Exercises on ”ns-2”

Chadi BARAKAT

INRIA, PLANETE research group
2004, route des Lucioles
06902 Sophia Antipolis, France

Email: Chadi.Barakat@sophia.inria.fr

November 21, 2003

The code provided between the lines is not complete. You are asked to add your own code.

1 Simulation of a M/D/1 queue

Consider a simple network composed of only two nodes (S and D), and one link connecting them. The link connecting the two nodes has a speed of 1Mbps, a propagation delay of 100 ms, and a DropTail buffer at its input of very large size.

```
$ns duplex-link $S $D 1Mb 100ms DropTail
$ns queue-limit $S $D 10000
```

We run on node S a Poisson source of traffic. The destination of the packets being node D. Let λ be the intensity of the Poisson traffic in terms of packets/s. Take λ = 1200 packets/s and set the packet size of the Poisson source to a fixed value 100 bytes.

Write the code of a simulation of this scenario, then run the simulation for 1000 seconds. The Poisson source can be created as follows:

```
set udp [new Agent/UDP]
$ns attach-agent $S $udp
set null [new Agent/Null]
$ns attach-agent $D $null
$ns connect $udp $null
set poisson [new Application/Traffic/Poisson]
$poisson attach-agent $udp
$ns at 0 ‘‘$poisson start’’
```

Don’t forget the `ns run` !

Verify the load of the queue. What is the expected load? You get the load from the simulation by computing the total number of packets that leave the queue, then by dividing it by the total number of packets that would leave the queue if the link was busy all time. Use the queue monitor object to obtain the total number of packets that leave the queue.

```
set monitor [$ns monitor-queue $S $D stdout]
```

We focus on the computation of the average queue length. Compute theoretically this average value. Recall that the average queue length in a M/G/1 queue is equal to

$$\bar{q} = \rho + \frac{\lambda^2 \mathbb{E}\left[\sigma^2\right]}{2(1-\rho)}.$$

This average queue length includes the packet in the server (physically, the packet being transmitted). The QueueMonitor object gives the number of packets waiting in the queue, without counting the packet in the server. You will notice that there is a difference ρ between the theoretical value and the measured value. ρ is the average number of packets occupying the server of the queue.

Now validate this average value using your simulation. You need to write a function that samples the queue length many times during the simulation, then you compute the average over all the samples you obtain.

```
proc queueLength {sum number} {
global ns monitor
set time 0.1
set len [$monitor set pkts_]
set now [$ns now]
set sum [expr $sum+$len]
set number [expr $number+1]
puts ‘‘[expr 1.0*$sum/$number]’’
$ns at [expr $now+$time] "queueLength $sum $number"}
$ns at 0 "queueLength 0 0"
```

As a final exercise on this scenario, write in a file the queue size in packets (on two columns, the first one for time, the second one for queue size), and plot this queue size versus time using `gnuplot`.

2 Simulation of a M/M/1/K queue

We want to simulate a M/M/1/K queue. This queue requires that the service time of packets is exponentially distributed. The Poisson source in ”ns” generates packets of constant size, that we fixe at the beginning of the simulation. We overcome this lack by considering a large number of Poisson sources, with the sizes of their packets generated from an exponential random variable. Why aggregating Poisson sources in this way gives a queue close to M/M/1/K?

The topology of the network is the same as that in the previous section. The only difference is the packet size at the input of the link S-D, which is now set to a finite value, say for example 20 packets.

We want the aggregate traffic to have a total arrival rate λ = 1200 packets/s, and an average packet size equal to 100

bytes. Suppose that we aggregate 1000 Poisson sources. The packet arrival rate of each source is then 1.2 packets/s. The packet size of each source is set by sampling an exponential random variable of average 100 bytes.

We give now the code that allows to generate the 1000 Poisson sources. Note that the total number of agents we can connect to a node is 256, so we have to connect the 1000 sources (and destinations) to more than one node, then connect these nodes to S (to D). While doing that, ensure that the links between the nodes carrying the sources and S do not impact the results of the simulation.

```
set rng [new RNG]              %Random number generator
$rng seed 0
set size [new RandomVariable/Exponential]
$size set avg_ 100
$size use-rng $rng
for {set i 1} {$i<=4} {set i [expr $i+1]} {
set s($i) [$ns node]
set d($i) [$ns node]
$ns duplex-link $s($i) $S 100Mb 0ms DropTail
$ns duplex-link $D $d($i) 100Mb 0ms DropTail
for {set j 1} {$j<=250} {set j [expr $j+1]} {
set udp($i,$j) [new Agent/UDP]
$ns attach-agent $s($i) $udp($i,$j)
set null($i,$j) [new Agent/Null]
$ns attach-agent $d($i) $null($i,$j)
$ns connect $udp($i,$j) $null($i,$j)
set poisson($i,$j) [new Application/Traffic/Poisson]
$poisson($i,$j) attach-agent $udp($i,$j)
$poisson($i,$j) set interval [expr 1.0/1.2]
$poisson($i,$j) set size [expr 1.0*[$size value]]
$ns at 0 ‘‘$poisson($i,$j) start’’
} }
```

First, measure the load of the queue and compare it to the expected value (using the variable `barrivals_` of the QueueMonitor object).

Second, measure the packet loss rate and the average queue length and compare them to what the analysis gives. To measure the packet loss rate, you need to use the variable `pdrops_` of the QueueMonitor object. Recall that the packet loss rate of a M/M/1/K queue is equal to

$$p = \frac{(1-\rho)\rho^K}{1-\rho^{K+1}}.$$

The average queue length is equal to

$$\bar{q} = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}.$$

Using `gnuplot`, plot the queue length versus time.

