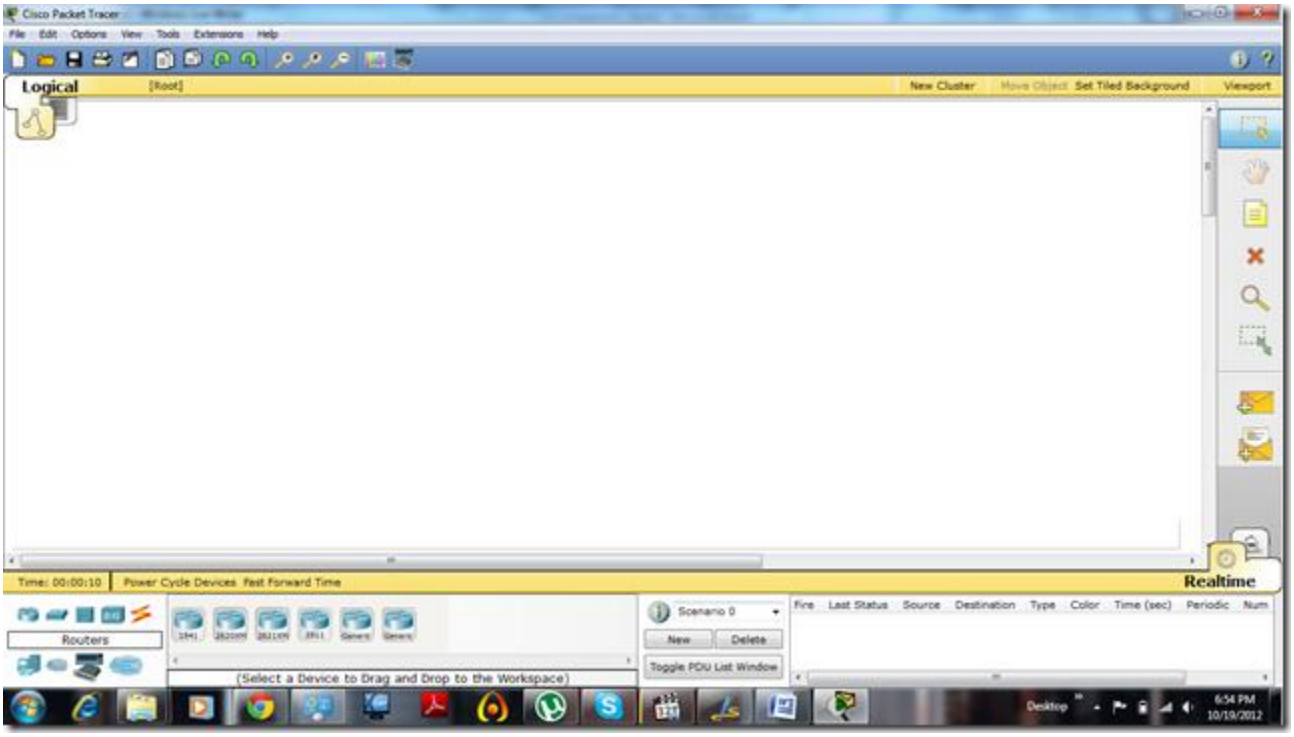
Experiment No.9
Packet Tracer Software

Aim:- Study and implementation of network using Cisco Packet Tracer Software .

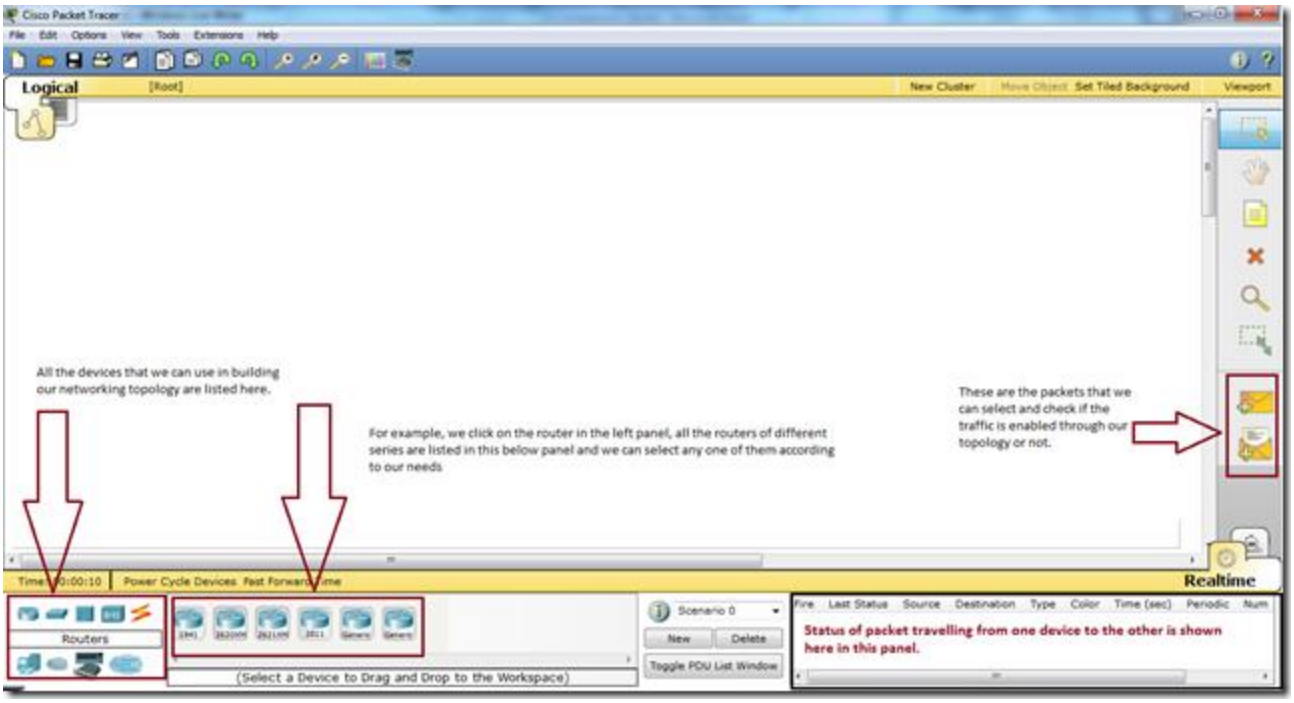
Objective: Student should be able to understand and design network using different network devices such machine,router,switches.

Theory: Packet Tracer is a powerful network simulator that can be utilized in training for CCNA and CCNP certification exam by allowing students to create networks with an almost unlimited number of devices and to experience troubleshooting without having to buy real Cisco routers or switches. The tool is created by Cisco Systems. The purpose of Packet Tracer is to offer students a tool to learn the principles of networking as well as develop Cisco technology specific skills. However, it is not be used as a replacement for Routers or Switches.

