# 17COP512 - Network Modelling and Performance

# Coursework Report

By Simon Yan Lung Yip – B424047

# Deliverable 1 - Introduction

In this deliverable, we need to create simulation scripts and run the simulation using NS-2 network simulator for the given network topology and configuration in the specification. There are two simulation scenarios to be written in TCL script. One will have DropTail queue management schemes and the other will have RED queue management scheme. Both simulation has a dumbbell topology that consists of two routers and eight edge nodes. Both simulation will produce graphs which allows us to analyse performance based on throughput and queue sized observed in both simulation.

## - Brief explanation of DropTail and RED queue management schemes

DropTail is a traditional queue management schemes which uses the FIFO (first-in-first-out) buffer process. DropTail sets a maximum queue length and the queue is fill with incoming packets. When the queue is full, it will drop any newly arrived packet. In other words, it will drop packet from the tail of the queue until the queue has enough space to accept new packets.

Random Early Detection (RED) is another queue management schemes which use FIFO. The idea of RED is to randomly drops packets before a queue becomes full. This allow the source node to respond to congestion early and notify end hosts about incoming congestion. This notification allows the host to reduce their transmission rates before buffers overflow. This approach is called active queue management.

RED drops packets based on a probability that increases with the average queue length. As the queue grow, the probability of dropped packets grows. There are two thresholds in a RED queue with determine the rate of packet drop, a lower threshold and an upper threshold. When the queue length is lower than the lower threshold, it will not drop any packets. When it is higher than the lower threshold but less than the upper threshold, packets are dropped with a probability that is piecewise linear and increases with the average queue length. When queue length exceed the upper threshold, all packets are dropped.

## 1.2 - Network topology diagram from NAM visualiser.

Figure 1 shows the network topology diagram from NAM visualiser

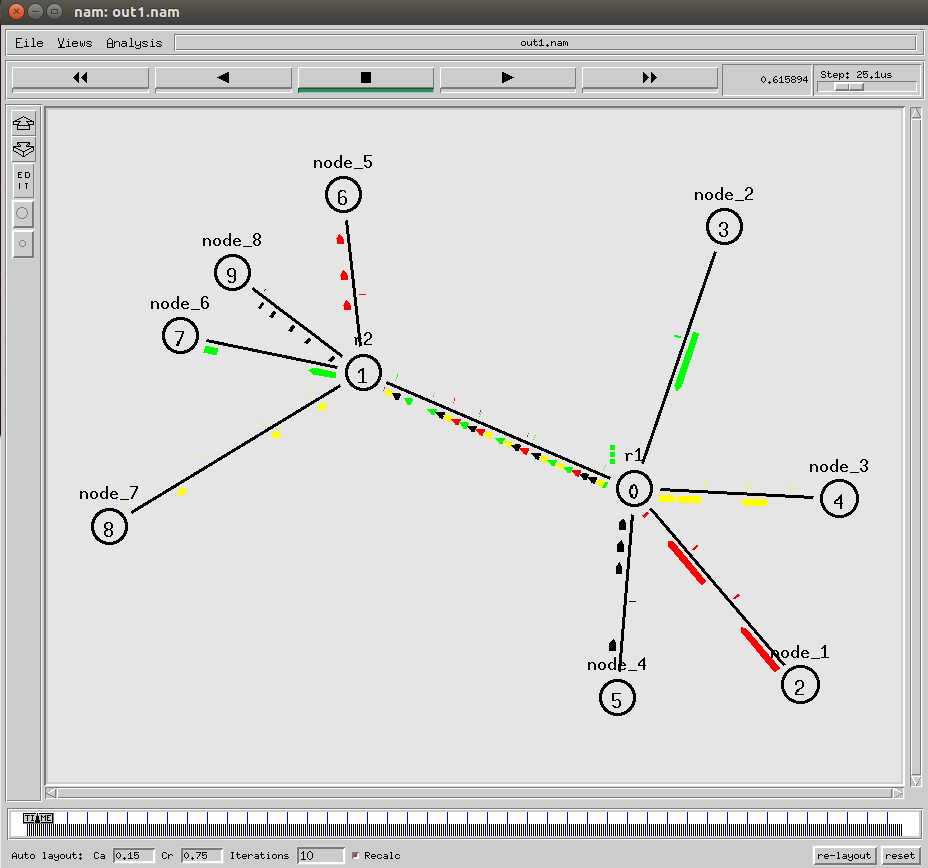


Figure 1, network topology of both simulations

Both simulation will have the same topology. Each node is labelled at the top of the node. The number in the centre of each node represent the order of the node which it was initialised. The label at the top of the node is the name of each node. We can see that node 1 to 4 are connected to router 1 and node 5 to 8 are connected to router 2. The packets from node 1 to node 5 are represented by the colour red. The packets from node 2 to node 6 are represented by the colour green. The packets from node 3 to node 7 are represented by the colour yellow. Lastly, the packets from node 4 to node 8 are represented by the colour black.

The two simulation scripts named DropTailSim.tcl and REDSim.tcl. Each of the script, when run in NS2 will generate NAM traces which are out1.nam and out2.nam respectively. The NAM traces generated from the simulation scripts can be found in the Appendix of the report.

## 1.3 - Link configurations

The specification specified various link configurations between each nodes. The network link configuration defined in the specification are shown in the table below.

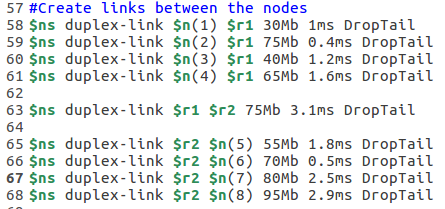
Table 1 – Link configuration in specificaton

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source | Destination | Bandwidth (Mbps) | Propagation Delay (ms) | Queue Management Scheme |
| N1 | R1 | 30 | 1 | DropTail |
| N2 | R1 | 75 | 0.4 | DropTail |
| N3 | R1 | 40 | 1.2 | DropTail |
| N4 | R1 | 65 | 1.6 | DropTail |
| R2 | N5 | 55 | 1.8 | DropTail |
| R2 | N6 | 70 | 0.5 | DropTail |
| R2 | N7 | 80 | 2.5 | DropTail |
| R2 | N8 | 95 | 2.9 | DropTail |
| R1 | R2 | 75 | 3.1 | DropTail/RED |

All the link configuration defined the simulation scripts have the following syntax:

*$ns duplex-link <$node\_1> <$node\_2> <Bandwidth> <Delay> <Queue\_Management\_Scheme>*

Figure 2 is the link configuration between N1 and R1 in the simulation script.

