LNMB Course: Algorithmic Methods in Queueing Theory (AlQT)
Lecturers: Rob van der Mei & Stella Kapodistria

Homework Assignment AlQT

Deadline: April 1, 2019

General information: This is the assignment for the course "Algorithmic Methods in Queueing Theory" (AlQT). It requires the implementation of the approaches discussed during the course, and can be made in groups of at most three persons. It is also required to submit a report with your findings.

The report of the assignment should be sent by email to s.kapodistria@tue.nl with the subject "Assignment for course AlQT" before 23:59 on April 1, 2019. To this purpose, you need to create a single pdf file with the answers. Since the assignment requires programming, you can use your favorite programming language or mathematical/statistical software. However, all original program code should be included as an appendix to the pdf file of the report and the original source code, in original format, and should be submitted together with the report (i.e. in the same email as the solutions of the assignment). Zip all files together (pdf file and code files) and submit one single zipped file by email to s.kapodistria@tue.nl. The findings reported in the assignment should be presented in a clear, concise way and the code should be documented and provide sufficient information for confirmation and replication of the results. Ensure that all used sources (such as books, articles, lecture notes or slides, code, etc.) are properly referred to and cited. Moreover, all graphs/tables should have a caption explaining what is depicted and all axes should be labeled.

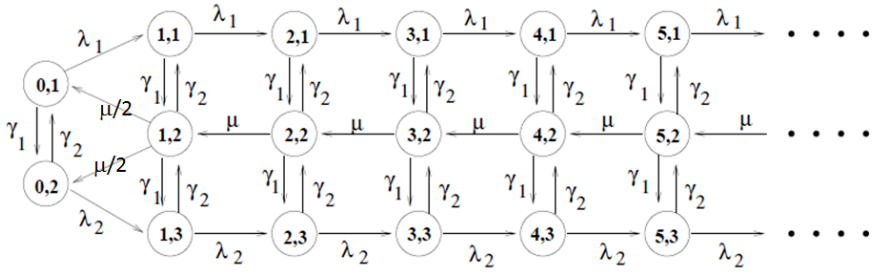
Lastly, but most importantly, create a cover for the report and include the names and affiliations of all the members of the group and the course name.

We wish you good luck, and above all, a lot of fun!

Rob and Stella

Exercise 1: Matrix geometric method for QBD processes

Consider the following example transition diagram for the QBD process that was discussed during the third lecture (by Rob).



Questions:

(1.1) The $\boldsymbol{Q}$-matrix of this QBD process has a block structure. Specify the $\boldsymbol{Q}$-matrix.

(1.2) What are the stability conditions for the QBD process?

(1.3) What are the equilibrium equations for this process?

(1.4) Formulate the equation that characterizes the $\boldsymbol{R}$-matrix of the matrix-geometric method.

(1.5) Assume that the model parameters take the following values:

$$\lambda_1 = \lambda_2 = 2,\ \mu = 6,\ \gamma_1 = \gamma_2 = 3.$$

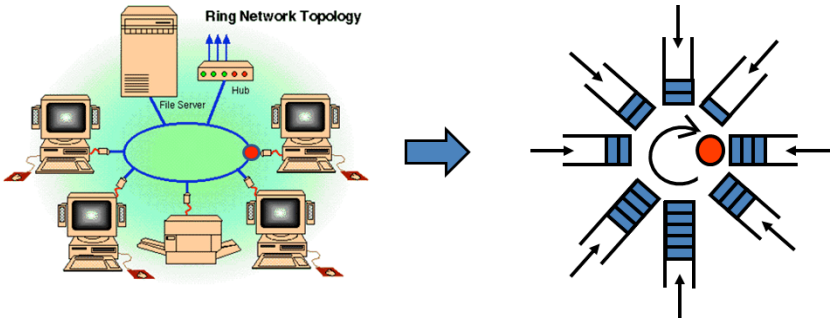
Use a program to calculate the equilibrium probabilities via the matrix-geometric method.

(1.6) Assume that the model parameters take the following values:

$$\lambda_1 = 1,\ \lambda_2 = 4,\ \mu = 6,\ \gamma_1 = 3,\ \gamma_2 = 5.$$

Use a program to calculate the equilibrium probabilities.