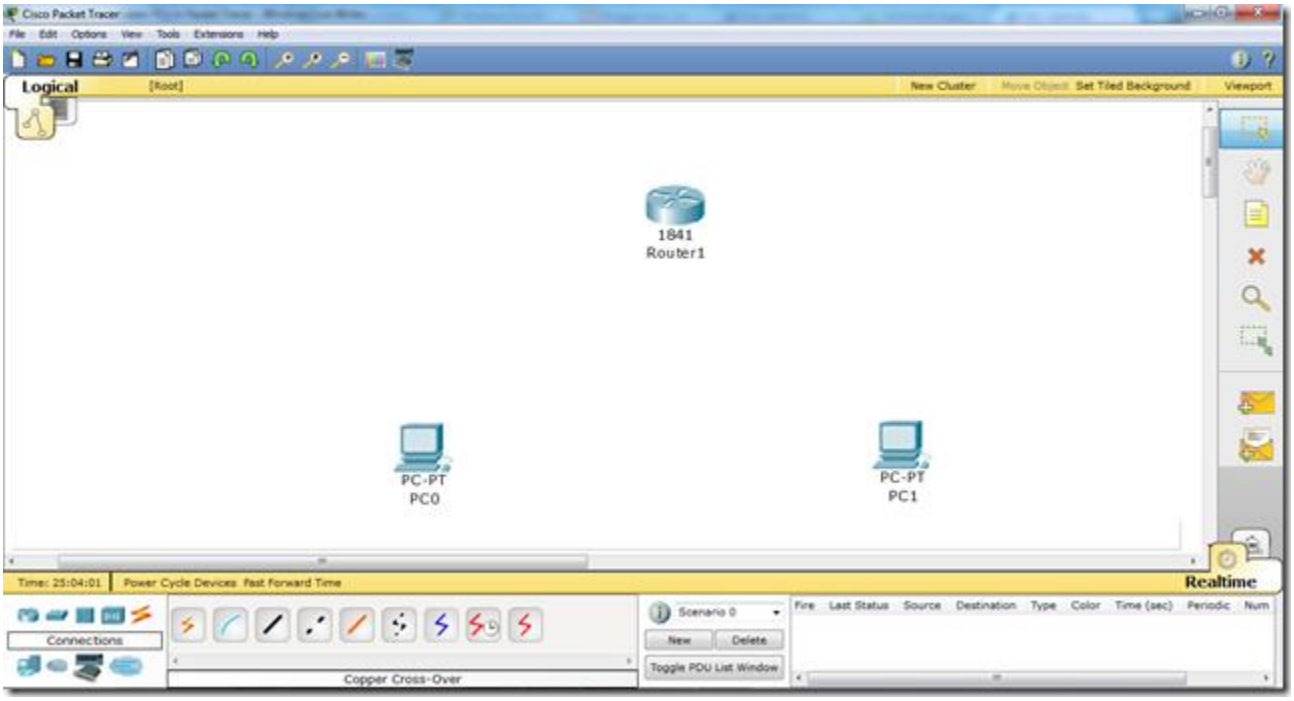
Here how it looks like after we start it.



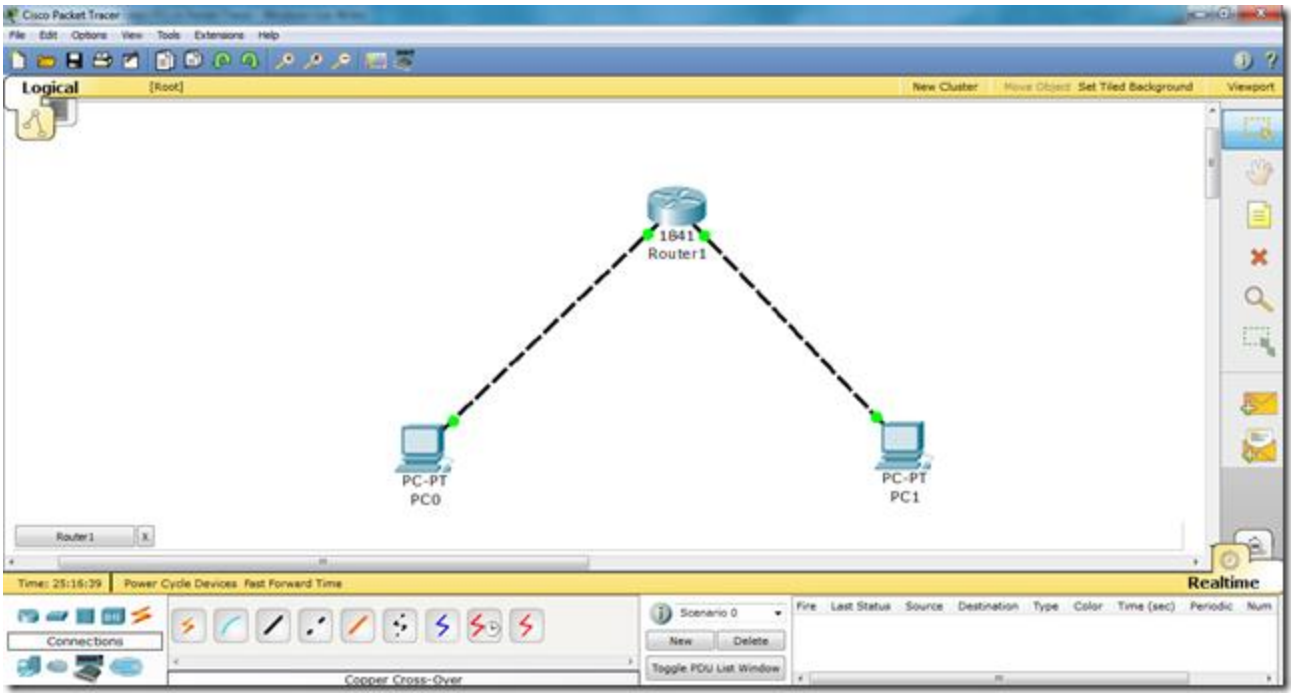
We are different modules and panels available in the packet tracer. Some important modules, which are important to understand for the working in Packet Tracer, are mentioned in the following diagram.



