PRACTICAL FILE

MOBILE COMPUTING
(ITMDC04)

Master of Technology
in
Mobile Communication and Network Technology

By

Ineesh Singh
2023PMN4207

Submitted to:

Mr. Karan Gupta
Department of Information Technology

PRACTICAL FILE

MOBILE COMPUTING
(ITMDC04)

Master of Technology
in
Mobile Communication and Network Technology

By

Harshit

2023PMN4206

Submitted to:

Mr. Karan Gupta
Department of Information Technology

PRACTICAL FILE

MOBILE COMPUTING
(ITMDC04)

Master of Technology
in
Mobile Communication and Network Technology

By

Himanshu
2023PMN4208

Submitted to:

Mr. Karan Gupta
Department of Information Technology

PRACTICAL FILE

MOBILE COMPUTING
(ITMDC04)

Master of Technology
in
Mobile Communication and Network Technology

By

Simarjeet Singh
2023PMN4221

Submitted to:

Mr. Karan Gupta
Department of Information Technology

PRACTICAL FILE

MOBILE COMPUTING
(ITMDC04)

Master of Technology
in
Mobile Communication and Network Technology

By

Ramanjeet

Singh 2023PMN4222

Submitted to:

Mr. Karan Gupta
Department of Information Technology

LAB ASSIGNMENT – 01

AIM/OBJECTIVE:

Download NS-2 simulator and Install in Unix System.

THEORY:

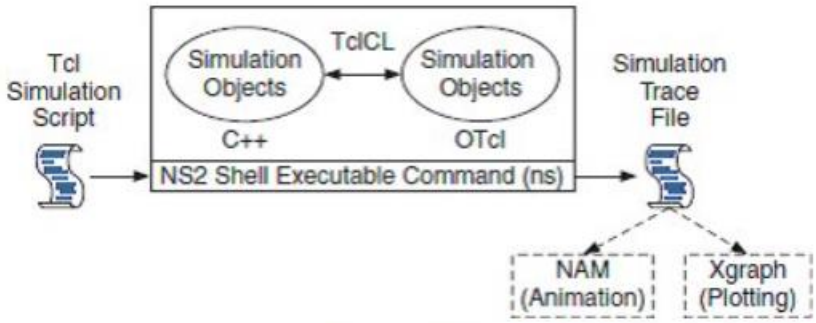
NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.

Features: -

- It's a networking research discrete event simulator.
- It has a lot of features for simulating protocols including TCP, FTP, UDP, HTTPS, and DSR.
- It is capable of simulating both wired and wireless networks.
- It is mostly based on Unix.
- Its scripting language is TCL.
- Tclcl is a C++ and otcl linkage language.
- Scheduler for discrete events.

Architecture: -

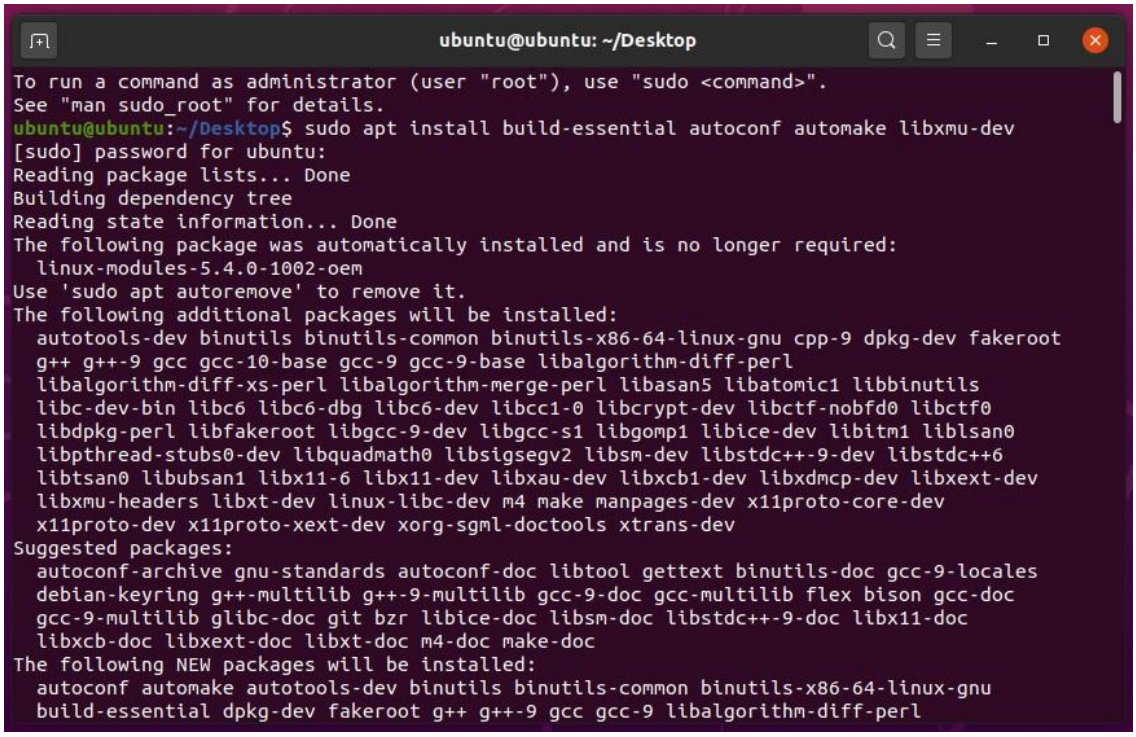
- NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl).
- The C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events.
- The C++ and the OTcl are linked together using TclCL.



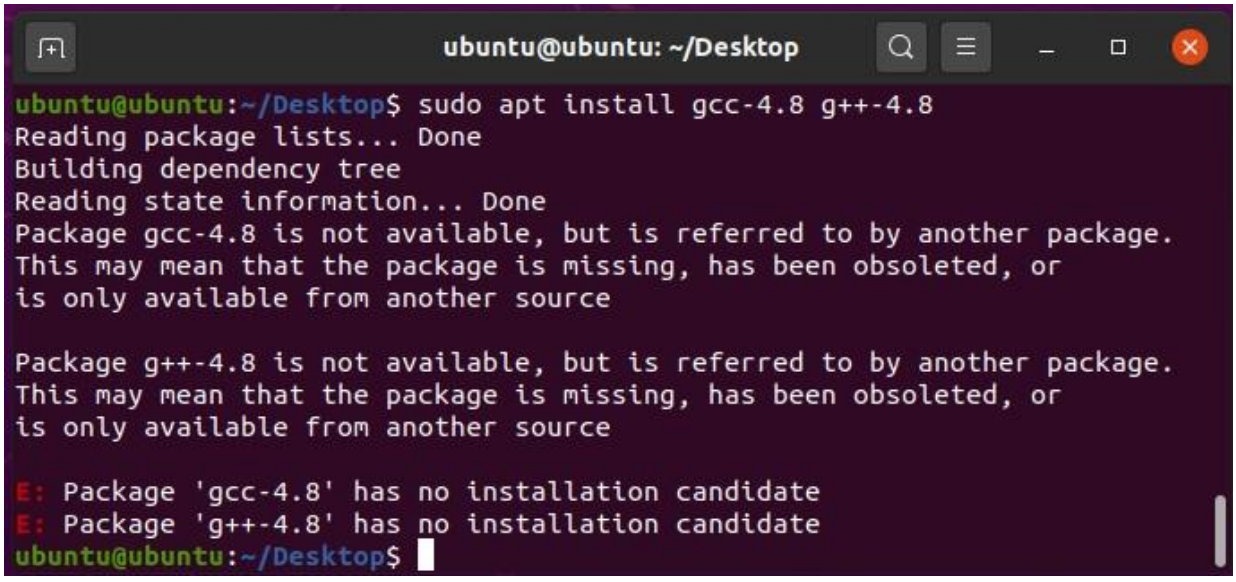
STEPS: -

1. Install the basic libraries like

   $] sudo apt install build-essential autoconf automake libxmu-dev

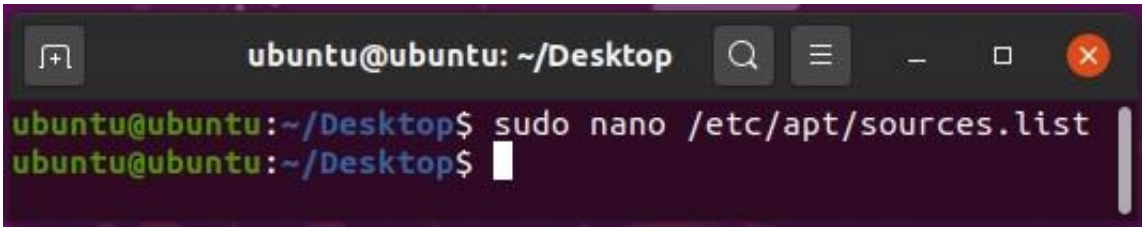


2. install gcc-4.8 and g++-4.8



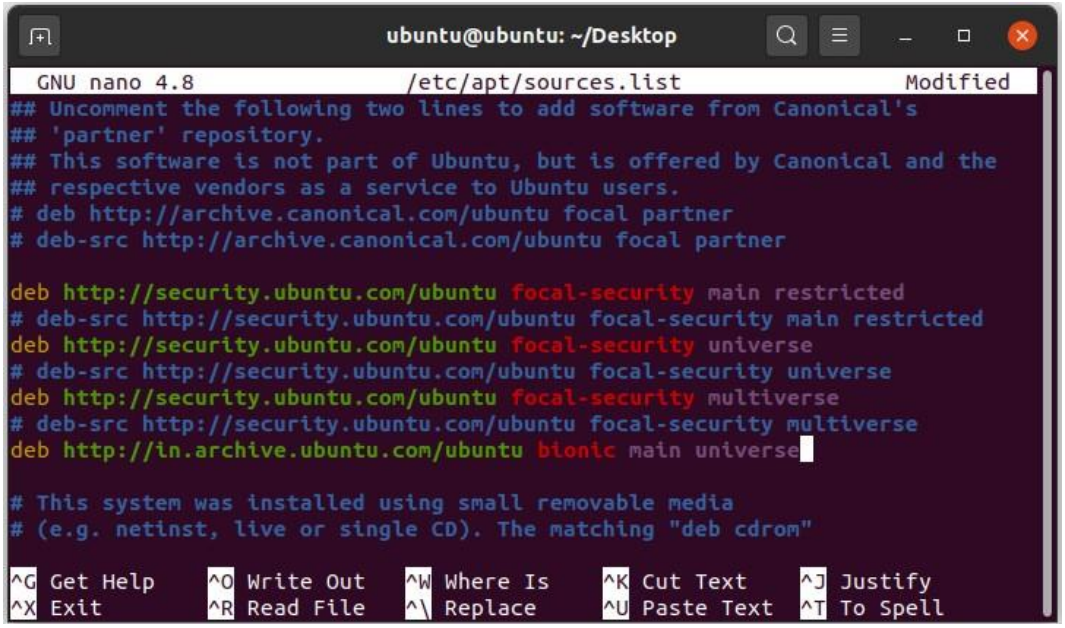
   open the file using sudo mode

   $] sudo nano /etc/apt/sources.list

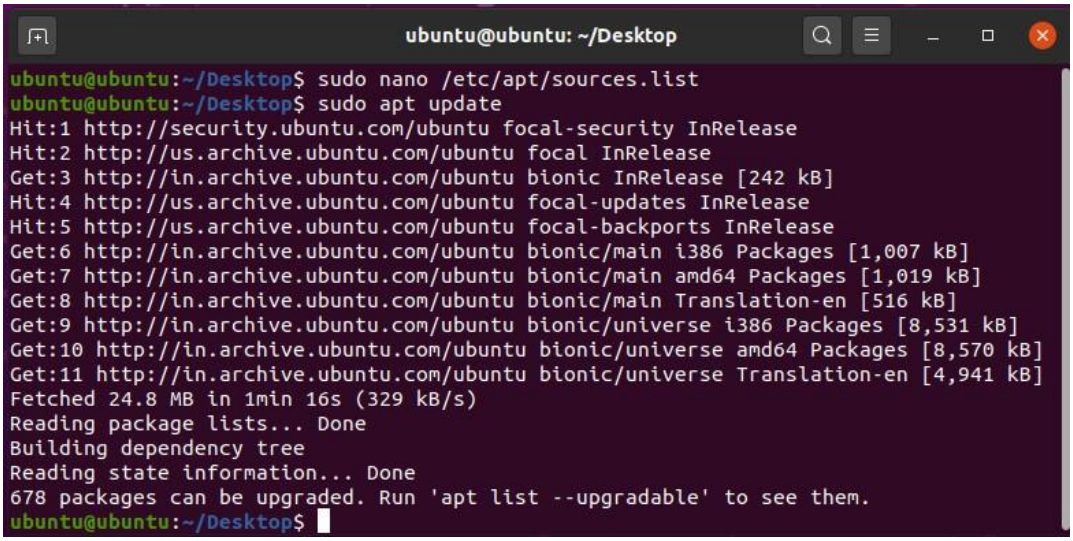


Include the following line

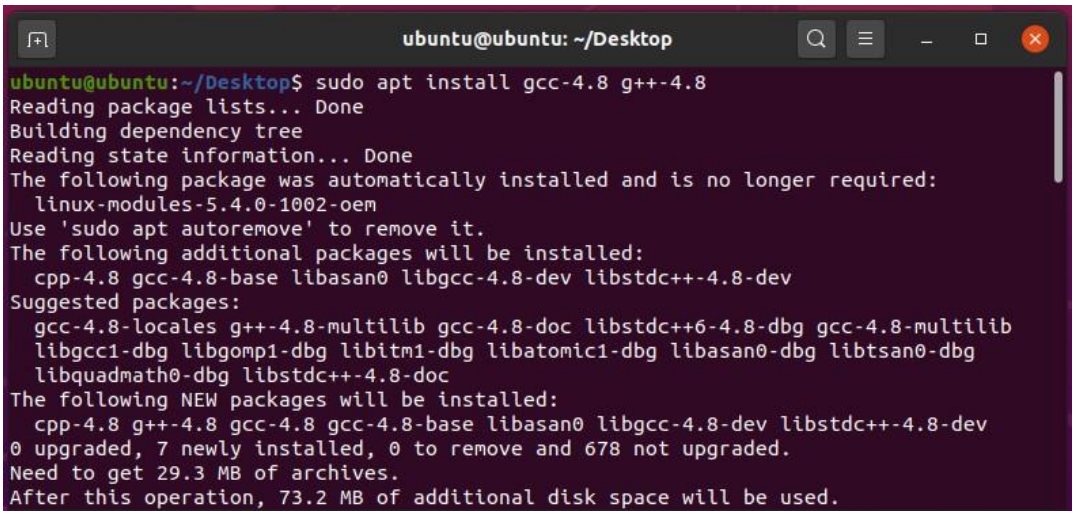
deb http://in.archive.ubuntu.com/ubuntu bionic main universe



$] sudo apt update

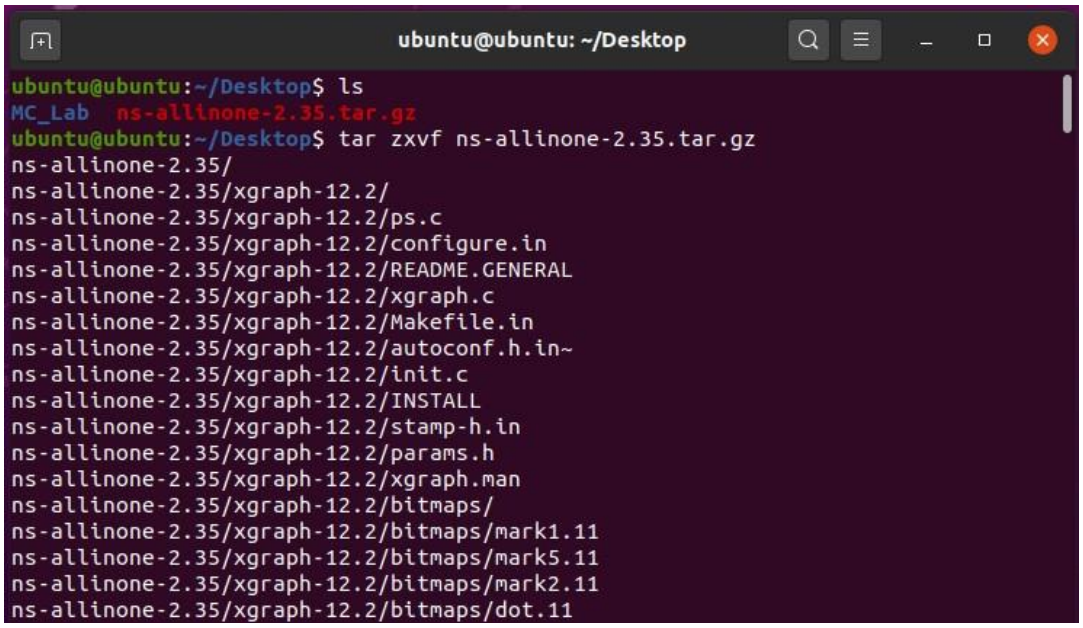


$] sudo apt install gcc-4.8 g++-4.8

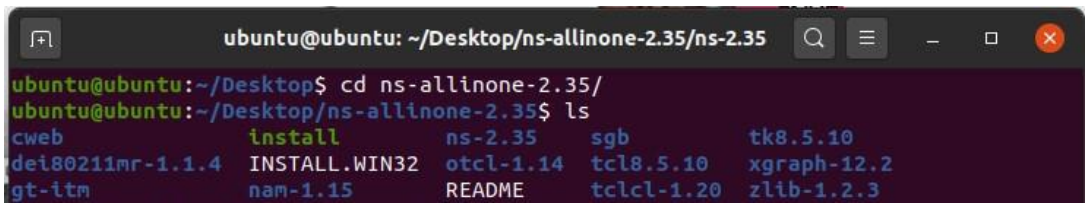


3. Unzip the ns2 packages to home folder

$] tar zxvf ns-allinone-2.35.tar.gz

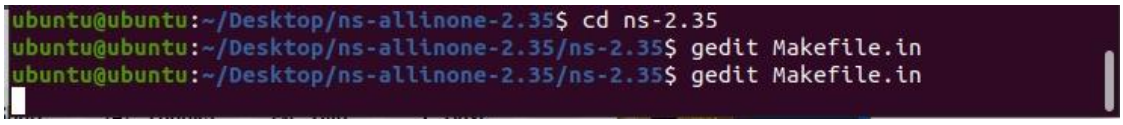


$] cd ns-allinone-2.35/ns-2.35

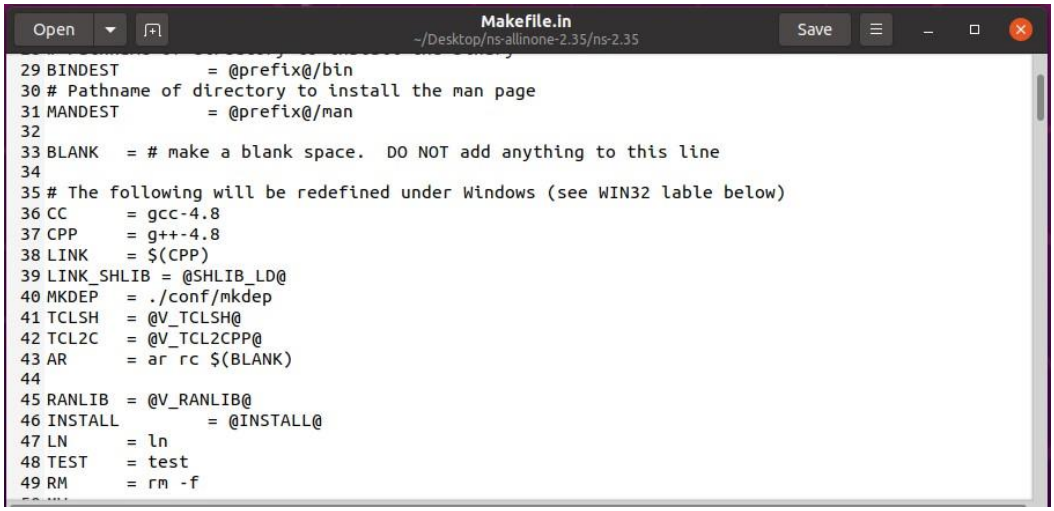


Modify the following make files.

