Coursework 2

Sketch model answer
Network Performance—DJW—2007

The simulator. The simulator for questions 1 and 2 is a standard Markov process simulator. It keeps track of the current state (n_0, n_1, n_2). It runs a fixed number of iterations (usually 500,000). In each iteration it generates two random variables, one specifying the duration to stay in the current state, the other specifying which state to jump to next, as described in lecture notes. The possible next states from (n_0, n_1, n_2) and the rates for each jump are as follows:

Some of you used a single simulator for the entire coursework, the simulator which I'll describe for question 3. That is perfectly fine.

transition	rate
$(n_0 + 1, n_1, n_2)$	ν_0
$(n_0, n_1 + 1, n_2)$	ν_1
$(n_0, n_1, n_2 + 1)$	ν_2
$(n_0 - 1, n_1, n_2)$	$Cy_0 1_{n_0>0}$
$(n_0, n_1 - 1, n_2)$	$C(1 - y_0)1_{n_1>0}$
$(n_0, n_1, n_2 - 1)$	$C(1 - y_0)1_{n_2>0}$

where Cy_0 is the total throughput allocated to flows on route 0, i.e. $Cy_0 = n_0 x_0$ where x_0 is the formula given in the question.

The simulator reports the average queue size for each of the three routes. It can do this by accumulating queue size times duration, i.e. using the formula

$$\text{mean queue on route } i = \frac{\sum_n q_i(n)t(n)}{\sum_n t(n)}$$

where $q_i(n)$ is the number of jobs active on route i during the nth step and $t(n)$ is the duration of stay for the nth step. Alternatively, the the average queue size can be computed by accumulating the queue occupancy at each arrival, i.e. using the formula

$$\text{mean queue on route } i = \frac{\sum_m q_i(m)}{M}$$

where $q_i(m)$ is the number of jobs active on route i at the mth arrival to the system, and M is the total number of arrivals simulated.

The second method is based on the PASTA property. You can sample queue size at every arrival of a job on any route, or you can sample q_i only when jobs arrive on route i.

Theory. Consider an M/G/1 processor-sharing queue, where arrivals are Poisson with rate ν and service rate is C and job sizes have mean μ. Let $\rho = \nu\mu/C$. We know that if $\rho \geq 1$ then the queue is unstable. We know that if $\rho < 1$ then the queue is stable, and the mean number of active jobs is equal to $\rho/(1-\rho)$. We might reasonably expect this two-link system to have similar properties.

What exactly is instability? To illustrate this, you could plot two simulation traces, i.e. plot the number of active jobs as a function of time, one at say $C = 0.5$ and one at say $C = 1.5$. When C is low, the number of active jobs seems to increase linearly with time. Therefore, the longer I run the simulation, the larger the value I will get when I measure the mean number of active jobs. In other words, the very idea of a mean number of active jobs is meaningless when the system is unstable (unless we specify the simulation time over which we measured it). You should still show the measured mean number of active jobs in my output, but the numbers should be interpreted with caution when the system is unstable.

Based on the single-link M/G/1 processor sharing model, we expect instability when $C \leq \mu\nu$, in other words when the total rate at which work arrives (in bits/sec) is more than the service capacity. If we naively apply this to the two-link model, we might expect

Many students correctly referred to the M/G/1 or M/M/1 processor sharing queues, and understood that $\rho \leq 1$ means instability, and concluded that $C \leq 0.9$ means instability. Fewer stated the $\rho/(1-\rho)$ formula. No one commented that in an unstable system it is meaningless to talk about the "mean number of active flows".

the first link to be unstable when $C \le \nu_0 + \nu_1 = 0.8$, and the second link to be unstable when $C \le \nu_0 + \nu_2 = 0.9$. Putting these together, we might expect the overall system to be unstable when $C \le 0.9$. Note $\mu = 1$ in this coursework.

Simulation outputs for Question 1. I varied C from 0.5 to 1.5 in steps of 0.1. At each value of C, ran 10 runs of 500,000 iterations each. I plotted the mean over the 10 runs, and the confidence interval, using the R function bwplot2. The measurements agree with my supposition that the system is stable for $C > 0.9$ and unstable for $C \le 0.9$.

As discussed above, we cannot actually tell whether the system is unstable simply by reporting the measured average queue size over a fixed-length simulation run. A clever way indicate this might be to plot a graph of the total number of jobs (summed over the three routes) as a function of C, and to include one curve for the result after 100,000 iterations, another curve for 200,000 iterations, and so on. For $C > 1$ the number of iterations makes very little difference to your answer. For $C < 0.8$ it is clear that the number of iterations makes a difference.

