Analysis of Queueing Systems with Sample Paths and Simulation

Nicky D. van Foreest

October 13, 2017

Contents

1 Introduction

Motivation and Examples Queueing systems abound, and the analysis and control of queueing systems are major topics in the control, performance evaluation and optimization of production and service systems.

At my local supermarket, for instance, any customer that joins a queue of 4 of more customers get his/her shopping for free. Of course, there are some constraints: at least one of the cashier facilities has to unoccupied by a server and the customers in queue should be equally divided over the cashiers that are open. (And perhaps there are some further rules, of which I am unaware.) The manager that controls the occupation of the cashier positions is focused on keeping $\pi(4)+\pi(5)+\cdots$, i.e., the fraction of customers that see upon arrival a queue length longer or equal than 4, very small. In a sense, this is easy enough: just hire many cashiers. However, the cost of personnel may then outweigh the yearly average cost of paying the customer penalties. Thus, the manager's problem becomes to plan and control the service capacity in such a way that both the penalties and the personnel cost are small.

Fast food restaurants also deal with many interesting queueing situations. Consider, for instance, the making of hamburgers. Typically, hamburgers are made-to-stock, in other words, they are prepared before the actual demand has arrived. Thus, hamburgers in stock can be interpreted as customers in queue waiting for service, where the service time is the time between the arrival of two customers that buy hamburgers. The hamburgers have a typical lifetime, and they have to be scrapped if they remain on the shelf longer than some amount of time. Thus, the waiting time of hamburgers has to be closely monitored. Of course, it is easy to achieve zero scrap cost, simply by keeping no stock at all. However, to prevent lost-sales it is very important to maintain a certain amount of hamburgers on stock. Thus, the manager has to balance the scrap cost against the cost of lost sales. In more formal terms, the problem is to choose a policy to prepare hamburgers such that the cost of excess waiting time (scrap) is balanced against the cost of an empty queue (lost sales).

Service systems, such as hospitals, call centers, courts, and so on, have a certain capacity available to serve customers. The performance of such systems is, in part, measured by the total number of jobs processed per year and the fraction of jobs processed within a certain time between receiving and closing the job. Here the problem is to organize the capacity such that the sojourn time, i.e., the typical time a job spends in the system, does not exceed some threshold, and such that the system achieves a certain throughput, i.e., jobs served per year.

Clearly, all the above systems can be seen as queueing systems that have to be monitored and controlled to achieve a certain performance. The performance analysis of such systems can, typically, be characterized by the following performance measures:

1. The fraction of time $p(n)$ that the system contains n customers. In particular, $1-p(0)$, i.e., the fraction of time the system contains jobs, is important, as this is a measure of the time-average occupancy of the servers, hence related to personnel cost.

2. The fraction of customers $\pi(n)$ that 'see upon arrival' the system with n customers. This measure relates to customer perception and lost sales, i.e., fractions of arriving customers

that do not enter the system.

3. The average, variance, and/or distribution of the waiting time.

4. The average, variance, and/or distribution of the number of customers in the system.

Here the system can be anything that is capable of holding jobs, such as a queue, the server(s), an entire court house, patients waiting for an MRI scan in a hospital, and so on.

It is important to realize that a queueing system can, typically, be decomposed into *two subsystems*, the queue itself and the service system. Thus, we are concerned with three types of waiting: waiting in queue, i.e., *queueing time*, waiting while being in service, i.e., the *service time*, and the total waiting time in the system, i.e., the *sojourn time*.

Organization In these notes we will be primarily concerned with making models of queueing systems such that we can compute or estimate the above performance measures. Part of our work is to derive analytic models. The benefit of such models is that they offer structural insights into the behavior of the system and scaling laws, such as that the average waiting time scales (more or less) linearly in the variance of the service times of individual customers. However, these models have severe shortcomings when it comes to analyzing real queueing systems, in particular when particular control rules have to be assessed. Consider, for example, the service process at a check-in desk of KLM. Business customers and economy customers are served by two separate queueing systems. The business customers are served by one server, server A say, while the economy class customers by three servers, say. What would happen to the sojourn time of the business customers if server A would be allowed to serve economy class customers when the business queue is empty? For the analysis of such cases simulation is a very useful and natural approach.