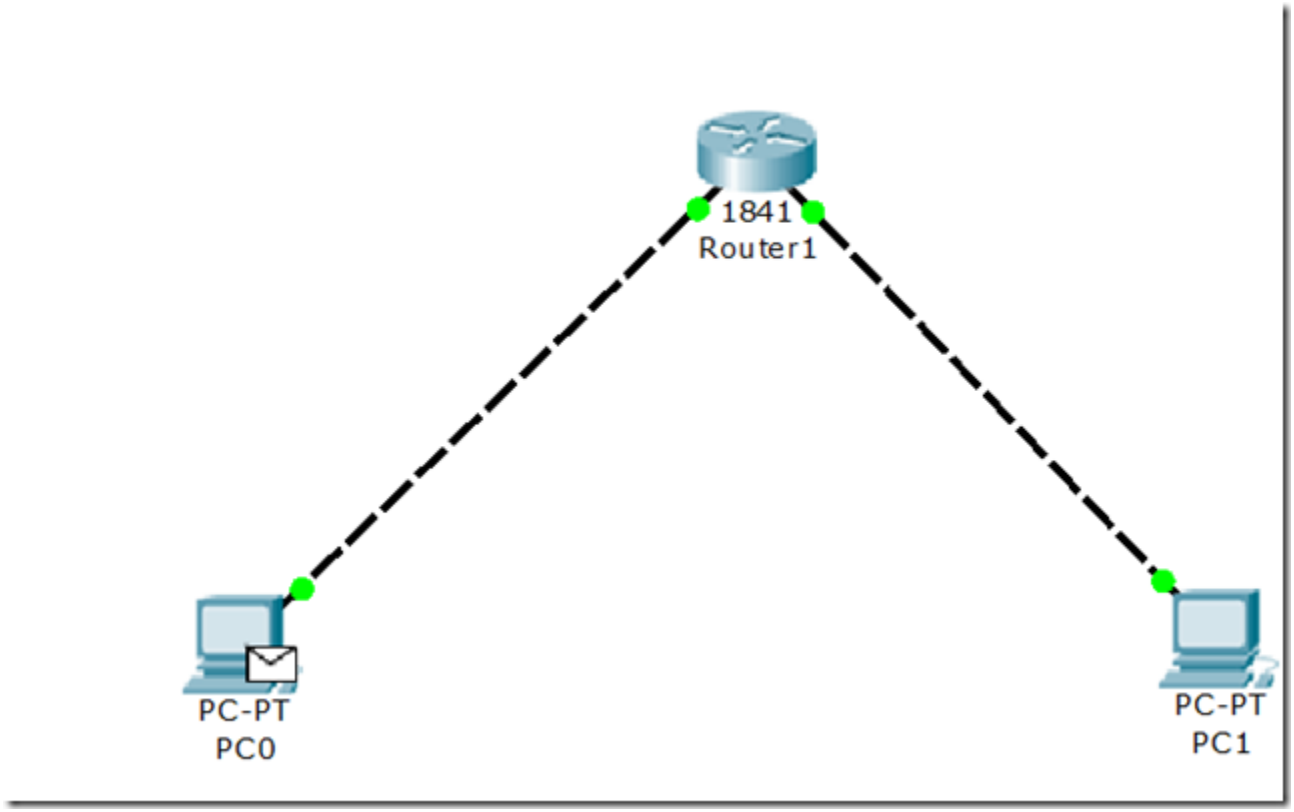
Now, in order to create a topology, we will have to select some of the devices and put them in our main window i.e. the white portion of packet tracer. and here how it looks after we add the devices.



Now, we will have to connect these devices and for that we use cables. To understand the cables, please refer to my following article.

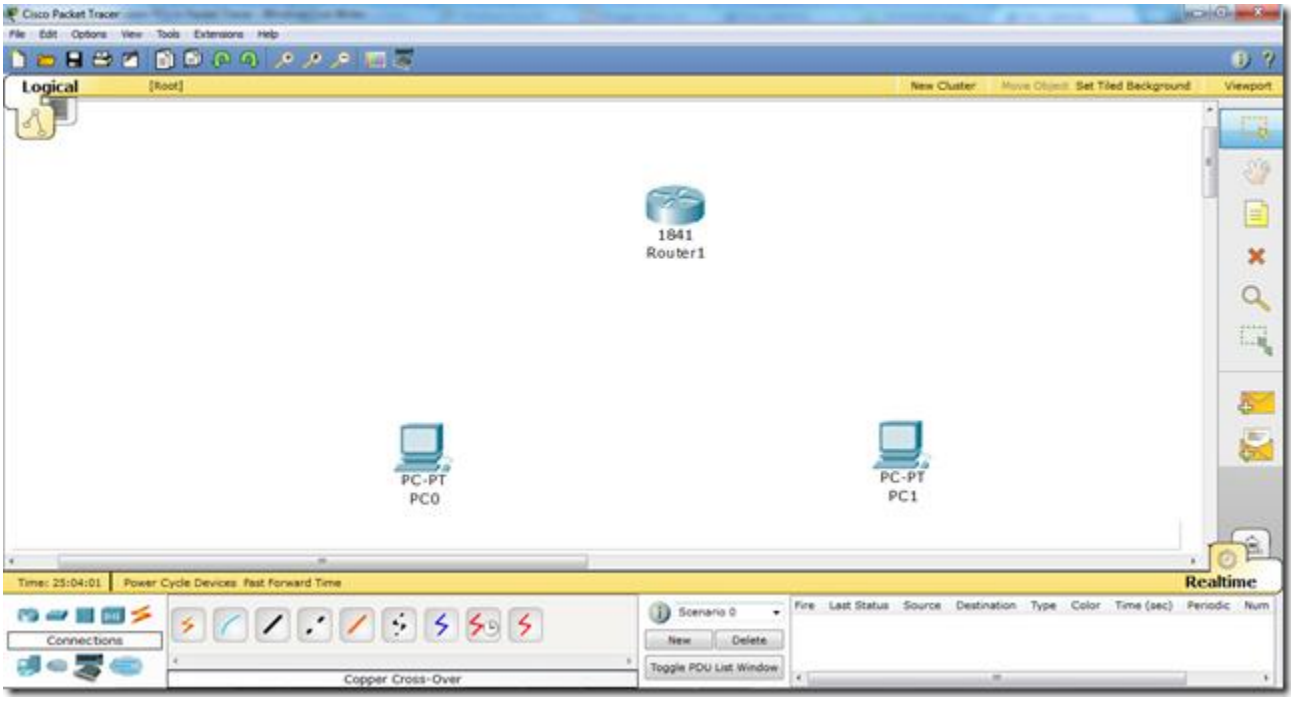


And after you successfully create the topology, you can check either the traffic is flowing or not by selecting the packet from right panel and putting it on both PCs as follows.

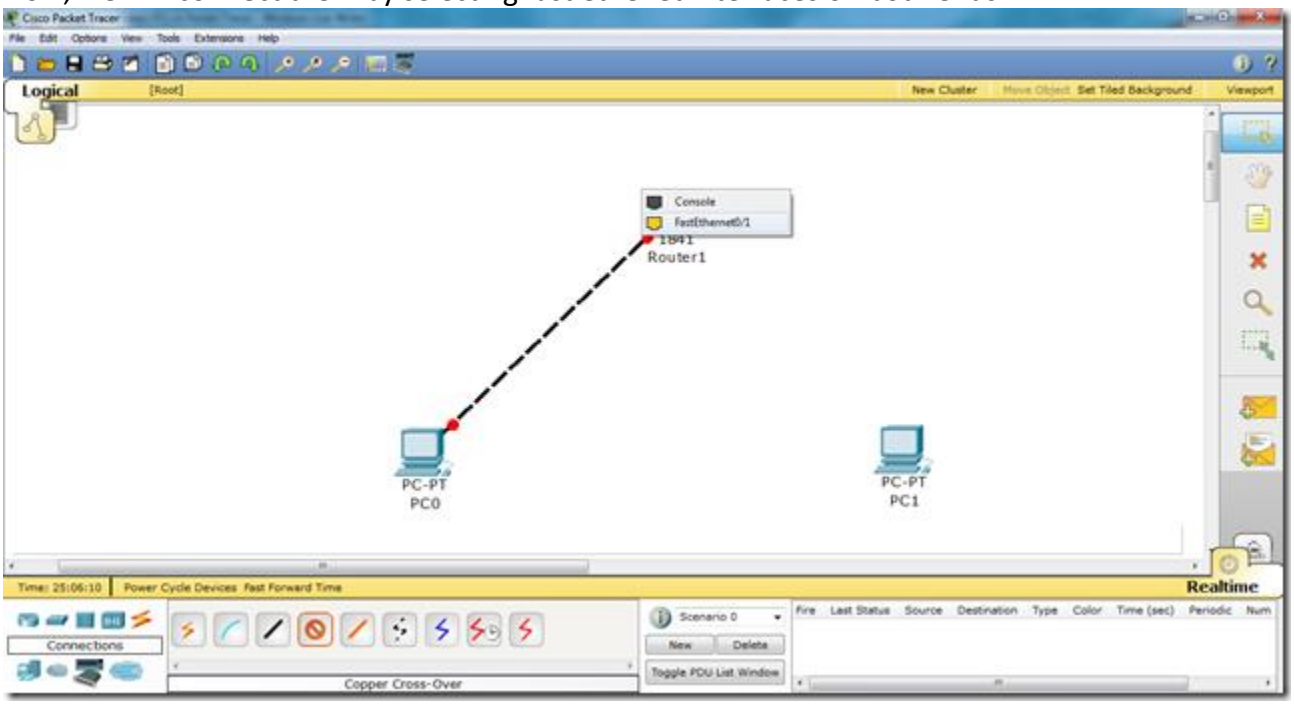


For detailed article on making the topology and successfully enabling the communication,
Here, we will see communication enabled between PCs via Router in Packet Tracer.
So, for this we need two PCs, a router ,and two cross over cables to connect them.
Important point is that we use cross over cable to connect PC to a router because they both use

