Prashanth Molakala
A20322954

ECE541

Performance Evaluation of Computer Networks

Course Project

NS2 Simulation with Poisson Traffic

Contents:

Prashanth Molakala
A20322954

Install Poisson Traffic Generator:

Successfully installed the Poisson Traffic generator by following all the instructions posted on the blackboard.

Had downloaded the poisson_test.tcl script from blackboard and ran it:

Script for poisson_test.tcl:

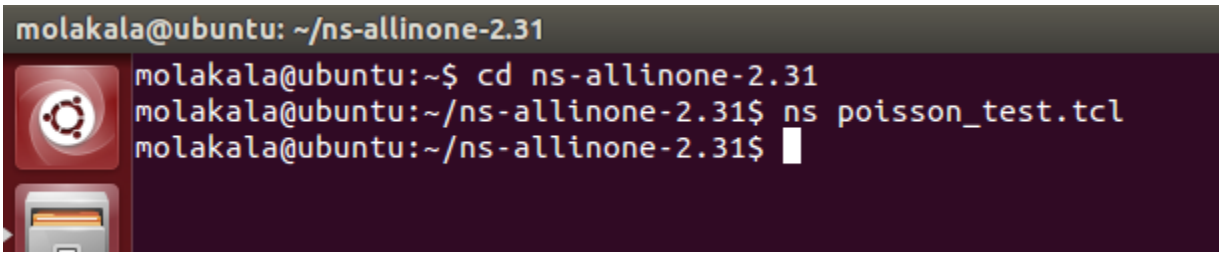
```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

set nd [open out.tr w]
$ns trace-all $nd

#Define a 'finish' procedure
proc finish {} {
        global ns nf nd
        $ns flush-trace
        #Close the trace file
```

```
        close $nf
        close $nd
        #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ

$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

#Monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Create a UDP agent and attach it to node n0
```

```
set udp0 [new Agent/UDP]
$udp0 set packetSize_ 1500
$udp0 set class_ 1
$ns attach-agent $n0 $udp0

# Create a Poisson traffic source and attach it to udp0
set Poi0 [new Application/Traffic/Poisson]
$Poi0 set packetSize_ 1500
$Poi0 set rate_ 1Mb
$Poi0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

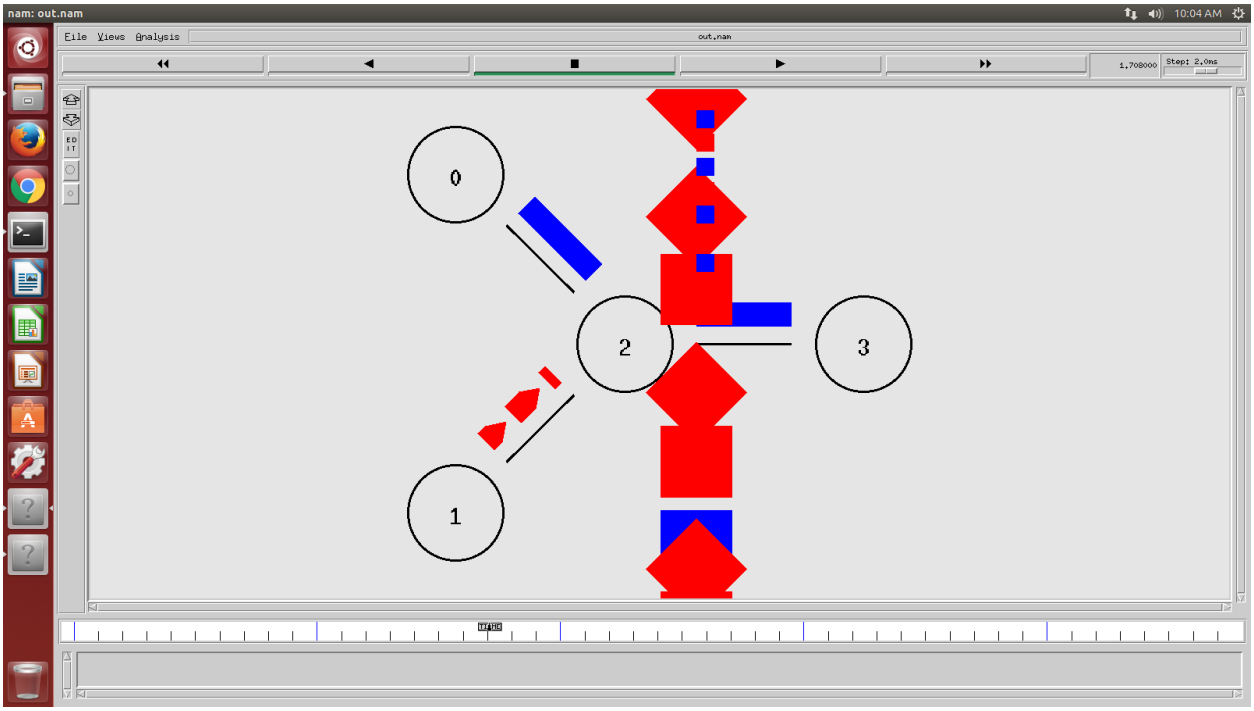
```
#Connect the traffic sources with the traffic sink

$ns connect $udp0 $null0

$ns connect $udp1 $null0


#Schedule events for the CBR agents

$ns at 0.5 "$Poi0 start"

$ns at 1.0 "$cbr1 start"

$ns at 4.0 "$cbr1 stop"

$ns at 4.5 "$Poi0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"


#Run the simulation

$ns run
```

Terminal command screenshot:



Prashanth Molakala
A20322954

Nam output (out.nam) screenshot for simulation of poisson_test.tcl:



NS2 Simulation Experiments:

1. Experiment 1: Throughput and loss rate under constant-rate traffic:

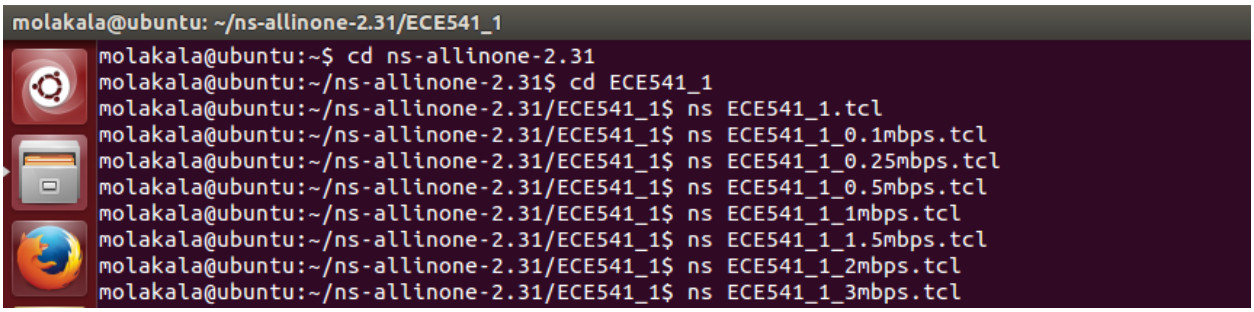
TCL script:

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open ECE541_1_a.nam w]
$ns namtrace-all $nf

set nd [open ECE541_1.tr w]
$ns trace-all $nd

#Define a 'finish' procedure
proc finish {} {
        global ns nf nd
        $ns flush-trace
        #Close the trace file
        close $nf
        close $nd
        #Execute nam on the trace file
        exec nam ECE541_1_a.nam &
```

```
        exit 0
}

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n1 queuePos 0.5


#Create a UDP agent and attach it to node n0
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

# Create a CBR traffic source and attach it to udp
set cbr [new Application/Traffic/CBR]
$cbr set rate_ cbrrate
$cbr attach-agent $udp

#Create a Null agent (a traffic sink) and attach it to node n1
set null [new Agent/Null]
$ns attach-agent $n1 $null

#Connect the traffic sources with the traffic sink
```

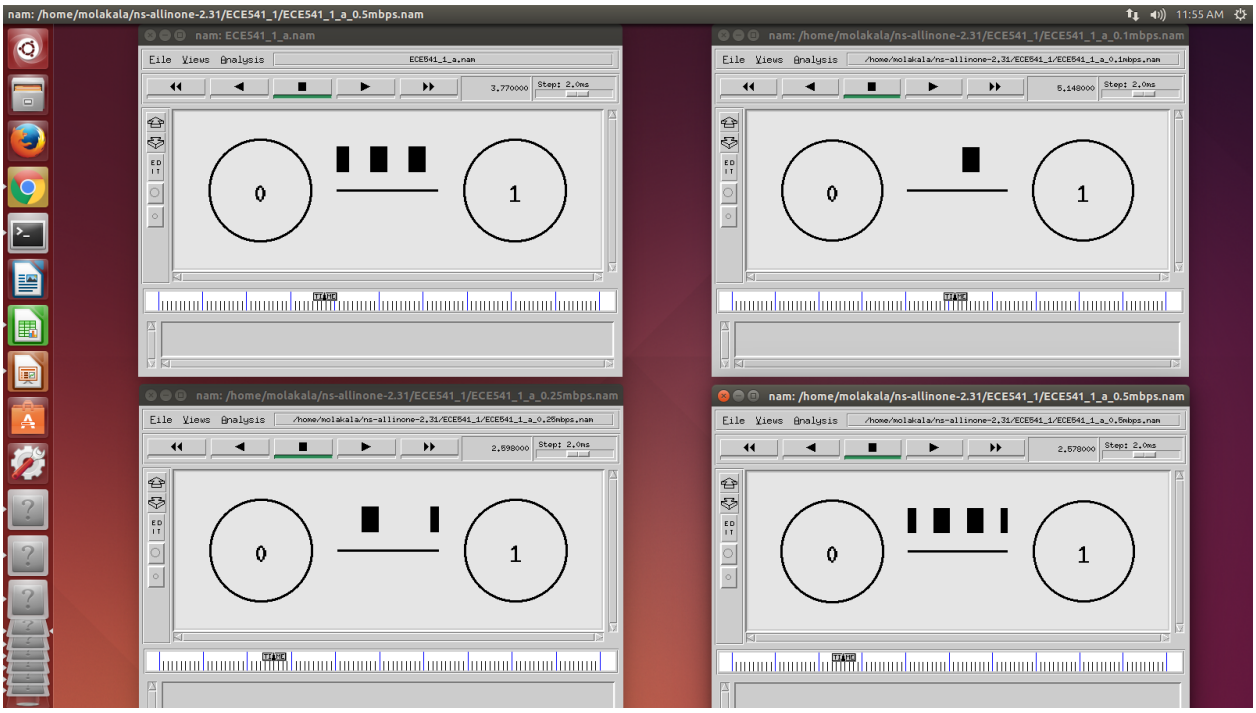
```
$ns connect $udp $null


#Schedule events for the CBR agents

$ns at 0 "$cbr start"

$ns at 10 "$cbr stop"


#Call the finish procedure after 5 seconds of simulation time

$ns at 10.1 "finish"


#Run the simulation

$ns run
```

Terminal commands screenshot:

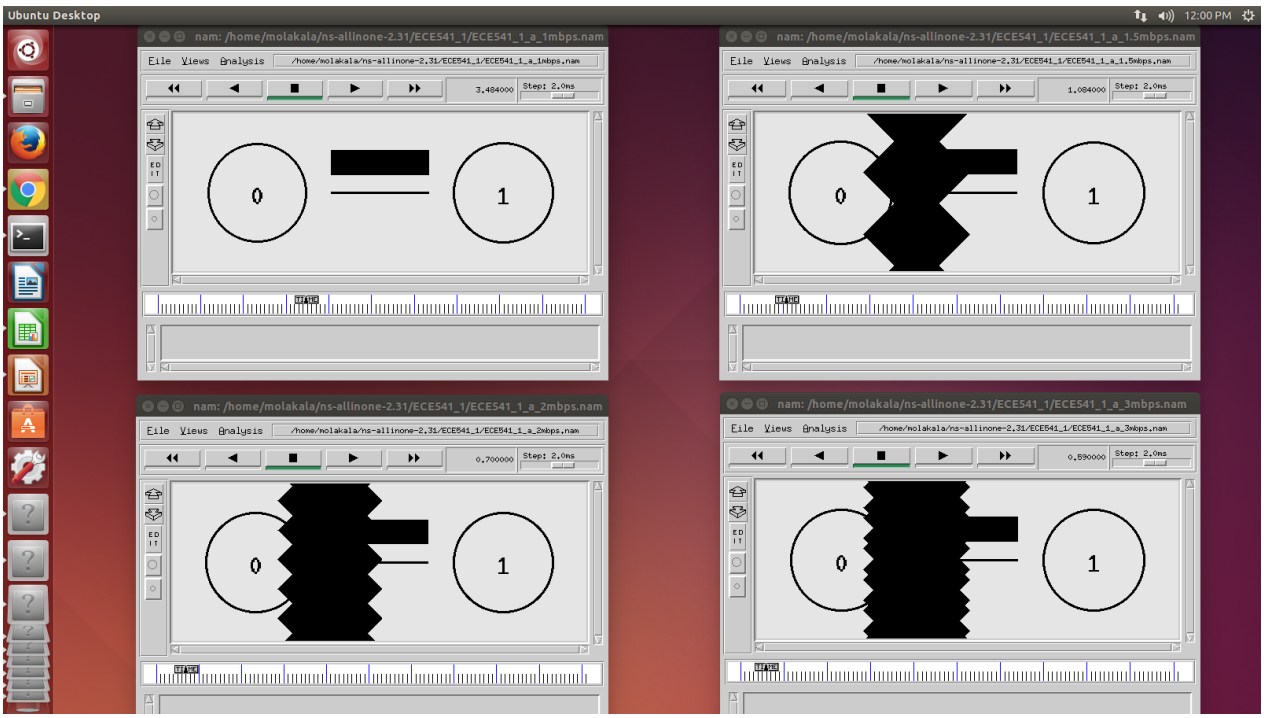


Nam output screenshots:

Nam output for cbr traffic rates: default cbr traffic rate, 0.1Mbps, 0.25Mbps and 0.5Mbps:

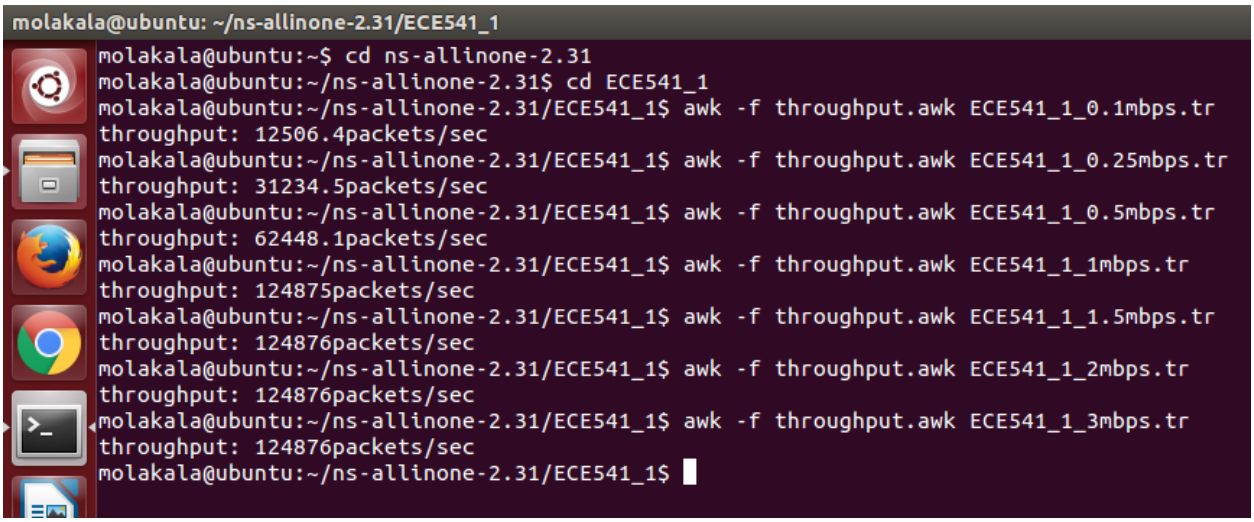


Nam output for cbr traffic rates: 1Mbps, 1.5Mbps, 2Mbps and 3Mbps:



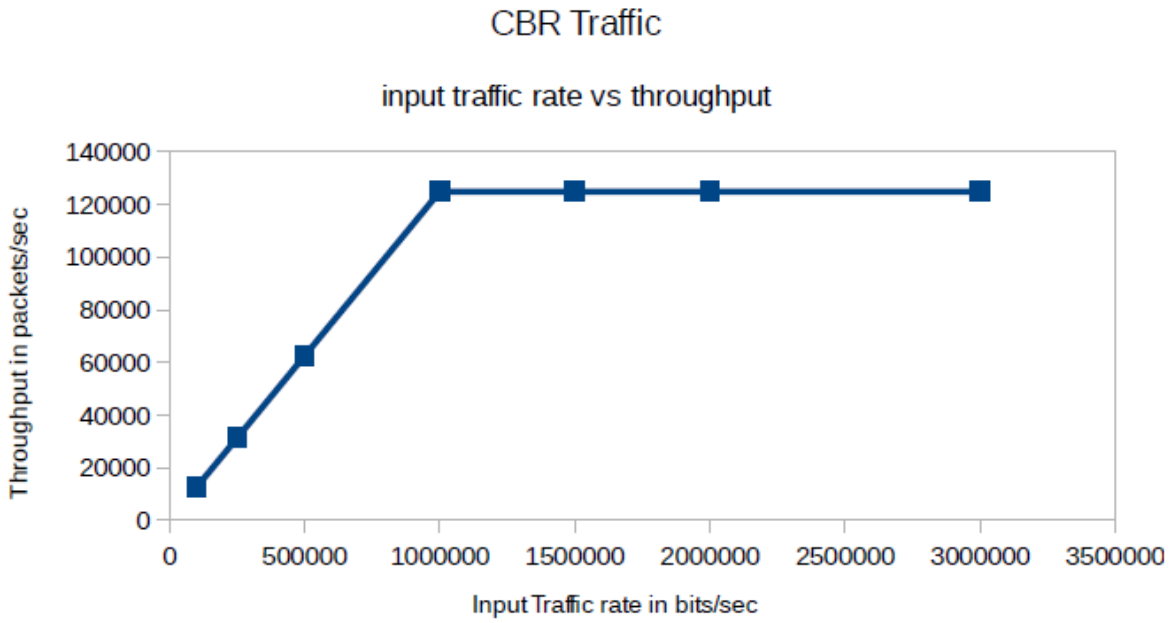
Graph between Input cbr traffic rate vs throughput:

Throughput values calculated using throughput.awk:



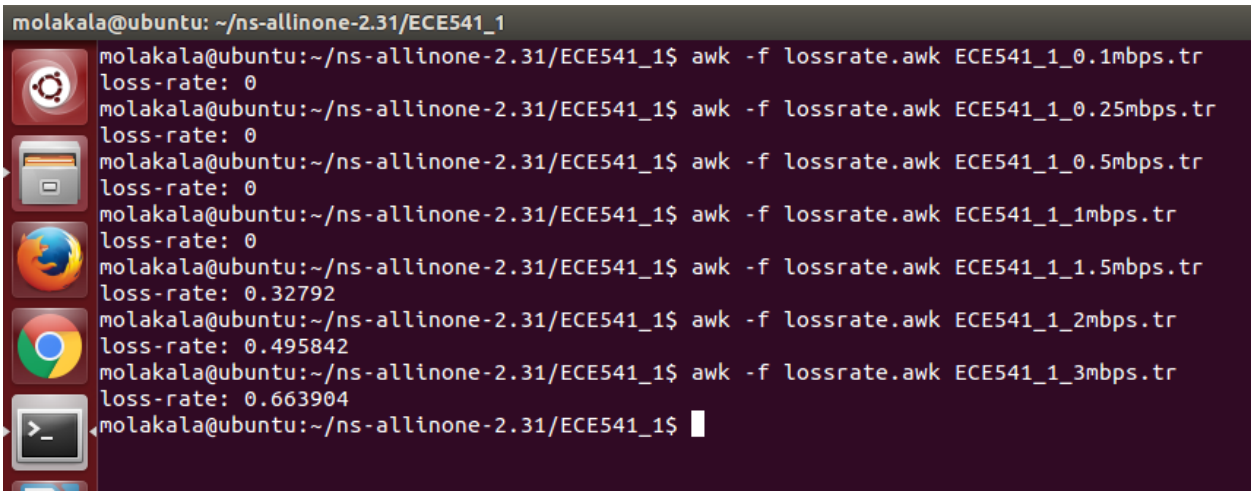
Graph between input cbr traffic rate vs throughput:

Input Traffic Rate (bits/sec)	Throughput (packets/sec)
100000	12506.4
250000	31234.5
500000	62448.1
1000000	124875
1500000	124876
2000000	124876
3000000	124876



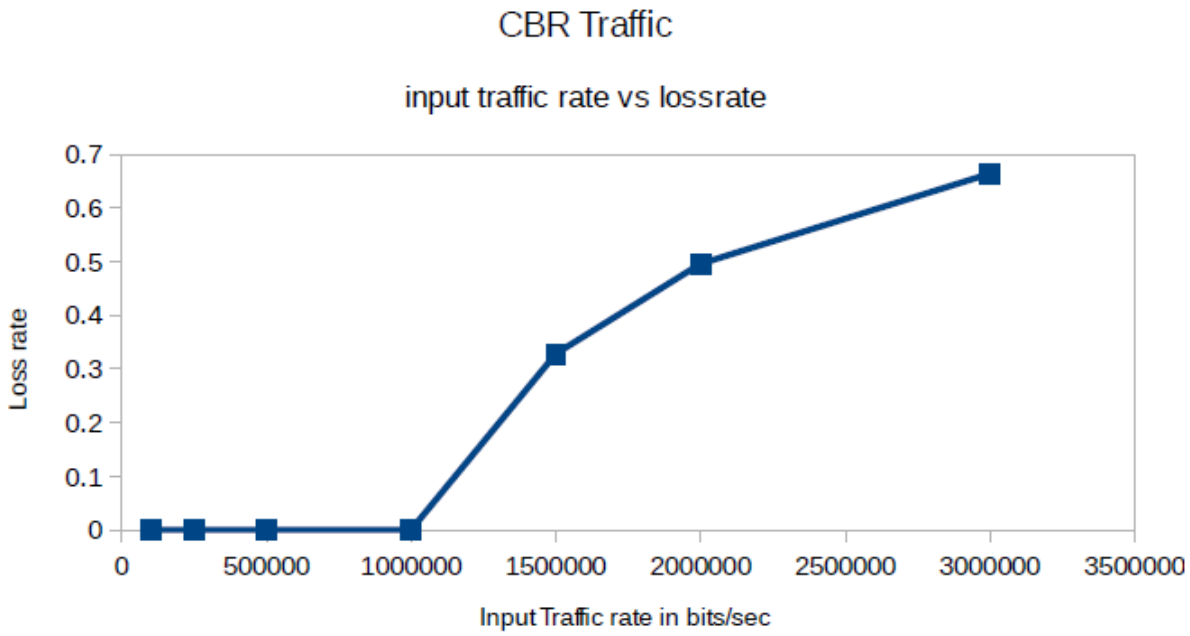
Graph between Input cbr traffic rate vs loss-rate:

Loss-rate values calculated using lossrate.awk:



Graph between input cbr traffic rate vs loss-rate:

Input Traffic Rate (bits/sec)	Loss rate
100000	0
250000	0
500000	0
1000000	0
1500000	0.32792
2000000	0.495842
3000000	0.663904



Analysis:

The source node n0 is sending packets to the destination node n1 at different CBR traffic rates of 0.1Mbps, 0.25Mbps, 0.5Mbps, 1Mbps, 1.5Mbps, 2Mbps and 3Mbps. For CBR traffic rate less than 1.5Mbps there is no loss of packets and vice versa. With increase in the CBR traffic rate the number of packets sent from the source node would increase and thus the throughput of the system increases. Throughput depends on the number of packets received at the destination node n1, thus with increase in CBR traffic rate the throughput increases linearly and after the throughput reaches its maximum value it remains constant. Loss-rate depends on the number of packets dropped in the system, thus until 1.5Mbps Poisson traffic rate the loss-rate equals zero and for Poisson rate greater than 1.5Mbps the loss-rate increases.

Throughput:

With increase in the CBR input traffic rate throughput of the system increases linearly until it reaches the maximum throughput after which throughput of the system remains constant even with increase in the CBR input traffic rate.

Loss-rate:

Loss-rate of the system remains zero until the CBR input traffic reaches a maximum threshold rate where the packets start dropping due to insufficient Queue size of the system. After this maximum threshold rate the loss-rate increases with increase in the CBR input traffic rate.

2. Experiment 2: Throughput and loss rate under Poisson traffic:

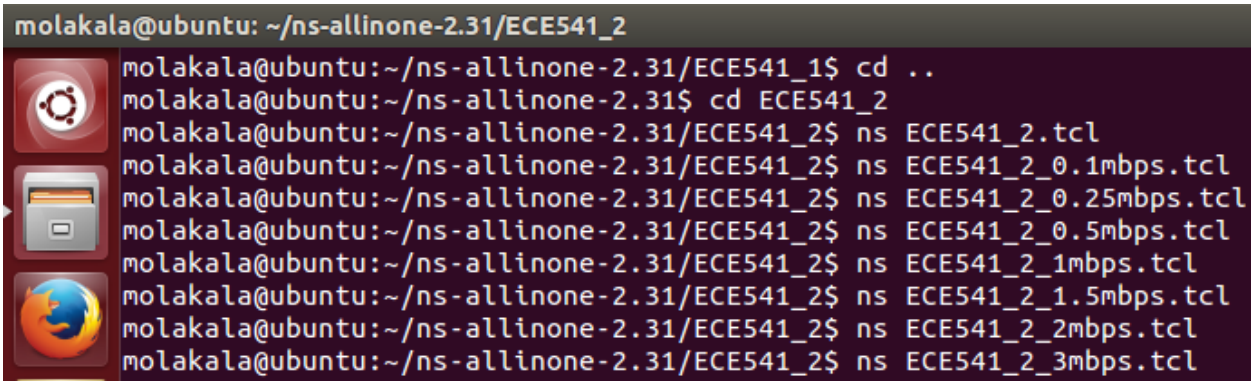
TCL Script:

```
#Create a simulator object
set ns [new Simulator]


#Open the nam trace file
set nf [open ECE541_2_a.nam w]
$ns namtrace-all $nf


set nd [open ECE541_2.tr w]
$ns trace-all $nd


#Define a 'finish' procedure
proc finish {} {
        global ns nf nd
        $ns flush-trace
        #Close the trace file
```

```
        close $nf
        close $nd
        #Execute nam on the trace file
        exec nam ECE541_2_a.nam &
        exit 0
}

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n1 queuePos 0.5

#Create a UDP agent and attach it to node n0
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

# Create a Poisson traffic source and attach it to udp
set Poi [new Application/Traffic/Poisson]
$Poi set rate_ poisson_rate
$Poi attach-agent $udp
```

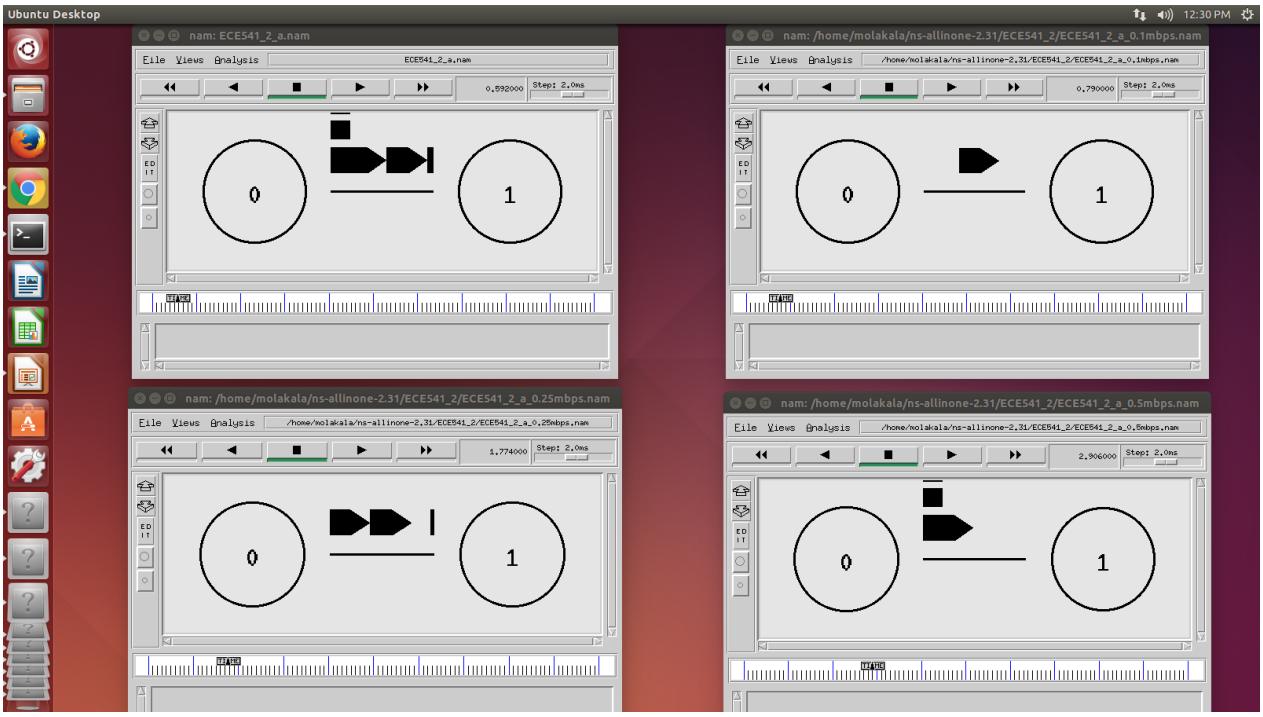
```
#Create a Null agent (a traffic sink) and attach it to node n1

set null [new Agent/Null]

$ns attach-agent $n1 $null


#Connect the traffic sources with the traffic sink

$ns connect $udp $null


#Schedule events for the CBR agents

$ns at 0 "$Poi start"

$ns at 10 "$Poi stop"


#Call the finish procedure after 5 seconds of simulation time

$ns at 10.1 "finish"


#Run the simulation

$ns run
```

Terminal commands screenshot:

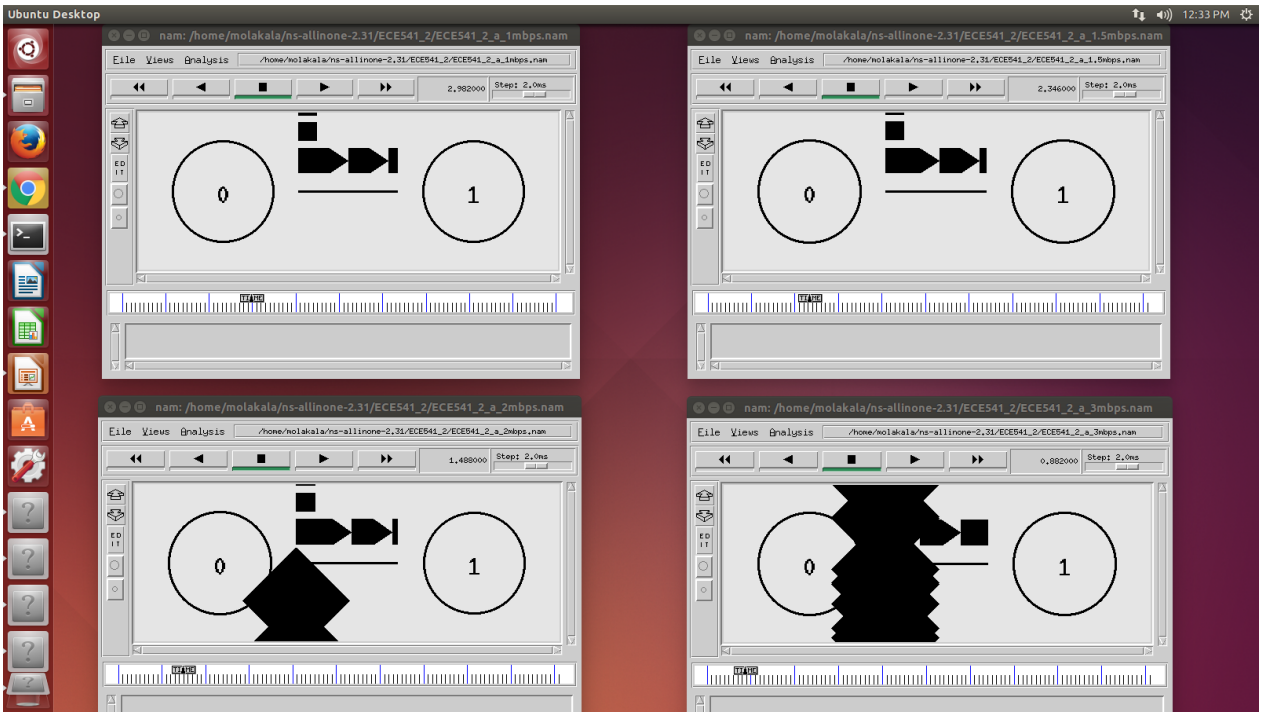


Nam output screenshots:

Nam output for Poisson input traffic rates: default, 0.1Mbps, 0.25Mbps and 0.5Mbps:

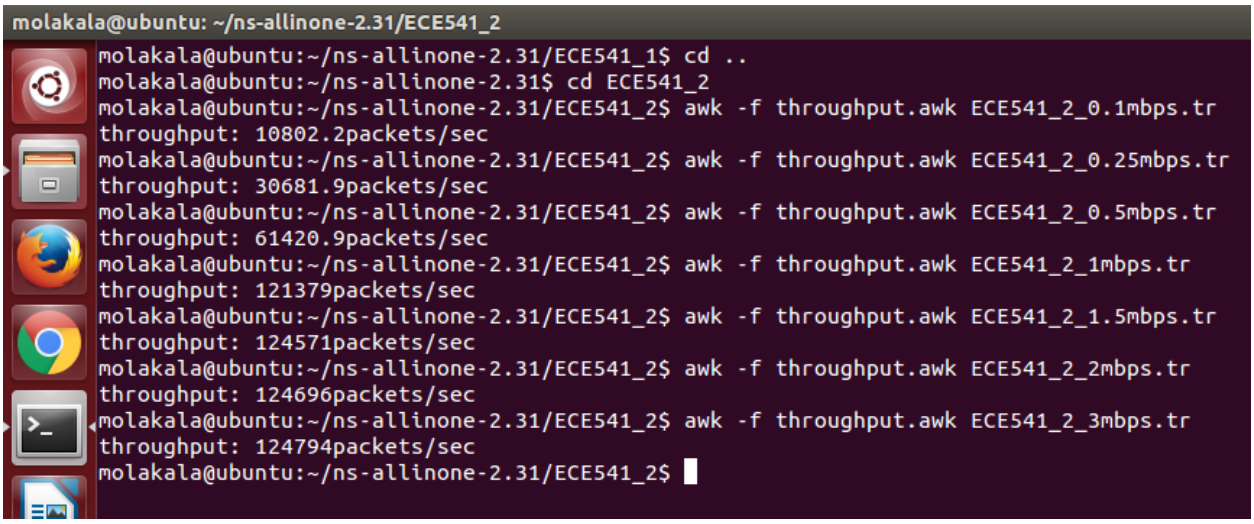


Nam output for Poisson input traffic rates: 1Mbps, 1.5Mbps, 2Mbps and 3Mbps:



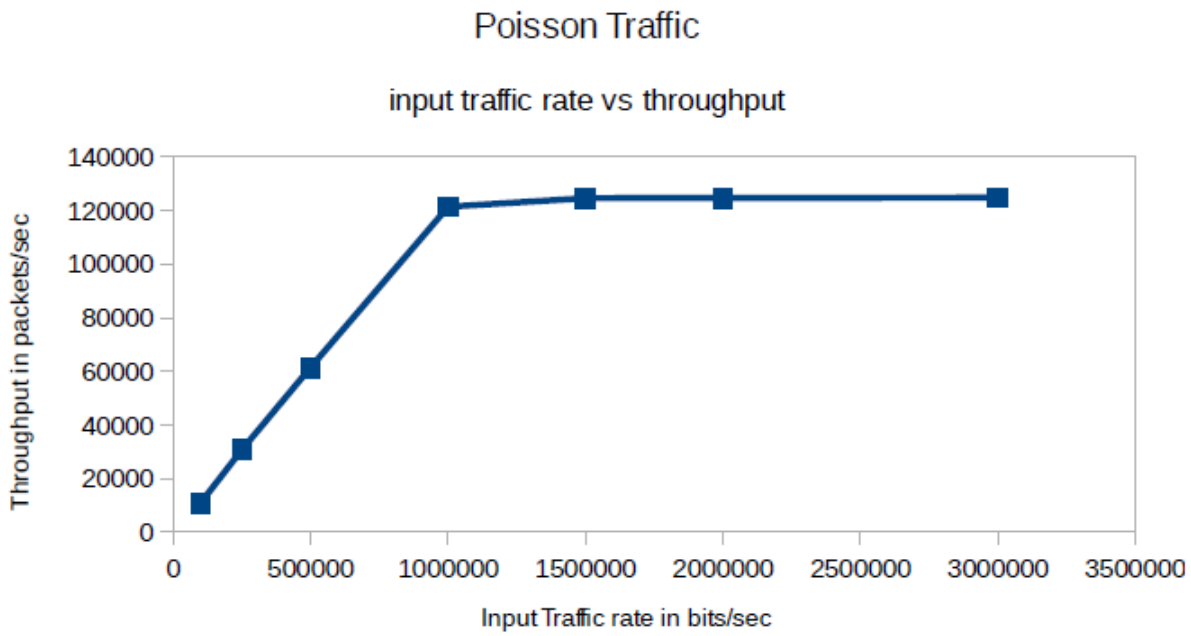
Graph between Poisson input traffic rate vs throughput:

Throughput values calculated using throughput.awk:



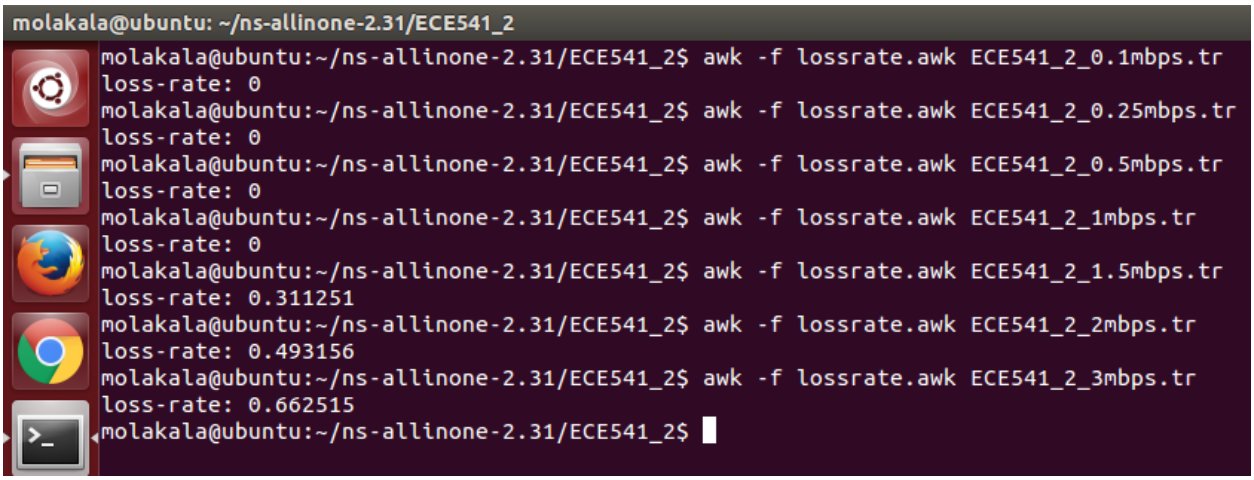
Graph between poisson input traffic rate vs throughput:

Input Traffic Rate (bits/sec)	Throughput (packets/sec)
100000	10802.2
250000	30681.9
500000	61420.9
1000000	121379
1500000	124571
2000000	124696
3000000	124794



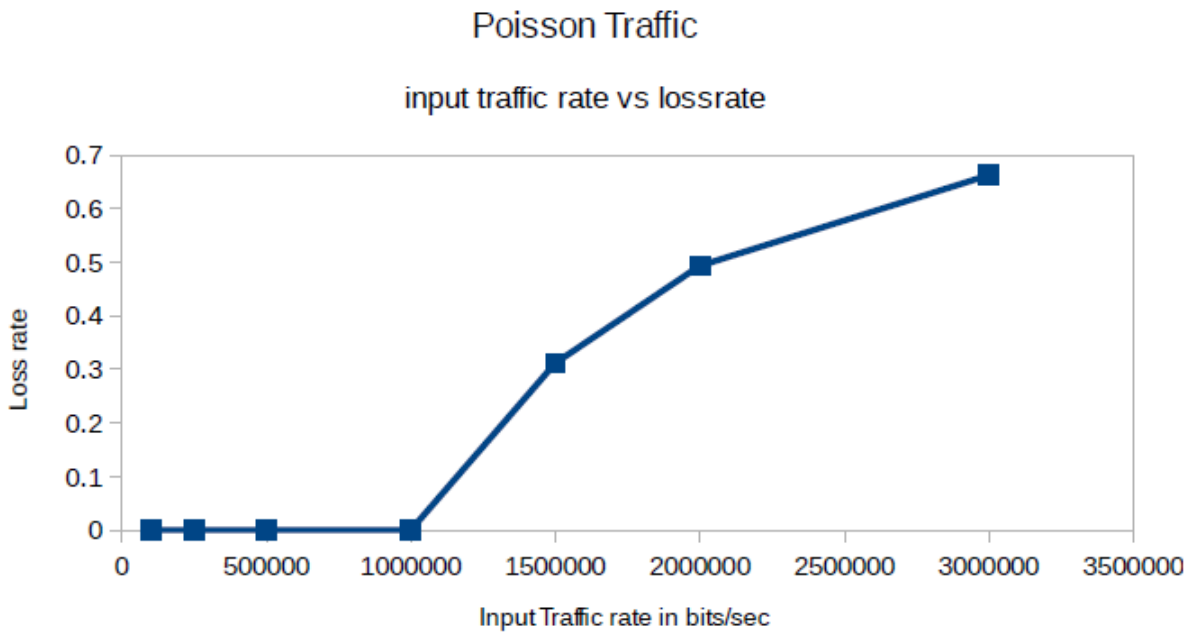
Graph between Poisson input traffic rate vs loss-rate:

Loss-rate values calculated using lossrate.awk:



Graph between poisson input traffic rate vs loss-rate:

Input Traffic Rate (bits/sec)	Loss rate
100000	0
250000	0
500000	0
1000000	0
1500000	0.311251
2000000	0.493156
3000000	0.662515



Analysis:

The source node n0 is sending packets to the destination node n1 at different Poisson traffic rates of 0.1Mbps, 0.25Mbps, 0.5Mbps, 1Mbps, 1.5Mbps, 2Mbps and 3Mbps. For Poison traffic rate less than 1.5Mbps there is no loss of packets and vice versa. With increase in the Poisson traffic rate the number of packets sent from the source node would increase and thus the throughput of the system increases. Throughput depends on the number of packets received at the destination node n1, thus with increase in Poisson traffic rate the throughput increases linearly and after the throughput reaches a particular value almost equal to the max value it increases at a very small rate or almost remains constant. Loss-rate depends on the number of packets dropped in the system, thus until 1.5Mbps Poisson traffic rate the loss-rate equals zero and for Poisson rate greater than 1.5Mbps the loss-rate increases.

Throughput:

With increase in the Poisson input traffic rate throughput of the system increases linearly until it reaches the a throughput value after which throughput of the system increases at a very small rate almost constant with increase in the Poisson input traffic rate.

Loss-rate:

Loss-rate of the system remains zero until the Poisson input traffic reaches a maximum threshold rate where the packets start dropping due to insufficient Queue size of the system. After this maximum threshold rate the loss-rate increases with increase in the Poisson input traffic rate.

3. Experiment 3: Queuing effect under Poisson traffic:

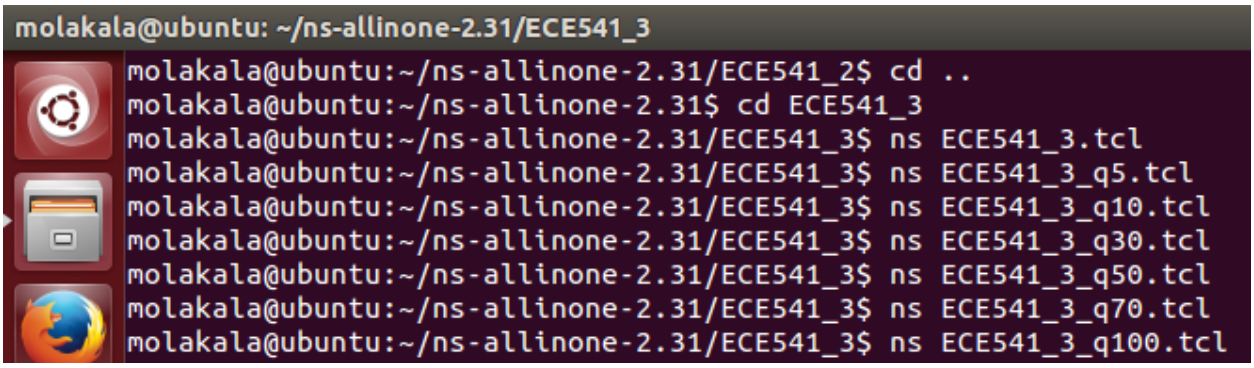
TCL Script:

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open ECE541_3_a.nam w]
$ns namtrace-all $nf

set nd [open ECE541_3.tr w]
$ns trace-all $nd

#Define a 'finish' procedure
proc finish {} {
        global ns nf nd
        $ns flush-trace
```

```
        #Close the trace file
        close $nf
        close $nd
        #Execute nam on the trace file
        exec nam ECE541_3_a.nam &
        exit 0
}

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns queue-limit $n0 $n1 queue_size
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n1 queuePos 0.5

#Create a UDP agent and attach it to node n0
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

# Create a Poisson traffic source and attach it to udp
set Poi [new Application/Traffic/Poisson]
$Poi set rate_ 0.95mbps
$Poi attach-agent $udp
```

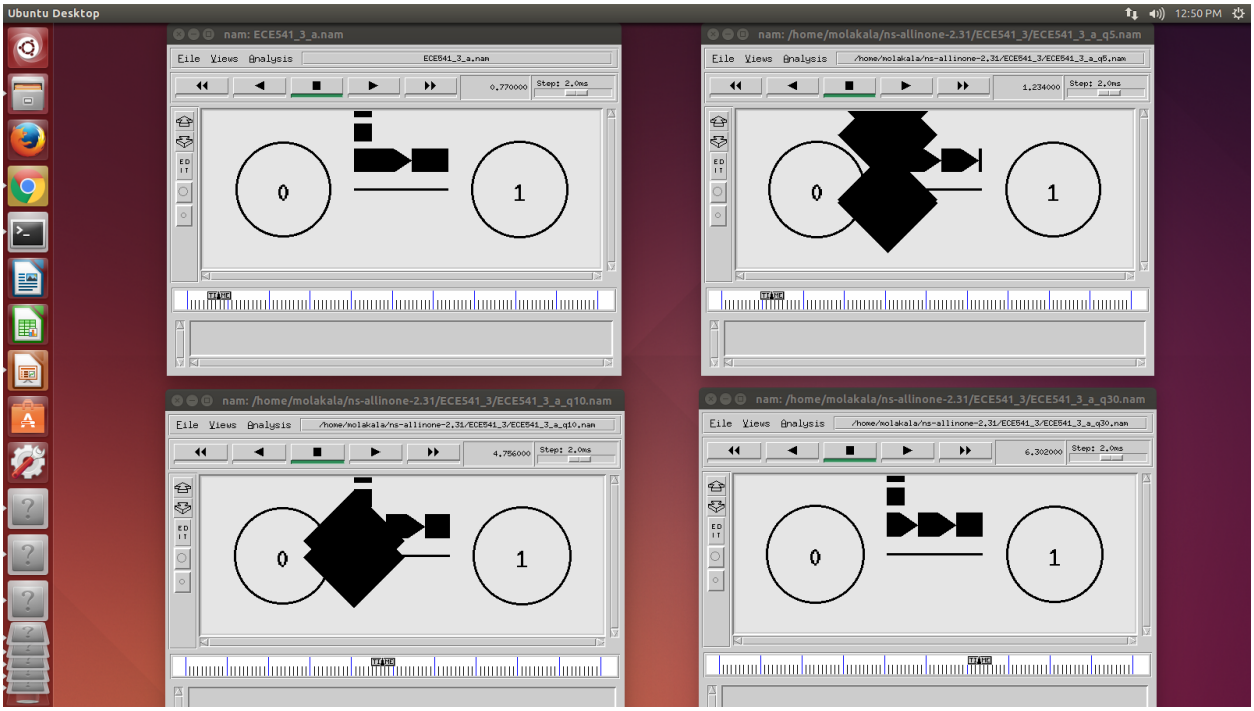
```
#Create a Null agent (a traffic sink) and attach it to node n1

set null [new Agent/Null]

$ns attach-agent $n1 $null


#Connect the traffic sources with the traffic sink

$ns connect $udp $null


#Schedule events for the CBR agents

$ns at 0 "$Poi start"

$ns at 10 "$Poi stop"


#Call the finish procedure after 5 seconds of simulation time

$ns at 10.1 "finish"


#Run the simulation

$ns run
```

Terminal commands screenshot:

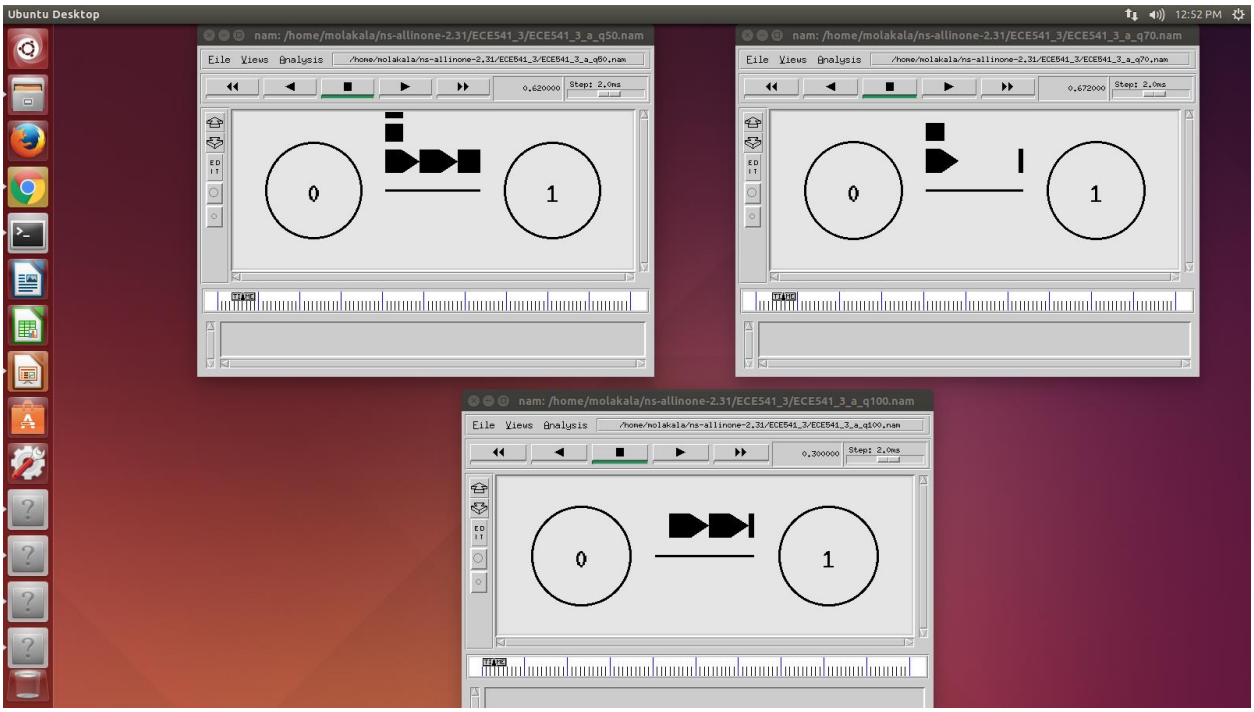


Nam output Screenshots:

Nam output for Poisson traffic rate 0.95Mbps and Queue sizes: default, 5packets, 10packets and 30packets:

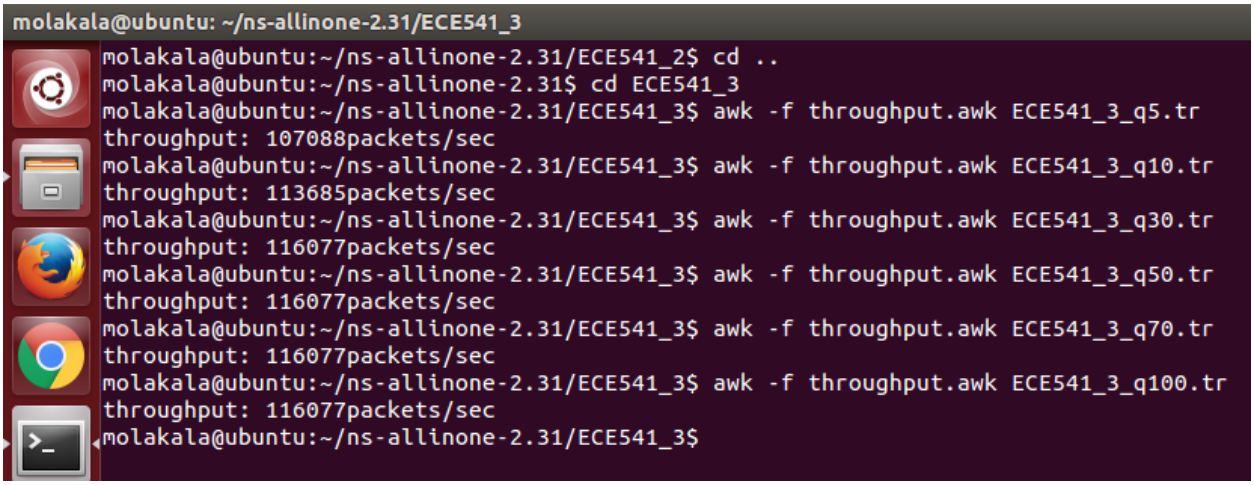


Nam output for Poisson traffic rate 0.95Mbps and Queue sizes: 50packets, 70packets and 100packets:



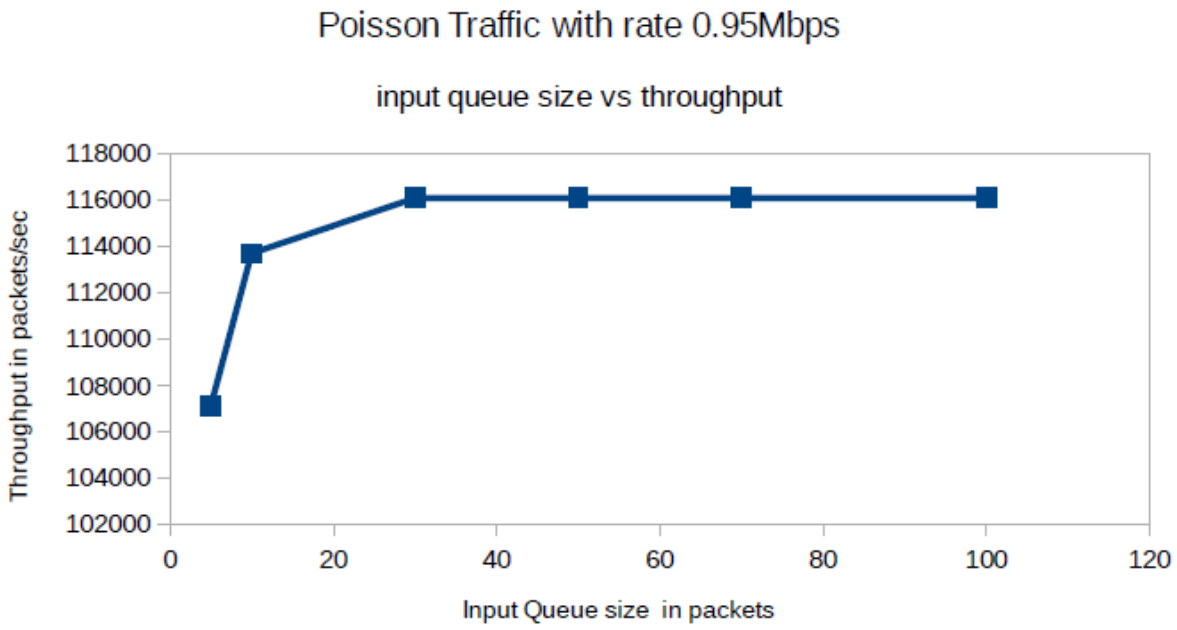
Graph between input Queue size vs Throughput:

Throughput values calculated using throughput.awk:



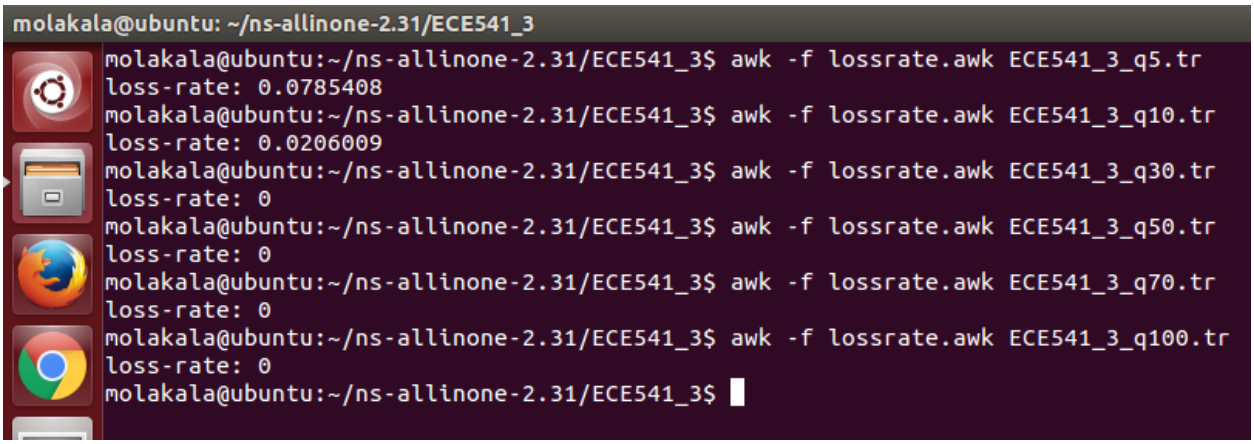
Graph between input queue size vs throughput with Poisson traffic rate 0.95Mbps:

Input Queue size (packets)	Throughput (packets/sec)
5	107088
10	113685
30	116077
50	116077
70	116077
100	116077



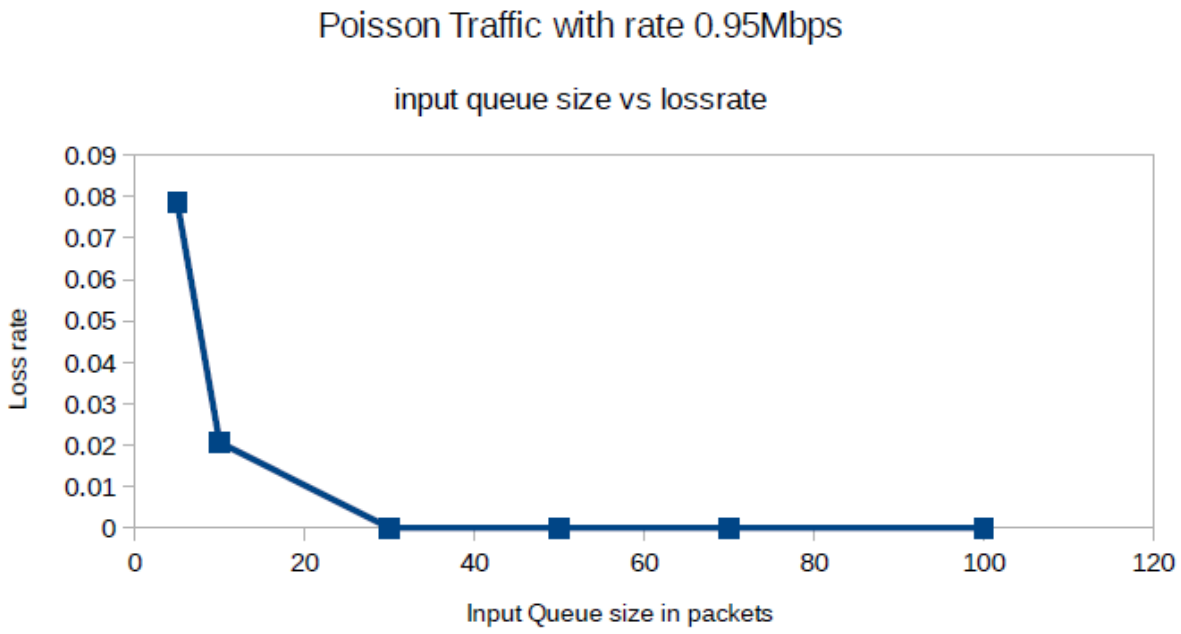
Graph between input Queue size vs Loss-rate:

Loss-rate values calculated using lossrate.awk:



Graph between input queue size vs loss-rate with Poisson traffic rate 0.95Mbps:

Input Queue size (packets)	Loss rate
5	0.0785408
10	0.0206009
30	0
50	0
70	0
100	0



Analysis:

The source node n0 is sending packets to the destination node n1 at a constant Poisson traffic rate of 0.95Mbps, for queue size less than 30packets, there will be loss of packets. Thus for queue size from 5packets to 30packets the throughput will increase and loss-rate will decrease as the loss of packets decreases with increase in queue size. From queue size 30packets to 100packets the throughput remains constant and loss-rate equals zero as there would be no loss of packets.

Throughput:

For a constant Poisson input traffic rate of 0.95Mbps, with increase in the queue size of the system Throughput of the system increases until it reaches its maximum value and then it remains constant with any further increase in queue size.

Loss-rate:

Loss-rate of the system remains zero until the Poisson input traffic reaches a maximum threshold rate where the packets start dropping due to insufficient Queue size of the system. After this maximum threshold rate the loss-rate increases with increase in the Poisson input traffic rate.

Script for throughput.awk:

```
BEGIN {
totalpckts = 0;
gotime = 0;
starttime = 0;
endtime = 0;
throughput = 0;
}
#body
{
        event = $1
      time = $2
        pcktsize = $6

 if(gotime == 0) {
      starttime = time;
      gotime = 1;
  } else {
      endtime = time;
  }

#============= CALCULATE throughput=================

if (( event == "r") && ( time >= starttime ))
{
duration = endtime - starttime;
totalpckts+=pcktsize*8/1000;
throughput = totalpckts/duration;
}

}


END {
print ("throughput: "throughput"kbps");
}
```

Script for lossrate.awk:

```
BEGIN {
totalpckts = 0;
pcktslost = 0;
lossrate = 0;
}
#body
{
        event = $1
      time = $2
        pcktsize = $6

if ((( event == "r") || ( event == "d")) && ( time >= 0 ))
{
totalpckts+=pcktsize*8/1000;
}

if (( event == "d") && ( time >= 0 ))
{
pcktslost+=pcktsize*8/1000;
}

}

END {
lossrate = pcktslost/totalpckts;
print ("loss-rate: "lossrate);
}
```