In the first part of these notes we concentrate on the analysis of *sample paths of a queueing process*. We assume that a typical sample path captures the 'normal' stochastic behavior of the system. This sample-path approach has two advantages. In the first place, most of the theoretical results follow from very concrete aspects of these sample paths. Second, the analysis of sample-paths carries over right away to simulation. In fact, simulation of a queueing system offers us one (or more) sample paths, and based on such sample paths we derive behavioral and statistical properties of the system. Thus, the performance measures defined for sample paths are precisely those used for simulation. Our aim is not to provide rigorous proofs for all results derived below; for the proofs and further background discussion we refer to **?**. As a consequence we tacitly assume in the remainder that results derived from the (long-run) analysis of a particular sample path are equal to their 'probabilistic counterpart'.

In the second part we construct algorithms to analyze open and closed queueing networks. Many of the sample path results developed for the single-station case can be applied to these networks. As such, theory, simulation and algorithms form a nicely round out part of work. For this part we refer to book of Prof. Zijm; the present set of notes augment the discussion there.

Exercises I urge you to make *all* exercises in this set of notes. Many exercises require many of the tools you learned previously in courses on calculus, probability, and linear algebra. Here you can see them applied. Moreover, many of these tools will be useful for other, future, courses. Thus, the investments made here will pay off for the rest of your (student) career. Moreover, the exercises are meant to *illustrate* the material and to force you to *think* about it. Thus, the main text does not contain many examples; the exercises form the examples.

You'll notice that many of these problems are quite difficult, often not because the problem itself is difficult, but because you need to combine a substantial amount of knowledge all at the same time. All this takes time and effort. Next to this, I did not include the exercises with the intention that you would find them easy. The problems should be doable, but hard.

The solution manual is meant to prevent you from getting stuck and to help you increase your knowledge of probability, linear algebra, programming (analysis with computer support), and queueing in particular. Thus, read the solutions very carefully.

As a guideline to making the exercises I recommend the following approach. First read the notes. Then attempt to make a exercises for 10 minutes or so by yourself. If by that time you have not obtained a good idea on how to approach the problem, check the solution manual. Once you have understood the solution, try to repeat the arguments *with the solution manual closed*.

Symbols The meaning of the symbols in the margin of pages are as follows:

- The symbol in the margin means that you have to memorize the *emphasized concepts*.
- The symbol in the margin means that this question has a *hint*.
- The symbol in the margin means that this question or its solution requires still some *work on my part*; you can skip it.

Acknowledgments I would like to acknowledge dr. J.W. Nieuwenhuis for our many discussions on the formal aspects of queueing theory and prof. dr. W.H.M. Zijm for allowing me to use the first few chapters of his book. Finally, without **?** I could not have written these notes.

2 Single-Station Queueing Systems

2.1 M(n)/M(n)/1 Queue

As it turns out, many more single-server queueing situations can be modeled and analyzed by making a judicious choice of $\lambda(n)$ and $\mu(n)$ in (**??**), to wit, $\lambda(n)p(n) = \mu(n+1)p(n+1)$. For these queueing systems we just present the results. In the exercises we ask you to derive the formulas.

For the $M/M/1/K$, i.e., a system that cannot contain more than K jobs, take

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n < K, \\ 0, & \text{if } n \geq K, \end{cases}$$
$$\mu(n) = \mu.$$

Then,

$$p(n) = \frac{\rho^n}{G}, \quad 0 \leq n \leq K, \tag{2.1a}$$
$$G = \frac{1-\rho^{K+1}}{1-\rho}, \tag{2.1b}$$
$$P_{\text{loss}} = \mathsf{P}\{L = K\} = \frac{\rho^K}{G} = \frac{1-\rho}{1-\rho^{K+1}}\rho^K. \tag{2.1c}$$

For the $M/M/c$ queue we can take

$$\lambda(n) = \lambda,$$
$$\mu(n) = \begin{cases} n\mu, & \text{if } n \leq c, \\ c\mu, & \text{if } n \geq c. \end{cases}$$

This model is also known as the *Erlang C*-formula. Define the load as

$$\rho = \frac{\lambda}{c\mu}. \tag{2.2a}$$

Then,

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!}, \quad n = 0, \ldots, c, \tag{2.2b}$$
$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!}, \quad n = c, c+1, \ldots \tag{2.2c}$$
$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!}, \tag{2.2d}$$
$$\mathsf{E}\{L_Q\} = \sum_{n=c}^{\infty}(n-c)p(n) = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2}, \tag{2.2e}$$
$$\mathsf{E}\{L_S\} = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) = \frac{\lambda}{\mu}. \tag{2.2f}$$

From this we can easily get the $M/M/c/c$ queue; here jobs cannot be in queue, only in service, and the system has c servers. This model is also known as the Erlang B-formula and is often used to determine the number of beds at hospitals, where the beds act as servers and the patients as jobs.