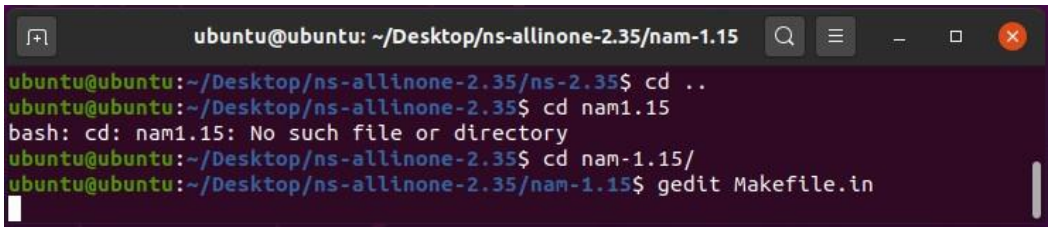
- ~ns-2.35/Makefile.in



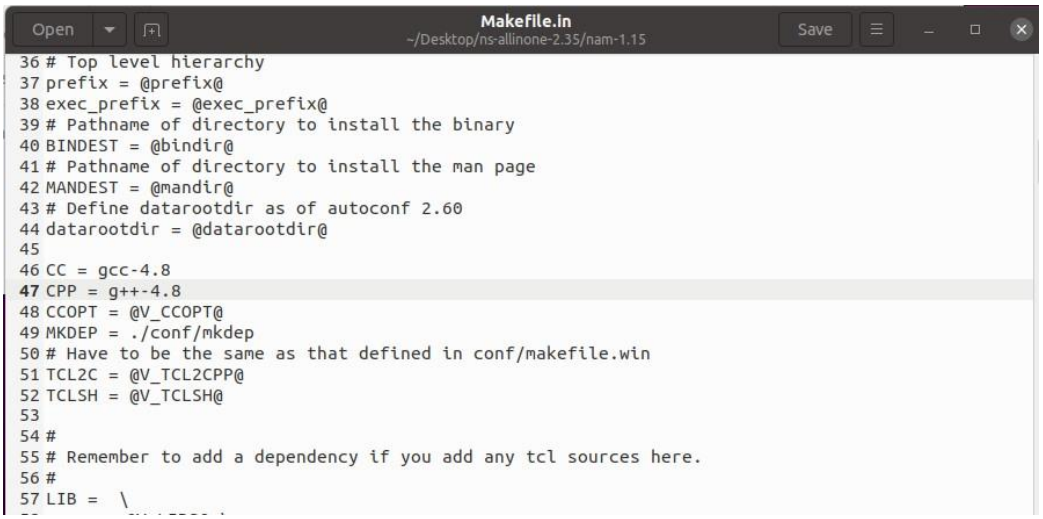
Change @CC@ to gcc-4.8
Change @CXX@ to g++-4.8



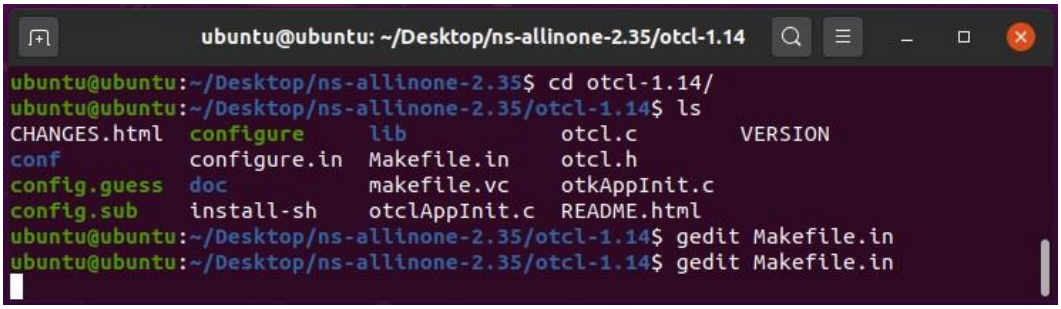
- ~nam-1.15/Makefile.in



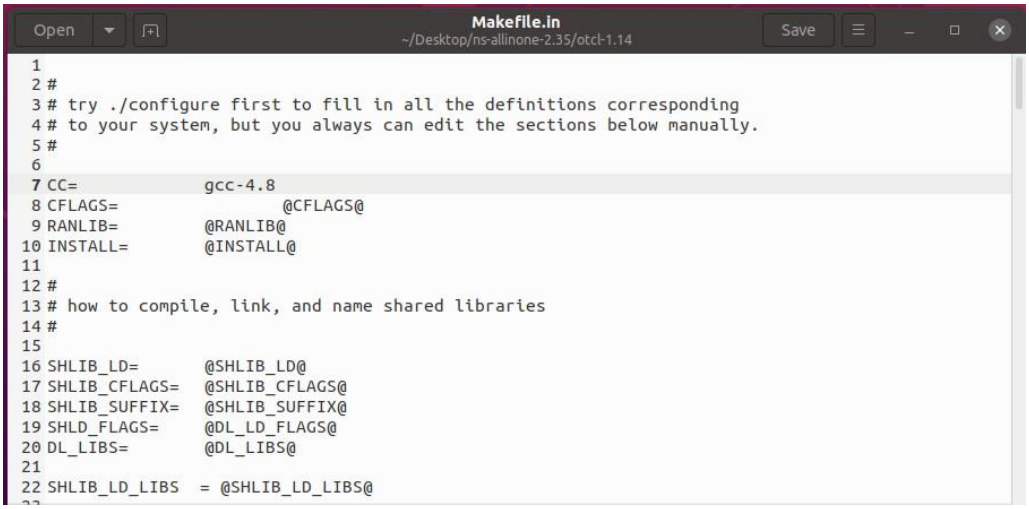
Change @CC@ to gcc-4.8
Change @CPP@ or @CXX@ to g++-4.8



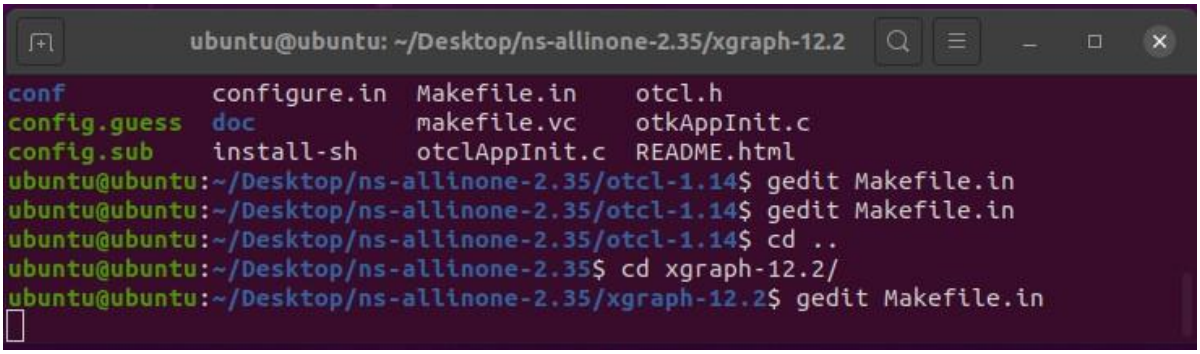
- ~otcl-1.14/Makefile.in



Change @CC@ to gcc-4.8
Change @CPP@ or @CXX@ to g++-4.8



- ~xgraph-12.2/Makefile.in



Change @CC@ to gcc-4.8
Change @CPP@ or @CXX@ to g++-4.8

Makefile.in
~/Desktop/ns-allinone-2.35/xgraph-12.2

```
106     || { find "$(distdir)" -type d ! -perm -200 -exec chmod u+w {} ';' \
107          && rm -fr "$(distdir)"; }; }
108 DIST_ARCHIVES = $(distdir).tar.gz
109 GZIP_ENV = --best
110 distuninstallcheck_listfiles = find . -type f -print
111 distcleancheck_listfiles = find . -type f -print
112 ACLOCAL = @ACLOCAL@
113 ADDITIONAL_INCLUDES = @ADDITIONAL_INCLUDES@
114 ADDITIONAL_LIBS = @ADDITIONAL_LIBS@
115 AMTAR = @AMTAR@
116 AUTOCONF = @AUTOCONF@
117 AUTOHEADER = @AUTOHEADER@
118 AUTOMAKE = @AUTOMAKE@
119 AWK = @AWK@
120 CC = gcc-4.8
121 CCDEPMODE = @CCDEPMODE@
122 CFLAGS = $(ADDITIONAL_INCLUDES)  $(C_WARNINGS) -g  $(X_CFLAGS)
123 CPP = g++-4.8
124 CPPFLAGS = @CPPFLAGS@
125 CYGPATH_W = @CYGPATH_W@
126 DEFS = @DEFS@
127 DEPDIR = @DEPDIR@
```

Plain Text Tab Width: 8 Ln 120, Col 5 INS

open the file:
~ns-2.35/linkstate/ls.h

```
ubuntu@ubuntu: ~/Desktop/ns-allinone-2.35
ubuntu@ubuntu:~/Desktop/ns-allinone-2.35/xgraph-12.2$ gedit Makefile.in
ubuntu@ubuntu:~/Desktop/ns-allinone-2.35/xgraph-12.2$ cd ..
ubuntu@ubuntu:~/Desktop/ns-allinone-2.35$ gedit ns-2.35/linkstate/ls.h
```

Change at the Line no 137
void eraseAll() { erase(baseMap::begin(), baseMap::end()); }

to This
void eraseAll() { this-> erase(baseMap::begin(), baseMap::end()); }

ls.h
~/Desktop/ns-allinone-2.35/ns-2.35/linkstate

```
128         // this next typedef of iterator seems extraneous but is required by gcc-2.96
129         typedef typename map<Key, T, less<Key> >::iterator iterator;
130         typedef pair<iterator, bool> pair_iterator_bool;
131         iterator insert(const Key & key, const T & item) {
132                 typename baseMap::value_type v(key, item);
133                 pair_iterator_bool ib = baseMap::insert(v);
134                 return ib.second ? ib.first : baseMap::end();
135         }
136
137         void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
138         T* findPtr(Key key) {
139                 iterator it = baseMap::find(key);
140                 return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
141         }
142 };
143
144 /*
145   LsNodeIdList -- A list of int 's. It manages its own memory
146 */
147 class LsNodeIdList : public LsList<int> {
148 public:
149         int appendUnique (const LsNodeIdList& x);
```

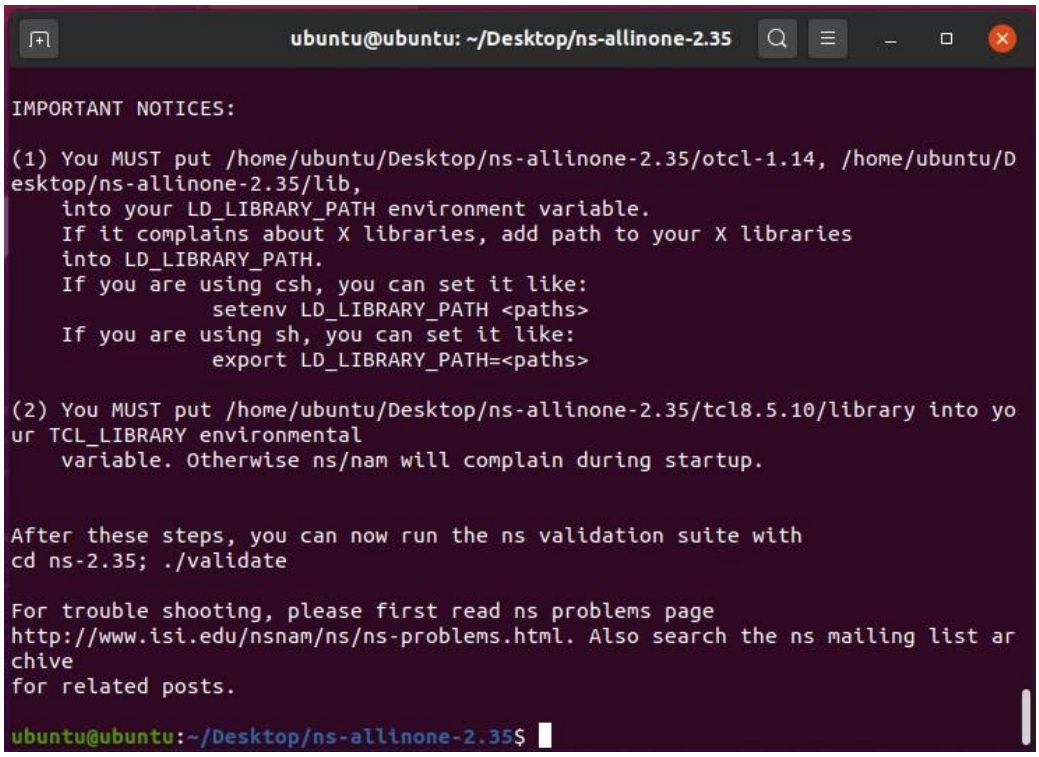
C/ObjC Header Tab Width: 8 Ln 137, Col 27 INS

4. Open a new Terminal

   Paste these lines.

   $] cd ns-allinone-2.35/
   $] ./install

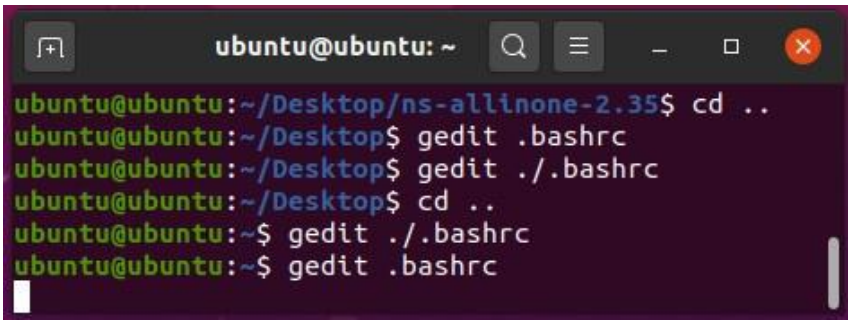
   The result would be as follows: -



5. Set the PATH

   Open a new Terminal,

   $] gedit .bashrc

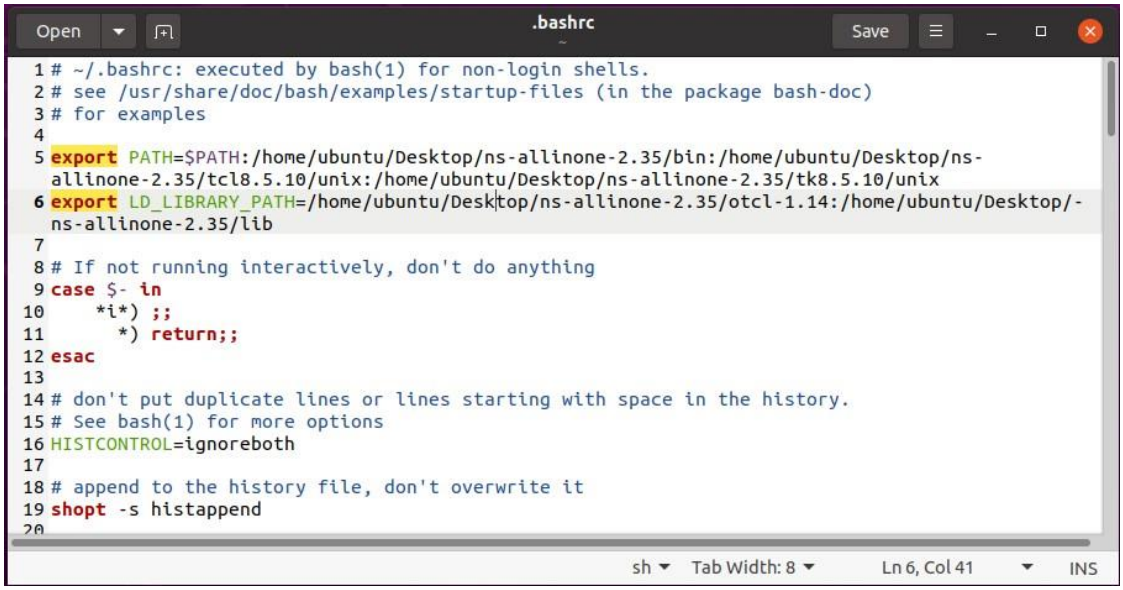


   Paste the following lines: -

- export PATH=$PATH:/home/yourusername/ns-allinone-2.35/bin:/home/yourusername/ns-allinone-2.35/tcl8.5.10/unix:/home/yourusername/ns-allinone-2.35/tk8.5.10/unix

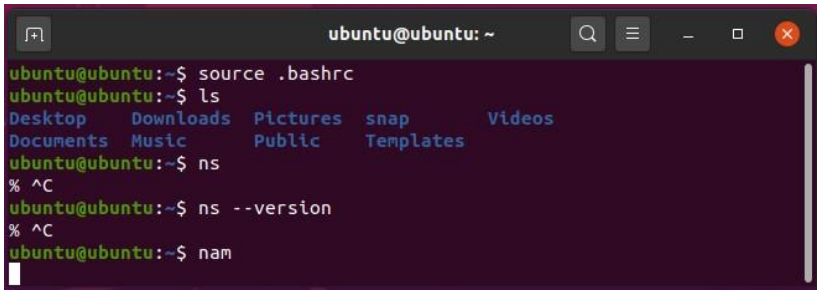
- export LD_LIBRARY_PATH=/home/yourusername/ns-allinone-2.35/otcl-

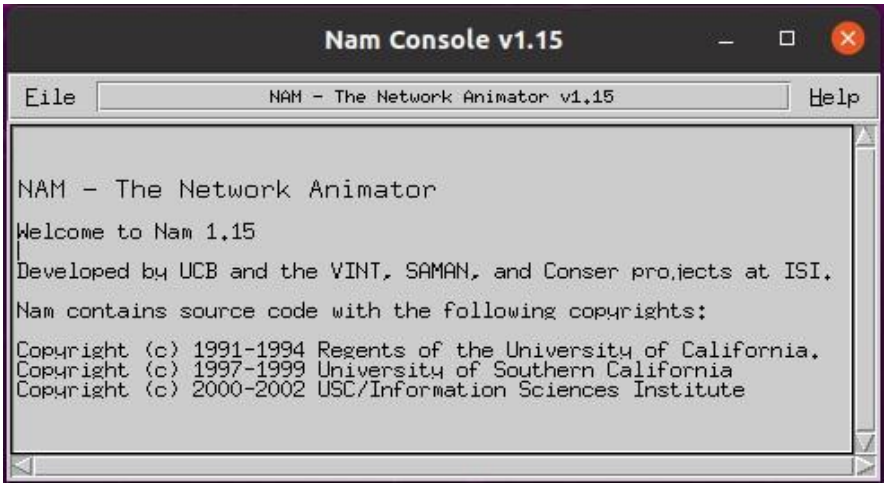
  1.14:/home/yourusername/ns-allinone-2.35/lib



Put the following line: -

$] source .bashrc





LAB ASSIGNMENT - 02

AIM/OBJECTIVE:

Implement a point – to – point network consisting of FOUR (04) nodes using the NS2 simulator with duplex links between them. Initiate a communication between these nodes. Set the queue size, vary the bandwidth, and find the number of packets dropped. Finally plot a graph showing the performance of this network in terms of the number of packets dropped with varying bandwidth.