Simulation outputs for Question 2. What value of C should we use? We expect that for $C \le 0.9$ the system will be unstable, and so we need to be cautious how we interpret the answers. But we need $C \approx 0.9$ to get 50–100 active jobs on average. I chose C to be just above 0.9, and I chose 500,000 iterations so as to get an answer in the range 50–100 for $\alpha = 2$.

There is no need to run an exhaustive set of values for α; it is sufficient to look at the values specified in the question. I first of all plotted the total number of jobs, summed over the three routes, with four different α values on the horizontal axis. Then I plotted three sub-plots with the same horizontal axis, one plot for the each route.

It seems that $\alpha = 1, 2, \infty$ give much the same answer (i.e. the error bars overlap). At $\alpha = 0$ the answer is somewhat different: route 0 has many more active jobs, which is not surprising. Furthermore the total mean number of jobs is higher, but you will only see this if your simulation runs are long enough. (In fact, there is theory that says that the $\alpha = 0$ case is stable only for $C > 1.03$.)

Simulator for Question 3. The simulator I described for Question 1 is no longer appropriate, since it only works when file sizes are exponential. What we need here is an event-driven simulator. The simulator needs to keep track of how much work is left for all of the jobs in the system, on each of the routes.

Nearly everyone understood that a Markov process simulator is not appropriate, and roughly 70% of you programmed an appropriate simulator. For every simulator that I saw, the step of calculating T involved iterating through all of the jobs in the system. I will award a bottle of good wine (or a £25 Waterstones gift token) to the first student who gives me a simulator which does not require this iteration.

At each timepoint, calculate the throughput that each of the flows receives, and the time T until the first of the jobs finishes. Also generate an exponential random variable $U \sim \mathrm{Exp}(\nu_0+\nu_1+\nu_2)$ which represents the next scheduled arrival time. Let T be the smaller of these two quantities. Run the system for time $\min(T, U)$, i.e. subtract an appropriate amount of work from each of the active jobs. If $T \le U$ then remove the appropriate job from the system, and go back to the beginning of this paragraph. If $U < T$ then generate a new job (on route 0 with probability $\nu_0/(\nu_0 + \nu_1 + \nu_2)$ etc.) and generate a file size for it, and add it into the list of jobs, then go back to the beginning of this paragraph.

You need to be able to generate a random variable from a heavy-tailed distribution, with mean size 1. The simplest heavy-tailed distribution you know about is the Pareto distribution, for which $P(X \ge x) = x^{-a}$ for $x \ge 1$, where a is some constant. You learnt how to generate Pareto random variables in Coursework 1. The Pareto distribution is not suitable on its own (because all sampled variables are larger than 1, the mean is larger than 1) but you could just generate a Pareto random variable and subtract 1. Choose a experimentally, so as to get mean close to 1. Or you could use a lognormal distribution, or anything else you find on Wikipedia.

Theory. You should mention symmetric queues here. A M/M/1 processor-sharing queue is symmetric, which means that you get exactly the same queue size distribution if you replace the exponential service times by some arbitrary distribution with the same mean, i.e. to get an M/G/1 processor-sharing queue. If this system is symmetric then the mean

Most students understood the point about symmetric queues.

number of jobs will not depend on the distribution you use. If the system is not symmetric then the mean number of jobs might depend on the distribution you use.

(There is theory here, which you may be interested to know: it has been proved that the $\alpha = 1$ case is symmetric, but no one has been able to prove whether or not the $\alpha \neq 1$ cases are symmetric.)

Simulation outputs for Question 3. An appropriate plot here will have the four values of α on the horizontal axis, the mean number of jobs in the system (summed across the three routes) on the vertical axis, and three different curves, one for each of the distributions.

You will likely find that there is not very much difference between the three distributions. However, the heavy-tailed distribution is likely to produce highly variable output, so you may just be unlucky and end up with the answer that heavy-tailed distributions leads to a different mean number of jobs.

I awarded grades on three parts: formulation, execution, interpretation. Your formulation grade is based on identifying the Markov process, understanding that it is not appropriate for Question 3, explaining how your simulator should work, explaining what measurements you will take (maybe referring to PASTA), programming it, generating a heavy-tailed random variable. Also, deciding what it is you have to measure, to answer the question about stability.

Your execution grade is based on running your code, obtaining values, and plotting the results. Your plots should be appropriate for answering the question. I ABSOLUTELY HATE it when, in order to make the comparison that the question asks, I have to flip through several pages of different plots each with different scales. If the question asks you to compare X, Y and Z, you NEED to put X, Y and Z on the same plot. Maybe the horizontal axis will have X, Y and Z. Maybe you will have three curves, one for each of X, Y and Z. Whichever, you have to rely on the eye of the reader more than the brain.

Your interpretation grade is based on whether you brought in the relevant theory, whether the case you argued was properly supported by the plots you presented, and whether there were features of the plots which cried out for explanation but you ignored.