Repeat the simulation with other seeds for the random number generator. Do you obtain close results? Do you estimate that taking 1000 sources is enough to simulate the M/M/1/K queue?

3 TCP with random losses

Consider a NewReno TCP connection that crosses a network where packets are dropped with probability p. Let RTT be the round-trip time of the connection, and let $rwnd$ be the receiver advertised window. Denote by S the size of TCP packets in bytes. We set S in this section to 500 bytes. The slow start threshold at the beginning of the connection is set to the receiver window. The receiver is supposed to acknowledge every other packet (delay ACK enabled). The objective of this simulation is to study the performance of TCP in such a scenario, particularly when we change the packet loss probability p.

Two approximations exist for TCP throughput in presence of random losses. The first one, called the square root formula, does not account for timeouts, nor for $rwnd$. It tells that the throughput of TCP is equal to,

$$\bar{X} = \frac{S}{RTT}\sqrt{\frac{3}{4p}}.$$

The second approximation accounts for timeouts and $rwnd$. We call it the PFTK formula (in correspondence with the initials of the authors). According to the PFTK formula, the throughput of TCP is equal to

$$\bar{X} = \min\left(\frac{rwnd}{RTT}, \frac{S}{RTT\left(\sqrt{\frac{4p}{3}} + 4\min\left(1, 3\sqrt{\frac{6p}{8}}\right)p(1+32p^2)\right)}\right).$$

We want to validate these two expressions for TCP throughput using "ns-2".

Consider a simple network similar to that studied in Section 1. A TCP connection is run between S and D (no Poisson traffic). What is the bandwidth-delay product of such a network? Set the receiver window and the slow start threshold to the bandwidth-delay product.

```
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $S $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $D $sink
$ns connect $tcp $sink
$tcp set packetSize_ 500

$set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0 "$ftp start"
```

Run the TCP connection for 1000 seconds. According to you, what should be the utilization of the link? What should be the queue size at the input of the link? If the receiver window is set to a very large value, how should we set the buffer at the input of the link in order to fully utilize the available bandwidth? Validate with the simulation your answer about the utilization of the link and the queue size.

Add the code the allows you to plot the window size versus time. Plot this window using `gnuplot`.

We insert now on the link between S and D an object that randomly drops TCP packets with probability p. This object belongs to the class `ErrorModel`. Here is the code:

```
set lossModel [new ErrorModel]
$lossModel set rate_ value-of-p
```

```
$lossModel unit packet
$lossModel drop-target [new Agent/Null]
$set lossyLink [$ns link $S $D]
$lossyLink install-error $lossModel
```

Plot the congestion window of the TCP connection for the two values of p: 10^{-4} and 10^{-2}. Interpret the results.

Change the packet drop probability p between $5 * 10^{-5}$ and $5 * 10^{-1}$. For each probability, compute the utilization of the link, the utilization given by the square root formula, and the utilization given by the PFTK formula.

How does the performance of TCP vary when p increases? Explain.

Which one of the two formulas approximates better the throughput of TCP?

Where can we consider the square root formula as a good approximation of TCP throughput, and why it does not work in the other regimes?

4 Fairness and TCP-friendliness

TCP congestion control aims at avoidance the congestion collapse in the Internet and at imposing fairness between flows. With TCP, all flows sharing the same path and having approximately the same round-trip time are known to realize the same throughput. This is done by detection congestion via packet losses and reacting to packet losses by dividing the congestion window by two. However, TCP is known to suffer when competing for the bandwidth with flows using UDP. What is your explanation for this unfairness when competing with UDP? The purpose of this exercise is to study this fairness issue.

Consider the same network as that of Section 2 (1Mbps, 100ms delay, a buffer of 20 packets). Run on this network a New Reno TCP connection of packet size 500 bytes, and of very large receiver window and slow start threshold. Keep this connection running for 1500s. At time 500s, run another TCP connection between S and D for 500s.

We want to plot the throughput of both connections as a function of time. We use to this end the `flow monitor` object described in the course. First we associate a color to both flows (1 and 2):

```
$tcp1 set fid_ 1
$tcp2 set fid_ 2
```

Then we create the flow monitor, we associate it to the buffer, and we get a pointer to its classifier (get the pointer to the classifier couple of seconds after the start of the simulation, you must be sure that at least one packet of each flow have already been monitored).

```
set flowmon [$ns makeflowmon Fid]
set L [$ns link $S $D]
$ns attach-fmon $L $flowmon
set fcl [$flowmon classifier]
#Run this at 1 second
set flowstats1 [$fcl lookup auto 0 0 1]
#And this at 501 seconds
set flowstats1 [$fcl lookup auto 0 0 2]
```

Now you can probe the two objects `flowstats1` and `flowstats2` every 10 seconds, read the number of bytes of each flow that cross the link, divide this number by 10 seconds, compute the throughput, save it in a file, then plot it with gnuplot. What do you conclude? Do both flows share fairly the available bandwidth?

Repeat now the simulation but this time with a CBR UDP flow of 800 kbps instead of the second TCP flow. CBR packets are of 500 bytes each. You must create first the UDP agent then the CBR traffic generator:

```
set cbr [new Application/Traffic/CBR]
$cbr set packet_size_ 500
$cbr set rate_ 800kb
```

Do both flows share fairly the available bandwidth? Who is favored in this scenario?