Notes on the course of Software Performance and Scalability

Andrea Marin

February 13, 2025

Contents

Preface

Chapter 1

Poisson processes

1.1 Introduction

Poisson processes play an important role in the performance evaluation of computer systems. In order to understand their importance, let us consider a web server and let us imagine to log the requests that arrive at the system. Specifically, for each request we record its arrival timestamp, i.e., the epoch of arrival. Informally, the arrival process is a Poisson process if the number of arrivals in a certain time interval follows a Poisson distribution with an intensity that is directly proportional to the length of the interval and to the speed of the arrivals. Indeed, if the speed increases or of we consider larger intervals, we expected to count more arrivals! At this point, we may think that requiring that the arrival process is a Poisson process seems to be quite restrictive, however this may be wrong. Indeed, if the users are many and behave independently of each other, and their interaction with the web server is sufficiently slow with respect to its speed, then the web server sees a Poisson arrival process. This is a very powerful observation that we will prove in this chapter and can drive us to decide if it is reasonable to assume that the arrivals follow a Poisson process. Another reason that justifies the importance of Poisson processes is their analytical tractability. In many cases, we have exact results for models of computer systems only if the arrivals follow a Poisson process. Therefore, Poisson processes allow us to reason on systems' performance by using relatively simple equations.

1.2 Background on the exponential distribution and its properties

Poisson processes are deeply connected with exponential random variables. Therefore, it may be useful to recall some properties of these random variables that are useful to understand what follows. A random variable X has an exponential

distribution if its cumulative density function is:

$$F_x(x) = Pr\{X \leq x\} = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\lambda \in \mathbb{R}^+$ is called the parameter or the rate of the random variable. The probability density function can be easily obtained by deriving the $F_X(x)$, thus:

$$f_X(x) = \frac{\partial F_X(x)}{\partial x} = \lambda e^{-\lambda x}\,.$$

The expected value of the random variable is:

$$E[X] = \frac{1}{\lambda}\,.$$

Notice that if we interpret the random variable as a random time, then λ^{-1} is the expectation of this random time and thus is measured in a time unit (e.g., seconds). Therefore λ is measured as a speed, e.g., s^{-1} or number of events that we can observe per unit of time. This is the reason why λ is also called rate of the exponential random variable. Let us consider two independent exponential random variables X_1 and X_2, with rates λ_1 and λ_2, respectively. Let us define the random variable $Y = \min(X_1, X_2)$. Then, we have:

$$\begin{aligned} F_Y(x) = Pr\{Y \leq x\} &= 1 - Pr\{X_1 > x \wedge X_2 > x\} \\ &= 1 - Pr\{X_1 > x\}Pr\{X_2 > x\} = 1 - e^{-(\lambda_1+\lambda_2)x}\,, \end{aligned}$$

i.e., Y is distributed as an exponential r.v. whose rate is the sum of the rates of X_1 and X_2. This result can be generalised to n independent exponential random variables $X_1, \ldots, X_n$: the distribution of the minimum is still exponential with a rate which is given by the sum of the rates of $X_1, \ldots, X_n$. Notice that Y is clearly *not* independent of $X_1, \ldots, X_n$. Now, we address another problem. Given the independent and exponential random variables X_i whose parameter is $\lambda_i > 0$ for $i = 1, \ldots, n$, what is the probability that the minimum is X_i? It is obvious that if $\lambda_1 = \lambda_2 = \ldots \lambda_n$, then the probability does not depend on i and is $1/n$. What happens for arbitrary rates? Let us reason on two independent exponential random variables X_1 and X_2, let us compute $Pr\{X_1 \leq X_2\}$. The probability that X_2 belongs to the interval $[x, x+dx]$ is $f_{X_2}(x)dx$, for small dx. So we can apply the law of total probability and we have:

$$\begin{aligned} Pr\{X_1 \leq X_2\} &= \int_0^\infty Pr\{X_1 \leq X_2 | X_2 = x\} f_{X_2}(x)dx = \int_0^\infty F_{X_1}(x) f_{X_2}(x)dx \\ &= \int_0^\infty \lambda_2 e^{-\lambda_2 x} dx - \int_0^\infty \lambda_2 e^{-(\lambda_1+\lambda_2)x} dx \\ &= 1 - \frac{\lambda_2}{\lambda_1+\lambda_2} \int_0^\infty (\lambda_1+\lambda_2) e^{-(\lambda_1+\lambda_2)x} dx = \frac{\lambda_1}{\lambda_1+\lambda_2}\,. \end{aligned}$$

In conclusion, the probability of variable i to be the minimum is directly proportional to its rate and normalised with the sum of all the rates of the concurring variables. This result holds also for an arbitrary number of random variables and the extension of the proof is simple.

It is worth of notice that the maximum of two (or more) exponential random variables is *not* an exponential random variable.

The last property that we present of the exponential distribution is the memoryless property. Assume that we have a light bulb whose lifetime is modelled as an exponential distribution with mean $\lambda^{-1} = 3$ years. Once installed, the light bulb works for 4 years, what is the distribution of the residual life time? Thanks to the memoryless property of the exponential distribution, we can say that the residual life time is an exponential random variable with rate λ, i.e., identical to the original one. Let us prove the result for general values. We have that λ^{-1} is the expected duration of the random variable, τ is the observation epoch and we want to determine the distribution of the residual time, i.e. $Pr\{X \leq \tau + t | X > \tau\}$. Then, we have:

$$\begin{aligned} Pr\{X \leq \tau + t | X > \tau\} &= \frac{Pr\{X \leq \tau + t \wedge X > \tau\}}{Pr\{X > \tau\}} \\ &= \frac{Pr\{\tau < X \leq \tau + t\}}{1 - F_X(\tau)} = \frac{F_X(\tau + t) - F_X(\tau)}{e^{-\lambda\tau}} = \frac{-e^{-\lambda\tau}e^{-\lambda t} + e^{-\lambda\tau}}{e^{-\lambda\tau}} \\ & 1 - e^{\lambda t}\,. \quad (1.1) \end{aligned}$$

The result is even more intriguing because the only non-negative continuous random variable that enjoys the memoryless property is the exponential one. Therefore, whenever we have a phenomenon in continuous time that by its intrinsic properties is memoryless, we have that its distribution must be exponential.

1.3 Definitions of Poisson processes

The Poisson process is a special type of continuous time counting process. It is widely used in the performance evaluation of computer and telecommunication systems to model the arrival process of jobs (or customers) at a service facility.

Definition 1.1 (Counting process) A stochastic process $X(t)$ is a counting process if:

1. $X(t) \in \mathbb{N}$ for all $t \in T$, where T denotes the time domain;
2. if $s \leq t$ then $X(s) \leq X(t)$.

We can think at a counting process as a stochastic process that counts the number of events that have been observed in the time interval $(0, t]$. Given two time epochs $t_1, t_2 \in T$ with $t_1 < t_2$, then $X(t_2) - X(t_1)$ is the number of

events that occurred in the interval $(t_1, t_2]$. A counting process has *independent increments* if the number of events that occur in disjoint time intervals are independent. A counting process is said to possess *stationary increments* if the distribution of the number of events that occur in any interval of time depends only on the length of the interval.

A Poisson process is a special type of counting process which satisfies some important properties. If the Poisson process is time-homogeneous, then it has stationary and independent increments. Henceforth, we will just use the name 'Poisson process' to denote time-homogeneous Poisson processes. We propose three different definitions Poisson processes and then we prove that they are equivalent. Each of these definitions shows some important aspects of this process and helps our intuition in understanding when we can safely consider a system to have Poisson arrival, how to simulate these arrivals or how to analyse them.

Definition 1.2 A Poisson process $X(t)$ with rate (or intensity) $\lambda \in \mathbb{R}^+$ is a continuous time counting process, i.e., $t \in \mathbb{R}$, that satisfies the following properties:

1. $X(0) = 0$
2. for any pair of disjoin intervals $(t_1, t_2]$ and $(t_3, t_4]$, the increments $X(t_2) - X(t_1)$ and $X(t_4) - X(t_3)$ are independent random variables
3. for any $t, s \geq 0$ the increment in the interval $(t, t + s]$ has a Poisson distribution with mean λs:

$$Pr\{X(t + s) - X(t) = k\} = \frac{(\lambda s)^k e^{-\lambda s}}{k!} \tag{1.2}$$

Le us see some examples.