  
Figure 2

As you can see, the Bandwidth if set to 30Mbps, the Propagation Delay is set to 1ms and the queue management scheme is set to DropTail.

Figure 3 shows all the link configurations (except R1 to R2)

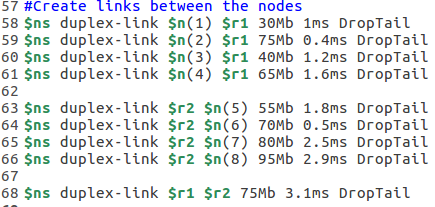
  
Figure 3

Figure 4 shows the link configuration between R1 to R2 in the first simulation script.

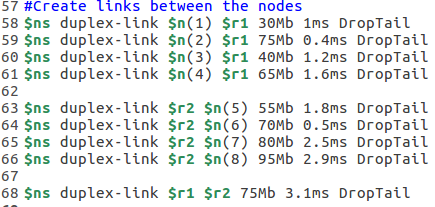
  
Figure 4

Figure 5 shows the link configuration between R1 to R2 in the second simulation script.

  
Figure 5

We can see that the link configuration in the specification matches with the link configurations in the simulation scripts.

## 1.4 - Generated Graphs

The graphs below show throughput, and current queue size of the bottleneck link between the router 1 and router 2.

