COMPUTER NETWORKS LABORATORY – Part 2

Laboratory Experiments:

Part-2: Simulation experiments using NS2 / NS3 / OPNET / NCTUNS / NetSim / QualNet / Packet Tracer or any other equivalent tool

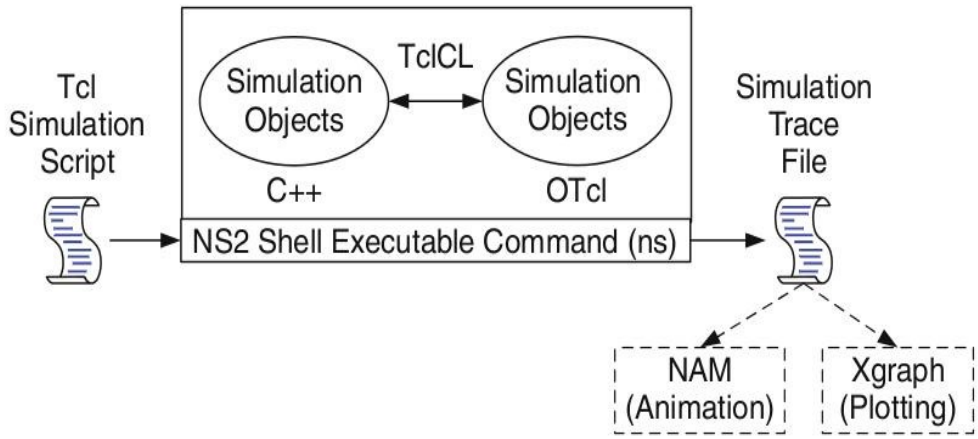
1. Implement a point-to-point network with four nodes and duplex links between them. Analyze the network performance by setting the queue size and vary the bandwidth and find the number of packets dropped.
2. Implement a four node point-to-point network with the links connected as follows: n0–n2, n1–n2 and n2–n3. Apply TCP agent between n0–n3 and UDP between n1–n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP.
3. Implement Ethernet LAN using n nodes (6-10), throughput by compare the throughput by changing the error rate and data rate.
4. Implement Ethernet LAN using n nodes and assign multiple traffic nodes and plot congestion window for different source / destination.
5. Implement ESS with transmitting nodes in wire-less LAN and obtain the performance parameters
6. Implementation of Link state routing algorithm.

SIMULATION USING NS2 SIMULATOR

Introduction to NS2

- Widely known as NS2, is simply an event driven simulation tool.
- 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.

Basic Architecture of NS2



TCL scripting

- Tcl is a general purpose scripting language. [Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage

Wired TCL Script Components

Create the event scheduler
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. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An NS simulation starts with the command

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

Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order 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

```
setnamfile [open out.nam w]

$ns namtrace-all $namfile
```

The above commands create a 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. Note that they begin 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 $ns flush-trace. In our case, this will be the file pointed by the pointer "$namfile" i.e., the file "out.tr".

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 namout.nam&
exit 0
}
```

The word proc declares a procedure, in this case finish 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 end the application and returns the number 0 as status to the system. exit 0 is the default for a clean exit.

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"
```

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

The simulation can then begin using the command

```
$ns run
```

Definition of a network of links and nodes

The way to define a node is

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

The node is created which is printed by the variable n0. When we shall refer to that node in the script we shall thus write $n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

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

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

```
set tcp [new Agent/TCP]
```

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command

```
set sink [new Agent /TCPSink]
```

Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

```
setudp [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

The below shows the definition of a CBR application using a UDP agent

The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

```
setcbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false
```

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command **$tcp set packetsize_ 552**.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of "1".We shall later give the flow identification of "2" to the UDP connection.

CBR over UDP

A UDP source and destination is defined in a similar way as in the case of TCP.

Instead of defining the rate in the command $cbr set rate_ 0.01Mb, one can define the time interval between transmission of packets using the command.

```
$cbr set interval_ 0.005
```

The packet size can be set to some value using

```
$cbr set packetSize_ <packet size>
```

Scheduling Events

NS is a discrete event based simulation. The tcp script defines when event should occur. The initializing command set ns [new Simulator] creates an event scheduler, and events are then scheduled using the format:

```
$ns at <time><event>
```

The scheduler is started when running ns that is through the command $ns run.
The beginning and end of the FTP and CBR application can be done through the following command

```
setcbr [new
Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false
```

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command **$tcp set packetSize_ 552**.
When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of "1".We shall later give the flow identification of "2" to the UDP connection.

XGRAPH

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

Trace Format Example

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop    (at queue)
```

```
src_addr : node.port (3.0)
dst_addr : node.port (0.0)
```

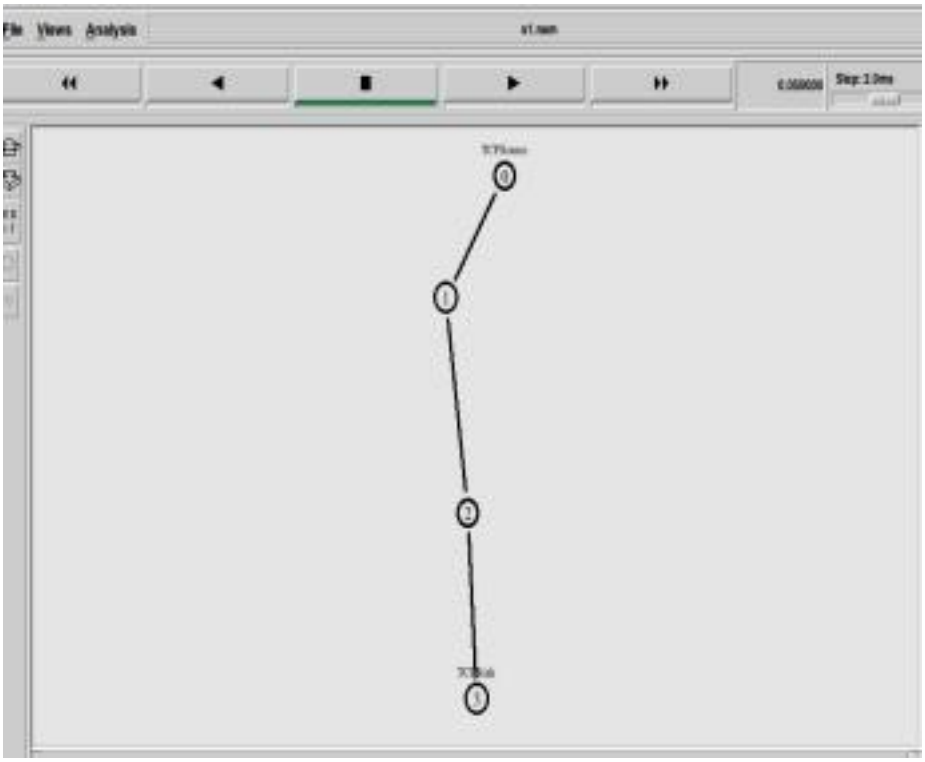
```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

1. An event (+, -, d, r) descriptor
2. Simulation time (in seconds) of that event
3. From node
4. To node (3, 4 identifies the link on which the event occurred)
5. Packet type (in Bytes)
6. Packet size (in Bytes)
7. Flags (appeared as "------" since no flag is set)
8. Flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script. Even though fid field may not be used in a simulation, users can use this field for analysis purposes. The fid field is also used when specifying stream color for the NAM display.
9. Source address in forms of "**node.port**".
10. Destination address in forms of "**node.port**".
11. Network layer protocol's packet sequence number. Note that even though UDP implementations do not use sequence number, NS keeps track of UDP packet sequence number for analysis purposes.
12. Unique id of the packet

Experiment No. 1

Four-Node Point-to-Point Network

Aim: Implement a point-to-point network with duplex links between them. Analyze the network performance by setting the queue size and vary the bandwidth and find the number of packets dropped.



s1.awk