1.1 Example
Assume that we have a Poisson process with intensity $\lambda = 1$ events per second. Let us answer to the following questions:

1. What is the probability that there is not any arrival in the interval $(0, 2]$?
2. What is the probability that there is not any arrival in the interval $(6, 8]$?
3. What is the probability of 2 arrivals in the interval $(2, 4]$ conditioned on the fact we had 3 arrivals in the interval $(0, 2]$?
4. What is the probability of 4 arrivals in the interval $(0, 5]$ knowing that in the interval $(2, 3]$ we have 2 arrivals?

Solution. Let us start with the first point. We need simply to apply the definition of Poisson process, and we obtain:

$$Pr\{X(2) = 0\} = e^{-2} \simeq 0.135\,.$$

Since the process has stationary increments, the same answer is correct for the interval $(6, 8]$. Let us consider the third bullet. The increments in the process are independent for disjoint intervals, so we do not care about what happened in the interval $(0, 2]$ when we consider the statistics of $(2, 4]$. Therefore, we have:

$$Pr\{X(4) - X(2) = 2 | X(2) = 3\} = Pr\{X(2) = 2\} = \frac{2^2 e^{-2}}{2!} = 2e^{-2} \simeq 0.27\,.$$

The fourth bullet is slightly more complicated because it considers overlapping intervals and hence we cannot assume the independence of the random variables. However, we can split the interval $(0, 5]$ into three non-overlapping parts:

$$(0, 5] = (0, 2] \cup (2, 3] \cup (3, 5]\,.$$

We know what happens in the interval $(2, 3]$. Therefore, we must account for the following disjoint events: 0 arrivals in $(0, 2]$ and 2 in $(3, 5]$, 1 arrival in $(0, 2]$ and 1 arrival in $(3, 5]$ and, finally, 2 arrivals in $(0, 2]$ and 0 in $(3, 5]$. Thus, we have:

$$Pr\{X(5) = 4 | X(3) - X(2) = 2\} = 2Pr\{X(2) = 2\}\,\mathrm{Pr}\{X(2) = 0\}$$
$$+ \left(Pr\{X(2) = 1\}\right)^2 = 2\frac{2^2 e^{-2}}{2!} e^{-2} + (2e^{-2})^2 = 4(e^{-4} + e^{-4}) = 0.1465\,.$$

The second definition that we propose for the Poisson process is the following.

Definition 1.3 A Poisson process $X(t)$ with rate (or intensity) $\lambda \in \mathbb{R}^+$ is a continuous time counting process that satisfies the following properties:

1. $X(0) = 0$
2. The process is stationary and has independent increments
3. $Pr\{X(h) = 1\} = \lambda h + o(h)$
4. $Pr\{X(h) \geq 2\} = o(h)$

where we say that a function f is $o(h)$ if:

$$\lim_{h\to 0} \frac{f(h)}{h} = 0\,.$$

Basically, bullets 3 and 4 state that in small intervals with width h the probability of observing one event is approximatively λh, while the probability of observing more than one event is very small. In fact, we can consider that in a Poisson process, the probability that two events occur at the same time epoch is 0.

Theorem 1.1 Definition 1.3 implies Definition 1.2.

Proof. Let $P_n(t) = Pr\{X(t) = n\}$. Then, we have that:

$$P_0(t+h) = Pr\{X(t+h) = 0\} = Pr\{X(t) = 0 \wedge X(t+h) - X(t) = 0\}\,.$$

Notice that the interval $(0, t]$ and $(t, t+h]$ are disjoint and by property 2 of Definition 1.3, we can write:

$$P_0(t+h) == Pr\{X(t) = 0\}Pr\{X(t+h) - X(t) = 0\}\,.$$

What is the probability that in the interval $(t, t+h]$ there are no arrivals? We can write this probability as the complementary of the probability that exactly one arrival occurs (bullet 3 of Definition 1.3) summed to the probability that at least two arrivals occur (bullet 4 of Definition 1.3). Recall that the sum of two functions which are $o(h)$ is still a function $o(h)$. Therefore, we write:

$$P_0(t+h) = P_0(t)\left(1 - \lambda h + o(h)\right)\,.$$

Then, we can write:

$$\frac{P_0(t+h) - P_0(t)}{h} = -\lambda P_0(t) + \frac{o(h)}{h}\,.$$

For $h \to 0$, the left hand side becomes the definition of the derivative of $P_0(t)$, obtaining:

$$P_0'(t) = -\lambda P_0(t)$$

which can be rewritten as $P_0'(t)/P_0(t) = -\lambda$. If we integrate on both hand sides we have:

$$\log P_0(t) = -\lambda t + c$$

and hence $P_0(t) = Ke^{-\lambda t}$. The first bullet of Definition 1.3 says that $P_0(0) = 1$ and hence we have:

$$P_0(t) = e^{-\lambda t}\,.$$

Let us now consider the case $P_n(t)$ with $n \geq 1$. We can write:

$$P_n(t+h) = Pr\{X(t+h) = n\} = Pr\{X(t) = n \wedge X(t+h) - X(t) = 0\}$$
$$+Pr\{X(t) = n-1 \wedge X(t+h) - X(t) = 1\} + Pr\{X(t+h) = n \wedge X(t+h) - X(t) \geq 2\}$$

By bullet 4 of Definition 1.3 we have that the last term is $o(h)$ and by using bullet 2 we obtain:

$$P_n(t+h) = P_n(t)P_0(h)+P_{n-1}(t)P_1(h)+o(h) = (1-\lambda h)P_n(t)+\lambda h P_{n-1}(t)+o(h)\,.$$

By following the same procedure used for $P_0(t)$ we obtain the differential equation:

$$P_n'(t) = -\lambda P_n(t) + \lambda P_{n-1}(t)$$

that is equivalent to:

$$e^{\lambda t}\left(P_n'(t) + \lambda P_n(t)\right) = \lambda e^{\lambda t} P_{n-1}(t)\,,$$

which can be written as

$$\frac{d}{dt}(e^{\lambda t}P_n(t)) = \lambda e^{\lambda t}P_{n-1}(t)\,.$$

For $n = 1$ we have to solve:

$$\frac{d}{dt}(e^{\lambda t}P_1(t)) = \lambda e^{\lambda t}e^{-\lambda t} = \lambda\,,$$

i.e., $P_1(t) = (\lambda t + c)e^{-\lambda t}$ which since $P_1(0) = 0$ by bullet 1, we have $P_1(t) = \lambda t e^{-\lambda t}$. The proof that $P_n(t) = e^{-\lambda t}(\lambda t)^n/n!$ can be derived by induction. □

At this point, it is important to recall some important properties of the exponential random variables. If M is an exponential random variable with parameter λ then its cumulative density function is:

$$Pr\{M < t\} = F_\lambda(t) = 1 - e^{-\lambda t}\,,$$

while its probability density function is:

$$f_\lambda(t) = \frac{dF_\lambda(t)}{dt} = \lambda e^{-\lambda t}\,.$$

If we interpret M as a delay then, its expected value is $1/\lambda$ and λ is called *rate* of the exponential random variable. Among the important properties of the exponential r.v.s we recall the *memoryless* property. This states that for any time t and increment s we have that:

$$Pr\{M < t + s | M \geq s\} = Pr\{M < t\}\,.$$

Basically, this states that if we imagine that someone set up an alarm which expires in an exponentially distributed random time with mean $1/\lambda$ and we observe that after s seconds the alarm is not expired, the distribution of the residual time is still the same that we had at the beginning of the experiment. The proof is straightforward:

$$\begin{aligned} Pr\{M \geq t + s | M \geq s\} = \frac{Pr\{M \geq t + s \wedge M \geq s\}}{Pr\{M \geq s\}} &= \frac{Pr\{M \geq t + s\}}{Pr\{M \geq s\}} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} = e^{-\lambda t}\,, \end{aligned}$$

and hence it follows that $Pr\{M < t+s | M \geq s\} = 1 - e^{-\lambda t}$. The other important property regards the distribution of the minimum of two independent exponential random variables M_1 and M_2 with rate λ_1 and λ_2, respectively. Let $Y = \min(M_1, M_2)$. We prove that Y is exponentially distributed with rate $\lambda_1 + \lambda_2$. We have that:

$$Pr\{Y > t\} = Pr\{M_1 > t\} Pr\{M_2 > t\} = e^{-\lambda_1 t} e^{-\lambda_2 t} = e^{-(\lambda_1+\lambda_2)t}\,.$$

Let us introduce the third definition of Poisson process. This definition states that, if the time interval between successive increments of the counting process are distributed according to i.i.d. exponential random variables, then the counting process is a Poisson process.