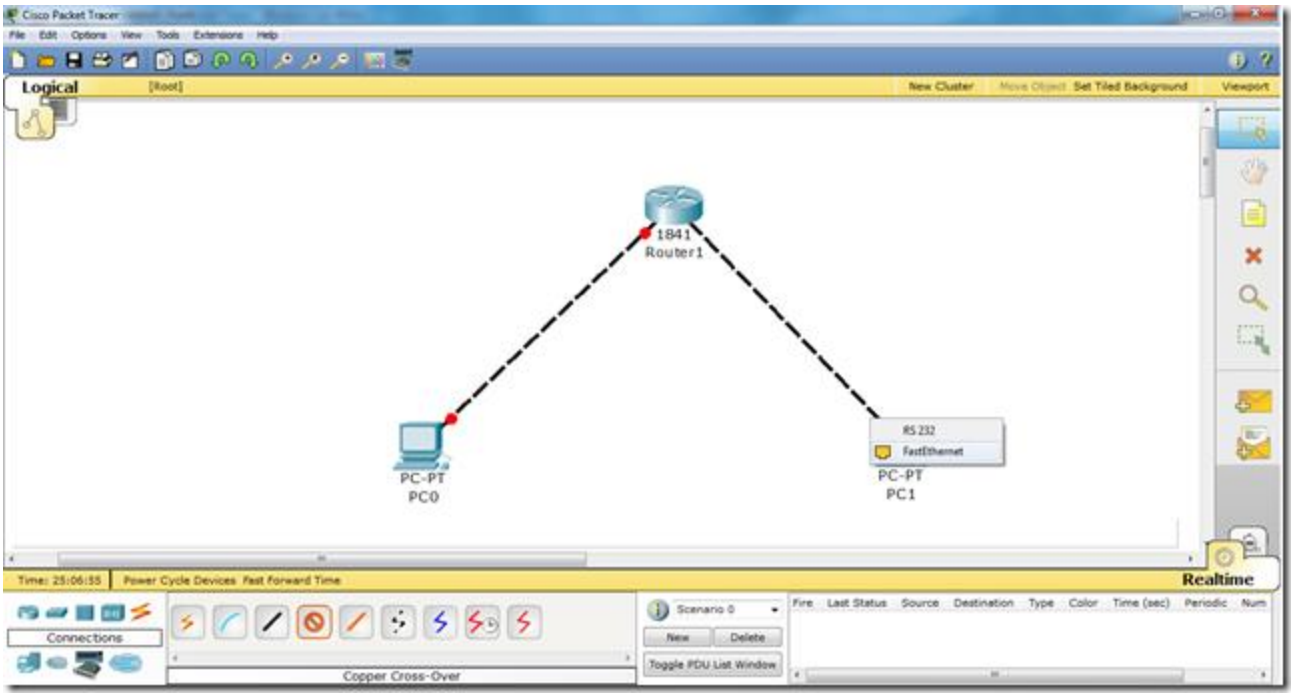
the same pins for transmission and receiving of data.



Now, we will connect them by selecting fast ethenet interfaces on both ends.



Similarly, on the PC side we will select fast Ethernet interface.



Now, we have connect the devices. Further, we will go to the router CLI mode and enter the following commands.

Step by step ,

we will have to do the following things.

i. Access the interfaces one by one

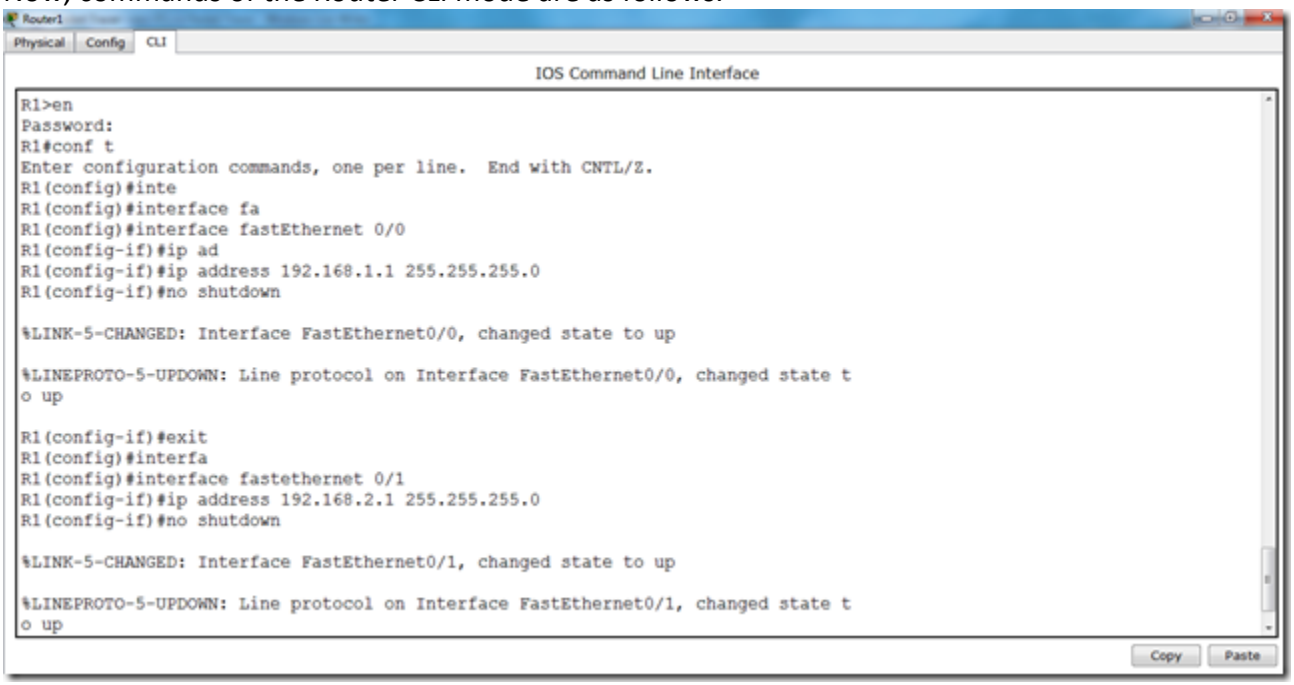
ii. Assign IP addresses to interfaces

iii. Change the status of the interfaces i.e. from Down to Up.