Figure 6 shows the xgraph of the throughput between link R1 and R2 for DropTail

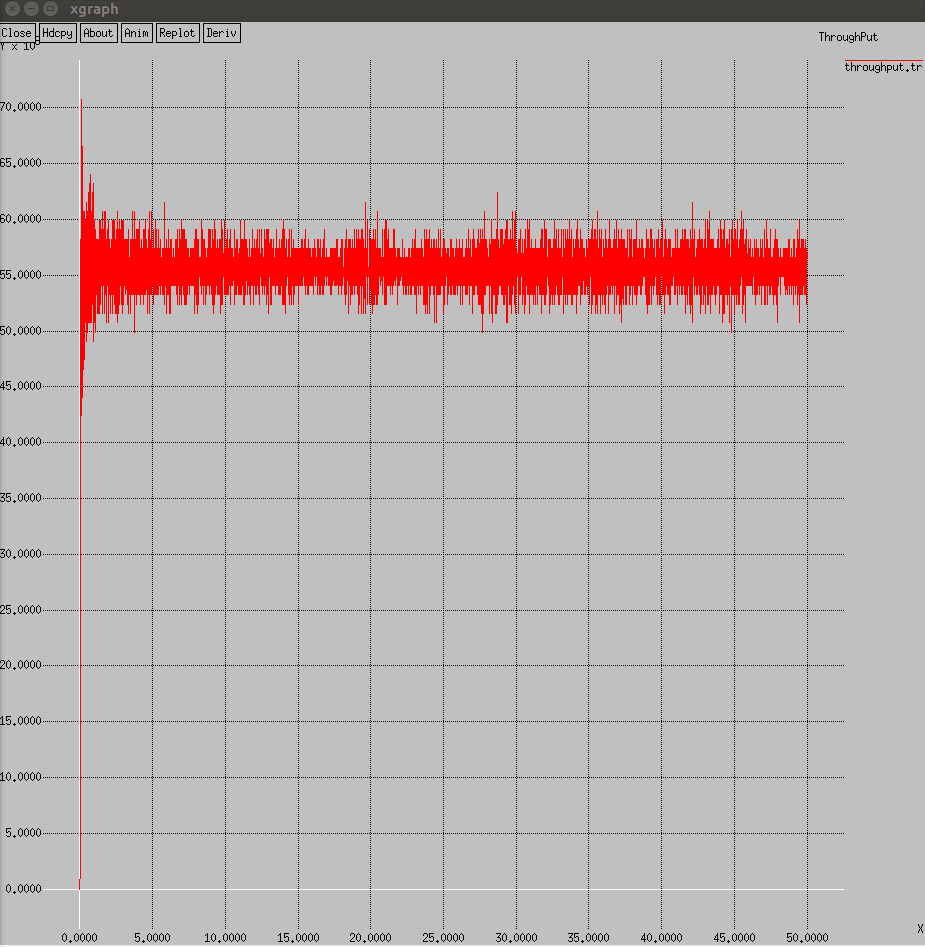
  
Figure 6

Figure 7 show the xgraph of the throughput between link R1 and R2 for RED

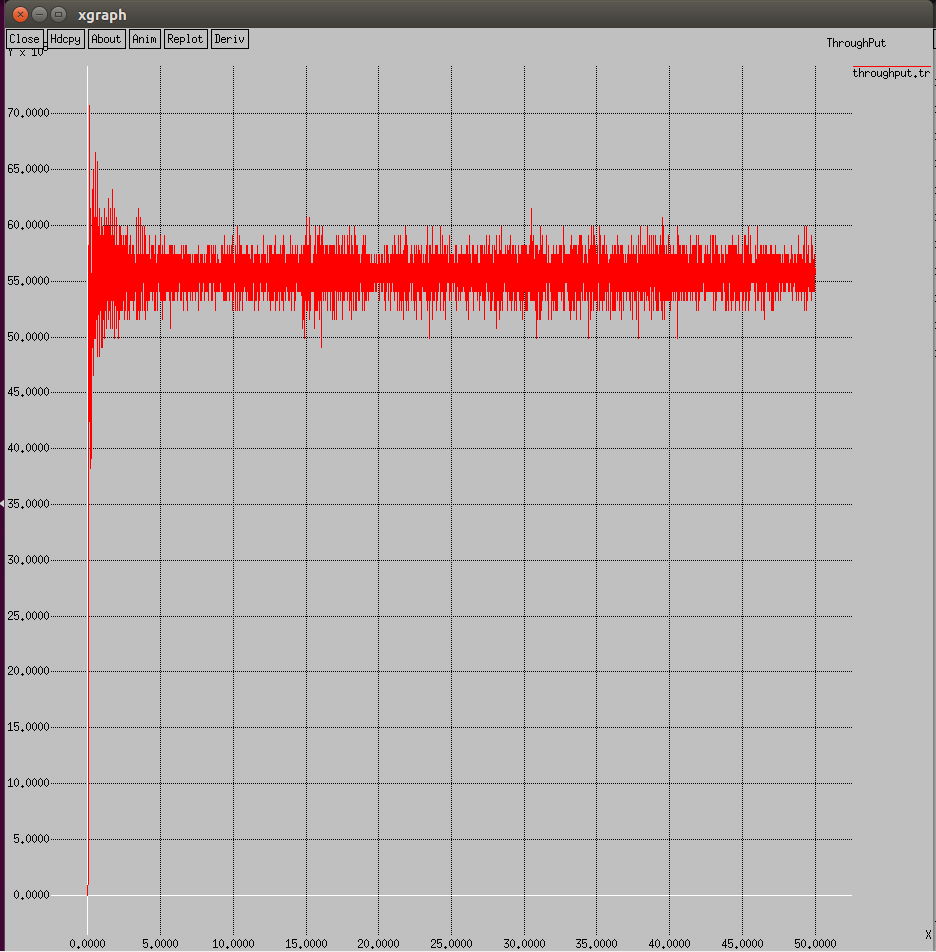
  
Figure 7

Figure 8 shows the xgraph of the current queue size between link R1 and R2 DropTail

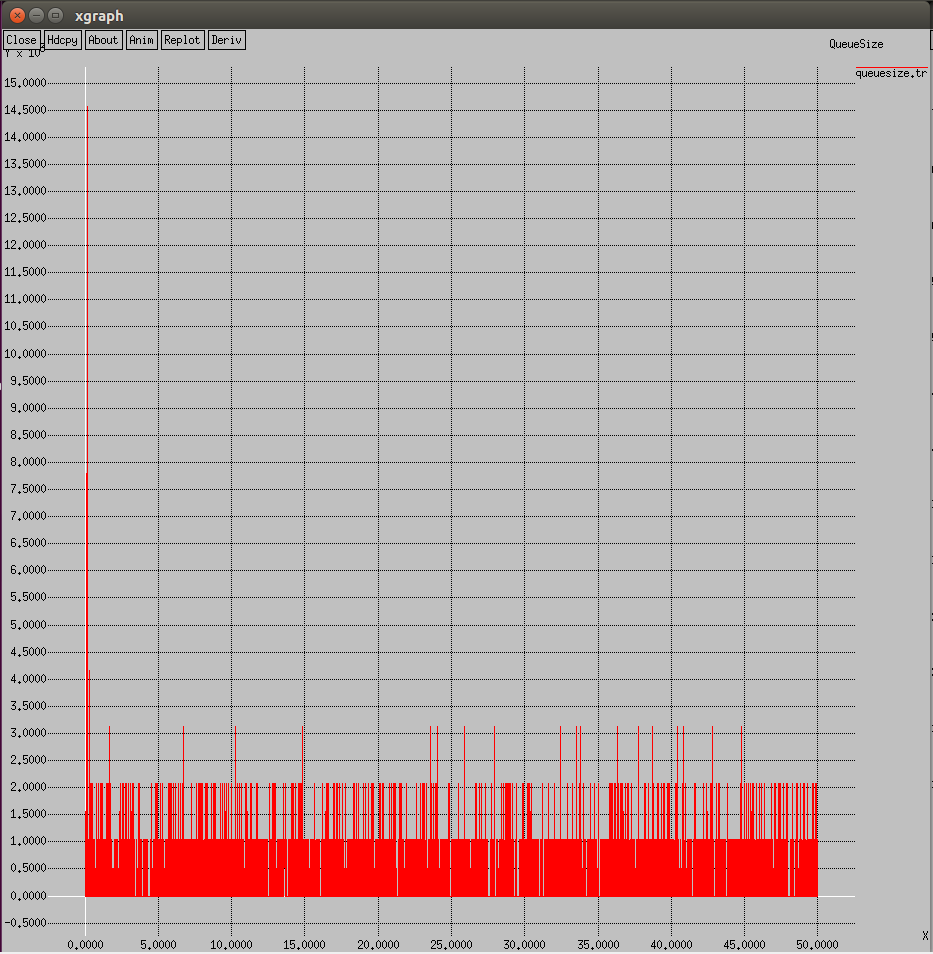
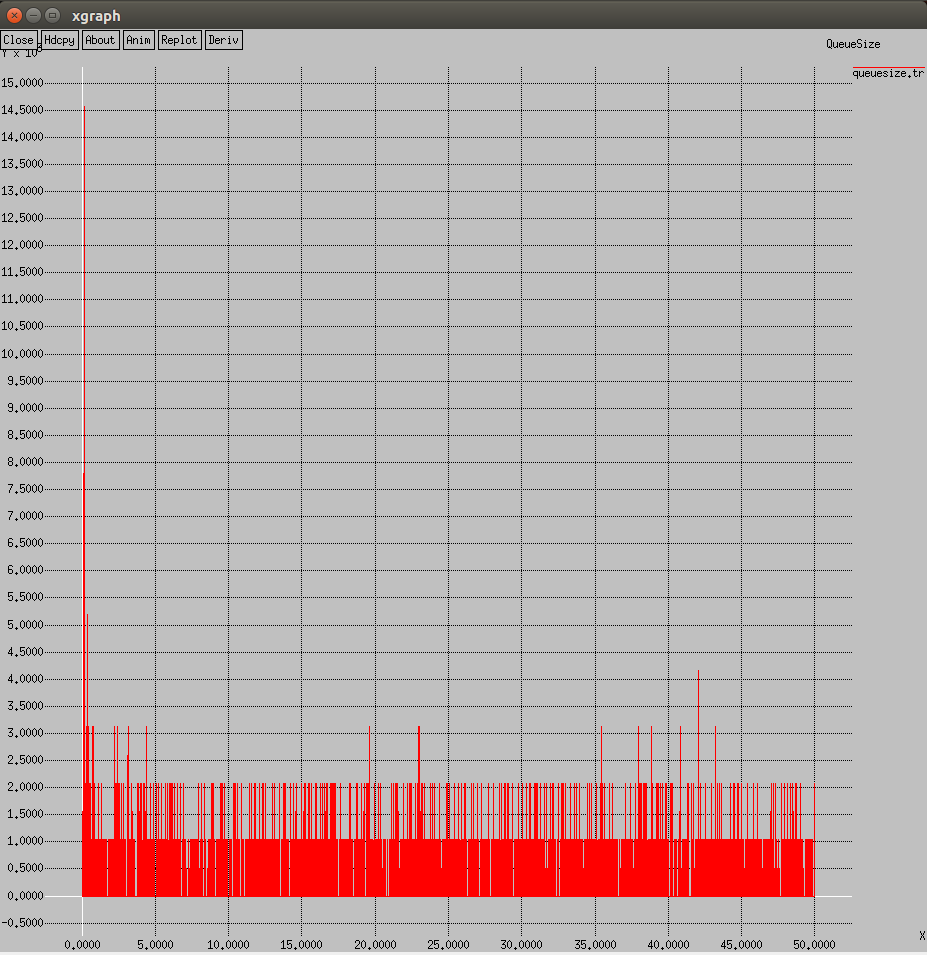
  
