A20345391 Kirubha Shankar Murugesan

ECE 541 NS2 Project

Experiment 1: Throughput and loss rate under constant-rate traffic 1. Run simulations under different scenarios, with the CBR traffic rate set to be 0.1 Mbps, 0.25Mbps, 0.5Mbps, 1 Mbps, 1.5 Mbps, 2 Mbps and 3 Mbps, respectively. Analyze the trace file and calculate the throughput and loss rate in each case. (You may need to change the trace-file name in different cases.)

TCL Script:

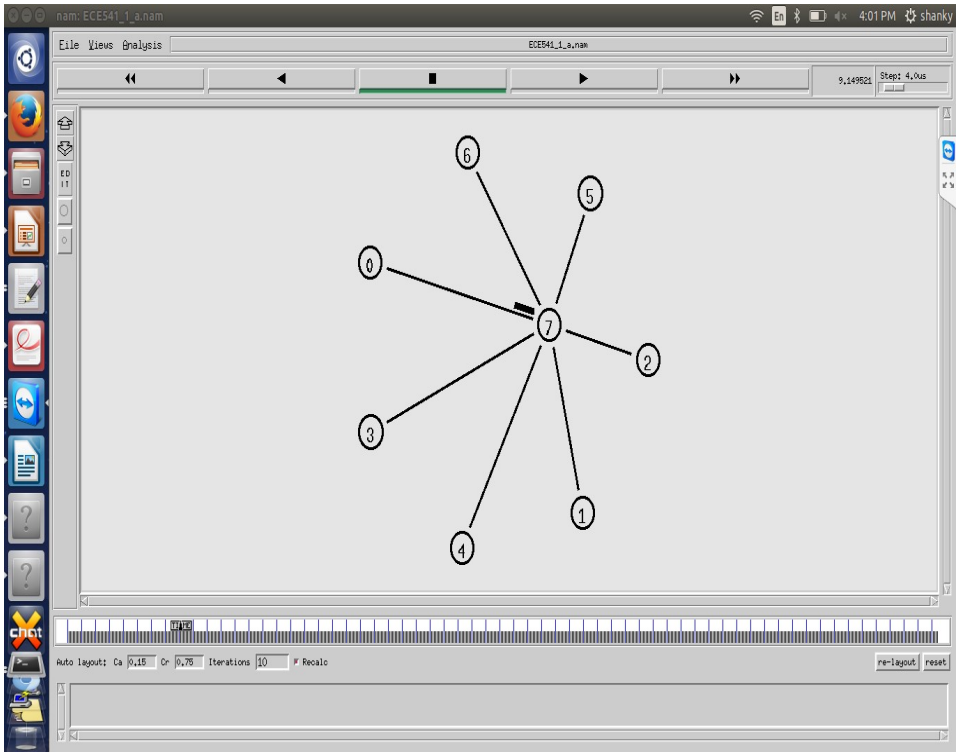
```
#Create a simulator object

set ns [new Simulator]

#Set up NAM trace file

set na [open ECE541_1_a.nam w]
$ns namtrace-all $na
$ns use-newtrace

#Define a 'finish' procedure
proc finish {} {

        global ns t0 t1 t2 t3 t4 t5 t6 na

            $ns flush-trace
        #Close the trace file
        close $t0
        close $t1
        close $t2
        close $t3
        close $t4
        close $t5
        close $t6

       #Execute nam on the trace file

        exec nam ECE541_1_a.nam &

            #exec xgraph ECE541_1_0.tr ECE541_1_1.tr ECE541_1_2.tr -geometry 800x400 &

        exit 0
}
#Create seven nodes
for {set i 0} {$i < 8} {incr i} {

        set n($i) [$ns node]
}
```