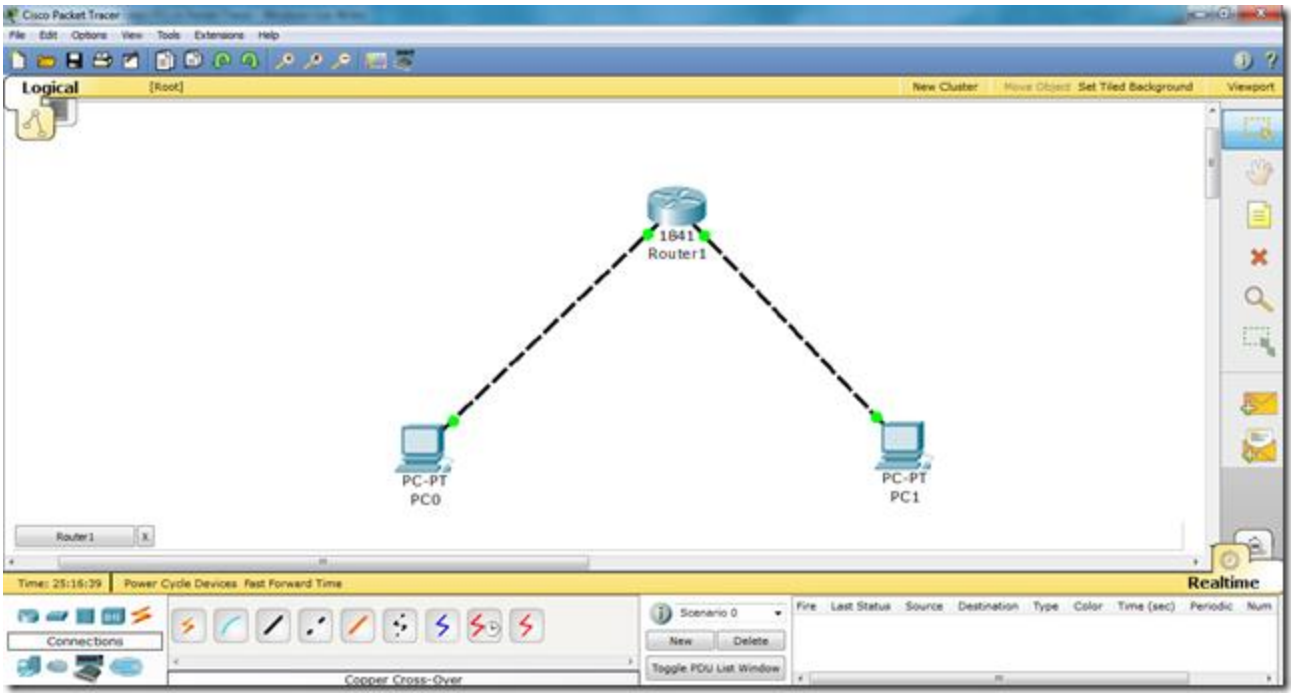
iv. Assign IP addresses to PCs.

v. Assign Default GateWay to PCs. FYI fast ethernetip address is the gateway address to the PC.

Now, commands of the Router CLI mode are as follows.

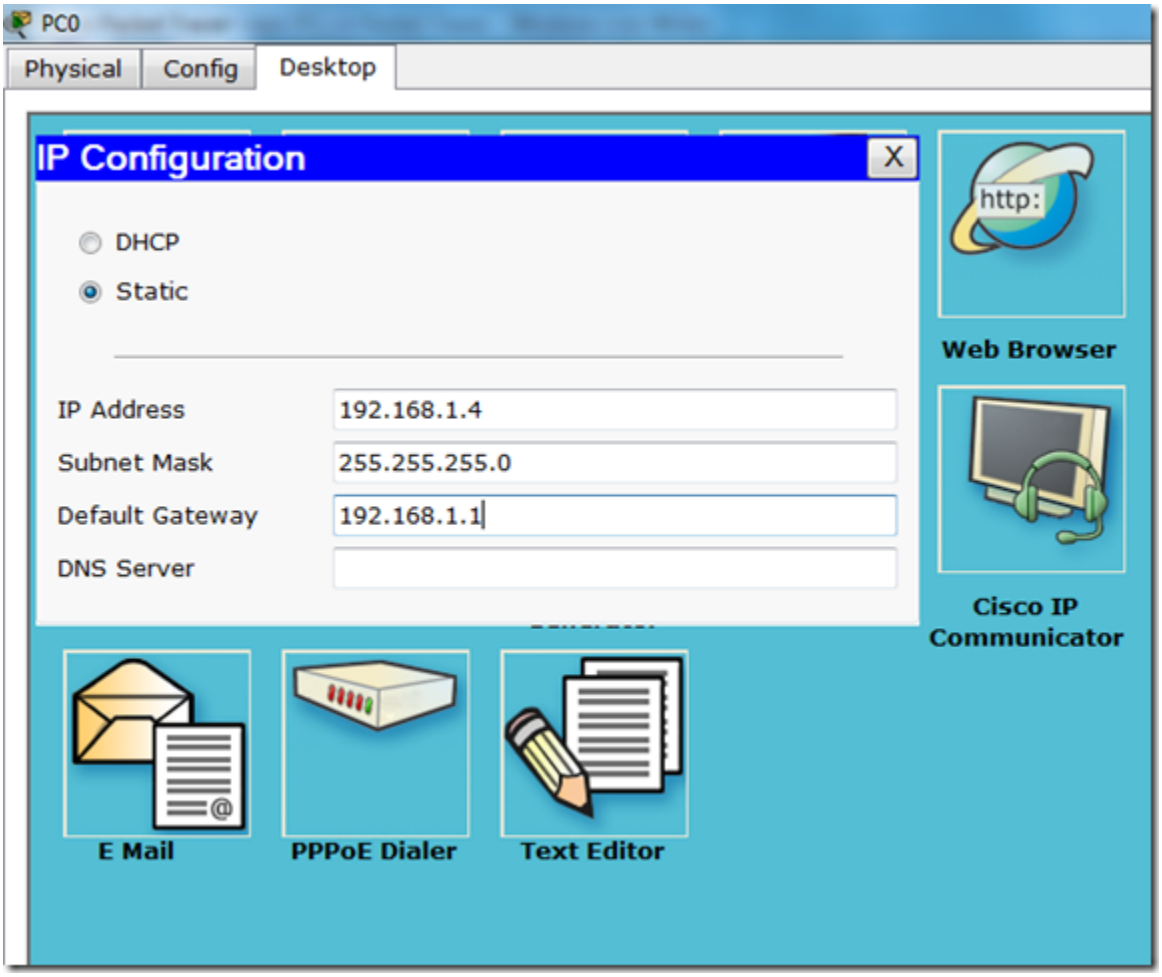


Now, we have accessed both interfaces one by one and we have assigned IP addresses respectively.