Figure 8

Figure 9 shows the xgraph of the current queue size between link R1 and R2 RED

  
Figure 9

## 1.5 - Analysis of Performance

From the throughput graph, we can see that the both result are quite similar to each other with minute differences. At the beginning of the graph, the throughput spike up to the 70Mbps since it is only the start of the simulation and a lot of bandwidth is available for use. Then, as the time increases, we can see the throughput fluctuate from 55-65 Mbps which quite stable. This implies that there are enough bandwidth to handled specified network link configuration smoothly for both queue management schemes in this network topology.

From the queue size graphs, again both results are quite similar to each other with some difference. At the beginning of the graph, we can see that both simulation has a high peak around 14.5x103 bits. This is probably due to the start of the simulation where the router initially received a lot of packets from the nodes. Later on, both queues decease to range 1-3x103 bits and continue this status until the end of the simulation. One difference between the graph is that RED has less peck at 3x103 bits compare to DropTail. This can be seen as slight improvement of performance compared to DropTail.

In my opinion, the reason that both results are very similar because the network link configuration is not inefficient enough for us to see that difference between queue management scheme. For example, if the bandwidth between the routers are lower, the queue in the DropTail simulation would fill up whereas RED would never has it queue fill to the limit due to it nature of dropping packets as the queue length increases pass a certain threshold.

RED and DropTail performance based on throughput and queue size observed from simulation results.

# Deliverable 2 - Introduction

In this deliverable, we have to correctly implement Little’s Result (Formula) using MATLAB to obtain the mean waiting time given a number of conditions specified in the specification and plot a series of resulting curves on the same chart.

## 2.1 - Discrete-time M/M/1

Bernoulli process is a random experiment with out two outcomes, 0 and 1. A discrete random variable X has a Bernoulli distribution with parameter *p (0 <= p <= 1),* if X takes the values 0 and 1 only than

*P( X = 0) = 1 – p, P( X = 1) = p.*

The system used is a Discrete-Time M/M/1 (Geo/Geo/1) queue where the arrivals of the packets is an independent Bernoulli process where an is element of {0,1}, n = 1,2,3,…. This queue has an unlimited buffer size and the service discipline if First in First Out. The notation M/M/1, stand for Arrival process/Service Distribution/Number of servers(/amount of buffers), so Discrete-Time M/M/1 is Geometric for arrival process and service distribution and has one server.

Let *alpha* be the probability of a packet arriving in any time slot, *beta* be the probability that a packet will depart in any time slot.

Figure 10 show the state transition diagram for a discrete time M/M/1 queue.

Infinity

1 - alpha

beta\*(1-alpha)

alpha\*beta+(1-alpha)(1-beta)

alpha\*(1-beta)

alpha

alpha\*(1-beta)

beta\*(1-alpha)

beta\*(1-alpha)

alpha\*beta+(1-alpha)(1-beta)

Figure 10

Formula for working out the mean is

W = ΣInfinityk=0 K\*PIk / alpha where PIk = PI0 \* [ sik/(1 – beta) ], k = 1,2,3,…  
si = alpha\*(1-beta)/ beta\*(1-alpha) ,  
PI0 = (1-beta)\*(1-si)/ [1- beta\*(1 – si)].

## 2.2 - Code analysis

In Figure 11, you can see the MATLAB code for the implementation.

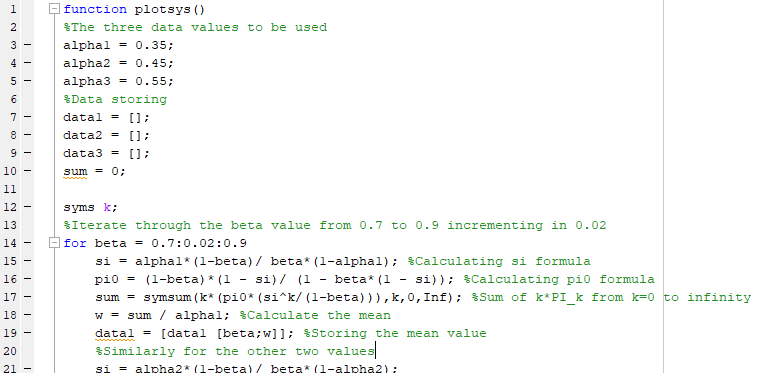
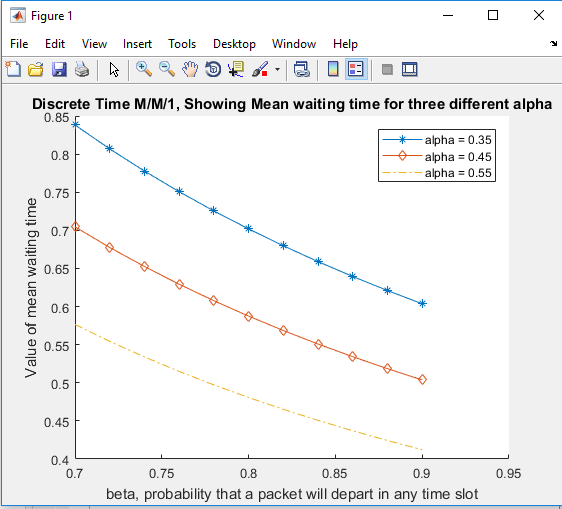


Figure 11

The rest of the code can be find in the Appendix section. Explanations are given in each line of the code.