Definition 1.4 A Poisson process $X(t)$ with rate (or intensity) $\lambda \in \mathbb{R}^+$ is a continuous time counting process that satisfies the following properties:

1. $X(0) = 0$
2. Let $S_i = \inf\{t : X(t) >= i\}$ be the time epoch in which the process jumps from $i-1$ to i, $i = 1, 2, \ldots$ and $S_0 = 0$. Then, $Y_i = S_{i+1} - S_i$ are i.i.d. exponential random variables with rate λ.

Definition 1.4 states that the time interval between two successive increments are independent and identically distributed random variables with rate λ.

Theorem 1.2 Definition 1.2 implies Definition 1.4.

Proof.

Let $Y_1, Y_2, \ldots$ be the sequence of inter-arrival times, i.e., $S_1 = Y_1$ is the arrival time of the first job, $S_2 = Y_1 + Y_2$ that of the second job, and so on. Let us determine the distribution of Y_n. We first consider Y_1. If $Y_1 = t_1$ is the arrival of the first customer, this means that in $(0, t_1]$ there is no customer arrival, i.e.,

$$Pr\{Y_1 > t\} = Pr\{X(t) = 0\} = e^{-\lambda t}\,.$$

In other words, $Pr\{Y_1 \leq t\} = 1 - e^{-\lambda t}$ which the distribution of the exponential random variable. Let us consider S_2 and we compute $Pr\{S_2 > t | S_1 = s\}$. This event has the same probability of 0 arrivals in the interval $(s, s+t]$ given that $Y_1 = s$. By using the property of independent increments, we obtain:

$$\begin{aligned} Pr\{Y_2 > t | Y_1 = s\} &= Pr\{0 \text{ events in } (s, t+s] | Y_1 = s\} \\ &= Pr\{0 \text{ events in } (s, t+s]\} = e^{-\lambda t}\,. \end{aligned}$$

The same arguments can be repeated for any $n > 0$. This shows that the distribution of the inter-arrival times are exponentially distributed and independent. □

Finally, we prove that Definition 1.4 implies Definition 1.3.

Theorem 1.3 Definition 1.4 implies Definition 1.3.

Proof. The proof of this theorem is very interesting for understanding the characteristics of Poisson processes. Let us look at Figure 1.1. We observe that the following relation holds:

$$Pr\{X(t) < n\} = Pr\{S_n > t\}\,,$$

or, equivalently:

$$Pr\{X(t) \geq n\} = Pr\{S_n \leq t\}\,.$$

First, we prove that the stochastic processes that satisfy Definition 1.4 have independent and stationary increments. Let us prove the latter property, i.e., the distribution of the number of arrivals in a time interval of length s starting from τ depends only of length s (and not on τ). This property immediately follows from the memoryless property of the exponential random variables, i.e., once we choose τ, the distribution for the first event in the interval is exponentially distributed with rate λ and does not depend on the time of occurrence of the previous arrival. The fact that in disjoint intervals the distribution of the number of arrivals are independent can be proved analogously. Thus, we can write:

$$Pr\{X(\tau + s) - X(\tau) = n\} = Pr\{X(s) = n\}\,,$$

since we can choose the interval $(0, s]$ for simplicity. What is the probability that $X(s) = 1$?

$$Pr\{X(s) = 1\} = Pr\{S_1 \leq s \wedge S_2 > s\}\,,$$

i.e., we have the first arrival before s, but the second arrival after s. Then, we can write:

$$\begin{aligned} Pr\{X(s) = 1\} = Pr\{S_1 \leq s \wedge S_2 > s\} &= Pr\{Y_1 \leq s \wedge Y_1 + Y_2 > s\} \\ &= \int_0^\infty Pr\{Y_1 \leq s \wedge Y_1 + Y_2 > s | Y_1 = t\} f_\lambda(t) dt\,. \end{aligned}$$

Clearly, when $Y_1 = t$ and $t > s$, we have $Pr\{Y_1 \leq s \wedge Y_1 + Y_2 > s\} = 0$, therefore:

$$\begin{aligned} Pr\{X(s) = 1\} = \int_0^s Pr\{Y_2 > s - t\} \lambda e^{-\lambda t} dt &= \lambda \int_0^s e^{-\lambda(s-t)} e^{-\lambda t} dt \\ &= \lambda s e^{-\lambda s} = \lambda s + (e^{-\lambda s} - 1)\lambda s\,. \end{aligned}$$

Now, to derive the third bullet of Definition 1.3, we let $s \to 0$ and we observe that the following limit holds:

$$\lim_{s \to 0} \frac{(e^{-\lambda s} - 1)\lambda s}{s} = 0\,,$$

as required.

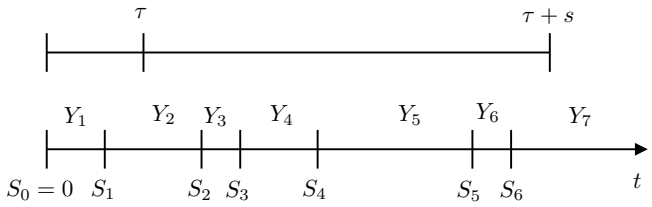


Figure 1.1: Example of Poisson process realisation.

We now proceed to the proof that the fourth bullet of Definition 1.3 is satisfied. We notice that:

$$\begin{aligned} Pr\{X(s) \geq 2\} &= Pr\{S_2 \leq s\} = Pr\{Y_1 + Y_2 \leq s\} \\ &= \int_0^\infty Pr\{Y_1 + Y_2 \leq s | Y_1 = t\} f_\lambda(t) dt = \int_0^s Pr\{Y_2 \leq s - t\} \lambda e^{-\lambda t} dt \\ &= \lambda \int_0^s (1 - e^{-\lambda(s-t)}) e^{-\lambda t} dt = \lambda \int_0^s (e^{-\lambda t} - e^{-\lambda s}) dt = 1 - e^{-\lambda s} - \lambda s e^{-\lambda s} \,. \end{aligned}$$

To conclude, we consider the following limit:

$$\lim_{s \to 0} \frac{1 - e^{-\lambda s} - \lambda s e^{-\lambda s}}{s} = \left[\frac{0}{0}\right] ,$$

that, by the rule of de l'Hôpital, becomes:

$$\lim_{s \to 0} \frac{1 - e^{-\lambda s} - \lambda s e^{-\lambda s}}{s} = \lambda e^{\lambda s} - \lambda(e^{-\lambda s} - s\lambda e^{-\lambda s}) = 0 \,.$$

This concludes the proof. □

The following corollary is very important to understand the nature of Poisson processes.

Corollary 1.1 Definitions 1.2, 1.3, 1.4 are equivalent.

Proof. We have shown that Definition 1.3 implies Definition 1.2 (see Theorem 1.1), and Definition 1.2 implies Definition 1.4 (see Theorem 1.2) and, finally, that Definition 1.4 implies Definition 1.3:

$$D3 \Rightarrow D2 \Rightarrow D4 \Rightarrow D3 \,.$$

□

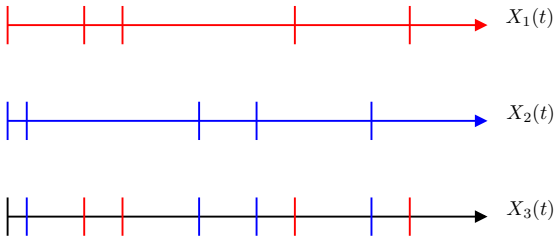


Figure 1.2: Superposition of independent Poisson processes.

1.4 Superposition and splitting of Poisson processes

If $Y_1(t)$ and $Y_2(t)$ are two counting processes, their superposition is the process $Y(t) = Y_1(t) + Y_2(t)$ that counts both the events counted by $Y_1(t)$ and those counted by $Y_2(t)$. In the context of performance evaluation, we can consider two arrival streams at a server. In this section, we will see that if the two arrival streams are independent Poisson processes, then their superposition is also a Poisson process whose intensity is the sum of the two original intensities. It is worth of notice that, for all counting processes (even dependent) it is true that the intensity of their superposition is the sum of the intensities of the original processes, i.e., in sufficient long time interval the expected number of events observed in the superposition is the sum of the expected numbers of events in the original processes. However, here we are saying something stronger. Specifically, we say that the process obtained by superposition still maintains the property of being a Poisson process. The properties of the exponential distributions are very useful to prove the following result on Poisson processes.

