



Telecommunication Network Management

NS-2 SIMULATION OF IP NETWORK WITH DIFFERENTIATED SERVICES

1. OBJECTIVE

This laboratory assignment will improve your understanding of some of the effects of service differentiation in IP networks, through simulation in NS-2.

2. SIMULATING WITH NS-2

NS-2 runs under Linux, so boot into the Linux virtual machine available on computers in the lab.

To simulate a network scenario in NS-2, a model of the network must first be created using a flavor of the Tool Command Language (TCL) called oTCL. oTCL is interpreted, that is, the *.tcl file containing the instructions is processed in run-time. Check Appendix A of this document for a short description of the syntax of the oTCL language.

Once a “myModel.tcl” file describing the network has been created it can be simulated in NS-2, by opening a Terminal and running the command:

```
ns ./myModel.tcl
```

A template of a *.tcl file can in be found in the “StartScript.tcl” file available on the virtual campus.

The results of the simulation are stored in two different formats: *.nam and *.tr. The *.nam file is used by the NAM tool to display an animation of the simulation results. The *.tr file contains a trace of all the packet operations at the different nodes, from which the network’s performance indicators can be computed.

3. NETWORK MODEL

3.1 BEST-EFFORT NETWORK

We will start by simulating a simple best-effort network (with FIFO queues), in which the nodes are connected with full-duplex links with different bandwidths, and identical delays, as shown in figure 1.

The traffic that should be generated is defined by the following table.

Source	Sink	Traffic type	Characteristics
s1	dest1	CBR video	2 Mbps constant bit rate (using UDP) Packet size 500 bytes Starts at t = 1 second and ends at t = 45 seconds
s2	dest2	FTP	Sends a 2Mbyte file over TCP starting a t = 5 seconds.

Telecommunication Network Management

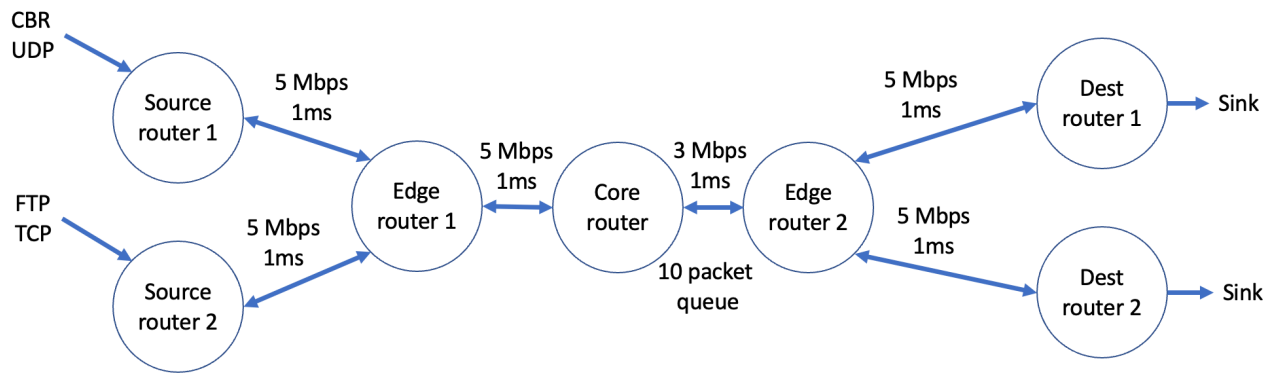


Figure 1. Topology of the best effort network

***The queue size between the ‘Core’ and ‘Edge 2’ router shall be limited to 10 packets.**

Tasks:

- Use the information available in the appendices and the NS-2 manual to define this network in a file named “BestEffort.tcl” based on the “StartScript.tcl” template. Run the simulation in ns2. Note: you can use the “Editor de textos” to modify the tcl file; the terminal application is called “Tilix”.
- In NAM, note that the nodes are not displayed as in Figure 1. Include the necessary commands in the TCL script to properly orient the links. You can find the required commands in section 49.2 of ns manual.
- Finally, observe in NAM that some packets are dropped.

