

CMR TECHNICAL CAMPUS
UGC AUTONOMOUS
Kandlakoya(V), Medchal Road, Hyderabad – 501 401

Accredited by NBA and NAAC with A Grade
Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad

DEPARTMENT OF CSE (AI & ML)



COMPUTER NETWORKS LAB MANUAL (R22)

COURSE CODE: 22AM506PC

Prepared by:
Mr. B. Prashanth
Assistant Professor

List of Experiments

S. No	Programs
1	Write a Program to implement the data link layer framing methods such as character, character-stuffing and bit stuffing.
2	Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP
3	Write a Program to implement a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.
4	Write a Program to Implement Dijkstra's algorithm to compute the shortest path through a network.
5	Write a Program to simulate the subnet of hosts and obtain a broadcast tree for the subnet.
6	Write a Program to Implement distance vector routing algorithm for obtaining routing tables at each node.
7	Write a Program to Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
8	Write a program for congestion control using Leaky bucket algorithm.
9	Write a program for frame sorting techniques used in buffers.
10	Demonstrate how to run NMap scan.
11	Demonstration of Operating System Detection using N map.
12	Study of basic Network configuration commands and utilities to debug the network issues.

EXPERIMENT-1

AIM: Write a Program to implement the data link layer framing methods such as character, character-stuffing and bit stuffing.

THEORY:

The framing method gets around the problem of re-synchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever loses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data is given to the network layer. This technique is called character stuffing

ALGORITHM:

Begin

Step 1: Initialize I and j as 0

Step 2: Declare n and pos as integer and a[20],b[50],ch as character

Step 3: read the string a

Step 4: find the length of the string n, i.e n=strlen(a)

Step 5: read the position, pos

Step 6: if pos > n then

Step 7: print invalid position and read again the position, pos

Step 8: end if

Step 9: read the character, ch

Step 10: Initialize the array b , b[0...5] as 'd', 'l', 'e', 's', 't', 'x' respectively

Step 11: j=6;

Step 12: Repeat step[(13to22) until i<n

Step 13: if i==pos-1 then

Step 14: initialize b array,b[j],b[j+1]...b[j+6] as 'd', 'l', 'e', 'ch', 'd', 'l', 'e' respectively

Step 15: increment j by 7, i.e j=j+7

Step 16: end if

Step 17: if a[i]=='d' and a[i+1]=='l' and a[i+2]=='e' then

Step 18: initialize array b, b[13...15]='d', 'l', 'e' respectively

Step 19: increment j by 3, i.e $j=j+3$

Step 20: end if

Step 21: $b[j]=a[i]$

Step 22: increment I and j;

Step 23: initialize b array, $b[j], b[j+1] \dots b[j+6]$ as 'd', 'l', 'e', 'e', 't', 'x', '\0' respectively

Step 24: print frame after stuffing

Step 25: print b

End

SOURCE CODE:

//PROGRAM FOR CHARACTER STUFFING

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
#include<string.h>
```

```
#include<process.h>
```

```
int main()
```

```
{
```

```
int i=0,j=0,n,pos;
```

```
char a[20],b[50],ch;
```

```
int clrscr();
```

```
printf("enter string\n");
```

```
scanf("%s",&a);
```

```
n=strlen(a);
```

```
printf("enter position\n");
```

```
scanf("%d",&pos);
```

```
if(pos>n)
```

```
{
```

```
printf("invalid position, Enter again :");
```

```
scanf("%d",&pos);
```

```
}
```

```
printf("enter the character\n");
```

```
ch=getche();
```

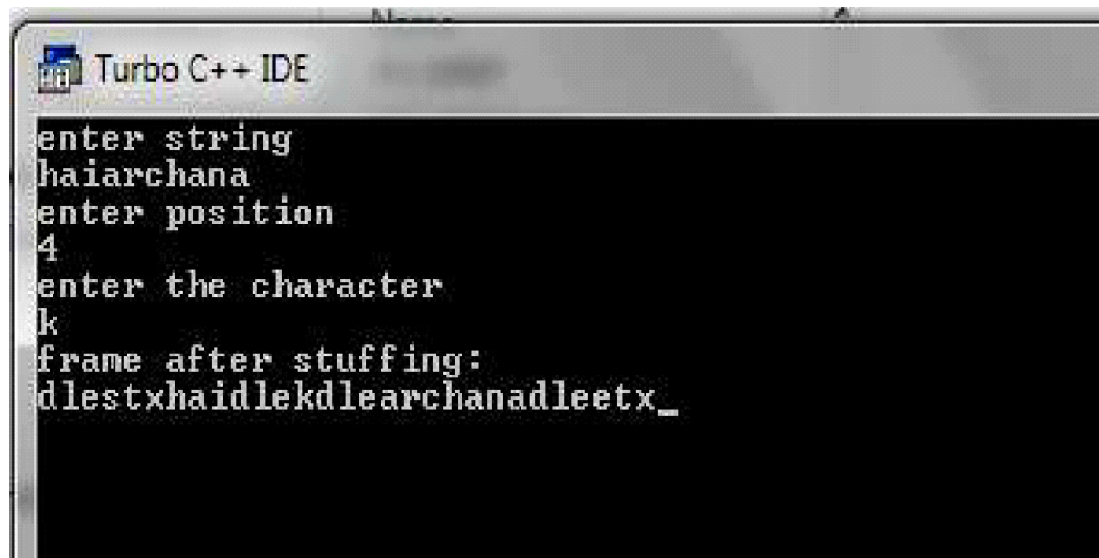
```
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]='ch';
b[j+4]='d';
b[j+5]='l';
b[j+6]='e';
j=j+7;
}
if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
j=j+3;
}
b[j]=a[i];
i++;
j++;
}
b[j]='d';
```

```

b[j+1]='l';
b[j+2]='e';
b[j+3]='e';
b[j+4]='t';
b[j+5]='x';
b[j+6]='\0';
printf("\nframe after stuffing:\n");
printf("%s",b);
getch();
}

```

OUTPUT:



The screenshot shows the Turbo C++ IDE window with the following text displayed in the console:

```

enter string
haiarchana
enter position
4
enter the character
k
frame after stuffing:
dlestxhaidlekdlearchanadleetx_

```

SOURCE CODE:

```


//PROGRAM FOR BIT STUFFING
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
int a[20],b[30],i,j,k,count,n;

```

```
int clrscr();
printf("Enter frame length:");
scanf("%d",&n);
printf("Enter input frame (0's & 1's only):");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
i =0; count=1; j=0;
while(i<n)
{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1;a[k]==1 && k<n && count<5;k++)
{
j++;
b[j]=a[k];
count++;
if(count==5)
{
j++;
b[j]=0;
}
i=k;
}}
else
{
b[j]=a[i];
}
i++;
j++;
}
printf("After stuffing the frame is:");
```

```
for(i=0;i<j;i++)  
printf("%d",b[i]);  
getch();  
}
```

OUTPUT:

 C:\Users\admin\Desktop\Exp1 - A.exe

```
Enter frame length:5  
Enter input frame (0's & 1's only):0  
1  
0  
0  
1  
After stuffing the frame is:01001
```


EXPERIMENT-2

AIM: Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP

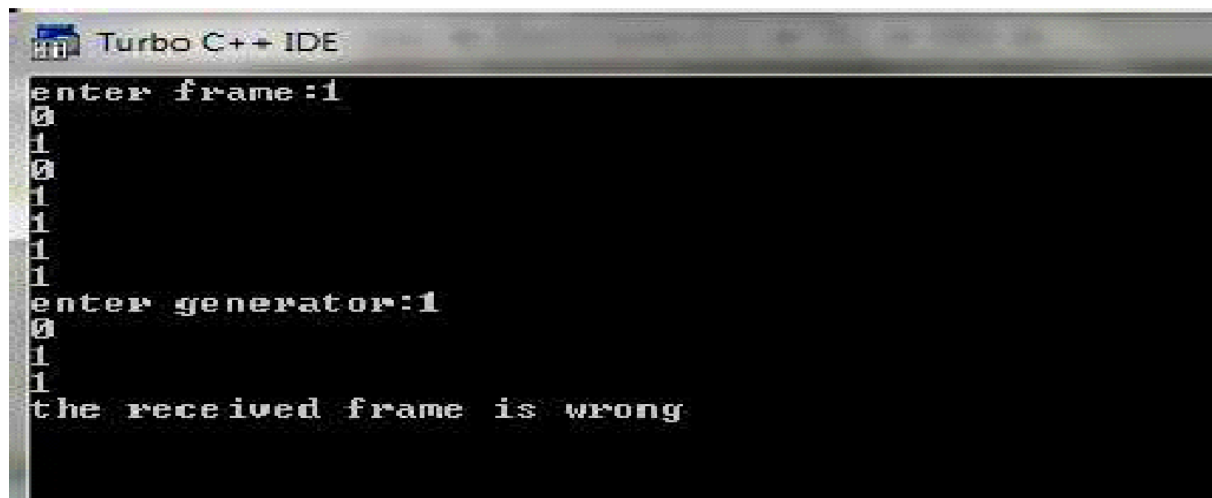
SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int gen[4],genl,frl,rem[4];
remainder(int fr[])
{
int k,k1,i,j;
for(k=0;k<frl;k++)
{
if(fr[k]==1)
{
k1=k;
for(i=0,j=k;i<genl;i++,j++)
{
rem[i]=fr[j]^gen[i];
}
for(i=0;i<genl;i++)
{
fr[k1]=rem[i];
k1++;
}
}
}
}
int main()
{
```

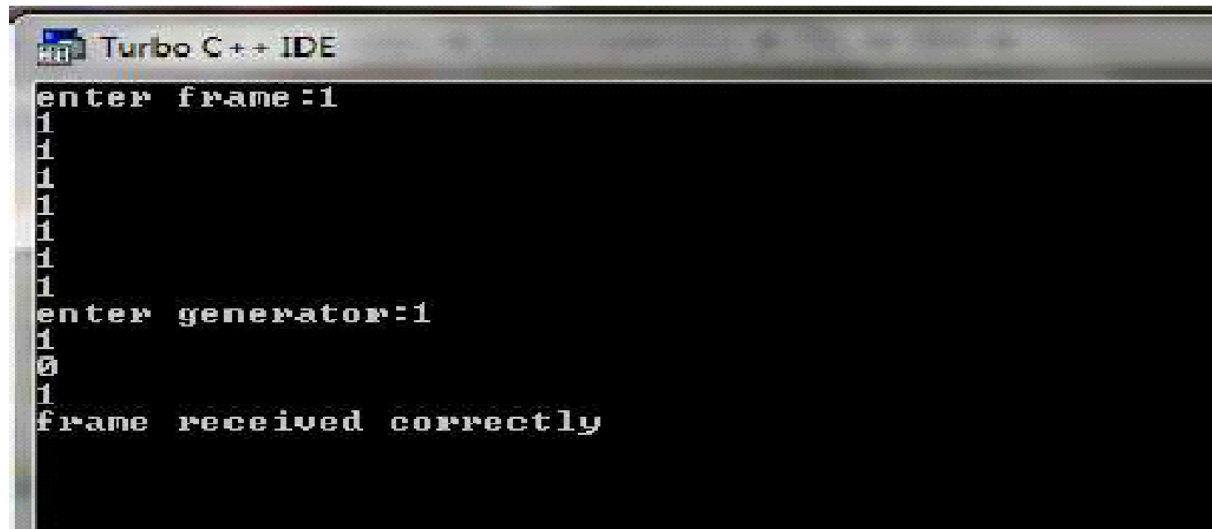
```
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
int clrscr();
frl=8; genl=4;
printf("enter frame:");
for(i=0;i<frl;i++)
{
scanf("%d",&fr[i]);
dupfr[i]=fr[i];
}
printf("enter generator:");
for(i=0;i<genl;i++)
scanf("%d",&gen[i]);
tlen=frl+genl-1;
for(i=frl;i<tlen;i++)
{
dupfr[i]=0;
}
remainder(dupfr);
for(i=0;i<frl;i++)
{
recfr[i]=fr[i];
}
for(i=frl,j=1;j<genl;i++,j++)
{
recfr[i]=rem[j];
}
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
flag++;
}
```

```
}  
if(flag==0)  
{  
printf("frame received correctly");  
}  
else  
{  
printf("the received frame is wrong");  
}  
getch();  
}
```

OUTPUT:



A screenshot of the Turbo C++ IDE window. The title bar reads "Turbo C++ IDE". The console output shows the following sequence of text: "enter frame:1", followed by eight lines of "1", then "enter generator:1", followed by two lines of "1", and finally "the received frame is wrong".



A screenshot of the Turbo C++ IDE window. The title bar reads "Turbo C++ IDE". The console output shows the following sequence of text: "enter frame:1", followed by eight lines of "1", then "enter generator:1", followed by two lines of "1", and finally "frame received correctly".

EXPERIMENT - 3

AIM: Write a Program to implement a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

SOURCE CODE:

```
#include<stdio.h>

int main()
{
    int w,i,f,frames[50];

    printf("Enter window size: ");
    scanf("%d",&w);
    printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);
    printf("\nEnter %d frames: ",f);
    for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);

    printf("\nWith sliding window protocol the frames will be sent in the following manner\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the\n\n");
    receiver\n\n",w);
    for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
    }
}
```

```

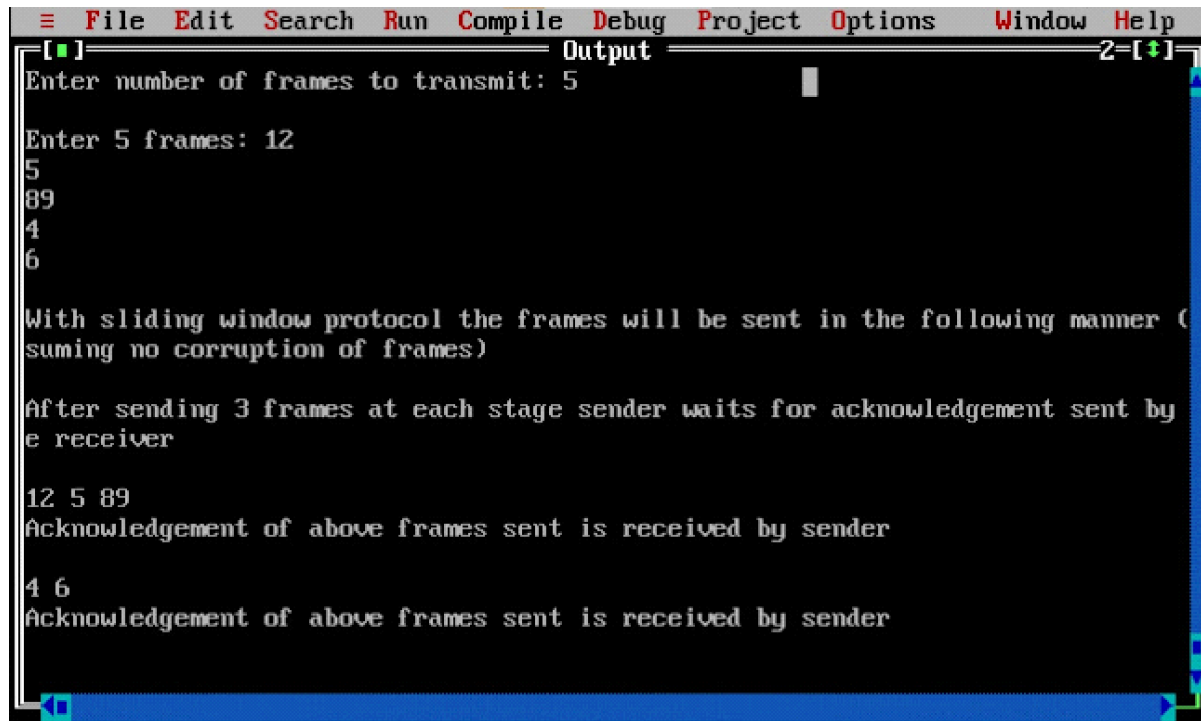
}

if(f%w!=0)
    printf("\nAcknowledgement of above frames sent is received by sender\n");

return 0;
}

```

OUTPUT:-



```

File Edit Search Run Compile Debug Project Options Window Help
Output
Enter number of frames to transmit: 5
Enter 5 frames: 12
5
89
4
6

With sliding window protocol the frames will be sent in the following manner (
suming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by
e receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender

```

EXPERIMENT-4

AIM: Write a Program to Implement Dijkstra's algorithm to compute the shortest path through a network.

ALGORITHM/FLOWCHART:

Begin

Step1: Declare array path [5] [5], min, a [5][5], index, t[5];

Step2: Declare and initialize st=1,ed=5

Step 3: Declare variables i, j, stp, p, edp

Step 4: print "enter the cost "

Step 5: i=1

Step 6: Repeat step (7 to 11) until (i<=5)

Step 7: j=1

Step 8: repeat step (9 to 10) until (j<=5)

Step 9: Read a[i] [j]

Step 10: increment j

Step 11: increment i

Step 12: print "Enter the path"

Step 13: read p

Step 14: print "Enter possible paths"

Step 15: i=1

Step 16: repeat step(17 to 21) until (i<=p)

Step 17: j=1

Step 18: repeat step(19 to 20) until (i<=5)

Step 19: read path[i][j]

Step 20: increment j

Step 21: increment i

Step 22: j=1

Step 23: repeat step(24 to 34) until(i<=p)

Step 24: t[i]=0

Step 25: stp=st

Step 26: $j=1$
 Step 27: repeat step(26 to 34) until($j \leq 5$)
 Step 28: $edp = \text{path}[i][j+1]$
 Step 29: $t[i] = t[i] + a[\text{stp}][edp]$
 Step 30: if ($edp == ed$) then
 Step 31: break;
 Step 32: else
 Step 33: $stp = edp$
 Step 34: end if
 Step 35: $min = t[\text{st}]$
 Step 36: $index = \text{st}$
 Step 37: repeat step(38 to 41) until ($i \leq p$)
 Step 38: $min > t[i]$
 Step 39: $min = t[i]$
 Step 40: $index = i$
 Step 41: end if
 Step 42: print" minimum cost" min
 Step 43: print" minimum cost pth"
 Step 44: repeat step(45 to 48) until ($i \leq 5$)
 Step 45: print $\text{path}[index][i]$
 Step 46: if($\text{path}[index][i] == ed$) then
 Step 47: break
 Step 48: end if
 End

SOURCE CODE:

```

//*****

//PROGRAM FOR FINDING SHORTEST //PATH FOR A GIVEN GRAPH

//*****

#include<stdio.h>
#include<conio.h>
int main()
  
```

```

{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
int clrscr();
printf("enter the cost matrix\n");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d",&a[i][j]);
printf("enter the paths\n");
scanf("%d",&p);
printf("enter possible paths\n");
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][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else
stp=edp;
}
}
min=t[st];index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{

```

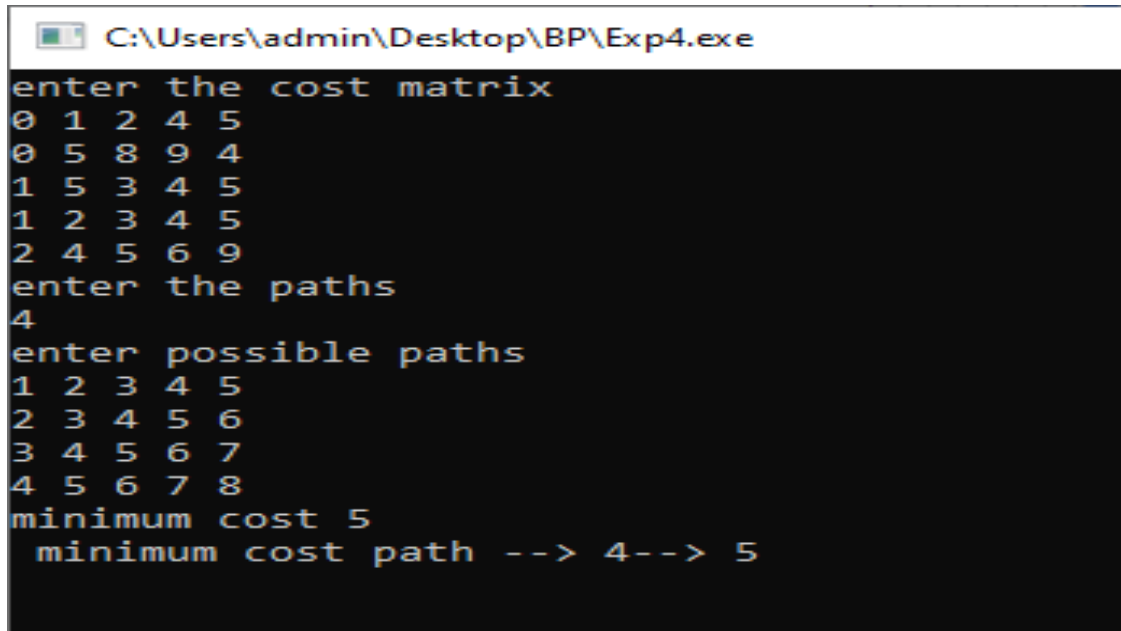


```

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();
}

```

OUTPUT:



```

C:\Users\admin\Desktop\BP\Exp4.exe
enter the cost matrix
0 1 2 4 5
0 5 8 9 4
1 5 3 4 5
1 2 3 4 5
2 4 5 6 9
enter the paths
4
enter possible paths
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
minimum cost 5
minimum cost path --> 4--> 5

```

EXPERIMENT-5

AIM: Write a Program to simulate the subnet of hosts and obtain a broadcast tree for the subnet.

PROGRAM:

// Write a 'c' program for Broadcast tree from subnet of host

```
#include<stdio.h>
```

```
int p,q,u,v,n;
```

```
int min=99,mincost=0;
```

```
int t[50][2],i,j;
```

```
int parent[50],edge[50][50];
```

```
main()
```

```
{
```

```
clrscr();
```

```
printf("\n Enter the number of nodes");
```

```
scanf("%d",&n);
```

```
for(i=0;i<n;i++)
```

```
{
```

```
printf("%c\t",65+i);
```

```
parent[i]=-1;
```

```
}
```

```
printf("\n");
```

```
for(i=0;i<n;i++)
```

```
{
```

```
printf("%c",65+i);
```

```
for(j=0;j<n;j++)
```

```
scanf("%d",&edge[i][j]);
```

```
}
```

```
for(i=0;i<n;i++)
```

```
{
```

```
for(j=0;j<n;j++)
```

```

if(edge[i][j]!=99)
if(min>edge[i][j])
{
min=edge[i][j];
u=i;
v=j;
}
p=find(u);
q=find(v);
if(p!=q)
{
t[i][0]=u;
t[i][1]=v;
mincost=mincost+edge[u][v];
union(p,q);
}
else
{
t[i][0]=-1;
t[i][1]=-1;
}
min=99;
}
printf("Minimum cost is %d\n Minimum spanning tree is\n",mincost);
for(i=0;i<n;i++)
if(t[i][0]!=-1 && t[i][1]!=-1)
{
printf("%c %c %d", 65+t[i][0], 65+t[i][1],
edge[t[i][0]][t[i][1]]);
printf("\n");
}
getch();

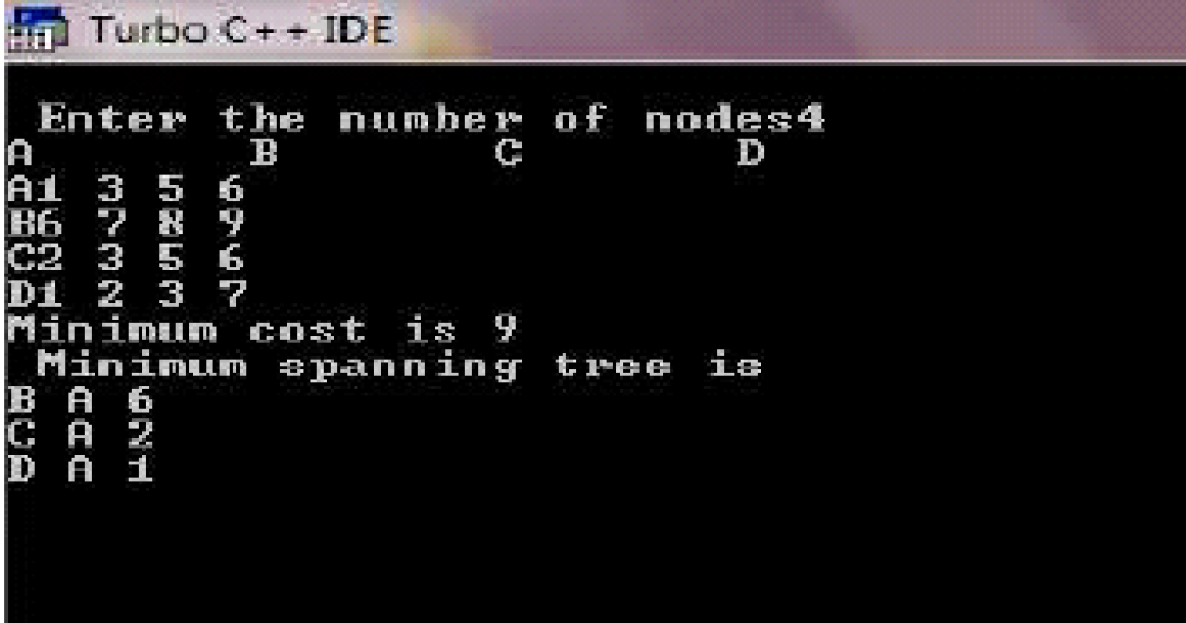
```

```

}
union(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)
l=parent[l];
return l;
}

```

OUTPUT:



```

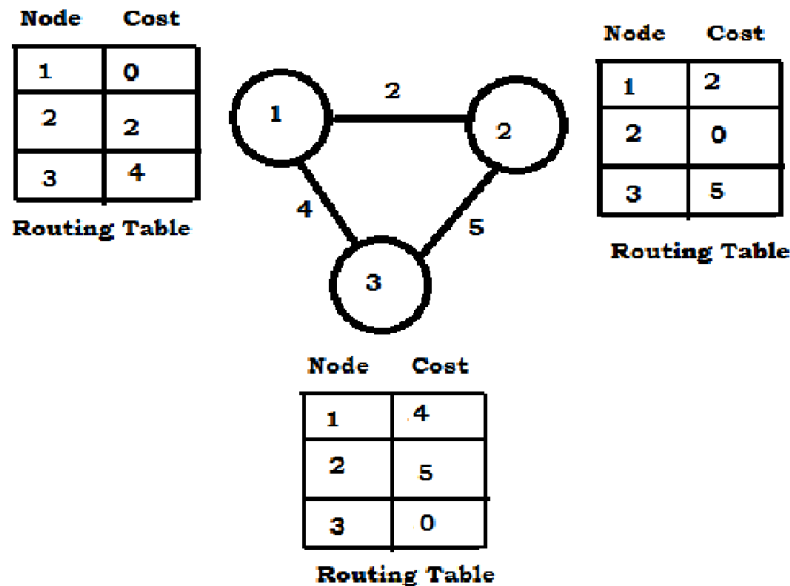
Turbo C++ IDE
Enter the number of nodes4
A      B      C      D
A1 3 5 6
B6 7 8 9
C2 3 5 6
D1 2 3 7
Minimum cost is 9
Minimum spanning tree is
B A 6
C A 2
D A 1

```

EXPERIMENT-6

AIM: Write a Program to Implement distance vector routing algorithm for obtaining routing tables at each node.

PROGRAM:



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 consider 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 converge.

ALGORITHM/FLOWCHART:

Begin

Step1: Create struct node unsigned dist[20], unsigned from[20], rt[10]

Step2: initialize int dmat[20][20], n, i, j, k, count=0,

Step3: write "the number of nodes "

Step4: read the number of nodes "n"

Step5: write" the cost matrix :"

Step6: initialize $i=0$

Step7: repeat until $i < n$

Step8: increment i

Step9: initialize $j=0$

Step10: repeat Step(10-16)until $j < n$

Step11: increment j

Step12: read $dmat[i][j]$

Step13: initialize $dmat[i][j]=0$

Step14: initialize $rt[i].dist[j]=dmat[i][j]$

Step15: initialize $rt[i].from[j]=j$

Step16: end

Step17: start do loop Step (17-33)until

Step18: initialize count =0

Step19: initialize $i=0$

Step20: repeat until $i < n$

Step21: increment i

Step22: initialize $j=0$

Step23: repeat until $j < n$

Step24: increment j

Step25: initialize $k=0$

Step26: repeat until $k < n$

Step27: increment k

Step28: if repeat Step(28-32) until $rt[i].dist[j] > dmat[i][k] + rt[k].dist[j]$

Step29: initialize $rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j]$

Step30: initialize $rt[i].from[j] = k;$

Step31: increment count

Step32: end if

Step33: end do stmt

Step34: while (count!=0)

Step35: initialize $i=0$

Step36: repeat Steps(36-44)until $i < n$

Step37:increment i

Step38:write ' state values for router',i+1

Step39:initialize j=0

Step40:repeat Steps (40-43)until j<n

Step41:increment j

Step42:write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]

Step43:end

Step44:end

end

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
int dmat[20][20];
int n,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][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("\nState value for router %d is \n",i+1);
for(j=0;j<n;j++)
{
printf("\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n");
}

```


OUTPUT:

```
Turbo C++ IDE
Enter the number of nodes : 3
Enter the cost matrix :
0 2 4
2 0 5
4 5 0
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
node 3 via 3 Distance4
State value for router 2 is
node 1 via 1 Distance2
node 2 via 2 Distance0
node 3 via 3 Distance5
State value for router 3 is
node 1 via 1 Distance4
node 2 via 2 Distance5
node 3 via 3 Distance0
```

EXPERIMENT- 7

AIM: Write a Program to Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

```
#Create a simulator object
```

```
set ns [new Simulator]
```

```
#Open a trace file
```

```
set nt [open lab1.tr w]
```

```
$ns trace-all $nt
```

```
#Open a nam trace file
```

```
set nf [open lab1.nam W]
```

```
$ns namtrace-all $nf
```

```
# Create the nodes.
```

```
set n0 [$ns node]
```

```
set n1 [$ns node]
```

```
set n2 [$ns node]
```

```
set n3 [$ns node]
```

```
#Assign color to the packets.
```

```
$ns color 1 Red
```

```
$ns color 2 Blue
```

```
#Label the nodes
```

```
$n0 label "Source/udp0"
```

```
$n1 label "Source/udp1"
```

```
$n2 label "Router"
```

```
$n3 label "Destination/Null"
```

#Create links, vary bandwidth to check the number of packets dropped.

\$ns duplex-link \$n0 \$n2 10Mb 30ms DropTail

\$ns duplex-link \$n1 \$n2 10Mb 30ms DropTail

\$ns duplex-link \$n2 \$n3 1Mb 30ms DropTail

Set the queue size between the nodes

\$ns set queue-limit \$n0 \$n2 10

\$ns set queue-limit \$n1 \$n2 10

\$ns set queue-limit \$n2 \$n3 5

#Create and attach UDP agent to n0, n1 and null agent to n3.

set udp0 [new Agent/UDP]

\$ns attach-agent \$n0 \$udp0

set cbr0 [new Application/Traffic/CBR]

\$cbr0 attach-agent \$udp0

set udp1 [new Agent/UDP]

\$ns attach-agent \$n1 \$udp1

set cbr1 [new Application/Traffic/CBR]

\$cbr1 attach-agent \$udp1

set null3 [new Agent/Null]

\$ns attach-agent \$n3 \$null3

#Set udp0 packets to red color and udp1 packets to blue color

\$udp0 set class_1

\$udp1 set class_2

#Connect the agents.

\$ns connect \$udp0 \$null3

```
$ns connect $udp1 $null3
```

```
#Set the packet size to 500
```

```
$cbr1 set packetSize_ 500Mb
```

```
#Set the data rate of the packets. if the data rate is high then packets drops are high
```

```
$cbr1 set interval_ 0.005
```

```
#Finish Procedure
```

```
proc finish {} {
```

```
global ns nf nt
```

```
$ns flush-trace
```

```
exec nam lab1.nam &
```

```
close $nt
```

```
close $nf
```

```
exit0
```

```
}
```

```
$ns at 0.1 "$cbr0 start"
```

```
$ns at 0.1 "cbr1 start"
```

```
$ns at 10.0 "finish"
```

```
$ns run
```

```
AWK FILE:
```

```
BEGIN{
```

```
count=0;  
}
```

```
{  
if($1=="d")  
count++;  
}
```

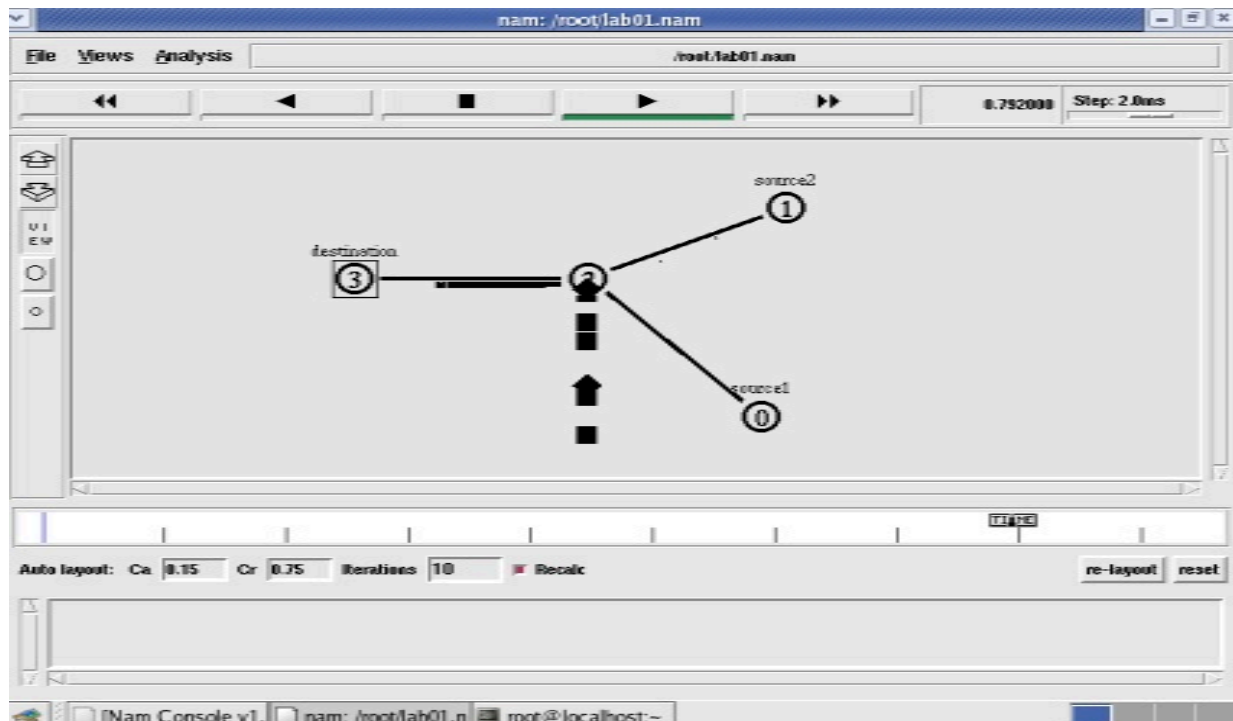
```

END{
    printf("No. of packets dropped is= %d\n", count);
}

```

Output:

root@localhost										
File	Edit	View	Terminal	Tab	Help					
+ 0.1	0	2	cbr	500	-----	0	0.0	3.0	0	0
- 0.1	0	2	cbr	500	-----	0	0.0	3.0	0	0
r 0.10108	0	2	cbr	500	-----	0	0.0	3.0	0	0
+ 0.10108	2	3	cbr	500	-----	0	0.0	3.0	0	0
- 0.10108	2	3	cbr	500	-----	0	0.0	3.0	0	0
+ 0.105	0	2	cbr	500	-----	0	0.0	3.0	1	1
- 0.105	0	2	cbr	500	-----	0	0.0	3.0	1	1
r 0.10608	0	2	cbr	500	-----	0	0.0	3.0	1	1
+ 0.10608	2	3	cbr	500	-----	0	0.0	3.0	1	1
- 0.10608	2	3	cbr	500	-----	0	0.0	3.0	1	1
+ 0.11	0	2	cbr	500	-----	0	0.0	3.0	2	2
- 0.11	0	2	cbr	500	-----	0	0.0	3.0	2	2
r 0.11108	0	2	cbr	500	-----	0	0.0	3.0	2	2
+ 0.11108	2	3	cbr	500	-----	0	0.0	3.0	2	2
- 0.11108	2	3	cbr	500	-----	0	0.0	3.0	2	2
+ 0.115	0	2	cbr	500	-----	0	0.0	3.0	3	3
- 0.115	0	2	cbr	500	-----	0	0.0	3.0	3	3
r 0.11608	0	2	cbr	500	-----	0	0.0	3.0	3	3
+ 0.11608	2	3	cbr	500	-----	0	0.0	3.0	3	3
- 0.11608	2	3	cbr	500	-----	0	0.0	3.0	3	3
+ 0.12	0	2	cbr	500	-----	0	0.0	3.0	4	4
- 0.12	0	2	cbr	500	-----	0	0.0	3.0	4	4
r 0.12108	0	2	cbr	500	-----	0	0.0	3.0	4	4
+ 0.12108	2	3	cbr	500	-----	0	0.0	3.0	4	4



```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# vi lab01.tcl  
[root@localhost ~]# awk -f PA1.awk lab01.tr  
cbr      139  
cbr      143  
cbr      130  
cbr      149  
cbr      151  
cbr      154  
cbr      139  
cbr      159  
cbr      163  
cbr      145  
cbr      169  
cbr      171  
cbr      174  
cbr      177  
cbr      179  
cbr      182  
The number of packets dropped =16  
[root@localhost ~]#
```

EXPERIMENT- 8

AIM: Write a program for congestion control using Leaky bucket algorithm.

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int no_of_queries, storage, output_pkt_size;
    int input_pkt_size, bucket_size, size_left;
    // initial packets in the bucket
    storage = 0;
    // total no. of times bucket content is checked
    no_of_queries = 4;
    // total no. of packets that can
    // be accommodated in the bucket
    bucket_size = 10;
    // no. of packets that enters the bucket at a time
    input_pkt_size = 4;
    // no. of packets that exits the bucket at a time
    output_pkt_size = 1;
    for(int i = 0; i < no_of_queries; i++) //space left
    {
        size_left = bucket_size - storage;
        if(input_pkt_size <= size_left)
        {
            // update storage
            storage += input_pkt_size;
            printf ("Buffer size= %d out of bucket size= %d\n", storage, bucket_size);
        }
        else
        {

```

```
printf("Packet loss = %d\n", (input_pkt_size-(size_left)));  
    // full size  
    storage=bucket_size;  
    printf("Buffer size= %d out of bucket size= %d\n", storage, bucket_size);  
}  
storage -= output_pkt_size;  
}  
return 0;  
}
```

Output :

Buffer size= 4 out of bucket size= 10

Buffer size= 7 out of bucket size= 10

Buffer size= 10 out of bucket size= 10

Packet loss = 3

Buffer size= 10 out of bucket size= 10

Difference between Leaky and Token buckets –

EXPERIMENT- 9

AIM: Write a program for frame sorting techniques used in buffers.

```
#include<stdio.h>
#include<string.h>
#define FRAM_TXT_SIZ 3
#define MAX_NOF_FRAM 127
char str[FRAM_TXT_SIZ*MAX_NOF_FRAM];
struct frame // structure maintained to hold frames
{ char text[FRAM_TXT_SIZ];
int seq_no;
}fr[MAX_NOF_FRAM], shuf_ary[MAX_NOF_FRAM];
int assign_seq_no() //function which splits message
{ int k=0,i,j; //into frames and assigns sequence no
for(i=0; i < strlen(str); k++)
{ fr[k].seq_no = k;
for(j=0; j < FRAM_TXT_SIZ && str[i]!='\0'; j++)
fr[k].text[j] = str[i++];
}
printf("\nAfter assigning sequence numbers:\n");
for(i=0; i < k; i++)
printf("%d:%s ",i,fr[i].text);
return k; //k gives no of frames
}
void generate(int *random_ary, const int limit) //generate array of random nos
{ int r, i=0, j;
while(i < limit)
{ r = random() % limit;
for(j=0; j < i; j++)
if( random_ary[j] == r )
Break;
```

```

if( i==j ) random_ary[i++] = r;
} }

void shuffle( const int no_frames ) // function shuffles the frames
{
int i, k=0, random_ary[no_frames];
generate(random_ary, no_frames);
for(i=0; i < no_frames; i++)
shuf_ary[i] = fr[random_ary[i]];
printf("\n\nAFTER SHUFFLING:\n");
for(i=0; i < no_frames; i++)
printf("%d:%s ",shuf_ary[i].seq_no,shuf_ary[i].text);
}

void sort(const int no_frames) // sorts the frames
{
int i,j,flag=1;
struct frame hold;
for(i=0; i < no_frames-1 && flag==1; i++) // search for frames in sequence
{
flag=0;
for(j=0; j < no_frames-1-i; j++) //(based on seq no.) and display
if(shuf_ary[j].seq_no > shuf_ary[j+1].seq_no)
{
hold = shuf_ary[j];
shuf_ary[j] = shuf_ary[j+1];
shuf_ary[j+1] = hold;
flag=1;
}
}
}

int main()
{
int no_frames,i;

```

```
printf("Enter the message: ");
gets(str);
no_frames = assign_seq_no();
shuffle(no_frames);
sort(no_frames);
printf("\n\nAFTER SORTING\n");
for(i=0;i<no_frames;i++)
printf("%s",shuf_ary[i].text);
printf("\n\n");
}
```

[root@localhost nwc]# ./a.out

Enter the message: Welcome To CMR Technical Campus

After assigning sequence numbers:

0:Wel 1:com 2:e T 3:o C 4:MRT 5:ech 6: ni 7:cal 8:Cam 9:Cam 10:pus

AFTER SHUFFLING:

1:com 4:MRT 9:Cam 5:ech 3:o C 10:pus 2:e T 6: ni 0:Wel 8:Cam 7:cal

AFTER SORTING

Welcome To CMR Technical Campus

EXPERIMENT- 10

AIM: Demonstrate how to run NMap scan.

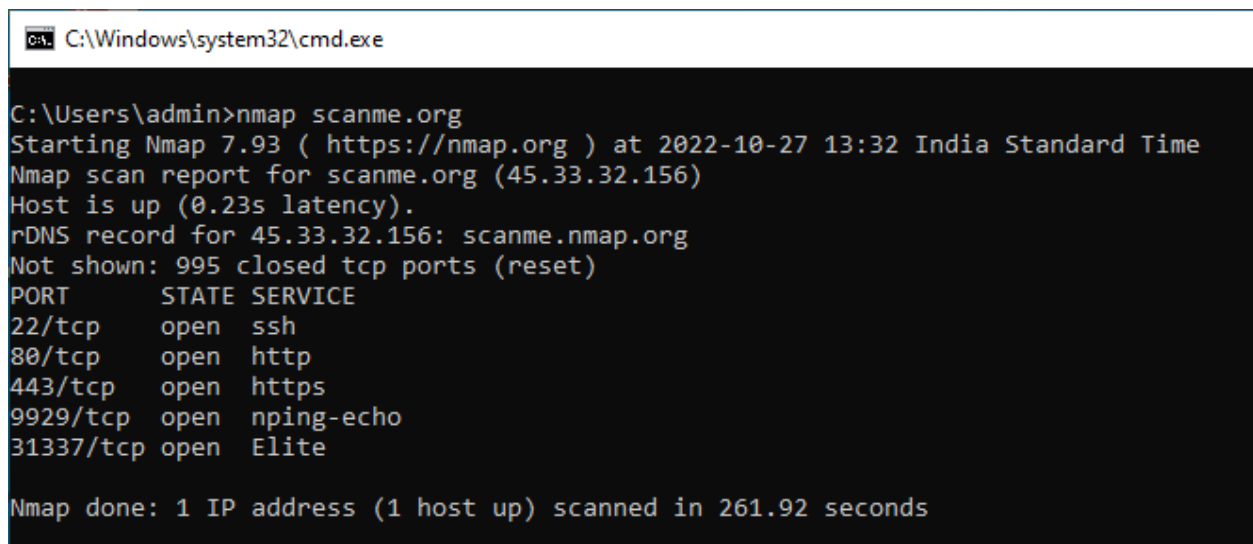
Nmap (<http://nmap.org>)

Nmap is a free, open source and multi-platform network security scanner used for network discovery and security auditing. Amongst other things, it allows you to create a network inventory, manage service upgrade schedules, monitor host or service uptime and scan for open ports and services on a host.

This post will focus on how to use Nmap to scan for open ports. Nmap can be extremely useful for helping you get to the root of the problem you are investigating, verify firewall rules or validate if your routing tables are configured correctly.

To get started, download and install Nmap from the nmap.org website and then launch a command prompt.

1. nmap.scanme.org is a server the NMAP team spun up to allow you to test tool functionality.



```
C:\Windows\system32\cmd.exe

C:\Users\admin>nmap scanme.org
Starting Nmap 7.93 ( https://nmap.org ) at 2022-10-27 13:32 India Standard Time
Nmap scan report for scanme.org (45.33.32.156)
Host is up (0.23s latency).
rDNS record for 45.33.32.156: scanme.nmap.org
Not shown: 995 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
9929/tcp  open  nping-echo
31337/tcp open  Elite

Nmap done: 1 IP address (1 host up) scanned in 261.92 seconds
```

When the scan is complete, you should see an Nmap scan report similar to the one shown in the image above. This confirms Nmap is installed and operating correctly.

You will notice the information returned is PORT | STATE | SERVICE. Before we take a deeper dive into the commands, it would be valuable to know what the different 'STATES' mean. The Nmap Reference Guide provides a pretty comprehensive explanation, but I'll give you a brief summary here.

STATE	Description
Open	The target port actively responds to TCP/UDP/SCTP requests.
Closed	The target port is active but not listening.
Filtered	A firewall or packet filtering device is preventing the port state being returned.
Unfiltered	The target port is reachable but Nmap cannot determine if it is open or closed.
Open/Filterec	Nmap cannot determine if the target port is open or filtered.
Closed/Filtere	Nmap cannot determine if the target port is closed or filtered.

Let us now look at some commands we can use for scanning open ports.

Nmap Port Scanning Commands

2. In any of the commands below, Nmap only shows you ports with an “Open” state.

nmap [ip_address]

nmap 172.16.131.2

C:\Windows\system32\cmd.exe

```
C:\Users\admin>nmap 172.16.131.2
Starting Nmap 7.93 ( https://nmap.org ) at 2022-10-27 13:45 India Standard Time
Nmap scan report for 172.16.131.2
Host is up (0.000060s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
902/tcp    open  iss-realsecure
912/tcp    open  apex-mesh
3389/tcp   open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
```

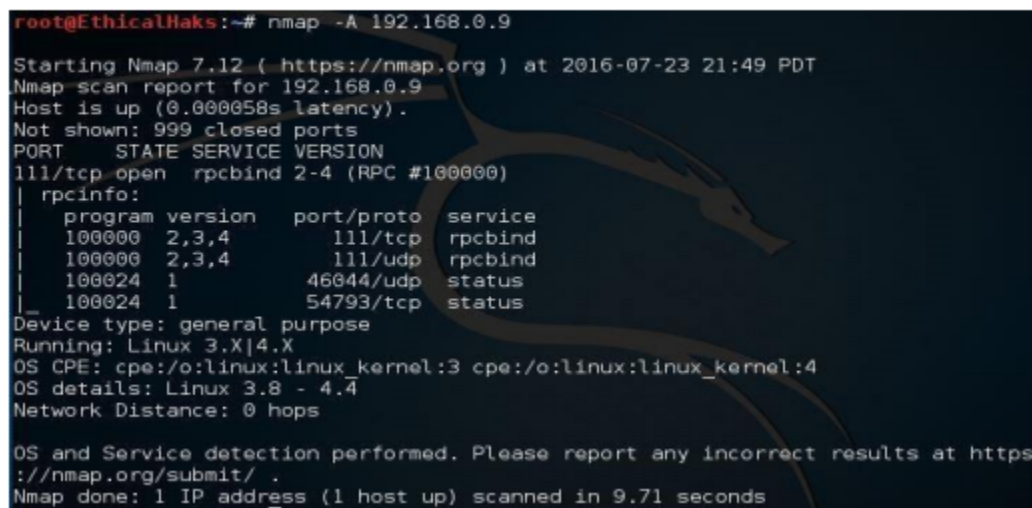
EXPERIMENT- 11

AIM: Demonstration of Operating System Detection using N map.

Detect OS and services

This is the command to scan and search for the OS (and the OS version) on a host. This command will provide valuable information for the enumeration phase of your network security assessment (if you only want to detect the operating system, type `nmap -O 192.168.0.9`):

`nmap -A 192.168.0.9`



```
root@EthicalHaks:~# nmap -A 192.168.0.9
Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:49 PDT
Nmap scan report for 192.168.0.9
Host is up (0.000058s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
111/tcp   open  rpcbind 2-4 (RPC #100000)
|_ rpcinfo:
|   program version  port/proto  service
|   100000   2,3,4      111/tcp    rpcbind
|   100000   2,3,4      111/udp    rpcbind
|   100024   1          46044/udp  status
|_   100024   1          54793/tcp  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.4
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.71 seconds
```

Standard service detection

This is the command to scan for running service. Nmap contains a database of about 2,200 well-known services and associated ports. Examples of these services are HTTP (port 80), SMTP

(port 25), DNS (port 53), and SSH (port 22):

`nmap -sV 192.168.0.9`

```
root@EthicalHaks:~# nmap -sV 192.168.0.9
```

```
Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:50 PDT
```

```
Nmap scan report for 192.168.0.9
```

```
Host is up (0.0000020s latency).
```

```
Not shown: 999 closed ports
```

```
PORT      STATE SERVICE VERSION
```

```
111/tcp   open  rpcbind 2-4 (RPC #100000)
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.78 seconds
```

EXPERIMENT- 12

AIM: Study of basic Network configuration commands and utilities to debug the network issues.

All commands related to Network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memory or permanent memory.

This commands includes

- Configuring the Router commands
- General Commands to configure network
- Privileged Mode commands of a router
- Router Processes & Statistics
- IP Commands
- Other IP Commands e.g. show ip route etc.

Procedure:

To do this EXPERIMENT- follows these steps:

ping:

ping(8) sends an ICMP ECHO_REQUEST packet to the specified host. If the host responds, you get an ICMP packet back. Sounds strange? Well, you can “ping” an IP address to see if the machine is alive. If there is no response, you know something is wrong.

Traceroute:

Tracert is a command which can show you the path a packet of information takes from your computer to one you specify. It will list all the routers it passes through until it reaches its destination, or fails to and is discarded. In addition to this, it will tell you how long each 'hop' from router to router takes.

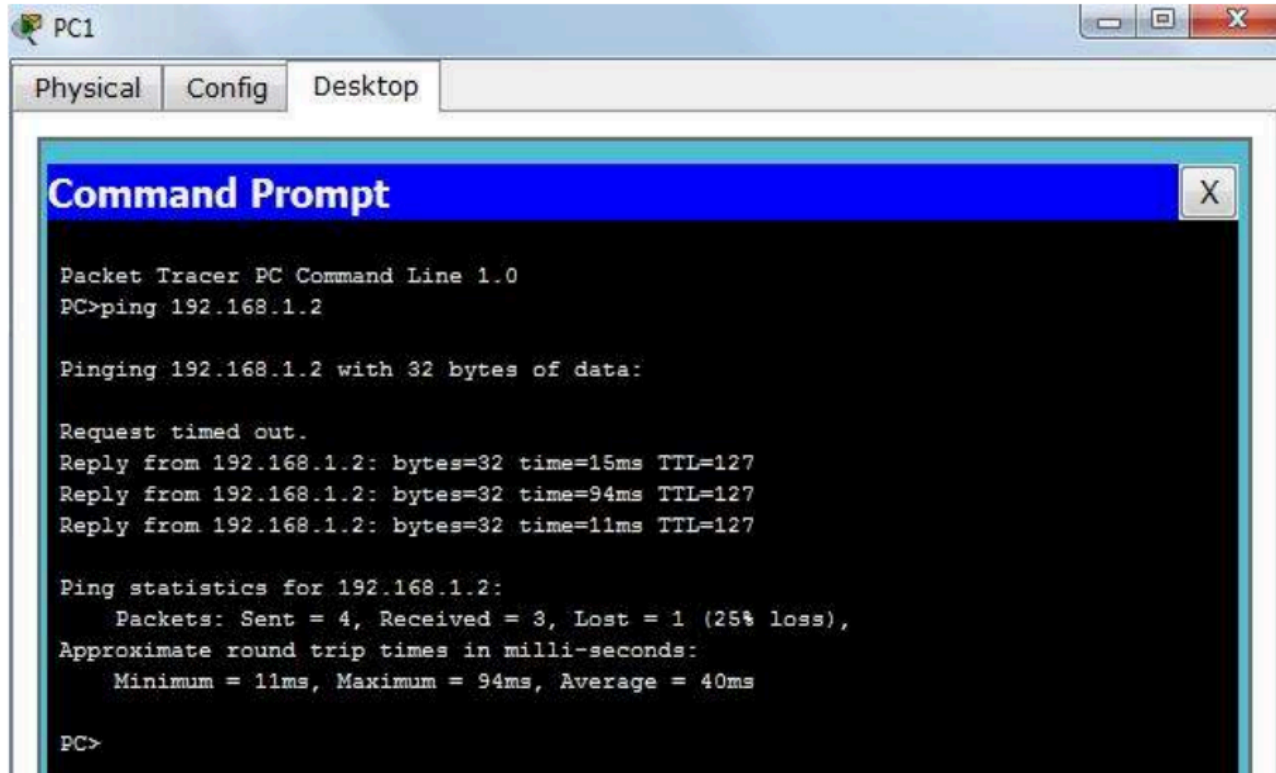
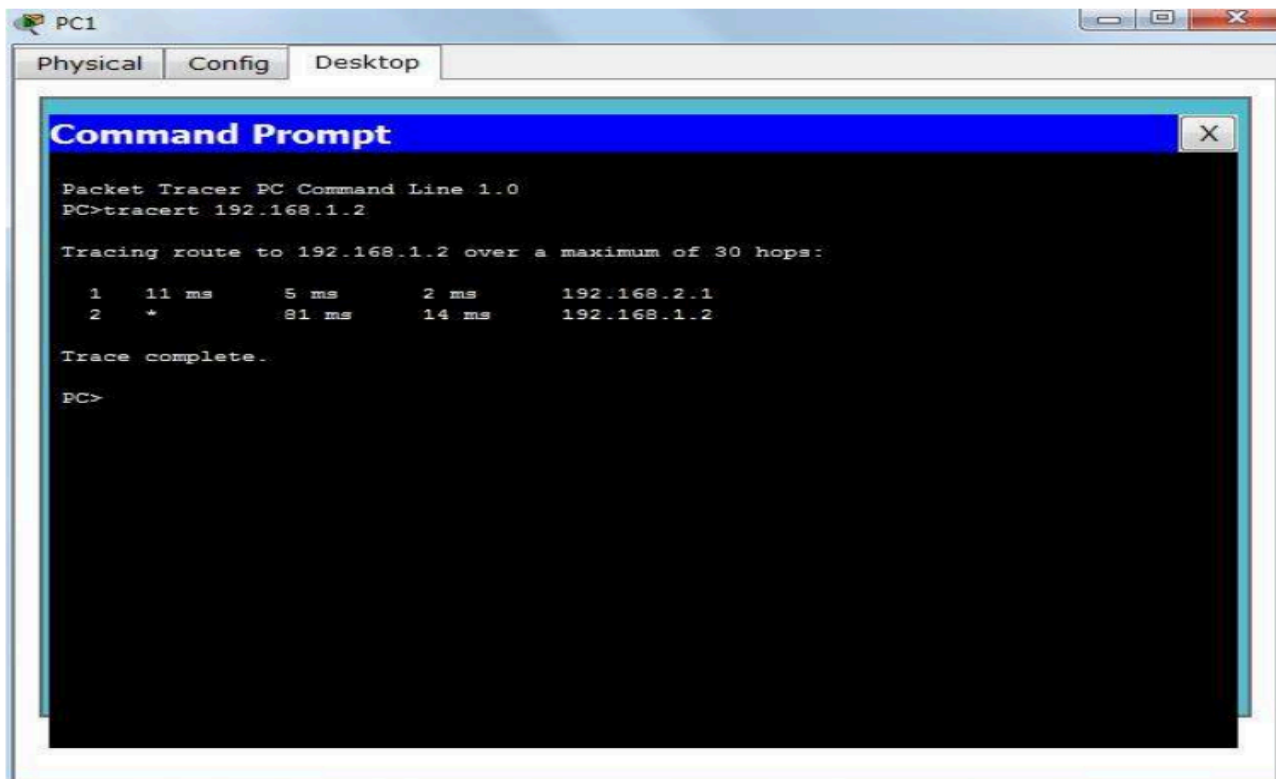
nslookup:

Displays information from Domain Name System (DNS) name servers.

NOTE :If you write the command as above it shows as default your pc's server name firstly.

pathping:

A better version of tracert that gives you statics about packet loss and latency



```
Administrator: C:\windows\system32\cmd.exe
C:\Users\lenovo>pathping 192.168.1.12
Tracing route to 192.168.1.12 over a maximum of 30 hops
  0  lenovo-PC.dronacharya [192.168.1.97]
  1  lenovo-PC.dronacharya [192.168.1.97]  reports: Destination host unreachable
.
Computing statistics for 25 seconds...
Hop  RTT      Source to Here   This Node/Link   Address
  0  ---      Lost/Sent = Pct  Lost/Sent = Pct  lenovo-PC.dronacharya [192.168.1.97]
  1  ---      100/ 100 =100%   100/ 100 =100%   |
  2  ---      0/ 100 = 0%      0/ 100 = 0%      lenovo-PC [0.0.0.0]
Trace complete.
C:\Users\lenovo>
```

Getting Help

In any command mode, you can get a list of available commands by entering a question mark (?).

Router>?

To obtain a list of commands that begin with a particular character sequence, type in those characters followed immediately by the question mark (?).

Router#co?

configure connect copy

To list keywords or arguments, enter a question mark in place of a keyword or argument. Include a space before the question mark.

Router#configure ?

memory Configure from NV memory

network Configure from a TFTP network host

terminal Configure from the terminal

You can also abbreviate commands and keywords by entering just enough characters to make the command unique from other commands. For example, you can abbreviate the show command to sh.

Configuration Files

Any time you make changes to the router configuration, you must save the changes to memory because if you do not they will be lost if there is a system reload or power outage. There are two types of configuration files: the running (current operating) configuration and the startup configuration.

Use the following privileged mode commands to work with configuration files.

- configure terminal – modify the running configuration manually from the terminal.
- show running-config – display the running configuration.

- show startup-config – display the startup configuration.
- copy running-config startup-config – copy the running configuration to the startup configuration.
- copy startup-config running-config – copy the startup configuration to the running configuration.
- erase startup-config – erase the startup-configuration in NVRAM.
- copy tftp running-config – load a configuration file stored on a Trivial File Transfer Protocol (TFTP) server into the running configuration.
- copy running-config tftp – store the running configuration on a TFTP server.

IP Address Configuration

Take the following steps to configure the IP address of an interface.

Step 1: Enter privileged EXEC mode:

Router>enable password

Step 2: Enter the configure terminal command to enter global configuration mode.

Router#config terminal

Step 3: Enter the interface type slot/port (for Cisco 7000 series) or interface type port (for Cisco 2500 series) to enter the interface configuration mode.

Example:

Router (config)#interface ethernet 0/1

Step 4: Enter the IP address and subnet mask of the interface using the ip address ip address subnet mask command.

Example,

Router (config-if)#ip address 192.168.10.1 255.255.255.0

Step 5: Exit the configuration mode by pressing Ctrl-Z

Router(config-if)#[Ctrl-Z]

Result:

The network configuration was done successfully.