THEORY:

1. An NS2 Simulation starts with the command: -

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

   where ns is the instance of the Simulator class
2. To have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using "open" command:

   ```
   #Open the Trace file
   set tracefile1 [open out.tr w]
   $ns trace-all $tracefile1
   #Open the NAM trace file
   set namfile [open out.nam w]
   $ns namtrace-all $namfile
   ```

   The above creates a dta trace file called "out.tr" and a nam visualization trace file called "out.nam". Within the tcl script, these files are not called explicitly by their names,but instead by pointers that are declared 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".

   The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

   The last line tells the simulator to record all simulation traces in NAM input format.It also gives the file name that the trace will be written to later by the command $ns flush-trace.In our case,this will be the file pointed at by the pointer "$namfile",i.e the file "out.tr".
3. The termination of the program is done using a "finish" procedure.

   ```
   #Define a finish procedure
   Proc finish { } {
   global ns tracefile1 namfile
   $ns flush-trace
   Close $tracefile1
   ```

```
Close $namfile
Exec nam out.nam &
Exit 0 }
```

The word proc declares a procedure in this case called finish and without arguments. The word global is used to tell that we are using variables declared outside the procedure.

The simulator method “flush-trace” will dump the traces on the respective files. The tcl command “close” closes the trace files defined before and exec executes the nam program for visualization. The command exit will ends the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails.

4. At the end of ns program we should call the procedure “finish” and specify at what time the termination should occur. For example,

   $ns at 125.0 “finish”

   will be used to call “finish” at time 125sec. Indeed, the at method of the simulator allows us to schedule events explicitly.

5. The way to define a node is

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

6. Way to define links between the nodes:

   ```
   $ns duplex-link $n0 $n2 10Mb 10ms DropTail
   ```

   Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

7. For setting up 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
   ```

8. Setup of CBR over UDP Connection

   ```
   set cbr [new Application/Traffic/CBR]
   $cbr attach-agent $udp
   $cbr set packetsize_ 100
   $cbr set rate_ 0.01Mb
   $cbr set random_ false
   ```

9. Scheduling Events

   Syntax: $ns at <time> <event>

Example: - $ns at 1.0 “$cbr stop”

CODE:

1. a.tcl

```
set ns [new Simulator]

set nf [open a.nam w]
$ns namtrace-all $nf

set tf [open a.tr w]
$ns trace-all $tf

proc finish { } {

    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam a.nam &
    exit 0
}

# defining 4 nodes
set a0 [$ns node]
set a1 [$ns node]
set a2 [$ns node]
set a3 [$ns node]

# setting up of bidirectional link between the following nodes and their positions
$ns duplex-link $a0 $a1 1.25Mb 10ms DropTail
$ns duplex-link $a1 $a2 1.25Mb 10ms DropTail
$ns duplex-link $a2 $a3 1.25Mb 10ms DropTail

#Creating orientation of the links between the nodes
$ns duplex-link-op $a0 $a1 orient right-down
$ns duplex-link-op $a1 $a2 orient right-up
$ns duplex-link-op $a2 $a3 orient right


$ns queue-limit $a0 $a1 5
$ns queue-limit $a1 $a2 5
$ns queue-limit $a2 $a3 5


# setting up TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $a0 $tcp0
set ftp0 [new Application/FTP]
# setting up of traffic over TCP connection
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
```

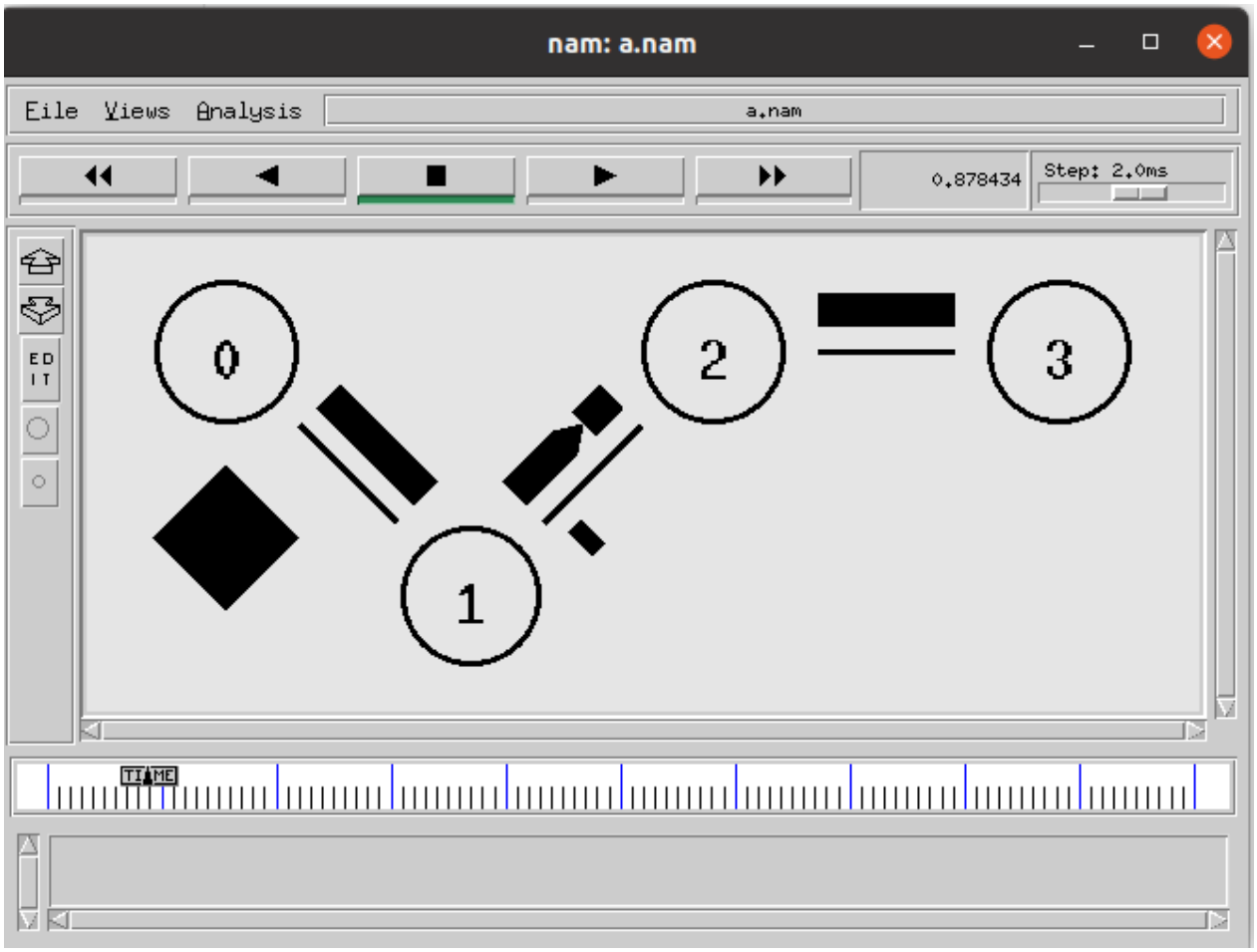
```
# setting up TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $a1 $tcp1
#set ftp1 [new Application/FTP]
# setting up of traffic over TCP connection
#$ftp1 attach-agent $tcp1
#$ftp1 set packetSize_ 500

# Setting up TCP Connection
set tcp2 [new Agent/TCP]
$ns attach-agent $a2 $tcp2

#set ftp2 [new Application/FTP]
# setting up of traffic over TCP connection
#$ftp2 attach-agent $tcp2

set null0 [new Agent/TCPSink]
$ns attach-agent $a3 $null0
$ns connect $tcp0 $null0

#$ns connect $tcp1 $null0

$ns at 0.2 "$ftp0 start"
$ns at 10 "finish"

$ns run
```

2. a.awk

```
BEGIN {
    dropped=0;
    received=0;
}

{
     event=$1;

     if(event == "d"){
        dropped++;
     }

     if(event == "r"){
        received++;
     }
}

END {
        printf("The no.of packets dropped : %d\n ",dropped);
        printf("The no.of packets recieved : %d\n ",received);
}
```

OUTPUT:

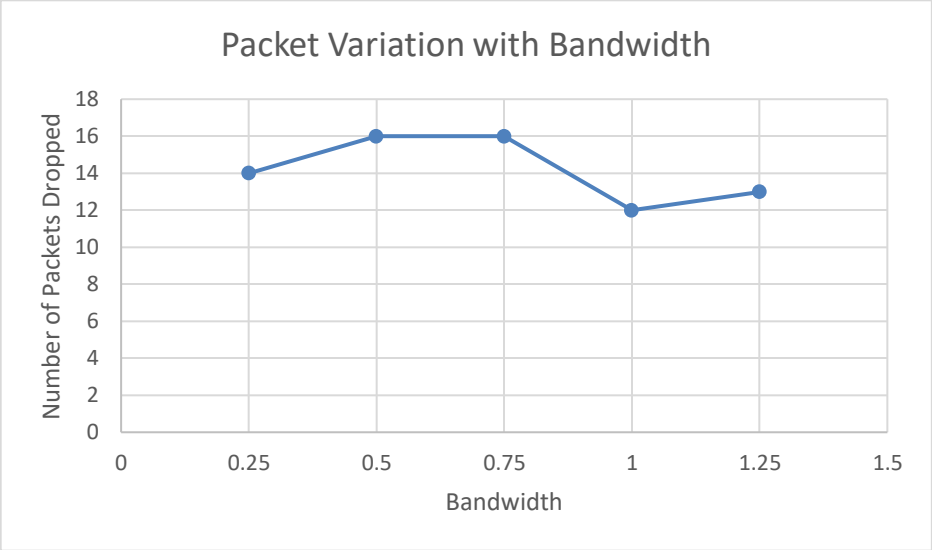
1. NAM Simulator




```
ubuntu@ubuntu: ~/Desktop/MC_Lab/Lab_02
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ ns a.tcl
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ awk -f a.awk a.tr
The no.of packets dropped : 14
 The no.of packets recieved : 1464
 ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ ns a.tcl
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ awk -f a.awk a.tr
The no.of packets dropped : 16
 The no.of packets recieved : 2802
 ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ ns a.tcl
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ awk -f a.awk a.tr
The no.of packets dropped : 16
 The no.of packets recieved : 4067
 ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ ns a.tcl
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ awk -f a.awk a.tr
The no.of packets dropped : 12
 The no.of packets recieved : 5412
 ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ ns a.tcl
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$ awk -f a.awk a.tr
The no.of packets dropped : 13
 The no.of packets recieved : 5736
 ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_02$
```

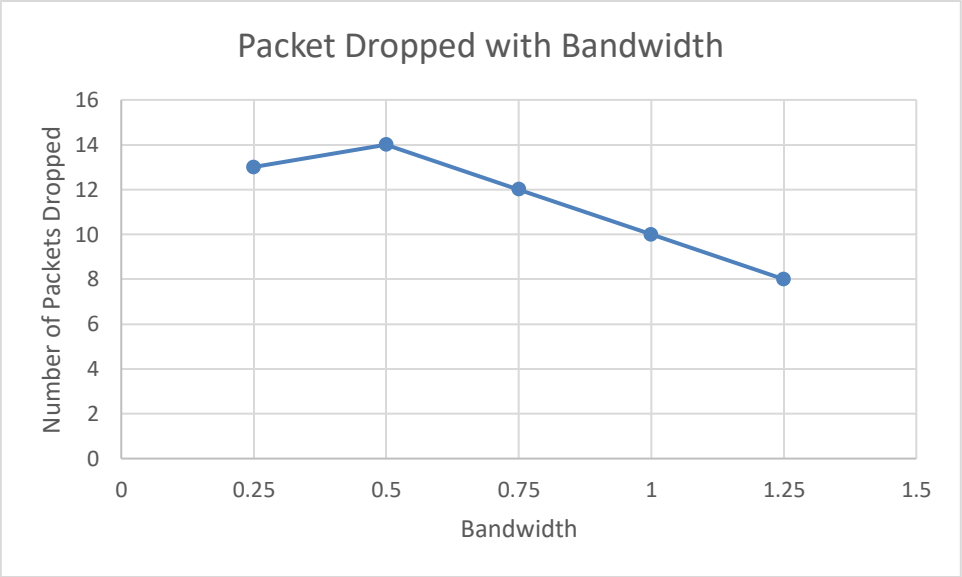
2. Graph
    a. FTP over TCP Connection
        i. For Queue Size = 5

Bandwidth	No of Packets dropped
0.25	14
0.5	16
0.75	16
1	12
1.25	13



        ii. For Queue Size = 7

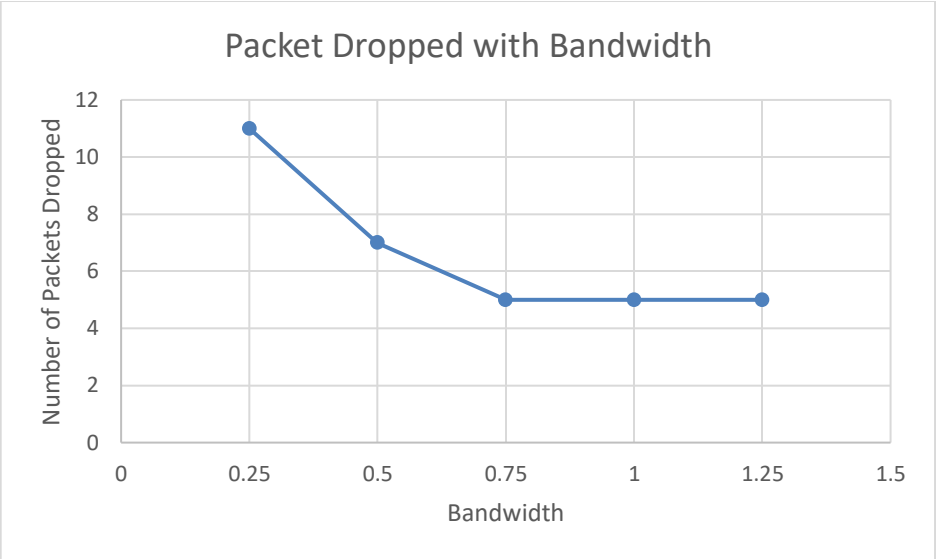
Bandwidth	No of Packets dropped
0.25	13
0.5	14
0.75	12
1	10
1.25	8



b. CBR over TCP Connection

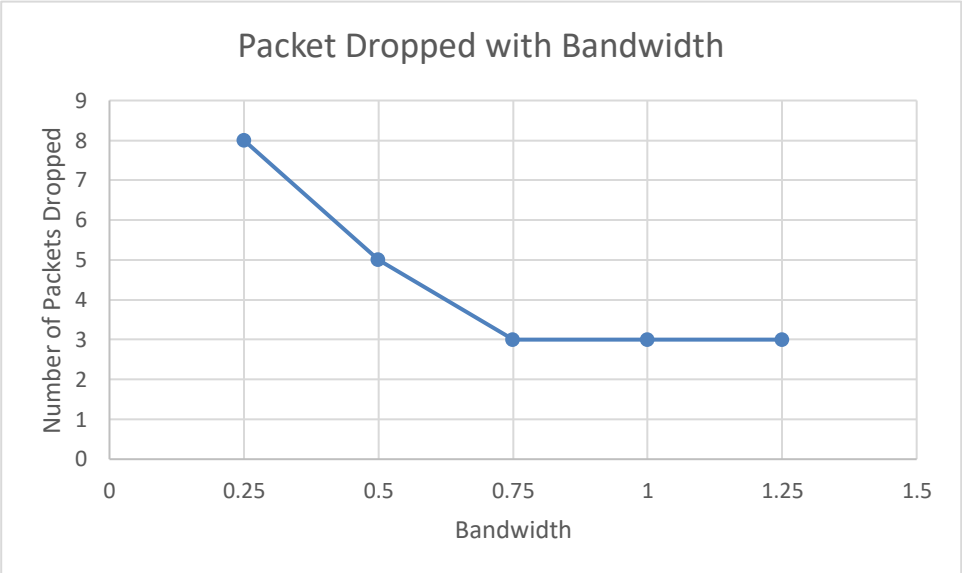
i. For Queue Size = 5

Bandwidth	No of Packets dropped
0.25	11
0.5	7
0.75	5
1	5
1.25	5



ii. For Queue Size = 7

Bandwidth	No of Packets dropped
0.25	8
0.5	5
0.75	3
1	3
1.25	3



LAB ASSIGNMENT - 03

AIM/OBJECTIVE:

Implement an Ethernet LAN using 'N' nodes and set multiple traffic nodes and plot congestion windows for different source/destination pairs in NS2/NS3.

THEORY:

- **Ethernet**

  Ethernet is a wired LAN. It is a protocol present in Data link Layer

  The data link layer consists of two sub layers:

  1. Media Access Control (MAC)
  2. Logical Link Control (LLC)

- **Congestion**

  Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle.

- **CWND and RWND**

  - Congestion Window (CWND) : cwnd is a TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK
  - Receiver Window (RWND) : rwnd is a variable that advertises the amount of data that the destination side can receive.

  Together, the two variables are used to regulate data flow TCP connections, minimize congestion, and improve network performance.

- **Slow Start and Congestion Avoidance in TCP**

  In slow start congestion control, TCP increases the window's size rapidly to reach the maximum transfer rate as fast as possible. This self-imposed window size increases as TCP confirms the network's ability to transmit the data without errors. However, this can only go up to a maximum advertised window (RWND).

  In this scenario, the sender uses two variables:

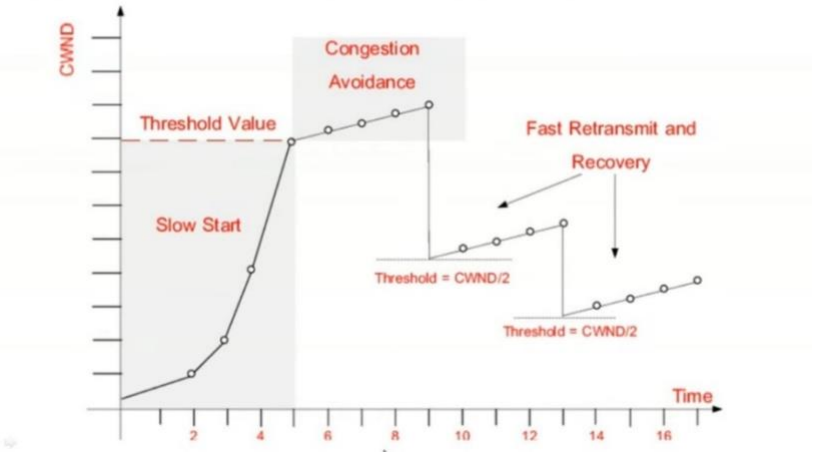
  - Congestion window with an initial value of one maximum segment size (MSS)
  - The slow start threshold value (ssthresh) with an initial value equal to the receiver window.

  The congestion window increases exponentially during congestion control, and lineally during the congestion avoidance.

  Sslow start occurs when cwnd < ssthresh and congestion avoidance occurs when cwnd >= ssthresh

  Whenever a packet has been lost or we received 3 duplicate ACK, then the congestion has dropped to zero.

Slow Start and Congestion Avoidance in TCP



CODE:

1. prog1.tcl

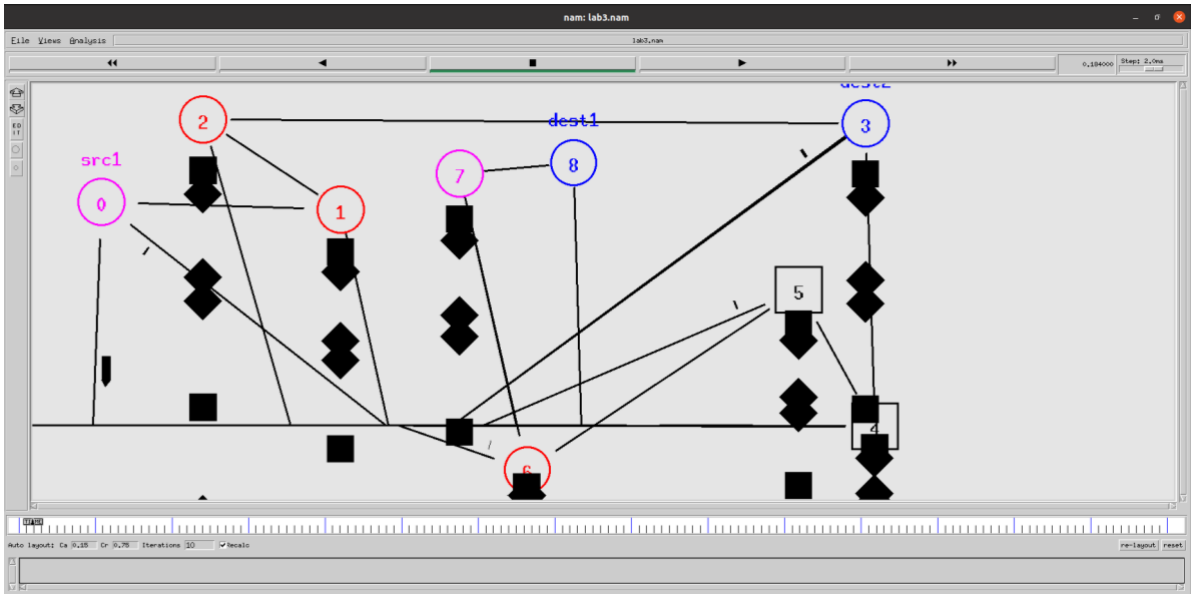
```
# Make a NS simulator
LanRouter set debug_ 0
set ns [new Simulator]
set tf [open lab3.tr w]