From the $M/M/c$ queue (or the $M/M/c/c$ queue) we can also obtain the $M/M/\infty$, i.e., a queueing system with ample servers. By taking the limit $c \to \infty$, note first that in (2.2d),

$$\frac{(c\rho)^c}{(1-\rho)c!} = \frac{(\lambda/\mu)^c}{(1-\rho)c!} \to 0, \quad \text{as } c \to \infty.$$

Hence

$$G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} \to \sum_{n=0}^{\infty} \frac{(c\rho)^n}{n!} = e^{\lambda/\mu}.$$

Next, for $n \geq c$:

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!} = \frac{\rho^n}{G}\frac{c^c}{c!} \to 0, \quad \text{as } c \to \infty,$$

since, from a standard limit $n^n/n! \to 0$ as $n \to \infty$. Therefore, for $n < c$:

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!} = \frac{1}{G}\frac{(\lambda/\mu)^n}{n!} \to e^{-\lambda/\mu}\frac{(\lambda/\mu)^n}{n!}, \quad \text{as } c \to \infty.$$

We see that the number of busy servers in the $M/M/\infty$ queue is Poisson distributed with parameter λ/μ, and $\mathsf{E}\{L\} = \mathsf{E}\{L_S\} = \lambda/\mu$. Observe that now λ/μ has no longer the interpretation of the fraction of time the server(s) are busy; it is the average number of busy servers.

We mention in passing—but do not prove it—that the same results also hold for the $M/G/\infty$ queue with $\lambda\,\mathsf{E}\{S\}$ rather than λ/μ.

Finally, we consider queues with *balking*, that is, queues in which customers leave when they find the queue too long at the moment they arrive. A simple example model with customer balking is given by x

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n = 0, \\ \lambda/2, & \text{if } n = 1, \\ \lambda/4, & \text{if } n = 2, \\ 0, & \text{if } n > 2, \end{cases}$$

and $\mu(n) = \mu$.

Observe that in the example with balking we made a subtle implicit assumption; in Section **??** we elaborate on this assumption. To make the problem clear, note that balking customers *decide at the moment they arrive* to either join or leave; in other words, they decide based on what they 'see upon arrival'. In yet other words, they make decisions based on the state of the system at arrival moments, not on time-averages. However, the notion of $p(n)$ is a long-run *time-average*, and is typically not the same as what customers 'see upon arrival'. As a consequence, the performance measure $\mathsf{P}\{L \leq n\}$ is not necessarily in accordance with the perception of customers. To relate these two 'views', i.e., time-average versus observer-average, we need a new concept, *PASTA*, to be developed in in Section **??**.

Exercises

Exercise 2.1.1. Consider the $M/M/2/2+1$ queue with arrival rate λ and service rate μ.

1. Derive the level-crossing equations for this queueing system.
2. Derive a simple and closed form expressions for the state probabilities in steady state.

Exercise 2.1.2.
1. Derive Eq. (2.1)
2. Show that as $K \to \infty$, the performance measures of the $M/M/1/K$ converge to those of the $M/M/1$ queue.

Exercise 2.1.3. (Multi-server queue with blocking) Consider the $M/M/c/c+K$ queue, try to derive the steady state probabilities $p(0)$, $p(1),\ldots$. You do not have to compute the normalization constant G.

Exercise 2.1.4. Check that Eq.(2.2) for the $M/M/c$ queue reduces to the $M/M/1$ queue if you take $c = 1$.

Exercise 2.1.5. Derive Eq.(2.2) for the $M/M/c$ queue.

Exercise 2.1.6.
1. Show that the $M/M/c$ queue converges to the $M/M/\infty$ queue as $c \to \infty$.
2. Show that the $M/M\infty$ queue is stable for any finite λ.
3. Why is $\mathsf{E}\{L\} = \rho$ for the $M/M/\infty$ queue?