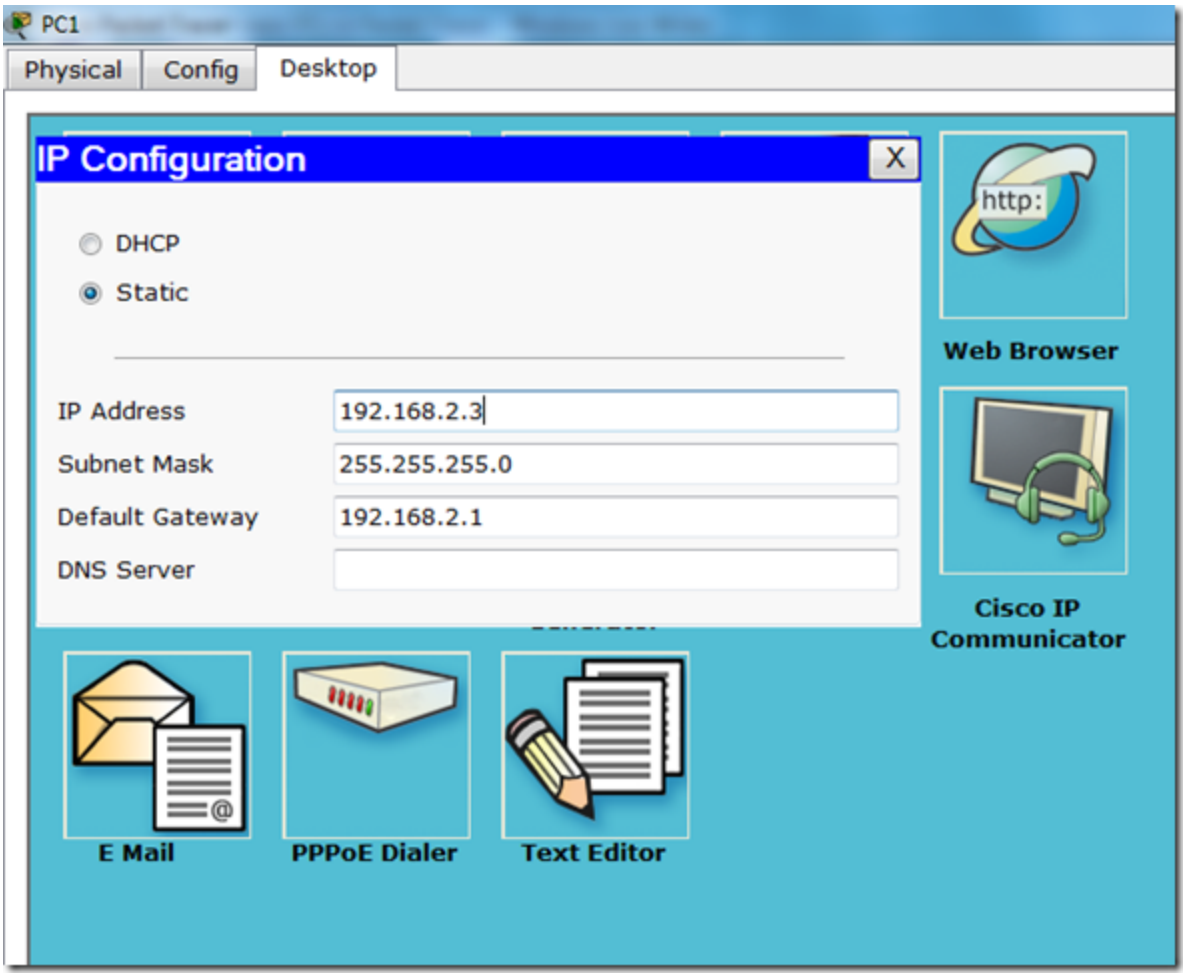


See the difference the lights have changed the color from Red to Green :)
Now, lets assign IP addresses to the PCs.
Click on PC1, go to Desktop, then click IP Configuration.

PC1:

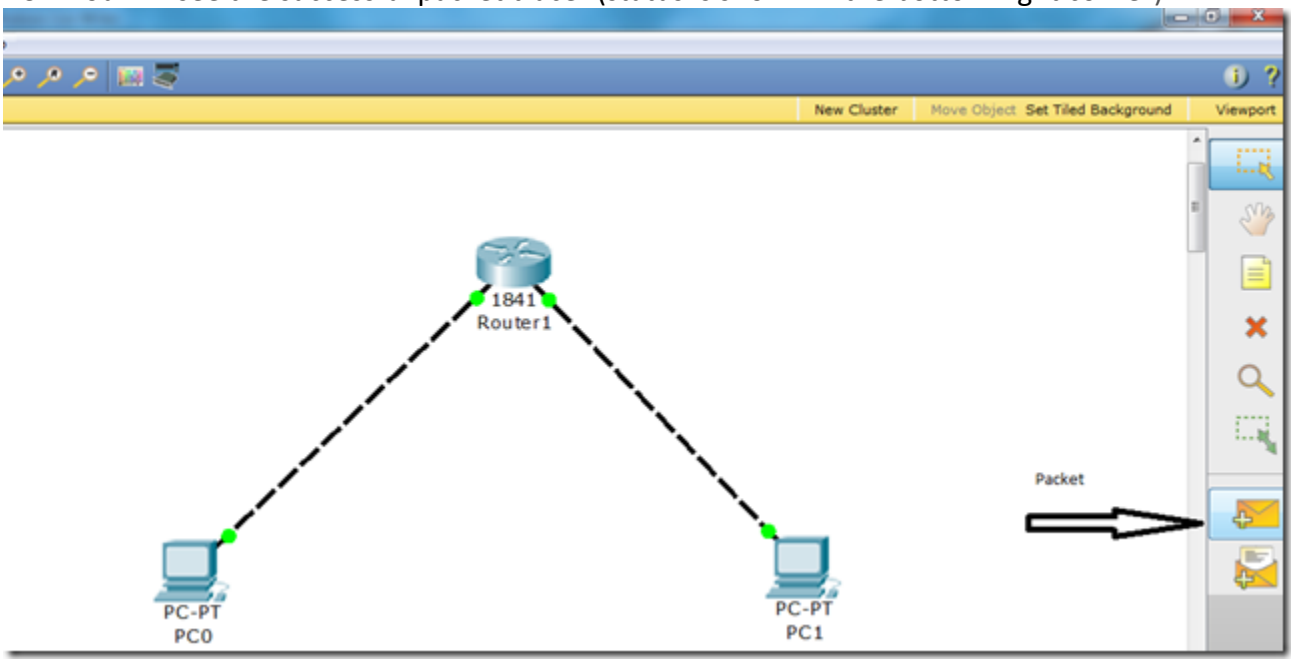


PC2:

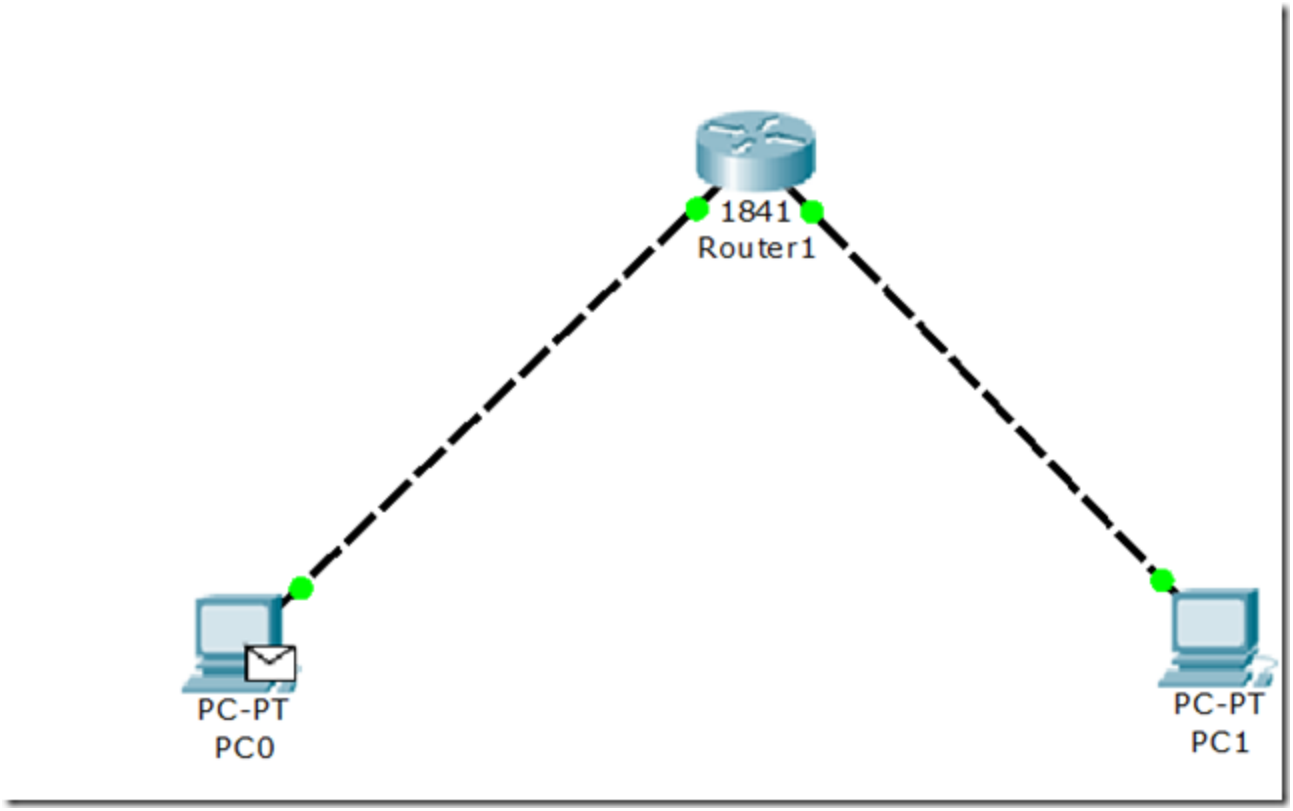


Now, our communication is enabled and we are able to communicate from PC1 to PC2 via Router.

Click on the packet in the right panel on the packet tracer, then click on PC1 and then click on PC2. You will see the successful packet tracer (status is shown in the bottom right corner)



Select it and click on both PCs.



Bingo, your communication is successful.

Conclusion:- Hence we implement network connection using Packet Tracer Software.

Experiment No.10
RSA algorithm

Aim:- Implement RSA algorithm using encryption and decryption key.

Objective: Student should be able to understand and implement the program of RSA algorithm.

Hardware Requirement: Intel based desktop pc: RAM 51

Software Requirement: Turbo C/Borland C

Theory:
RSA method is based on some principle from number theory. In encryption process divide the plain text into blocks, so that each plain text message p falling the interval 0<p<n this can be done by grouping the plain text into blocks of k bits .where k is the largest integer for which 2 power k <n is true .the security of this method is based on the difficult of factoring large no. he encryption and decryption function are inverse.

Algorithm:
Step 1:Start
Step 2:initalize variables as int a,b,i,j,t,x,n,k=0,flag=0,prime[100];
Step 3:initalize the variables as char m[20],pp[20];
Step 4: initalize variables as float p[20],c[20];
Step 5: initalize variables as double e,d;
Step 6: initalize i=0;
Step 7:Repeate step (7 – 16)until i<50;
Step 8:increment i;
Step 9: initalize flag=0;
Step 10:initalize j=2;
Step 11:repeate until j<i/2.
Step 12:if'i%j==0' repeate until (12-14)
Step 13:initalize flag=1;
Step 14:break;
Step 15:if'(flag==0)'.
Step 16:initalize prime[k++]=i;
Step 17:initalize a=prime[k-1];
Step 18:initalize b=prime[k-2];
Step 19:initalize n=a*b;
Step 20:initalize t=(a-1)*(b-1);
Step 21:initalize e=(double)prime[2];
Step 22:initalize d=1/(float)e;
Step 23:write "\n key of encription is :d.
Step 24:write "\n enetr the plain the text".
Step 25:read m;
Step 26:initalize x=strlen(m);

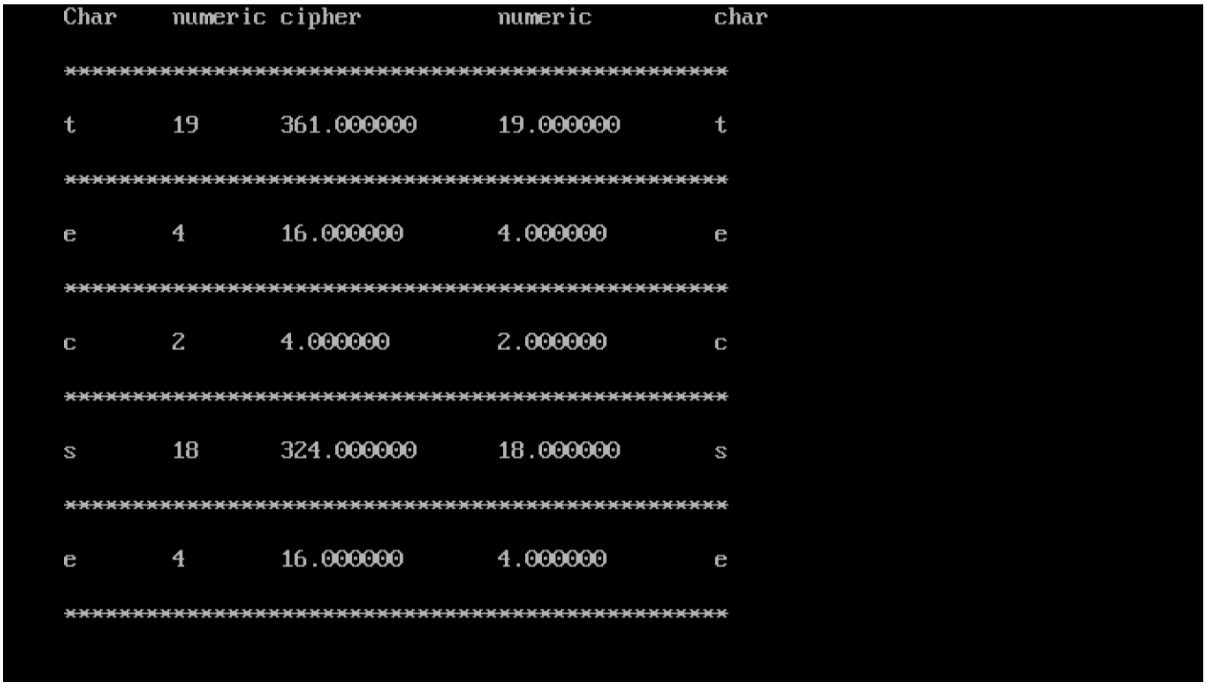
Step 27:write "\n Decription status from source to destination :.
Step 28:write "\n source\t>--------------------<destination \n".
Step 29:write "\nChar\tnumeric\tcipher\t\tnumeric\t\tchar\n".
Step 30:write "\n**\n"
Step 31:write "\n"
Step 32:initalize i=0;
Step 33:repeate steps(33-46)until i<x.
Step 34:incerement i.
Step 35:write"\n%c",m[i].
Step 36:write"\t%d",m[i]-97.
Step 37:init c[i]=pow(m[i]-97,(float)e);
Step 38:init c[i]=fmod(c[i],(float)n);
Step 39:write "\t%f",c[i].
Step 40:init p[i]=pow(c[i],(float)d);
Step 41:init p[i]=fmod(p[i],(float)n);
Step 42:write "\t%f",p[i].
Step 43:init pp[i]=p[i]+97;
Step 44:write "\t%c\n",pp[i]
Step 45:write "\n**\n"
Step 46:write "\n"
Step 27: end.
Step 27:write