Theorem 1.4 Let $X_1(t)$, $X_2(t)$, ..., $X_N(t)$ be independent Poisson processes with intensities λ_1, λ_2, ..., λ_N, respectively. Then, the process $X(t) = X_1(t) + X_2(t) + \cdots + X_N$ (see Figure 1.2) defined as the superposition of the Poisson processes is still a Poisson process whose intensity is $\lambda_1 + \lambda_2 + \cdots + \lambda_N$.

The proof is left for exercise.

Finally, let us consider a Poisson process $X(t)$ and assume that we discard the arrivals with probability $1 - p$ and keep them with probability p. Assume that the probability of discarding is independent of the Poisson process. The following theorem states that the process obtained by the observation of the valid arrivals is still a Poisson process. The usual application of this result, is to study random dispatchers. Suppose that requests arrive at a dispatcher according to a Poisson process with intensity λ. For each request, the dispatcher probabilistically decides to forward it to one of N servers. Server i is chosen with probability p_i. Then, the arrival process seen by server i is a Poisson process

with intensity λp_i.

Theorem 1.5 Let $X(t)$ be a Poisson process, and $Y_i(t)$ be the process obtained by keeping each arrival event with probability p_i, with $i = 1, \ldots, N$ and $\sum_{i=1}^{N} p_i = 1$. Assume that the these choices are independent of $X(t)$. Then,

- $Y_i(t)$ is a Poisson process with intensity λp;
- Y_i and Y_j are independent, for $i \neq j$.

Proof. We prove the result for $N = 2$, the extension to an arbitrary number of splits is straightforward. Suppose that at time t_0 we have that an event is selected for flow 1, i.e., with probability p_1. What is the distribution of the number n of events that must occur in $Y(t)$ before observing the next event in $Y_1(t)$? Since, for each event in $Y(t)$ we have a probability p_1 to observe it in $Y_1(t)$ and $1-p_1$ to observe it in $Y_2(t)$, the desired distribution is geometric, i.e., $p_1(1-p_1)^{n-1}$. The time between successive events in $Y(t)$ is exponentially distributed so, in $Y_1(t)$ the time between successive events is the sum of n i.d.d. exponential random variables with rate λ. It is known that the sum of a geometric number of i.i.d. exponential random variables has exponential distribution with rate given by the product of the mean of the geometric distribution and that of the exponential random variables. In our case, it is an exponential random variable with mean $1/(\lambda p_1)$ and hence with rate λp_1. By Definition 1.4, we conclude that $Y_1(t)$ is a Poisson process with intensity λp_1. □

1.5 The law of rare events

The law of rare events is helpful in performance modelling because it helps in understanding when it is reasonable to model a certain stochastic process by means of a Poisson process. Informally, we can think that when a large set of independent users interacts with a system with a very slow speed (with respect to that of the system), the requests arrive at the system according to a Poisson process. For example, the traffic seen by a web server can be reasonably approximated with a Poisson process, although some events may trigger the users and cause a burst. This is the case for web pages of newspapers when important news are posted and attract many visitor simultaneously. This traffic will not follow a Poisson process because the event causes a synchronisation and hence the independence assumption is not satisfied.

Let us formally introduce the law of rare event. Assume that we have a population of N independent individuals that generate an arrival to our system rarely, i.e., with low probability. From the point of view of an external observer it is impossible to distinguish who originated an arrival. Although each individual

generates very rare requests, if N is large enough we may have a non-trivial counting process for the arrivals. The following theorem states that this arrival process is a Poisson process. The idea is that we may consider an interval or arbitrary length s. Then, in this interval of time, each individual generates an arrival with a small p, such that the expected number of arrivals during s is $\lambda s = Np > 0$, where λ is the traffic intensity. Now, since each individual is independent, the probability of observing k arrivals in the time interval is a Binomial random variables with parameters N and p.

Theorem 1.6 (The law of rare events) For each $N \geq 1$, let B_N be a Binomial random variable with parameters N and $p = \lambda s/N$. Then, for any $k \geq 0$:

$$\lim_{N\to\infty} Pr\{B_N = k\} = e^{-\lambda s}\frac{(\lambda s)^k}{k!}$$

which is the probability density function of a Poisson random variable with parameter λ.

Proof. We have that:

$$\begin{aligned} Pr\{B_N = k\} = \binom{N}{k} p^k (1-p)^{N-k} &= \frac{N!}{(N-k)!k!}\left(\frac{\lambda s}{N}\right)^k \left(1-\frac{\lambda s}{N}\right)^{N-k} \\ &= \frac{N!}{N^k(N-k)!}\frac{(\lambda s)^k}{k!}\left(1-\frac{\lambda s}{N}\right)^{N}\left(1-\frac{\lambda s}{N}\right)^{-k}. \end{aligned}$$

The proof follows from the fact that:

$$\lim_{N\to\infty}\left(1-\frac{\lambda s}{N}\right)^N = e^{-\lambda s}, \qquad \lim_{N\to\infty}\left(1-\frac{\lambda s}{N}\right)^{-k} = 1,$$

and:

$$\lim_{N\to\infty}\frac{N!}{(N-k)!N^k} = 1.$$

□

Exercises

Exercise 1 Complete the proof of Theorem 1.1 by showing the induction steps and that Definition 1.2 implies Definition 1.3.

Exercise 2 Give the proof of Theorem 1.4. Hint: the simplest way is to use the property of the minimum of independent exponential random variables.

Exercise 3 Give the proof of Theorem 1.5. Hint: the simples way is to use Definition 1.4 to characterise both $X(t)$ and $Y(t)$.

Exercise 4 In a web server, customers arrive at a rate of 0.5 jobs per second according to a Poisson process. What is the probability that:

- In one second we do not observe any job arrival;
- In one second we observe more than 3 jobs arrivals;
- Given that in the time interval $(0, 5]$ we have 3 arrivals, determine the distribution of the number of arrivals in $(6, 20]$.

Recall the properties of Poisson processes.

Exercise 5 Use Definition 1.4 to design an algorithm that generates a potentially infinite stream of Poisson arrivals.

Chapter 2

Queueing systems

2.1 Introduction to queueing systems

Queueing systems play a pivotal role in the assessment of performance of computer and telecommunication systems. They allow for the modelling and evaluation of the resource contention that characterises these types of systems: for example, a system may be equipped with a processor (the resource) and several jobs compete for its usage.

Any queueing system consists of three parts:

- An *Arrival process* that describes the way the jobs arrive at the system;
- A *waiting room* where the customers wait for their service;
- A *Service room* where the customers are served. The time spent in the service room is called service time.

We may have two equivalent ways of thinking at the service time of a job. The first consists in modelling the *job size* (or the *service time requirement*) as a random variable with a certain distribution and consider a server with constant unitary speed. In this view, when a job arrives at the system it has a certain random size (or *service demand* which is consumed during the time spent in the service room with the processor speed. The second view consists in thinking that the jobs are identical and the service time is a random variable. In single queueing systems, the two approaches are identical, but in queueing networks (i.e., when we compose several queueing systems) we may encounter some problems with the former interpretation. For example, if we say that jobs correspond to packets and the service demand corresponds to the job size, we expect that once a job is served in a queue and moves to another queue, the service times at the two stations should be correlated: if the packet is large, they will be both long. In Chapter 3 we will discuss some possible ways to address this problem.

The *scheduling discipline* or *queueing discipline* is the policy adopted by the system to assign the resource(s) to the jobs waiting to be served. These can be classified in:

- *Preemptive:* if a job that is in service can be put back in the waiting room because of some event. Preemptive policies can be further divided into policies with *resume* and *without resume* (*with restart*): in the former case if we preempt a job, once we it returns in service, it continues from the point it was left. In the latter case, if a job is preempted, the work done for is lost.
- *Non-preemptive:* a job that is in service does never leave the service room unless its service is finished.