Exercise 2: Waiting-time analysis of single-server multi-queue models



Consider the following three-queue polling model: The arrival processes to the queues are Poisson with rates 1/2, 2 and 1, respectively. The service times at queue 1 are exponentially distributed with mean 1. The service times at queue 2 are deterministic with mean 1/10 and the service times at queue 3 are gamma-distributed with mean 1/10 and variance 1. All queues receive gated service. The switchover times from queue 1 to queue 2, from queue 2 to queue 3, and from queue 3 to queue 1 are all independent and exponentially distributed with means 5/100, 1/10 and 1, respectively.

Questions:

(2.1) What is the load of the system? What is the mean switch-over time per cycle? What is the second moment of the total switch-over time per cycle? What are the first two moments of the service times of an arbitrarily chosen customer in the system (i.e., weighted proportionally to the arrival rates)?

(2.2) Formulate the pseudo-conservation law (PCL) for this model. Be clear about all the parameters that you mention.

(2.3) Calculate the expected waiting time at each of the three the queues, either by using the Buffer Occupancy Method (BOM) or the Descendant Set Approach (DSA). To this end, whatever method you use, write down the formulas and equations you use.
Hint: Using the DSA is the easiest. Validate the correctness of your results by comparing the left-hand side and right-hand side of the PCL (which should be the same).

(2.4) Suppose now that queue 2 receives exhaustive instead of gated service. What would you expect to happen to the mean waiting time at each of the three queues? Modify your program accordingly, and run the program to calculate the mean waiting times at each of the queues. Do the results meet your expectations? Explain what you see.

Exercise 3: Performance analysis of fork-join queues

In the big data era, more and more mainstream computing infrastructures become distributively deployed, and inevitably recruit the notion of task replication to facilitate the storing and to minimize errors in the processing of large-scale datasets. This notion of replication of tasks is usually modeled using the so-called fork-join queueing systems: In a fork-join queueing system, a job (upon arrival at a control node) is forked into n replica-tasks, and each replica-task is sent to a single node to be served. Results of finished replica-tasks are summarized at a central join node. When the job arrival rate λ is high, a replica-task may have to wait for service in the corresponding queue node in a first-come-first-serve order. There are mainly two versions of fork-join queues: The (n, k)-*purging* one removes all the remaining replica-tasks of a job from all sub-queues and service stations once the k-th replica-task is served, see Figure 1 for a visualization of the $(3, 1)$-purging fork-join queue. Contrary to the purging version, the *non-purging* one keeps queuing and executing all remaining replica-tasks, this can be in short referred to as the (n, n) fork-join queue.

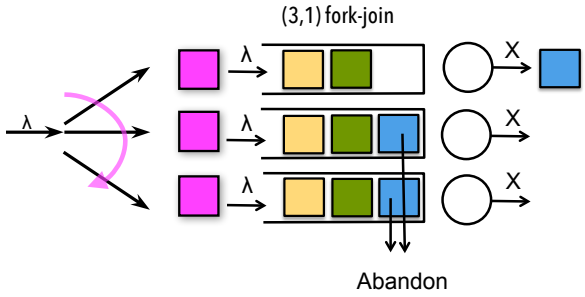


Figure 1: Visualization of the $(3, 1)$-purging fork-join queue (where X marks the random service time at the queue).

For the purpose of this exercise, consider the following basic fork-join models:

The $(2, 1)$ fork-join system: Each incoming job is forked into two replica-tasks and both replica-tasks join a separate first-come first-serve queue with a single server. Thus, there are two separate single-server queues, and one replica of a job enters the first queue and the other replica the other queue. When the service of the first one of the two replica-tasks is finished, the second one is canceled and abandons its queue immediately. Note that the two replicas of the same job are taken into service at the different queues at the same time.

The $(2, 2)$ fork-join system: Each incoming job is forked into two replica-tasks and each replica-task joins a first-come first-serve queue with a single server. For this model, it is assumed that both replica-tasks need to be served for a job to be completed.