Exercise 2.1.7. (Hall 5.1) Give two examples of systems that ordinarily have a finite buffer size. Give two examples of systems that have a finite calling population.

Exercise 2.1.8. In what way is a queueing system with balking, at level b say, different from a queueing system with finite calling population of size b?

Exercise 2.1.9. (Systems with finite calling population)

1. Derive the steady state probabilities $p(n)$ for a single-server queue with a finite calling population with N members, i.e., jobs that are in service cannot arrive to the system.
2. Check the answer you obtained for the cases $N = 1$ and $N = 2$. Interpret the results.
3. Derive the steady state probabilities $p(n)$ for a queue with a finite calling population with N members and N servers, i.e., the number of servers in the queueing system is equal the size of the calling population.

Exercise 2.1

What would be the difference between a multi-server queue and a single-server queue with a fast server? We can use the formulas for the $M/M/1$ queue and the $M/M/c$ queue to obtain some basic understanding of the difference. To this end, suppose we have an $M/M/3$ queue, with arrival rate $\lambda = 5$ per day and $\mu = 2$ per server, and we compare it to an $M/M/1$ with the same arrival rate but with a service rate of $\mu = 3 \cdot 2 = 6$.

When is it OK to approximate the $M/M/c$ queue by an $M/M/1$ queue with a fast server?

Hints

Hint 2.1.1: Think about what would be the appropriate model choices for $\lambda(n)$ and $\mu(n)$ and use the level-crossing equations $\lambda(n)p(n) = \mu(n+1)p(n+1)$. For instance, realize that $\lambda(3) = 0$: the system cannot contain more than 3 jobs, hence a state with 4 jobs must be impossible. We can achieve that by setting $\lambda(3) = 0$. For the service rate, how many servers are busy when the system contains 2 or more jobs? What does this say about $\mu(k)$ for $k = 2$ or $k = 3$.

Hint 2.1.2: Use that $\sum_{i=0}^{n} x^i = (1 - x^{n+1})/(1 - x)$. BTW, is it necessary for this expression to be true that $|x| < 1$? What should you require for $|x|$ when you want to take the limit $n \to \infty$?

Hint 2.1.3: Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$ and find suitable expressions for $\lambda(n)$ and $\mu(n+1)$.

Hint 2.1.4: Fill in $c = 1$. Realize that this is a check on the formulas.

Hint 2.1.5: Use $\lambda(n)p(n) = \mu(n+1)p(n+1) = \min\{c, n+1\}\mu p(n+1)$.

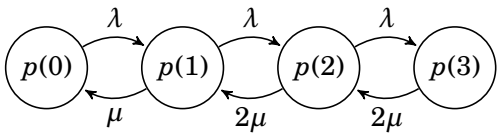
Hint 2.1.6: Use that for any x, $x^n/n! \to 0$ as $n \to \infty$.

Hint 2.1.9: Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$, and realize that for this case $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$.

Hint 2.1.10: Implement the formulas for $\mathsf{E}\{L_Q(M/M/3)\}$ for the $M/M/3$ queue in some computer program (R, excel, python, whatever) and compare this to $\mathsf{E}\{L_Q(M/M/1)\}$ for the fast $M/M/1$ case. Make plots of $\mathsf{E}\{L_Q(M/M/3)\}$ and $\mathsf{E}\{L(M/M/3)\}$ as functions of ρ.

Solutions

Solution 2.1.1: 1. Use this figure. Make sure you understand why $\mu(2) = 2\mu$ and so on.



From this figure it follows right away that:

$$\lambda p(0) = \mu p(1)$$
$$\lambda p(1) = 2\mu p(2)$$
$$\lambda p(2) = 2\mu p(3)$$

2. From the above, with $\rho = \lambda/\mu$:

$$p(1) = \rho p(0),$$
$$p(2) = (\rho/2)p(1) = (\rho^2/2)p(0),$$
$$p(3) = (\rho/2)p(2) = (\rho^3/4)p(0).$$

Now we normalize to find $p(0)$. Thus, we want that:

$$1 = p(0) + p(1) + p(2) + p(3) = p(0)\left(1 + \rho + \frac{\rho^2}{2} + \frac{\rho^3}{4}\right),$$

hence,

$$p(0) = (1 + \rho + \rho^2/2 + \rho^3/4)^{-1}.$$

Solution 2.1.2: Note that

$$1 = \sum_{i=0}^{K} p(i) = p(0) \sum_{i=0}^{K} \rho^i = p(0) \frac{1 - \rho^{K+1}}{1 - \rho}.$$

Thus, for the normalization constant G we take

$$G = \frac{1}{p(0)} = \frac{1 - \rho^{K+1}}{1 - \rho},$$

and the result follows.