The simulation will also generate an “out.tr” file with a format like the following:

```

Column 1 2 3 4 5 6 7 8 9 10 11 12
+ 1 0 2 cbr 210 ----- 0 0.0 3.0 0 0
- 1 0 2 cbr 210 ----- 0 0.0 3.0 0 0

```

The format is relatively straightforward:

- Column 1 is the event type – see also figure 2.
 - + a packet is enqueued.
 - – a packet is dequeued
 - r a packet is received
 - d a packet is dropped.
- Column 2 is the time at which the event occurs.
- Columns 3 and 4 show the node numbers between which the event happens.
- Column 5 is the packet type (cbr, tcp, ack, ...)
- Column 6 is the size of the packet in bytes.
- Column 7 shows flags related to congestion, packet priorities, etc.
- Column 8 is a flow identifier.
- Column 9 and 10 are the source and destination node.
- Column 11 is a sequence number.
- Column 12 is a unique packet identifier assigned by NS2.

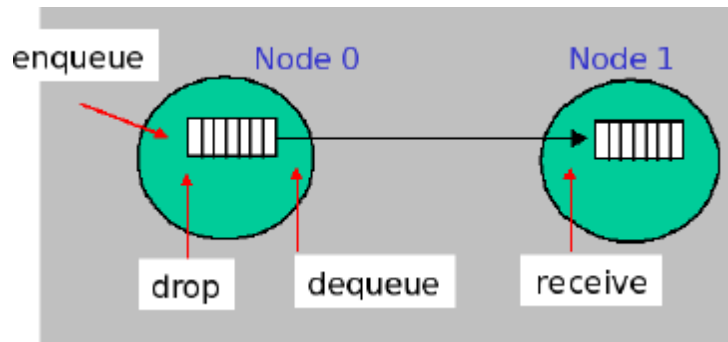


Figure 2. Definition of the enqueue, dequeue, receive and drop operations.

To analyze the out.tr trace file we are going to use a simple Python script (“PostProcessing.py”) available on the virtual campus. You can use the Spyder3 environment to run this script. With this script we can calculate the throughput or the loss rate at a certain node, using the following syntax:

```
from PostProcessing import *
time_axis, computed_rate=rate(trace_file,event_type,to_node,packet_type, granularity)
```

Where:

trace_file is the path of the file we want to analyze.

event_type is the type of event, as described above.

to_node is the identifier of the destination node.

packet_type is the packet type as described above.

granularity is the time window (in seconds) over which averages are taken.

For instance, the throughput of the CBR source, averaged every 200ms for node dest1 (node 5 – this number may be different for you: check in nam) can be computed using the following command in Python:


```
from PostProcessing import *
time_axis,computed_rate=rate(trace_file='DiffServ/out.tr',
                             event_type_requested='r',
                             to_node_requested=5,
                             packet_type_requested='cbr',
                             granularity=0.2)
```

We can make a plot of the data using the **matplotlib** library:

```
import matplotlib.pyplot as plt
plt.plot(time_axis,computed_rate)
```

The module matplotlib.pyplot has the same plotting functions as Matlab and with a very similar syntax. For example, to set the x axis label we use plt.xlabel('Time (s)').

To generate the plots in a separate window type “%matplotlib qt” in the Spyder 3 terminal.

Save all the relevant plots that you generate. To do so, click on the save icon ().

Calculate and plot the following:

- *Throughput for FTP and CBR as function of time, i.e., how many Mbps reach the destination.*
- *Loss rate for FTP and CBR as a function of time, i.e., (lost packets)/(total of transmitted packets).*
- *Average throughput and average loss rate for FTP and CBR.*
- *Find a quantitative explanation for the results you observe.*

It can be helpful to create a script to avoid re-typing commands. To execute a system command from Python you can use the **system** function:

```
import os  
  
os.system("ns BestEffort_model.tcl")
```

Be aware that this will not print the command output (such as error messages).

3.2 DIFFSERV NETWORK

The DiffServ module implemented in NS-2 is based on the “Assured Forwarding” per hop behavior. Within assured forwarding four classes are defined (AF1 to AF4), which are implemented as four FIFO queues. Within each queue three drop precedence levels can be specified.