```
BEGIN{
        count = 0;
}
{
        event = $1;
        if(event == "d")   { count++; }
}

END{
        printf("\nNumber of packets dropped is: %d\n", count);
}
```

s1.tcl

```
#create new simulation instance
set ns [new Simulator]

#open trace file
set tracefile [open s1.tr w]
$ns trace-all $tracefile

#open nam:animation file
set namfile [open s1.nam w]
$ns namtrace-all $namfile

#define finish procedure to perform at the end of simulation
proc finish {} {
        global ns tracefile namfile
        $ns flush-trace

        #dump all traces and close files
        close $tracefile
        close $namfile

        #execute nam animation file
        exec nam s1.nam &

        #execute awk file in background
        exec awk -f s1.awk s1.tr &
        exit 0
}

#create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#create labels
$n0 label "TCPSource"
$n3 label "TCPSink"

#set color
$ns color 1 red

#create link between nodes /create topology
$ns duplex-link $n0 $n1 1Mb 10ms DropTail $ns
duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
```

```
#set queue size of N packets between n2 and
n3 $ns queue-limit $n2 $n3 5

#create TCP agent and attach to node 0
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

#create TCP sink agent and attach to node
3 set tcpsink [new Agent/TCPSink] $ns attach-
agent $n3 $tcpsink

#create traffic: FTP: create FTP source agent on top of TCP and attach to TCP
agent
set ftp [new Application/FTP]
$ftp attach-agent $tcp

#connect TCP agent with TCP sink agent
$ns connect $tcp $tcpsink

#set the color
$tcp set class_  1

#schedule events
$ns at 0.2 "$ftp start"
$ns at 2.5 "$ftp stop"
$ns at 3.0 "finish"
$ns run
```

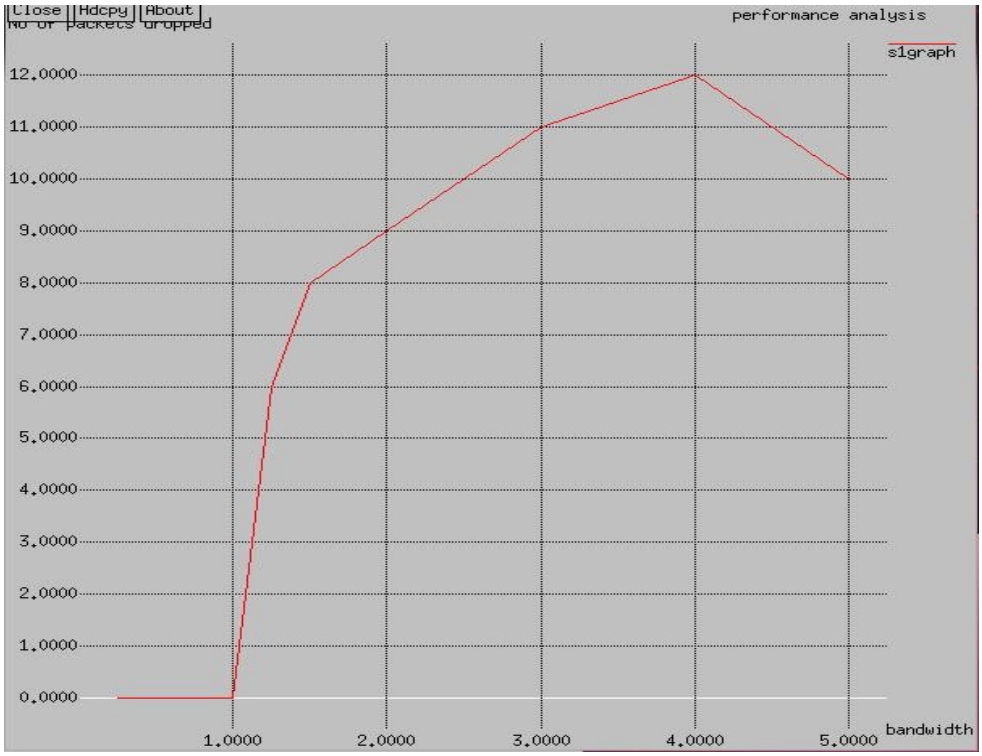
Steps

1) In the command line type **gedit s1.awk**
2) **gedit s1.tcl**
3) to run **sudo ns s1.tcl**
4) Vary the bandwidth from node 0 to 2 and keeping same bandwidth between 2 to 3 and note down packets dropped.
5) **gedit s1graph** and type the above result with reference to step 4.

Output: s1graph