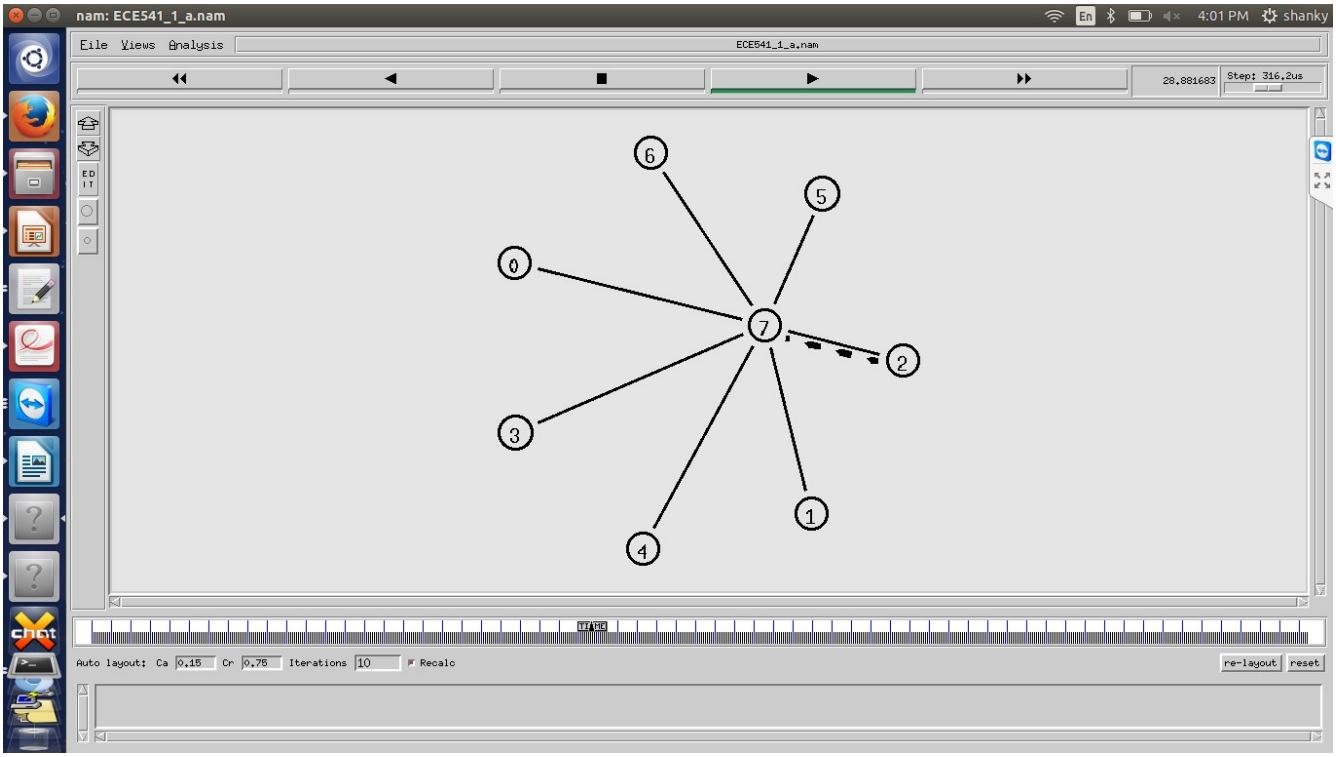
```
#Create links between the nodes
for {set i 0} {$i < 7} {incr i} {

      $ns duplex-link $n($i) $n(7) 1Mb 10ms DropTail
}
#Setting up Trace Files

set t0 [open ECE541_1_0.tr w]
$ns trace-queue $n(0) $n(7) $t0

set t1 [open ECE541_1_1.tr w]
$ns trace-queue $n(1) $n(7)  $t1

set t2 [open ECE541_1_2.tr w]
$ns trace-queue $n(2) $n(7)  $t2

set t3 [open ECE541_1_3.tr w]
$ns trace-queue $n(3) $n(7)  $t3

set t4 [open ECE541_1_4.tr w]
$ns trace-queue $n(4) $n(7)  $t4

set t5 [open ECE541_1_5.tr w]
$ns trace-queue $n(5) $n(7) $t5

set t6 [open ECE541_1_6.tr w]
$ns trace-queue $n(6) $n(7) $t6

#Create a Null agent (a traffic sink) and attach it to node n7
set null0 [new Agent/Null]
$ns attach-agent $n(7) $null0

#Create UDP port and attach to all source nodes

for {set i 0} {$i < 7} {incr i} {
set udp($i) [new Agent/UDP]
$ns attach-agent $n($i) $udp($i)
}

#Connect the UDP sources to sink
for {set i 0} {$i < 7} {incr i} {
$ns connect $udp($i) $null0
}

# Create CBR traffic
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 attach-agent $udp(0)

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp(1)
set cbr2 [new Application/Traffic/CBR]

$cbr2 attach-agent $udp(2)
set cbr3 [new Application/Traffic/CBR]

$cbr3 attach-agent $udp(3)
set cbr4 [new Application/Traffic/CBR]

$cbr4 attach-agent $udp(4)
set cbr5 [new Application/Traffic/CBR]

$cbr5 attach-agent $udp(5)
set cbr6 [new Application/Traffic/CBR]

$cbr6 attach-agent $udp(6)


#Rates changed based on the experiment necessity
$cbr0 set rate_ 0.1Mb
$cbr1 set rate_ 0.25Mb
$cbr2 set rate_ 0.5Mb
$cbr3 set rate_ 1Mb
$cbr4 set rate_ 1.5Mb
$cbr5 set rate_ 2Mb
$cbr6 set rate_ 3Mb


#Schedule events for the CBR agents
$ns at 0.0 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"

$ns at 10.0 "$cbr1 start"
$ns at 20.0 "$cbr1 stop"

$ns at 20.0 "$cbr2 start"
$ns at 30.0 "$cbr2 stop"

$ns at 30.0 "$cbr3 start"
$ns at 40.0 "$cbr3 stop"

$ns at 40.0 "$cbr4 start"
$ns at 50.0 "$cbr4 stop"
```

```
$ns at 50.0 "$cbr5 start"
$ns at 60.0 "$cbr5 stop"

$ns at 60.0 "$cbr6 start"
$ns at 70.0 "$cbr6 stop"

#Call the finish procedure after 10 seconds of simulation time

$ns at 75.0 "finish"

#Run the simulation

$ns run
```

This file creates all the .tr files for the question and save it separately.
The screenshots are shown below:





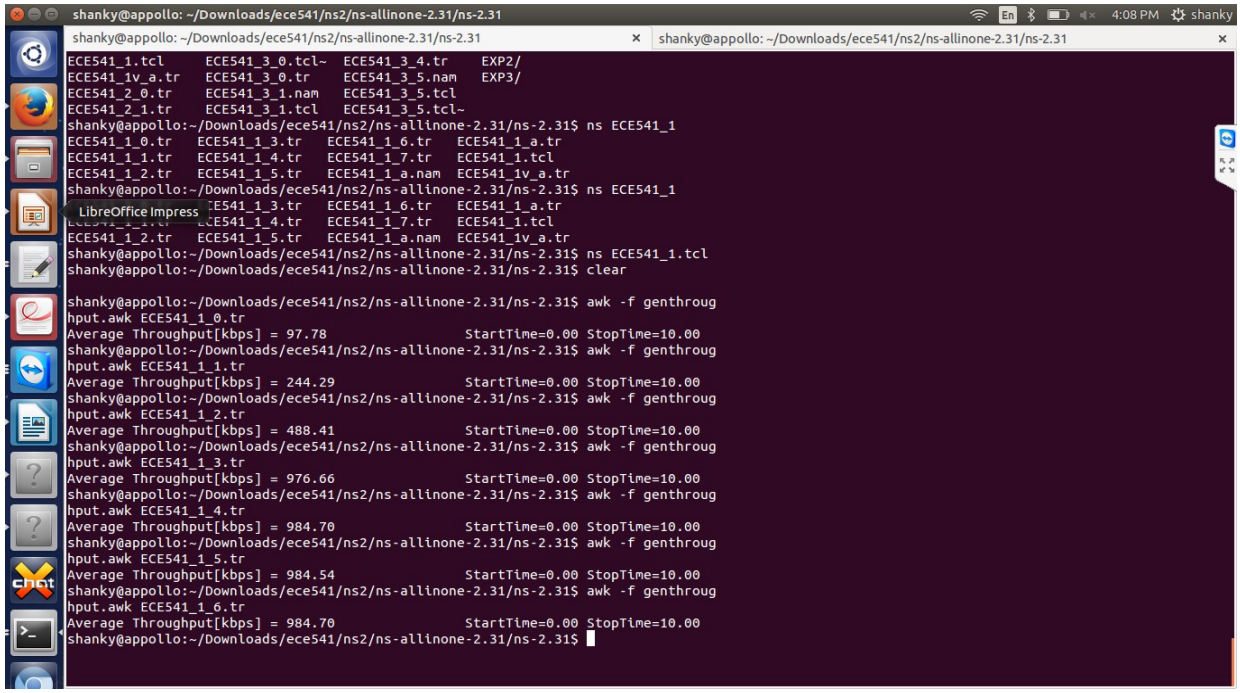
After every 10 seconds the corresponding node is connected above is the value for rate .1 Mb and .5Mb

2) We define the throughput to be "N/T", where N is the total number of packets received by n1 and T represents the simulation duration. Plot the curve of the throughput (at node n1) versus the input traffic rate (at node n0). Recommend to use the input traffic rate (i.e., 0.1Mbps, 0.25Mbp, and so on) as the x-axis. (Note: Please map your throughput from packets/second to bits/second.)

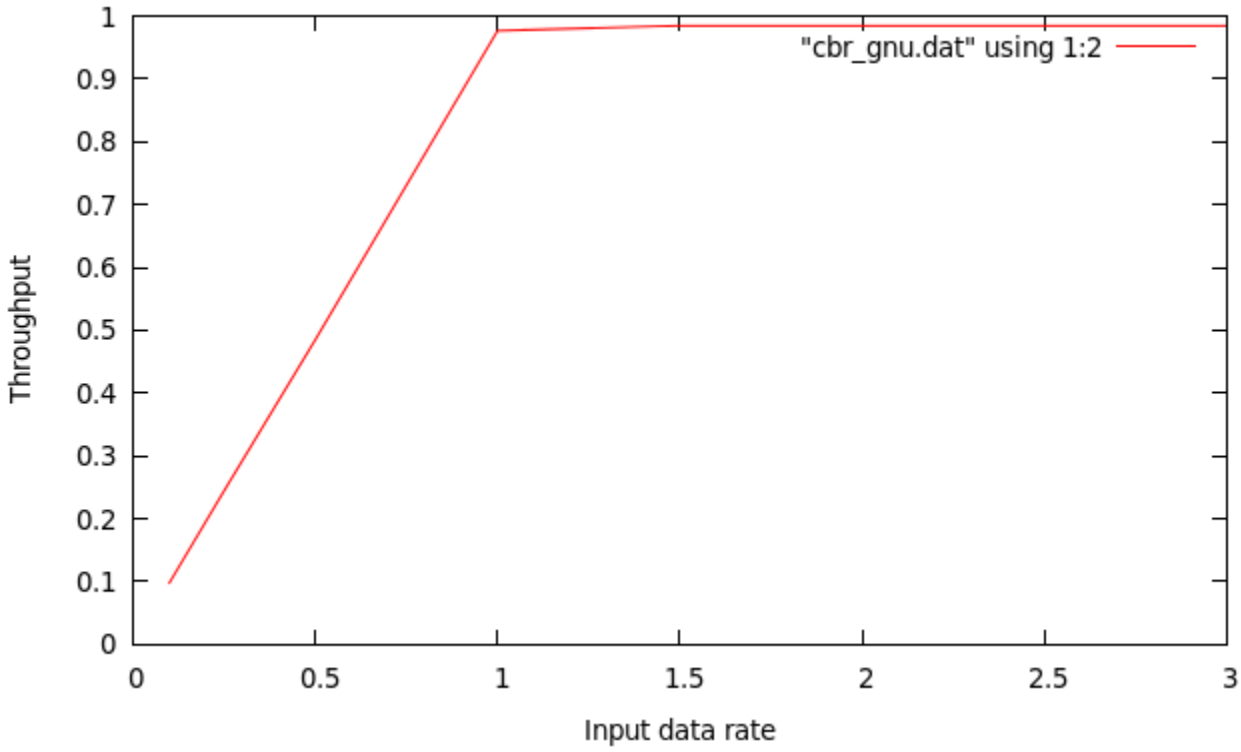
Throughput Calculation - The throughput in kbps can be given by the mathematical equation =

Packets received∗size of each packet (in bytes)∗8/Simtime * 1024

An awk file is written to calculate the throughput from the obtained tracefiles and the output screenshot is given. The name of the awk file is genthroughput.awk.

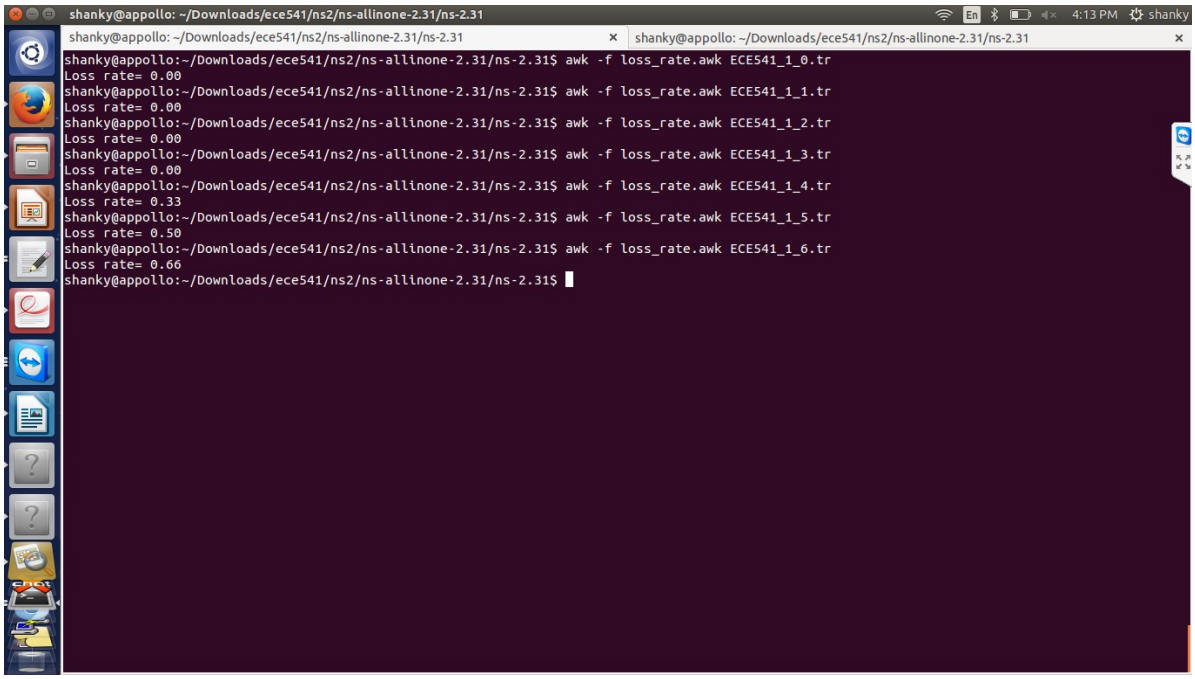


Corresponding GNU plot:

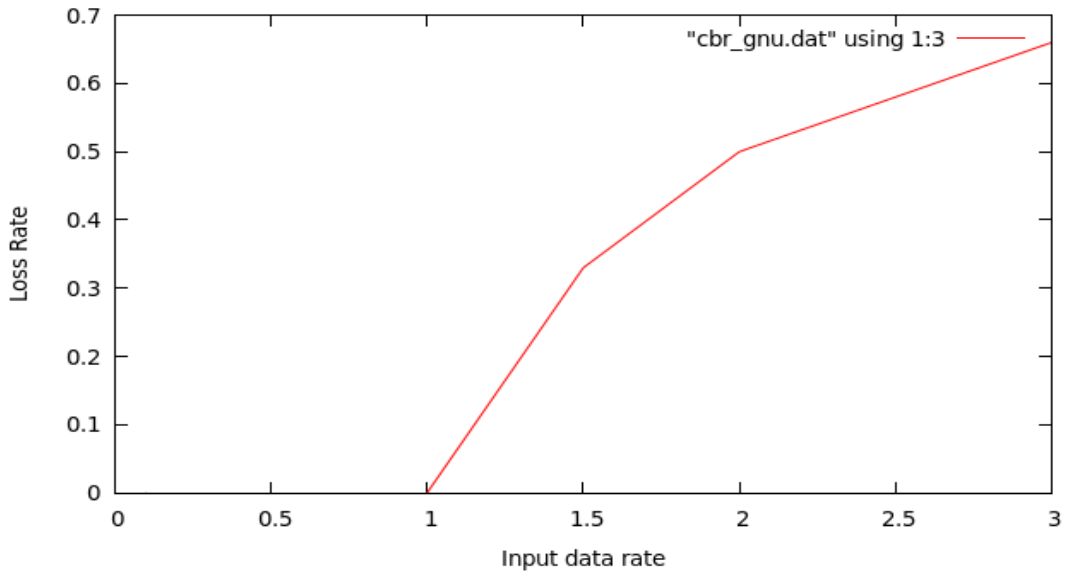


3) We define the loss rate to be "L/M", where L is the total number of lost packets and M represents the total number of packets sent out from n0. Plot the loss rates of the CBR flow versus the input traffic rate at node n0.

Similarly loss_rate.awk calculates the loss_rate from the tracefiles.



Corresponding gnu plot :



4. Explain your simulation results.

Input v/s Throughput Rate – It can be inferred that initially the throughput is almost equal to the input data rate. But it achieves a constant value after input cross 1 Mbps mark. For example at 0.5 mbps the throughput is 0.4999 mbps, but fwhen the transmission is above the link capacity such as 1,1.5,2,3 Mbps it has a constant value around 998 kbps. So no matter however high the data rate goes, the link will reach its saturation at 1Mbps and above this it will start dropping the packets.

Input v/s Loss Rate – As the rate increases more and more above 1 Mbps (link capacity) the number of packets being dropped increases. So they are directly proportional

Experiment 2: Throughput and loss rate under Poisson traffic 1. Create a Poisson traffic agent and attach it to node n0. Create a "Null" agent and attach it to node n1. Keep the rest of configurations unchanged 2. Run simulations under different scenarios, with the mean rate of the Poisson traffic set to be 0.1 Mbps, 0.25Mbps, 0.5Mbps, 1 Mbps, 1.5 Mbps, 2 Mbps and 3 Mbps, respectively. Analyze the trace file and calculate the throughput and loss rate in each case. (You may need to change the trace-file name in difference cases.)

TCL script:

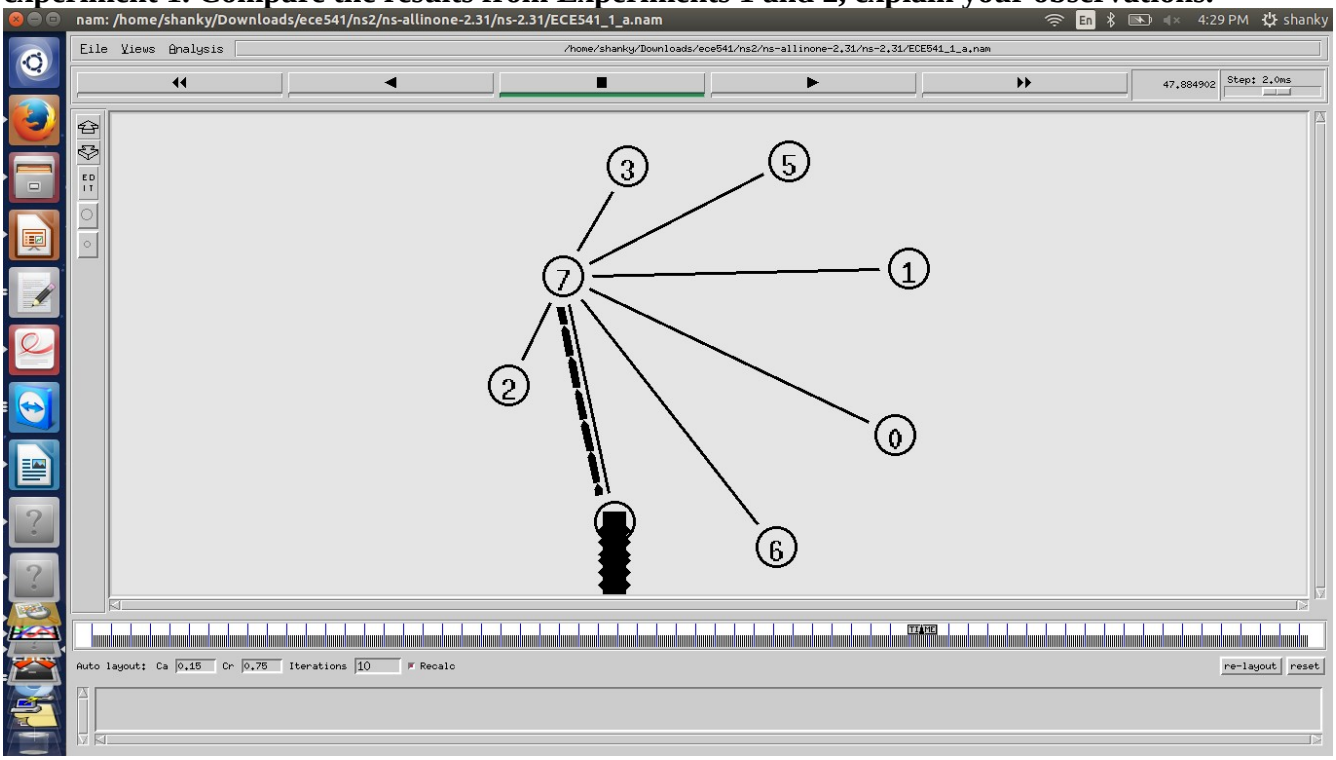
```
#Create a simulator object

set ns [new Simulator]

#Set up NAM trace file

set na [open ECE541_2_a.nam w]
$ns namtrace-all $na

#Define a 'finish' procedure
proc finish {} {

        global ns t0 t1 t2 t3 t4 t5 t6 na

           $ns flush-trace
        #Close the trace file
        close $t0
        close $t1
        close $t2
        close $t3
        close $t4
        close $t5
        close $t6

       #Execute nam on the trace file

        exec nam ECE541_1_a.nam &
```

```
        exit 0
}

#Create seven nodes
for {set i 0} {$i < 8} {incr i} {

    set n($i) [$ns node]
}
#Create links between the nodes
for {set i 0} {$i < 7} {incr i} {

    $ns duplex-link $n($i) $n(7) 1Mb 10ms DropTail
}
#Setting up Trace Files

set t0 [open ECE541_2_0.tr w]
$ns trace-queue $n(0) $n(7) $t0

set t1 [open ECE541_2_1.tr w]
$ns trace-queue $n(1) $n(7)  $t1

set t2 [open ECE541_2_2.tr w]
$ns trace-queue $n(2) $n(7)  $t2

set t3 [open ECE541_2_3.tr w]
$ns trace-queue $n(3) $n(7)  $t3

set t4 [open ECE541_2_4.tr w]
$ns trace-queue $n(4) $n(7)  $t4

set t5 [open ECE541_2_5.tr w]
$ns trace-queue $n(5) $n(7) $t5

set t6 [open ECE541_2_6.tr w]
$ns trace-queue $n(6) $n(7) $t6

#Create a Null agent (a traffic sink) and attach it to node n7
set null0 [new Agent/Null]
$ns attach-agent $n(7) $null0

#Create UDP port and attach to all source nodes

for {set i 0} {$i < 7} {incr i} {
set udp($i) [new Agent/UDP]
$ns attach-agent $n($i) $udp($i)
}
```

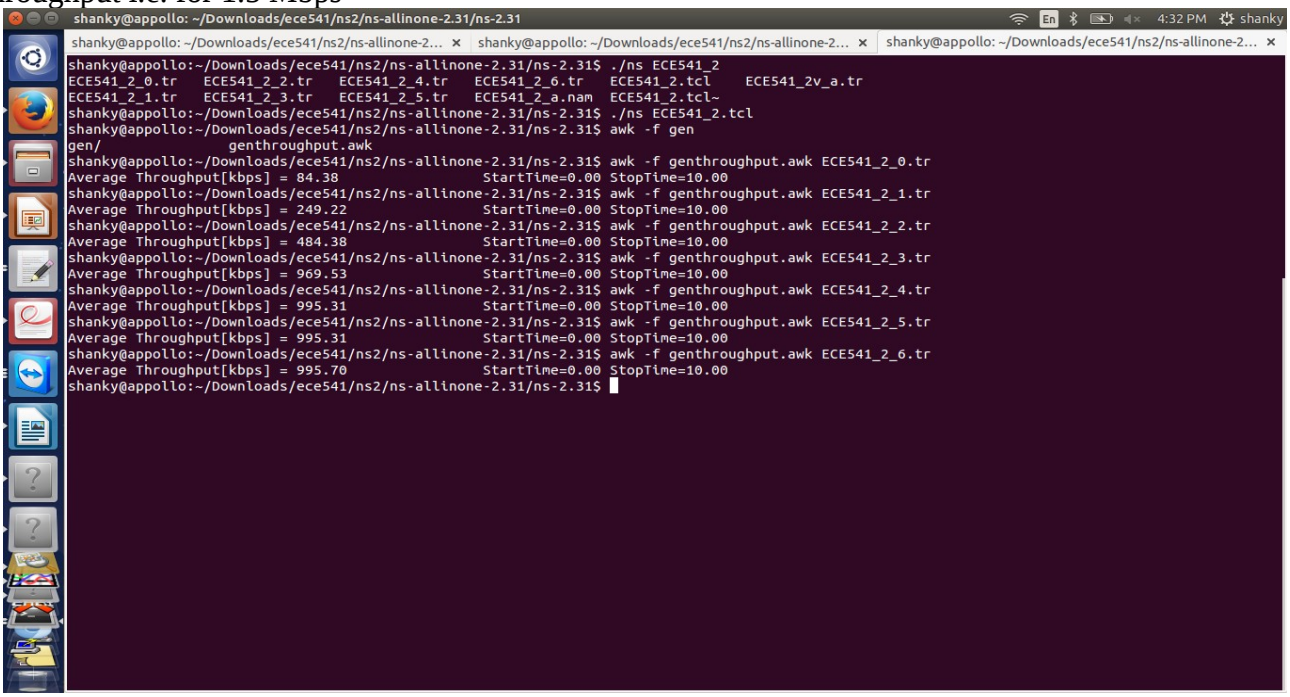
```
#Connect the UDP sources to sink
for {set i 0} {$i < 7} {incr i} {
$ns connect $udp($i) $null0
}

# Create Poisson traffic
set poi0 [new Application/Traffic/Poisson]

$poi0 attach-agent $udp(0)

set poi1 [new Application/Traffic/Poisson]

$poi1 attach-agent $udp(1)
set poi2 [new Application/Traffic/Poisson]

$poi2 attach-agent $udp(2)
set poi3 [new Application/Traffic/Poisson]

$poi3 attach-agent $udp(3)
set poi4 [new Application/Traffic/Poisson]

$poi4 attach-agent $udp(4)
set poi5 [new Application/Traffic/Poisson]

$poi5 attach-agent $udp(5)
set poi6 [new Application/Traffic/Poisson]

$poi6 attach-agent $udp(6)


#Rates changed based on the experiment necessity
$poi0 set rate_ 0.1Mb
$poi1 set rate_ 0.25Mb
$poi2 set rate_ 0.5Mb
$poi3 set rate_ 1Mb
$poi4 set rate_ 1.5Mb
$poi5 set rate_ 2Mb
$poi6 set rate_ 3Mb


#Schedule events for the Poisson agents
$ns at 0.0 "$poi0 start"
$ns at 10.0 "$poi0 stop"

$ns at 10.0 "$poi1 start"
$ns at 20.0 "$poi1 stop"
```

```
$ns at 20.0 "$poi2 start"
$ns at 30.0 "$poi2 stop"

$ns at 30.0 "$poi3 start"
$ns at 40.0 "$poi3 stop"

$ns at 40.0 "$poi4 start"
$ns at 50.0 "$poi4 stop"

$ns at 50.0 "$poi5 start"
$ns at 60.0 "$poi5 stop"

$ns at 60.0 "$poi6 start"
$ns at 70.0 "$poi6 stop"

#Call the finish procedure after 10 seconds of simulation time

$ns at 75.0 "finish"

#Run the simulation

$ns run
```

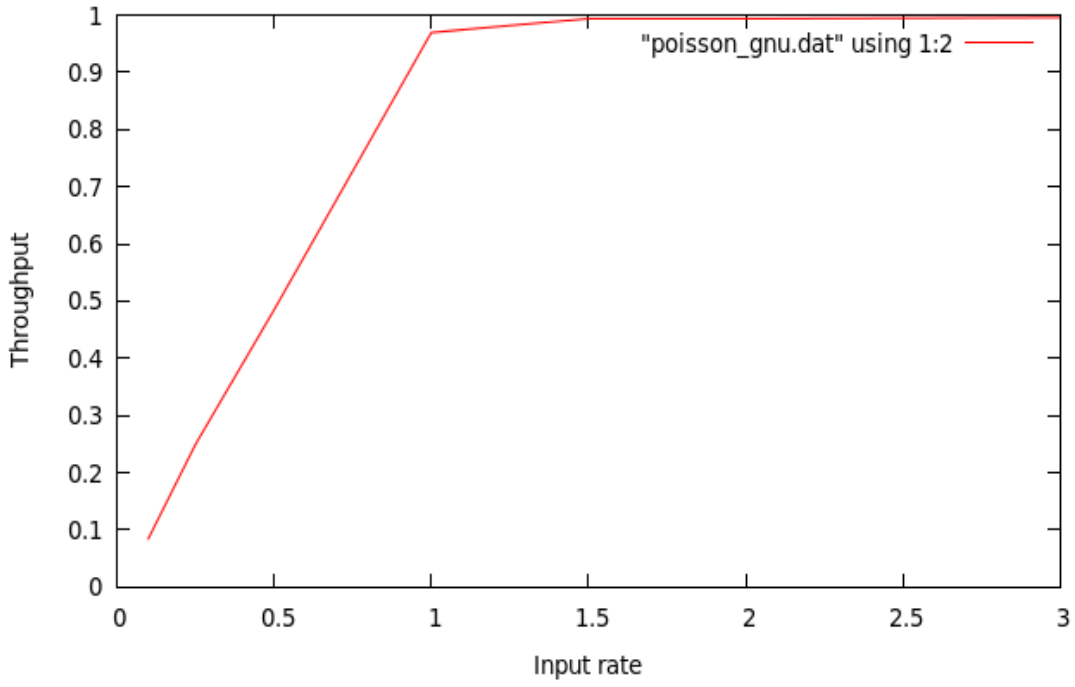
2. Plot the “throughput vs. input rate” and “loss rate vs. input rate” curves, as you have done in experiment 1. Compare the results from Experiments 1 and 2, explain your observations.



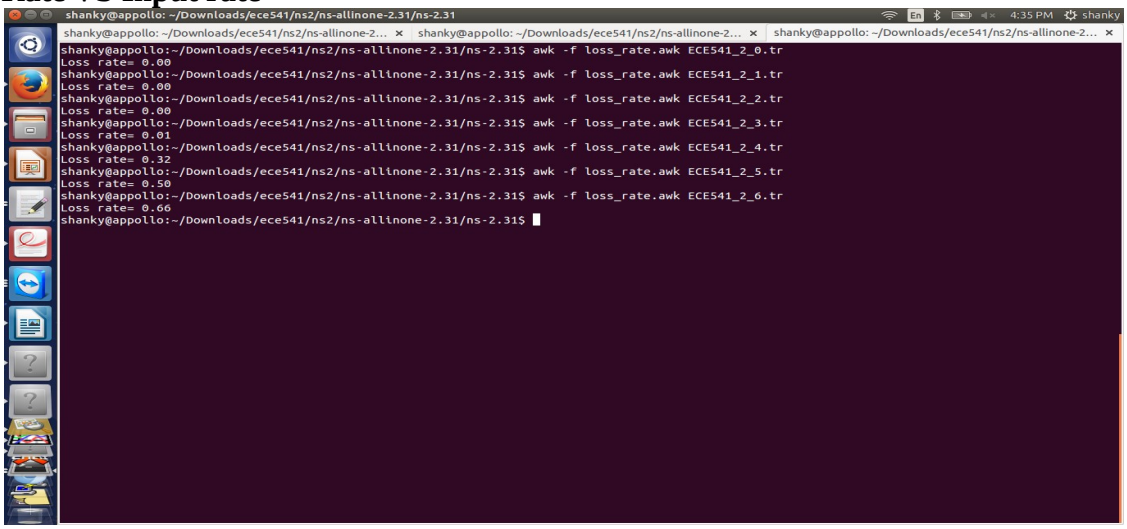
The queue at the link between 4 and 7 is shown above the same awk files are used to find the throughput i.e. for 1.5 Mbps



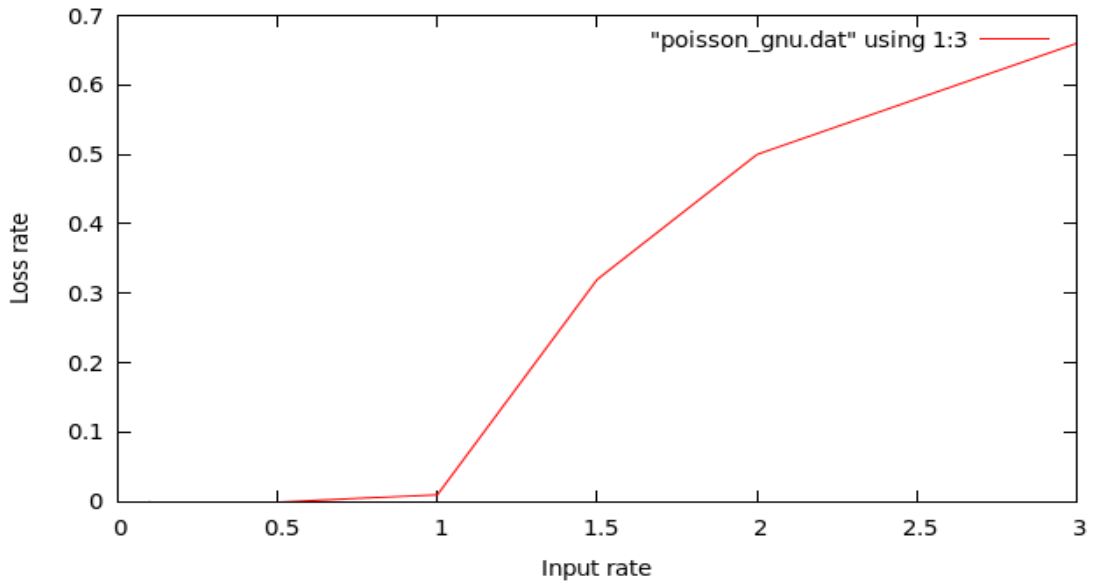
GNU plot for the above data:



3. Loss Rate VS Input rate



GNU plot for Input rate vs Loss rate:



4. Analysis & Comparison:

Input Vs Throughput – Using various input rates the throughput was calculated. For the initial rates upto 1 Mbps it has a linear increase in the throughput then for a very high input data rates Poisson traffic achieves a constant values of throughput. This is because the throughput for a poisson is dependent more on link capacity than input rate.

Loss Rate vs Input Rate – At the link rate = input rate we observe a small percentage of packets being dropped. This rate that's a relatively higher increase from 1 to 1.5 Mbps. And then the packets lost linearly increase above this rate.

Comparison: In the first experiment the input data type was CBR and it achieved a constant throughput at and after the link capacity of 1 Mbps. This is the maximum throughput as compared to data rates below 1 Mbps. The throughput for both the experiments below 1 Mbps follows a similar (but not same values) pattern I.e. there is increase in throughput with the increase in input data rate.

3. Set the rate of Poisson traffic to be 0.95Mbps. In previous experiments, we use the default queue size for the link, which is 50. Now, run simulations under six scenarios, with queue size set to be 5, 10, 30, 50, 70, and 100, respectively. 2. Based on your simulation results, plot the "throughput vs. queue size" and "loss rate vs. queue size" curves. Explain your simulation results.

1. TCL script:

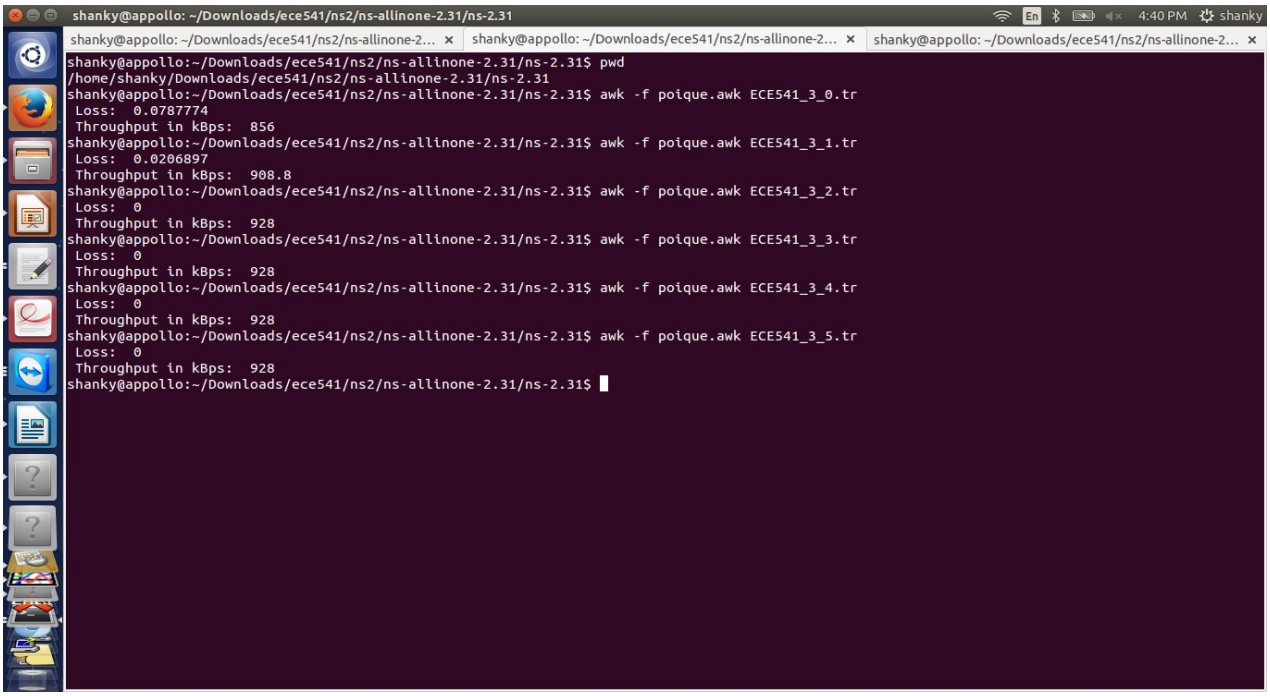
```
#This step is to create a simulator

set ns [new Simulator]

#To set a nam & trace file

set nf [open ECE541_3_1.nam w]

$ns namtrace-all $nf

set nd [open ECE541_3_2.tr w]

$ns trace-all $nd

#Code for 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_1.nam &

 exit 0
```

```
}
set n0 [$ns node]

set n1 [$ns node]

#Code to create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#now we need to set queue-size of link n0-n1

$ns queue-limit $n0 $n1 30

#A UDP agent is created and we need to attach it to node n0

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

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

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

#Generation of Poisson traffic source and attaching it to udp0

set Poi1 [new Application/Traffic/Poisson]

$Poi1 attach-agent $udp0

$Poi1 set rate_ 0.95Mb


#Connect the traffic sources with the traffic sink

$ns connect $udp0 $null0

#Start and stop - event scheduling for the CBR agents

$ns at 0.0 "$Poi1 start"

$ns at 10.0 "$Poi1 stop"

$ns at 10.0 "finish"
```

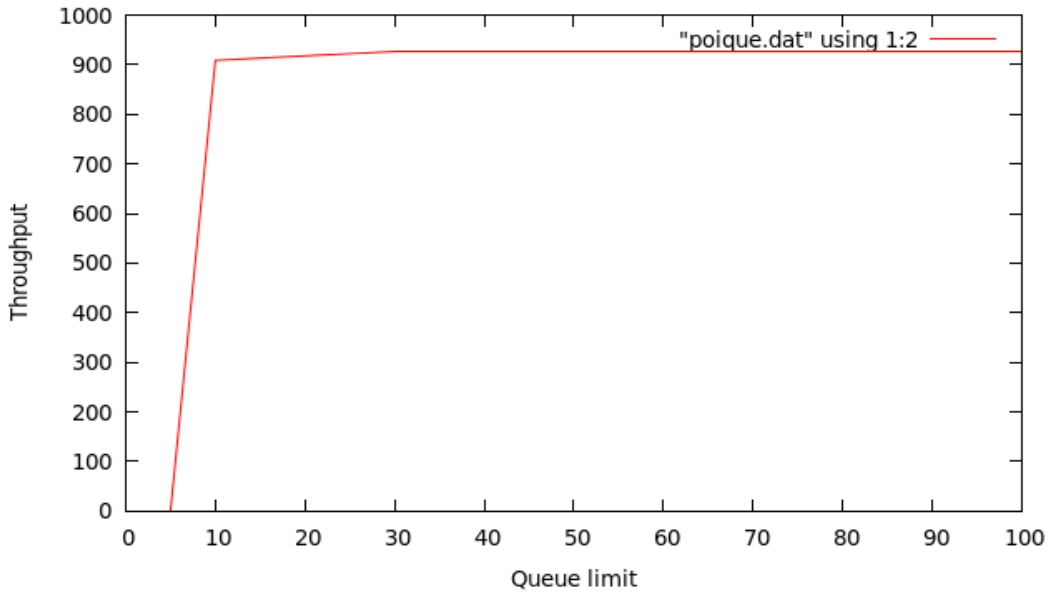
#Last step is to Run the simulation

$ns run

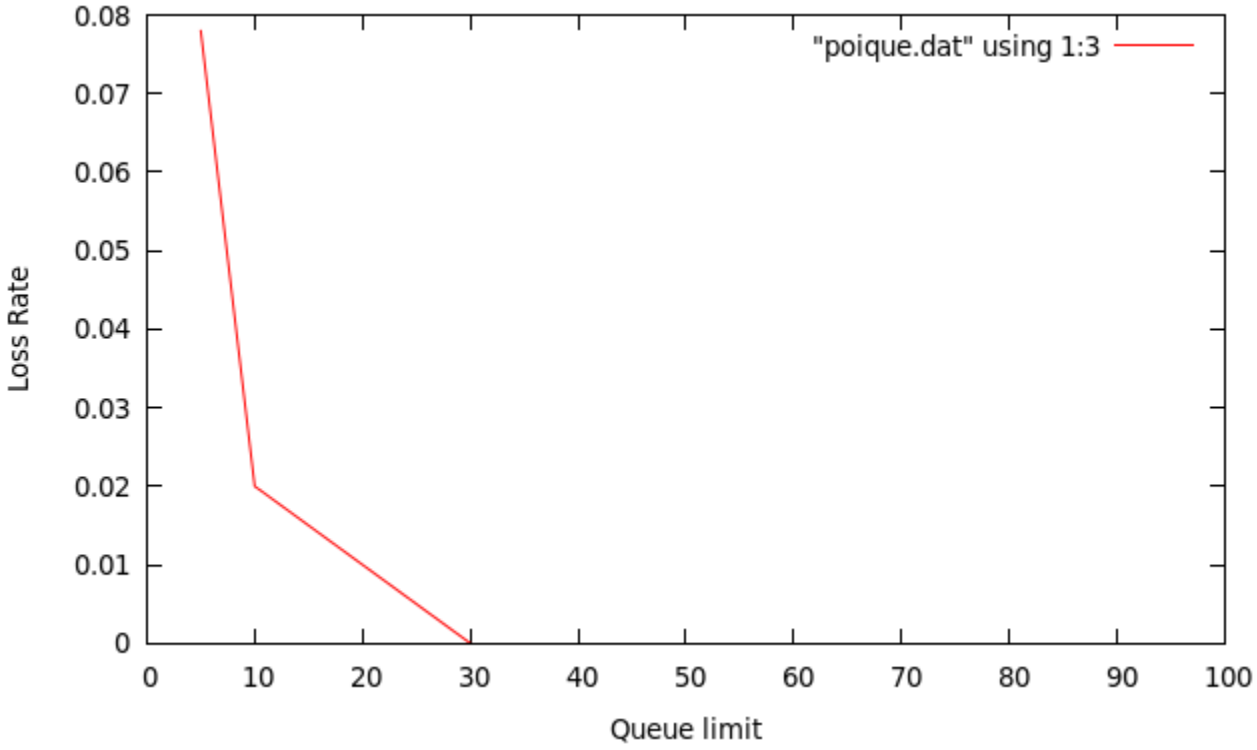
2. Throughput and Loss are calculated together using poique.awk



GNU plot for throughput vs queue-limit:



GNU plot for loss vs queue-limit:



Analysis – It was observed that if the queue size was small then the throughput was less and when this queue size was increases the throughput increased. It is found that for small queue size a lot of packets were dropped, this was due to less queuing capacity. All the losses occur at 5 & 10 queue size, with a higher queue size we can achieve a high throughput and zero packet loss as it enhances the queuing capacity of node.