$ns trace-all $tf
set nf [open lab3.nam w]
$ns namtrace-all $nf

# Create the nodes, color, and label
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"

set n1 [$ns node]
$n1 color "red"

set n2 [$ns node]
$n2 color "red"
#$n2 label "src2"

set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"

set n4 [$ns node]
$n4 shape square

set n5 [$ns node]
$n5 shape square

set n6 [$ns node]
$n6 color "red"

set n7 [$ns node]
$n7 color "magenta"
$n2 label "src2"

set n8 [$ns node]
$n8 color "blue"
$n8 label "dest1"


$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 $n6 1Mb 10ms DropTail
```

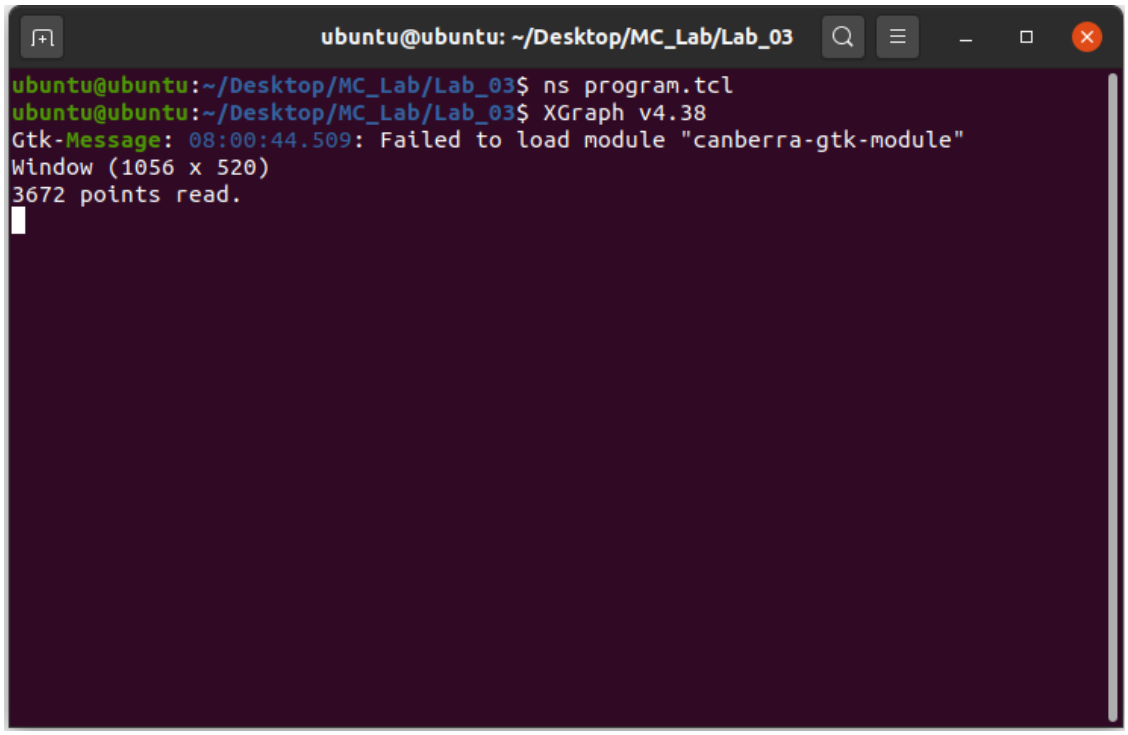
```
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail


# Create LAN 1 with make-lan
set lan1 [$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8" 10Mb 10ms LL Queue/DropTail]
Mac/802_3 change

# Create LAN 2 with make-lan
#set lan2 [$ns make-lan "$n5 $n6 $n7 $n8" 10Mb 10ms LL Queue/DropTail]
#Mac/802_3 change

# Connect LAN 1 to LAN 2 with a duplex link
#set link_($n4:$n5) [$ns duplex-link $n4 $n5 5Mb 10ms DropTail]
#$ns duplex-link-op $n4 $n5 orient right-up

# Add a TCP sending module to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

# Setup a FTP traffic generator on "tcp0"
$tcp0 set window_ 10
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001

# Add a TCP receiving module to node n5
set sink0 [new Agent/TCPSink]
$ns attach-agent $n8 $sink0

# Direct traffic from "tcp0" to "sink0"
$ns connect $tcp0 $sink0

# Add a TCP sending module to node n2
set tcp1 [new Agent/TCP]
$ns attach-agent $n7 $tcp1
$tcp1 set window_ 10
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.001

# Add a TCP receiving module to node n3
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1

set file1 [open file14.tr w]
$tcp0 attach $file1
set file2 [open file24.tr w]
```

```
$tcp1 attach $file2
$tcp0 trace cwnd_
$tcp1 trace cwnd_

$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient down
$ns duplex-link-op $n5 $n6 orient left
$ns duplex-link-op $n6 $n7 orient left
$ns duplex-link-op $n7 $n8 orient left

# Define a 'finish' procedure
proc finish { } {
   global ns nf tf
   $ns flush-trace
   close $tf
   close $nf
   exec nam lab3.nam &
   # Generate xgraph for file14.tr with red color
        #exec xgraph -color red file14.tr &
        exec awk -f exp4.awk file14.tr > a1 &

        # Generate xgraph for file24.tr with blue color
        exec awk -f exp4.awk file24.tr > a2 &
        #exec xgraph -color blue file24.tr &

        exec xgraph a1 a2 &
   exit 0
}

# Schedule start/stop times
$ns at 0.1 "$ftp0 start"
$ns at 2 "$ftp0 stop"
$ns at 3 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 9 "$ftp0 start"
$ns at 10 "$ftp1 start"
$ns at 14 "$ftp0 stop"
$ns at 15 "$ftp1 stop"

# Set simulation end time
$ns at 16 "finish"
$ns run
```

OUTPUT:

1. NAM Simulator

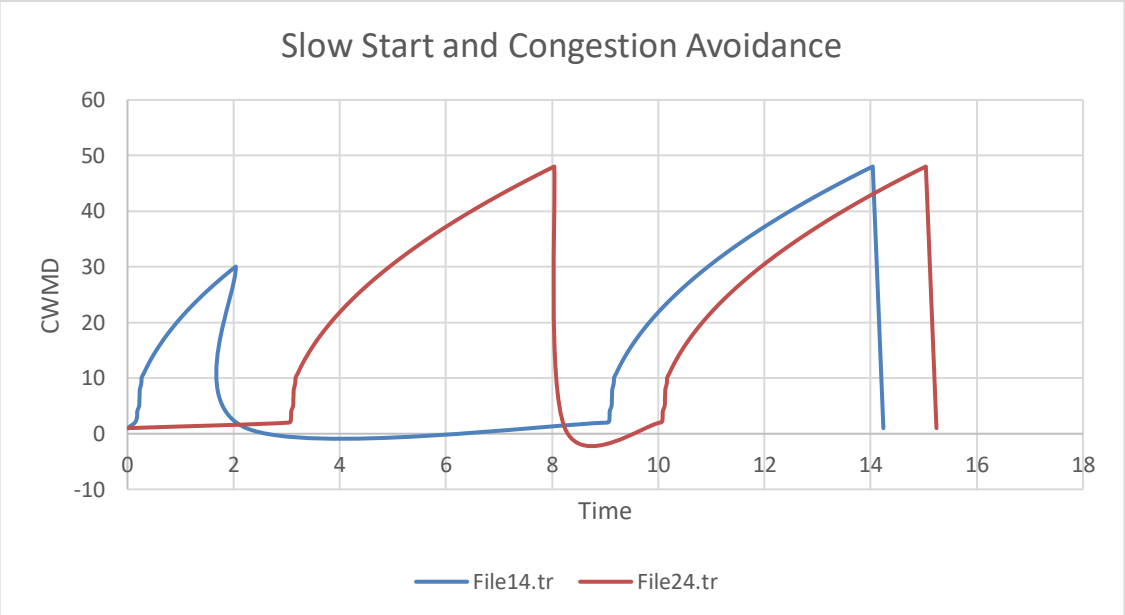




```
ubuntu@ubuntu: ~/Desktop/MC_Lab/Lab_03
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_03$ awk -f exp4.awk file14.tr
0.000000        1.000000
0.140010        2.000000
0.180020        3.000000
0.184430        4.000000
0.220020        5.000000
0.224440        6.000000
0.228860        7.000000
0.233270        8.000000
0.260030        9.000000
0.264450        10.000000
0.268860        10.100000
0.273280        10.199000
0.277700        10.297000
0.282110        10.394000
0.286530        10.490000
0.290940        10.586000
0.300040        10.680000
```

```
ubuntu@ubuntu: ~/Desktop/MC_Lab/Lab_03
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_03$ awk -f exp4.awk file24.tr
0.000000        1.000000
3.040010        2.000000
3.080020        3.000000
3.084430        4.000000
3.120020        5.000000
3.124440        6.000000
3.128860        7.000000
3.133270        8.000000
3.160030        9.000000
3.164450        10.000000
3.168860        10.100000
3.173280        10.199000
3.177700        10.297000
3.182110        10.394000
```

2. Graph



LAB ASSIGNMENT - 04

AIM/OBJECTIVE:

Implement an Ethernet LAN comprising of 'N' nodes. Set multiple traffic nodes and plot the performance for varying congestion windows for different source/destination pairs in NS2/NS3.

THEORY:

- **Ethernet**

Ethernet is a wired LAN. It is a protocol present in Data link Layer

The data link layer consists of two sub layers:

3. Media Access Control (MAC)
4. Logical Link Control (LLC)

- **Congestion**

Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle.

- **CWND and RWND**
  - Congestion Window (CWND) : cwnd is a TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK
  - Receiver Window (RWND) : rwnd is a variable that advertises the amount of data that the destination side can receive.

Together, the two variables are used to regulate data flow TCP connections, minimize congestion, and improve network performance.

- **Slow Start and Congestion Avoidance in TCP**

In slow start congestion control, TCP increases the window's size rapidly to reach the maximum transfer rate as fast as possible. This self-imposed window size increases as TCP confirms the network's ability to transmit the data without errors. However, this can only go up to a maximum advertised window (RWND).

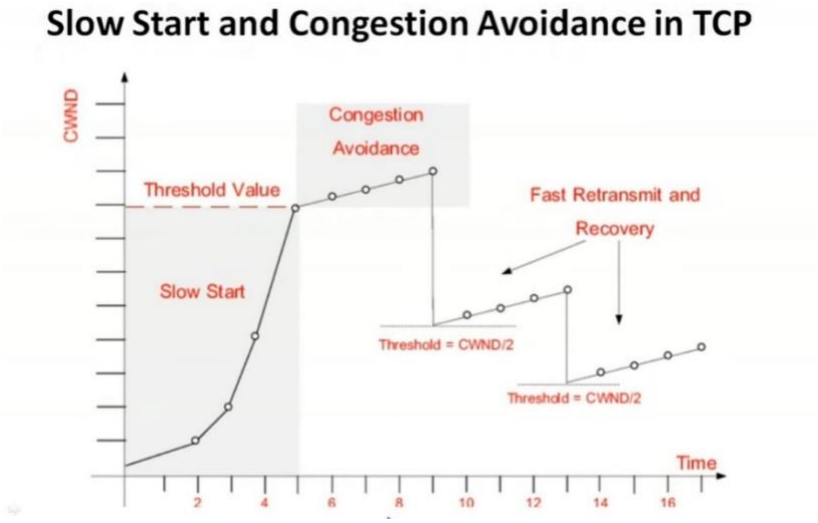
In this scenario, the sender uses two variables:

- Congestion window with an initial value of one maximum segment size (MSS)
- The slow start threshold value (ssthresh) with an initial value equal to the receiver window.

The congestion window increases exponentially during congestion control, and lineally during the congestion avoidance.

Sslow start occurs when cwnd < ssthresh and congestion avoidance occurs when cwnd >= ssthresh

Whenever a packet has been lost or we received 3 duplicate ACK, then the congestion has dropped to zero.



CODE:

1. exp4.tcl

```
# Make a NS simulator
LanRouter set debug_ 0
set ns [new Simulator]
set tf [open lab4.tr w]

$ns trace-all $tf
set nf [open lab4.nam w]
$ns namtrace-all $nf

# Create the nodes, color, and label
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"

set n1 [$ns node]
$n1 color "red"

set n2 [$ns node]
$n2 color "red"
#$n2 label "src2"

set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"

set n4 [$ns node]
$n4 shape square

set n5 [$ns node]
$n5 shape square

set n6 [$ns node]
$n6 color "red"
```

```
set n7 [$ns node]
$n7 color "magenta"
$n2 label "src2"

set n8 [$ns node]
$n8 color "blue"
$n8 label "dest1"

# Create LAN 1 with make-lan
set lan1 [$ns make-lan "$n0 $n1 $n2 $n3 $n4" 10Mb 10ms LL Queue/DropTail]
Mac/802_3 change

# Create LAN 2 with make-lan
set lan2 [$ns make-lan "$n5 $n6 $n7 $n8" 10Mb 10ms LL Queue/DropTail]
Mac/802_3 change

# Connect LAN 1 to LAN 2 with a duplex link
set link_($n4:$n5) [$ns duplex-link $n4 $n5 5Mb 10ms DropTail]
$ns duplex-link-op $n4 $n5 orient right-up

# Add a TCP sending module to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
# Setup a FTP traffic generator on "tcp0"
$tcp0 set window_ 10
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001

# Add a TCP receiving module to node n5
set sink0 [new Agent/TCPSink]
$ns attach-agent $n8 $sink0

# Direct traffic from "tcp0" to "sink0"
$ns connect $tcp0 $sink0

# Add a TCP sending module to node n2
set tcp1 [new Agent/TCP]
$ns attach-agent $n7 $tcp1
$tcp1 set window_ 10
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 500
$ftp1 set interval_ 0.001

# Add a TCP receiving module to node n3
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1
```

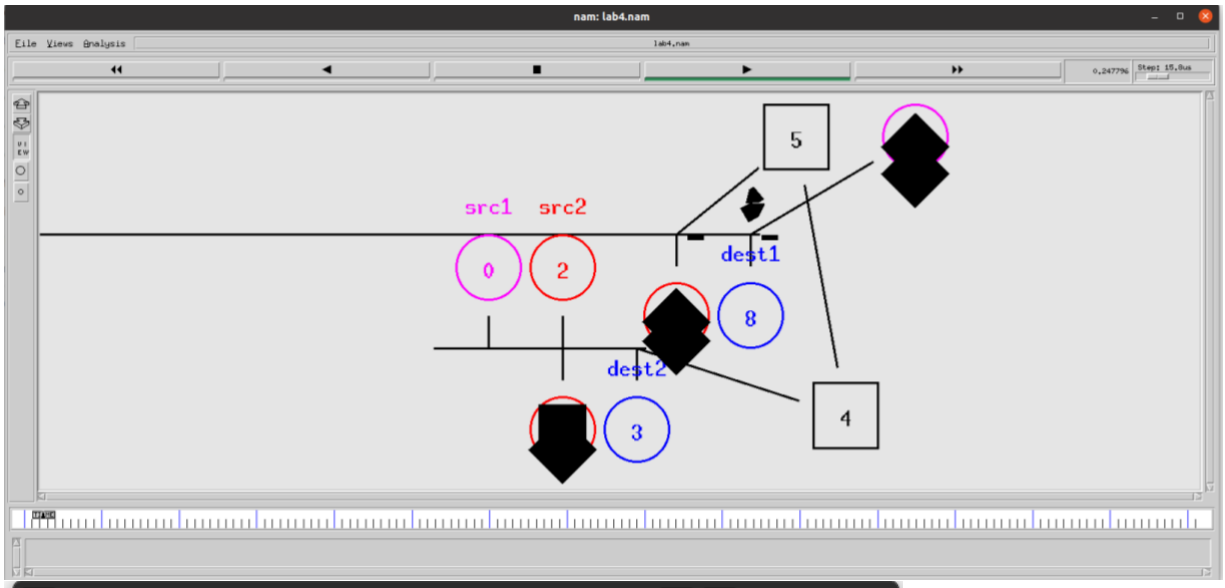
```
set file1 [open file14.tr w]
$tcp0 attach $file1
set file2 [open file24.tr w]
$tcp1 attach $file2
$tcp0 trace cwnd_
$tcp1 trace cwnd_