Assume that jobs arrive to the system according to a Poisson process at rate λ and upon arrival are forked into two replica-tasks, each joining a server that requires an exponentially distributed service time at rate μ. Note that the $(2, 1)$ fork-join system can also be modeled as an $M/M/1$ queue with

service rate 2μ. Furthermore, the $(2,2)$ fork-join system can be modeled as a two-dimensional stochastic process by considering the corresponding two queue lengths, i.e. $\{(X_1(t), X_2(t)),\ t \geq 0\}$, with $X_i(t)$ the number of customers in the queue (including the customer in service, if any) of the i-th server, $i = 1, 2$. Note that the two-dimensional stochastic process $\{(X_1(t), X_2(t)),\ t \geq 0\}$ is a Markov chain defined on $\mathbb{N}_0 \times \mathbb{N}_0$. This modeling renders the Markov chain to a homogeneous random walk in the quadrant, see Figure 2.

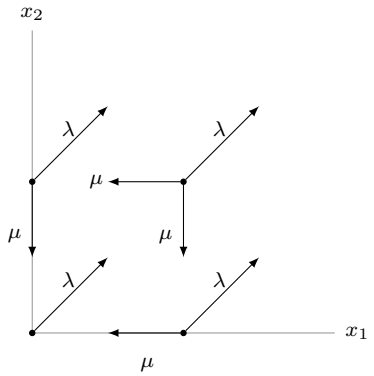


Figure 2: Transition rate diagram on the state space $\mathbb{N}_0 \times \mathbb{N}_0$ for the symmetric (2,2) fork-join system. Only the transitions at a few selected states are depicted as an indication.

Questions:

(3.1) Provide the stability condition, in terms of the system utilization ρ, for the two fork-join systems.

(3.2) For the $(2,2)$ fork-join system investigate if the equilibrium distribution can be computed using the compensation approach by: (i) checking the compensation approach conditions, and (ii) by numerically computing the fractions

$$\frac{\pi_{x_1+1,x_2}}{\pi_{x_1,x_2}} \text{ and } \frac{\pi_{x_1,x_2+1}}{\pi_{x_1,x_2}}. \tag{1}$$

In order to compute the fractions in (1):

(a) truncate the state-space by cancelling all arrivals when the system is in state $x_1, x_2 \geq N$, for a sufficiently large value of N;

(b) truncate the state-space by not replicating arrivals when the system is in state $x_1 \geq N$;

and use appropriate numerical techniques for a few selected values of the parameters.

(3.3) Do the different truncation approaches in Question (3.2) affect the computations of the model at hand? Explain intuitively your findings.

(3.4) Comment on the numerical complexity of the computations in (3.2(a)) and (3.2(b)), accounting for the particular characteristics (e.g., nearest neighbor transitions only) of the model at hand.

(3.5) Compute for the two fork-join systems, for a few selected values of the parameters, the equilibrium distribution, say π_{x_1,x_2}, and the expected response (aka sojourn) time of a job in the system.
Hint: For the the $(2,2)$ fork-join system, if the equilibrium distribution cannot be computed using the compensation approach, then opt for PSA.

(3.6) For the the $(2,2)$ fork-join system, write a fast procedure to compute $\mathbb{E}[X_2 \,|\, X_1 = m]$.
Hint: You can use for this purpose the results in references [1, 2].

(3.7) For the (2,1) and the (2,2) fork-join systems, replicate the computations for the equilibrium distribution and the expected response time in the case that the service distributions are two-phase hyperexponentially distributed with parameters $\mu_1 = 0.1$, $\mu_2 = 1.5$ and $p = 0.5$.

(3.8) Argue whether the (2,1) or the (2,2) fork-join system is better.

References

[1] L. Flatto, "Two Parallel Queues Created by Arrivals with Two Demands II", SIAM Journal on Applied Mathematics, vol. 45, no. 5, pp. 861–878, 1985.

[2] L. Flatto and S. Hahn, "Two Parallel Queues Created by Arrivals with Two Demands I", SIAM Journal on Applied Mathematics, vol. 44, no. 5, pp. 1041–1053, 1984.