Examples of important scheduling disciplines are:

- *FCFS:* the acronym stands for First Come First Served, i.e., jobs are served according to their arrival order. The policy is non-preemptive.
- *LCFS:* the acronym stands for Last Come First Served, i.e., jobs are served in the inverse arrival order. If the policy is preemptive, then the jobs enter in service as soon as they arrive. In case of preemption both the restart or the resume policies can be implemented. In the case of LCFS with preemption and resume, we use the acronym *LCFSPR*.
- *RR:* this is the Round Robin discipline, i.e., the system defines a time slide Δt and assigns the resource to each job in the system for at most Δt units of time. The discipline is preemptive with resume since the job looses the resource either if it finishes its work or because the time slice expires. If $\Delta t \to 0$, then we can imagine that the capacity of the processor is shared among all the customer present in the system and we call the RR as *Processor Sharing*, i.e., *PS*.
- *SJF:* the Shortest Job First discipline assumes that the size of the jobs in the waiting room are known. SJF is a non-preemptive discipline that chooses the smallest job in the waiting room to be put in service, regardless its arrival time.
- *SRPT:* the Shortest Remaining Processing Time is a preemptive discipline with resume that ensures that at each time epoch the job with the shortest remaining work to be done is in service. This discipline plays an important role because it is known that it minimises the expected response time with respect to any other discipline. However, large jobs may suffer problems of starvation and hence its fairness is not a strength.
- *RAND*: the Random policy is non-preemptive and when the server is free, it chooses one of the jobs in queue at random. If not differently specified, we assume that this random selection occurs according to a uniform distribution among the jobs.

For each discipline, we have shown the acronym that is used in Kendall's notation that we are going to describe in the next section. The following definition identifies the so-called *work-conserving* disciplines. This class includes most of the disciplines mentioned above.

Definition 2.1 (Work conserving discipline) A queueing discipline is work-conserving if:

- it never leaves idle a server that is allowed to work;
- it never wastes the work done on a job.

Thus, based on our previous definitions, all the disciplines are work-conserving with the exception of those which implement a pre-emptive discipline with restart. In the following paragraph, we will write *CONS* to identify any work-conserving discipline.

Is preemption an important feature of scheduling disciplines? The answer is positive. For example, if we consider a single server queueing system with a Poisson arrival process, i.e., the counting process of the number of arrivals is Poisson, then we have that FCFS, LCFS and RAND have all the same expected response time. However, if the job size has high variability, the LCFSPR has a better performance than the others (and is independent of the distribution of the job size) while with low variability the expected response time becomes higher.

2.1.1 Kendall's notation

Queueing systems are classified according to *Kendall's notation*. This is a way to describe the characteristics of the queues. According to this approach, we describe a queueing systems with the string A/B/m/K/P/D where:

- A and B describe the inter-arrival times of the jobs and the distribution of the service times. They are replaced by letters that describes these distributions, such as:
  - M: denotes the exponential distribution; Notice that if the first slot (A) is M, then this means that the arrival process is a Poisson process;
  - D: denotes the deterministic distribution;
  - E_k: denotes the Erlang distribution with k stages;
  - PH: phase-type distributions, i.e., a distribution defined in terms of the time to absorption of a continuous time Markov chain;
  - G: general distribution, sometimes denoted by GI to stress the fact that it is independent of any other process characterising the queue and inter-arrivals time are independent;

  - MAP: this acronym stands for Markovian Arrival Process and allows the arrival process to have interdependency among the inter-arrival times.

- m denotes the number of identical servers available in the system.
- K is the capacity of the queue, i.e., the maximum amount of jobs that can be stored in the waiting room and in the service room.
- P is the population size, i.e., the maximum amount of jobs that can try to enter in the system.
- D is the scheduling discipline.

In general, if K and P are omitted, they should be intended as ∞. Whereas if D is omitted, then it is implicitly considered as a FCFS discipline. Let us give some examples:

- M/M/1 is a queue with infinite capacity and infinity population, we have a Poisson arrival process and an exponentially distributed service time. The queueing discipline is FCFS.
- G/M/1 is a queue with an arrival process characterised by general inter-arrival times, exponentially distributed service times and FCFS queueing discipline. Population and buffer size are infinite.
- M/M/m/K is a queue with Poisson arrival process and exponentially distributed service time, infinite population, FCFS queueing discipline, m servers and finite capacity K.

2.1.2 Performance indices of queueing systems

In this book, we use queueing system to assess the performance of computer systems, specifically of software architectures. Therefore, it becomes of pivotal importance the definition of performance indices on the mathematical model that reflect important behaviours of the associated physical system. One important index is the number of jobs in the system at a certain epoch t, denoted by $N(t)$. This is clearly a stochastic process since, in general, we are not able to predict how many jobs there will be in the system at time t, but we may be able to give a probabilistic characterisation of this quantity. The distribution of $N(t)$ when $t \to \infty$ is particularly important. Indeed, we can define the random variable N as:

$$N = \lim_{t \to \infty} N(t)\,, \tag{2.1}$$

whenever this limit exists. In some cases, we are able to give the distribution of N, but we are probably more interested in its expectation that will be denoted by $E[N]$ or, more simply, by $\overline{N}$. Henceforth, in this section, we assume that the system admits a stationary behaviour (we will discuss the conditions under

which this is true in the following sections), or, equivalently that the system is stable[1].

By N_w we denote the r.v. modelling the number of job in the waiting room, and its expectation is $E[N_w]$ or $\overline{N}_w$. Then, W is the random variable denoting the stationary waiting time of a job, i.e., the time spent in the waiting room. $E[W]$ or $\overline{W}$ denote its expectation. S is the service time, i.e., the time spent in the service room. Its expectation if $E[S]$ and is usually denoted as μ^{-1}, when the service rate is constant. Indeed, μ usually models the service rate of the queueing model.

The (stationary) response time R is the total time spent in the system, i.e., the sum of the times spent in the waiting room and in the service room:

$$R = S + W\,,$$

and, passing to the expectations, we have:

$$E[R] = E[S] + E[W] = \mu^{-1} + E[W]\,.$$

The (stationary) throughput X of the queueing system is the expected number of jobs that leave the system in the unit of time when the queue is in equilibrium. Finally, we define the utilisation of the queue U, i.e., the fraction of time that the server is busy. Clearly, we have $0 \leq U \leq 1$.

2.2 Little's law and Little's theorem

Little's law and theorem have important applications in the analysis of queueing systems thanks to the very loose conditions that they require to be applied. They relate the expected number of jobs, the expected response time and the throughput of a queueing system.

Assume that we observe a continuous time system in an interval $[0, t]$. Then, let us define the following quantities:

- $A(t)$: number of arrivals observed in $[0, t]$;
- $C(t)$: number of departures in $[0, t]$.

Therefore, $N(t) = A(t) - C(t)$ is the number of customers in the systems at time t assuming that at time 0 it is empty. Now, we need a quantity which is called *cumulative work* and it measures the amount of pending work that has been stored in the system in the interval $[0, t]$:

$$W(t) = \int_0^t N(\tau)d\tau = \int_0^t (A(\tau) - C(\tau))d\tau\,.$$

In Figure 2.1, we show a realisation of the stochastic process $N(t)$. At time t_1 we have the first arrival followed by a second arrival at t_2. At time epoch t_3,

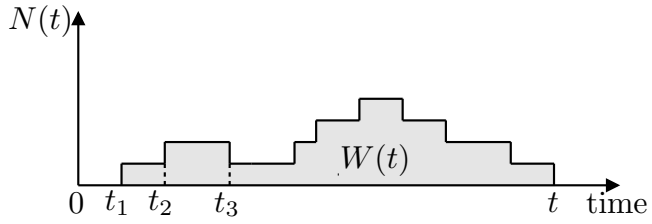


Figure 2.1: Little's law: example of realisation of the stochastic process associated with $N(t)$.

we observe the first departure, and so on. The expected number of jobs in $[0, t]$, $\overline{N}(t)$ is then:

$$\overline{N}(t) = \frac{W(t)}{t} ,$$

Now, let us derive the expected response time $\overline{R}(t)$ of the jobs in $[0, t]$ based on the quantities that we have defined so far. To simplify our reasoning, assume that both at time 0 and t the system is empty. Therefore, we had $C(t)$ job completions in $[0, t]$, and we need the cumulative time spent in the system by all the jobs. This is nothing more than $W(t)$, and hence we can write:

$$\overline{R}(t) = \frac{W(t)}{C(t)} .$$

Finally, let $X(t)$ be the throughput of the system in $[0, t]$, i.e., $X(t) = C(t)/t$, we can write:

$$\overline{N}(t) = \frac{W(t)}{t} = \frac{W(t)}{t}\frac{C(t)}{C(t)} = \overline{R}(t)X(t) .$$

This relation is known as Little's law although it has a formal proof and hence it is a theorem. Notice that, the reasoning that we have shown is rather informal and relies on the assumption of the empty system at times 0 and t. However, Little's law holds even without requiring these assumptions.

