STUDY OF NETWORK SIMULATOR(NS2)

AIM: To study about NS2 simulator in detail

THEORY:

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Test bed (VINT) project.Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers andm developers in the community are constantly working to keep NS2 strong and versatile.

BASICARCHITECTURE:

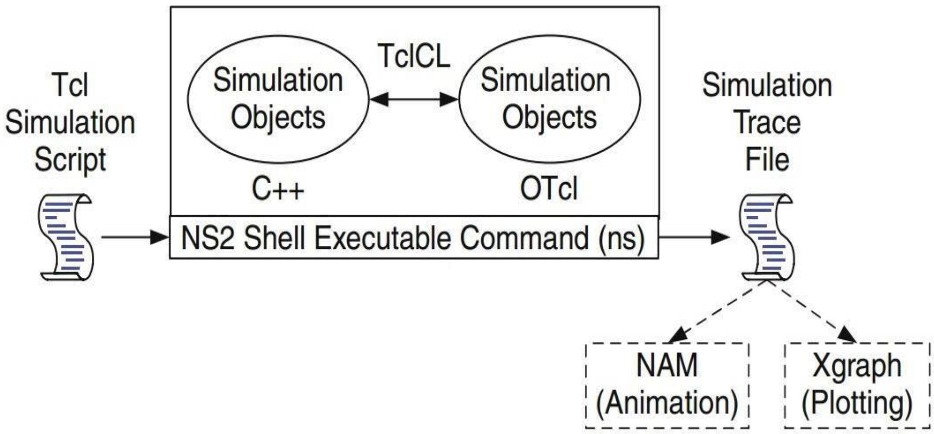


Fig. 2.1. Basic architecture of NS.

Figure2.1shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of anNS2 executable command ns.

In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language(OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, theOTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events(i.e., a front end).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables inthe OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Nodehandle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead,thefunctionality(e.g.,receivingapacket)isdefinedinthemappedC++object(e.g.,ofclassConnector). IntheOTcldomain,ahandleactsasafrontendwhichinteractswithusersandotherOTcl

objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. We refer the readers to [14] for the detail of C++, while a brief tutorial of Tcl and OTcl tutorial are given in Appendices A.1 and A.2,respectively.

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface top up together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results.To interpret these results graphically and interactively,tools such as NAM (Network Animator) and XGraph are used. To analyze a particular behaviour of the network,users can extract a relevant subset of text-based data and transformit to a more conceivable presentation.

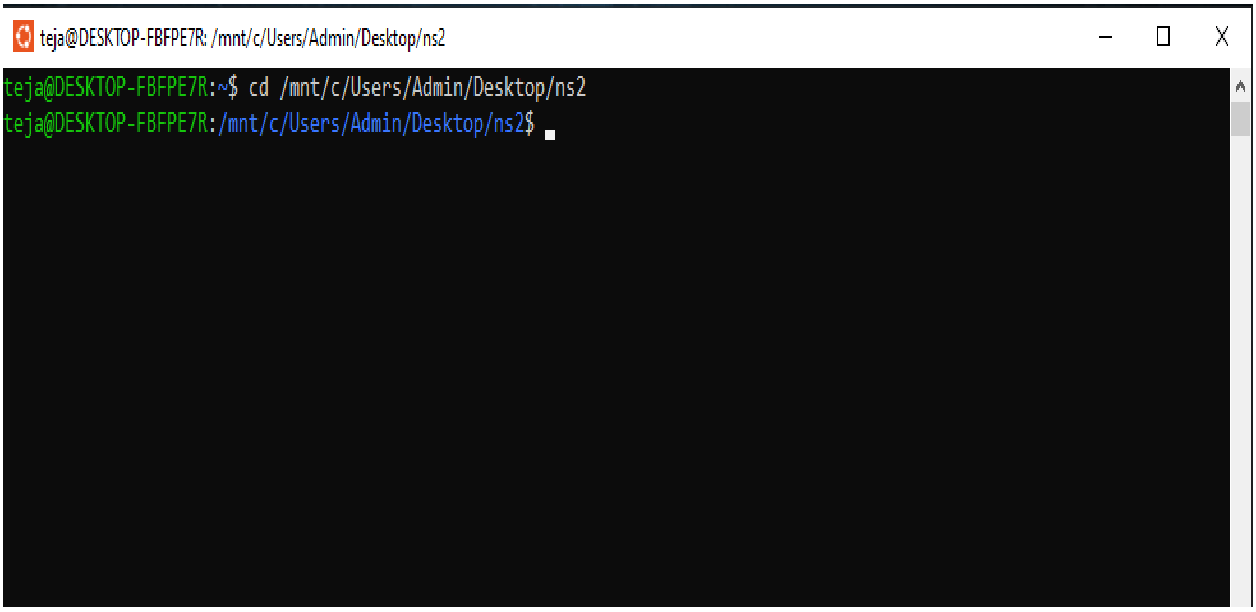
PROCEDURE FOR NS2 SOFTWARE:

Open ubuntu software in windows

To change the location to the above directory, type the following command and hit Enter key.

```
cd /mnt/c/Users/Admin/Desktop/ns2
```

You can see the above command in action in the below figure:



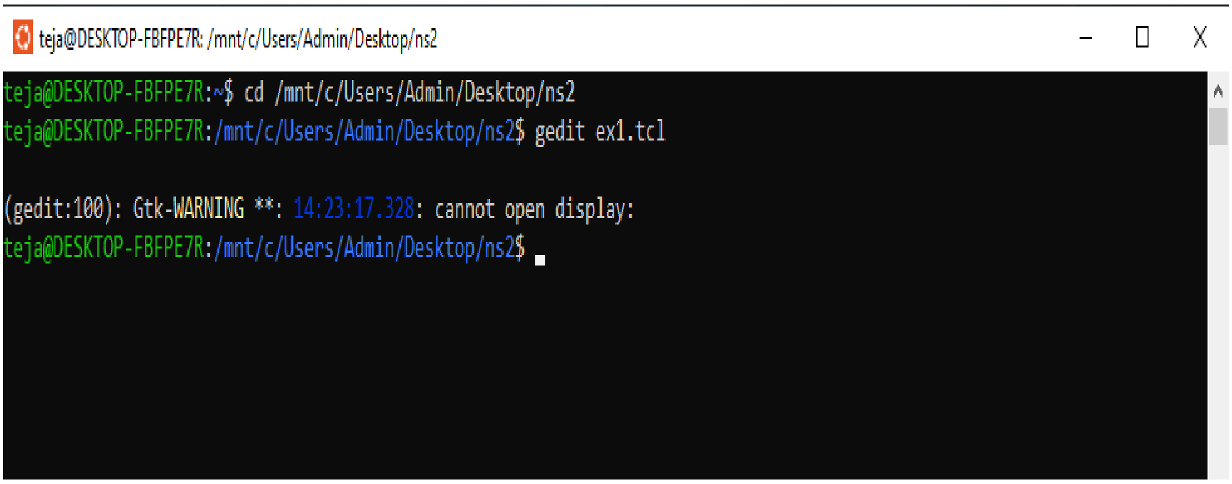
Note the path may vary from system to system. My ns2 programs are going to be saved in the “ns2” folder as shown above.

Opening gedit and typing the program.

Type the following command in the terminal (black window) and hit Enter key.

```
gedit ex1.tcl
```

You might get a warning message as shown in below figure:



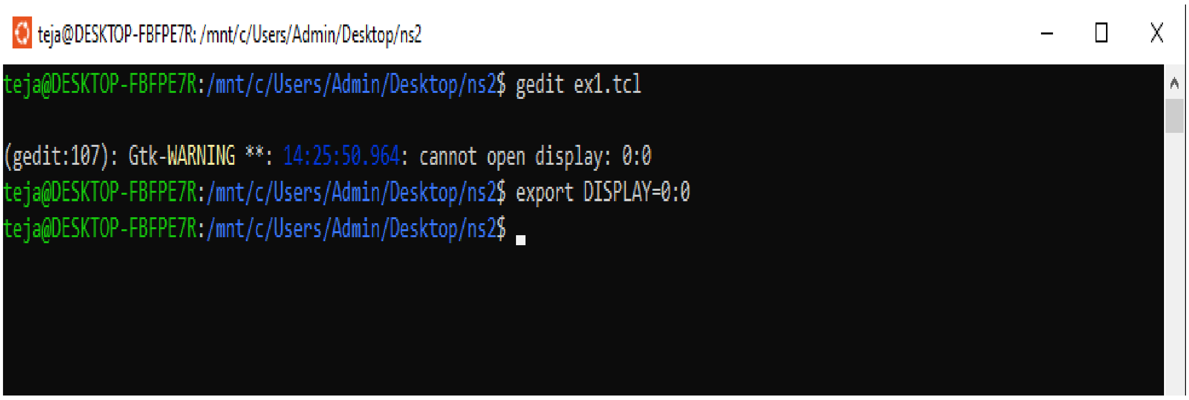
For opening the “gedit” application we have to start the “Xming” application.

Open search and type “Xming” and click on “Open”.

The “Xming” server will be started and it will available in the system tray on the taskbar. Sometimes it might get hidden in the system tray and is not always visible.

Xming version : xming 6-9-0-31

Now, in the terminal type the following command as shown in the figure below and hit enter key.



Now, type the following command in the terminal (black window) and hit Enter key.

```
gedit ex1.tcl
```

You should be able to see a blank gedit window as shown in below image.

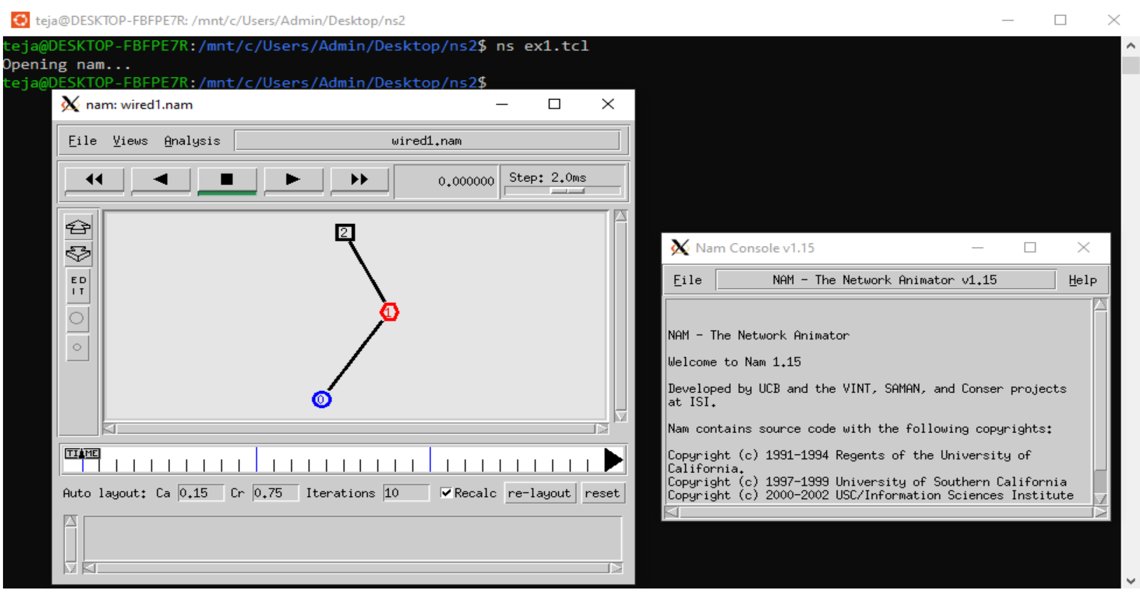
Type the following sample program and save (CTRL + s) the file. Now close gedit window.

Running the program using “ns” command.

Type the following command in the terminal to run the program and see the output.

ns ex1.tcl

Now, you should be able to see the network animator (nam) window with the topology as shown in the below figure.



If you can see the above output nam successfully on your system.

CONCEPT OVER VIEW:

NS uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For thesetasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile,re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. Ns meets both of these needs with two languages, C++and OTcl.

Tcl scripting

Tclisa general purpose scripting language.[Interpreter]

• Tcl run son most of the platforms such as Unix, Windows ,and Mac.

• The strength of Tclisits simplicity.

• It is not necessary to declare a data type for variable priorto the usage.

Basics of TCL

Syntax:commandarg1arg2 arg3

Hello World!

Puts std out{Hello,World!}Hello,World!

Variables Command Substitution

set a 5se tlen[string length foobar]

Set b$a setlen [expr[string length foobar] +9]

Wired TCL Script Components

Create the events cheduler

Open new files & turn on the tracing

Create the nodes

Setup the links

Configure the traffic type (e.g., TCP, UDP,

etc)Set the time of traffic generation (e.g., CBR,

FTP)Terminate the simulation

NS Simulator Preliminaries.

1. Initializationandterminationaspectsofthenssimulator.
2. Definitionofnetworknodes,links,queuesand topology.
3. Definitionofagents andofapplications.
4. Thenam visualizationtool.
5. Tracingandrandomvariables.

InitializationandTerminationofTCLScriptinNS-2

Anns simulationstarts withthecommand

setns[newSimulator]

Which is thus the first line in the tcl script. This line declares a new variable as using the set command,youcancallthisvariableasyouwish,Ingeneralpeopledeclaresitasnsbecauseitisaninstanceofthe Simulator class, so an object the code[new Simulator] is indeed the installation of theclassSimulatorusingthe reserved word new.

In order to have output files with data on the simulation (trace files) orfiles used for visualization(namfiles), weneed tocreate thefilesusing―open command:

#Openthe Tracefile

settracefile1[openout.trw]
$ns trace-all $tracefile1

#Openthe NAMtracefile

setnamfile[open out.nam w]
$nsnamtrace-all$namfile

Theabove createsadtatrace filecalled out.trandanamvisualizationtrace filecalled out.nam.
Within the tcl script, these files are not called explicitly by their names, but instead by pointers that aredeclared above and called —tracefile1 and —namfile respectively. Remark that they begins with a #symbol. The second line open the file —out.tr to be used for writing, declared with the letter —w. Thethird line uses a simulator method called trace-all that have as parameter the name of the file where thetraceswillgo.

Define a "finish"

procedureProcfinish{}{
globalnstracefile1namfile
$ns flush-
traceClose
$tracefile1Close
$namfile
Exec
namout.nam&Exit 0
}
Definitionof anetworkof linksandnodes
Thewaytodefineanodeis

setn0[$ns node]

Oncewedefineseveral nodes,wecan definethelinks that connect them.An exampleof adefinitionof alink is:

$nsduplex-link$n0$n210Mb10msDropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagationdelayand acapacityof 10Mb per secforeach direction.
Todefineadirectional link insteadofabi-directional one, weshould replace—duplex-link by —simplex-link.
Inns,anoutputqueueofanodeisimplementedasapartofeachlinkwhoseinputisthatnode.Weshouldalso define thebuffercapacityof thequeuerelated to each link. Anexamplewould be:

#setQueueSizeoflink(n0-n2)to 20

$nsqueue-limit$n0$n220

FTPoverTCP
TCPisadynamicreliablecongestioncontrolprotocol.ItusesAcknowledgementscreatedbythedestination to know whetherpackets arewell received.
Therearenumbervariants oftheTCP protocol,suchas Tahoe,Reno,NewReno,Vegas.
Thetypeofagentappears in the firstline:

settcp[newAgent/TCP]

Thecommand **$nsattach-agent$n0 $tcp**defines thesourcenodeof thetcp connection.
The command **set sink [new Agent /TCPSink]** Defines the behavior of the destination node of TCPandassigns to it apointercalled sink.