# Define a 'finish' procedure
proc finish { } {
   global ns nf tf
   $ns flush-trace
   close $tf
   close $nf
   exec nam lab4.nam &
   # Generate xgraph for file14.tr with red color
    #exec xgraph -color red file14.tr &
    exec awk -f exp4.awk file14.tr > a1 &

    # Generate xgraph for file24.tr with blue color
    exec awk -f exp4.awk file24.tr > a2 &
    #exec xgraph -color blue file24.tr &

    exec xgraph a1 a2 &
   exit 0
}

# Schedule start/stop times
$ns at 0.1 "$ftp0 start"
$ns at 2 "$ftp0 stop"
$ns at 3 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 9 "$ftp0 start"
$ns at 10 "$ftp1 start"
$ns at 14 "$ftp0 stop"
$ns at 15 "$ftp1 stop"

# Set simulation end time
$ns at 16 "finish"
$ns run
```

2. exp4.awk

```
BEGIN {
}
{
if($6=="cwnd_")
printf("%f\t%f\t\n",$1,$7);
}
END {
}
```

OUTPUT:

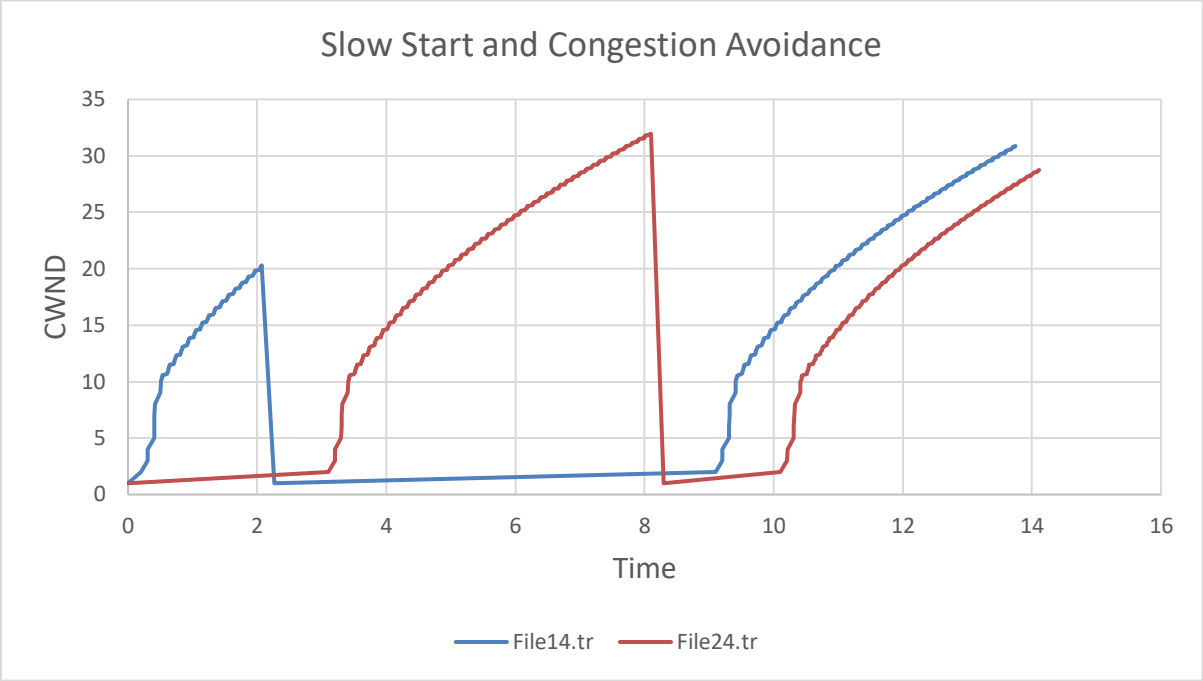
1. NAM Simulator




```
ubuntu@ubuntu: ~/Desktop/MC_Lab/Lab_04
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_04$ awk -f exp4.awk file14.tr
0.000000        1.000000
0.200140        2.000000
0.301890        3.000000
0.306300        4.000000
0.403630        5.000000
0.408050        6.000000
0.412460        7.000000
0.416880        8.000000
0.505380        9.000000
0.509790        10.000000
0.514210        10.100000
0.518620        10.199000
0.523040        10.297000
0.527460        10.394000
0.531870        10.490000
0.536290        10.586000
0.607120        10.680000
0.611540        10.774000
0.615950        10.867000
0.620370        10.959000
0.624780        11.050000
0.629200        11.140000
0.633620        11.230000
```

```
ubuntu@ubuntu: ~/Desktop/MC_Lab/Lab_04
14.300450       1.000000
ubuntu@ubuntu:~/Desktop/MC_Lab/Lab_04$ awk -f exp4.awk file24.tr
0.000000        1.000000
3.100140        2.000000
3.201890        3.000000
3.206300        4.000000
3.303630        5.000000
3.308050        6.000000
3.312460        7.000000
3.316880        8.000000
3.405380        9.000000
3.409790        10.000000
3.414210        10.100000
3.418620        10.199000
3.423040        10.297000
3.427460        10.394000
3.431870        10.490000
3.436290        10.586000
3.507120        10.680000
3.511540        10.774000
3.515950        10.867000
3.520370        10.959000
3.524780        11.050000
3.529200        11.140000
3.533620        11.230000
3.538030        11.319000
```

2. Graph


Slow Start and Congestion Avoidance
CWND
Time
0
5
10
15
20
25
30
35
0
2
4
6
8
10
12
14
16
File14.tr
File24.tr

LAB ASSIGNMENT - 05

AIM/OBJECTIVE:

Implement and study the performance of GSM on NS2/NS3 using MAC layer

THEORY:

- **GSM**

  GSM stands for Global System for Mobile Communication. GSM is an open and digital cellular technology used for mobile communication. It uses 4 different frequency bands of 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz. It uses the combination of FDMA and TDMA.

- **Features of GSM**
  - Supports international roaming
  - Clear voice clarity
  - Ability to support multiple handheld devices.
  - Spectral / frequency efficiency
  - Low powered handheld devices.
  - Ease of accessing network
  - International ISDN compatibility.
  - Low service cost.
  - New features and services.

CODE:

1. program.tcl

```
#==================================
# Simulation parameters setup
#==================================
set val(stop)   50.0 ; # time of simulation end
# General Parameters
set opt(title) zero;
set opt(stop) 50; # Stoptime
set opt(ecn) 0;

# Topology
set opt(type) gsm; # type of link;
set opt(secondDelay) 55; # average delay of access link in ms

# AQM parameters
set opt(minth) 30;
set opt(maxth) 0;
set opt(adaptive) 1; # ! for Adaptive RED, 0 for plain RED

# Traffic Gneration
set opt(flows) 0; # number of long-lived TCP flows
```

```
set opt(window) 30; # window for long-lived traffic
set opt(web) 2; # number of web sessions

# default downlink brandwidth in bps
set bwDL(gsm) 9600

# default uplink bandwidth in bps
set bwUL(gsm) 9600

# default downlink propogation delay in seconds
set propDL(gsm) .500

# default uplink propogation delay in second
set propUL(gsm) .500

# default buffer size in packets
set buf(gsm) 10
# end
#====================================

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

#Open the NS trace file
set tracefile [open s5.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open s5.nam w]
$ns namtrace-all $namfile

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

#====================================
# Links Definition
#====================================
#Create links between nodes

proc cell_topo {} {
    global ns nodes
    $ns duplex-link $nodes(n0) $nodes(n1) 3.0Mb 10ms DropTail
    $ns duplex-link $nodes(n1) $nodes(n2) 1.0Mb 10ms RED
    $ns duplex-link $nodes(n2) $nodes(n3) 1.0Mb 10ms RED
```

```
    $ns duplex-link $nodes(n3) $nodes(n4) 3.0Mb 10ms DropTail
    puts "Cell Toplogy"
}

proc set_link_params {t} {
    global ns nodes bwUL bwDL propUL propDL buf
    $ns bandwidth $nodes(n0) $nodes(n1) $bwDL($t) duplex
    $ns bandwidth $nodes(n1) $nodes(n2) $bwUL($t) duplex
    $ns bandwidth $nodes(n2) $nodes(n3) $bwDL($t) duplex
    $ns bandwidth $nodes(n3) $nodes(n4) $bwUL($t) duplex
    $ns delay $nodes(n0) $nodes(n1) $propDL($t) duplex
    $ns delay $nodes(n1) $nodes(n2) $propDL($t) duplex
    $ns delay $nodes(n2) $nodes(n3) $propDL($t) duplex
    $ns delay $nodes(n3) $nodes(n4) $propDL($t) duplex
    $ns queue-limit $nodes(n0) $nodes(n1) $buf($t)
    $ns queue-limit $nodes(n1) $nodes(n2) $buf($t)
    $ns queue-limit $nodes(n2) $nodes(n3) $buf($t)
    $ns queue-limit $nodes(n3) $nodes(n4) $buf($t)
}

switch $opt(type) {
    gsm -
    gprs -
    umts {cell_topo}
}

set_link_params $opt(type)

#=================================
# Agents Definition
#=================================

#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $nodes(n0) $tcp0
set sink2 [new Agent/TCPSink/Sack1]
$ns attach-agent $nodes(n4) $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500

#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $nodes(n0) $tcp1
set sink3 [new Agent/TCPSink/Sack1]
$ns attach-agent $nodes(n4) $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500

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

#Setup a FTP Application over TCP connection
```

```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"
$ns at 35.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 45.0 "$ftp1 stop"

#===================================
# Termination
#===================================

# Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    #set GETRC "../../../bin/getrc"
    #set RAW2XG "../../../bin/raw2xg"
    #exec $GETRC -s $sid -d $did -f 0 out.tr | \$RAW2XG -s 0.01 -m $wrap -r > plot.xgr
    #exec $GETRC -s $did -d $sid -f 0 out.tr | \ $RAW2XG -a-s 0.01 -m $wrap >> plot.xgr
    #exec xgraph -x time -y packets plot.xgr
    exec nam s5.nam &
    exec awk -f lab5.awk s5.tr > a1 &
    exec xgraph a1 &
    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
```

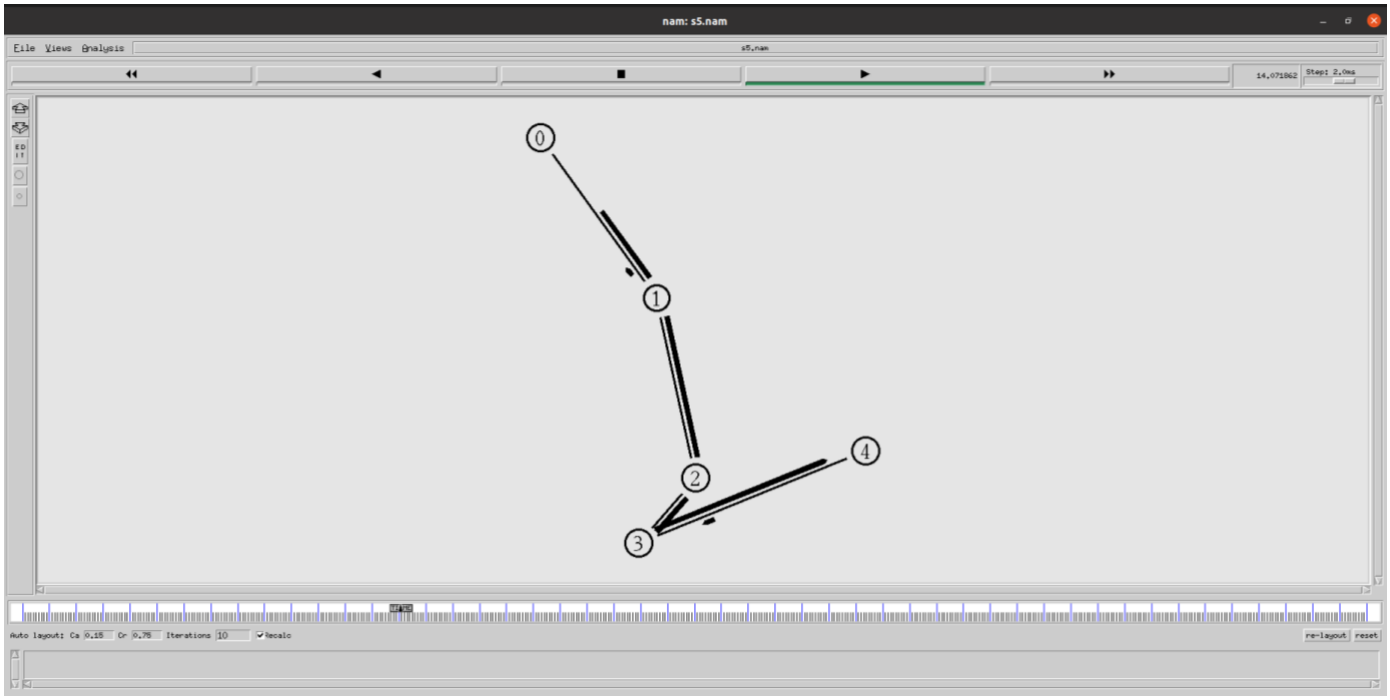
2. lab5.awk

```
BEGIN{
    count = 0;
    pack = 0;
    time = 0;
}

{
    if($1 == "r" && $5 == "tcp") {
            count++;
            pack = $6;
            time = $2
            printf("%f %f\n", time, ((count * pack * 8)/(time*1000000)));
```

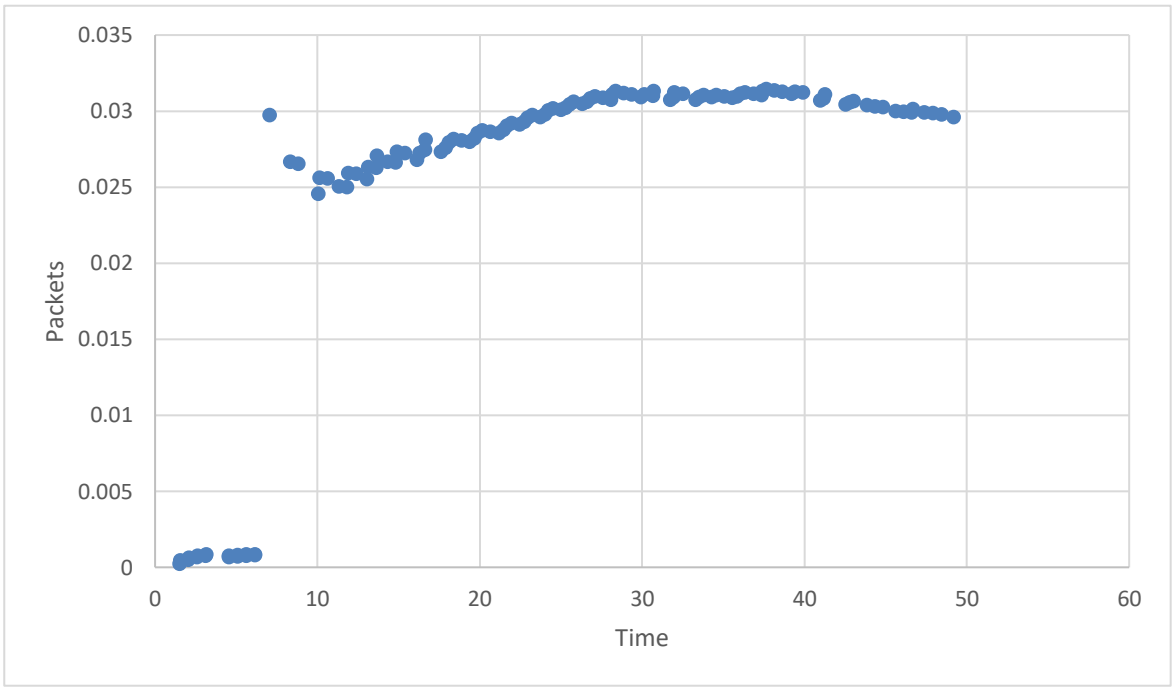
```
        }
    }

    END {
    }
```

OUTPUT:

1. NAM Simulator



2. Graph



LAB ASSIGNMENT - 06

AIM/OBJECTIVE:

Implement an Infrastructure-less wireless network comprising of 'N' nodes in an area of 400m x 300m and set multiple traffic nodes. After simulation plot the performance in terms of the End-to-End delay, Throughput, packet delivery ratio for varying for No. of nodes with different source/destination pair in NS2/NS3.

THEORY:

- **End-to-End Delay**

  It is the total latency experienced by a packet to traverse the network from the source to the destination. The end-to-end delay of a path is the summation of the node delay at each node plus the link delay at each link on the path.

- **Throughput**

  It is the measure of the average number of bits transmitted per second.

- **Packet Delivery Ratio**

  It is the measure of the average number of bits transmitted per second.

CODE:

1. program.tcl

```
# Defining Node Configuration paramaters
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(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
set val(nn) 8 ;# number of mobilenodes
set val(rp) DSDV ;# routing protocol
set val(x) 400 ;# X dimension of the topography
set val(y) 300 ;# Y dimensionof the topography

# Set the Mac Parameters, for more parameters, refer the ~ns-2.35/lib/ns-default.tcl Mac/802_11
set RTSThreshold_ 3000Mac/802_11
set basicRate_ 1MbMac/802_11
set dataRate_ 2Mb

# creation of tracefiles for variousmetrics
# *** Throughput Trace ***
set f0 [open thru02.tr w]
set f1 [open thru12.tr w]
set f2 [open thru22.tr w]
set f3 [open thru32.tr w]
```

```
#   ***  Packet  LossTrace ***
set f4 [open pktloss02.tr w]
set f5 [open pktloss12.tr w]
set f6 [open pktloss22.tr w]
set f7 [open pktloss32.tr w]

# *** Packet Delay Trace ***
set f8 [open pktdelay02.tr w]
set f9 [open pktdelay12.tr w]
set f10 [open pktdelay22.tr w]
set f11 [open pktdelay32.tr w]

# Simulator Object
set ns [new Simulator]

# Trace file initialization
set tracef [open wireless3.tr w]
$ns trace-all $tracef

# Network Animator
set namf [open wireless3.nam w]
$ns namtrace-all-wireless $namf $val(x) $val(y)

# Topography
set topo [new Topography]
$topo load_flatgrid 500 500

#creation of god (General Operations Director) object
create-god $val(nn)

# configure nodes
$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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace OFF

# Create Nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) random-motion 0 ; # disable random motion
}
```

```
#initial position of nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 10.0
$node_(1) set Y_ 15.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 35.0
$node_(2) set Y_ 250.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 10.0
$node_(3) set Y_ 50.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 235.0
$node_(4) set Y_ 10.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 400.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 285.0
$node_(6) set Y_ 150.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 120.0
$node_(7) set Y_ 115.0
$node_(7) set Z_ 0.0