Theorem 2.1 (Little's law) Given a queueing system without internal loss or generation of jobs, we have that the following relation holds for any finite time horizon t:

$$\overline{N}(t) = \overline{R}(t)X(t) .$$

It is interesting to note that the hypothesis of Little's law are loose: we do not require neither a specific arrival process nor a special service time distribution. This is the reason why it will be widely used in analysing both queueing systems and queueing networks.

[1] The notion of stability of a queueing system can have different interpretations. It is outside the scope of this book to investigate this topic and we consider the requirement of existence of Limit (2.1) to be equivalent to the requirement that the stationary probability of finding the empty queue to be strictly positive.

2.1 Example
In a small post office we notice that during the 6 opening hours the clerks have served 120 customers. We know that we have counted an average of 10 persons sitting at the chairs and 5.5 at the 6 available desks. We desire to compute the expected time spent by a customer in the post office and the expected service time at the desks.

Solution. In total, we have $10 + 5.5 = 15.5$ persons in the office. Since we have served 120 customers in 6 hours (360 minutes), this means that we have a throughput of $X(360) = 120/360 = 0.33$ customers per minute. Therefore, by Little's law:

$$\overline{R}(t) = \frac{\overline{N}(t)}{X(t)} = \frac{15.5}{0.33} \simeq 47\text{minutes}\,.$$

In order to understand the expected time spent by a customer at the desk, we must think at the set of the desks as a single sub-system. This sub-system sees exactly the same throughput of the entire post office, but the expected number of customers is 5.5. Then, we have that the customers have spent in the average $5.5/0.33 \simeq 17$ minutes at the desk. We could newly apply Little's law to obtain the service time by considering the waiting room as another sub-system, or more simply, we can subtract the expected response time and the expected service time.

Little's theorem can be derived by Little's law by letting $t \to \infty$.

Theorem 2.2 (Little's theorem) Let us consider a queueing system without internal loss or generation of customers and assume that the following limits exist and are finite:

$$\lambda = \lim_{t\to\infty} A(t)/t$$

$$\overline{N} = \lim_{t\to\infty} \overline{N}(t) \tag{2.2}$$

$$\overline{R} = \lim_{t\to\infty} \sum_{i=1}^{C(t)} \frac{r_i}{C(t)} \tag{2.3}$$

where $C(t)$ is the number of jobs served in $[0, t]$ and r_i is the response time of the i-th service. Then, the following relation holds:

$$\overline{N} = \lambda \overline{R}\,.$$

Notice that, while in Theorem 2.1 we were using the throughput in $[0, t]$, for Theorem 2.2 we use the intensity of the arrival process. This step is crucial, and it follows from the assumptions on the existence of the limits. Indeed, if those limits are well defined and are finite, then the queueing system is stable and for $t \to \infty$ the throughput of the system must be balanced with the intensity of the arrival process. One further observation is that since Theorem 2.2 holds

for finite time horizons, then we can apply it also to queueing systems whose arrival rates are higher than their service rates. In Section 2.3 we discuss some sufficient conditions that grants the hypothesis of Little's theorem.

2.3 Invariant properties of queueing systems

Many queueing systems share some common behaviours that we are going to analyse in this section. The results that we are going to state will be valid for most of the systems that we are going to consider in this book.

2.3.1 Stability of the queueing system

The concerns about the stability of queueing systems are connected to the fact that we wish to study their behaviour when $t \to \infty$, i.e., their long run behaviour. Many important information can be drawn by the transient analysis of these systems, i.e., in finite time horizons, but this is out of the scope of this book. Indeed, despite what the intuition may suggest, transient analysis is in general more complicated than the stationary one.

However, when $t \to \infty$, we have to understand if the behaviour of our queueing system is well-defined. For example, let us consider a M/M/1 queue with arrival rate $\lambda = 4$ jobs per second and service rate $\mu = 2$ jobs per second. What happens when $t \to \infty$? We observe that the speed of job arrival is higher than the speed of service, and that, in the average, every second we accumulate 2 jobs in the queue. Let $N(t)$ be the stochastic process underlying the population of in the system. For any finite t, in principle, we can derive $Pr\{N(t) = n\}$, i.e., the probability that at time t the system contains exactly n jobs. However, if $t \to \infty$, the limit

$$\lim_{t\to\infty} N(t) \tag{2.4}$$

cannot be finite becase, as t grows, more and more jobs are expected to be found in the system (recall that the M/M/1 queue has infinite capacity). We will see that, in this example, we say that the queue is not stable.

Definition 2.2 (Stability of a queue) We say that a queue is stable if, in the long run ($t \to \infty$), we have a finite expected number of jobs in the sytem and the expected response time for the jobs is finite.

Let us give the following intuitive rule for the stability of a simple class of queueing systems:

Proposition 2.1 (Stability of G/G/m/CONS systems) Let us consider a queueing system with the following characteristics:

- Infinite capacity,

- Arrival rate with constant intensity λ,
- m identical servers,
- Expected service time μ^{-1},
- A work-conserving queueing discipline.

Then, the queueing system is stable if:

$$\lambda < m\mu\,.$$

The stability of a queueing system is necessary for the existence of Limit (2.4). Observe that, in general, it is not true that if $\lambda < \mu$ then waiting room of the queue is always empty. Indeed, the randomness of the inter-arrival and service times usually creates events in which at the arrival of a job the server is busy and hence it must enter the waiting room. It will enter in service later, according to the scheduling discipline. Moreover, the case $\lambda = \mu$ gives a stale system only if the inter-arrival and service times are deterministic, which is not an interesting case.

The notion of stability of a queue can have several different definitions that may vary according to the study one is willing to perform. Among these, which we mention:

- The queue is stable if the stationary probability of the empty queue exists and is strictly positive;
- The queue is stable if the expected number of jobs in the system when $t \to \infty$ is finite;
- The queue is stable when Limit (2.4) exists and has a finite mean;
- If the queue has an underlying Markov chain, then this is ergodic.

For the queues that we study in this book, the four possible definitions are interchangeable, but in general this is untrue and the reader should put some care in specifying which stability condition he/she intends to consider for special analyses.

Let us consider now, a more complicated case. Assume that $\lambda(n)$ is the intensity of the arrival process when the system has n jobs and, similarly, $\mu(n)$ is the service rate of the system when it contains n jobs. Then, we give the following proposition without proving it.

Proposition 2.2 (Stability of queueing systems with state dependent arrival and service rates) Let us consider a queue with infinite capacity, state dependent arrival rate $\lambda(n)$ and $\mu(n)$. Then, a sufficient condition for the queue to be stable is that there exists a state n^* such that for all $n > n^*$ we have that $\lambda(n) < \mu(n)$.

Basically, Proposition 2.2 states that the queue is stable if, from a certain state n^* the arrival rate become lower than the service rate. Notice that this condition is a generalisation of Proposition (2.1). In fact, if we have m identical servers, then for all the states with $n \geq m$ customers, the expected service rate is $m\mu$ and hence $\lambda < m\mu$

2.3.2 The utilisation of the single server queue with work-conserving queueing discipline

Let us consider a single server queue with general arrival process and general service time. In this paragraph, we assume the view in which the job has expected size μ^{-1} and the single server has a constant speed of 1 job per unit of time. For each sufficiently large interval of time Δt, the average work that enters in the system per unit of time is $\lambda\mu^{-1}$. In fact, μ^{-1} is the average job size and λ is the expected amount of jobs that arrive at the system per unit of time. Let U be the utilisation of the server, i.e., the faction of time that the server is busy. In the period Δt, the server has worked for a fraction of time U and since it has a constant speed of 1 job per unit of time, we have that it has served U amount of work. If the system is stable, in a sufficiently large interval of time, the amount of work that enters in the system must be equal to the amount of work that exists the system. Thus, we have the following proposition.

Proposition 2.3 (Utilisation of the single server queue) In a single server queue with constant arrival rate and service rate, infinite capacity and work-conserving queueing discipline, the utilisation of the queue is:

$$U = \frac{\lambda}{\mu},$$

where λ is the intensity of the arrival process and μ^{-1} the expected job size or equivalently the expected service time.