#Setup a UDP

connectionsetudp[newAg

ent/UDP]
$ns attach-agent $n1
$udpsetnull[newAgent/Nu
ll]
$nsattach-agent$n5$null
$nsconnect$udp$null
$udpsetfid_2

#setupaCBR overUDPconnection
Thebelow showsthe definitionofaCBR applicationusingaUDP agent
Thecommand**$nsattach-agent$n4$sink** definesthedestinationnode.Thecommand**$nsconnect tcpsink**finallymakestheTCP connectionbetween thesourceanddestination nodes.

setcbr[newApplication/Traffic/CBR]
$cbrattach-agent$udp
$cbrset packetsize_100
$cbrsetrate_0.01Mb
$cbrsetrandom_false

TCP has many parameters with initial fixed defaults values that can be changed if mentionedexplicitly.Forexample,thedefaultTCPpacketsizehasasizeof1000bytes.Thiscanbechangedt oanothervalue, say552bytes, usingthe command**$tcp set packetSize_ 552**.
When we have several flows, we may wish to distinguish them so that we can identify them withdifferent colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns tothe TCP connection a flow identification of ―1.We shall later give the flow identification of ―2‖ totheUDP connection.

EXPT.N 01	TELNET AND FTP BETWEEN N SOURCES
DATE:	

AIM: To Simulate A Program of TELNET and FTP Between N Sources – N Sinks (N=1,2,3).

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. Network Simulator (Version 2), is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator, the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers and developers in the community are constantly working to keep NS2 strong and versatile

https://www.nsnam.com/2023/02/telnet-and-ftp-in-wired-networks-using.html

PROGRAM:

Node 0 to Node 2 is enabled with Telnet Application and Node 1 to Node 3 is enabled with FTP Application. Save the following file as Filename.tcl

```
#===================================
#     Simulation parameters setup
#===================================
set val(stop)   10.0  ;# time of simulation end


#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]
```

```
#Open the NS trace file
set tracefile [open ftp.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open ftp.nam w]
$ns namtrace-all $namfile

#===================================
#       Nodes Definition
#===================================
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#===================================
#       Links Definition
#===================================
#Createlinks between nodes
$ns duplex-link $n0 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n4 50
$ns duplex-link $n1 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n4 50
$ns duplex-link $n2 $n4 100.0Mb 10ms DropTail
```

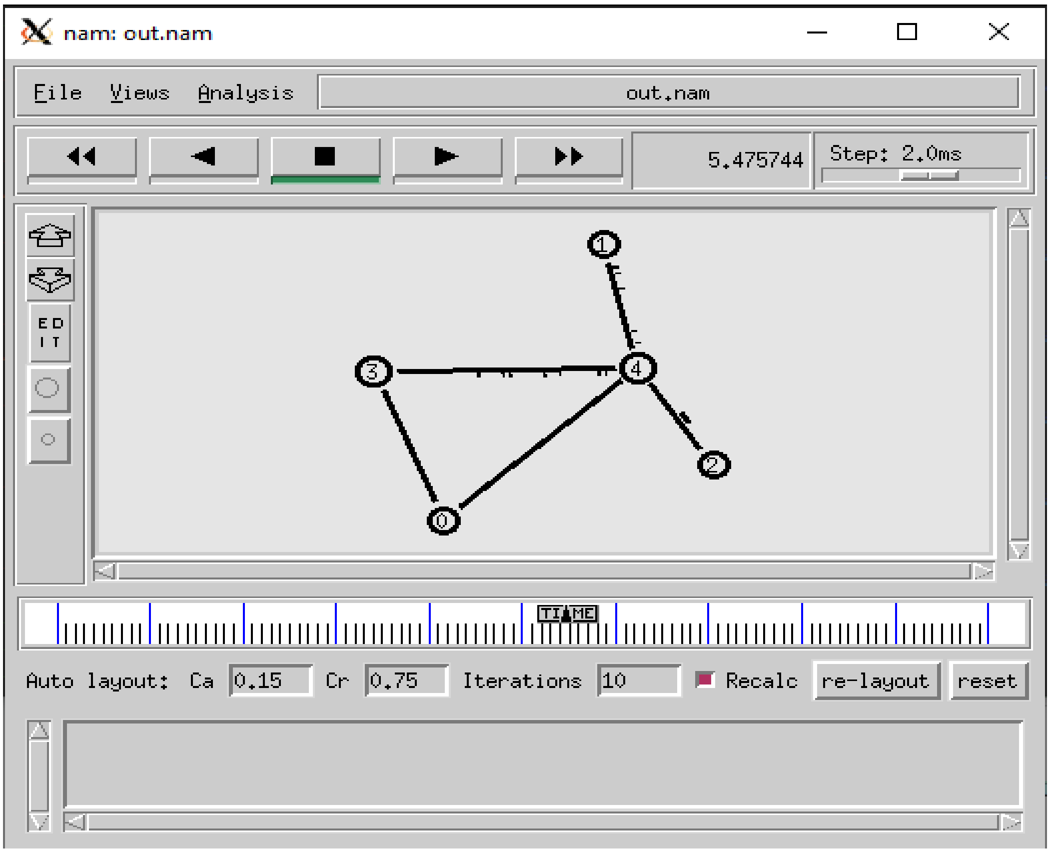
```
$ns queue-limit $n2 $n4 50
$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n3 $n0 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n0 50

#       Agents Definition
#===================================
#Setup a UDP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns connect $tcp0 $sink1

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
$ns connect $udp1 $null3
$udp1 set packetSize_ 1500


#===================================
#       Applications Definition
#===================================
```

```
#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp0
$ns at 1.0 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"

#Setup a Telnet Application over UDP connection
set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.001
$telnet0 attach-agent $udp1
$ns at 1.0 "$telnet0 start"
$ns at 10.0 "$telnet0 stop"
#$ns at 10.0 "$cbr1 stop"
$telnet0 set type_ Telnet

#       Termination       #=
#Define a 'finish' procedure
proc finish {} {
   global ns tracefilenamfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec namout.nam&
   exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
```

```
$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run
```

When you run the above file using the command
$ ns filename.tcl
and the output generated is out.nam and out.tr
The packet transmission is shown like this in the animation window:

OUTPUT:



Telnet in ns2

Node 0 to Node 2 is enabled with Telnet Application and Node 1 to Node 3 is enabled with FTP Application, which is shown in the above picture.
To get the throughput of the above file in bits per second, here is the awk script:
Save the following in a file called telnet.awk and store in the same place where the filename.tcl is also stored.
Example: type in ubuntu : gedit telnet.awk
type the program below and save it

```
BEGIN
 {
numTCP1=0;
tcpSize1=0;
numTCP2=0;
tcpSize2=0;
totaltcp1=0;
totaltcp2=0;
}
{
event=$1;
pkttype= $5;
fromnode=$9;
tonode=$10;
pktsize=$6;
if(event == "r" &&pkttype == "udp" &&fromnode == "1.0" &&tonode == "3.0")
{
numTCP1++;
tcpSize1 = pktsize;
}
if(event == "r" &&pkttype == "tcp" &&fromnode == "0.0" &&tonode == "2.0")
{
numTCP2++;
tcpSize2 = pktsize;

}

}

END {
totaltcp1=numTCP1*tcpSize1*8;
totaltcp2=numTCP2*tcpSize2*8;
throughputtcp1= totaltcp1/24; # because simulation time is 24.5   0.5 = 24
throughputtcp2= totaltcp2/24; # because simulation time is 24.5   0.5 = 24
printf("The Throughput of FTP application is %d \n", throughputtcp1);
printf("The Throughput of TELNET application is %d \n", throughputtcp2);
}
```

The above file can be run using the command:

$ gawk -f telnet.awk filename.tr

The above command will print the following two lines which informs the throughput of using Telnet and FTP:

OUTPUT:

The Throughput of FTP application is 8962000
The Throughput of TELNET application is 3058293

RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. What protocols do ns support?

2. What is Simulation?

3. Define Network

4. What is meant by Protocol?

5. What are the constituent parts of NS2?

EXPNO:02 DATE:	THE EFFECT OF VARIOUS QUEUEING DISCIPLINES (RED / Weighted RED / Adaptive RED) ON NETWORK PERFORMANCE

AIM: To study and compare the various queue management schemes practically using NS-2

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Queue Management is defined as the algorithm that manage the length of the packet queues by dropping packets when necessary. From the point of packet dropping, Queue management can be classified into 2 types

1. Passive Queue Management: In Passive Queue Management the packet drop occurs only when the buffer gets full. Ex: Drop Tail.
2. Active Queue Management: Active Queue Management employs preventive packet drops.It provides implicit feedback mechanism to notify senders of the onset of congestion. Arriving packets are randomly dropped. Ex: RED.

Drop Tail: In this packets are dropped from the tail of the queue. Once buffer gets full, all arriving packets are discarded. Packets already in the queue are not affected.

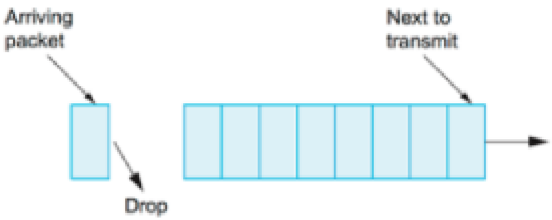
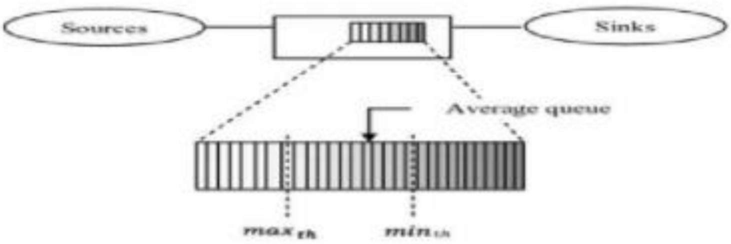


Figure 2.1: Drop Tail mechanism

As shown in figure 2.1, the arriving packet gets dropped from the tail when the queue is filled

Random Early Detection: RED accepts all packets until the queue reaches minth, after which it drops a packet with a linear probability distribution function. When the queue length reaches maxth all packets are dropped with probability of one



Weighted random early detection (WRED)

It is a queueing discipline for a network scheduler suited for congestion avoidance. It is an extension to random early detection (RED) where a single queue may have several different sets of queue thresholds. Each threshold set is associated to a particular traffic class

For example, a queue may have lower thresholds for lower priority packet. A queue buildup will cause the lower priority packets to be dropped, hence protecting the higher priority packets in the same queue. In this way quality of service prioritization is made possible for important packets from a pool of packets using the same buffer

It is more likely that standard traffic will be dropped instead of higher prioritized traffic. WRED proceeds in this order when a packet arrives:

1. Calculation of the average queue size
2. The arriving packet is queued immediately if the average queue size is below the minimum queue threshold.
3. Depending on the packet drop probability the packet is either dropped or queued if the average¬ queue size is between the minimum and maximum queue threshold.
4. The packet is automatically dropped if the average queue size is greater than the maximum¬ threshold

Adaptive RED Queue Discipline:

The motivation of Adaptive RED is the same as self-configuring RED. Self-configuring RED tries to keep the average queue size with minimum and maximum threshold values. But Sally Floyd says that why don’t keep the average queue size in a tight range just in the center of mi nimum and maximum threshold values. Also, Adaptive RED removes the kns and automatically sets them. Maximum drop probability is adapted based on the network availability, it is no longer a knob just like previous versions of RED.

ALGORITHM:

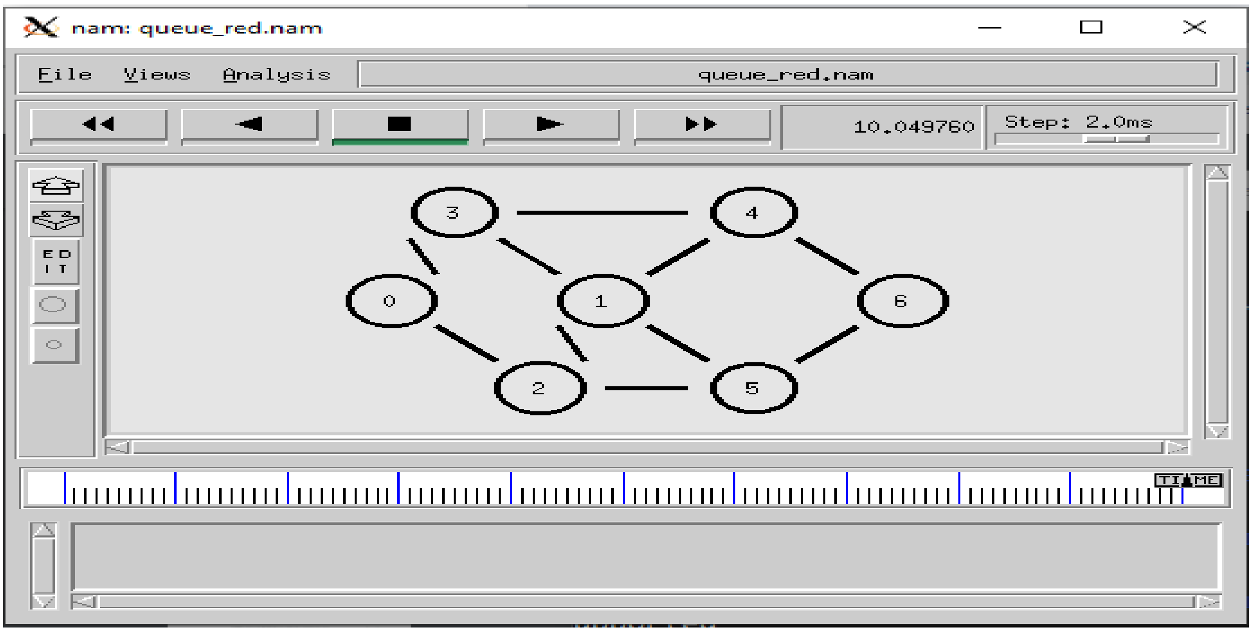
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam and trace file (output files)
4. Create 8 nodes that forms a network numbered from 0 to 7 for RED queue mechanism and 6 nodes that form a network numbered from 0 to 5 for Drop tail mechanism.
5. Create duplex links between the nodes with bandwidth 100 Mbps.
6. Create duplex links between n3 and n4 with bandwidth 2Mbps for RED queue mechanism and create duplex link between n2 and n3 with bandwidth 1Mbps for Drop tail.
7. Setup TCP Connection between n0 and n5 and also setup UDP connection between n1 and n4 in case of Drop tail. While in the case of RED queue mechanism, setup TCP connection between n0 and n5 , TCP connection between n2 and n7 and also setup UDP connection between n1 and n6
8. Apply CBR Traffic over UDP, FTP Traffic over TCP.
9. Define finish procedure then close the trace file, and execute nam file.
10. Schedule events and run the program

PROGRAM:

```
set val(stop) 10.0 ; # time of simulation end
#Create a ns simulator
set ns [new Simulator]
#Open the NS trace file
set nr [open queue_red.tr w]
$ns trace-all $nr
#Open the NAM trace file
set nf [open queue_red.nam w]
$ns namtrace-all $nf
#Create 7 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
$ns duplex-link $n0 $n3 100.0Mb 10ms RED
$ns queue-limit $n0 $n3 50
$ns duplex-link $n4 $n1 100.0Mb 10ms RED
$ns queue-limit $n4 $n1 50
$ns duplex-link $n6 $n5 100.0Mb 10ms RED
$ns queue-limit $n6 $n5 50
$ns duplex-link $n4 $n6 100.0Mb 10ms RED
$ns queue-limit $n4 $n6 50
$ns duplex-link $n5 $n2 100.0Mb 10ms RED
$ns queue-limit $n5 $n2 50
$ns duplex-link $n0 $n2 100.0Mb 10ms RED
$ns queue-limit $n0 $n2 50
$ns duplex-link $n2 $n1 100.0Mb 10ms RED
$ns queue-limit $n2 $n1 50
$ns duplex-link $n1 $n5 100.0Mb 10ms RED
$ns queue-limit $n1 $n5 50
$ns duplex-link $n3 $n4 100.0Mb 10ms RED
$ns queue-limit $n3 $n4 50
$ns duplex-link $n3 $n1 100.0Mb 10ms RED
$ns queue-limit $n3 $n1 50
#Give node position (for NAM)
$ns duplex-link-op $n0 $n3 orient right-up
 $ns duplex-link-op $n4 $n1 orient left-down
 $ns duplex-link-op $n6 $n5 orient left-down
 $ns duplex-link-op $n4 $n6 orient right-down
 $ns duplex-link-op $n5 $n2 orient left
 $ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n2 $n1 orient right-up
 $ns duplex-link-op $n1 $n5 orient right-down
 $ns duplex-link-op $n3 $n4 orient right
 $ns duplex-link-op $n3 $n1 orient right-down

#Setup a TCP connection
set tcp0 [new Agent/TCP]
```

```
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n4 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500
#Setup a TCP/FullTcp/Tahoe connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1
 set sink3 [new Agent/TCPSink]
$ns attach-agent $n6 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500
#Setup a FTP Application over TCP connection
 set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
 $ns at 1.0 "$ftp0 start"
 $ns at 10.0 "$ftp0 stop"
#Setup a FTP Application over TCP/FullTcp/Tahoe connection
 set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"
#Define a 'finish' procedure
proc finish { } {
global ns nr nf
$ns flush-trace
close $nr
close $nf
exec nam queue_red.nam &
exit 0
 }
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1 Define Queue

2 List the types queuing disciplines

3 What is meant by Random Early Detection?

4 What is Drop tail, FQ and SFQ?

5 What is meant by Weighted RED and Adaptive RED?

EXPNO:03	
DATE:	SIMULATE HTTP, FTP AND DBMS ACCESS IN NETWORKS

AIM: To write a TCL script to simulate the HTTP, FTP AND DBMS access in networks using NS2

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

ALGORITHM:

1. Create a simulator object
2. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
3. Create two nodes that forms a network numbered from 0 to 1
4. Create duplex links between the nodes n(0) to n(1)
5. Setup TCP Connection between n(0) and n(1)
6. Apply FTP Traffic over TCP.
7. Schedule events and run the program

THEORY:

HTTP:

- HTTP stands for Hyper Text Transfer Protocol.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that
- allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP is similar to the FTP as it also transfers the files from one host to another host. But,
- HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files.
- HTTP is used to carry the data in the form of MIME-like format.

FTP:

- FTP stands for File transfer protocol.
- FTP is a standard internet protocol provided by TCP/IP used for transmitting the files from one host to another.
- It is mainly used for transferring the web page files from their creator to the computer
- that acts as a server for other computers on the internet.
- It is also used for downloading the files to computer from other servers.

DBMS:

Data is the cornerstone of any modern software application, and databases are the most common way to store and manage data used by applications. With the explosion of web and cloud technologies, databases have evolved from traditional relational databases to more advanced types of databases such as NoSQL, columnar, key-value, hierarchical, and distributed databases. Each type has the ability to handle structured, semi-structured, and even unstructured data.
On top of that, databases are continuously handling mission-critical and sensitive data. When this is coupled with compliance requirements and the distributed nature of most data sets, managing databases has become highly complex. As a result, organizations require robust, secure, and userfriendly tools to maintain these databases. This is where database management systems come into play—by offering a platform to manage databases. Let’s take a look.
A database management system (DBMS) is a software tool that enables users to manage a database easily. It allows users to access and interact with the underlying data in the database.

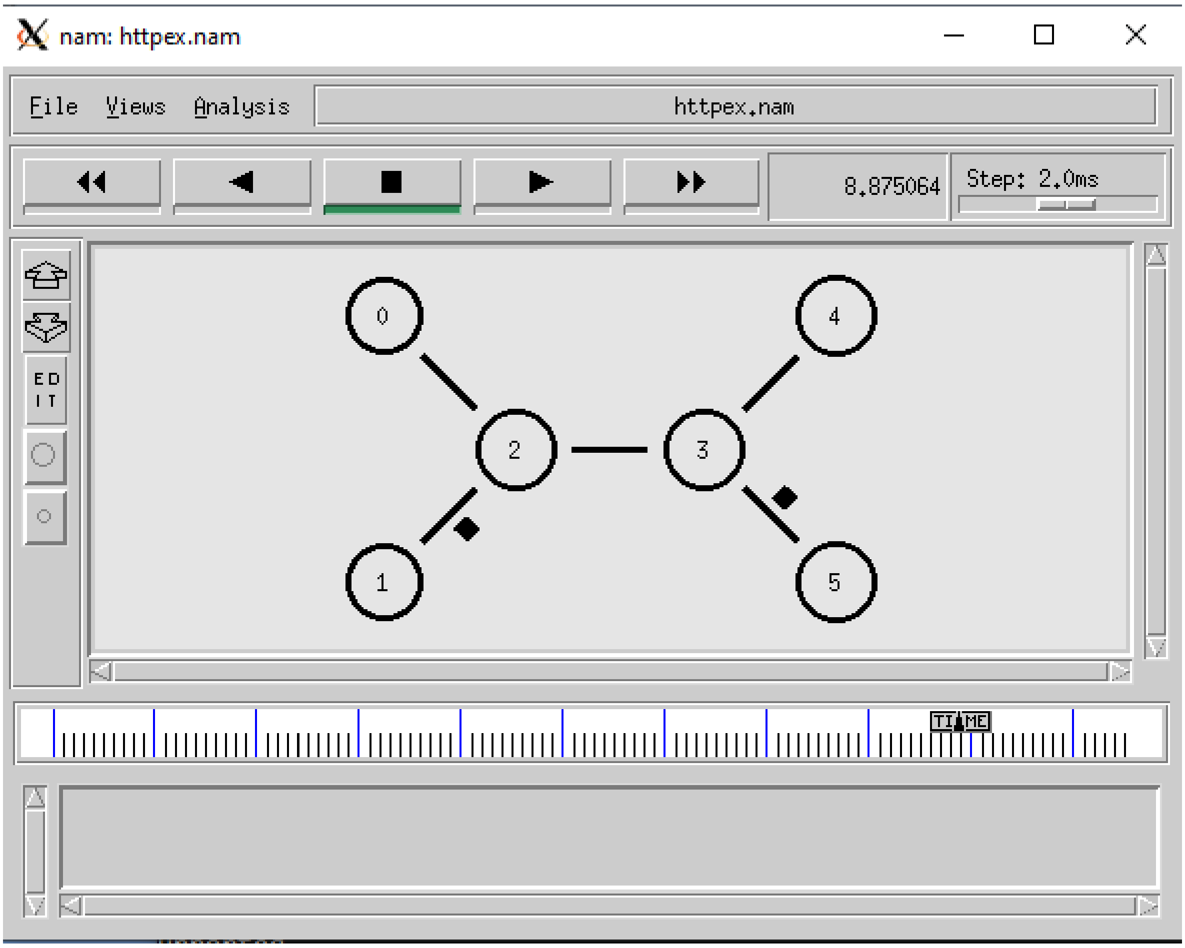
These actions can range from simply querying data to defining database schemas that fundamentally affect the database structure. Furthermore, DBMS allow users to interact with a database securely and concurrently without interfering with each user and while maintaining dataintegrity.

PROGRAM:

```
set val(stop) 10.5
#Create a ns simulator
set ns [new Simulator]
#Open the NS trace file
set tracefile [open httpex.tr w]
$ns trace-all $tracefile
#Open the NAM trace file
set namfile [open httpex.nam w]
$ns namtrace-all $namfile
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Createlinks between nodes
$ns duplex-link $n0 $n2 100.0Mb 10ms SFQ
$ns queue-limit $n0 $n2 50
$ns duplex-link $n3 $n2 100.0Mb 10ms SFQ
$ns queue-limit $n3 $n2 50
$ns duplex-link $n1 $n2 100.0Mb 10ms SFQ
$ns queue-limit $n1 $n2 50
$ns duplex-link $n3 $n4 100.0Mb 10ms SFQ
$ns queue-limit $n3 $n4 50
$ns duplex-link $n3 $n5 100.0Mb 10ms SFQ
$ns queue-limit $n3 $n5 50
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n5 $sink3
$ns connect $tcp0 $sink3
$tcp0 set packetSize_ 1500
#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n4 $tcp1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n1 $sink2
```

```
$ns connect $tcp1 $sink2
$tcp1 set packetSize_ 1500
#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n2 $udp4
set null5 [new Agent/Null]
$ns attach-agent $n5 $null5
$ns connect $udp4 $null5
$udp4 set packetSize_ 48
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"
#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"
#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp4
$cbr2 set packetSize_ 48
$cbr2 set interval_ 50ms
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
#Define a 'finish' procedure
proc finish {} {
global ns tracefile namfile
 $ns flush-trace
close $tracefile
close $namfile
exec nam httpex.nam &
exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

OUTPUT



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. Difference between FTP and HTTP

2. What is HTTP and FTP?

3. What is HTTP in database?

4. What is FTP database?

5. Mention the types of Network Protocols and their uses

EXPT.NO:4	IMPLEMENTATION OF IP ADDRESS CONFIGURATION
DATE:	

AIM: To implementation of IP address configuration.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

IP Address Configuration:

An Internet Protocol (IP) address is a unique number assigned to every device on a network. Just as a streetaddress determines where a letter should be delivered, an IP address identifies computers on the Internet.NetworkdevicesuseIPaddressestocommunicatewitheachother.IPaddressesarerequiredbyanynetworkadapteronanycomputerthatneedstoconnecttotheInternetoranothercomputer.Addressesaregivenoutto networkcomputers inoneoftwomanners,dynamicallyor statically.

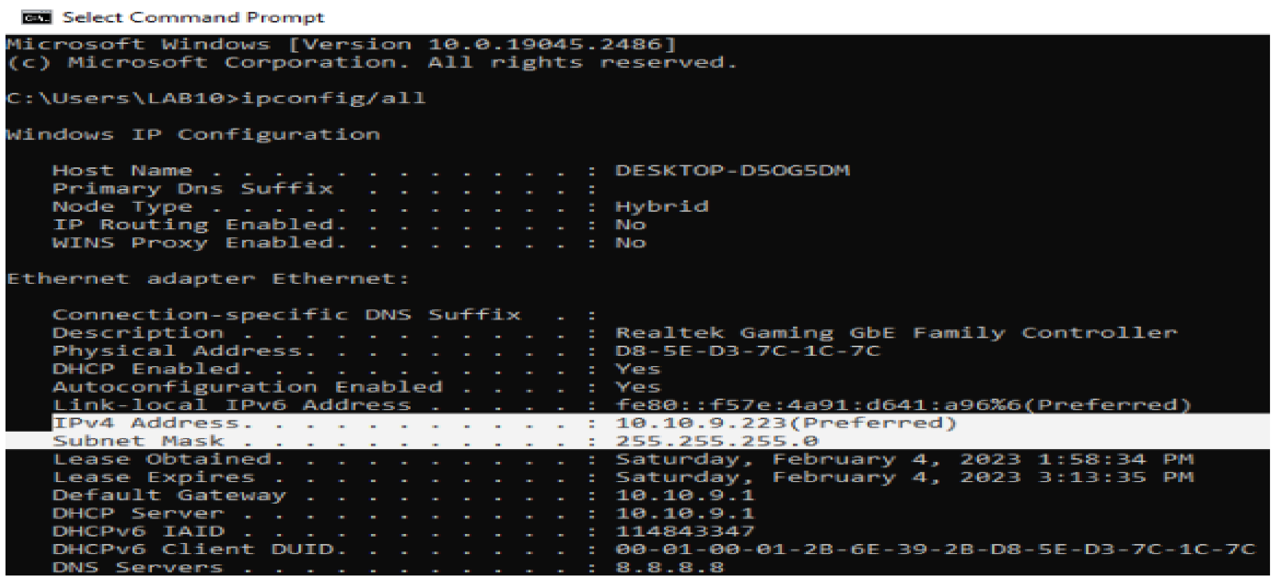
To set astatic IP addressin Windows 7,8,and10:

1. Click StartMenu>ControlPanel> Network andSharingCenteror NetworkandInternet>
2. NetworkandSharingCenter.
3. ClickonLocalAreaConnection.
4. ClickDetails.
5. ViewfortheInternetProtocolVersion4(TCP/IPv4)address.

1. Open the Command Prompt.

   Click the Start icon, type command prompt into the search bar and press click the Command Prompt icon.

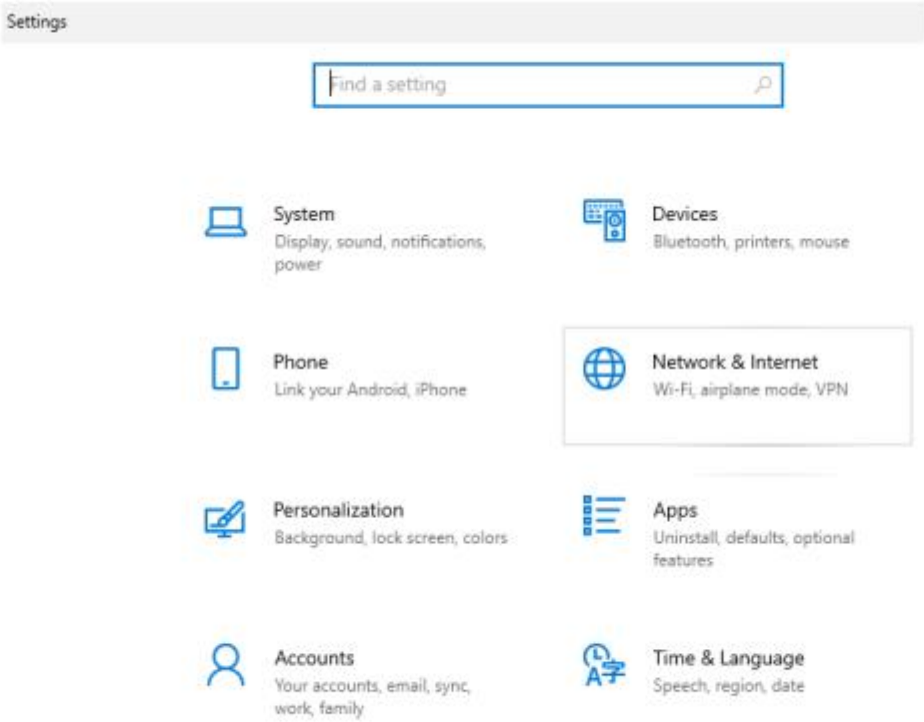
2. Type ipconfig/all and press Enter.



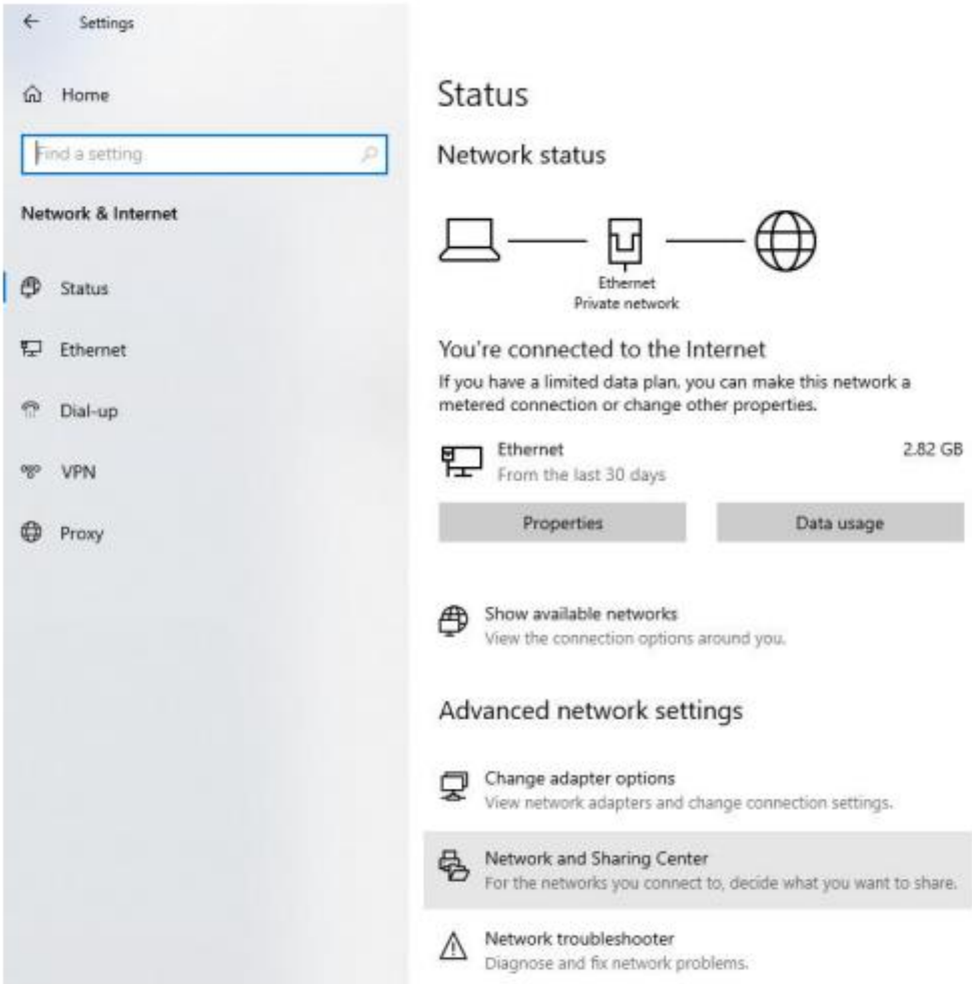
3. The IP Address will be displayed along with other LAN details.

II). Using the Control Panel :

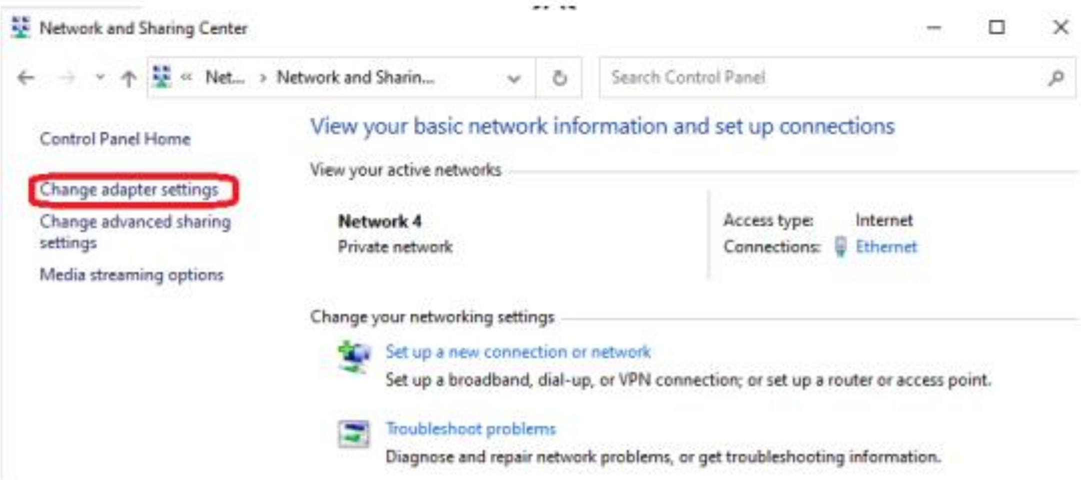
1. Click the Start button, go to settings and then click the settings icon.
2. Click Network and Internet when the Control Panel opens.



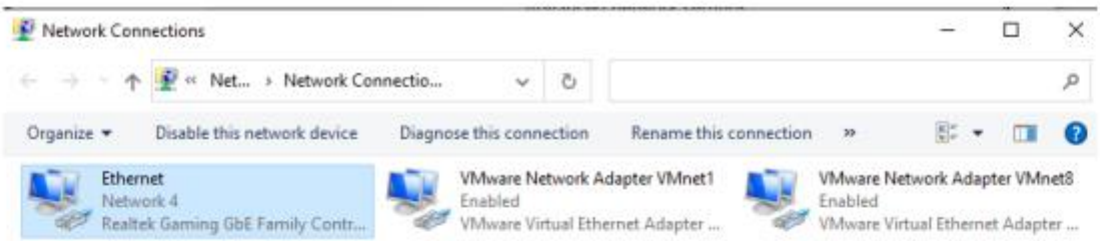
3. Select Network and Sharing Center.



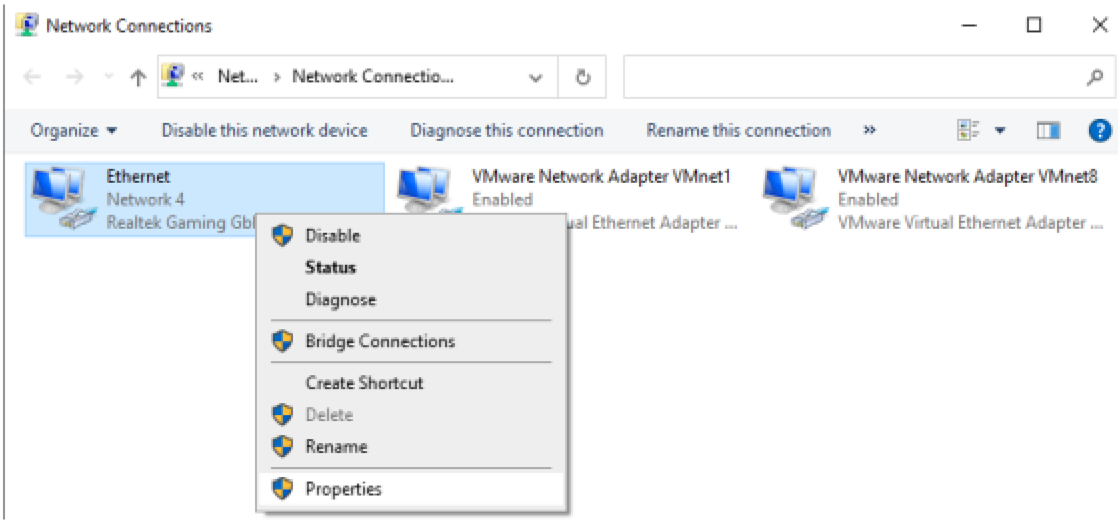
4. Click the Change adapter settings link, located on the left.



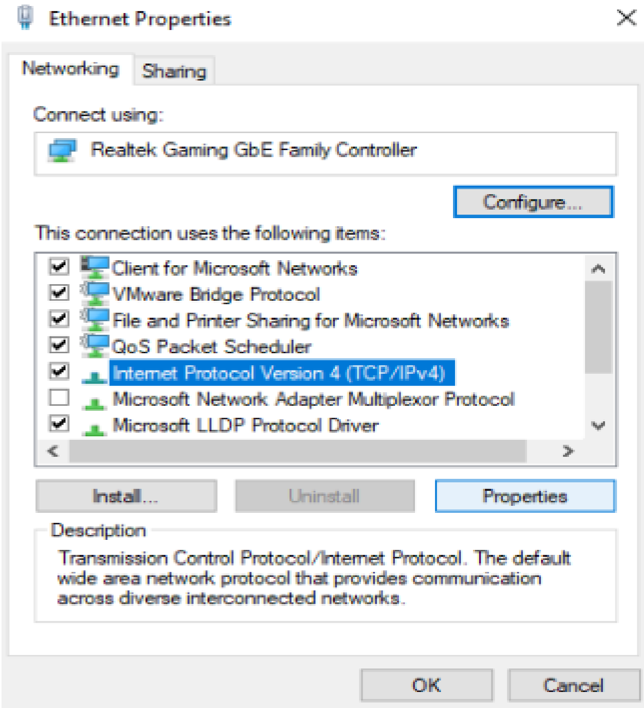
5. Double-click Ethernet.



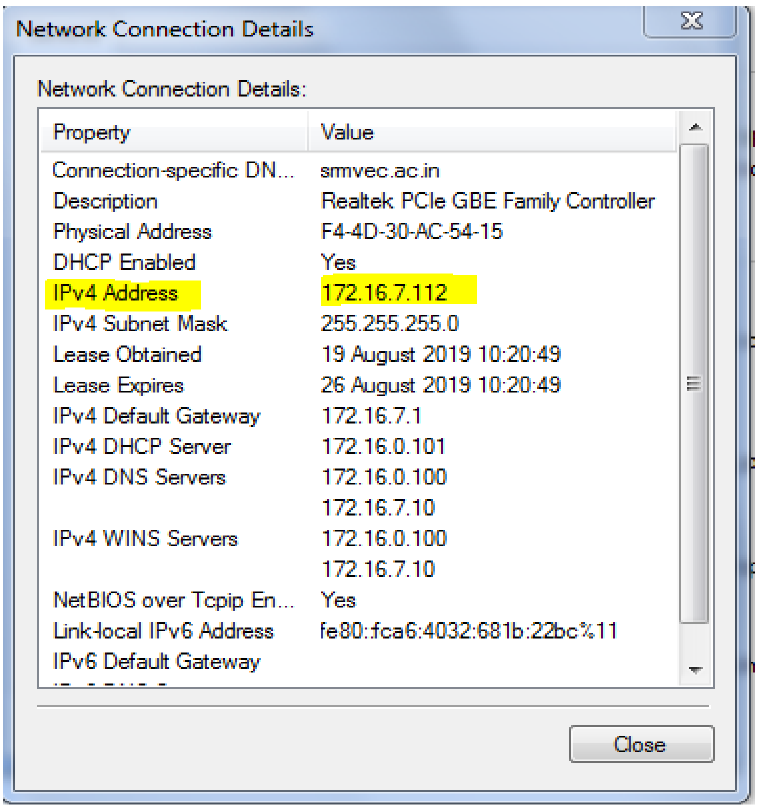
6. In the “Network Connections” window, right-click the adapter for which you want to set a static IP address, and then select the “Properties” command.



7. In the properties window for the adapter, select “Internet Protocol Version 4 (TCP/IPv4)” and then click the “Properties” button.



8. Select the "Use the following IP address" option, and then type in the IP address, subnet mask, and default gateway that corresponds with your network setup. Next, type in your preferred and alternate DNS server addresses. Finally, select the "Validate settings upon exit" option so that Windows immediately checks your new IP address and corresponding information to ensure that it works. When you're ready, click the "OK" button.



9. Close out of the network adapter's properties window.

RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. WhatisIP CONFIG?

2. Howto getavalidipconfig?

3. WhatisTCP/IPconfiguration?

4. Whataretheparametersipconfigdisplays?

5. Whatis theuseofDNSandDHCP inipconfig?

.

EXPT.NO:05	PERFORMANCEAN ALYSIS OF CSMA/CA AND CSMA/CD PROTOCOLS
DATE:	

AIM: To create scenario and study the performance of network with CSMA/ CA protocol and compare CSMA/ CD protocols through simulation.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Ethernet is a LAN (Local area Network) protocol operating at the MAC (Medium Access Control) layer.Ethernet has been standardized as per IEEE 802.3. The underlying protocol in Ethernet is known as the CSMA /CD – Carrier Sense Multiple Access / Collision Detection. The working of the Ethernet protocol is as explainedbelow, A node which has data to transmit senses the channel. If the channel is idle then, the data is transmitted.If the channel is busy then, the station defers transmission until the channel is sensed to be idle and thenimmediately transmitted. If more than one node starts data transmission at the same time, the data collides. Thiscollision is heard by the transmitting nodes which enter into contention phase. The contending nodes resolvecontention usingan algorithmcalledTruncated binaryexponentialbackoff.

ALGORITHM:

1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create sixnodesthatformsanetworknumberedfrom0to5
5. Createduplexlinksbetween thenodesandaddOrientationtothe nodesfor settingaLANtopology
6. SetupTCPConnectionbetweenn(0) andn(4)
7. ApplyFTPTrafficoverTCP
8. SetupUDPConnectionbetweenn(1)andn(5)
9. ApplyCBRTraffic over UDP.
10. ApplyCSMA/CAandCSMA/CDmechanismsandstudytheirperformance
11. Scheduleeventsandruntheprogram.

PROGRAM:

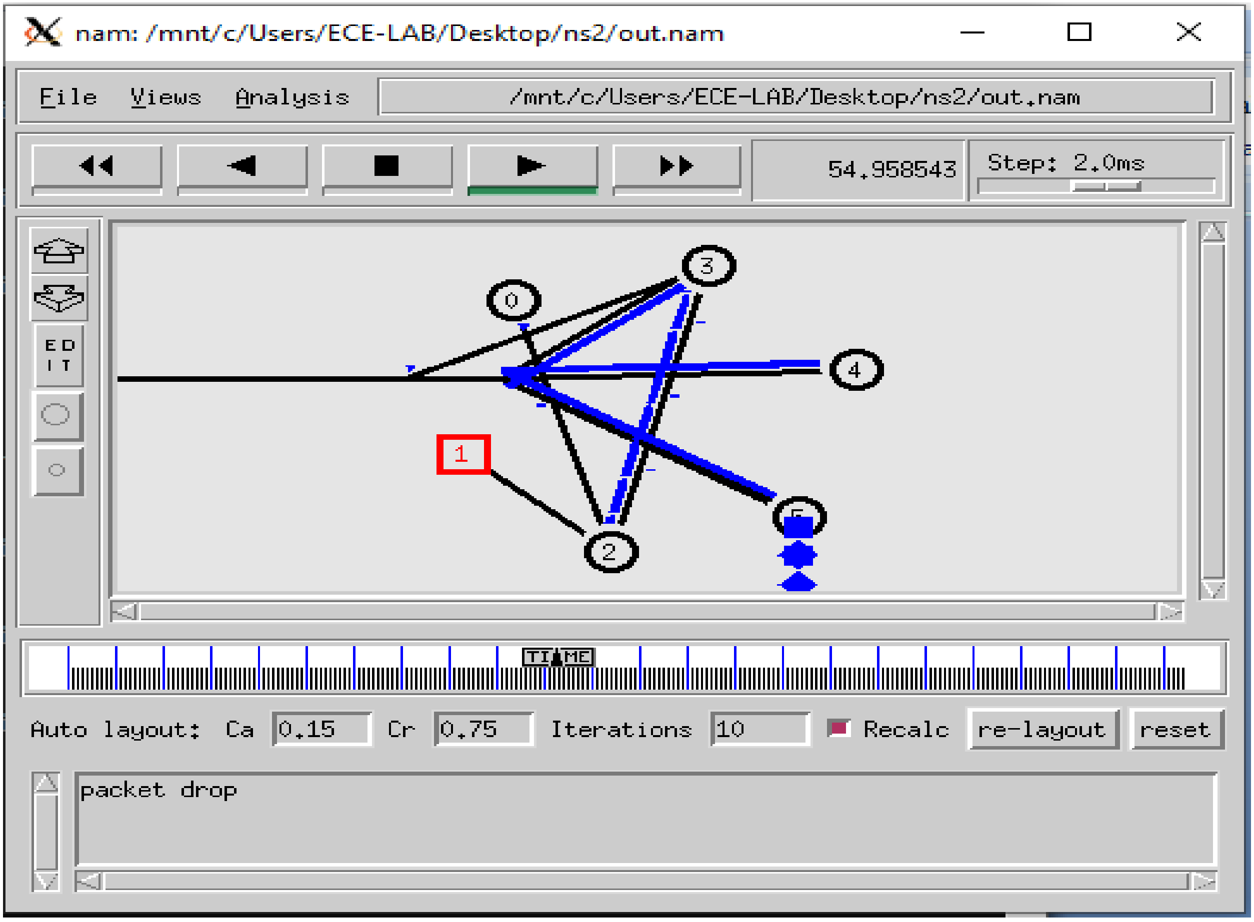
CSMA/CA

```
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
$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"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.
```

```
proc plotWindow {tcpSource file} { global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now
$cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
 $ns at 5 "$ns trace-annotate \"packet drop\""
 # PPP
$ns at 125.0 "finish"
$ns run
```

OUTPUT:

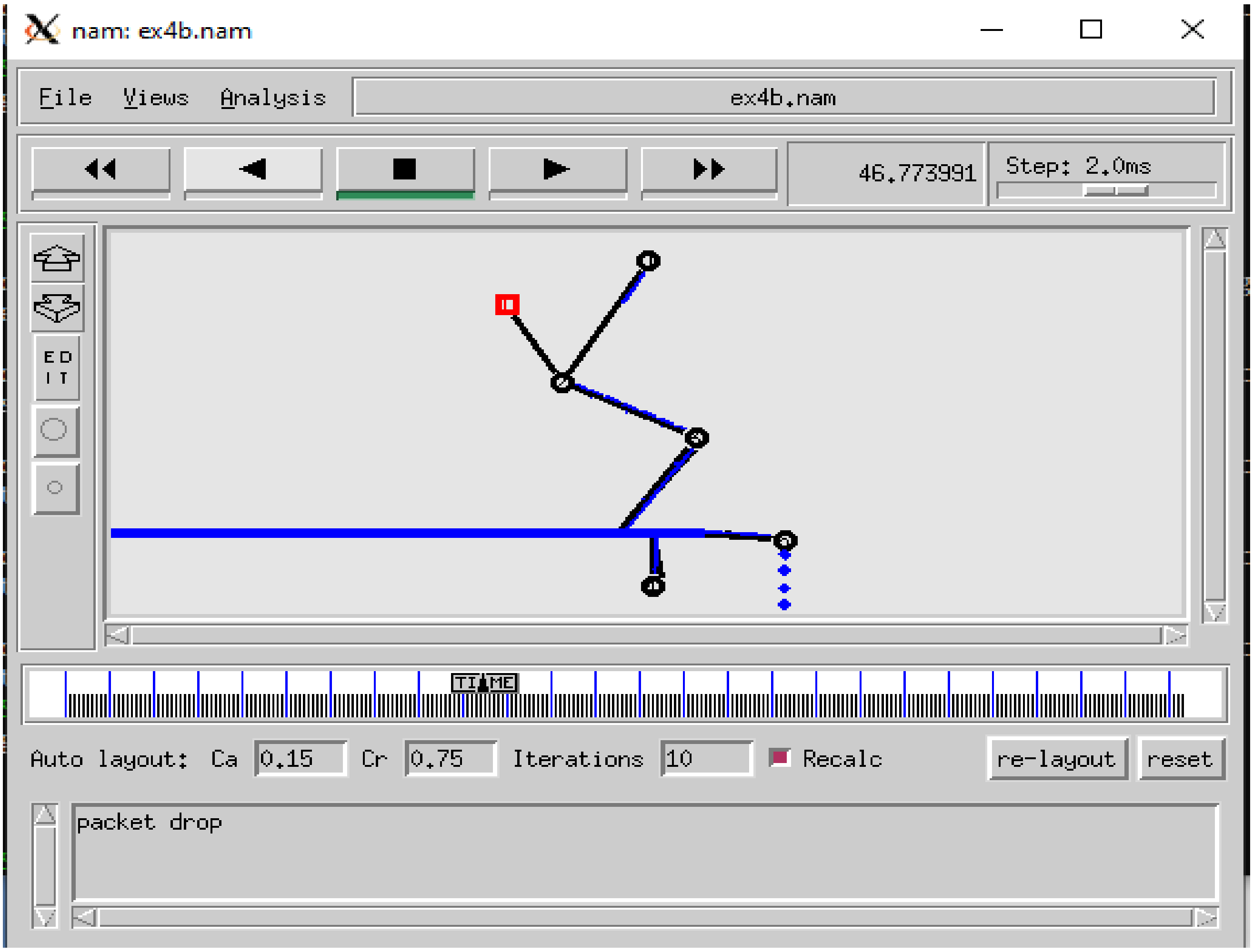


CSMA/CD

```
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 ex4b.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open ex4b.nam w]
$ns namtrace-all $file2
 #Define a 'finish' procedure
 proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam ex4b.nam &
exit 0
 }
#Create six nodes
set n0 [$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]
#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
$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"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# next procedure gets two arguments: the name of the
 # tcp source node, will be called here "tcp",
# and the name of output file.
proc plotWindow {tcpSource file} { global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
 $ns at 5 "$ns trace-annotate \"packet drop\""
 # PPP
$ns at 125.0 "finish"$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1.Explain the concept of CSMA?

2.Compare CSMA/CA and CSMA/CD.

3.What is the function of MAC layer?

4.What is DCF?

5.howdoesthecollision isavoided byCSMA/CD?

EXPT.NO.6a	SIMULATION OF DISTANCE VECTOR ROUTING ALGORITHM
DATE:	

AIM: To simulate and study the Distance Vector routing algorithm using simulation.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Distance Vector Routing is one of the routing algorithm in a Wide Area Network for computing shortest pathbetween source and destination. The Router is one main devices used in a wide area network. The main task ofthe router is Routing. It forms the routing table and delivers the packetsdepending upon the routes in the table-eitherdirectlyor viaan intermediatedevices.
Eachrouterinitiallyhasinformationaboutits allneighbors.Thenthisinformationwillbe sharedamongnodes.

ALGORITHM:

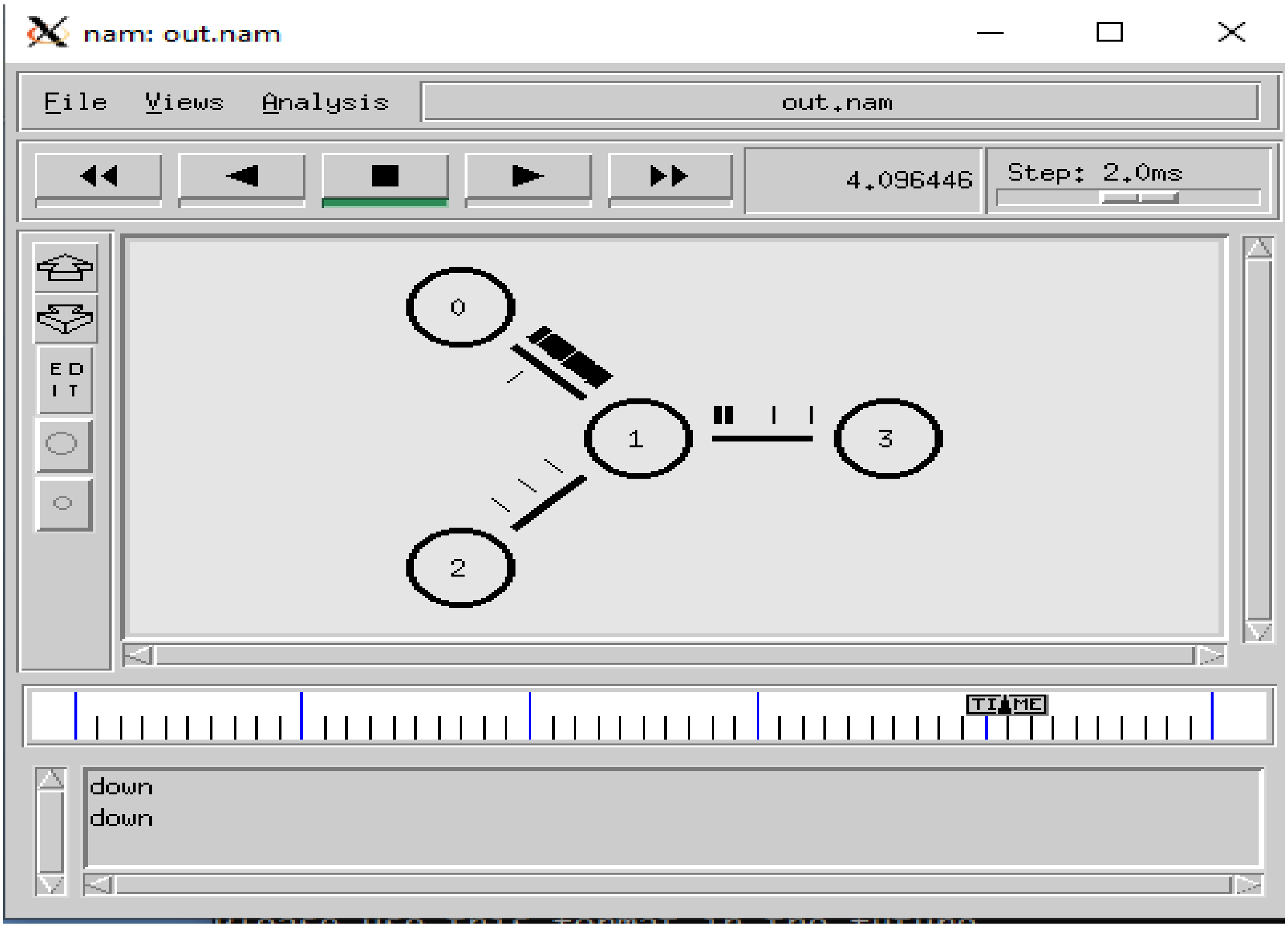
1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create n number of nodesusingforloop
5. Createduplexlinksbetween thenodes
6. SetupUDPConnectionbetweenn(0)andn(5)
7. Setupanother UDPconnectionbetweenn(1)andn(5)
8. ApplyCBRTrafficoverbothUDPconnections
9. Choosedistancevectorroutingprotocoltotransmit datafromsendertoreceiver.
10. Scheduleeventsandruntheprogram.

PROGRAM:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tr [open out.tr w]
$ns trace-all $tr
proc finish {} {
 global nf ns tr
 $ns flush-trace
 close $tr
 exec nam out.nam &
 exit 0
 }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right
```

```
$ns duplex-link-op $n2 $n1 orient right-up
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $tcp $sink
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto DV
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



EXPT.NO.6b	SIMULATION OF LINK STATE ROUTING ALGORITHM
DATE:	

AIM: To simulate and study the link state routing algorithm using simulation.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

In **link state routing,** each router shares its knowledge of its neighborhood with every other router in theinternet work. (i) **Knowledge about Neighborhood:** Instead of sending its entire routing table a router sendsinfo about its neighborhood only. (ii)**To all Routers:** each router sends this information to every other routeron the internet worknot just to its neighbor .It does so by a process called **flooding. (iii)Information sharingwhenthere isa change:**Eachroutersendsoutinformation abouttheneighbors whenthere ischange.

PROCEDURE:

The Dijkstra algorithm follows four steps to discover what is called the **shortest path tree**(routing table)
foreachrouter:Thealgorithmbeginstobuildthetreebyidentifyingitsroots.Therootrouter'streestherouteritself.Thealgorithmthenattachesallnodesthatcanbereachedfromtheroot.Thealgorithmcomparesthetree's temporary arcs and identifies the arc with the lowest cumulative cost. This arc and the node to which itconnects are now a permanent part of the shortest path tree. The algorithm examines the database and identifiesevery node that can be reached from its chosen node. These nodes and their arcs are added temporarily to thetree.
Thelasttwosteps arerepeated until everynodeinthenetworkhasbecomea permanentpart ofthetree.

ALGORITHM:

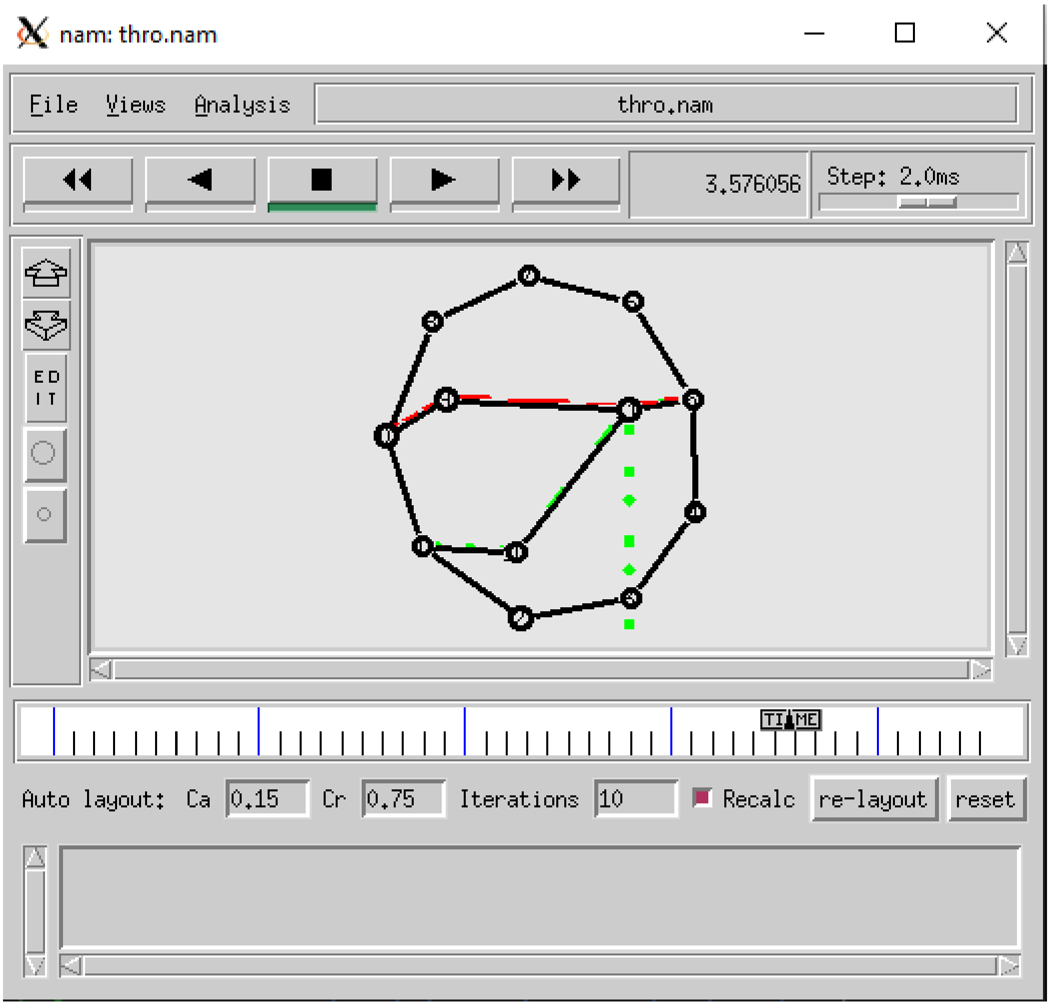
1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create n number of nodesusingforloop
5. Createduplexlinksbetween thenodes
6. SetupUDPConnectionbetweenn(0)andn(5)
7. Setupanother UDPconnectionbetweenn(1)andn(5)
8. ApplyCBRTraffic overbothUDPconnections
9. ChooseLinkstateroutingprotocoltotransmitdatafromsender toreceiver.
10. Scheduleeventsandruntheprogram.

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 1} {
$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 packet Size_ 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 4.5 "finish"
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. What is meant by subnet?

2. What is meant by Gateway?

3. What is an IP address?

4. What is MAC address?

5. What is meant by port?

EXPT.NO:07	CONGESTION CONTROL ALGORITHM (RED)
DATE:	

AIM : To Simulate Congestion Control Algorithm **(RED)** .

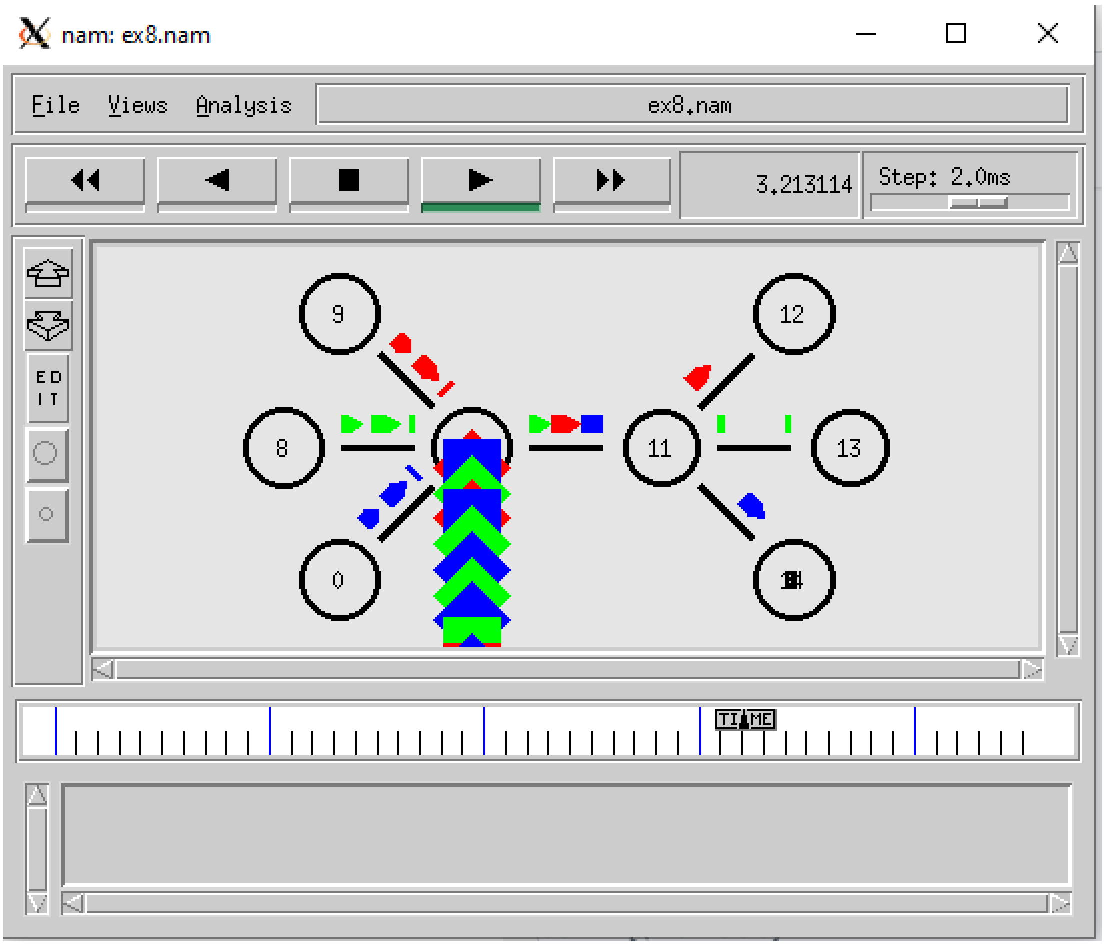
SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

PROGRAM:

```
#Create a simulator object
set ns [new Simulator]
set nr [open ex8_red.tr w]
$ns trace-all $nr
set nf [open ex8.nam w]
$ns namtrace-all $nf
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam ex8.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
$ns duplex-link $n0 $n3 1Mb 10ms RED
$ns duplex-link $n1 $n3 1Mb 10ms RED
$ns duplex-link $n2 $n3 1Mb 10ms RED
$ns duplex-link $n3 $n4 1Mb 10ms RED
$ns duplex-link $n4 $n5 1Mb 10ms RED
$ns duplex-link $n4 $n6 1Mb 10ms RED
$ns duplex-link $n4 $n7 1Mb 10ms RED
$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n6 $n4 orient left
$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
```

```
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n6 $n4 orient left
set udp0 [new Agent/UDP]
$ns attach-agent $n2 $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 $n5 $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $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 $n6 $null0
$ns connect $udp1 $null0
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set  interval_ 0.005
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
$ns connect $udp2 $null0
$udp0 set fid_ 1
$udp1 set fid_ 2
$udp2 set fid_ 3
$ns color 1 Red
$ns color 2 Green
$ns color 3 blue
$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 0.5 "$cbr2 start"
$ns at 4.0 "$cbr2 stop"
$ns at 4.2 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 0.5 "$cbr2 start"
$ns at 4.0 "$cbr2 stop"
$ns at 4.2 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. How do you classify congestion control algorithms?

2. Differentiate between flow control and congestion control

3. What is meant by traffic shaping?

4. How do you generate busty traffic?

5. Differentiate between Leaky bucket and Token bucket.

6. How do you implement Leaky bucket?

EXPT.NO.8	IMPLEMENTING A WIRELESS SENSOR NETWORK
DATE:	

AIM: To simulate a wireless sensor network using NS2.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

A wireless sensor network (WSN) consists of a large number of small sensor nodes that are deployed inthe area in which a factor is to be monitored. In wireless sensor network, energy model is one of the optionalattributes of a node. The energy model denotes the level of energy in a mobile node. The components requiredfor designing energy model includes initialEnergy, txPower, rxPower, and idlePower. The “initialEnergy”represents the level of energy the node has at the initial stage of simulation. “txPower” and “rxPower” denotesthe energy consumedfor transmitting and receiving the packets.Ifthe node is a sensor, theenergy modelshould include a special component called “sensePower”. It denotes the energy consumed during the sensingoperation. Apart from these components, it is important to specify the communication range (RXThresh_) andsensing range of a node (CSThresh_). The sample 18.tcl designs a WSN in which sensor nodes are configuredwith different communication and sensing range. Base Station is configured with highest communication range.DataTransmissionis establishedbetween nodesusingUDP agentandCBRtraffic.

ALGORITHM:

1. Createasimulatorobject
2. Defineasettingoptionsforwirelesschannel
3. Createtracefileandnamefile
4. Setuptopographyobjectandnodes
5. Provideinitiallocationofmobilenodes
6. SetupaUDPconnectionbetweennodes
7. Printingthe windowsize

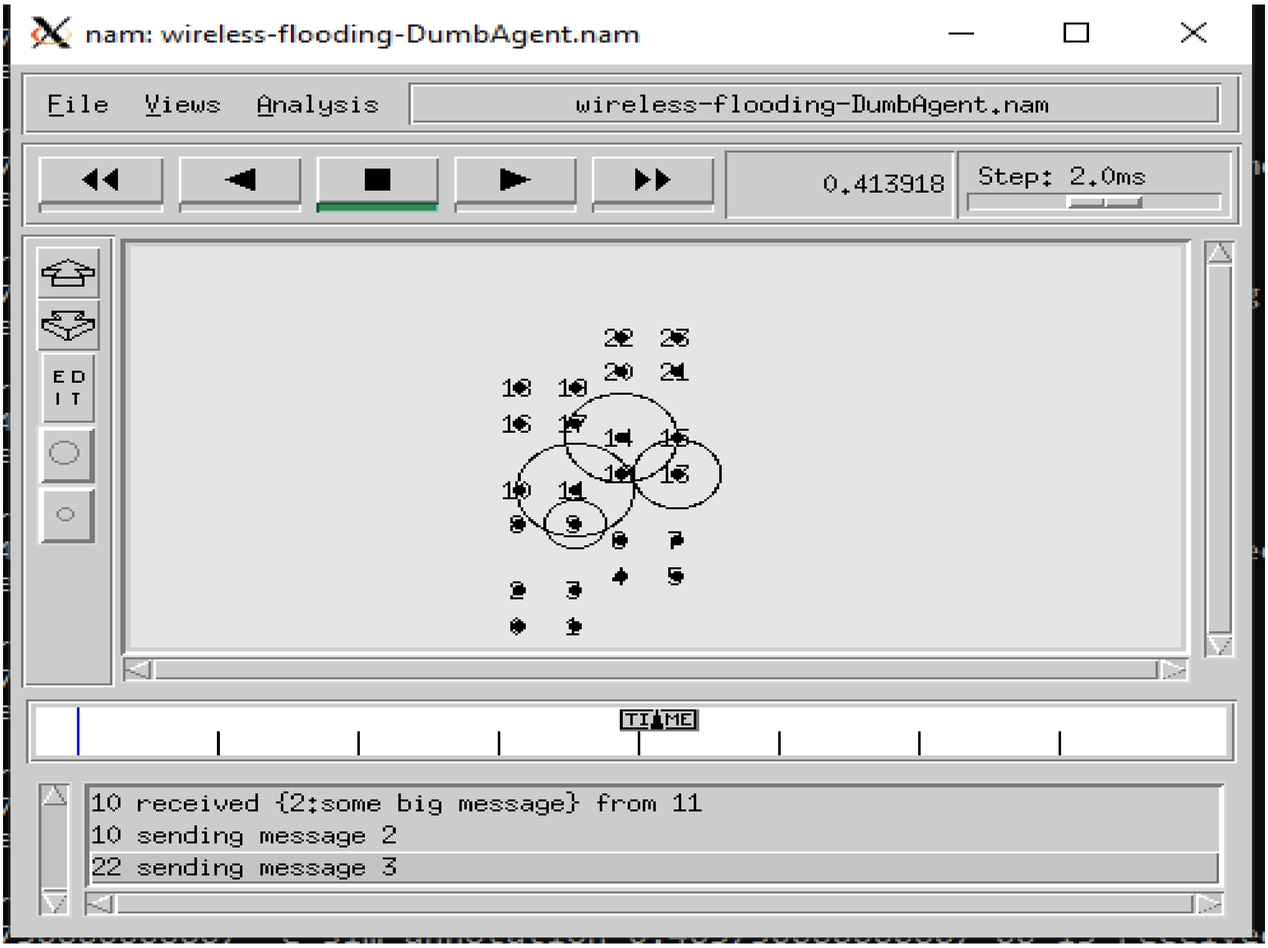
PROGRAM:

```
Mac/Simple set bandwidth_ 1Mb
set MESSAGE_PORT 42
set BROADCAST_ADDR -1
# variables which control the number of nodes and how they're grouped
# (see topology creation code below)
set group_size 4
set num_groups 6
set num_nodes [expr $group_size * $num_groups]
set val(chan)          Channel/WirelessChannel    ;#Channel Type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type
#set val(mac)          Mac/802_11                 ;# MAC type
#set val(mac)          Mac              ;# MAC type
set val(mac)           Mac/Simple
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                   ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                   ;# max packet in ifq
# DumbAgent, AODV, and DSDV work.  DSR is broken
set val(rp) DumbAgent
#set val(rp)           DSDV
#set val(rp)           DSR
#set val(rp)              AODV
```

```
# size of the topography
set val(x)              [expr 120*$group_size + 500]
set val(y)              [expr 240*$num_groups + 200]
set ns [new Simulator]
set f [open wireless-flooding-$val(rp).tr w]
$ns trace-all $f
set nf [open wireless-flooding-$val(rp).nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)
$ns use-newtrace
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $num_nodes
set chan_1_ [new $val(chan)]
$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace OFF \
                -macTrace ON \
                -movementTrace OFF \
                -channel $chan_1_
# subclass Agent/MessagePassing to make it do flooding
Class Agent/MessagePassing/Flooding -superclass Agent/MessagePassing
Agent/MessagePassing/Flooding instproc recv {source sport size data} {
    $self instvar messages_seen node_
    global ns BROADCAST_ADDR
    # extract message ID from message
    set message_id [lindex [split $data ":"] 0]
    puts "\nNode [$node_ node-addr] got message $message_id\n"
    if {[lsearch $messages_seen $message_id] == -1} {
        lappend messages_seen $message_id
        $ns trace-annotate "[$node_ node-addr] received {$data} from $source"
        $ns trace-annotate "[$node_ node-addr] sending message $message_id"
        $self sendto $size $data $BROADCAST_ADDR $sport
    } else {
        $ns trace-annotate "[$node_ node-addr] received redundant message $message_id from $source"
    }
}
Agent/MessagePassing/Flooding instproc send_message {size message_id data port} {
    $self instvar messages_seen node_
    global ns MESSAGE_PORT BROADCAST_ADDR
    lappend messages_seen $message_id
    $ns trace-annotate "[$node_ node-addr] sending message $message_id"
    $self sendto $size "$message_id:$data" $BROADCAST_ADDR $port
}
# create a bunch of nodes
for {set i 0} {$i < $num_nodes} {incr i} {
```

```
    set n($i) [$ns node]
    $n($i) set Y_ [expr 230*floor($i/$group_size) + 160*(($i%$group_size)>=($group_size/2))]
    $n($i) set X_ [expr (90*$group_size)*($i/$group_size%2) + 200*($i%($group_size/2))]
    $n($i) set Z_ 0.0
    $ns initial_node_pos $n($i) 20
}
# attach a new Agent/MessagePassing/Flooding to each node on port $MESSAGE_PORT
for {set i 0} {$i < $num_nodes} {incr i} {
    set a($i) [new Agent/MessagePassing/Flooding]
    $n($i) attach  $a($i) $MESSAGE_PORT
    $a($i) set messages_seen {}
}
# now set up some events
$ns at 0.2 "$a(1) send_message 200 1 {first message}  $MESSAGE_PORT"
$ns at 0.4 "$a([expr $num_nodes/2]) send_message 600 2 {some big message} $MESSAGE_PORT"
$ns at 0.7 "$a([expr $num_nodes-2]) send_message 200 3 {another one} $MESSAGE_PORT"
$ns at 1.0 "finish"
proc finish {} {
        global ns f nf val
        $ns flush-trace
        close $f
        close $nf
#       puts "running nam..."
        exec nam wireless-flooding-$val(rp).nam &
        exit 0
}
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. WhatisaWireless SensorNetwork?
2. HowistheNetworkConfigured?
3. Howlongdoesanodeoperatesin a batterypower?
4. Whatis therangeof awireless sensor node?
5. CanIusethe wireless nodes in outsideenvironment?

XPT.NO 9.a	NETWORKTOPOLOGY
DATE:	BUS TOPOLOGY

AIM: To create scenario and study the performance of token bus protocol through simulation.

.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)

2. PC (Windows 10)

THEORY:

Token bus is a LAN protocol operating in the MAC layer. Token bus is standardized as per IEEE 802.4. Tokenbus can operate at speeds of 5Mbps, 10 Mbps and 20 Mbps. The operation of token bus is as follows: Unliketoken ring in token bus the ring topology is virtually created and maintained by the protocol. A node can receivedata even if it is not part of the virtual ring, a node joins the virtual ring only if it has data to transmit. In tokenbus data is transmitted to the destination node only where as other control frames is hop to hop. After each datatransmission there is a solicit_successsor control frame transmitted which reduces the performance of theprotocol.

ALGORITHM:

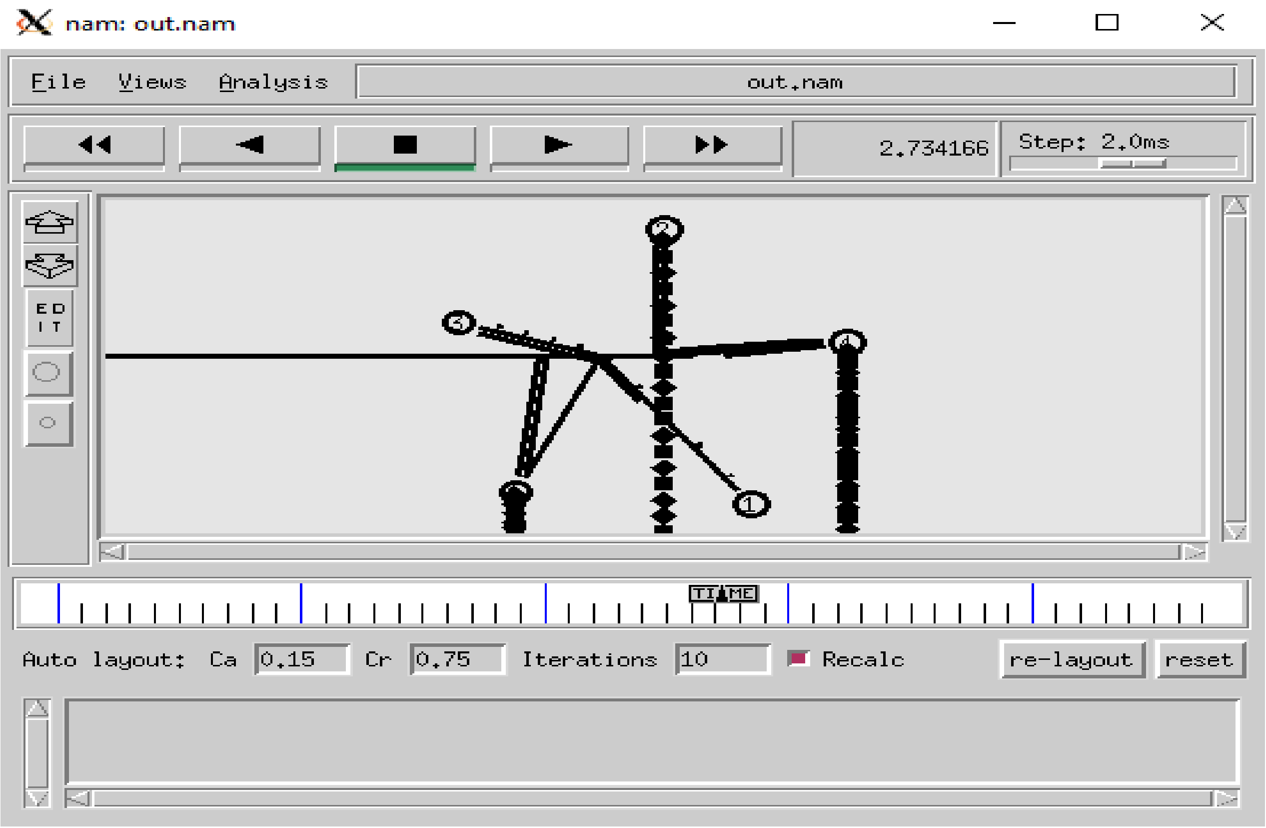
1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create fivenodes thatformsanetworknumbered from0to 4
5. Createduplexlinksbetween thenodesandaddOrientationtothe nodesfor settingaLANtopology
6. SetupTCPConnectionbetweenn(1) andn(3)
7. ApplyCBRTrafficoverTCP.
8. Scheduleeventsandruntheprogram.

PROGRAM:

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Closethetracefile
close $nf
#Executenamonthetracefile
exec nam out.nam &
exit 0
}
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Create Lanbetweenthe nodes
```

```
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LLQueue/DropTailMAC/Csma/CdChannel]
#CreateaTCPagentand 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
#Connectthetraffic sources withthetrafficsink
$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
#ScheduleeventsfortheCBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Callthefinishprocedure after 5secondsofsimulationtime
$ns at 5.0 "finish"
#Runthesimulation
$ns run
```

OUTPUT:



EXPT.NO 9.b	NETWORK TOPOLOGY
DATE:	RING TOPOLOGY

AIM: To create scenario and study the performance of token ring protocol through simulation.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Token ring is a LAN protocol operating in the MAC layer. Token ring is standardized as per IEEE 802.5. Tokenring can operate at speeds of 4mbps and 16 mbps. The operation of token ring is as follows: When there is notraffic on the network a simple 3-byte token circulates the ring. If the token is free (no reserved by a station ofhigher priority as explained later) then the station may seize the token and start sending the data frame. As theframe travels around the ring ach station examines the destination address and is either forwarded (if therecipient is another node) or copied. After copying4 bits of the last byte is changed. This packet then continuesaround the ring till it reaches the originating station. After the frame makes a round trip the sender receives theframeandreleases anew token onto the ring.

ALGORITHM:

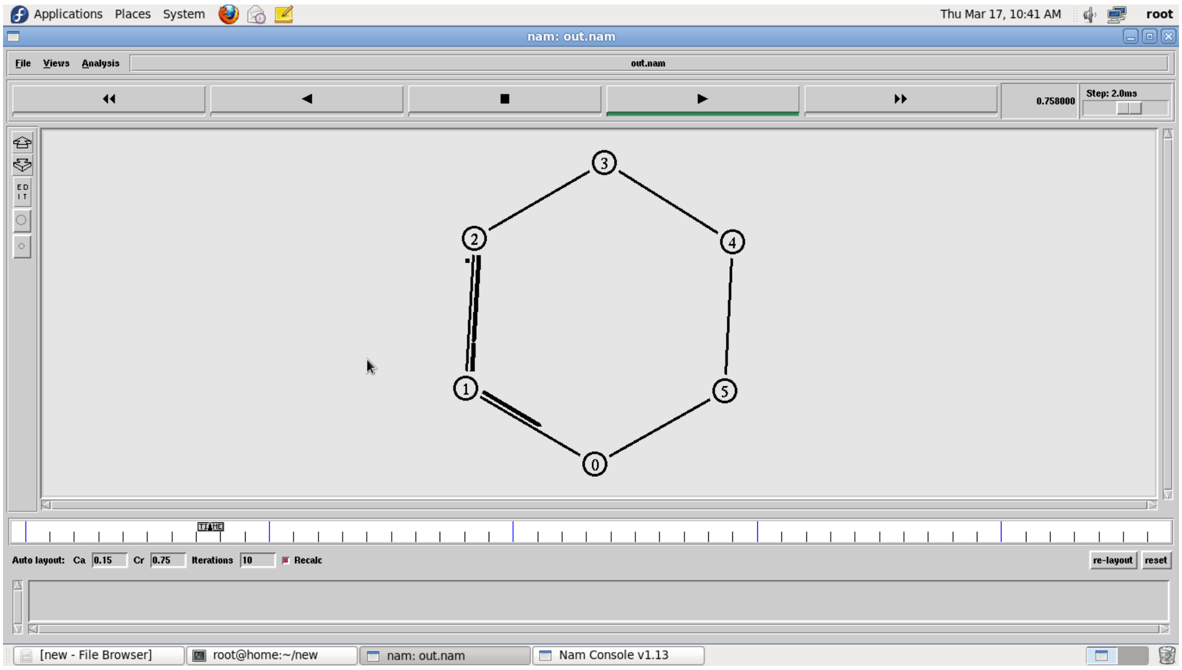
1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create fivenodes thatformsanetworknumbered from0to 4
5. Create duplexlinksbetweenthenodestoformaRingTopology.
6. SetupTCPConnectionbetweenn(1) andn(3)
7. ApplyCBRTrafficoverTCP
8. Scheduleeventsandruntheprogram.

PROGRAM:

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file close $nf
#Execute nam on the trace file
exec nam out.nam &
exit 0
}
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
```

```
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n0 $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 $n4 $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 10 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

OUTPUT:



EXPT.NO 9.C	NETWORK TOPOLOGY STAR TOPOLOGY
DATE:	

AIM: To create scenario and study the performance of star Topology through simulation.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Star networks are one of the most common computer network topologies. In its simplest form, a star networkconsists of one central switch, hub or computer, which acts as a conduit to transmit messages. This consists of acentral node, to which all other nodes are connected; this central node provides a common connection point forallnodes through ahub. In startopology, every node(computerworkstation orany otherperipheral)isconnected to a central node called a hub or switch. The switch is the server and the peripherals are the clients.Thus, the hub and leaf nodes, and the transmission lines between them, form a graph with the topology of a star.If the central node is passive, the originating node must be able to tolerate the reception of an echo of its owntransmission, delayed by the two-way transmission time (i.e. to and from the central node) plus any delaygenerated in the central node. An active star network has an active central node that usually has the means topreventecho-related problems.

The star topology reduces the damage caused by line failure by connecting all of the systems to a central node.When appliedto abus-basednetwork,thiscentralhubrebroadcastsalltransmissionsreceivedfrom anyperipheral node to all peripheral nodes on the network, sometimes including the originating node. All peripheralnodes may thus communicate with all others by transmitting to, and receiving from, the central node only. Thefailure of a transmission line linking any peripheral node to the central node will result in the isolation of thatperipheralnodefromallothers,buttherestof the systems willbeunaffected.

ALGORITHM:

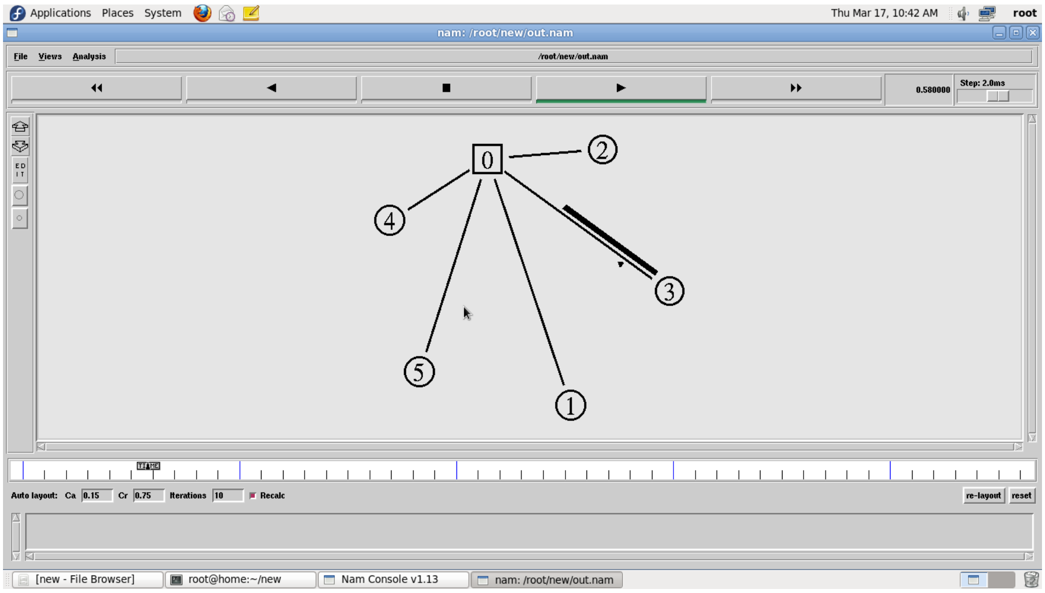
1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create sixnodesthatformsanetworknumberedfrom0to5
5. Create duplexlinksbetween thenodestoformaSTARTopology
6. SetupTCPConnectionbetweenn(1) andn(3)
7. ApplyCBRTrafficoverTCP
8. Scheduleeventsandrunthe program.

PROGRAM:

```
set ns [new Simulator]
set nf [open ex1.nam w]
#Open the nam trace file
set nf [open ex1.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Executenam on the trace file
exec nam ex1.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]
#Change the shape of center node in a star topology
$n0 shape square
#Create links between the nodes
$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
#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 1.0 "finish"
 #Run the simulation
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1.WhataretheDifferenttopologiesavailableinnetworks?

2.Whichtopologyrequires multipointconection?

3.Datacommunicationsystemwithinacampusiscalledas?

4.What is meant byWAN?

5.Explaintheworkingof Ringtopology?

EXPNO:10	IMPLEMENTATION OF DIFFERENT LANs USING SWITCH / HUB / ROUTER AS INTERCONNECTING DEVICE
DATE:	

AIM: To SimulateDifference between Hub, Switch and Router using NS2.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Hub:

A Hub is just a connector that connects the wires coming from different sides. There is no signal processing or regeneration. It is an electronic device that operates only on physical layers of the OSI model.

It is also known as a repeater as it transmits signal to every port except the port from where signal is received. Also, hubs are not that intelligent in communication and processing information for 2nd and 3rd layer.

Switch:

Switch is a point to point communication device. It operates at the data link layer of OSI model. It uses switching table to find out the correct destination.

Basically, it is a kind of bridge that provides better connections. It is a kind of device that set up and stop the connections according to the requirements needed at that time. It comes up with many features such as flooding, filtering and frame transmission

Router:

Routers are the multiport devices and more sophisticated as compared to repeaters and bridges. It contains a routing table that enables it to make decision about the route i.e. to determine which of several possible paths between the source and destination is the best for a particular transmission.

PROGRAM:

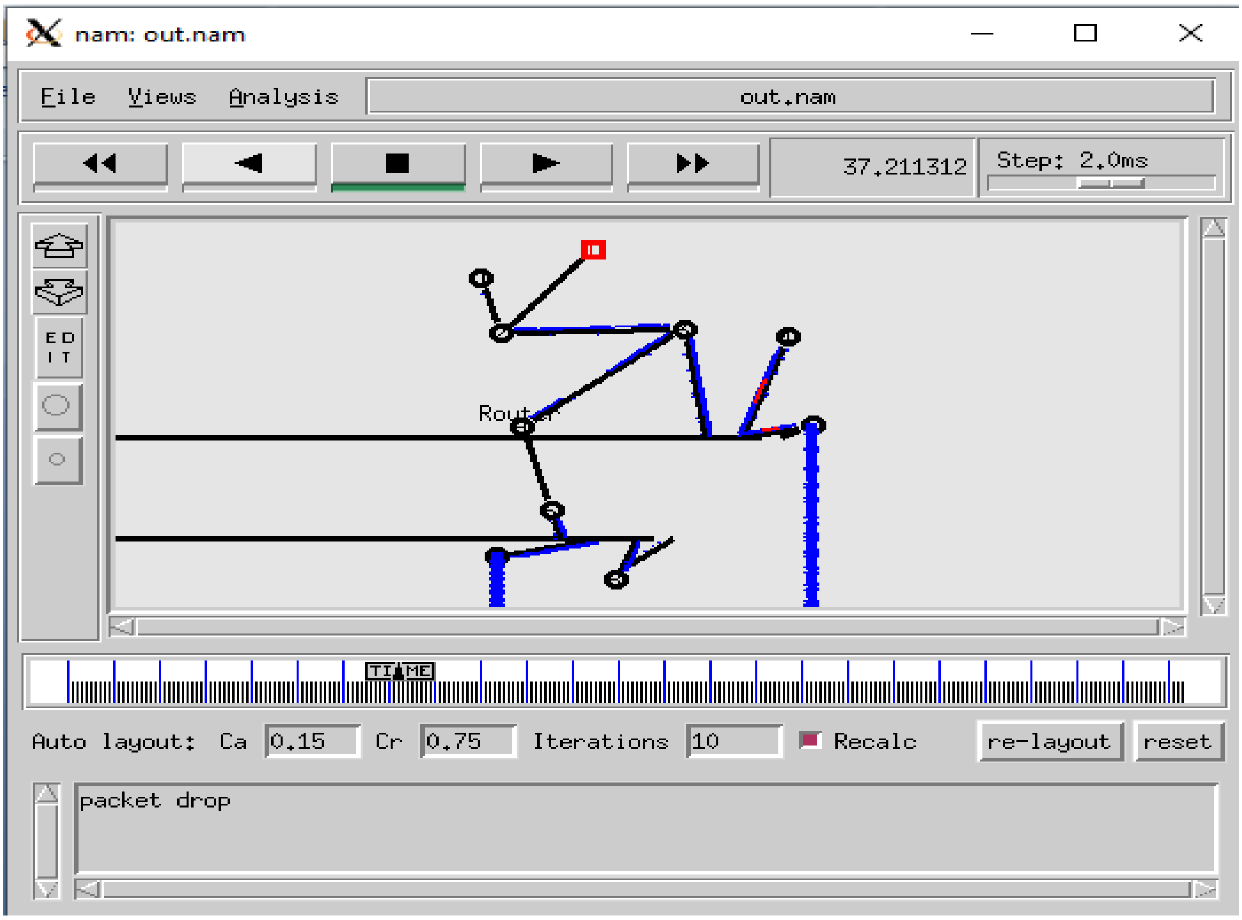
```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#Define a 'finish' 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]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
$n9 label "Router"
$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
$ns duplex-link $n9 $n3 2Mb 10ms DropTail
$ns duplex-link $n9 $n6 2Mb 10ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd
Channel]
set lan [$ns newLan "$n6 $n7 $n8" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd
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
set tcp3 [new Agent/TCP]
$ns attach-agent $n9 $tcp3
set sink4 [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink4
$ns connect $tcp3 $sink4
$tcp3 set fid_ 1
$tcp3 set window_ 8000
$tcp3 set packetSize_ 552
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

```
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp3
$ftp2 set type_ FTP
#Setup a TCP connection
set tcp1 [new Agent/TCP/Newreno]
$ns attach-agent $n6 $tcp1
set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n8 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
#Setup a FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 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"
$ns at 1.0 "$ftp start"
$ns at 1.0 "$ftp1 start"
$ns at 1.0 "$ftp2 start"
$ns at 200.0 "$ftp1 stop"
$ns at 200.0 "$ftp2 stop"
$ns at 200.0 "$ftp stop"
$ns at 200.5 "$cbr stop"
# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
```

```
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
# PPP
$ns at 125.0 "finish"
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. Mention the difference between Router and Hub
2. State the difference between Hub and Switch
3. What is the difference between Router and Layer-3 Switch
4. State the difference between Router and Switch
5. State the difference between Thread Context Switch and Process Context Switch

ADVANCED EXPERIMENT

EXPT.NO:01	SIMULATE A MOBILE ADHOC NETWORK
DATE:	

AIM: To simulate a Mobile Adhoc network(MANET) using NS2.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

A mobile ad hoc network or MANET does not depend on a fixed infrastructure for its networking operation. MANET is an autonomous and short-lived association of group of mobile nodes that communicate with each other over wireless links. A node can directly communicate to then odes that lie within its communication range. If a node wants to communicate with a node that is not directly within its communication range, it uses intermediate nodes as routers.

ALGORITHM:

1. Create simulator object
2. Set the values for the parameter
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create the nodes that forms a network numbered from 0 to 3
5. Schedule events and run the program.

PROGRAM:

```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen)  50
set val(nn)     3
set val(rp)   DSDV
#routing protocol
#set val(x)  100    ;
#X dimension of topography
#set val(y)  100             ;
set ns [new Simulator]
set tf [open $val(rp).tr w]
$ns trace-all $tf
set tf1 [open $val(rp).nam w]
$ns namtrace-all-wireless  $tf1 100 100
set topo [new Topography]
$topo load_flatgrid 100 100
#create-god
create-god $val(nn)
set chan_1_ [new $val(chan)]
$ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
```

```
		-ifqType $val(ifq) \
		-ifqLen $val(ifqlen) \
		-antType $val(ant) \
		-propType $val(prop) \
		-phyType $val(netif) \
		-topoInstance $topo \
		-agentTrace ON \
		-routerTrace ON \
		-macTrace ON \
		-movementTrace ON \
		-channel $chan_1_
set node0 [$ns node]
set node1 [$ns node]
set node2 [$ns node]
$ns initial_node_pos $node0 10
$ns initial_node_pos $node1 10
$ns initial_node_pos $node2 10
$node0 set X_ 25.0
$node0 set Y_ 50.0
$node0 set Z_ 0.0
$node1 set X_ 50.0
$node1 set Y_ 50.0
$node1 set Z_ 0.0
$node2 set X_ 65.0
$node2 set Y_ 50.0
$node2 set Z_ 0.0
set tcp1 [new Agent/TCP]
$ns attach-agent $node0 $tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $node2 $sink1
$ns connect $tcp1 $sink1
$tcp1 set packetSize_ 1500
$ns at 10.0 "$node1 set dest 50.0 90.0.0.0"
$ns at 50.0 "$node1 set dest 50.0 10.0.0.0"
$ns at 0.5 "$ftp start"
$ns at 10.0 "$ftp stop"
$ns at 10.0  "finish"
proc finish {} {
global ns tf tf1 val
$ns flush-trace
close $tf
close $tf1
exec nam $val(rp).nam &
exit 0
}$ns run
```

OUTPUT:

RESULT:

CONCLUSION:

VIVAQUESTIONS:

1. What is the meaning of the word Adhoc?

2. Mention some applications of Adhoc Networks?

3. Mention some ofthe routing protocolsfor Adhoc Network.

4. What are the common issues in Adhoc network?

5. Is Multihop communication possible in Adhoc network?

EXPT.NO:02	SIMULATION OF GO BACK N PROTOCOL AND SELECTIVE REPEAT PROTOCOLS
DATE:	

AIM: To Simulate and to study of Go Back N protocol and Selective Repeat using NS2.

SOFTWARE REQUIRED: 1. Network Simulation tool (ns2)
2. PC (Windows 10)

THEORY:

Go Back N is a connection oriented transmission. The sender transmits the frames continuously. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size. The sender has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously. The size of the window depends on the protocol designer.

Selective Repeat ARQ is a specific instance of the Automatic Repeat-reQuest (ARQ) Protocol. It may be used as a protocol for the delivery and acknowledgement of message units, or it may be used as a protocol for the delivery of subdivided message sub-units. When used as the protocol for the delivery of messages, the sending process continues to send a number of frames specified by a window size even after a frame loss. Unlike Go Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error. The receiver process keeps track of the sequence number of the earliest frame it has not received, and sends that number with every ACK it sends. If a frame from the sender does not reach the receiver, the sender continues to send subsequent frames until it has emptied its window. The receiver continues to fill its receiving window with the subsequent frames, replying each time with an ACK containing the sequence number of the earliest missing frame. Once the sender has sent all the frames in its window, it re-sends the frame number given by the ACKs, and then continues where it left off. The size of the sending and receiving windows must be equal, and half the maximum sequence number (assuming that sequence numbers are numbered from 0 to n-1) to avoid miscommunication in all cases of packets being dropped. To understand this, consider the case when all ACKs are destroyed. If the receiving window is larger than half the maximum sequence number, some, possibly even all, of the packages that are resent after timeouts are duplicates that are not recognized as such. The sender moves its window for every packet that is acknowledged.

OPERATIONS:

1. A station may send multiple frames as allowed by the window size.
2. Receiver sends an ACK i if frame i has an error. After that, the receiver discards all incoming frames until the frame with error is correctly retransmitted.
3. If sender receives an ACK i it will retransmit frame i and all packets i+1, i+2,... which have been sent, but not been acknowledged

ALGORITHM FOR GO BACK N

1. The source node transmits the frames continuously.
2. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
3. The source node has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
4. The size of the window depends on the protocol designer.
5. For the first frame, the receiving node forms a positive acknowledgement if the frame is received without error.
6. If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
7. If the subsequent frame is received with error, the cumulative acknowledgment error-free frames are

transmitted. If in the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly even the frames received without error after the receipt of a frame with error are neglected.
8. The source node retransmits all frames of window from the first error frame

ALGORITHM: SELECTIVE REPEAT

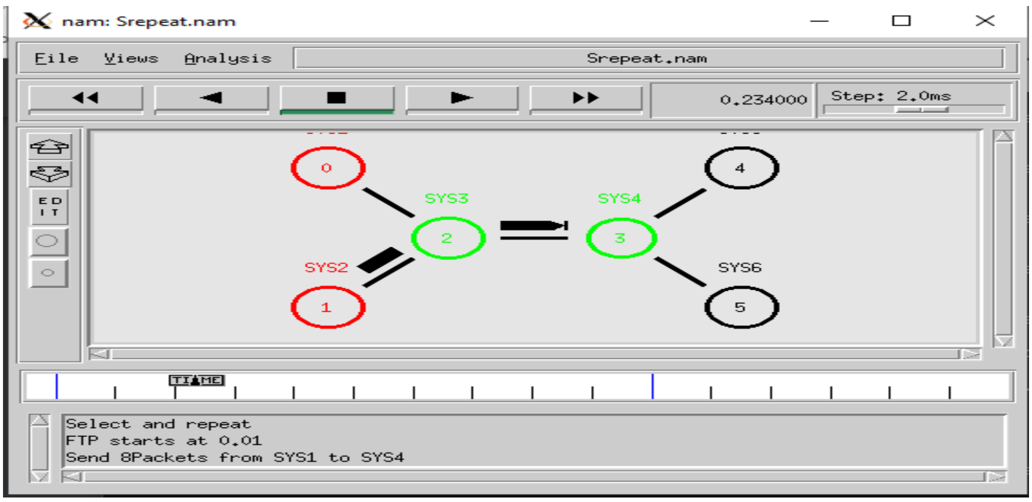
1. The source node transmits the frames continuously.
2. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
3. The source node has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
4. The receiver has a buffer to store the received frames. The size of the buffer depends upon the window size defined by the protocol designer.
5. The size of the window depends according to the protocol designer.
6. The source node transmits frames continuously till the window size is exhausted. If any of the frames are received with error only those frames are requested for retransmission (with a negative acknowledgement)
7. If all the frames are received without error, a cumulative positive acknowledgement is sent.
8. If there is an error in frame 3, an acknowledgement for the frame 2 is sent and then only Frame 3 is retransmitted. Now the window slides to get the next frames to the window.
9. If acknowledgment is transmitted with error, all the frames of window are retransmitted. Else ordinary window sliding takes place. (* In implementation part, Acknowledgment error is not considered)
10.If all the frames transmitted are errorless the next transmission is carried out for the new window.
11.This concept of repeating the transmission for the error frames only is called Selective Repeat transmission flow control protocol.

PROGRAM FOR GOBACK N:

```
#send packets one by one
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]
$n0 color "red"
$n1 color "red"
$n2 color "green"
$n3 color "green"
$n4 color "black"
$n5 color "black"
$n0 shape circle ;
$n1 shape circle ;
$n2 shape circle ;
$n3 shape circle ;
$n4 shape circle ;
$n5 shape circle ;
$ns at 0.0 "$n0 label SYS1"
$ns at 0.0 "$n1 label SYS2"
$ns at 0.0 "$n2 label SYS3"
$ns at 0.0 "$n3 label SYS4"
$ns at 0.0 "$n4 label SYS5"
$ns at 0.0 "$n5 label SYS6"
set nf [open Srepeat.nam w]
$ns namtrace-all $nf
set f [open Srepeat.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
$ns queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link $n3 $n5 1Mb 10ms DropTail
$ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP]
$tcp set fid 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 8"
$ns at 0.06 "$tcp set maxcwnd 8"
$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.30 "$tcp set windowlnit 1"
$ns at 0.30 "$tcp set maxcwnd 1"
$ns at 0.30 "$ns queue-limit $n3 $n4 10"
$ns at 0.47 "$ns detach-agent $n1 $tcp;$ns detach-agent $n4 $sink"
$ns at 1.75 "finish"
$ns at 0.0 "$ns trace-annotate \"Select and repeat\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 8Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs in 4th packet \""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 from SYS1 to SYS4\""
$ns at 1.5 "$ns trace-annotate \"FTP stops\""
proc finish {} {
global ns nf
$ns flush-trace
close $nf
puts "filtering..."
#exec tclsh../bin/namfilter.tcl srepeat.nam
#puts "running nam..."
exec nam Srepeat.nam &
exit 0
}
$ns run
```

OUTPUT:

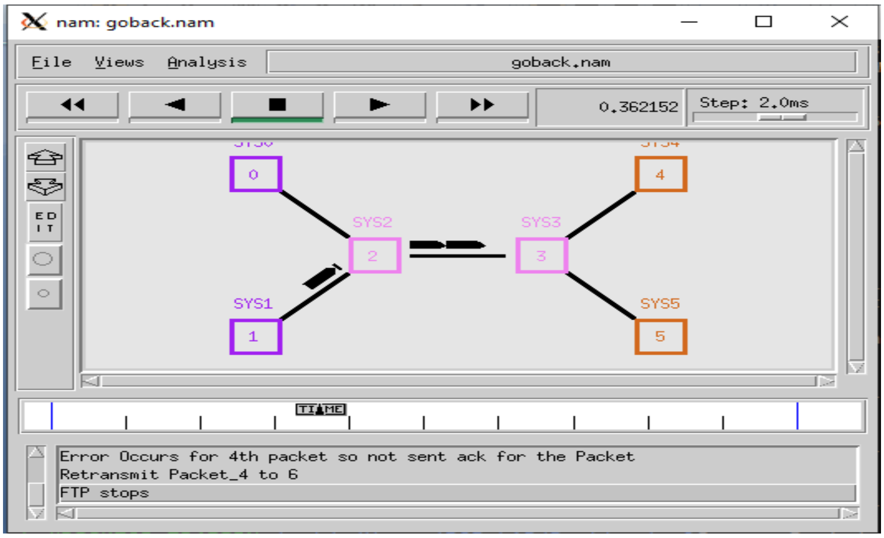


PROGRAM FOR SELECTIVE REPEAT:

```
#send packets one by one
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]
$n0 color "purple"
$n1 color "purple"
$n2 color "violet"
$n3 color "violet"
$n4 color "chocolate"
$n5 color "chocolate"
$n0 shape box ;
$n1 shape box ;
$n2 shape box ;
$n3 shape box ;
$n4 shape box ;
$n5 shape box ;
$ns at 0.0 "$n0 label SYS0"
$ns at 0.0 "$n1 label SYS1"
$ns at 0.0 "$n2 label SYS2"
$ns at 0.0 "$n3 label SYS3"
$ns at 0.0 "$n4 label SYS4"
$ns at 0.0 "$n5 label SYS5"
set nf [open goback.nam w]
$ns namtrace-all $nf
set f [open goback.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 20ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
$ns queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 20ms DropTail
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link $n2 $n3 1Mb 20ms DropTail
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 20ms DropTail
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link $n3 $n5 1Mb 20ms DropTail
$ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP]
$tcp set fid 1
```

```
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 6"
$ns at 0.06 "$tcp set maxcwnd 6"
$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.305 "$tcp set windowlnit 4"
$ns at 0.305 "$tcp set maxcwnd 4"
$ns at 0.368 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n4 $sink"
$ns at 1.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Goback N end\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 6Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs for 4th packet so not sent ack for the Packet\""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 to 6\""
$ns at 1.0 "$ns trace-annotate \"FTP stops\""
proc finish {} {
global ns nf
$ns flush-trace
close $nf
puts "filtering..."
#exec tclsh../bin/namfilter.tcl goback.nam
#puts "running nam..."
exec nam goback.nam &
exit 0
}
$ns run
```

OUTPUT:



RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is Go back-N protocol?

2. What is Go back-N protocol ARQ?

3. What is flow control?

4. What is fixed size framing?

5. What is pipelining?