In this section we will create a new script “DiffServ.tcl”, to simulate the DiffServ network shown in Figure 3. Note that is sufficient to implement DiffServ in one direction in the network (from Sources to Sinks), whereas the traffic in the opposite direction can be treated as best effort.

The queues size between the ‘core’ and ‘e2’ router shall be limited to 10 packets.

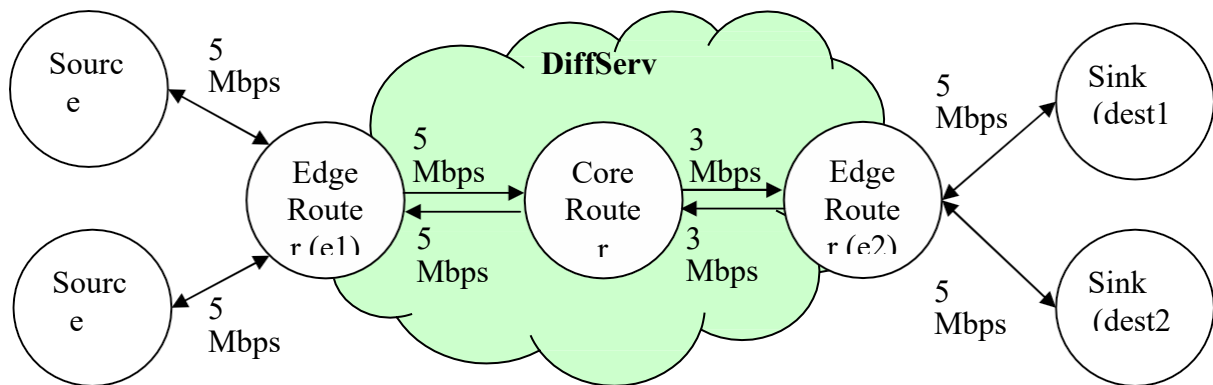


Figura 3. Network architecture with DiffServ.

The syntax to create the DiffServ links is as follows:

```
$ns simplex-link $e1 $core 5Mb 1ms dsRED/edge;
$ns simplex-link $core $e2 3Mb 1ms dsRED/core;
```

In NS-2 the QoS behavior is implemented in queues for each link.

```
set q [$ns link $n1 $n2] queue]
```

In each link we include two FIFO queues, and within these, 2 drop precedence levels, giving a total of four PHB. For example:

```
$q set numQueues_ 2      #two FIFO queues numbered '0' and '1'; the _ is not a typo.
$q setNumPrec 2          #two drop precedence levels numbered '0' and '1'; there is no
                           space between set and Num
```

Packet classification is based on the source and destination nodes. CBR traffic will be assigned the code 10, and FTP traffic will be assigned the code 20. This code is equivalent to the DSCP. Traffic conditioning is performed with a Token Bucket algorithm:

```
$q addPolicyEntry [$s1 id] [$dest id] TokenBucket 10 $cir $cbs
```

Here, CIR is the committed information rate and CBS is the committed burst size.

■ Use the ns2 manual to find out what the units of CIR and CBS are.

We are going to use the following values for the two traffic types:

Traffic type	CIR	CBS
CBR	2500000	10000
FTP	1000000	10000

The packets that do not fulfill these criteria will be down-graded, and will be assigned codes 11 (for CBR) and 21 (for FTP):

```
$q addPolicerEntry TokenBucket 10 11
$q addPolicerEntry TokenBucket 20 21
```

Based on their DSCP packets will be placed into different queues:

```
$q addPHBEntry <codepoint> <queueNumber> <precedenceLevel>
$q addPHBEntry 10 0 0 #packets marked with 10 are sent to queue 0, with precedence level 0
```

To schedule the different FIFO queues, we will first use priority scheduler

```
$q setSchedulerMode PRI #yes its really Scheduler, not Scheduler
$q addQueueRate 0 cir_CBR
$q addQueueRate 1 cir_FTP
```

The dropping mechanism we are going to use is Weighted Random Early Detection (WRED). An average packet size has to be set before WRED can be configured.

```
$q meanPktSize 1000
```

■ Use the ns2 manual to find out what the units the meanPktSize is given.