```
0.25 0
0.50 0
0.75 0
1.00 0
1.25 6
1.50 8
2.00 9
3.00 11
4.00 12
5.00 10
```

6) **xgraph -x "Bandwidth(Mbps)" -y "No of packets dropped" -t "Performance analysis" s1graph**



Exercise:

1) Implement a point – to – point network with three nodes (for given orientation) with duplex links between them. Analyze the network performance by setting the queue size and vary the bandwidth and find the number of packets dropped using UDP application

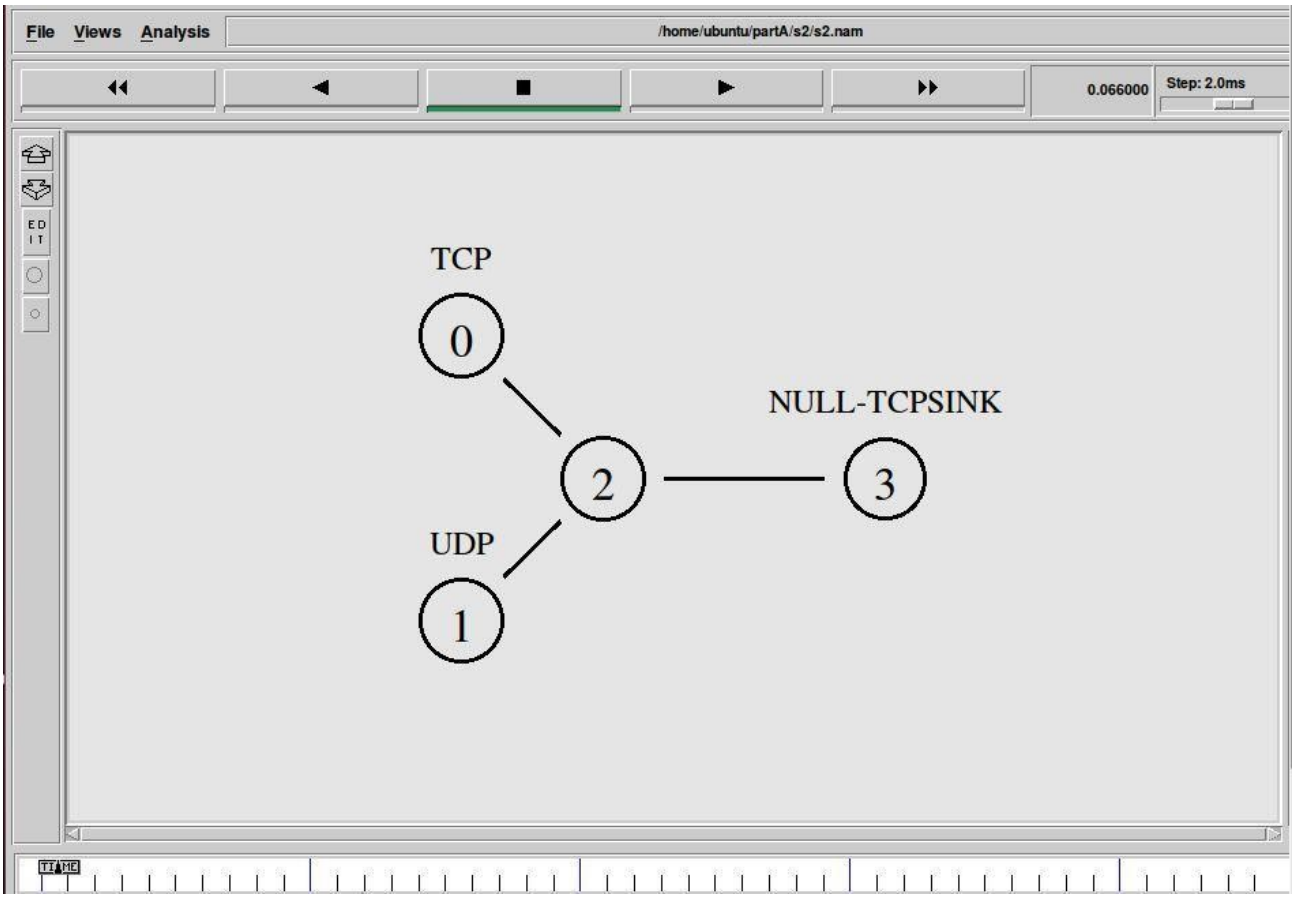
2) Implement a ring topology network with four nodes with duplex links between them. Set colors to the nodes, change the buffer type and use CBR application.

3) Implement a star topology network with three nodes and duplex links between them and set colors to the nodes and change shapes.

Experiment No. 2

Point-to-Point Network for the given Application

Aim: To implement a four node point-to-point network with the links connected as follows: n0–n2, n1–n2 and n2–n3. Apply TCP agent between n0–n3 and UDP between n1–n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP



s2.awk

```
BEGIN {
        ctcp = 0;
        cudp = 0;
}
{
        pkt = $5;
        if(pkt == "cbr") { cudp++; }
        if(pkt == "tcp") { ctcp++; }
}
END {
        printf("\nNo of packets sent\nTcp : %d \n Udp : %d\n", ctcp, cudp);
}
```

s2.tcl

```
set ns [new Simulator]

set namfile [open s2.nam w]
$ns namtrace-all $namfile
set tracefile [open s2.tr w]
$ns trace-all $tracefile

proc finish {} {
        global ns namfile tracefile
        $ns flush-trace
        close $namfile
        close $tracefile

        exec nam s2.nam &
        exec awk -f s2.awk s2.tr &
        exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$n0 label "TCP"
$n1 label "UDP"
$n3 label "NULL-TCPSINK"

$ns color 1 red
$ns color 2 blue

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2.75Mb 20ms DropTail

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

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

set tcpsink [new Agent/TCPSink]
$ns attach-agent $n3 $tcpsink

set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp

$ns connect $tcp $tcpsink

#Create a UDP Agent and attach to the node n1
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp

#Create a Null Agent and attach to the node n3
set null [new Agent/Null]
$ns attach-agent $n3 $null

#Create a CBR Traffic source and attach to the UDP
Agent set cbr [new Application/Traffic/CBR] $cbr attach-agent
$udp

#Specify the Packet Size and
interval $cbr set packetSize_ 500 $cbr
set interval_ 0.005

#Connect the CBR Traffic source to the Null
agent $ns connect $udp $null

$tcp set class_ 1
$udp set class_ 2

$ns at 0.2 "$cbr start"
$ns at 0.1 "$ftp start"
$ns at 4.5 "$cbr stop"
$ns at 4.4 "$ftp stop"
$ns at 5.0 "finish"
$ns run
```

s2graph

```
"TCP"
0.25    869
0.50    1608
0.75    2334
1.00    3066
1.25    3780
1.50    4500
1.75    5220
2.00    5940
2.25    6642
2.50    7110
2.75    7110