## 2.3 - Graph Analysis

Figure 12 is the graph generated from the calculated values

  
Figure 12

From the graph in figure 12, we can clearly that as the probability of a packet arriving in any time slot (alpha) increases, the value of mean waiting time decrease. This implies that in Discrete-Time M/M/1, the more packet arriving, the mean waiting time will decrease. The same trend can be seen for probability that a packet will depart in any time slot (beta) as well. In the graph, we can see that both three curve that as x increase, y decrease. In order words, as probability of packet departing, the mean waiting time will decrease.

To achieve the minimum mean waiting time for the Discrete-time M/M/1, it is recommended that both the value of probability of a packet arrival and packet departure should be high.

# Deliverable 3 - Introduction

In this section, we need to create a MATLAB based GUI and plot the output of a Quality of Service related simulation for a number of scenarios. The plots will shows the amount of network resources required as a percentage of the total available to each node at a given point in time.

## 3.1 - Simulation Parameters

The following simulation parameters are added to the existing GUI:

* The number of nodes – ‘numOfNodes’
* The number of applications – ‘numOfApplications’
* The random seed value – ‘seed’
* The simulation time – ‘totalSimulationTime’
* Whether the simulation will perform prioritised negotiation or simply show the output of the configuration – ‘prioritise’

Figure 13 show the new GUI format

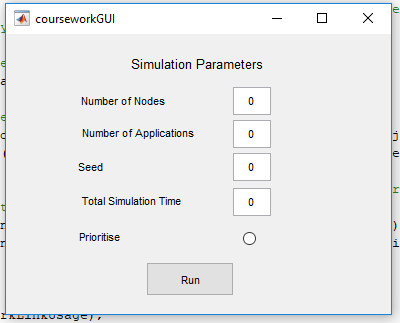


Figure 13

The user can set any number of nodes in the first text box, than the number of application in the second text box, the seed and the total simulation time in the third and fourth text box respectively. Lastly, we allow the user to chose whether to perform prioritised negotiation or not through the use of an radio button. Clicking the run button will submit the information in the GUI and run the simulation.

## 3.2 - Scenario Given and series of graphs

There are 4 scenario given in the specification. They are shown in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | Number of nodes | Number of Applications | Seed Value | Simulation Time | Prioritised Negotiation |
| 1 | 15 | 25 | 4 | 10 | No |
| 2 | 15 | 25 | 4 | 10 | Yes |
| 3 | 10 | 50 | 8 | 30 | No |
| 4 | 10 | 50 | 8 | 30 | Yes |

Figure 14 is the screenshot of the final plot at time 10 for scenario 1.

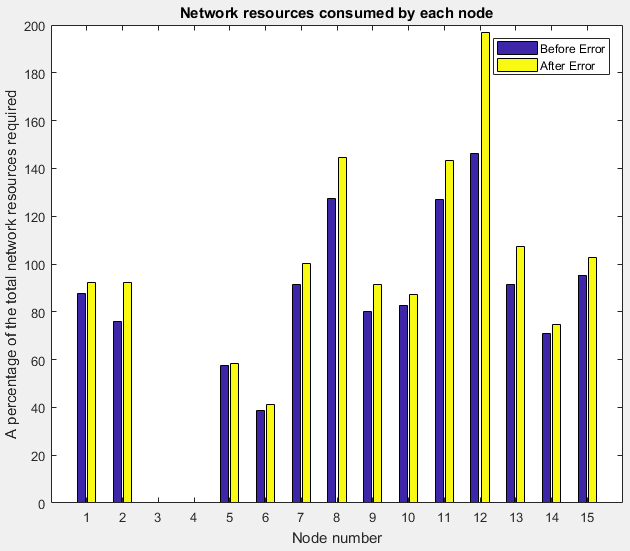
Figure 14

Figure 15 is the screenshot of the final plot at time 10 for scenario 2.

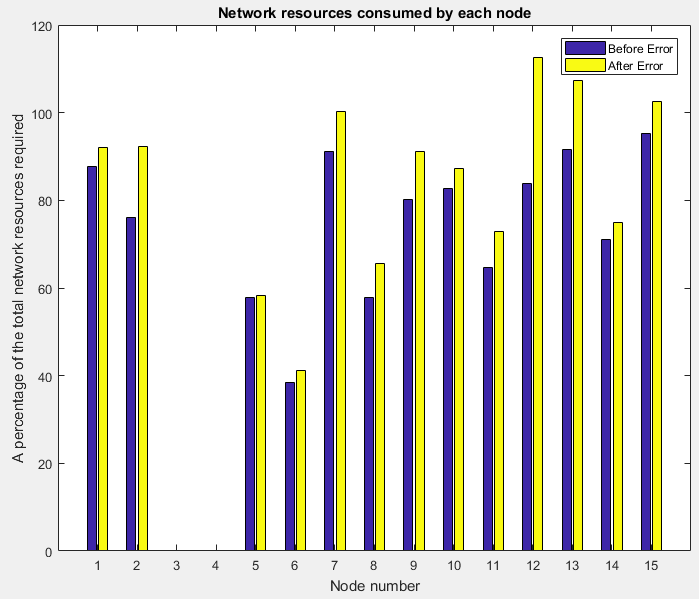
  
Figure 15

Figure 16 is the screenshot of the final plot at time 30 for scenario 3.