We may also derive another interesting property of single server queues with work-conserving disciplines. Suppose that we observe the queue in the time interval $(0, t)$ and that during this period the server has been busy for B units of time and idle for $t - B$ units of time, with $0 < B < t$. Moreover, let $C(t)$ be the total number of job completions up to time t. Then, by definition, we have:

$$X(t) = \frac{C(t)}{t},$$

which can be rewritten as:

$$X(t) = \frac{C(t)}{B}\frac{B}{t}.$$

Observe that $C(t)/B$ is the service rate μ and that B/t is the utilisation of the station. Taking the limit $t \to \infty$, we have the following important result.

Proposition 2.4 In a single server queue with constant arrival rate and service rate and work-conserving queueing discipline, we have the following relation between the throughput and the utilisation:

$$X = \mu U$$

Notice that, as expected, in stable systems we have $X = \lambda$.

2.4 PASTA property

In this section, we introduce the *Poisson Arrivals See Time Averages* property, usually known as PASTA or the *random observer property*. This property is widely used for the analysis of queueing systems and allows for the simple derivation of some performance indices.

Theorem 2.3 (PASTA) In a queueing system with a Poisson arrival process, the distribution seen by a job immediately before its arrival is the same as the random observer's one.

Proof. Let us consider a queueing system with Poisson arrival process and let us denote by $PA(n)$ the probability of seeing state n immediately before the arrival. Since we have a Poisson arrival process, we can use the memoryless property to write:

$$PA(n) = \lim_{\Delta t \to 0} Pr\{\text{the system is in state } n \text{ at } t | \text{arrival in } (t, t+\Delta t)\}$$

since the probability of arrival in $(t, t+\Delta t)$ is independent of the last arrival time.

Then, we can write:

$$\begin{aligned} PA(n) &= \lim_{\Delta t \to 0} Pr\{\text{the system is in state } n \text{ at } t | \text{arrival in } (t, t+\Delta t)\} \\ &= \lim_{\Delta t \to 0} \frac{Pr\{\text{the system is in state } n \text{ at } t \wedge \text{arrival in } (t, t+\Delta t)\}}{Pr\{\text{arrival in } (t, t+\Delta t)\}} \\ &= \lim_{\Delta t \to 0} \Pr\{\text{arrival in } (t, t+\Delta t) | \text{the system is in state } n \text{ at } t\} \\ &\quad \cdot \frac{Pr\{\text{the system is in state } n \text{ at } t\}}{Pr\{\text{arrival in } (t, t+\Delta t)\}} \end{aligned}$$

Since the arrival process is independent of the state of the queue at any time epoch, we have that:

$$PA(n) = Pr\{\text{the system is in state } n \text{ at } t\},$$

as required. □

Let us apply the PASTA property to derive an important property of queueing systems. We desire to characterise the residual service time seen by a customer at its arrival epoch conditioned to the fact that a job is actually being served. First, we give a definition of *Residual life of a job*.

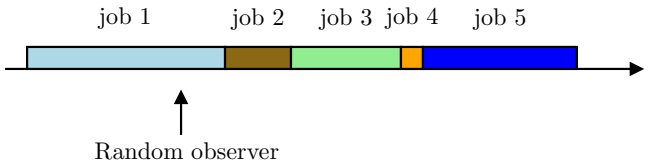


Figure 2.2: The random observer has more probability to find a large job in service than a small one.

Definition 2.3 (Residual life) The residual life of a customer in service is the amount of remaining service time of a customer in service from the point of view of a random observer.

By the memoryless property of the exponential distributions, we know that in queues with this type of service time, the residual life is identical to the original service time. However, this is in general untrue. By the PASTA property, if the queue has a Poisson arrival process, then the residual life seen by a customer at the moment of its arrival has the same statistics of the residual life as given in Definition 2.3. In Figure 2.2, we intuitively support the idea that the random observer does not see half of the residual service time of a generic job because, at its arrival time, it has more probability to see a large job in service rather than a small one. A formal derivation of this property is possible by resorting to the renewal theory. Let $f(x)$ be the p.d.f. of the service time distribution, then the probability that a random observer sees a job with service time x is directly proportional to x and weighted on the relative occurrence of such intervals, i.e., $f(x)dx$. Therefore, if we call $g(x)$ the service time distribution seen by a random observer, we have:

$$g(x)dx = Kxf(x)dx\,,$$

where K has to be determined. Since $g(x)$ must be a p.d.f., the following relation must hold:

$$\int_0^\infty g(x)dx = 1\,,$$

i.e.:

$$\int_0^\infty Kxf(x)dx = 1 \quad \Longrightarrow \quad K = \mu\,,$$

where $1/\mu$ is the first moment of the service time distribution. Let Y be the random variable modelling the residual life and X the random variable modelling the service time of the customer in service, then:

$$Pr\{Y \leq y | X = x\} = \frac{y}{x} \quad 0 \leq y \leq x\,,$$

since we have randomly chosen a point in this selected interval. Now, we can derive the p.d.f. of the joint distribution of X and Y (using Bayes' rule):

$$Pr\{y < Y \leq y + dy \wedge x < X \leq x + dx\} = \left(\frac{dy}{x}\right)\mu x f(x)dx = \mu f(x)dy\ dx\,,$$

with $0 \leq y \leq x$. Thus, we have that the unconditional p.d.f. of Y is:

$$f_Y(y) = \mu \int_y^\infty f(x)dx = \mu(1 - F(y)) \,.$$

As a sanity check, let us consider the case in which $F(x) = 1 - e^{-\mu x}$, i.e., the service time is exponentially distributed. Then, we have:

$$f_Y(y) = \mu(1 - (1 - e^{-\mu x})) = \mu e^{-\mu x} \,,$$

which is the p.d.f. of the exponential random variable with rate μ as expected. We are now in position to give the following important result.

Proposition 2.5 (Average residual life) The average residual life for a service time with c.d.f. $F(x)$ is:

$$E[Y] = \frac{M_2 \mu}{2} \,,$$

where:

$$M_2 = \int_0^\infty x^2 f(x)dx$$

is the second moment of the service time distribution. Equivalently, we have:

$$E[Y] = \frac{1}{2}\left(\frac{1}{\mu} + \sigma^2 \mu\right) \,,$$

where σ^2 is the variance of the service time distribution.

Proof. We need to compute:

$$\int_0^\infty y f_Y(y)dy = \int_0^\infty \mu y(1 - F(y))dy$$

Let us first consider the indefinite integral and use the decomposition and the rule of integration by parts:

$$\begin{aligned}\int \mu y(1 - F(y))dy &= \mu\left(\frac{y^2}{2} - \left(\frac{y^2}{2}F(y) - \int \frac{y^2}{2} f(y)dy\right)\right) \\ &= \frac{\mu y^2}{2}(1 - F(y)) + \frac{\mu}{2}\int y^2 f(y)dy \,.\end{aligned}$$

Thus, we have:

$$\int_0^\infty y f_Y(y)dy = \frac{\mu y^2}{2}(1 - F(y))\Big|_0^\infty + \frac{\mu}{2}M_2 = \frac{M_2 \mu}{2} \,.$$

Now, by recalling that $\sigma^2 = M_2 - \mu^{-2}$, we conclude the proof. □

2.5 M/M/1 queue

The M/M/1 queueing system consists of a single server that works on jobs with exponentially distributed size. Arrivals occur according to a Poisson process. Service times are independent of the arrival process and the queueing discipline is FCFS. The analysis of a M/M/1 queue widely exploits the memoryless property of both the inter-arrival times and service times (and, of course, their independence). We propose two ways of studying the M/M/1 system. The first is based on the PASTA property and will allow us to derive the average performance indices in a very simple manner. The second approach is based on the analysis of the stochastic process underlying the queueing system. This is a Continuous Time Markov Chain and we will propose its stationary analysis.