"CBR"
0.25    4478
0.50    5166
0.75    5166
1.00    5166
1.25    5166
1.50    5166
1.75    5166
2.00    5166
2.25    5166
2.50    5166
2.75    5166
```

Steps:

gedit s2.awk
gedit s2.tcl
sudo ns s2.tcl

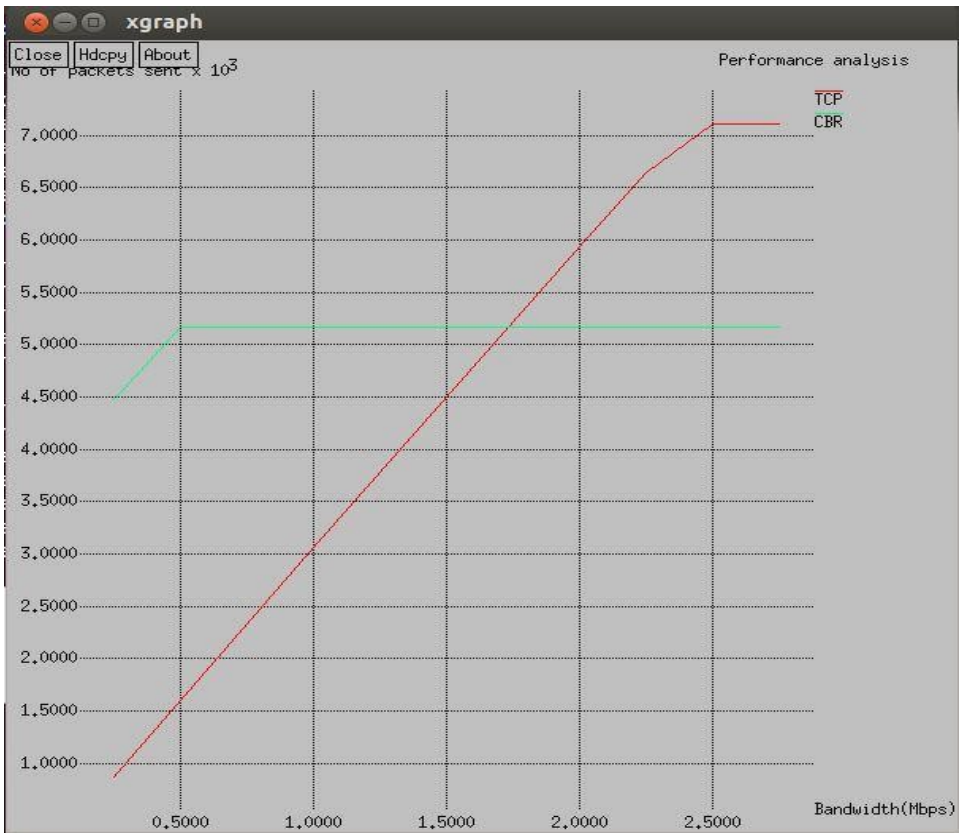
Vary the bandwidth and note down results

Output:

No of packets sent
Tcp : 5940
Udp : 5166

gedit s2graph

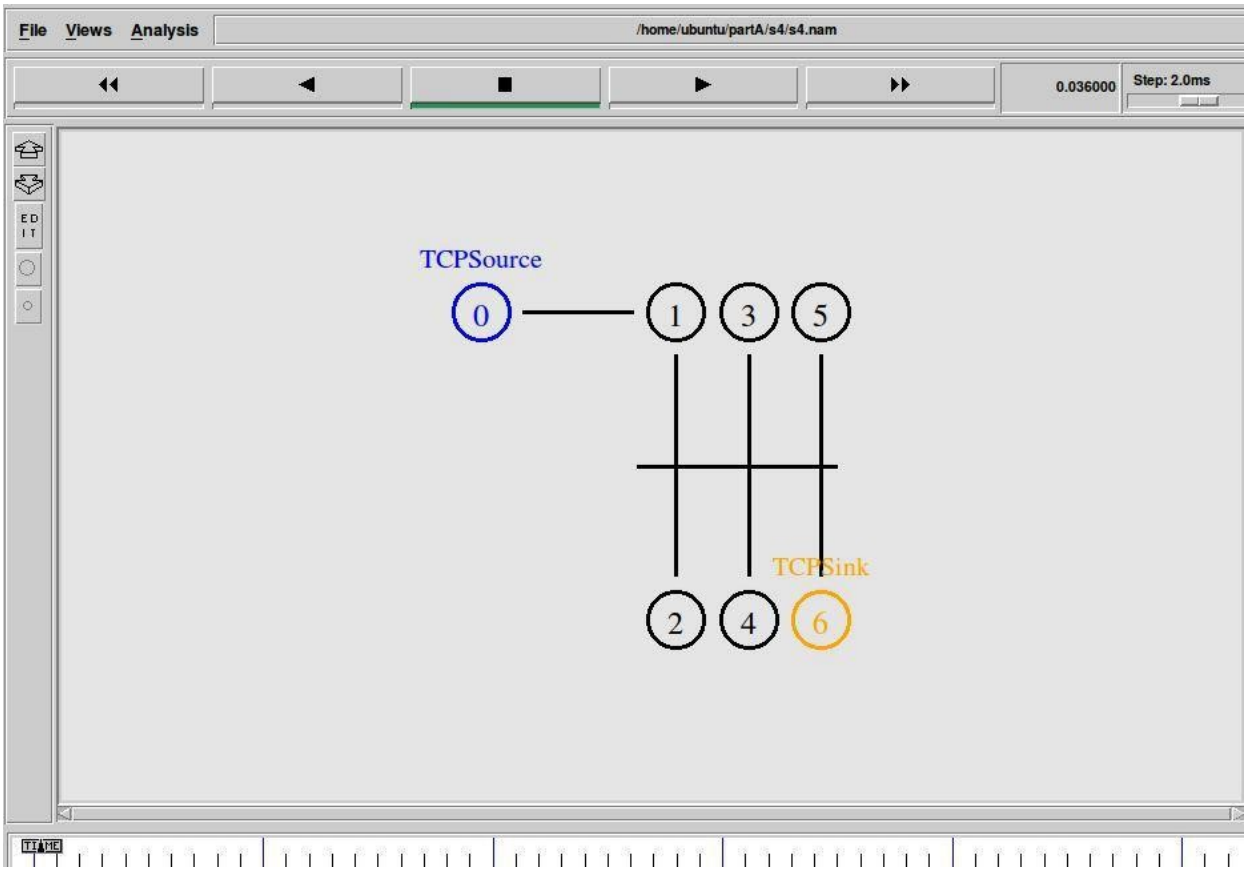
xgraph -x "Bandwidth (Mbps)" -y "No of packets sent" -t "Performance analysis" s2graph



Experiment No. 3

Ethernet LAN using N Nodes

Aim: To implement Ethernet LAN using N nodes (6-10), change error rate and data rate, and compare throughput.



s3.awk

```
BEGIN {
        sSize = 0;
        startTime = 5.0;
        stopTime = 0.1;
        Tput = 0;
}
{
        event = $1;
        time = $2;
        size = $6;

        if(event == "+")
        {
                if(time < startTime)
```

```
                {
                        startTime = time;
                }
        }
        if(event == "r")
        {
                if(time > stopTime)
                {
                        stopTime = time;
                }
                sSize += size;
        }
        Tput = (sSize / (stopTime-startTime))*(8/1000);
        printf("%f\t%.2f\n", time, Tput);
}
END {
}
```

s3.tcl

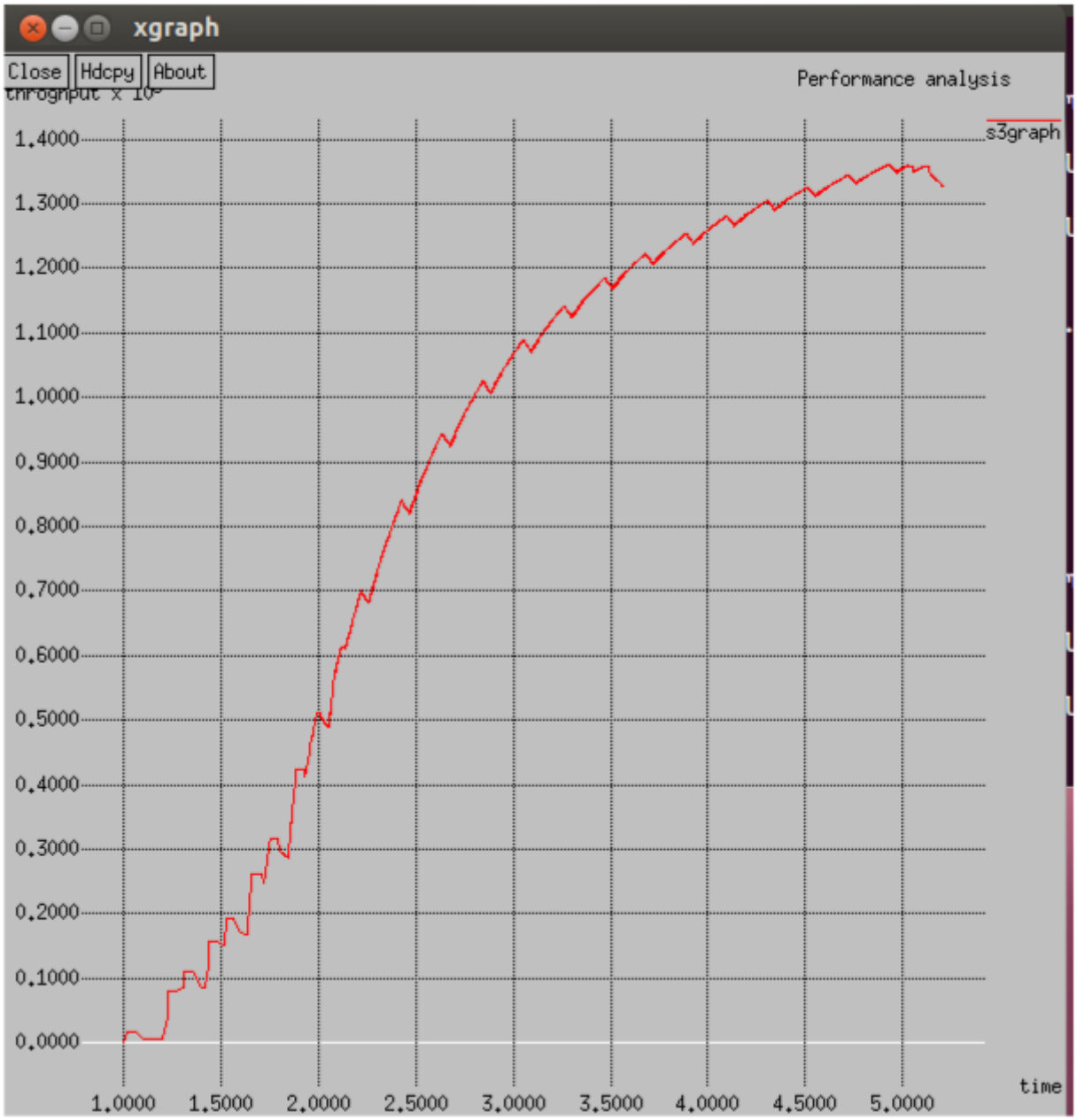
```
set ns [new Simulator]

set namfile [open s3.nam w]
$ns namtrace-all $namfile

set tracefile [open s3.tr w]
$ns trace-all $tracefile

proc finish {} {
        global ns namfile tracefile
        $ns flush-trace
        close $namfile
        close $tracefile
        exec nam s3.nam &
        exec awk -f s3.awk s3.tr  >  s3graph &
        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]

$n0 label "TCPSource"
$n6 label "TCPSink"
```

```
$ns color 1 red
$n0 color blue
$n6 color orange

$ns duplex-link $n0 $n1 3Mb 20ms DropTail
$ns make-lan "$n1 $n2 $n3 $n4 $n5 $n6" 2Mb 40ms LL Queue/DropTail Mac/802_3

$ns duplex-link-op $n0 $n1 orient right

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

$tcp set class_ 1

set tcpsink [new Agent/TCPSink]
$ns attach-agent $n6 $tcpsink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns connect $tcp $tcpsink

$ns at 1.0 "$ftp start"
$ns at 5.0 "$ftp stop"
$ns at 5.5 "finish"
$ns run
```

Steps:

gedit s3.tcl
sudo ns s3.tcl

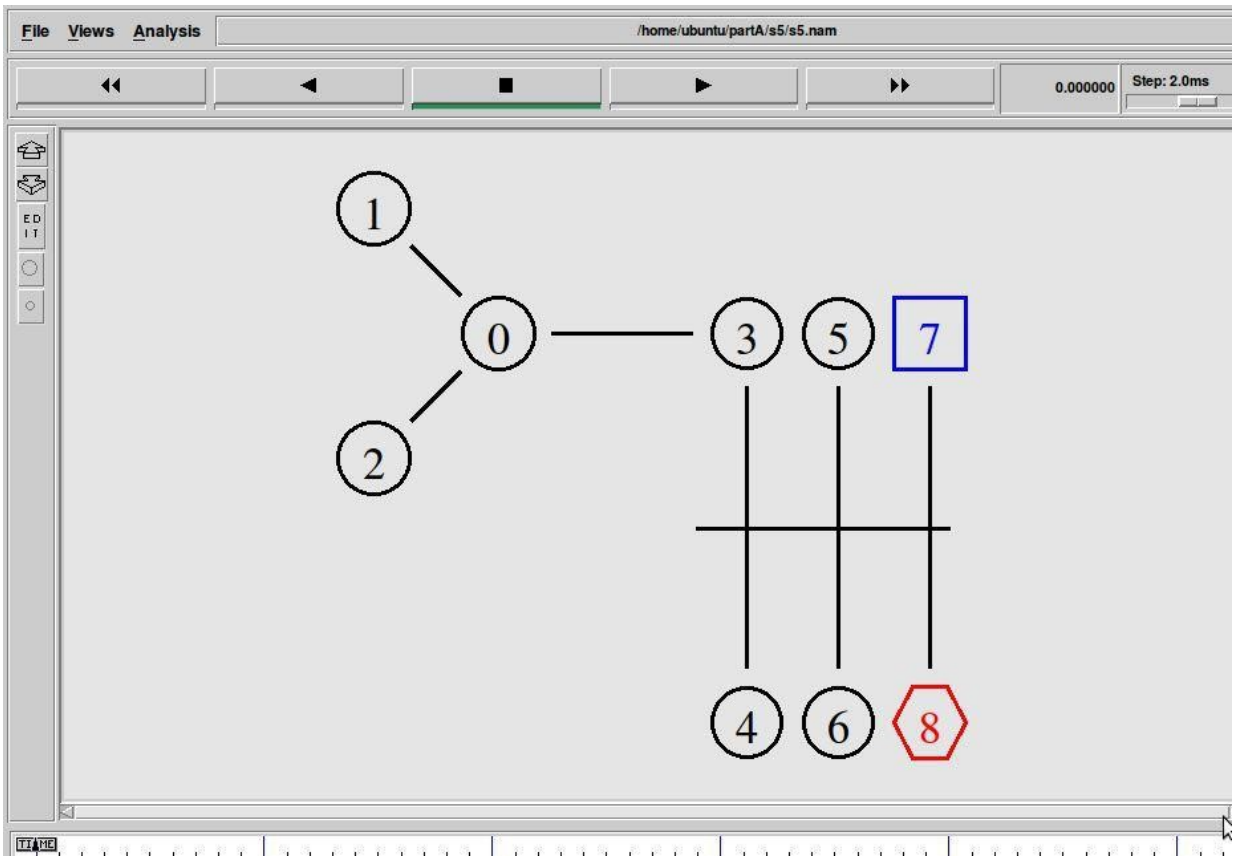
xgraph –x “Time” –y “Throughput” –t “Performance analysis” s3graph


xgraph
Close
Hdcpy
About
Performance analysis
throughput x 10^
s3graph
1.4000
1.3000
1.2000
1.1000
1.0000
0.9000
0.8000
0.7000
0.6000
0.5000
0.4000
0.3000
0.2000
0.1000
0.0000
1.0000
1.5000
2.0000
2.5000
3.0000
3.5000
4.0000
4.5000
5.0000
time

Experiment No. 4

Ethernet LAN for Multiple Traffic

Aim: To simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.



s4.awk

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

s4.tcl

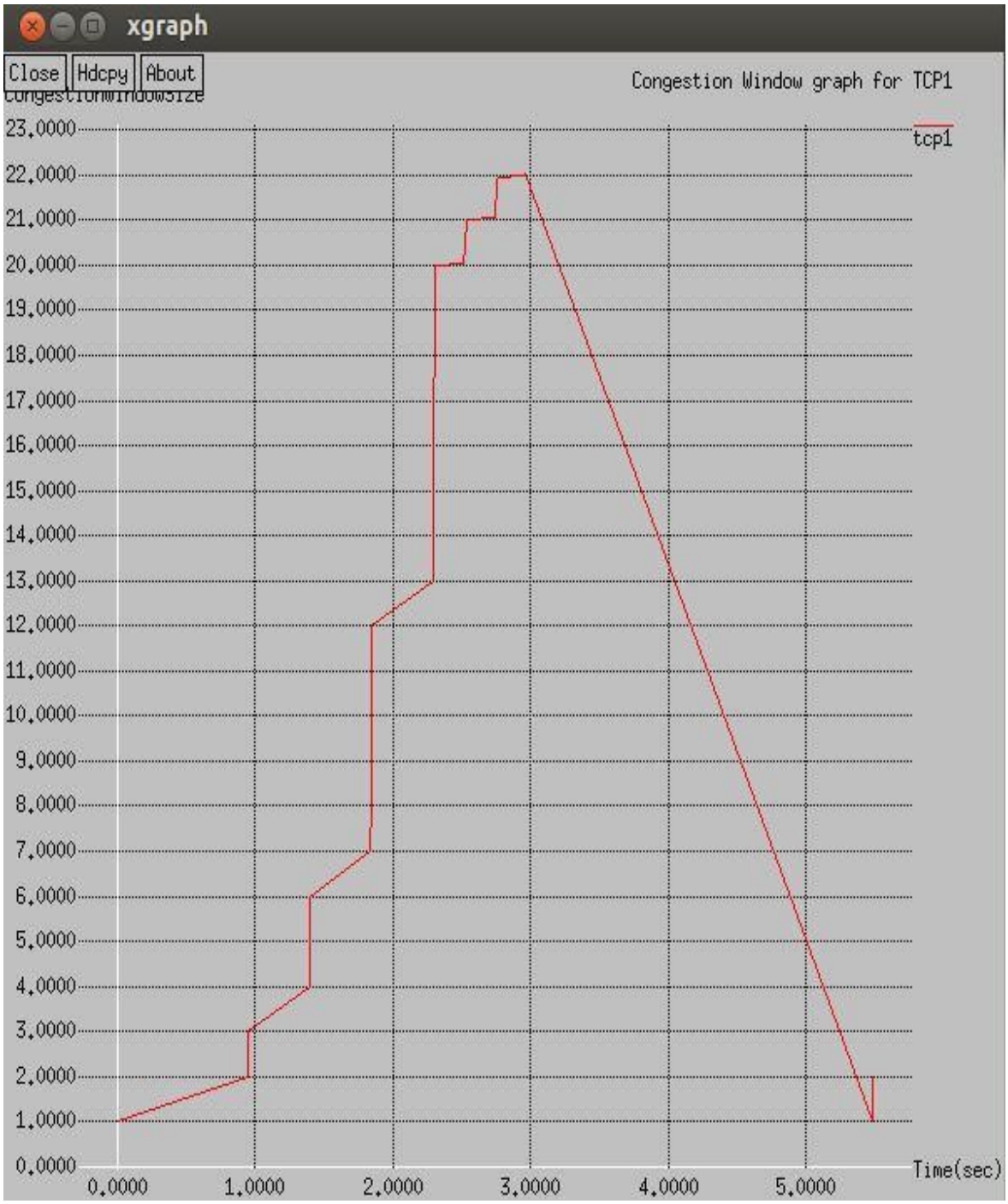
```
set ns [new Simulator]

set namfile [open s4.nam w]
$ns namtrace-all $namfile

set tracefile [open s4.tr w]
$ns trace-all $tracefile

proc finish {} {
        global ns namfile tracefile
        $ns flush-trace
        close $namfile
        close $tracefile

        exec nam s4.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 n8 [$ns node]

$ns color 1 Blue
$ns color 2 Red

$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red

$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail

$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail Mac/802_3

$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right
$ns queue-limit $n0 $n3 20
```

```
set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns connect $tcp1 $sink1

$tcp1 set class_ 1
$tcp1 set packetSize_ 55

set tfile1 [open cwnd1.tr w]
$tcp1 attach $tfile1
$tcp1 trace cwnd_

set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2

set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

$ns connect $tcp2 $sink2

$tcp2 set class_ 2
$tcp2 set packetSize_ 55

set tfile2 [open cwnd2.tr w]
$tcp2 attach $tfile2
$tcp2 trace cwnd_

$ns at 0.5 "$ftp1 start"
$ns at 1.0 "$ftp2 start"
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp1 stop"
$ns at 5.5 "finish"
$ns run
```

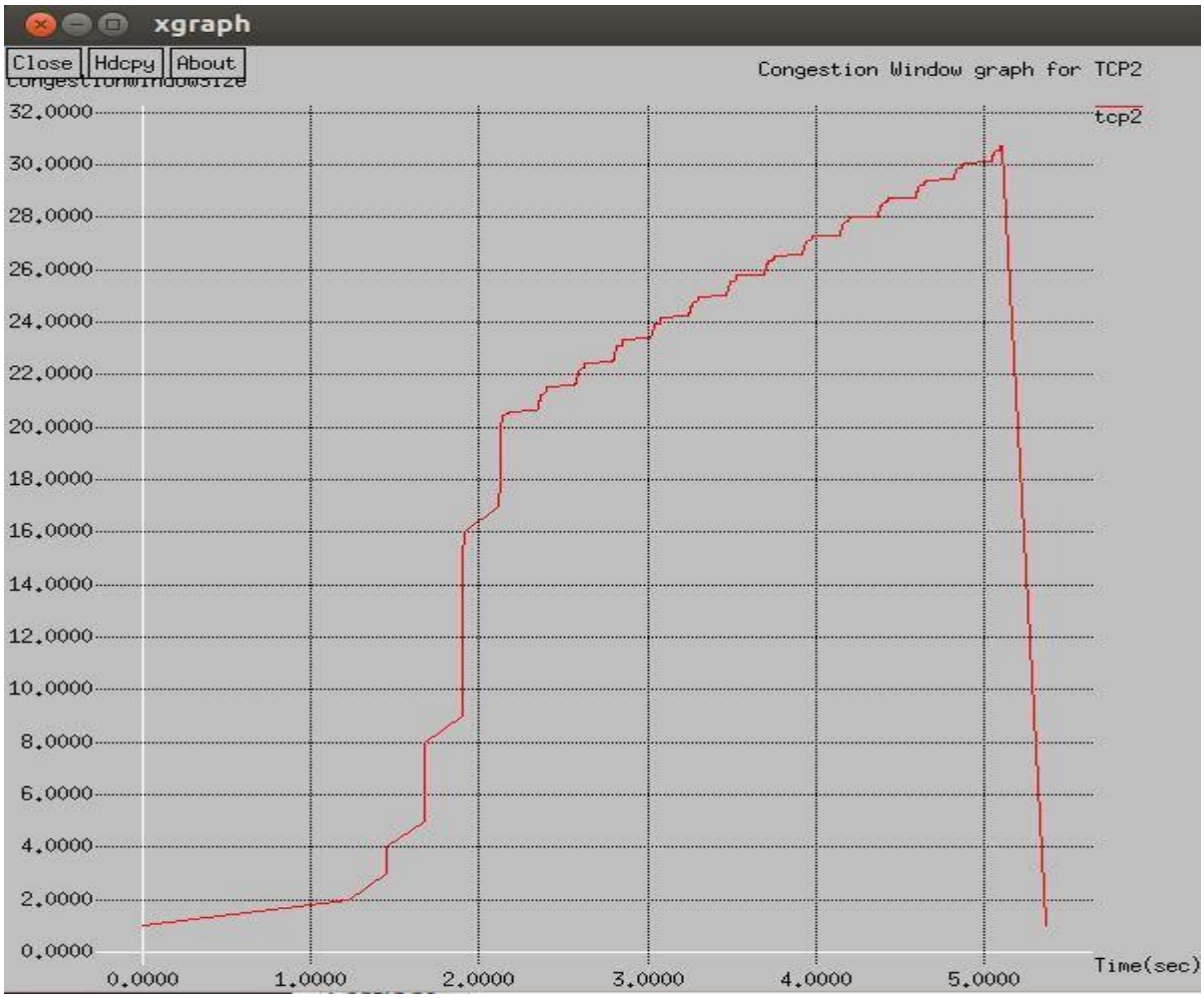
Steps:

gedit s4.tcl
sudo ns s4.tcl

awk -f s4.awk cwnd1.tr >tcp1
xgraph -x "Time(sec)" -y "Congestion Window Size" -t "Congestion Window graph for TCP1" tcp1



awk -f s4.awk cwnd2.tr>tcp2
xgraph -x "Time (sec)" -y "Congestion Window Size" -t "Congestion Window graph for TCP2" tcp2



Exercise: Implement Ethernet LAN using n nodes and assign multiple traffic nodes and plot congestion window for different source / destination.

cwnd trace file example

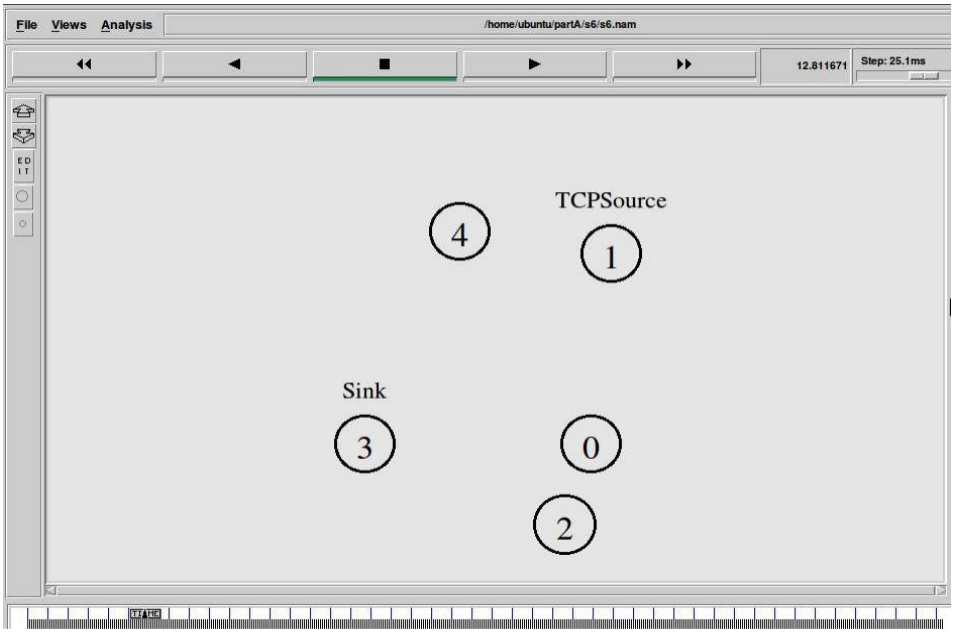
0.00000	**2**	**0**	**1**	**0**	**cwnd_**	**1.000**
(1)	(2)	(3)	(4)	(5)	(6)	(7)

1. Timestamp
2. Source node id of the flow
3. Source port id
4. Destination node id of the flow
5. Destination port id
6. Name of the variable being traced **(cwnd_ , t_seqno_ , throughput, reverse_feedback)**
7. Value of the traced variable

Experiment No. 5

Wireless LAN

Aim: To simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.



s5.awk

```
BEGIN{
        PacketRcvd = 0;
        Throughput = 0.0;
        }
        {
        if(($1 == "r")&&($3 == "_3_")&&($4 == "AGT")&&($7 == "tcp")&&($8 > 1000))
                {
                        PacketRcvd++;
                }
        }
END     {
        Throughput = ((PacketRcvd*1000*8) / (95.0*1000000));
        printf("\nThe throughput is:%f\n", Throughput); }
```

s6.tcl

```
set ns [new Simulator]
set tracefile [open s5.tr w]
$ns trace-all $tracefile
```

```
set namfile [open s5.nam w]
$ns namtrace-all-wireless $namfile 750 750

proc finish {} {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile
        exec nam s5.nam &
        exec awk -f s5.awk s5.tr &
        exit 0
}

#get the number of nodes value from the user set nn 5
#create new topography object
set topo [new Topography]
$topo load_flatgrid 750 750

#Configure the nodes
$ns     node-config     -adhocRouting      AODV \
-llType         LL \
-macType        Mac/802_11 \
-ifqType        Queue/DropTail \
-channelType Channel/WirelessChannel \
-propType       Propagation/TwoRayGround \
-antType        Antenna/OmniAntenna \
-ifqLen         50 \
-phyType        Phy/WirelessPhy \
-topoInstance           $topo \
-agentTrace   ON \
-routerTrace  ON \
-macTrace     OFF \
-movementTrace ON

#general operational descriptor storing the hop details in the network set god_ [create-god $nn]

#create mobile nodes
for {set i 0} {$i < $nn} {incr i}
{
        set n($i) [$ns node]
}

#label node
$n(1) label "TCPSource"
$n(3) label "Sink"
```

```
#Randomly placing the nodes
for {set i 0} {$i < $nn} {incr i} {
        set XX [expr rand()*750]
        set YY [expr rand()*750]
        $n($i) set X_ $XX
        $n($i) set Y_ $YY
}

#define the initial position for the nodes
for {set i 0} {$i < $nn} {incr i} {
        $ns initial_node_pos $n($i) 100
}

#define the destination procedure to set the destination to each
node proc destination {} {
        global ns nn
        set now [$ns now]
        set time 5.0
        for {set i 0} {$i < $nn} {incr i}
        {
                set XX [expr rand()*750]
                set YY [expr rand()*750]
                $ns at [expr $now + $time] "$n($i) setdest $XX $YY 20.0"
        }
        $ns at [expr $now + $time] "destination"
}
set tcp [new Agent/TCP]
$ns attach-agent $n(1) $tcp

set ftp [new Application/FTP]
$ftp attach-agent $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n(3) $sink

$ns connect $tcp $sink

$ns at 0.0 "destination"
$ns at 1.0 "$ftp start"
$ns at 100 "finish"
$ns run
```

Steps:

gedit s5.tcl

sudo ns s5.tcl

Output:

The throughput is: 0.579368

Experiment No. 6

Link State Routing Algorithm

Aim: To implement Link state routing algorithm

```
set ns [new Simulator]
set namfile [open s6.nam w]
$ns namtrace-all $namfile

set tracefile [open s6.tr w]
$ns trace-all $tracefile

proc finish {} {
        global ns namfile tracefile
        $ns flush-trace
        close $namfile
        close $tracefile
        exec nam s6.nam &
        exit 0
        }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$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 $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n4 1Mb 10ms DropTail
$ns duplex-link $n2 $n4 1Mb 10ms DropTail

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

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $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 $n4 $null0
$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n2 $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 $n4 $null0
$ns connect $udp1 $null0

$ns rtproto LS

$ns rtmodel-at 20.0 down $n1 $n4
$ns rtmodel-at 20.0 up $n1 $n4
$ns rtmodel-at 25.0 down $n2 $n4
$ns rtmodel-at 40.0 up $n2 $n4
$udp0 set class_ 1
$udp1 set class_ 2
$ns color 1 Red
$ns color 2 Green

$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"
$ns run
```

Output

gedit s6.tcl

sudo ns s6.tcl