Program:

```
#include <stdio.h>
#include<conio.h>
#include<ctype.h>
#include<math.h>
#include<string.h>
void main()
{
int a,i,b,j,t,x,n,k=0,flag=0,prime[100];
char m[20],pp[20];
float p[20],c[20];
double e,d;
clrscr();
for(i=0;i<50;i++)
{
flag=0;
for(j=2;j<i/2;j++)
if(i%j==0)
{
flag=1;
break;
}
if(flag==0)
{
```

```
prime[k++]=i;
}
a=prime[k-1];
b=prime[k-2];
n=a*b;
t=(a-1)*(b-1);
e=(double)prime[2];
d=1/(float)e;
printf("\n key of encription  is :%1f\n",d);
printf("\n enetr the plain the text");
scanf("%s",&m);
x=strlen(m);
printf("\n Decription status from source to destination :\n");
printf("\n source\t>--------------------<destination \n");
printf("\nChar\tnumeric\tcipher\t\tnumeric\t\tchar\n");
printf("\n**************************************\n");
printf("\n");
for(i=0;i<x;i++)
{
printf("\n%c",m[i]);
printf("\t%d",m[i]-97);
c[i]=pow(m[i]-97,(float)e);
c[i]=fmod(c[i],(float)n);
printf("\t%f",c[i]);
p[i]=pow(c[i],(float)d);
p[i]=fmod(p[i],(float)n);
printf("\t%f",p[i]);
pp[i]=p[i]+97;
printf("\t%c\n",pp[i]);
printf("\n*********************************************\n");
printf("\n");
}
getch();
}
}
```

Observations:-



Char	numeric	cipher	numeric	char
t	19	361.000000	19.000000	t
e	4	16.000000	4.000000	e
c	2	4.000000	2.000000	c
s	18	324.000000	18.000000	s
e	4	16.000000	4.000000	e

Conclusion: - Hence we studied and implemented RSA algorithm.

3.Quiz on the subject:-

1. What is the purpose of Domain Name System?
Domain Name System can map a name to an address and conversely an address to name.
2. Discuss the three main division of the domain name space.
Domain name space is divided into three different sections: generic domains, country domains & inverse domain.
i)Generic domain: Define registered hosts according to their generic behavior, uses generic suffixes.
ii)Country domain: Uses two characters to identify a country as the last suffix. Inverse domain: Finds the domain name given the IP address.
3. Discuss the TCP connections needed in FTP.
FTP establishes two connections between the hosts. One connection is used for data transfer, the other for control information. The control connection uses very simple rules of communication. The data connection needs more complex rules due to the variety of data types transferred.
4. Discuss the basic model of FTP.
The client has three components: the user interface, the client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.
5. What is the function of SMTP?
The TCP/IP protocol supports electronic mail on the Internet is called Simple Mail Transfer (SMTP). It is a system for sending messages to other computer users based on e-mail addresses. SMTP provides mail exchange between users on the same or different computers.
6. What is the difference between a user agent (UA) and a mail transfer agent (MTA)?
The UA prepares the message, creates the envelope, and puts the message in the envelope. The MTA transfers the mail across the Internet.
7. How does MIME enhance SMTP?
MIME is a supplementary protocol that allows non-ASCII data to be sent through SMTP. MIME transforms non-ASCII data at the sender site to NVT ASCII data and deliverers it to the client SMTP to be sent through the Internet.
8. Why is an application such as POP needed for electronic messaging
Workstations interact with the SMTP host, which receives the mail on behalf of every host in the organization, to retrieve messages by using a client-server protocol such as Post Office Protocol, version 3(POP3). Although POP3 is used to download messages from the server, the SMTP client still needed on the desktop to forward messages from the workstation user to its SMTP mail server.
9. Write down the three types of WWW documents.
The documents in the WWW can be grouped into three broad categories: static, dynamic and active.
Static: Fixed-content documents that are created and stored in a server.
Dynamic:Created by web server whenever a browser requests the document. Active: A program to be run at the client side.
10. What is the purpose of HTML?
HTML is a computer language for specifying the contents and format of a web document. It allows additional text to include codes that define fonts, layouts, embedded graphics and

hypertext links.
11. Define CGI.
CGI is a standard for communication between HTTP servers and executable programs. It is used in crating dynamic documents.
12. Name four factors needed for a secure network.
Privacy: The sender and the receiver expect confidentiality.
Authentication: The receiver is sure of the sender's identity and that an imposter has not sent the message.
Integrity: The data must arrive at the receiver exactly as it was sent. Non-Reputation: The receiver must able to prove that a received message came from a specific sender.
13. How is a secret key different from public key?
In secret key, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data.
The public key is announced to the public.
14. What is a digital signature?
Digital signature is a method to authenticate the sender of a message. It is similar to that of signing transactions documents when you do business with a bank. In network transactions, you can create an equivalent of an electronic or digital signature by the way you send data.
15. What are the advantages & disadvantages of public key encryption?
Advantages:
a) Remove the restriction of a shared secret key between two entities. Here each entity can create a pair of keys, keep the private one, and publicly distribute the other one.
b) The no. of keys needed is reduced tremendously. For one million users to communicate, only two million keys are needed.
Disadvantages:
If you use large numbers the method to be effective. Calculating the cipher text using the long keys takes a lot of time. So it is not recommended for large amounts of text.
16. What are the advantages & disadvantages of secret key encryption?
Advantages:
Secret Key algorithms are efficient: it takes less time to encrypt a message. The reason is that the key is usually smaller. So it is used to encrypt or decrypt long messages.
Disadvantages:
a) Each pair of users must have a secret key. If N people in world want to use this method, there needs to be N (N-1)/2 secret keys. For one million people to communicate, a halfbillion secret keys are needed.
b) The distribution of the keys between two parties can be difficult.
17. Define permutation.
Permutation is transposition in bit level.
Straight permutation: The no. of bits in the input and output are preserved. Compressed permutation: The no. of bits is reduced (some of the bits are dropped). Expanded permutation: The no. of bits is increased (some bits are repeated).
18. Define substitutional & transpositional encryption.
Substitutional: A character level encryption in which each character is replaced by another character in the set.
Transpositional: A Character level encryption in which the characters retain their plaintext
19. Define Routers

Routers relay packets among multiple interconnected networks. They Route packets from one network to any of a number of potential destination networks on Internet routers operate in the physical, data link and network layer of OSI model

4. Conduction of Viva-Voce Examinations:

Teacher should conduct oral exams of the students with full preparation. Normally, the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested. Oral examinations are to be conducted in cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other in case of difference of opinion, which should be critically suppressed in front of the students.

5. Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become. It is a primary responsibility of the teacher to see that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with just circumstances.