NS2 SIMULATION, PART 1

General

The objectives of this assignment (Part 1) are as follows:

[1] to get acquainted with the network simulator ns2, which includes writing simulation scripts and work with trace files;

[2] to get a deeper understanding of TCP congestion control algorithms.

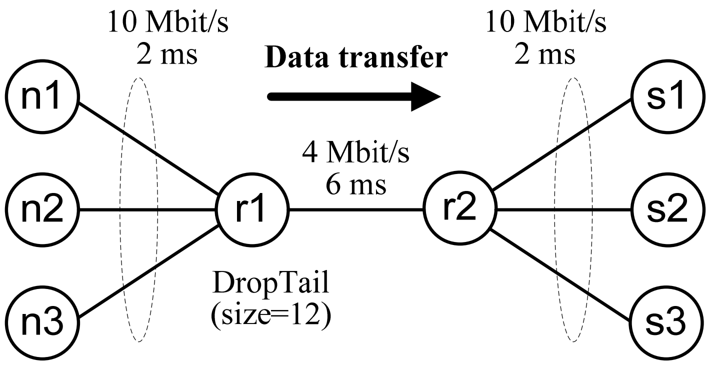
Your task is to write a Tcl script according to these instructions and perform some simulations with it. The instructions contain some hints; however it is up to you to decide how to carry out tasks and either to use provided tips or not.

You can find all required information about ns2 at the official website (including documentation and ns-related mailing lists)
http://www.isi.edu/nsnam/ns/
but a better way to start is to read at first the following tutorials:
http://www.isi.edu/nsnam/ns/tutorial/index.html (Marc Greis's tutorial)
http://nile.wpi.edu/NS/ (NS by example)
http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/ns.htm (NS simulator for beginners)
The purpose of these tutorials is to make it easier for new ns2 users to use this software, to create their own simulation scenarios, and to eventually contribute new modules and add new functionality to the network simulator.

Before starting this assignment it is absolutely essential for you to get acquainted with some of the abovementioned tutorials. Thus, it is presumed that students know the basic terms related to ns2, such as creating nodes and links, and how to use protocol and software agents to create traffic in the simulated network.

The network to be simulated

Create the network topology shown below (also known as "dumbbell topology"). In the figure, every circle represents a node, while lines represent links.



All the links are 10 Mbit/s full-duplex links with 2 ms delay, except for the link between r1-r2: it is 4 Mbit/s full-duplex link with 6 ms delay. Thus, the link r1-r2 is a bottleneck link.

The queuing discipline of all links is DropTail (FIFO).

Limit the maximum queue size at r1-r2 link to 12 packets.

Attach one-way TCP senders to the nodes n1, n2, and n3. Use TCP Tahoe agent at n1, TCP Reno at n2, and TCP NewReno (Slow-but-Steady variant) at n3 node.

Use the following TCP parameters:

```
#TCP Tahoe parameters
Agent/TCP set maxcwnd_ 20
Agent/TCP set windowInit_ 2
Agent/TCP set packetSize_ 1460
Agent/TCP set tcpTick_ 0.001

#TCP Reno parameters
Agent/TCP/Reno set maxcwnd_ 20
Agent/TCP/Reno set windowInit_ 2
Agent/TCP/Reno set packetSize_ 1460
Agent/TCP/Reno set tcpTick_ 0.001

#TCP NewReno parameters (Slow-but-Steady)
Agent/TCP/Newreno set maxcwnd_ 20
Agent/TCP/Newreno set windowInit_ 2
Agent/TCP/Newreno set packetSize_ 1460
Agent/TCP/Newreno set tcpTick_ 0.001
Agent/TCP/Newreno set newreno_changes_ 0
Agent/TCP/Newreno set newreno_changes1_ 0
```

Let the sink for source n1 be s1, the sink for n2 be s2, and the sink for n3 be s3.

Use `Agent/TCPSink` as a TCP receiver.

Set different colours for data flows. Generate TCP traffic using FTP application.

Start sending data from all sources at the same time, and stop each flow after 4 seconds, i.e. the duration of the simulation is 4 seconds.

Note: There are two types of agent variables: instance variables and class variables. Changing the instance variable of a particular agent only affects the values used by that agent, for example:

```
$tcp set maxcwnd_ 20
#Changes maxcwnd_ for the $tcp object only
```

Changing the class variable changes the default value for all agents that are created subsequently(!), for example:

```
Agent/TCP set maxcwnd_ 20
#Changes the class variable for all subsequent TCP Tahoe
senders
```

Thus, all TCP parameters given above should be declared before the creating their corresponding agents (e.g., at the very beginning of the script).

Tasks

[1] Trace evolution of congestion window size of TCP Tahoe/Reno/NewReno senders and plot a graph, where the values of Tahoe/Reno/NewReno windows are on the y-axis and x-axis is time line.

Although you can get the values of cwnd by using "`$tcp trace cwnd_`" command and import them into Excel, Mathcad or any other mathematical program, but a much more elegant way would be to plot a graph using xgraph utility. See example script below (assuming tcp1, tcp2, and tcp3 agents).

```
set nf [open out.nam w]
$ns namtrace-all $nf

set tcpTahoe [open tcpTahoe w]
set tcpReno [open tcpReno w]
set tcpNewReno [open tcpNewReno w]

#Plotting window size with xgraph
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file"
}
```

```
#Define the finish procedure that closes the trace files and
starts nam and xgraph
proc finish {} {
     global ns nf tcpTahoe tcpReno tcpNewReno
     $ns flush-trace
     #Close the trace files
     close $nf
     close $tcpTahoe
     close $tcpReno
     close $tcpNewReno
     #Execute nam and xgraph on the trace files
     exec nam out.nam &
     exec xgraph tcpTahoe tcpReno tcpNewReno -geometry 800x400
-t "cwnd" -x "Time, secs" -y "Window, pkts"
     exit 0
}

$ns at 0.0 "plotWindow $tcp1 $tcpTahoe"
$ns at 0.0 "plotWindow $tcp2 $tcpReno"
$ns at 0.0 "plotWindow $tcp3 $tcpNewReno"
```

On a base of simulation results (provide the graph) explain the difference between these types of TCP senders (Tahoe/Reno/NewReno). See RFC 2001, RFC 2581, and RFC 3782 for details.

[2] Monitor the behaviour of the network using nam. You should observe packet drops in the queue of the bottleneck link. Write down the time when the first four packet drops occur. Also define which TCP sender(s) sent these packets.

Note that running the script generates a nam trace file that is used as an input to network animator and can be used for simulation analysis. Each trace line starts with an event (+, -, d, r) descriptor followed by the simulation time (in seconds) of that event. Therefore, it is possible to select all trace lines corresponding to packet drops by performing search for lines starting with “`d -t`”:

```
grep “d -t” out.nam > loss.txt
```

[3] TCP congestion control will gradually eliminate the congestion, and the queue will become empty. Record the time when it happens.

For example, this can be done by tracing the queue size as follows.

```
set queue [open queue w]
set qmon [$ns monitor-queue $r1 $r2 stdout];
[$ns link $r1 $r2] queue-sample-timeout;

proc queueLength {} {
     global ns qmon queue
     set time 0.05
     set length [$qmon set pkts_]
     set now [$ns now]
     puts $queue "$now $length"
     $ns at [expr $now+$time] "queueLength"
}

$ns at 0.0 "queueLength"
```

Tip: Do not forget to declare the trace file `queue` within the finish procedure.

[4] Calculate the number of packets that can be in transit in the network. It can be found as a sum of bandwidth-delay product of the network (BDP) and buffer size:

$$BDP + queue\_size = \frac{RTT_{\min} \times bottleneck\_bandwidth}{MTU} + queue\_size,$$

where $RTT_{\min} = 20\text{ms} = 20 \times 10^{-3}\text{s}$; $bottleneck\_bandwidth = 4\text{Mbit/s} = 4 \times 10^{6}\text{bit/s}$; $MTU = 1500\text{Bytes} = 12 \times 10^{3}\text{bits}$; $queue\_size = 12\text{pkt}$.

Set `maxcwnd_` parameters of the TCP senders (corresponding to receiver window size of real TCP implementation) in such a way that

$$maxcwnd\text{Tahoe} + maxcwnd\text{Reno} + maxcwnd\text{NewReno} \leq BDP + queue\_size\,.$$

Trace evolution of congestion window size of TCP Tahoe/Reno/NewReno senders and plot a graph, where the values of Tahoe/Reno/NewReno windows are on the y-axis and x-axis is time line.
On a base of simulation results (provide the graph) explain the observed behavior of the TCP senders.

[5] Read the paper by M. Mathis, J. Semke, J. Mahdavi, and Teunis Ott, “The Macroscopic Behavior of the Congestion Avoidance Algorithm”.
Restore previous TCP settings (i.e., set `maxcwnd_ 20`).

Start sending data from all sources at the same time, and stop each flow after 10 seconds, i.e. the duration of the simulation is 10 seconds.

Trace number of data packets (`ndatapack_`) sent by TCP NewReno sender. In order to remove transient phase from the analysis examine only the last 8 seconds of the trace.

Calculate TCP NewReno send rate as

$$BW_{\text{onserved}} = \frac{ndatapack}{time}.$$

Use presented in the paper model to calculate TCP NewReno send rate as a function of p. For this you need to trace `rtt_` and `ncwndcuts_` (number of congestion signals). Do not forget to skip transient phase (first 2 seconds). Then

$$p = \frac{ncwndcuts}{ndatapack}.$$

To calculate the average value of RTT copy the awk script provided at the course web page (`a.awk`) to your work directory and run it:

```
awk –f a.awk rtt.txt
```

where `rtt.txt` is your trace file.

Substitute obtained $\overline{RTT}$ and p values to the model and calculate send rate (in pkt/s):

$$BW_{\text{calculated}} = \frac{1}{\overline{RTT}} \frac{\sqrt{3/2}}{\sqrt{p}}.$$

Compare experimental and analytical results. Why $BW_{\text{observed}} < BW_{\text{calculated}}$? Explain.

Report

Provide your report with a cover page containing your name, student number, e-mail, and name of your work directory. Also include script text into the report.

You should leave your report (hard copy only) in the report box 247: Tietotalo Building, 2nd floor, nearby the entrance to the G-wing. Also send it via e-mail to dunaytse@cs.tut.fi.

The report will be graded using pass/fail. The following issues will be evaluated: the correctness of the results, the written part of the report, the presentation, and appropriate commentary of the results.

In order to save disk space, please remove all your trace files after getting notification about successful pass.