Each “queue” is configured with different RED parameters:

```
$q configQ <queueNum> <virtualQueueNum> <minTh> <maxTh> <maxP>
$q configQ 0 0 20 40 0.02
$q configQ 0 1 10 20 0.10
$q configQ 1 0 20 40 0.02
$q configQ 1 1 10 20 0.10
```

Calculate and plot the following:

- Throughput for FTP and CBR as function of time.
- Loss rate for FTP and CBR as a function of time.
- Average throughput and average loss rate for FTP and CBR.
- Find a quantitative explanation for the results you observe.
- Compare your results with those obtained with the Best Effort scheme.
- Repeat your analysis with a round robin scheduler.
- Repeat your analysis with a weighted round robin scheduler. What are the optimum scheduler weights?

APPENDIX A – oTCL syntax overview

Operation	Syntax	Example
Assign literal value to variable	<code>set varname value</code>	<code># assign a number</code> <code>set num 43</code> <code># assign a string</code> <code>set str hello</code>
Get variable value	<code>\$var</code>	<code>puts \$str</code>
Assign variable b value to variable a	<code>set a \$b</code>	
Assign the result of a command to a variable	<code>set var [command ..]</code>	<code>set x [expr \$num + 5]</code>
Instantiate an object obj of class MyClass	<code>set obj [new MyClass]</code>	<code>set ns [new Simulator]</code>
Call class method	<code>\$obj method arg1 arg2 ...</code>	<code>\$q addQueueRate 0 cir CBR</code>
Assign a new value to a class attribute	<code>\$obj set attribute name <value></code>	<code>\$q set numQueues 2</code>

APPENDIX B – SOME BASIC NS-2 COMMANDS

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

Create an event: `$ns at <time> <event>`

Run the simulation: `$ns run`

Create network nodes `set nodeName [$ns node]`

Creating links `$ns duplex-link $node1 $node2 5Mb 1ms DropTail`
 Bidirectional link
 5 mega bits per second
 1 millisecond delay
 DropTail dropping mechanism.

Limit queue size `$ns queue-limit $node1 $node2 numberOfPackets`

A UDP source `set udpsource [new Agent/UDP]`

A UDP sink `set udpsink [new Agent/Null]`

Attach source to node `$ns attach-agent $node1 $udpsource`

Attach sink to node `$ns attach-agent $node2 $udpsink`

Connect source/sink `$ns connect $udpsource $udpsink`

A TCP source `set tcpsource [new Agent/TCP/Newreno]`

A TCP sink `set tcpsink [new Agent/TCPSink]`

Create FTP traffic `set ftp [new Application/FTP]`

Attach to TCP `$ftp attach-agent $tcpsource`

Start FTP traffic `$ns at <time> "$ftp send <bytes>"`

Create CBR traffic `set cbr [new Application/Traffic/CBR]`
`$cbr set packetSize_ 512 # DO NOT PUT A COMMENT HERE.`
`# the _ is not a typo`
`$cbr set interval_ 0.27 # DO NOT PUT A COMMENT HERE.`
`# the _ is not a typo`

Attach to UPD `$cbr attach-agent $udpsource`

Start UDP traffic `$ns at <time> "$cbr start"`

Show configuration and statistics (only for DiffServ):

`$ns at 48.0 "$qe1_c0 printPolicyTable"`

`$ns at 48.0 "$qe1_c0 printPolicerTable"`

`$ns at 48.0 "$qe1_c0 printPHBTable"`

`$ns at 48.0 "$qe1_c0 printStats"`

`$ns at 48.0 "$qe1_c0 getAverage 0"`

APPENDIX C – INSTALLING THE SOFTWARE

The software that you need for this assignment is already pre-installed on the computers in lab 1.1.6. If you want to work on your own computer, you should install Ubuntu 20.04 LTS. Then, from the terminal, run

```
sudo apt update
```

```
sudo apt install ns2 python3 python3-numpy python3-matplotlib python3-pandas spyder3 nam
```