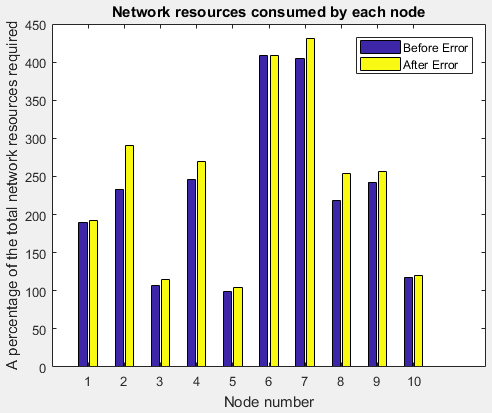
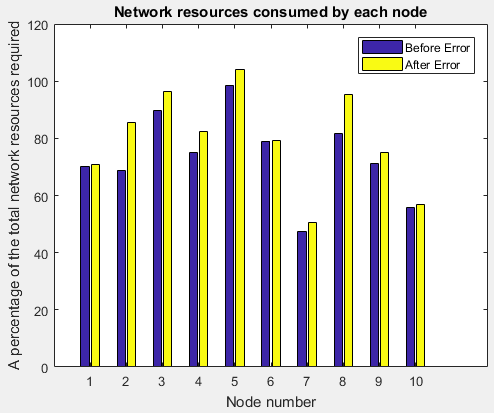
  
Figure 16

Figure 17 is the screenshot of the final plot at time 30 for scenario 4.

  
Figure 17

## 3.3 - Text output from .csv files

The following tables show the text output from the .csv files for the results of scenarios 1 formatted into a table. Scenario 1 Data Table

|  |  |  |
| --- | --- | --- |
|  | Network link utilisation before errors as % of the maximum available | Network link utilisation after errors as % of the maximum available |
| Node 1 | 87.645271 | 92.122501 |
| Node 2 | 76.134278 | 92.388018 |
| Node 3 | 0.000000 | 0.000000 |
| Node 4 | 0.000000 | 0.000000 |
| Node 5 | 57.719765 | 58.197041 |
| Node 6 | 38.520492 | 41.229102 |
| Node 7 | 91.252252 | 100.341109 |
| Node 8 | 127.410956 | 144.498510 |
| Node 9 | 80.164089 | 91.226490 |
| Node 10 | 82.800390 | 87.170248 |
| Node 11 | 126.967820 | 143.353796 |
| Node 12 | 146.301182 | 196.691881 |
| Node 13 | 91.625335 | 107.352560 |
| Node 14 | 70.964001 | 74.880153 |
| Node 15 | 95.339566 | 102.653484 |

The following table show the text output from the .csv files for the results of scenarios 2 formatted into a table. Scenario 2 Data Table

|  |  |  |
| --- | --- | --- |
|  | Network link utilisation before errors as % of the maximum available | Network link utilisation after errors as % of the maximum available |
| Node 1 | 87.645271 | 92.122501 |
| Node 2 | 76.134278 | 92.388018 |
| Node 3 | 0.000000 | 0.000000 |
| Node 4 | 0.000000 | 0.000000 |
| Node 5 | 57.719765 | 58.197041 |
| Node 6 | 38.520492 | 41.229102 |
| Node 7 | 91.252252 | 100.341109 |
| Node 8 | 57.733741 | 65.476627 |
| Node 9 | 80.164089 | 91.226490 |
| Node 10 | 82.800390 | 87.170248 |
| Node 11 | 64.635997 | 72.977668 |
| Node 12 | 83.807027 | 112.672786 |
| Node 13 | 91.625335 | 107.352560 |
| Node 14 | 70.964001 | 74.880153 |
| Node 15 | 95.339566 | 102.653484 |

## 3.4 - Discussion of the prioritisation scheme

In the simulation, the maximum priority of an application is 10. The generateRandomTopology function will create random number set to decide which node to put an application on and create a random number set for application sending rate and application priority.

The prioritisation scheme in the simulation, determine which node are allowed to send data based on the available network link bandwidth. If the total sending rate without network errors for the given node exceed the maximum Bandwidth for that node, it will deactivate that the lowest priority application in that node.

The advantages of this prioritisation scheme, network resources require for each node will decrease since all application with low priorities will not send any data. This is evidenced by comparing the graph with and without prioritisation where the graph with prioritisation has a lower percentage number of network resources required for each node compared to the graph without prioritisation. This scheme allows us to see the lowest possible network resources required for each node assuming that it has the most efficient network possible.

However, the limitation of this prioritisation scheme is the there is no way to tell how any application are disabled in this prioritisation data so the data we obtained in the prioritisation scheme for 10 applications is not for 10 applications since some unknown number of application maybe disabled due to the scheme.

A way in which the algorithm could be improved could be to allow the user to set the number of high priority application for themselves, set a maximum number of application for each node or set a limitation on how number application can be ignore in priority scheme for each node. These suggestion is to be able to extend the freedom of control and knowledge for the user in the simulation without leaving too much information unknown.

# Appendix

## 4.1 - DropTailSim.tcl source code

#Create a simulator object

set ns [new Simulator]

# Define colour scheme for different flow at the beginning after the

# simulator object has been created. Notes: The numeric number will

# be the flow ID for the traffic flows in the simulation later.

$ns color 1 Red

$ns color 2 green

$ns color 3 Yellow

$ns color 4 Black

#Open the nam trace file

set nf [open out1.nam w]

$ns namtrace-all $nf

#Open trace file

set f0 [open throughput.tr w]

set f1 [open queuesize.tr w]

#Define a 'finish' procedure

proc finish {} {

global ns nf f0 f1

$ns flush-trace

#Close the output files

close $nf

close $f0

close $f1

#Execute nam on the trace file

exec nam out1.nam &

#Call Xgraph to display the results

exec xgraph throughput.tr -t "ThroughPut" -geometry 800x400 &

exec xgraph queuesize.tr -t "QueueSize" -geometry 800x400 &

exit 0

}

#Configuring the routing protocol used by the nodes

set r1 [$ns node]

set r2 [$ns node]

$r1 label r1

$r2 label r2

for {set i 1} {$i < 9} {incr i} {

set n($i) [$ns node]

}

for {set i 1} {$i < 9} {incr i} {

$n($i) label node\_$i

}

#Create links between the nodes

$ns duplex-link $n(1) $r1 30Mb 1ms DropTail

$ns duplex-link $n(2) $r1 75Mb 0.4ms DropTail

$ns duplex-link $n(3) $r1 40Mb 1.2ms DropTail

$ns duplex-link $n(4) $r1 65Mb 1.6ms DropTail

$ns duplex-link $r2 $n(5) 55Mb 1.8ms DropTail

$ns duplex-link $r2 $n(6) 70Mb 0.5ms DropTail

$ns duplex-link $r2 $n(7) 80Mb 2.5ms DropTail

$ns duplex-link $r2 $n(8) 95Mb 2.9ms DropTail

$ns duplex-link $r1 $r2 75Mb 3.1ms DropTail

$ns queue-limit $r1 $r2 30

$ns duplex-link-op $r1 $r2 orient 0deg

#Monitor the queue for the link from r1 to r2.

$ns duplex-link-op $r1 $r2 queuePos 0.5

#Create a queue monitor object to monitor internal queue state between node r1 and node r2

set qmon [$ns monitor-queue $r1 $r2 [open queue.tmp w] 0.01]

#Initialize old\_bdeparture to zero

set old\_bdeparture 0

proc record {} {

#Global variables needed to be accessed in this procedure

global ns old\_bdeparture f0 f1 qmon

#Queue monitor objects to be accessed in this procedure

$qmon instvar bdepartures\_

#Set the time after which the procedure should be called again

set time 0.01

#Get the current time

set now [$ns now]

$qmon instvar size\_ pkts\_ barrivals\_ bdepartures\_ parrivals\_ pdepartures\_ bdrops\_ pdrops\_

puts $f1 "$now [$qmon set size\_]"

#Calculate throughput (in Mbps) and write it to the files

puts $f0 "$now [expr ($bdepartures\_-$old\_bdeparture)\*8/$time]"

#Set old\_bdeparture to new value

set old\_bdeparture $bdepartures\_

#Re-schedule the procedure

$ns at [expr $now+$time] "record"

}

for {set i 1} {$i < 5} {incr i} {

set tcp($i) [new Agent/TCP/Newreno]

$tcp($i) set class\_ $i

$ns attach-agent $n($i) $tcp($i)

set sink($i) [new Agent/TCPSink]

set j [expr $i+4]

$ns attach-agent $n($j) $sink($i)

$ns connect $tcp($i) $sink($i)

#$tcp1 set fid\_ 1

set ftp($i) [new Application/FTP]

$ftp($i) set packetSize\_ 1000

#$ftp set interval\_ 0.005

$ftp($i) attach-agent $tcp($i)

#$ftp1 set type\_ FTP

}

#Schedule procedure "record" to get throughput

$ns at 0.0 "record"

#Schedule events for the CBR traffic

$ns at 0.0 "$ftp(1) start"

$ns at 0.0 "$ftp(2) start"

$ns at 0.0 "$ftp(3) start"

$ns at 0.0 "$ftp(4) start"

#$ns at 10.0 "$ftp1 stop"

#$ns at 10.0 "$ftp2 stop"

#$ns at 10.0 "$ftp3 stop"

#$ns at 10.0 "$ftp4 stop"

#Call the finish procedure after 10 seconds of simulation time

$ns at 50.0 "finish"

#Run the simulation

$ns run

## 4.2 - REDSim.tcl source code

#Create a simulator object

set ns [new Simulator]

# Define colour scheme for different flow at the beginning after the

# simulator object has been created. Notes: The numeric number will

# be the flow ID for the traffic flows in the simulation later.

$ns color 1 Red

$ns color 2 green

$ns color 3 Yellow

$ns color 4 Black

#Open the nam trace file

set nf [open out2.nam w]

$ns namtrace-all $nf

#Open trace file

set f0 [open throughput.tr w]

set f1 [open queuesize.tr w]

#Define a 'finish' procedure

proc finish {} {

global ns nf f0 f1

$ns flush-trace

#Close the output files

close $nf

close $f0

close $f1

#Execute nam on the trace file

exec nam out2.nam &

#Call Xgraph to display the results

exec xgraph throughput.tr -t "ThroughPut" -geometry 800x400 &

exec xgraph queuesize.tr -t "QueueSize" -geometry 800x400 &

exit 0

}

#Configuring the routing protocol used by the nodes

set r1 [$ns node]

set r2 [$ns node]

$r1 label r1

$r2 label r2

for {set i 1} {$i < 9} {incr i} {

set n($i) [$ns node]

}

for {set i 1} {$i < 9} {incr i} {

$n($i) label node\_$i

}

#Create links between the nodes

$ns duplex-link $n(1) $r1 30Mb 1ms DropTail

$ns duplex-link $n(2) $r1 75Mb 0.4ms DropTail

$ns duplex-link $n(3) $r1 40Mb 1.2ms DropTail

$ns duplex-link $n(4) $r1 65Mb 1.6ms DropTail

$ns duplex-link $r2 $n(5) 55Mb 1.8ms DropTail

$ns duplex-link $r2 $n(6) 70Mb 0.5ms DropTail

$ns duplex-link $r2 $n(7) 80Mb 2.5ms DropTail

$ns duplex-link $r2 $n(8) 95Mb 2.9ms DropTail

$ns duplex-link $r1 $r2 75Mb 3.1ms RED

$ns queue-limit $r1 $r2 30

$ns duplex-link-op $r1 $r2 orient 0deg

#Monitor the queue for the link from r1 to r2.