# Create traffic flow using UDP with Constant Bit Rate Application
# this includes priority and the sink is LossMonitor agent to trace the bytes received (because the
    Null Agent does not handle this)
set agent1 [new Agent/UDP]
$agent1 set prio_ 0

set sink [new Agent/LossMonitor]
$ns attach-agent $node_(0) $agent1
$ns attach-agent $node_(1) $sink
$ns connect $agent1 $sink

set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ 512 ; # setting the packet size
$app1 set rate_ 600Kb ; # setting the rate at which the packets are transmitted
$app1 attach-agent $agent1 ; # attaching the agent

set agent2 [new Agent/UDP]
$agent2 set prio_ 1
set sink2 [new Agent/LossMonitor]
$ns attach-agent $node_(2) $agent2
```

```
$ns attach-agent $node_(3) $sink2
$ns connect $agent2 $sink2

set app2 [new Application/Traffic/CBR]
$app2 set packetSize_ 512
$app2 set rate_ 600Kb
$app2 attach-agent $agent2

set agent3 [new Agent/UDP]
$agent3 set prio_ 2
set sink3 [new Agent/LossMonitor]
$ns attach-agent $node_(4) $agent3
$ns attach-agent $node_(5) $sink3
$ns connect $agent3 $sink3

set app3 [new Application/Traffic/CBR]
$app3 set packetSize_ 512
$app3 set rate_ 600Kb
$app3 attach-agent $agent3

set agent4 [new Agent/UDP]
$agent4 set prio_ 3
set sink4 [new Agent/LossMonitor]
$ns attach-agent $node_(6) $agent4
$ns attach-agent $node_(7) $sink4
$ns connect $agent4 $sink4

set app4 [new Application/Traffic/CBR]
$app4 set packetSize_ 512
$app4 set rate_ 600Kb
$app4 attach-agent $agent4

# Define node size in Network Animator
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns initial_node_pos $node_($i) 20
}

# Initialize Flags
set ht 0
set hs 0

set ht1 0
set hs1 0

set ht2 0
set hs2 0

set ht3 0
set hs3 0

set hr1 0
set hr2 0
set hr3 0
```

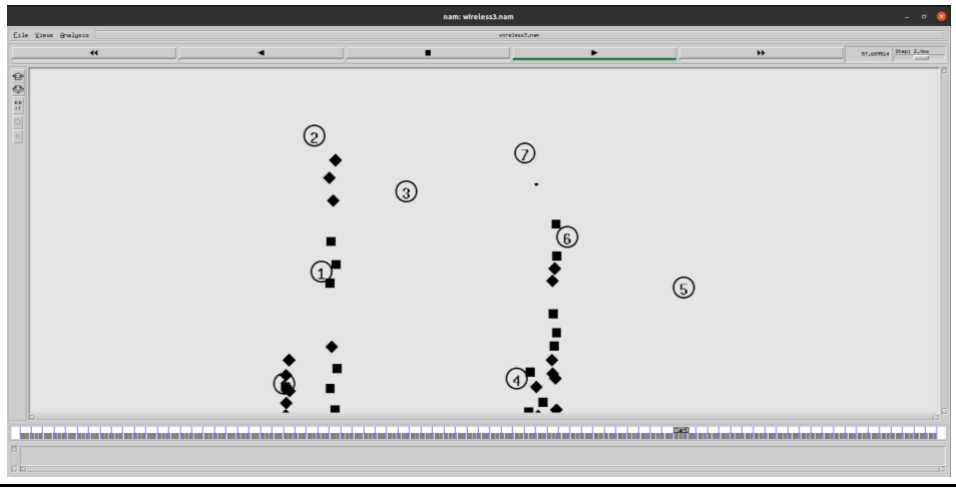
```
set hr4 0

# Function To record Statistcis (Bit Rate, Delay, Drop)
proc record {} {
    global sink sink2 sink3 sink4 f0 f1 f2 f3 f4 f5 f6 f7 f8 ht hs ht1 hs1 ht2 hs2 ht3 hs3 f8 f9 f10 f11
    hr1 hr2 hr3 hr4
    set ns [Simulator instance]
    set time 0.9 ; #Set Sampling Time to 0.9 Sec
    set bw0 [$sink set bytes_]
    set bw1 [$sink2 set bytes_]
    set bw2 [$sink3 set bytes_]
    set bw3 [$sink4 set bytes_]
    set bw4 [$sink set nlost_]
    set bw5 [$sink2 set nlost_]
    set bw6 [$sink3 set nlost_]
    set bw7 [$sink4 set nlost_]
    set bw8 [$sink set lastPktTime_]
    set bw9 [$sink set npkts_]
    set bw10 [$sink2 set lastPktTime_]
    set bw11 [$sink2 set npkts_]
    set bw12 [$sink3 set lastPktTime_]
    set bw13 [$sink3 set npkts_]
    set bw14 [$sink4 set lastPktTime_]
    set bw15 [$sink4 set npkts_]
    set now [$ns now]

    # Record the Bit Rate in Trace Files
    puts $f0 "$now [expr (($bw0+$hr1)*8)/(2*$time*1000000)]"
    puts $f1 "$now [expr (($bw1+$hr2)*8)/(2*$time*1000000)]"
    puts $f2 "$now [expr (($bw2+$hr3)*8)/(2*$time*1000000)]"
    puts $f3 "$now [expr (($bw3+$hr4)*8)/(2*$time*1000000)]"

    # Record Packet Loss Rate inFile
    puts $f4 "$now [expr $bw4/$time]"
    puts $f5 "$now [expr $bw5/$time]"
    puts $f6 "$now [expr $bw6/$time]"
    puts $f7 "$now [expr $bw7/$time]"

    # Record Packet Delay in File
    if { $bw9 > $hs } {
        puts $f8 "$now [expr ($bw8 - $ht)/($bw9 - $hs)]"
    } else {
        puts $f8 "$now [expr ($bw9 - $hs)]"
    }

    if { $bw11 > $hs1 } {
        puts $f9 "$now [expr ($bw10 - $ht1)/($bw11 - $hs1)]"
    } else {
        puts $f9 "$now [expr ($bw11 - $hs1)]"
    }

    if { $bw13 > $hs2 } {
        puts $f10 "$now [expr ($bw12 - $ht2)/($bw13 - $hs2)]"
```

```
        } else {
                puts $f10 "$now [expr ($bw13 - $hs2)]"
        }

        if { $bw15 > $hs3 } {
                puts $f11 "$now [expr ($bw14 - $ht3)/($bw15 - $hs3)]"
        } else {
                puts $f11 "$now [expr ($bw15 - $hs3)]"
        }

        # Reset Variables
        $sink set bytes_ 0
        $sink2 set bytes_ 0
        $sink3 set bytes_ 0
        $sink4 set bytes_ 0

        $sink set nlost_ 0
        $sink2 set nlost_ 0
        $sink3 set nlost_ 0
        $sink4 set nlost_ 0

        set ht $bw8
        set hs $bw9
        set hr1 $bw0
        set hr2 $bw1
        set hr3 $bw2
        set hr4 $bw3

        $ns at [expr $now+$time] "record" ;# Schedule Record after $time interval sec
}

# Start Recording at Time 0
$ns at 0.0 "record"
$ns at 1.4 "$app1 start" ;

# Start transmission at 2 Sec
$ns at 10.0 "$app2 start" ;

# Start transmission at 5 Sec
$ns at 20.0 "$app3 start" ;

# Start transmission at 15 Sec
$ns at 30.0 "$app4 start" ;

# Start transmission at 25 Sec
# Stop Simulation at Time 70 sec
$ns at 80.0 "finish"

# Reset Nodes at time 80 sec
for {set i 0} {$i < $val(nn) } {incr i} {
        $ns at 80.0 "$node_($i) reset";
}
```

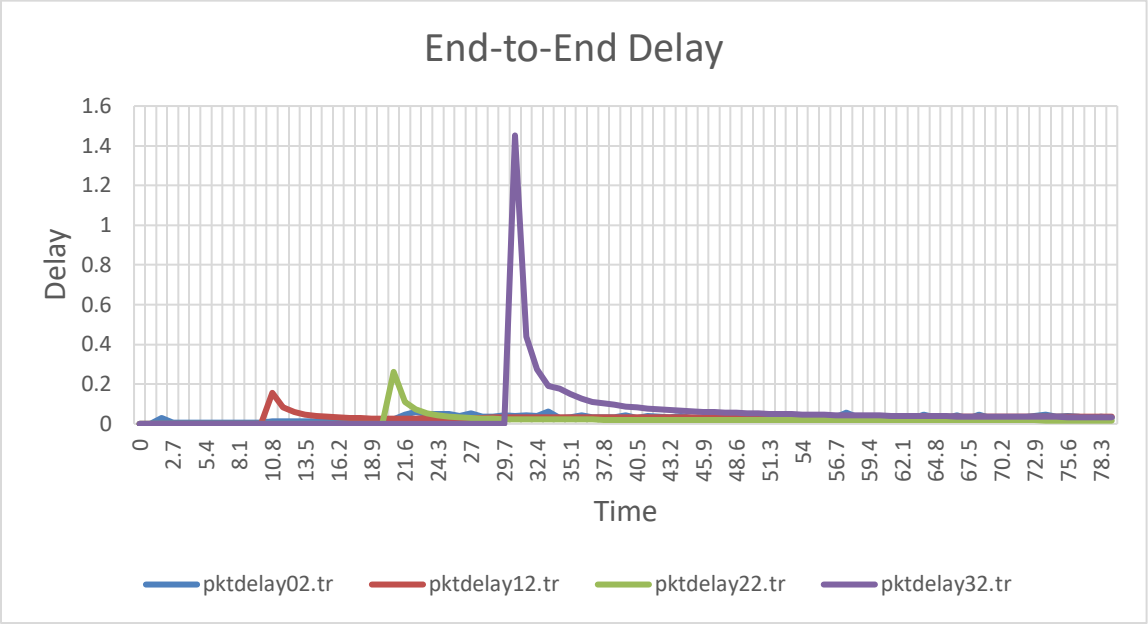
```
# Exit Simulation at Time70.01 sec
$ns at 80.01 "puts \"NS EXITING...\" ; $ns halt"

proc finish {} {

    global ns tracef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11

    # Close Trace Files
    close $f0
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    close $f6
    close $f7
    close $f8
    close $f9
    close $f10
    close $f11

    exec nam wireless3.nam &
    # Plot the characteristics using xgraph
    exec xgraph thru02.tr thru12.tr thru22.tr thru32.tr -geometry 800x400 &
    exec xgraph pktloss02.tr pktloss12.tr pktloss22.tr pktloss32.tr -geometry 800x400 &
    exec xgraph pktdelay02.tr pktdelay12.tr pktdelay22.tr pktdelay32.tr -geometry 800x400 &

    # Reset Trace File
    $ns flush-trace
    close $tracef
    exit 0
}

puts "Starting Simulation..."
$ns run
```

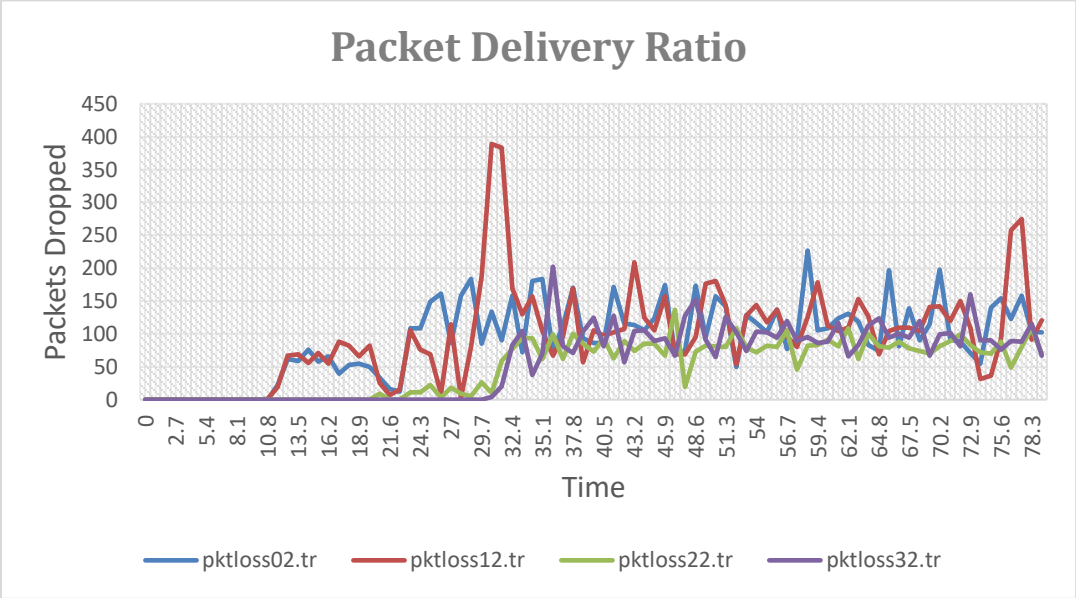
OUTPUT:

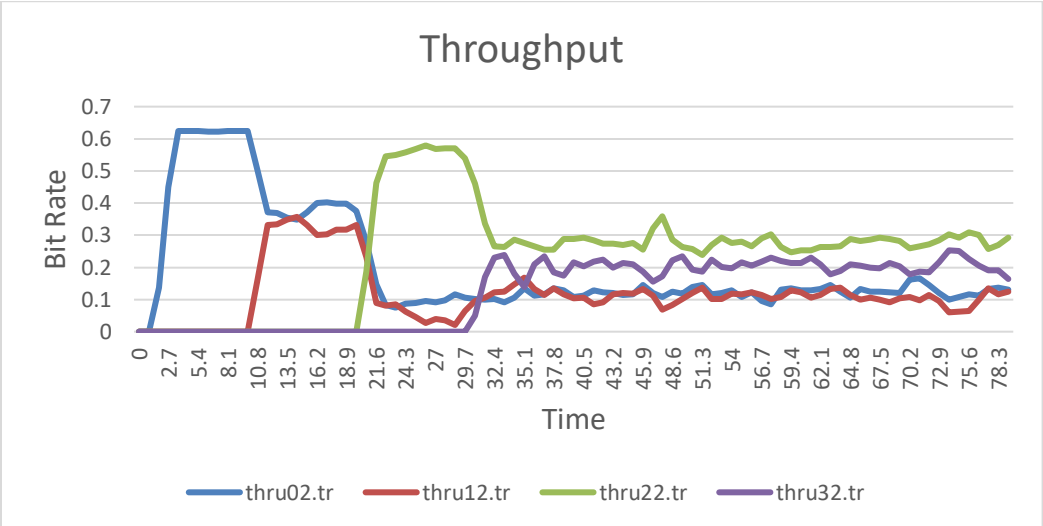
1. NAM Simulator



2. Graph


End-to-End Delay
Delay
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 2.7 5.4 8.1 10.8 13.5 16.2 18.9 21.6 24.3 27 29.7 32.4 35.1 37.8 40.5 43.2 45.9 48.6 51.3 54 56.7 59.4 62.1 64.8 67.5 70.2 72.9 75.6 78.3
Time
pktdelay02.tr
pktdelay12.tr
pktdelay22.tr
pktdelay32.tr


Packet Delivery Ratio
Packets Dropped
450
400
350
300
250
200
150
100
50
0
0 2.7 5.4 8.1 10.8 13.5 16.2 18.9 21.6 24.3 27 29.7 32.4 35.1 37.8 40.5 43.2 45.9 48.6 51.3 54 56.7 59.4 62.1 64.8 67.5 70.2 72.9 75.6 78.3
Time
pktloss02.tr
pktloss12.tr
pktloss22.tr
pktloss32.tr


Throughput
Bit Rate
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 2.7 5.4 8.1 10.8 13.5 16.2 18.9 21.6 24.3 27 29.7 32.4 35.1 37.8 40.5 43.2 45.9 48.6 51.3 54 56.7 59.4 62.1 64.8 67.5 70.2 72.9 75.6 78.3
Time
thru02.tr
thru12.tr
thru22.tr
thru32.tr

LAB ASSIGNMENT - 07

AIM/OBJECTIVE:

Implement an Infrastructure-less wireless network comprising 'N' nodes in an area of 1000 x 1000 square meters. At the routing layer use AODV and DSR protocols.

Perform the simulations with a varying number of nodes.

Plot and compare the performance of AODV and DSR in terms of the End-to-End delay, Throughput, and packet delivery ratio using the NS2/NS3 simulator.

THEORY:

- **End-to-End Delay**

  It is the total latency experienced by a packet to traverse the network from the source to the destination.

  The end-to-end delay of a path is the summation of the node delay at each node plus the link delay at each link on the path.

- **Throughput**

  It is the measure of the average number of bits transmitted per second.

- **Packet Delivery Ratio**

  It is the measure of the average number of bits transmitted per second.

- **AODV protocol**

  It is referred as Ad Hoc On-Demand Distance Vector. It is routing protocol which is designed for wireless and mobile ad hoc network. AODV Protocol establishes route with destination only when it is required. AODV Protocol supports both unicast and multicast routing protocol.

- **DSR protocol**

  It is also referred to as Dynamic Source Routing Protocol. It is a reactive/on-demand routing protocol. In this type of routing, the route is discovered only when it is required/needed. The process of route discovery occurs by flooding the route request packets throughout the mobile network. In this protocol, Source node stores the complete path information and intermediate nodes do not need to maintain routing information.

CODE:

1. program.tcl (For AODV)

```
# Defining Node Configuration paramaters
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(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
set val(nn) 8 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 1000 ;# X dimension of the topography
set val(y) 1000;# Y dimensionof the topography

# Set the Mac Parameters, for more parameters, refer the ~ns-2.35/lib/ns-default.tcl Mac/802_11
set RTSThreshold_ 3000Mac/802_11
set basicRate_ 1MbMac/802_11
set dataRate_ 2Mb

# creation of tracefiles for variousmetrics
# *** Throughput Trace ***
set f0 [open thru02.tr w]
set f1 [open thru12.tr w]
set f2 [open thru22.tr w]
set f3 [open thru32.tr w]

#   ***  Packet  LossTrace ***
set f4 [open pktloss02.tr w]
set f5 [open pktloss12.tr w]
set f6 [open pktloss22.tr w]
set f7 [open pktloss32.tr w]


# *** Packet Delay Trace ***
set f8 [open pktdelay02.tr w]
set f9 [open pktdelay12.tr w]
set f10 [open pktdelay22.tr w]
set f11 [open pktdelay32.tr w]

# Simulator Object
set ns [new Simulator]

# Trace file initialization
set tracef [open wireless3.tr w]
$ns trace-all $tracef

# Network Animator
set namf [open wireless3.nam w]
$ns namtrace-all-wireless $namf $val(x) $val(y)

# Topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#creation of god (General Operations Director) object
create-god $val(nn)

# configure nodes
```

```
$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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace OFF

# Create Nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) random-motion 0 ; # disable random motion
}

#initial position of nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 10.0
$node_(1) set Y_ 15.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 35.0
$node_(2) set Y_ 250.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 10.0
$node_(3) set Y_ 50.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 235.0
$node_(4) set Y_ 10.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 400.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 285.0
$node_(6) set Y_ 150.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 120.0
$node_(7) set Y_ 115.0
```

```
$node_(7) set Z_ 0.0

# Create traffic flow using UDP with Constant Bit Rate Application
# this includes priority and the sink is LossMonitor agent to trace the bytes received (because the
    Null Agent does not handle this)
set agent1 [new Agent/UDP]
$agent1 set prio_ 0

set sink [new Agent/LossMonitor]
$ns attach-agent $node_(0) $agent1
$ns attach-agent $node_(1) $sink
$ns connect $agent1 $sink

set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ 512 ; # setting the packet size
$app1 set rate_ 600Kb ; # setting the rate at which the packets are transmitted
$app1 attach-agent $agent1 ; # attaching the agent

set agent2 [new Agent/UDP]
$agent2 set prio_ 1
set sink2 [new Agent/LossMonitor]
$ns attach-agent $node_(2) $agent2
$ns attach-agent $node_(3) $sink2
$ns connect $agent2 $sink2

set app2 [new Application/Traffic/CBR]
$app2 set packetSize_ 512
$app2 set rate_ 600Kb
$app2 attach-agent $agent2

set agent3 [new Agent/UDP]
$agent3 set prio_ 2
set sink3 [new Agent/LossMonitor]
$ns attach-agent $node_(4) $agent3
$ns attach-agent $node_(5) $sink3
$ns connect $agent3 $sink3

set app3 [new Application/Traffic/CBR]
$app3 set packetSize_ 512
$app3 set rate_ 600Kb
$app3 attach-agent $agent3

set agent4 [new Agent/UDP]
$agent4 set prio_ 3
set sink4 [new Agent/LossMonitor]
$ns attach-agent $node_(6) $agent4
$ns attach-agent $node_(7) $sink4
$ns connect $agent4 $sink4

set app4 [new Application/Traffic/CBR]
$app4 set packetSize_ 512
$app4 set rate_ 600Kb
```

```
$app4 attach-agent $agent4

# Define node size in Network Animator
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns initial_node_pos $node_($i) 20
}

# Initialize Flags
set ht 0
set hs 0

set ht1 0
set hs1 0

set ht2 0
set hs2 0

set ht3 0
set hs3 0

set hr1 0
set hr2 0
set hr3 0
set hr4 0

# Function To record Statistcis (Bit Rate, Delay, Drop)
proc record {} {
    global sink sink2 sink3 sink4 f0 f1 f2 f3 f4 f5 f6 f7 f8 ht hs ht1 hs1 ht2 hs2 ht3 hs3 f8 f9 f10 f11
    hr1 hr2 hr3 hr4
    set ns [Simulator instance]
    set time 0.9 ; #Set Sampling Time to 0.9 Sec
    set bw0 [$sink set bytes_]
    set bw1 [$sink2 set bytes_]
    set bw2 [$sink3 set bytes_]
    set bw3 [$sink4 set bytes_]

    set bw4 [$sink set nlost_]
    set bw5 [$sink2 set nlost_]
    set bw6 [$sink3 set nlost_]
    set bw7 [$sink4 set nlost_]

            set bw8 [$sink set lastPktTime_]
    set bw9 [$sink set npkts_]
    set bw10 [$sink2 set lastPktTime_]
    set bw11 [$sink2 set npkts_]
    set bw12 [$sink3 set lastPktTime_]
    set bw13 [$sink3 set npkts_]
    set bw14 [$sink4 set lastPktTime_]
    set bw15 [$sink4 set npkts_]

    set now [$ns now]
```

```
# Record the Bit Rate in Trace Files
puts $f0 "$now [expr (($bw0+$hr1)*8)/(2*$time*1000000)]"
puts $f1 "$now [expr (($bw1+$hr2)*8)/(2*$time*1000000)]"
puts $f2 "$now [expr (($bw2+$hr3)*8)/(2*$time*1000000)]"
puts $f3 "$now [expr (($bw3+$hr4)*8)/(2*$time*1000000)]"

# Record Packet Loss Rate inFile
puts $f4 "$now [expr $bw4/$time]"
puts $f5 "$now [expr $bw5/$time]"
puts $f6 "$now [expr $bw6/$time]"
puts $f7 "$now [expr $bw7/$time]"

# Record Packet Delay in File
if { $bw9 > $hs } {
        puts $f8 "$now [expr ($bw8 - $ht)/($bw9 - $hs)]"
} else {
        puts $f8 "$now [expr ($bw9 - $hs)]"
}

if { $bw11 > $hs1 } {
        puts $f9 "$now [expr ($bw10 - $ht1)/($bw11 - $hs1)]"
} else {
        puts $f9 "$now [expr ($bw11 - $hs1)]"
}

if { $bw13 > $hs2 } {
        puts $f10 "$now [expr ($bw12 - $ht2)/($bw13 - $hs2)]"
} else {
        puts $f10 "$now [expr ($bw13 - $hs2)]"
}

if { $bw15 > $hs3 } {
        puts $f11 "$now [expr ($bw14 - $ht3)/($bw15 - $hs3)]"
} else {
        puts $f11 "$now [expr ($bw15 - $hs3)]"
}

# Reset Variables
$sink set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0
$sink4 set bytes_ 0

$sink set nlost_ 0
$sink2 set nlost_ 0
$sink3 set nlost_ 0
$sink4 set nlost_ 0

set ht $bw8
set hs $bw9
set hr1 $bw0
set hr2 $bw1
```

```
    set hr3 $bw2
    set hr4 $bw3

    $ns at [expr $now+$time] "record" ;# Schedule Record after $time interval sec
}


# Start Recording at Time 0
$ns at 0.0 "record"
$ns at 1.4 "$app1 start" ;

# Start transmission at 2 Sec
$ns at 10.0 "$app2 start" ;

# Start transmission at 5 Sec
$ns at 20.0 "$app3 start" ;

# Start transmission at 15 Sec
$ns at 30.0 "$app4 start" ;

# Start transmission at 25 Sec
# Stop Simulation at Time 70 sec
$ns at 80.0 "finish"

# Reset Nodes at time 80 sec
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns at 80.0 "$node_($i) reset";
}

# Exit Simulation at Time70.01 sec
$ns at 80.01 "puts \"NS EXITING...\" ; $ns halt"


proc finish {} {
    global ns tracef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11

    # Close Trace Files
    close $f0
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    close $f6
    close $f7
    close $f8
    close $f9
    close $f10
    close $f11

    exec nam wireless3.nam &
    # Plot the characteristics using xgraph
```

```
        exec xgraph thru02.tr thru12.tr thru22.tr thru32.tr &
        exec xgraph pktloss02.tr pktloss12.tr pktloss22.tr pktloss32.tr -geometry 800x400 &
        exec xgraph pktdelay02.tr pktdelay12.tr pktdelay22.tr pktdelay32.tr -geometry 800x400 &

        # Reset Trace File
        $ns flush-trace
        close $tracef
        exit 0
    }

    puts "Starting Simulation..."
    $ns run
```

2. program.tcl (For DSR)

```
# Defining Node Configuration paramaters
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(ifq) CMUPriQueue ;# 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
set val(nn) 8 ;# number of mobilenodes
set val(rp) DSR ;# routing protocol
set val(x) 1000 ;# X dimension of the topography
set val(y) 1000;# Y dimensionof the topography

# Set the Mac Parameters, for more parameters, refer the ~ns-2.35/lib/ns-default.tcl Mac/802_11
set RTSThreshold_ 3000Mac/802_11
set basicRate_ 1MbMac/802_11
set dataRate_ 2Mb

# creation of tracefiles for variousmetrics
# *** Throughput Trace ***
set f0 [open thru02.tr w]
set f1 [open thru12.tr w]
set f2 [open thru22.tr w]
set f3 [open thru32.tr w]

#   ***  Packet  LossTrace ***
set f4 [open pktloss02.tr w]
set f5 [open pktloss12.tr w]
set f6 [open pktloss22.tr w]
set f7 [open pktloss32.tr w]


# *** Packet Delay Trace ***
set f8 [open pktdelay02.tr w]
set f9 [open pktdelay12.tr w]
set f10 [open pktdelay22.tr w]
```

```
set f11 [open pktdelay32.tr w]

# Simulator Object
set ns [new Simulator]

# Trace file initialization
set tracef [open wireless3.tr w]
$ns trace-all $tracef

# Network Animator
set namf [open wireless3.nam w]
$ns namtrace-all-wireless $namf $val(x) $val(y)

# Topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#creation of god (General Operations Director) object
create-god $val(nn)

# configure nodes
$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) \
        -channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF \
        -movementTrace OFF

# Create Nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) random-motion 0 ; # disable random motion
}

#initial position of nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 10.0
$node_(1) set Y_ 15.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 35.0
```

```
$node_(2) set Y_ 250.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 10.0
$node_(3) set Y_ 50.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 235.0
$node_(4) set Y_ 10.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 400.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 285.0
$node_(6) set Y_ 150.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 120.0
$node_(7) set Y_ 115.0
$node_(7) set Z_ 0.0

# Create traffic flow using UDP with Constant Bit Rate Application
# this includes priority and the sink is LossMonitor agent to trace the bytes received (because the Null
    Agent does not handle this)

set agent1 [new Agent/UDP]
$agent1 set prio_ 0
set sink [new Agent/LossMonitor]
$ns attach-agent $node_(0) $agent1
$ns attach-agent $node_(1) $sink
$ns connect $agent1 $sink

set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ 512 ; # setting the packet size
$app1 set rate_ 600Kb ; # setting the rate at which the packets are transmitted
$app1 attach-agent $agent1 ; # attaching the agent


set agent2 [new Agent/UDP]
$agent2 set prio_ 1
set sink2 [new Agent/LossMonitor]
$ns attach-agent $node_(2) $agent2
$ns attach-agent $node_(3) $sink2
$ns connect $agent2 $sink2

set app2 [new Application/Traffic/CBR]
$app2 set packetSize_ 512
$app2 set rate_ 600Kb
$app2 attach-agent $agent2
```

```
set agent3 [new Agent/UDP]
$agent3 set prio_ 2
set sink3 [new Agent/LossMonitor]
$ns attach-agent $node_(4) $agent3
$ns attach-agent $node_(5) $sink3
$ns connect $agent3 $sink3

set app3 [new Application/Traffic/CBR]
$app3 set packetSize_ 512
$app3 set rate_ 600Kb
$app3 attach-agent $agent3


set agent4 [new Agent/UDP]
$agent4 set prio_ 3
set sink4 [new Agent/LossMonitor]
$ns attach-agent $node_(6) $agent4
$ns attach-agent $node_(7) $sink4
$ns connect $agent4 $sink4

set app4 [new Application/Traffic/CBR]
$app4 set packetSize_ 512
$app4 set rate_ 600Kb
$app4 attach-agent $agent4

# Define node size in Network Animator
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns initial_node_pos $node_($i) 20
}

# Initialize Flags
set ht 0
set hs 0

set ht1 0
set hs1 0

set ht2 0
set hs2 0

set ht3 0
set hs3 0

set hr1 0
set hr2 0
set hr3 0
set hr4 0

# Function To record Statistcis (Bit Rate, Delay, Drop)
proc record {} {
     global sink sink2 sink3 sink4 f0 f1 f2 f3 f4 f5 f6 f7 f8 ht hs ht1 hs1 ht2 hs2 ht3 hs3 f8 f9 f10 f11 hr1
```

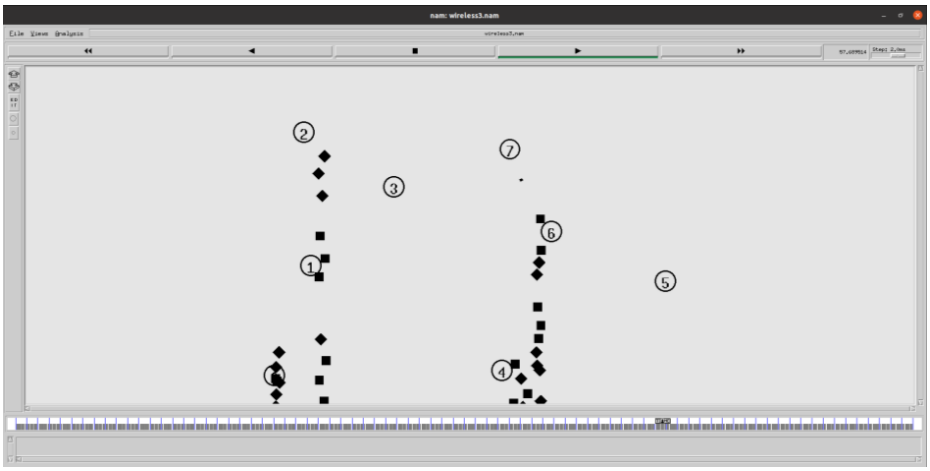
```
hr2 hr3 hr4
set ns [Simulator instance]
set time 0.9 ; #Set Sampling Time to 0.9 Sec

set bw0 [$sink set bytes_]
set bw1 [$sink2 set bytes_]
set bw2 [$sink3 set bytes_]
set bw3 [$sink4 set bytes_]

set bw4 [$sink set nlost_]
set bw5 [$sink2 set nlost_]
set bw6 [$sink3 set nlost_]
set bw7 [$sink4 set nlost_]

set bw8 [$sink set lastPktTime_]
set bw9 [$sink set npkts_]
set bw10 [$sink2 set lastPktTime_]
set bw11 [$sink2 set npkts_]
set bw12 [$sink3 set lastPktTime_]
set bw13 [$sink3 set npkts_]
set bw14 [$sink4 set lastPktTime_]
set bw15 [$sink4 set npkts_]

set now [$ns now]

# Record the Bit Rate in Trace Files
puts $f0 "$now [expr (($bw0+$hr1)*8)/(2*$time*1000000)]"
puts $f1 "$now [expr (($bw1+$hr2)*8)/(2*$time*1000000)]"
puts $f2 "$now [expr (($bw2+$hr3)*8)/(2*$time*1000000)]"
puts $f3 "$now [expr (($bw3+$hr4)*8)/(2*$time*1000000)]"

# Record Packet Loss Rate inFile
puts $f4 "$now [expr $bw4/$time]"
puts $f5 "$now [expr $bw5/$time]"
puts $f6 "$now [expr $bw6/$time]"
puts $f7 "$now [expr $bw7/$time]"

# Record Packet Delay in File
if { $bw9 > $hs } {
    puts $f8 "$now [expr ($bw8 - $ht)/($bw9 - $hs)]"
} else {
    puts $f8 "$now [expr ($bw9 - $hs)]"
}

if { $bw11 > $hs1 } {
    puts $f9 "$now [expr ($bw10 - $ht1)/($bw11 - $hs1)]"
} else {
    puts $f9 "$now [expr ($bw11 - $hs1)]"
}

if { $bw13 > $hs2 } {
    puts $f10 "$now [expr ($bw12 - $ht2)/($bw13 - $hs2)]"
```

```
        } else {
            puts $f10 "$now [expr ($bw13 - $hs2)]"
        }

        if { $bw15 > $hs3 } {
            puts $f11 "$now [expr ($bw14 - $ht3)/($bw15 - $hs3)]"
        } else {
            puts $f11 "$now [expr ($bw15 - $hs3)]"
        }

        # Reset Variables
        $sink set bytes_ 0
        $sink2 set bytes_ 0
        $sink3 set bytes_ 0
        $sink4 set bytes_ 0

        $sink set nlost_ 0
        $sink2 set nlost_ 0
        $sink3 set nlost_ 0
        $sink4 set nlost_ 0

        set ht $bw8
        set hs $bw9
        set hr1 $bw0
        set hr2 $bw1
        set hr3 $bw2
        set hr4 $bw3

        $ns at [expr $now+$time] "record" ;# Schedule Record after $time interval sec
    }

    # Start Recording at Time 0
    $ns at 0.0 "record"
    $ns at 1.4 "$app1 start" ;

    # Start transmission at 2 Sec
    $ns at 10.0 "$app2 start" ;

    # Start transmission at 5 Sec
    $ns at 20.0 "$app3 start" ;

    # Start transmission at 15 Sec
    $ns at 30.0 "$app4 start" ;

    # Start transmission at 25 Sec
    # Stop Simulation at Time 70 sec
    $ns at 80.0 "finish"

    # Reset Nodes at time 80 sec
    for {set i 0} {$i < $val(nn) } {incr i} {
        $ns at 80.0 "$node_($i) reset";
    }
```

```
# Exit Simulation at Time70.01 sec
$ns at 80.01 "puts \"NS EXITING...\" ; $ns halt"


proc finish {} {
    global ns tracef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11

    # Close Trace Files
    close $f0
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    close $f6
    close $f7
    close $f8
    close $f9
    close $f10
    close $f11

    exec nam wireless3.nam &
    # Plot the characteristics using xgraph
    exec xgraph thru02.tr thru12.tr thru22.tr thru32.tr &
    exec xgraph pktloss02.tr pktloss12.tr pktloss22.tr pktloss32.tr -geometry 800x400 &
    exec xgraph pktdelay02.tr pktdelay12.tr pktdelay22.tr pktdelay32.tr -geometry 800x400 &

    # Reset Trace File
    $ns flush-trace
    close $tracef
    exit 0
}

puts "Starting Simulation..."
$ns run
```

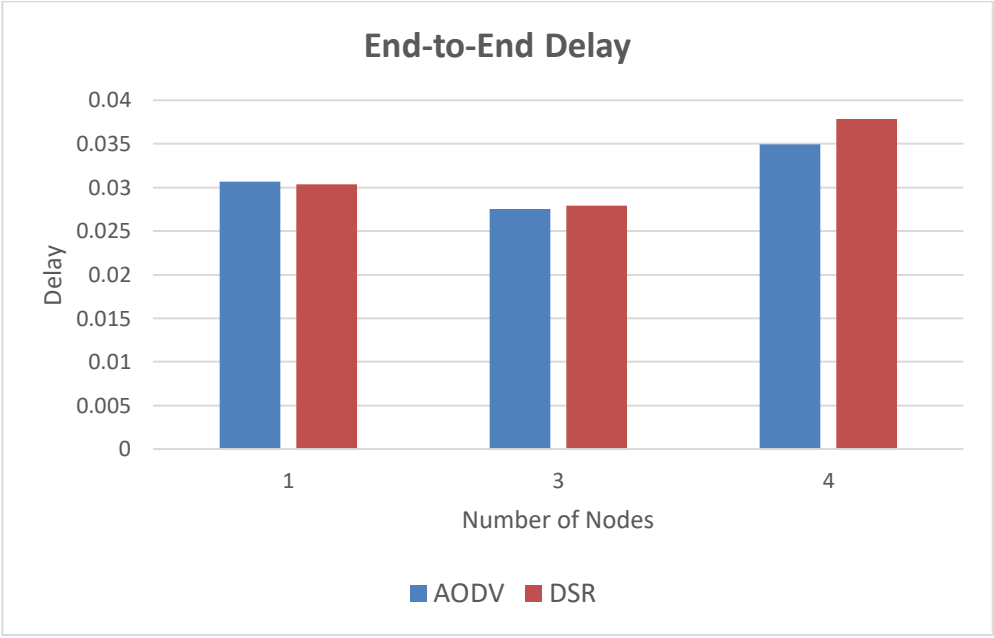
OUTPUT:

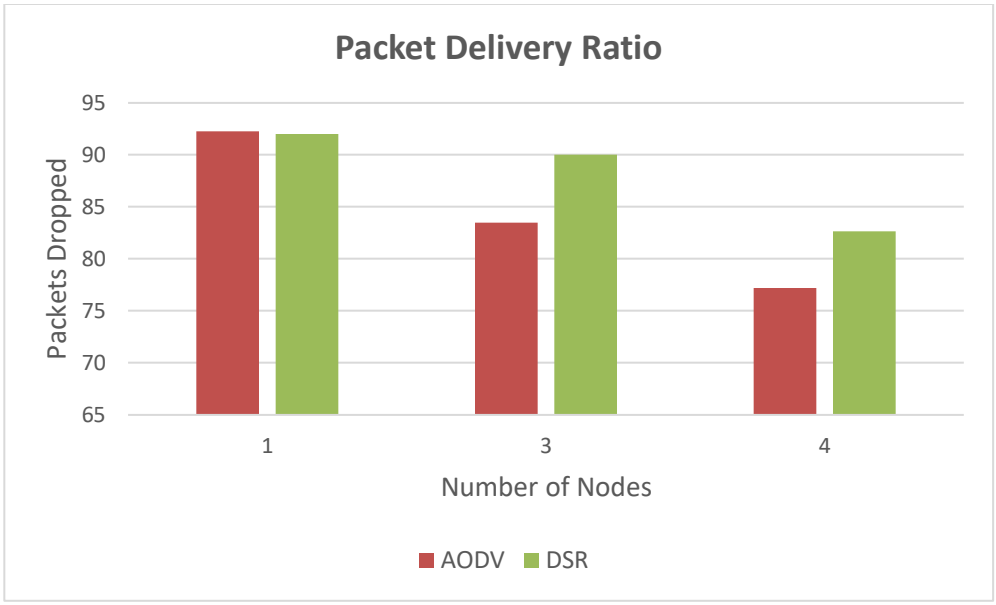
1. NAM Simulator (For AODV)

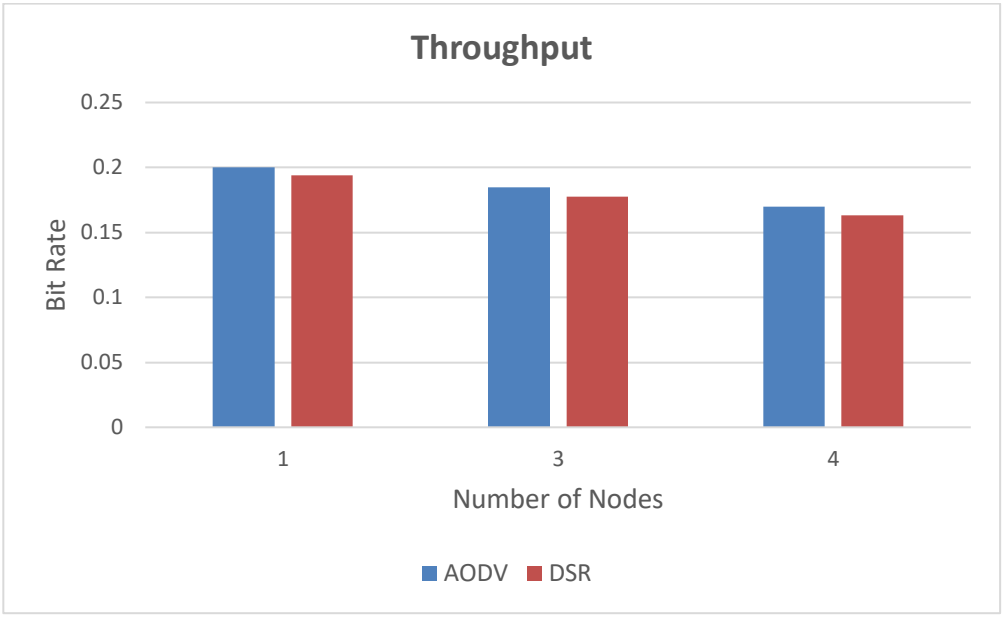


NAM Simulator (For DSR)

2. Graph




Packet Delivery Ratio
95
90
85
80
75
70
65
Packets Dropped
1
3
4
Number of Nodes
AODV
DSR


Throughput
0.25
0.2
0.15
0.1
0.05
0
Bit Rate
1
3
4
Number of Nodes
AODV
DSR

LAB ASSIGNMENT - 08

AIM/OBJECTIVE:

Implement an Infrastructure-less wireless network comprising of 'N' MOBILE nodes.

Perform the simulations with varying speed of the nodes.

Plot and compare the performance of different MAC layer protocols using NS2/NS3 simulator in terms of

(1) Number of collisions,

(2) Number of Control packets

THEORY:

- **IEEE 802.11**

  IEEE 802.11 standard, popularly known as **Wi-Fi**, lays down the architecture and specifications of **wireless LANs (WLANs)**. Wi-Fi or WLAN uses high-frequency radio waves instead of cables for connecting the devices in **LAN**. Users connected by WLANs can move around within the area of network coverage.

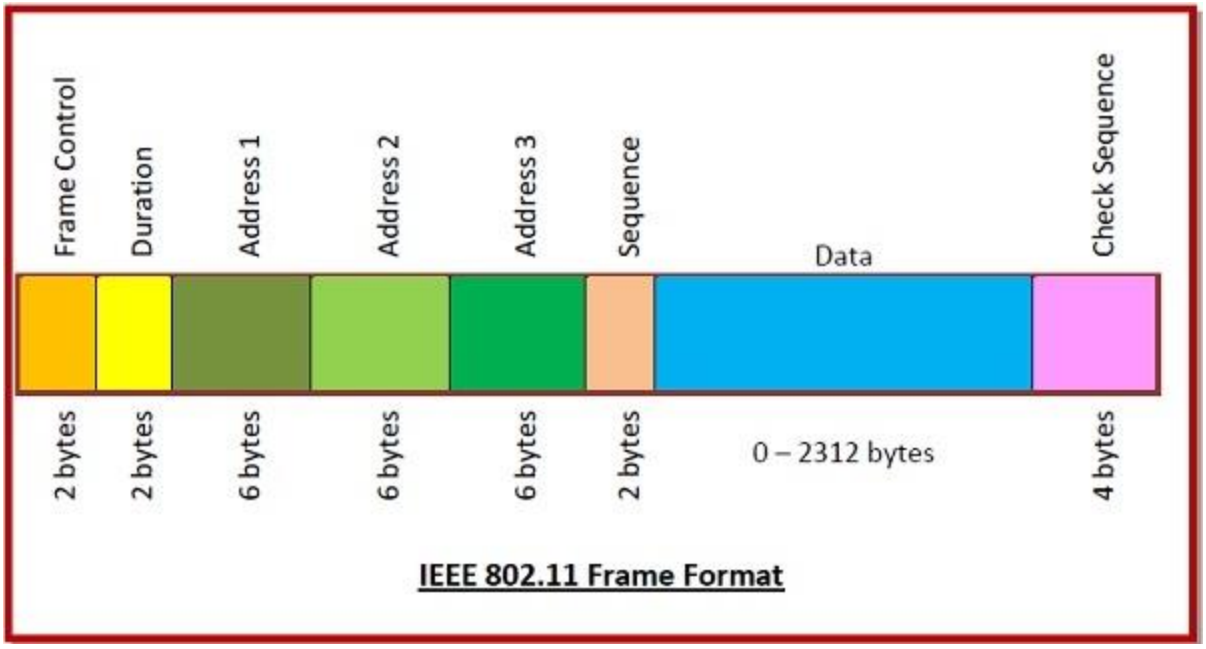
- **IEEE 802.11 Architecture**

  - **Stations (STA)** − Stations comprises of all devices and equipment that are connected to the wireless LAN. A station can be of two types−
    - Wireless Access Point (WAP) − WAPs or simply access points (AP) are generally wireless routers that form the base stations or access.
    - Client. Clients are workstations, computers, laptops, printers, smartphones, etc.
  - Each station has a wireless network interface controller.
  - **Basic Service Set (BSS)** − A basic service set is a group of stations communicating at the physical layer level. BSS can be of two categories depending upon the mode of operation−
    - Infrastructure BSS − Here, the devices communicate with other devices through access points.
    - Independent BSS − Here, the devices communicate in a peer-to-peer basis in an ad hoc manner.
  - **Extended Service Set (ESS)** − It is a set of all connected BSS.
  - **Distribution System (DS)** − It connects access points in ESS.

- **IEEE 802.11 Frame Format**

  - **Frame Control** − It is a 2 bytes starting field composed of 11 subfields. It contains control information of the frame.
  - **Duration** − It is a 2-byte field that specifies the time period for which the frame and its acknowledgment occupy the channel.

- **Address fields** − There are three 6-byte address fields containing addresses of source, immediate destination, and final endpoint respectively.
- **Sequence** − It a 2 bytes field that stores the frame numbers.
- **Data** − This is a variable-sized field that carries the data from the upper layers. The maximum size of the data field is 2312 bytes.
- **Check Sequence** − It is a 4-byte field containing error detection information.



IEEE 802.11 Frame Format

CODE:

3. program.tcl

```
# Defining Node Configuration paramaters
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(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
set val(nn) 8 ;# number of mobilenodes
set val(rp) DSDV ;# routing protocol
set val(x) 400 ;# X dimension of the topography
set val(y) 300 ;# Y dimensionof the topography


# Simulator Object
set ns [new Simulator]

# Trace file initialization
set tracef [open wireless3.tr w]
$ns trace-all $tracef
```

```
# Network Animator
set namf [open wireless3.nam w]
$ns namtrace-all-wireless $namf $val(x) $val(y)

# Topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#creation of god (General Operations Director) object
create-god $val(nn)

set chan_1_ [new $val(chan)]

# configure nodes
$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_


# Create Nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) random-motion 0 ; # disable random motion
}

# Define node size in Network Animator
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns initial_node_pos $node_($i) 20
}

#initial position of nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 10.0
$node_(1) set Y_ 15.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 35.0
$node_(2) set Y_ 250.0
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 10.0
$node_(3) set Y_ 50.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 235.0
$node_(4) set Y_ 10.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 400.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 285.0
$node_(6) set Y_ 150.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 120.0
$node_(7) set Y_ 115.0
$node_(7) set Z_ 0.0


# simple node movements

#$ns at 3.0 "$node_(1) setdest 50.0 40.0 25.0"
#$ns at 3.0 "$node_(2) setdest 48.0 38.0 5.0"
#$ns at 3.0 "$node_(5) setdest 40.0 60.0 30.0"
#$ns at 3.0 "$node_(6) setdest 58.0 48.0 5.0"
#$ns at 3.0 "$node_(7) setdest 248.0 78.0 5.0"

#$ns at 20.0 "$node_(1) setdest 290.0 280.0 50.0"
#$ns at 20.0 "$node_(3) setdest 190.0 290.0 50.0"
#$ns at 20.0 "$node_(5) setdest 90.0 20.0 50.0"
#s$ns at 20.0 "$node_(7) setdest 110.0 50.0 10.0"

# Create traffic flow using TCP with Constant Bit Rate Application
# this includes priority and the sink is TCPSink agent to trace the bytes received (because the Null
    Agent does not handle this)
set agent1 [new Agent/TCP]
$agent1 set class_ 0
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $agent1
$ns attach-agent $node_(1) $sink
$ns connect $agent1 $sink
set app1 [new Application/FTP]
#$app1 set packetSize_ 150
#$app1 set interval_ 0.5
$app1 attach-agent $agent1 ; # attaching the agent


set agent2 [new Agent/TCP]
$agent2 set class_ 1
set sink2 [new Agent/TCPSink]
```

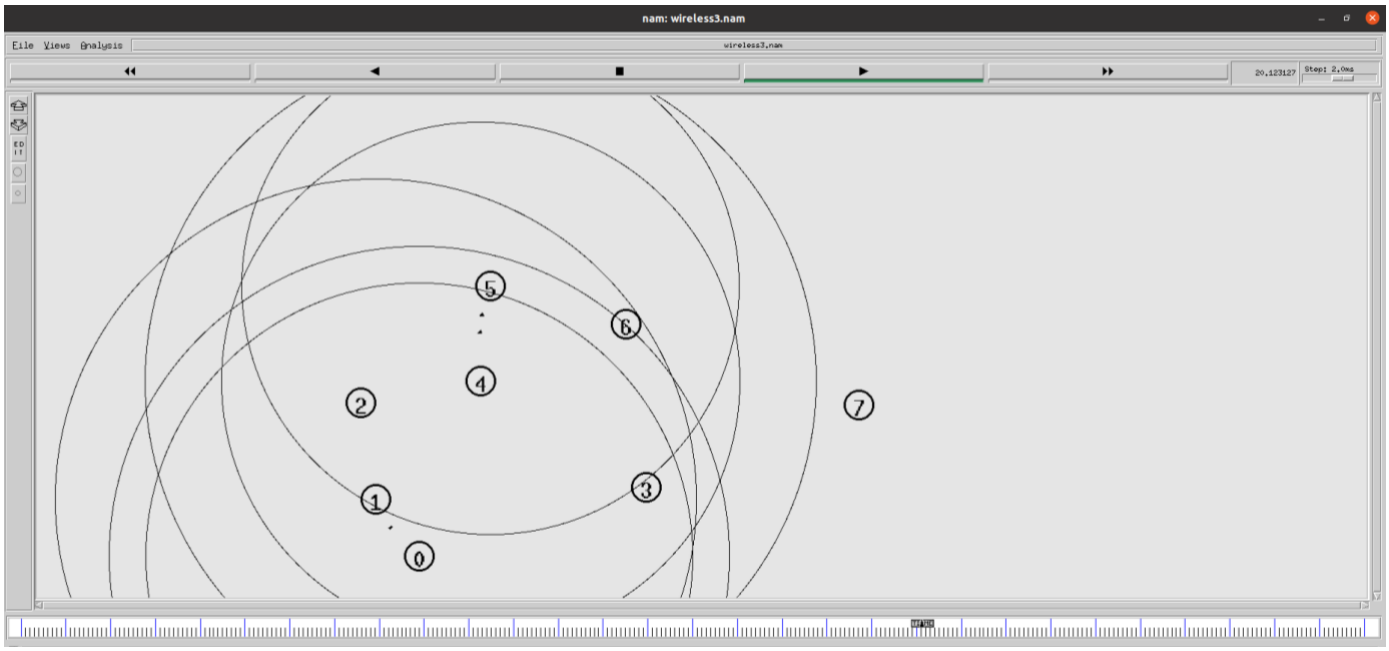
```
$ns attach-agent $node_(2) $agent2
$ns attach-agent $node_(5) $sink2
$ns connect $agent2 $sink2
set app2 [new Application/FTP]
#$app2 set packetSize_ 150
#$app2 set interval_ 0.5
$app2 attach-agent $agent2


set agent3 [new Agent/TCP]
$agent3 set class_ 2
set sink3 [new Agent/TCPSink]
$ns attach-agent $node_(4) $agent3
$ns attach-agent $node_(5) $sink3
$ns connect $agent3 $sink3
set app3 [new Application/FTP]
#$app3 set packetSize_ 150
#$app3 set interval_ 0.5
$app3 attach-agent $agent3


# Reset Nodes at time 80 sec
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns at 30.0 "$node_($i) reset";
}

$ns at 3.0 "$app1 start" ;

# Start transmission at 2 Sec
$ns at 6.0 "$app2 start" ;

$ns at 9.0 "$app3 start" ;

# Start transmission at 25 Sec
# Stop Simulation at Time 80 sec
$ns at 30.0 "finish"

proc finish {} {

    global ns tracef
    # Reset Trace File
    $ns flush-trace
    close $tracef
    exec nam wireless3.nam &
    exec awk -f exp8.awk wireless3.tr &
    exit 0
}

puts "Starting Simulation..."
$ns run
```

4. exp4.awk

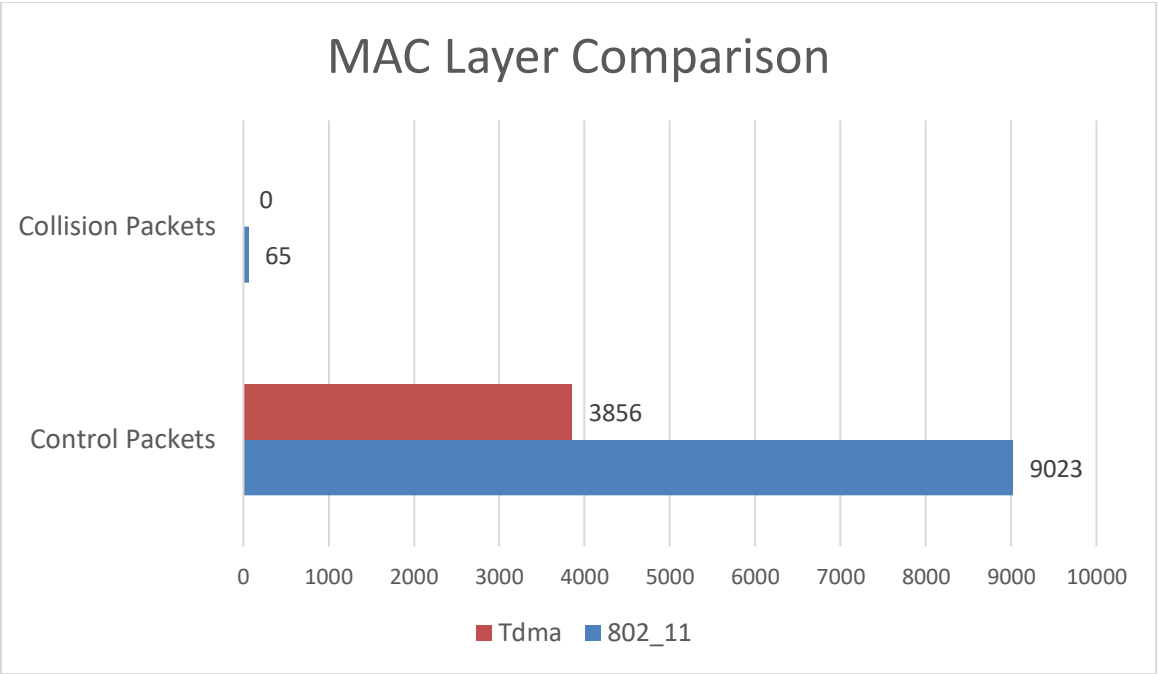
```
# Initialize variables
BEGIN {
   control_pac = 0
   collision_pac = 0
}

# Process each line in the trace file
{
   # Check if the line represents a packet transmission event
   if ($4 == "RTR") {        # Extract relevant information

      # Increment total packets sent
      control_pac++
   }

   # Check if the line represents a packet reception event
   if ($5 == "COL") {    # Extract relevant information
     collision_pac++
   }
}

# Calculate and print metrics
END {
   # Print metrics
   print "control packets : " , control_pac
   print "collision packets : " , collision_pac
}
```

OUTPUT:

1. NAM Simulator



2. Graph


MAC Layer Comparison
Collision Packets
Control Packets
0
65
3856
9023
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Tdma
802_11