To take the limit $K \to \infty$—mind, not the limit $n \to \infty$—, write

$$G = \frac{1 - \rho^{K+1}}{1 - \rho} = \frac{1}{1 - \rho} - \frac{\rho^{K+1}}{1 - \rho}.$$

Since $\rho^{K+1} \to 0$ as $K \to \infty$ (recall, $\rho < 1$), we get

$$G \to \frac{1}{1 - \rho},$$

as $K \to \infty$. Therefore $p(n) = \rho^n / G \to \rho^n (1 - \rho)$, and the latter are the steady-state probabilities of the $M/M/1$ queue. Finally, if the steady state probabilities are the same, the performance measures (which are derived from $p(n)$) must be the same.

Solution 2.1.3: $\lambda(n) \equiv \lambda$ for all $n < c + K$. When $n = c + K$, $\lambda(n) = 0$, since then the system is full, and all arriving jobs will be dropped; in other words, there will still be jobs arriving to the system when $L = c + K$, but these jobs will be rejected, hence cannot generate a transition from state $c + K$ to $c + K + 1$. When $n < c$, $\mu(n) = n\mu$ since only n servers are active/occupied when the system contains n jobs. When $n \geq c$, $\mu(n) = c\mu$. Thus, using $\rho = \lambda/(c\mu)$, for $n < c$,

$$p(n) = \frac{\lambda}{n\mu} p(n-1) = \frac{(\lambda/\mu)^n}{n!} p(0) = \frac{(c\rho)^n}{n!} p(0).$$

For $c \leq n \leq c + K$ and using the above to get $p(c-1)$:

$$\begin{aligned} p(n) &= \frac{\lambda}{c\mu} p(n-1) = \rho p(n-1) = \rho^2 p(n-2) = \ldots \\ &= \rho^{n-c+1} p(c-1) = \rho^{n-c+1} \frac{(c\rho)^{c-1}}{(c-1)!} p(0) \\ &= \rho^n \frac{(c)^{c-1}}{(c-1)!} p(0) = \rho^n \frac{(c)^{c-1} c}{(c-1)! c} p(0) = \frac{c^c \rho^n}{c!} p(0). \end{aligned}$$

The normalization is trivial, numerically at least.

Solution 2.1.4: Take $c = 1$

$$p(0) = \frac{1}{G}\frac{(c\rho)^0}{0!} = \frac{1}{G}, \quad n = 0,\ldots,1-1 \tag{2.3}$$

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!} = \frac{1}{G}\frac{1^1\rho^n}{1!} = \frac{\rho^n}{G}, \quad n = 1,1+1,\ldots \tag{2.4}$$

$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} = \sum_{n=0}^{0}\frac{\rho^0}{0!} + \frac{\rho}{(1-\rho)} = 1 + \frac{\rho}{1-\rho} = \frac{1}{1-\rho} \tag{2.5}$$

$$\mathsf{E}\{L_Q\} = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2} == \frac{\rho}{1/(1-\rho)}\frac{\rho}{(1-\rho)^2} = \frac{\rho^2}{1-\rho} \tag{2.6}$$

$$\mathsf{E}\{L_S\} = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) = p(1) + 1\sum_{n=2}^{\infty} p(n) = 1 - p(0) = \rho. \tag{2.7}$$

Everything is in accordance to the formulas we derived earlier for the $M/M/1$ queue.

Solution 2.1.5: First we use the hint to establish a generic relation for $p(n+1)$:

$$\begin{aligned} p(n+1) &= \frac{\lambda}{\mu(n+1)}p(n) = \frac{\lambda}{\min\{c,n+1\}\mu}p(n) \\ &= \frac{1}{\min\{c,n+1\}}\frac{\lambda}{\mu}p(n) = \frac{1}{\min\{c,n+1\}}(c\rho)p(n) \\ &= \frac{1}{\min\{c,n+1\}\min\{c,n\}}(c\rho)^2p(n-1) = \frac{1}{\Pi_{k=1}^{n+1}\min\{c,k\}}(c\rho)^{n+1}p(0). \end{aligned}$$

Thus, if $n < c$:

$$p(n) = \frac{1}{\Pi_{k=1}^{n}\min\{c,k\}}(c\rho)^n p(0) = \frac{(c\rho)^n}{n!}p(0),$$

since $\min\{c,k\} = k$ when $k < c$. If $n \geq c$:

$$\begin{aligned} p(n) &= \frac{1}{\Pi_{k=1}^{n}\min\{c,k\}}(c\rho)^n p(0) \\ &= \frac{1}{\Pi_{k=1}^{c-1}k \cdot \Pi_{k=c}^{n}c}(c\rho)^n p(0) \\ &= \frac{1}{(c-1)!c^{n-c+1}}c^n\rho^n p(0) \\ &= \frac{1}{c!c^{n-c}}c^n\rho^n p(0) = \\ &= \frac{c^c}{c!}\rho^n p(0). \end{aligned}$$

For the normalization constant G set $p(0) = 1$ for the time being.

$$\begin{aligned} G = \sum_{n=0}^{\infty} p(n) &= \sum_{n=0}^{c-1} p(n) + \sum_{n=c}^{\infty} p(n) \\ &= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \sum_{n=c}^{\infty} \frac{c^c}{c!}\rho^n \\ &= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \sum_{n=c}^{\infty} \frac{(c\rho)^c}{c!}\rho^{n-c} \\ &= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!}\sum_{n=0}^{\infty} \rho^n \\ &= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}. \end{aligned}$$

Next,

$$\begin{aligned} \mathsf{E}\{L_Q\} &= \sum_{n=c}^{\infty} (n-c)p(n) \\ &= \sum_{n=c}^{\infty} (n-c)\frac{c^c}{c!}\rho^n p(0) \\ &= \frac{c^c\rho^c}{Gc!}\sum_{n=c}^{\infty} (n-c)\rho^{n-c} \\ &= \frac{c^c\rho^c}{Gc!}\sum_{n=0}^{\infty} n\rho^n = \frac{c^c\rho^c}{Gc!}\frac{\rho}{(1-\rho)^2}, \end{aligned}$$

where, with our common trick (if we don't want to use generating functions),

$$\begin{aligned} \sum_{n=0}^{\infty} n\rho^n &= \sum_{n=0}^{\infty}\sum_{i=1}^{\infty} \mathbb{1}_{i\le n}\rho^n = \sum_{i=1}^{\infty}\sum_{n=0}^{\infty} \mathbb{1}_{i\le n}\rho^n \\ &= \sum_{i=1}^{\infty}\sum_{n=i}^{\infty} \rho^n = \sum_{i=1}^{\infty} \rho^i \sum_{n=0}^{\infty} \rho^n \\ &= \frac{1}{1-\rho}\sum_{i=1}^{\infty} \rho^i = \frac{\rho}{1-\rho}\sum_{i=0}^{\infty} \rho^i = \frac{\rho}{(1-\rho)^2}. \end{aligned}$$

Observe again that using indicators and Fubini's theorem (interchanging summations/integrals) makes the above computation painless. Realize, by the way, that

$$\sum_{n=0}^{\infty} np(n) = \sum_{n=1}^{\infty} np(n).$$

We next show that the expected number of jobs in service is given by

$$\mathsf{E}\{L_S\} = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n).$$

This expression is not the easiest to start with. With a slight modification the entire derivation becomes easier. I also premultiply by the normalization constant G to get rid of it on the right

hand side.

$$\begin{aligned} G\,\mathsf{E}\{L_S\} &= G\sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) \\ &= \sum_{n=1}^{c} n\frac{(c\rho)^n}{n!} + \sum_{n=c+1}^{\infty} c\frac{c^c\rho^n}{c!} = \sum_{n=1}^{c}\frac{(c\rho)^n}{(n-1)!} + \frac{c^{c+1}}{c!}\sum_{n=c+1}^{\infty}\rho^n \\ &= \sum_{n=0}^{c-1}\frac{(c\rho)^{n+1}}{n!} + \frac{(c\rho)^{c+1}}{c!}\sum_{n=0}^{\infty}\rho^n = c\rho\left(\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}\right). \end{aligned}$$

Observe that the right hand side is precisely equal to ρcG, and hence,

$$\mathsf{E}\{L_S\} = c\rho = \frac{\lambda}{\mu}.$$

Solution 2.1.6: 1. The second term in (2.2d) is $(c\rho)^c/c! = (\lambda/\mu)^c/c!$. It is well known that $x^c/c! \to 0$ as $c \to \infty$.

2. No matter how many jobs are in service, there is always another free server available when a new job arrives. Thus, jobs never have to wait in queue, and only spend time in service. Since $\mathsf{E}\{S\} < \infty$ by assumption, jobs spend a finite time (with probability one) at a server.

3. Write $\rho = \lambda/\mu$. Then, from the formulas for the $M/M/\infty$ queue, it follows that $p(n) = e^{-\rho}\rho^n/n!$. Interestingly, we see that this is equal to $\mathsf{P}\{N = n\}$ where N is a Poisson r.v. with parameter ρ. Thus, the number in the system L is Poisson distributed with parameter ρ, thus $\mathsf{E}\{L\} = \rho$.

   Another way to see that $\mathsf{E}\{L\} = \rho$ is by noting that in the $M/M/\infty$ queue jobs do not interact with each other in the queue. When they arrive, there is always a free server available. Since work arrives at rate ρ, and all jobs are in service simultaneously, the average number of busy servers must also be ρ.

Solution 2.1.7: Finite buffer size. Formally the number of customers that fit into a shop is necessarily finite. This answer, however, is not intended. Typically, the number of customers in a restaurant is limited. Example 2: Sometimes call centers reject callers when the system is too busy.

A finite calling population occurs for instance at a factory with a number of machines. When a machine breaks down, it becomes a (repair) job at the repair department. Thus, a break down forms an arrival at the repair shop. The mechanics at the repair department form a set of parallel servers. Typically, the number of machines quite small, 10 or so, and when a machine is 'down', i.e., broken, it cannot break again. Hence, when 2, say, machines are in repair, the number of 'customers' that can arrive to the queueing system is only 8.

Solution 2.1.8: In a queueing system with balking, customers may decide to balk at a level b. Thus, whether only b customers are admitted to the system (i.e., blocked), or balk at level b, the effect is the same: the number of people in the system remains at or below b. However, a fraction of customer may already balk at lower levels, like in the example above, so that the arrival stream is 'thinned' due to balking customers. In that respect, a queueing system with balking behaves differently.

Solution 2.1.9: 1) Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$, and solve Eq. (**??**) and (**??**). Thus:

$$\begin{aligned} p(n+1) &= \frac{(N-n)\lambda}{\mu} p(n) = \rho(N-n)p(n) \\ &= \rho^2(N-n)(N-(n-1))p(n-1) \\ &= \rho^3(N-n)(N-(n-1))(N-(n-2))p(n-2) \\ &= \rho^{n+1}(N-n)(N-(n-1))\cdots(N-(0))p(0) \\ &= \rho^{n+1}\frac{N!}{(N-(n+1))!}p(0). \end{aligned}$$

Next, we need to normalize this. Observe that $p(N+1) = P(N+2) = \ldots = 0$ since there are just N customers, so that the system can never contain more than N customers. Thus, we want $p(0)$ to be such that

$$1 = \sum_{n=0}^{N} p(n) = p(0) \sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!}$$

We see from this that $p(0)$ times some constant must be 1. Hence, dividing by this constant, we get

$$p(0) = \left(\sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!} \right)^{-1}.$$

I asked WolframAlpha to simplify this, but the answer I got was not particularly revealing.

2)

3) Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = n\mu$. Then

$$\begin{aligned} p(n+1) &= \frac{\lambda(n)}{\mu(n+1)} p(n) = \frac{(N-n)\lambda}{(n+1)\mu} p(n) = \frac{(N-n)(N-(n-1))}{(n+1)n} \frac{\lambda^2}{\mu^2} p(n-1) \\ &= \frac{N!}{(N-(n+1))!} \frac{1}{(n+1)!} \rho^{n+1} p(0) = \binom{N}{n+1} \rho^{n+1} p(0). \end{aligned}$$

Hence, after normalization, i.e., requiring that $p(0)$ is such that $\sum_{n=0}^{N} p(n) = 1$, so that $p(0) = \left(\sum_{k=0}^{N} \rho^k \binom{N}{k} \right)^{-1}$, the final result becomes

$$p(n) = \frac{\rho^n \binom{N}{n}}{\sum_{k=0}^{N} \rho^k \binom{N}{k}}.$$

Solution 2.1.10: I am going to implement the formulas of Eq. (2.2) in python. First the results for the $M/M/3$ queue.

```
from math import exp, factorial

labda = 5
mu = 2
c = 3

rho = labda / mu / c
rho
```

```
0.8333333333333334
```

```
G = sum((c * rho)**n / factorial(n) for n in range(c))
G += (c * rho)**c / ((1 - rho) * factorial(c))
G
```

```
22.250000000000004
```

```
ELQ = (c * rho)**c / (factorial(c) * G) * rho / (1 - rho)**2
ELQ
```

```
3.511235955056181
```

```
ELS = rho * c
ELS
```

```
2.5
```

```
EL = ELQ + ELS
EL
```

```
6.011235955056181
```

Now for the $M/M/1$ queue:

```
labda = 5
c = 3
mu = 2*c
```

```
rho = labda / mu
rho
```

```
0.8333333333333334
```

```
ELQ = rho**2/(1-rho)
ELQ
```

```
4.166666666666668
```

```
ELS = rho
ELS
```

```
0.8333333333333334
```

```
EL = ELS + ELQ
EL
```

```
5.000000000000001
```

```
rho/(1-rho) # this must also be EL, just a check
```

```
5.000000000000002
```

Note the last check. As a rule, you should always compare your results with known results. BTW, that is one of the reasons I prefer to code the formulas instead of using a calculator. Testing with code is relatively easy, whereas with a calculator it is impossible (You simply can't check what you typed at the calculator.)

So, returning to the results, as expected, the number of jobs in queue is smaller for the $M/M/3$ queue, but the number in service is higher.

To put things in a larger perspective, see Figure 2.1 where we plot the ratio of the queue lengths and the system length as functions of ρ. We see, in case of high load, that $\mathsf{E}\{L_Q\}$ and $\mathsf{E}\{L\}$ are nearly the same for both systems. This is as expected: when the load is high, most jobs should be in the queue. Therefore, $\mathsf{E}\{L_Q\}/\mathsf{E}\{L\} \to 1$ as $\rho \to 1$. When ρ is small, the difference is quite large. This is also reasonable, because the service time in the fast $M/M/1$ is 3 times as small as the service time in the $M/M/3$ queue. Hence, as ρ is small, the time in the system is dominated by service time, as there is hardly any queueing time, if at all. Thus, there must be more jobs in the system on average in the $M/M/3$ queue than in the fast $M/M/1$ queue.

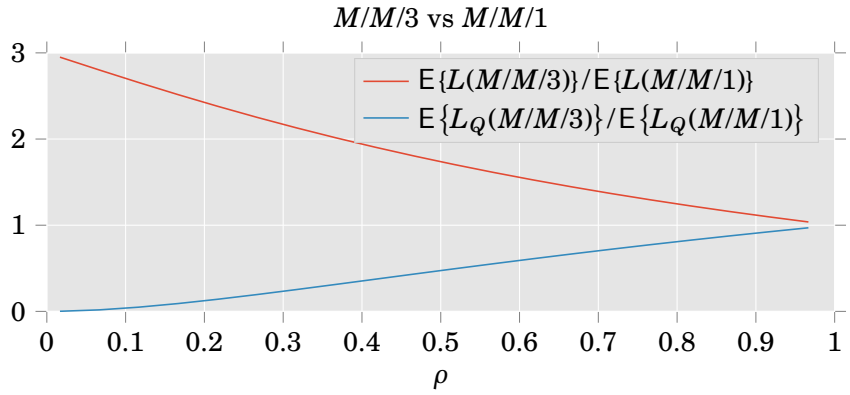


Figure 2.1: multi server versus single fast server

The code can be found on `github` in the `progs` directory.