$ns duplex-link-op $r1 $r2 queuePos 0.5

#Create a queue monitor object to monitor internal queue state between node r1 and

#node r2

set qmon [$ns monitor-queue $r1 $r2 [open queue.tmp w] 0.01]

#Initialize old\_bdeparture to zero

set old\_bdeparture 0

proc record {} {

#Global variables needed to be accessed in this procedure

global ns old\_bdeparture f0 f1 qmon

#Queue monitor objects to be accessed in this procedure

$qmon instvar bdepartures\_

#Set the time after which the procedure should be called again

set time 0.01

#Get the current time

set now [$ns now]

$qmon instvar size\_ pkts\_ barrivals\_ bdepartures\_ parrivals\_ pdepartures\_ bdrops\_ pdrops\_

puts $f1 "$now [$qmon set size\_]"

#Calculate throughput (in Mbps) and write it to the files

puts $f0 "$now [expr ($bdepartures\_-$old\_bdeparture)\*8/$time]"

#Set old\_bdeparture to new value

set old\_bdeparture $bdepartures\_

#Re-schedule the procedure

$ns at [expr $now+$time] "record"

}

for {set i 1} {$i < 5} {incr i} {

set tcp($i) [new Agent/TCP/Newreno]

$tcp($i) set class\_ $i

$ns attach-agent $n($i) $tcp($i)

set sink($i) [new Agent/TCPSink]

set j [expr $i+4]

$ns attach-agent $n($j) $sink($i)

$ns connect $tcp($i) $sink($i)

#$tcp1 set fid\_ 1

set ftp($i) [new Application/FTP]

$ftp($i) set packetSize\_ 1000

#$ftp set interval\_ 0.005

$ftp($i) attach-agent $tcp($i)

#$ftp1 set type\_ FTP

}

#Schedule procedure "record" to get throughput

$ns at 0.0 "record"

#Schedule events for the CBR traffic

$ns at 0.0 "$ftp(1) start"

$ns at 0.0 "$ftp(2) start"

$ns at 0.0 "$ftp(3) start"

$ns at 0.0 "$ftp(4) start"

#$ns at 10.0 "$ftp1 stop"

#$ns at 10.0 "$ftp2 stop"

#$ns at 10.0 "$ftp3 stop"

#$ns at 10.0 "$ftp4 stop"

#Call the finish procedure after 10 seconds of simulation time

$ns at 50.0 "finish"

#Run the simulation

$ns run

## 4.3 - MATLAB Source code for Discrete Time M/M/1

function plotsys()

%The three data values to be used

alpha1 = 0.35;

alpha2 = 0.45;

alpha3 = 0.55;

%Data storing

data1 = [];

data2 = [];

data3 = [];

sum = 0;

syms k;

%Iterate through the beta value from 0.7 to 0.9 incrementing in 0.02

for beta = 0.7:0.02:0.9

si = alpha1\*(1-beta)/ beta\*(1-alpha1); %Calculating si formula

pi0 = (1-beta)\*(1 - si)/ (1 - beta\*(1 - si)); %Calculating pi0 formula

sum = symsum(k\*(pi0\*(si^k/(1-beta))),k,0,Inf); %Sum of k\*PI\_k from k=0 to infinity

w = sum / alpha1; %Calculate the mean

data1 = [data1 [beta;w]]; %Storing the mean value

%Similarly for the other two values

si = alpha2\*(1-beta)/ beta\*(1-alpha2);

pi0 = (1-beta)\*(1 - si)/ (1 - beta\*(1 - si));

sum = symsum(k\*(pi0\*(si^k/(1-beta))),k,0,Inf);

w = sum / alpha2;

data2 = [data2 [beta;w]];

si = alpha3\*(1-beta)/ beta\*(1-alpha3);

pi0 = (1-beta)\*(1 - si)/ (1 - beta\*(1 - si));

sum = symsum(k\*(pi0\*(si^k/(1-beta))),k,0,Inf);

w = sum / alpha3;

data3 = [data3 [beta;w]];

end

figure

hold on;

plot(data1(1,:), data1(2,:),'-\*');

plot(data2(1,:), data2(2,:),'-d');

plot(data3(1,:), data3(2,:),'-.');

grid on;

title('Discrete Time M/M/1, Showing Mean waiting time for three different alpha')

xlabel('beta, probability that a packet will depart in any time slot')

ylabel('Value of mean waiting time')

legend('alpha = 0.35','alpha = 0.45','alpha = 0.55');

hold off;

%axis([0.7 0.9 0 50]);

End

## 4.4 - MATLAB GUI Source code for Buttons

function num\_nodes\_Callback(hObject, eventdata, handles)

% hObject handle to num\_nodes (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of num\_nodes as text

% str2double(get(hObject,'String')) returns contents of num\_nodes as a double

input = str2num(get(hObject,'String'));

if(isempty(input))

set(hObject,'String','0')

end

guidata(hObject,handles);

function num\_app\_Callback(hObject, eventdata, handles)

% hObject handle to num\_app (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of num\_app as text

% str2double(get(hObject,'String')) returns contents of num\_app as a double

input = str2num(get(hObject,'String'));

if(isempty(input))

set(hObject,'String','0')

end

guidata(hObject,handles);

function seed\_num\_Callback(hObject, eventdata, handles)

% hObject handle to seed\_num (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of seed\_num as text

% str2double(get(hObject,'String')) returns contents of seed\_num as a double

input = str2num(get(hObject,'String'));

if(isempty(input))

set(hObject,'String','0')

end

guidata(hObject,handles);

function sim\_time\_Callback(hObject, eventdata, handles)

% hObject handle to sim\_time (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of sim\_time as text

% str2double(get(hObject,'String')) returns contents of sim\_time as a double

input = str2num(get(hObject,'String'));

if(isempty(input))

set(hObject,'String','0')

end

guidata(hObject,handles);

function prioritise\_button\_Callback(hObject, eventdata, handles)

% hObject handle to prioritise\_button (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

input = get(hObject,'Value');

if(isempty(input))

set(hObject,'Value','0')

end

guidata(hObject,handles);

% Hint: get(hObject,'Value') returns toggle state of prioritise\_button

## 4.5 - MATLAB GUI Source code for plotting results and writing output to .cvs file

figure

bar(networkLinkUsage);

title('Network resources consumed by each node');

xlabel('Node number')

ylabel('A percentage of the total network resources required')

legend('Before Error','After Error');

fid=fopen('courseworkGUIOutput.csv','w');

for i=1:numberOfNodes

fprintf(fid,'%f,%f\n',networkLinkUsage(i,1),networkLinkUsage(i,2));

end

fclose(fid);

## 4.6 - NAM Traces for DropTailSim.tcl

## 4.7 - NAM Traces for REDSim.tcl