2.5.1 Derivation of the mean performance indices

We know that since the system has a single server and infinite capacity, the stability condition is $\lambda < \mu$. By the PASTA property, at the customer arrival, it sees on average $\overline{N}$ job. Because of the FCFS scheduling discipline, the arriving job has to wait the service of all the customers present at the queue at its arrival before beginning its own service. By the memoryless property of the exponential distribution, the job that is in service (if any) has a residual life that is exponentially distributed with rate μ. Thus, the arriving job has a waiting time that is on average:

$$\overline{W} = \overline{N}\frac{1}{\mu}\,,$$

and the average service time is $1/\mu$. Thus, we can express the average response time as:

$$\overline{R} = (\overline{N}+1)\frac{1}{\mu}\,.$$

Unfortunately, this equation has two unknowns, $\overline{N}$ and $\overline{R}$. But we have also Little's law because its assumptions are satisfied, in particular the system is stable. Thus, we have:

$$\begin{cases} \overline{R} = (\overline{N}+1)\mu^{-1} \\ \overline{N} = \lambda\overline{R} \end{cases}$$

With a few algebraic derivation, we get the expression of the expected response time and number of jobs:

$$\overline{R} = \frac{1}{\mu-\lambda}\,, \quad \overline{N} = \frac{\rho}{1-\rho}\,,$$

where $\rho = \lambda/\mu$ is the utilisation of the queue, also said *offered load* or *load factor*.

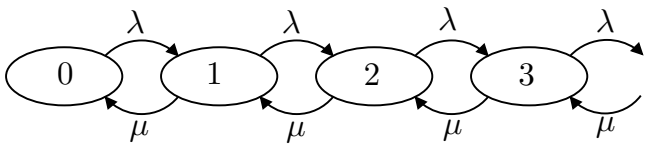


Figure 2.3: CTMC underlying a M/M/1 queueing system.

2.5.2 Markov chain analysis

In this part, we derive the results of the previous section in another way that will also unveil some interesting characteristics of the M/M/1 queueing system. Thanks to the Markov chain analysis, we are able to get not only the average number of jobs in the system, but also the distribution of the jobs. Intriguingly, beside the different approach, we will use exactly the same assumptions although for different reasons.

The model that we propose for the M/M/1 queue will be able to provide us insights on its expected performance indices. Let us describe the stochastic process underlying the queue length, i.e., $n(t) \in \mathbb{N}$ and $t \in \mathbb{R}$. At a certain epoch t_0, $n(t_0) = i$ and $i > 0$, we have two possible events that determine the change of state in the next future:

- an arrival of a new job that takes the state from i to $i+1$. For every t_0, the distribution of the time to be waited for this event is exponentially distributed with rate λ, which is also the intensity of the arrival process.
- a departure of a job in the system that takes the state from i to $i-1$. For every t_0, the distribution of the time that we must wait for this event is exponential with rate μ.

The two delays to the next departure and to the next arrival are independent and exponentially distributed. The minimum of the two is exponentially distributed with rate $\lambda + \mu$ and is the residual residence time in state i. The case of $i = 0$ is even simpler. We conclude that $n(t)$ is continuous time Markov chain because for each state i and each time epoch $t_0 \in \mathbb{R}$, the current state of the process contains all the information for the probabilistic characterisation of its future evolution. The Markov chain is shown in Figure 2.3. The system of global balance equations is:

$$\begin{cases} \pi(0)\lambda = \pi(1)\mu \\ \pi(i)(\lambda+\mu) = \pi(i+1)\mu + \pi(i-1)\lambda \quad i > 0\,, \end{cases} \tag{2.5}$$

where $\pi(i)$ is the stationary probability of state i, assuming it exists. Let us consider the system for $i = 0$, we immediately get:

$$\pi(1) = \pi(0)\frac{\lambda}{\mu}\,.$$

Now, let us consider the equation whose left-hand-side is π_1. We may derive the following relation:

$$\pi(2) = \pi(0) \left(\frac{\lambda}{\mu}\right)^2 ,$$

and, by induction,

$$\pi(n) = \pi(0) \left(\frac{\lambda}{\mu}\right)^n .$$

In stability, this equation gives the stationary distribution of the queue length as function of $\pi(0)$. How can we derive $\pi(0)$? We may impose the normalising condition of the probability distribution:

$$1 = \sum_{i=0}^{\infty} \pi(i) = \sum_{i=0}^{\infty} \pi(0) \left(\frac{\lambda}{\mu}\right)^n = \pi(0) \sum_{i=0}^{\infty} \rho^n ,$$

where $\rho = \lambda/\mu$ is the *load factor* of the queue. If $\rho < 1$, i.e., $\lambda < \mu$, the geometric series converges to $(1-\rho)^{-1}$ and thus we have $\pi_0 = 1 - \rho$ and:

$$\pi(n) = (1-\rho)\rho^n , \quad \rho < 1 . \tag{2.6}$$

Notice that the condition derived algebraically is coherent with the observation that, in a system with infinite capacity and without customer rejection, the arrival rate must be lower than the service rate. From Equation (2.6), we can derive the expected number of jobs in the system as:

$$E[N] = \sum_{n=1}^{\infty} n\pi(n) = \frac{\rho}{1-\rho} .$$

In fact, we have:

$$\begin{aligned} E[N] &= \sum_{n=1}^{\infty} n(1-\rho)\rho^n = (1-\rho)\rho \sum_{n=1}^{\infty} n\rho^{n-1} \\ &= (1-\rho)\rho \sum_{n=1}^{\infty} \frac{\partial \rho^n}{\partial \rho} = (1-\rho)\rho \frac{\partial}{\partial \rho} \sum_{n=1}^{\infty} \rho^n = (1-\rho)\rho \frac{\partial}{\partial \rho} \frac{\rho}{1-\rho} \\ &= (1-\rho)\rho \frac{1}{(1-\rho)^2} = \frac{\rho}{1-\rho} . \end{aligned}$$

Thus, by Little's theorem, we can derive the expected response time:

$$E[R] = \frac{E[N]}{\lambda} = \frac{1}{\mu - \lambda} .$$

The system utilization is $1 - \pi(0) = \rho$, as expected, and the expected waiting time is:

$$E[W] = E[R] - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)} .$$

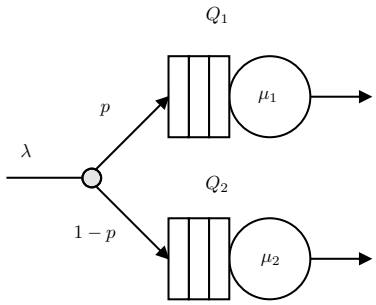


Figure 2.4: Optimal routing between asymmetric systems.

2.2 Example
Let us consider a system consisting of two functionally equivalent servers that have different speed. The former is capable of serving 1 job per second and the latter 3 jobs per second. When a request arrives at the system, it is sent with probability p to the slowest server and with probability $1-p$ to the other. We can see a sketch of the system in Figure 2.4. We assume that $\mu_1 = 1\text{j/s}$ and $\mu_2 = 3\text{j/s}$. We address the following questions:

- Determine the stability condition on the arrival rate $\lambda < \lambda_{\max}$ for the system assuming that p can be chosen in an optimal way.
- Given λ, establish the conditions on p that ensure the stability of the system.
- If the arrival process is a Poisson process and the service times are independent and exponentially distributed, assuming $\lambda = 0.5\lambda_{\max}$ and $\lambda = 0.9\lambda_{\max}$ determine the optimal value of p to minimise the expected response time of the system and the corresponding expected response time. Is the optimal value for the routing probability independent of the arrival rate?

Solution. The maximum capacity of our system is $\mu_1 + \mu_2 = 4$ jobs per second, i.e., the stability condition is:

$$\lambda < \lambda_{\max} = 4\text{j/s}\,.$$

Fixed λ, we have the following constraints for p which are derived by imposing the stability of both Q_1 and Q_2:

$$\lambda p < \mu_1 \ \wedge \ \lambda(1-p) < \mu_2 \ \wedge \ 0 \leq p \leq 1$$

that can be rewritten as

$$1 - \mu_2/\lambda < p < \mu_1/\lambda \quad 0 \leq p \leq 1 \tag{2.7}$$

Notice that if $\lambda < 1$, the lower limit of the inequality becomes 0 and the upper limit becomes 1 (included) since we can send all the jobs to the first or to the second queue (although it may be not optimal, clearly). The last point is the derivation of the optimal value p^* for the routing probability. Before proceeding with the algebra, let us try to answer to the last question of the bullet: is the optimal routing probability independent of the arrival rate? The naif (but *wrong*) answer to this question is that we should send the jobs to the queues in proportion with their service rate, i.e., 1/4 to Q_1 and 3/4 to Q_2. We insist that this is wrong. