# VELAMMAL INSTITUTE OF TECHNOLOGY

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**LABORATORY MANUAL**

# EC 8563 – COMMUNICATION NETWORKS LABORATORY

**(Regulation – 2017)**

| | | |
|---|---|---|
| **Semester/Branch** | **:** | **V semester ECE** |
| **Academic Year** | **:** | **2020 -21 (ODD)** |
| **Prepared By** | **:** | **Mr.K.BALACHANDER** |

## LIST OF EXPERIMENTS:

| S.NO | DATE | TITLE | REMARKS |
|---|---|---|---|
| 1. | | Implementation of Error Detection / Error Correction Techniques | |
| 2. | | Implementation of Stop and Wait Protocol and sliding window | |
| 3. | | Implementation and study of Goback-N and selective repeat protocols | |
| 4. | | Implementation of High Level Data Link Control | |
| 5. | | Implementation of IP Commands such as ping, Traceroute, nslookup. | |
| 6. | | Implementation of IP address configuration. | |
| 7. | | To create scenario and study the performance of network with CSMA / CA protocol and compare with CSMA/CD protocols. | |
| 8. | | Network Topology - Star, Bus, Ring | |
| 9. | | Implementation of distance vector routing algorithm | |
| 10. | | Implementation of Link state routing algorithm | |
| 11. | | Study of Network simulator (NS) and simulation of Congestion Control Algorithms using NS | |
| 12. | | Implementation of Encryption and Decryption Algorithms using any programming language | |

| EXPT NO:1 | IMPLEMENTATION OF ERROR DETECTION / ERROR CORRECTION TECHNIQUES USING C PROGRAMMING LANGUAGE |
|---|---|
| DATE: | |

**Aim:**

To implement a Error Detection and Error Correction Techniques using C Programming language.

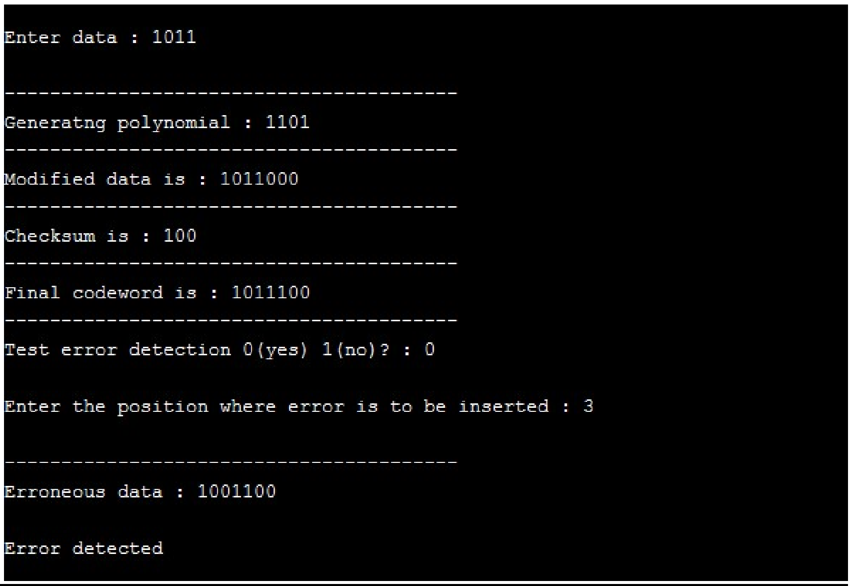
**Procedure:**

1. Open the text editor and type the program
2. Save the program with .c extension in any directory (Desktop)
3. Map the directory before compiling **cd Desktop** in terminal
4. Open the terminal and compile the program using **gcc filename.c**
5. If you have errors in the program, correct the errors and compile the program successfully.
6. Use the command **./a.out** in the terminal to execute the program.
7. If successfully executed, give inputs and note down the results

***Program for Error Detection – CRC (Cyclic Redundancy Check)***

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)
char t[28],cs[28],g[]="1101";
int a,e,c;
void xor()
{
for(c = 1;c < N; c++)
cs[c] = (( cs[c] == g[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
cs[e]=t[e];
do{
if(cs[0]=='1')
xor();
for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}
while(e<=a+N-1);
}
int main()
{
printf("\nEnter data : ");
scanf("%s",t);
printf("\n----------------------------------------");
printf("\nGeneratng polynomial : %s",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\n----------------------------------------");
printf("\nModified data is : %s",t);
printf("\n----------------------------------------");
crc();
printf("\nChecksum is : %s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("\n----------------------------------------");
printf("\nFinal codeword is : %s",t);
printf("\n----------------------------------------");
printf("\nTest error detection 0(yes) 1(no)? : ");
scanf("%d",&e);
if(e==0)
{
do{
printf("\nEnter the position where error is to be inserted : ");
scanf("%d",&e);
}while(e==0 || e>a+N-1);
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n----------------------------------------");
printf("\nErroneous data : %s\n",t);
}
crc();
for(e=0;(e<N-1) && (cs[e]!='1');e++);
if(e<N-1)
printf("\nError detected\n\n");
else
printf("\nNo error detected\n\n");
printf("\n----------------------------------------\n");
return 0;
}
```

**OUTPUT**

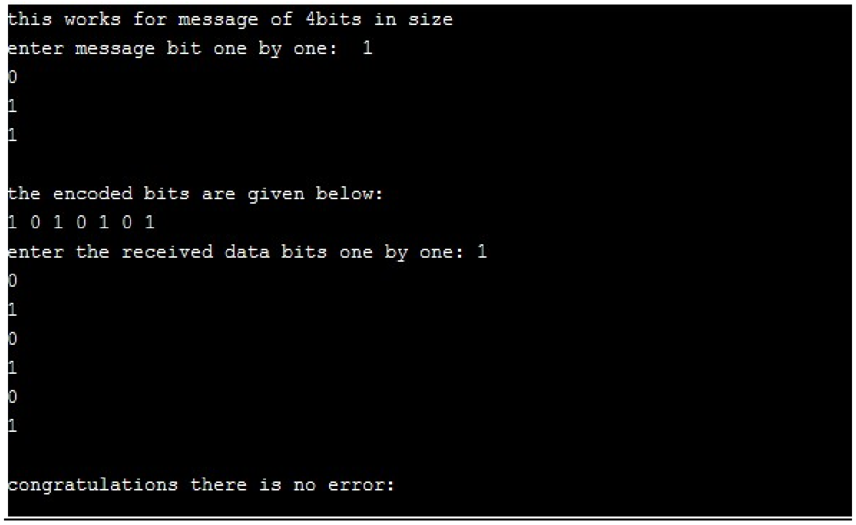


```
Enter data : 1011

----------------------------------------
Generatng polynomial : 1101
----------------------------------------
Modified data is : 1011000
----------------------------------------
Checksum is : 100
----------------------------------------
Final codeword is : 1011100
----------------------------------------
Test error detection 0(yes) 1(no)? : 0

Enter the position where error is to be inserted : 3

----------------------------------------
Erroneous data : 1001100

Error detected
```

***Program for Error Correction – Hamming Code***

```
#include<stdio.h>
#include<conio.h>
void main()
{
int data[7],rec[7],i,c1,c2,c3,c;
printf("this works for message of 4bits in size \nenter message bit one by one:  ");
scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);
data[6]=data[0]^data[2]^data[4];
data[5]=data[0]^data[1]^data[4];
data[3]=data[0]^data[1]^data[2];
printf("\nthe encoded bits are given below: \n");
for (i=0;i<7;i++) {
printf("%d ",data[i]);
}
printf("\nenter the received data bits one by one: ");
for (i=0;i<7;i++) {
scanf("%d",&rec[i]);
}
c1=rec[6]^rec[4]^rec[2]^rec[0];
c2=rec[5]^rec[4]^rec[1]^rec[0];
c3=rec[3]^rec[2]^rec[1]^rec[0];
c=c3*4+c2*2+c1 ;
if(c==0)
{
printf("\ncongratulations there is no error: ");
} else
{
printf("\n erron on the postion: %d\n the correct message is \n",c);
if(rec[7-c]==0)
rec[7-c]=1; else
rec[7-c]=0;
for (i=0;i<7;i++) {
printf("%d ",rec[i]);
}
}
getch();
}
```

**OUTPUT**



```
this works for message of 4bits in size
enter message bit one by one:  1
0
1
1

the encoded bits are given below:
1 0 1 0 1 0 1
enter the received data bits one by one: 1
0
1
0
1
0
1

congratulations there is no error:
```

**RESULT:**
Thus the error detection and error correction techniques using C programming language were successfully implemented and the outputs are verified.

|  |  |
|---|---|
| EXPT NO:2 | IMPLEMENTATION OF STOP AND WAIT PROTOCOL AND SLIDING WINDOW |
| DATE: |  |

**Aim:**

To implement stop and wait protocol using Network Simulator (NS-2) and sliding window protocol using C Program.

**Procedure:**

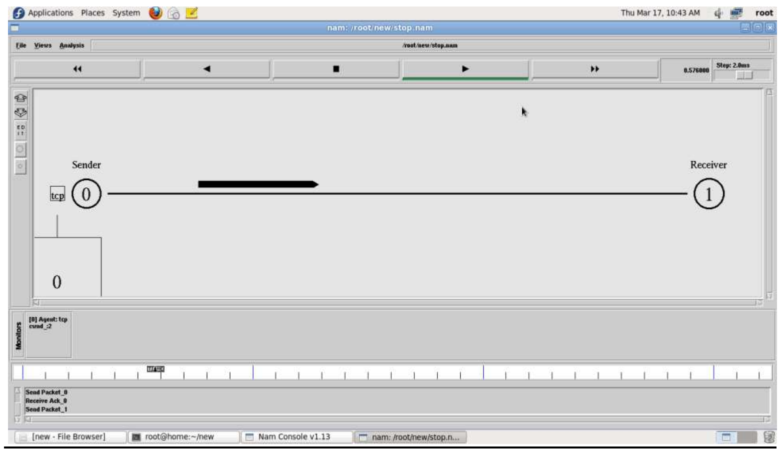
1. Open a Text editor and type the program
2. Save the file with .tcl extension
3. Open the terminal and map the file location
4. Now compile the program in Ns-2 using ns filename.tcl
5. If any errors will be displayed in the terminal, if no errors the output will be displayed in the NAM (Network Animator) window

***Program for Stop and Wait Protocol using NS-2:***

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

set f [open stopwait.tr w]
$ns trace-all $f
et nf [open stopwait.nam w]
$ns namtrace-all $nf

$ns duplex-link $n0 $n2 0.2Mb 20ms DropTail
$ns duplex-link $n1 $n2 0.2Mb 20ms DropTail
$ns duplex-link $n2 $n3 0.2Mb 20ms DropTail
$ns duplex-link $n3 $n4 0.2Mb 20ms DropTail
$ns duplex-link $n3 $n5 0.2Mb 20ms DropTail

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

$ns queue-limit $n0 $n2 10
```

```
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP]
$tcp set window 1
```

**<u>OUTPUT FOR STOP AND WAIT PROTOCOL</u>**
**(SCREENSHOT MAY BE PASTED)**



```
Send Packet_0
Receive Ack_0
Send Packet_1
```

**Algorithm Steps:**

Step 1: start the program.

Step 2: Open the input file in read mode.

Step 3: Read the size of the window

Step 4: Select randomly the number of packets is to be transferred.

Step 5: Read the content of the input file.

Step 6: Transfer the packet until it reaches the maximum defined size.

Step 7: Resume the window size and repeat the above two steps until packets in.

Step 8: Close the file.

Step 9: Stop the program.

**Program for sliding window protocol :**

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

```
int w,i,f,frames[50];

printf("\n enter the window size:");
scanf("%d",&w);
printf("\n enter the no of frames to transmit:");
scanf("%d",&f);
printf("\n enter%dframes:",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]);
printf("\n frames will be sent in the following manner assuming that no frame has been  corrupted:");
printf("\n after sending the frames the sender waits for acknowledgement from the receiver");
for(i=1;i<=f;i++)
{
if(i%w==0)
{
printf("%d\n",frames[i]);
printf("\n acknowledgement of above frames sent is received by the sender"); }
else
printf("\n%d",frames[i]);
}
if(f%w!=0)
printf("acknowledgement received by sender!");
getch();
}
```

**Output :**

```
enter the window size:2

 enter the no of frames to transmit:2

 enter2 frames:10
11

 frames will be sent in the following manner assuming that no frame has been  corrupted:
 after sending the frames the sender waits for acknowledgement from the
receiver
1011

 acknowledgement of above frames sent is received by the sender
```

**RESULT**

Thus the working of sliding window protocol was implemented and understood and  the working of simple error control functionality was implemented using the simple Stop and wait protocol.

| EXPT NO:3 | IMPLEMENTATION AND STUDY OF GOBACK-N AND SELECTIVE REPEAT PROTOCOLS. |
|---|---|
| DATE: | |

**AIM:**
To write a C program to demonstrate the working of selective repeat and Goback-N Protocol.
**ALGORITHM:**
1. Start the program
2. Include the required header files.
3. Declare and Initialize the variables and file pointers.
4. Enter the choice to choose protocol - Selective Retransmission or Go-Back-N
5. Get the size of the data to be sent
6. Check for the acknowledgement. If yes, then enter the data one by one\.
7. If the acknowledgement is not received, then ask enter the position that has be sent again.
8. If the protocol is Selective Retransmission, then the data of that particular position will be sent.
9. If the protocol is Go-Back-N, then all the data from the position will be sent**.**
10. It the data are transmitted then choose the Exit option to quit the program.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
int main()
{
int data[5],rec[5];
int n,m,a,i,c=0,j;
char r[5],ch='y';
clrscr();
do
{
printf("\n Enter the choice:");
printf("\n 1.Select Retransmission\n");
printf("\n 2.Go back n");
printf("\n 3.Exit\n");
scanf("%d",&a);
printf("\n Enter the size of data: ");
scanf("%d",&n);
printf("\n Enter the data one by one :");
for(i=0;i<n;i++)
{
scanf("%d", &data[i]);
}
switch(a)
{
case 1:
{
printf("\n data is ready to send");
for(i=0;i<n;i++)
```

```
{
rec[i]=data[i];
printf("\n %d is sent \n", data[i]);
printf("\nIf the data is received (y/n)?\n");
scanf("%s", r);
if(strcmp(r,"n"))
printf("\n ack is rec %d\n",i+1);
else
c++;
}
if(c!=0)
{
printf("\n enter position of data to send:");
scanf("%d",&m);
}
while(m>n)
{
printf("\n Wrong choice");
printf("\n Enter position of data again:");
scanf("%d",&m);
}
if(a==1)
printf("\n %d is sent again\n", data[m-1]);
getch();
}
break;
case 2:
{
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
rec[j]=data[j];
printf("\n %d is sent\n",data[j]);
printf("\nIf the data is received (y/n)?\n");
scanf("%s",r);
if(strcmp(r,"n"))
printf("\n ack is rec %d\n",j);
else
c++;
}
if(c!=0)
{
printf("\n enter starting position of data to be sent");
scanf("%d",&m);
for(i=m;i<n;i++)
{
printf("\n %d is sent",data[i]);
}
}
}
}
getch();
break;
default:exit(0);
}}
while(ch=='y');
return(0);
}
```

**OUTPUT:**

```
Enter the choice:
 1.Select Retransmission
 2.Go back n
 3.Exit
2
 Enter the size of data: 3
 Enter the data one by one :10
11
10
 10 is sent
If the data is received (y/n)?
y
 ack is rec 0
 11 is sent
If the data is received (y/n)?
y
 ack is rec 1
 10 is sent
If the data is received (y/n)?
y
 ack is rec 2
 10 is sent
If the data is received (y/n)?
n
 11 is sent
If the data is received (y/n)?
y
 ack is rec 1
 10 is sent
If the data is received (y/n)?
y
 ack is rec 2
 enter starting position of data to be sent1
 11 is sent
 10 is sent
```

```
Enter the choice:
1.Select Retransmission
2.Go back n
3.Exit
```

**RESULT:**

Thus the working of selective repeat and GoBack n were implemented and understood.

| EXPT NO:4 | IMPLEMENTATION OF HIGH LEVEL DATA LINK CONTROL |
|---|---|
| DATE: | |

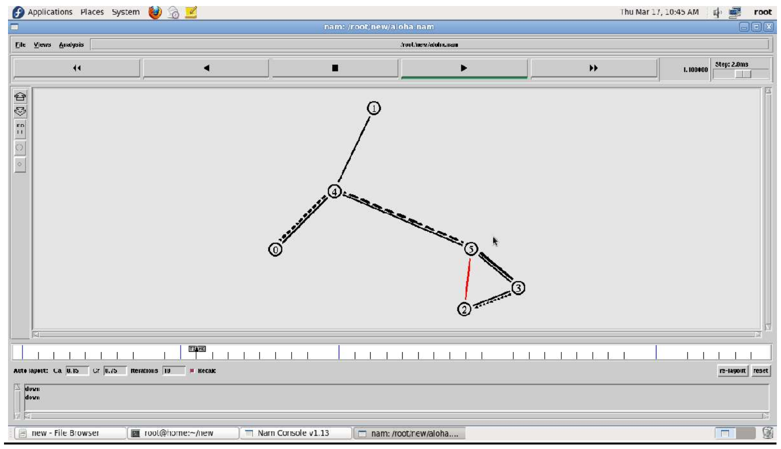
**AIM:**
To study and implement the concept and different frames of HDLC protocol.

**ALGORITHM:**
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes
6. Setup UDP Connection between n(0) and n(2)
7. Apply CBR Traffic over UDP
8. Choose distance vector routing protocol as a high level data link control.
9. Make any one of the links to go down to check the working nature of HDLC
10. Schedule events and run the program.

**PROGRAM**
```
set ns [new Simulator]
#Tell the simulator to use dynamic routing
$ns rtproto DV
$ns macType Mac/Sat/UnslottedAloha #Open the nam trace file
set nf [open aloha.nam w]
$ns namtrace-all $nf #Open the output files set f0 [open aloha.tr w]
$ns trace-all $f0
#Define a finish procedure proc finish {} {
global ns f0 nf
$ns flush-trace #Close the trace file close $f0
close $nf
exec nam aloha.nam & exit 0
}
# Create six nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set
n5 [$ns node]
# Create duplex links between nodes with bandwidth and distance
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 1Mb 1ms DropTail
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link $n2 $n3 1Mb 50ms DropTail
# Create a duplex link between nodes 4 and 5 as queue position
$ns duplex-link-op $n4 $n5 queuePos 0.5 #Create a UDP agent and attach it to node n(0) set udp0 [new
Agent/UDP]
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0 set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
```

```
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n(2) set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n5 $n2
$ns rtmodel-at 2.0 up $n5 $n2
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish" #Run the simulation
$ns run
```

**<u>OUTPUT</u>**



```
down
down
```

**RESULT:**
Thus the HDLC is studied and simulated.

| EXPT NO:5 | IMPLEMENTATION OF IP COMMANDS SUCH AS PING, TRACEROUTE, NSLOOKUP. |
|---|---|
| DATE: | |

**AIM:** To implement IP Commands such as ping, Traceroute, nslookup.

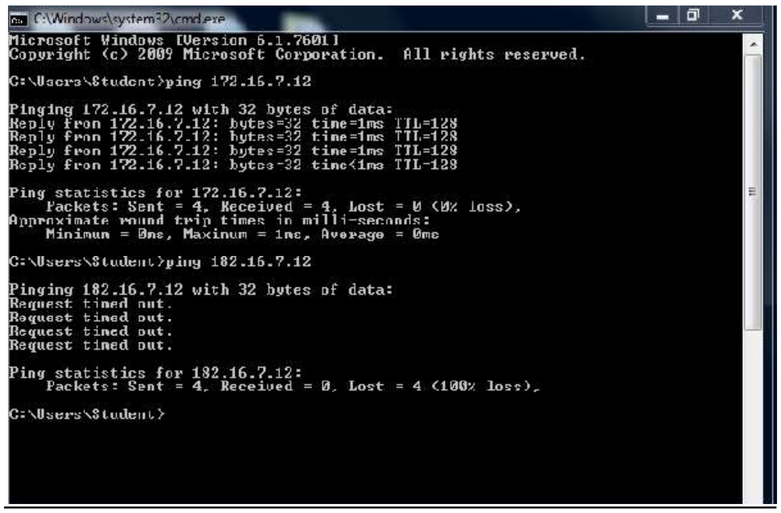
**IP Commands**
**(i)Ping**
Ping is a standard application found on most laptop and desktop computers. Apps that support ping can also be installed on smartphones and other mobile devices. Additionally, websites that support Internet speed test services often include ping as one of their features.
A ping utility sends test messages from the local client to a remote target over the TCP/IP network connection. The target can be a Web site, a computer, or any other device with an IP address. Besides determining whether the remote computer is currently online, ping also provides indicators of the general speed or reliability of network connections.
**Running Ping:**
Microsoft Windows, Mac OS X, and Linux provide command line ping programs that can be run from the operating system shell. Computers can be pinged by either IP address or by name.
**To ping a computer by IP address:**

- Open a shell prompt (in Microsoft Windows, the Command Prompt or MS-DOS Prompt on the Start Menu).
- Type ping followed by a space and then the IP address.
- Press the Enter (or Return) key.

**Interpreting the Results of Ping**

- **Reply from**: By default, Microsoft Windows ping sends a series of four messages to the address. The program outputs a confirmation line for each response message received from the target computer.
- **Bytes**: Each ping request is 32 bytes in size by default.
- **Time:** Ping reports the amount of time (in milliseconds) between the sending of requests and receipt of responses.
- **TTL (Time-to-Live):** A value between 1 and 128, TTL can be used to count how many different networks the ping messages passed through before reaching the target computer. A value of 128 indicates the device is on the local network, with 0 other networks in between.

<u>**OUTPUT**</u>



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Student>ping 172.16.7.12

Pinging 172.16.7.12 with 32 bytes of data:
Reply from 172.16.7.12: bytes=32 time=1ms TTL=128
Reply from 172.16.7.12: bytes=32 time=1ms TTL=128
Reply from 172.16.7.12: bytes=32 time=1ms TTL=128
Reply from 172.16.7.12: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.7.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\Student>ping 182.16.7.12

Pinging 182.16.7.12 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 182.16.7.12:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\Student>
```

**(ii) Traceroute**

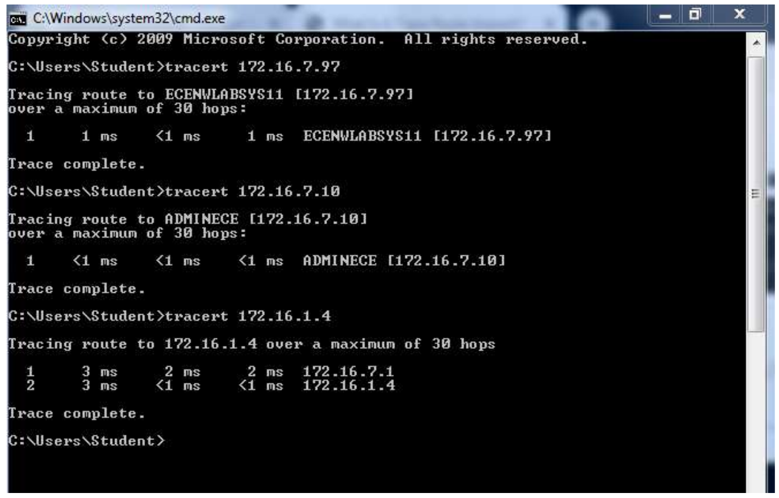
A *traceroute* is a function which traces the path from one network to another. It allows us to diagnose the source of many problems.

**To run traceroute on Windows:**

**Go To START > RUN**

- • To open the command prompt type **cmd** and press the Enter key. This will bring up a command prompt window. It has a line that looks like this: **C:\Documents and Settings\yourname> _** with a cursor blinking next to the ">" symbol.
- • In the command prompt, type: **tracert hostname** where **hostname** is the name of the server connection you are testing. (IP address)
- • You may have to wait up to a minute or more for the test to complete. It will generate a list of the connections along the way and some information about the speed of the steps along the way.
- • It sends us the complete results (every line) for analysis.
-

**OUTPUT**



```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Student>tracert 172.16.7.97

Tracing route to ECENWLABSYS11 [172.16.7.97]
over a maximum of 30 hops:

  1     1 ms    <1 ms     1 ms  ECENWLABSYS11 [172.16.7.97]

Trace complete.

C:\Users\Student>tracert 172.16.7.10

Tracing route to ADMINECE [172.16.7.10]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  ADMINECE [172.16.7.10]

Trace complete.

C:\Users\Student>tracert 172.16.1.4

Tracing route to 172.16.1.4 over a maximum of 30 hops

  1     3 ms     2 ms     2 ms  172.16.7.1
  2     3 ms    <1 ms    <1 ms  172.16.1.4

Trace complete.

C:\Users\Student>
```

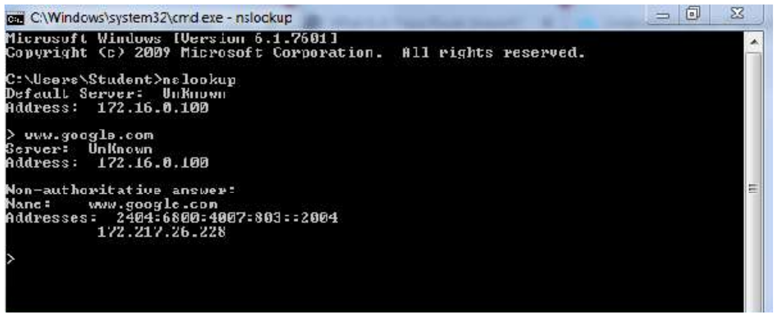
**iii) Nslookup**

*nslookup is a network administration command-line tool available for many computer operating systems.*

- • The main use of **nslookup** is for troubleshooting DNS related problems.
- • Nslookup can be used in **interactive** and **non-interactive** mode.
- • To use in interactive mode **type nslookup** at the command line and hit return.

To run nslookup on Windows:
1. Go to Start > Run and type cmd.
2. At a command prompt, type nslookup, and then press Enter.
3. Type server <IP address>;,where IP address is the IP address of your external DNS server
4. Type set q=MX, and then press Enter
5. Type <domain name>, where domain name is the name of your domain, and then press Enter.

**OUTPUT**



C:\Windows\system32\cmd.exe - nslookup

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Student>nslookup
Default Server:  UnKnown
Address:  172.16.0.100

> www.google.com
Server:  UnKnown
Address:  172.16.0.100

Non-authoritative answer:
Name:    www.google.com
Addresses:  2404:6800:4007:803::2004
          172.217.26.228

>
```

**Result:**

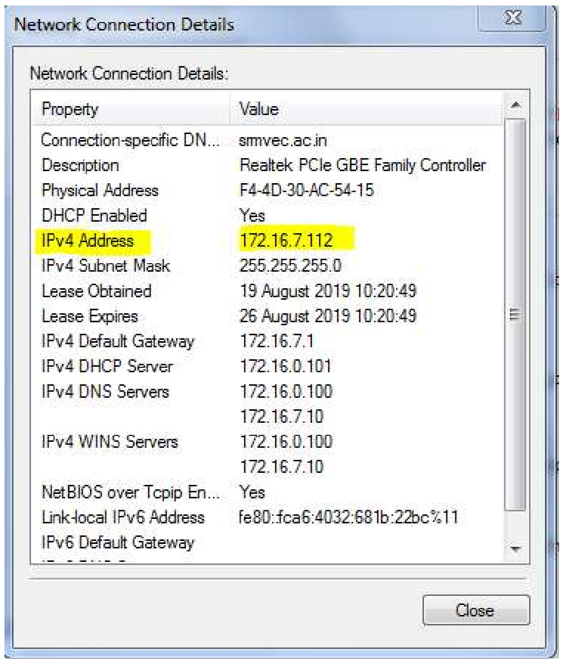
Thus IP commands were implemented and analyzed the results of ping, Traceroute and nslookup commands

| EXPT NO:6 | IMPLEMENTATION OF IP ADDRESS CONFIGURATION. |
|---|---|
| DATE: | |

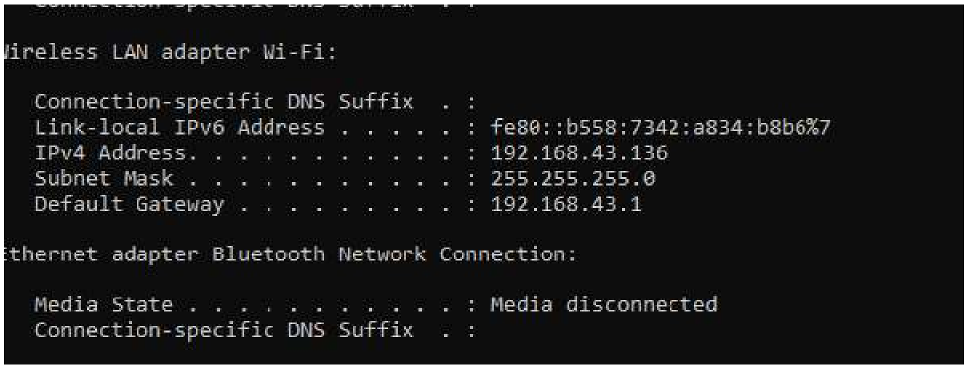
**Aim:**
To implement IP address and determine IP address of the device.

**A. IP Address Configuration:**
An Internet Protocol (IP) address is a unique number assigned to every device on a network. Just as a street address determines where a letter should be delivered, an IP address identifies computers on the Internet. Network devices use IP addresses to communicate with each other.
IP addresses are required by any network adapter on any computer that needs to connect to the Internet or another computer. Addresses are given out to network computers in one of two manners, dynamically or statically.
To set a static IP address in Windows 7, 8, and 10:

- • Click Start Menu > Control Panel > Network and Sharing Center or Network and Internet > Network and Sharing Center.
- • Click on Local Area Connection.
- • Click Details.
- • View for the Internet Protocol Version 4 (TCP/IPv4) address.



| Property | Value |
|---|---|
| Connection-specific DN... | smvec.ac.in |
| Description | Realtek PCIe GBE Family Controller |
| Physical Address | F4-4D-30-AC-54-15 |
| DHCP Enabled | Yes |
| IPv4 Address | 172.16.7.112 |
| IPv4 Subnet Mask | 255.255.255.0 |
| Lease Obtained | 19 August 2019 10:20:49 |
| Lease Expires | 26 August 2019 10:20:49 |
| IPv4 Default Gateway | 172.16.7.1 |
| IPv4 DHCP Server | 172.16.0.101 |
| IPv4 DNS Servers | 172.16.0.100 |
| | 172.16.7.10 |
| IPv4 WINS Servers | 172.16.0.100 |
| | 172.16.7.10 |
| NetBIOS over Tcpip En... | Yes |
| Link-local IPv6 Address | fe80::fca6:4032:681b:22bc%11 |
| IPv6 Default Gateway | |



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b558:7342:a834:b8b6%7
   IPv4 Address. . . . . . . . . . . : 192.168.43.136
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.43.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

**RESULT:**
**Thus IP address implemented and determined IP address of the device.**

| EXPT NO:7 | TO CREATE SCENARIO AND STUDY THE PERFORMANCE OF NETWORK WITH CSMA / CA PROTOCOL AND COMPARE WITH CSMA/CD PROTOCOLS. |
|---|---|
| DATE: | |

**AIM:**
To create scenario and study the performance of CSMA / CD protocol through simulation.

**SOFTWARE REQUIREMENTS:**
Ns-2

**ALGORITHM:**

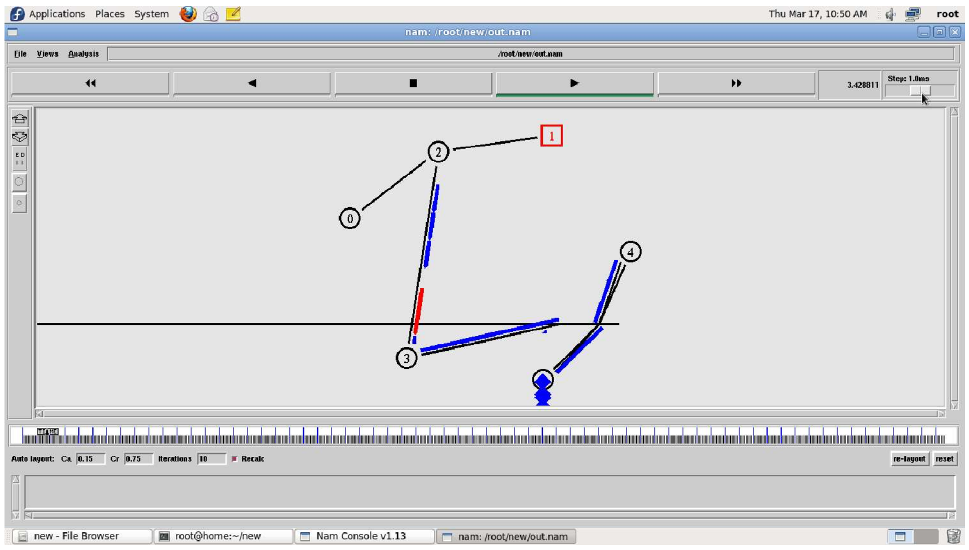
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between n(0) and n(4)
7. Apply FTP Traffic over TCP
8. Setup UDP Connection between n(1) and n(5)
9. Apply CBR Traffic over UDP.
10. Apply CSMA/CA and CSMA/CD mechanisms and study their performance
11. Schedule events and run the program.

PROGRAM:
CSMA/CA

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red #Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file set file2 [open out.nam w]
$ns namtrace-all $file2 #Define a 'finish' procedure proc finish {} {
global ns file1 file2
$ns flush-trace close $file1 close $file2
exec nam out.nam & exit 0
}
#Create six nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set
n5 [$ns node]
$n1 color red
$n1 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Ca Channel]
```

```
Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP connection set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP #Setup a UDP connection set udp [new Agent/UDP]
$ns attach-agent $n1 $udp set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$ns at 125.0 "finish"
$ns run
```

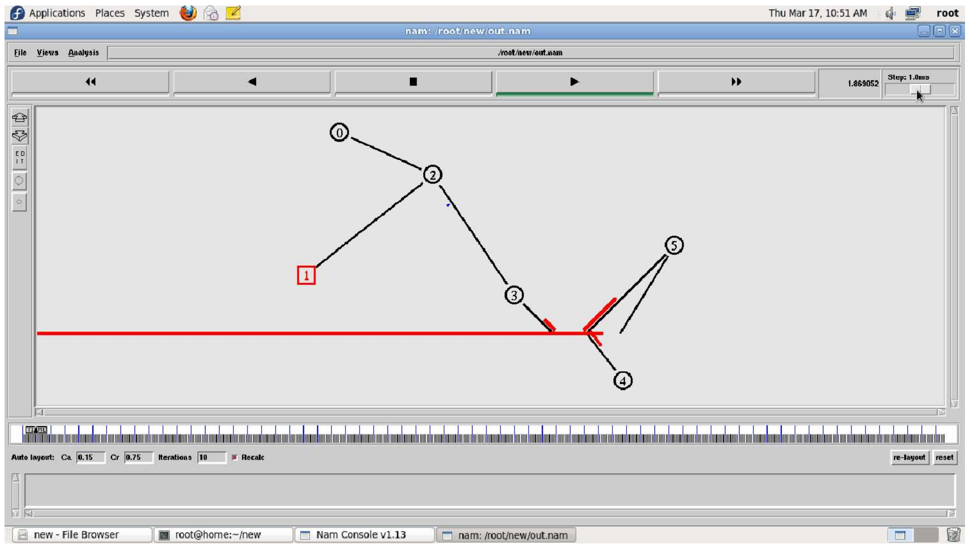
OUTPUT:



CSMA/CD

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red #Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file set file2 [open out.nam w]
$ns namtrace-all $file2 #Define a 'finish' procedure proc finish {} {
global ns file1 file2
$ns flush-trace close $file1 close $file2
exec nam out.nam & exit 0
}
#Create six nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set
n5 [$ns node]
$n1 color red
$n1 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
```

OUTPUT



**RESULT:**
Thus CSMA/CA and CSMA/CD were implemented and simulated using NS 2.

| | |
|---|---|
| EXPT NO:8 | NETWORK TOPOLOGY - STAR, BUS, RING |
| DATE: | |

**AIM:**
To build and simulate a network under different topologies like Star,bus, mesh and ring, and observe the performance parameters.

**REQUIREMENTS:**
Operating System : Windows NT/2000/XP or LINUX
Programming Tool : Network Simulator (NS2)

**Algorithm:**
1. Open a terminal and type the TCL file in the vim editor with the command “vifilename.tcl” and save it.
2. Run the saved file with the command “ns filename.tcl”.
3. If any errors, edit them in the vim editor and rerun it again.
4. The output is displayed in the nam console.

**STAR TOPOLOGY**
```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 shape square
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
```

```
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

**BUS TOPOLOGY**

```
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 5
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

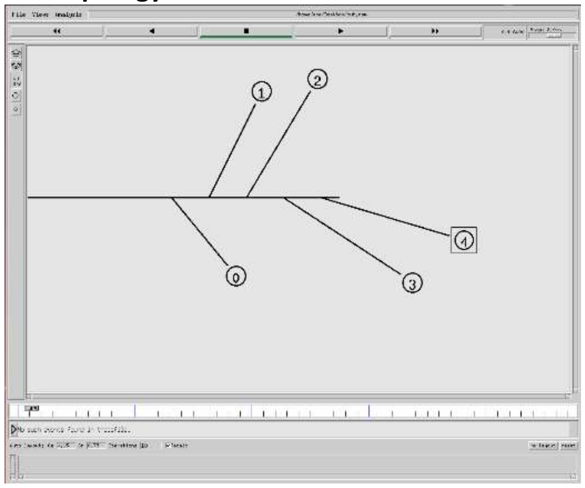
**RING TOPOLOGY**

```
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

```
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
MESH TOPOLOGY
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
# source and attach it to tcp0 set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

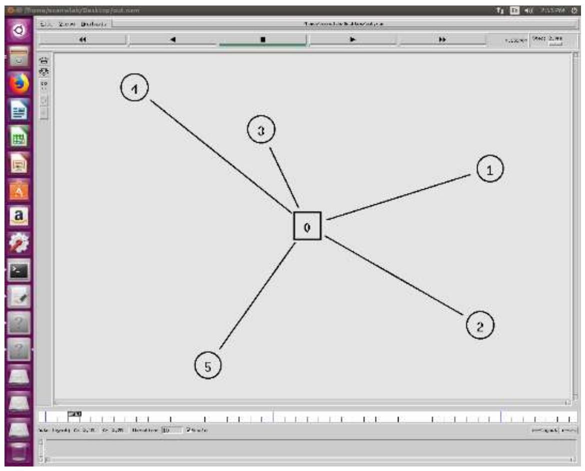
**OUTPUT:**

**Bus Topology**

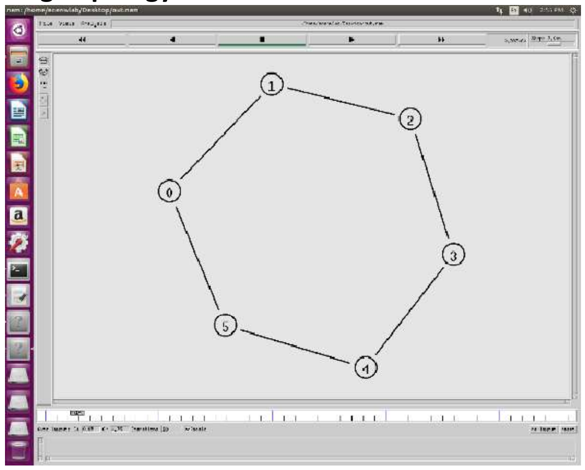


```
No such events found in tracefile.
```

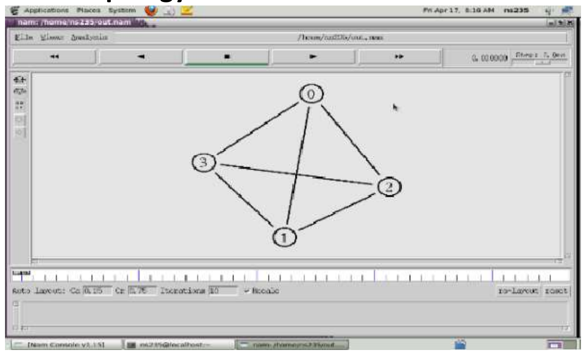
**Star Topology**



## Ring Topology



## Mesh Topology



**RESULT:**

Thus the network topologies like star, mesh, bus and ring have been implemented using network simulator.

| EXPT NO:9 | IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM |
|---|---|
| DATE: | |

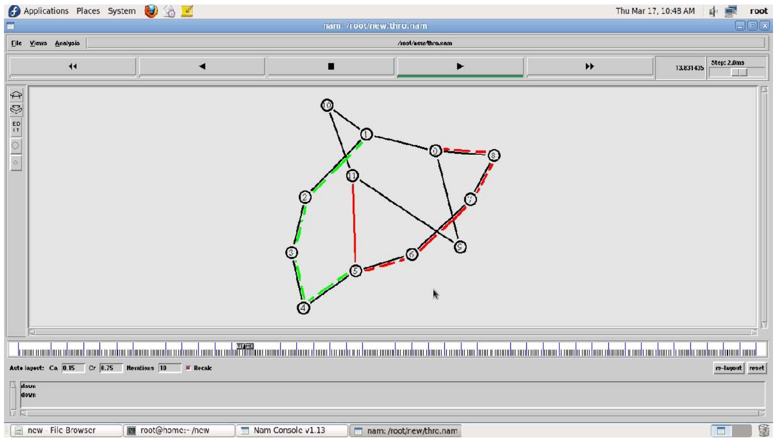
**AIM:**
To implement and simulate the distance vector routing algorithm using NS2 simulator.

**REQUIREMENTS:**
Operating System : Windows NT/2000/XP or LINUX
Programming Tool : Network Simulator (NS2)

**ALGORITHM:**
**Step 1 :** Start.
**Step 2 :** Create a simulator object.
**Step 3 :** Configure the simulator to use dynamic routing.
**Step 4 :** Open the nam.trace file.
**Step 5 :** Open the output file.
**Step 6 :** Define the finish procedure.
**Step 7 :** Close the trace file.
**Step 8 :** Create all the nodes.
**Step 09 :** Create links between the nodes.
**Step 10 :** Create a TCP agent to attach the node.
**Step 11 :** Create a CBR traffic router and attach.
**Step 12 :** Create a Null agent to the traffic sink.
**Step 13 :** Connect the traffic source to the sink.
**Step 14 :** Schedule the events for CBR agent.
**Step 15 :** Stop.

PROGRAM:
```
set ns [new Simulator] set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf proc finish { } { global ns nr nf
$ns flush-trace close $nf
close $nr
exec nam thro.nam & exit 0
}
for { set i 0 } { $i < 12} { incr i 1 } { set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0 set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1 set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto DV
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
SRM VEC/ECE/LM/EC8563/CNL/2020-21/ODD 58
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```



```
down
down
```

**RESULT:**
Thus the distance vector routing algorithm have been implemented and simulated using NS2 simulator.

| EXPT NO:10 | IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM |
|---|---|
| DATE: | |

**EX. NO. 10 IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM**

**AIM:**

To implement and simulate the link state routing algorithm using NS2 simulator.

**REQUIREMENTS:**

Operating System : Windows NT/2000/XP or LINUX
Programming Tool : Network Simulator (NS2)

**ALGORITHM:**

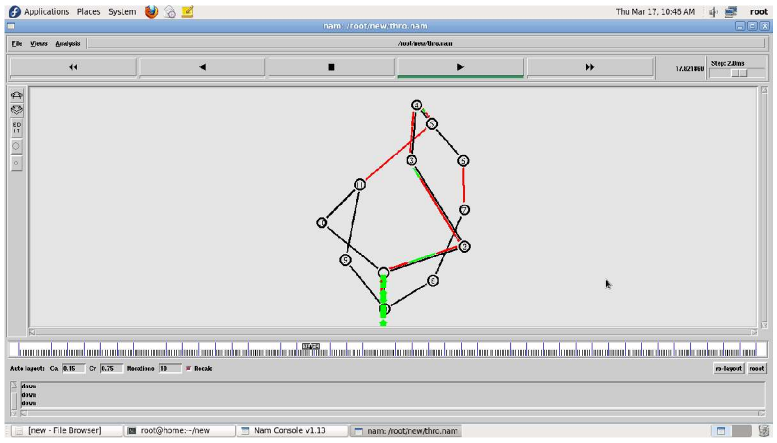
**Step 1 :** Start.
**Step 2 :** Create a simulator object.
**Step 3 :** Configure the simulator to use Link state routing.
**Step 4 :** Open the nam trace file.
**Step 5 :** Open the output file.
**Step 6 :** Define the finish procedure.
**Step 7 :** Close the trace file.
**Step 8 :** Call x-graph to display the result.
**Step 9 :** Create 7 nodes.
**Step 10 :** Create links between the nodes.
**Step 11 :** Create a UDP agent to attach the node.
**Step 12 :** Create a CBR traffic router and attach.
**Step 13 :** Create a Null agent to the traffic sink.
**Step 14 :** Connect the traffic source to the sink.
**Step 15 :** Schedule the events for CBR agent.
**Step 16 :** Stop.

**PROGRAM:**

```
set ns [new Simulator] set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf proc finish { } { global ns nr nf
$ns flush-trace close $nf
close $nr
exec nam thro.nam & exit 0
}
for { set i 0 } { $i < 12} { incr i 1 } { set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
```

```
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0 set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1 set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

OUTPUT:



RESULT:
Thus the Link state routing was implemented and analyzed using NS2 program.

| EXPT NO:11 | STUDY OF NETWORK SIMULATOR (NS) AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS |
|---|---|
| DATE: | |

**AIM:**
To write a program in NS2 to simulate the TCP congestion control algorithm.

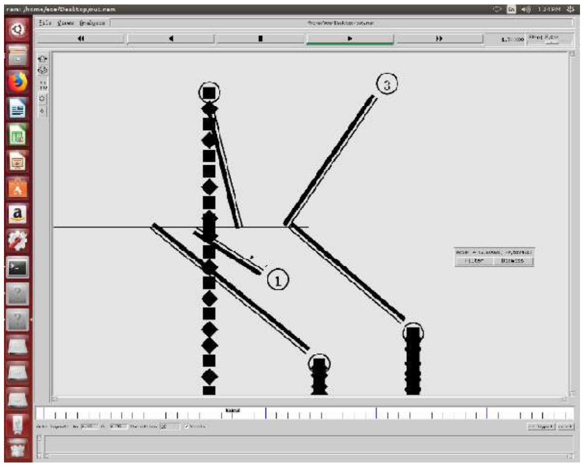
**ALGORITHM:**
**Step 1 :** Start.
**Step 2 :** Create a simulator object.
**Step 3 :** Configure the simulator to use Link state routing.
**Step 4 :** Open the nam trace file.
**Step 5 :** Open the output file.
**Step 6 :** Define the finish procedure.
**Step 7 :** Close the trace file.
**Step 8 :** Create the required nodes.
**Step 10 :**.Set the queue size limits.
**Step 11 :** Create links between the nodes and
**Step 12 :** Create a TCP and UDP agent to attach the node.
**Step 13 :** Create a CBR traffic router and attach.
**Step 14 :** Create a Null agent to the traffic sink.
**Step 15 :** Connect the traffic source to the sink.
**Step 16 :** Schedule the events for CBR agent.
**Step 17 :** Stop.

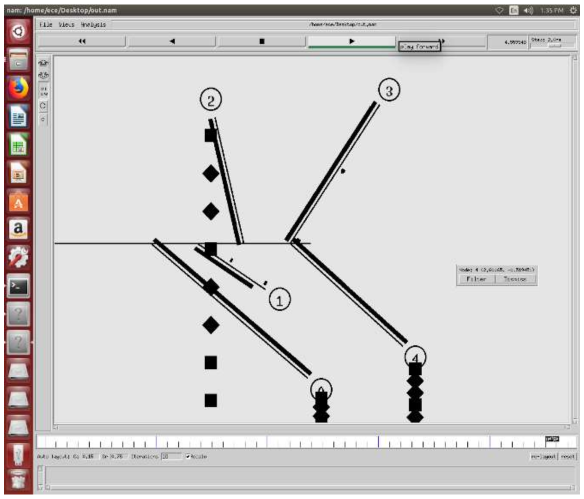
**CONGESTION CONTROL**

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#Define a 'finish' procedure
proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$n5 color red
$n5 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 10ms DropTail
$ns duplex-link $n4 $n3 2Mb 10ms DropTail
$ns duplex-link $n5 $n3 2Mb 10ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 50
#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 800
$tcp set packetSize_ 50
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 3.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"
```

$ns run

**OUTPUT:**





**RESULT:**

The TCP congestion control algorithm was studied and analyzed using NS2 programs.

| EXPT NO:12 | IMPLEMENTATION OF ENCRYPTION AND DECRYPTION ALGORITHMS USING ANY PROGRAMMING LANGUAGE |
|---|---|
| DATE: | |

**AIM:** To write a C program to implement the encryption and decryption concepts.

**ALGORITHM:**

1. Get the message string
2. Get the key value
3. Do encryption on message by key value
4. Store the encrypted message
5. Do decryption on received message by using same key
6. **Store decrypted message.**

PROGRAM

```
//Simple C program to encrypt and decrypt a string
#include <stdio.h>
int main()
{
int i, x;
char str[100];
printf("\nPlease enter a string:\t");
gets(str);
printf("\nPlease choose following options:\n");
printf("1 = Encrypt the string.\n");
printf("2 = Decrypt the string.\n");
scanf("%d", &x);
//using switch case statements
switch(x)
{
case 1:
for(i = 0; (i < 100 && str[i] != '\0'); i++)
str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value
printf("\nEncrypted string: %s\n", str);
break;
case 2:
for(i = 0; (i < 100 && str[i] != '\0'); i++)
str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value
printf("\nDecrypted string: %s\n", str);
break;
default:
printf("\nError\n");
}
return 0;
}
```

OUTPUT

```
Please enter a string:     I LOVE INDIA
Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1
Encrypted string: L#ORYH#LQGLD
Please enter a string:     L#ORYH#LQGLD
Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
2
Decrypted string: I LOVE INDIA
```

**RESULT:** Thus the encryption and decryption is implemented using C program.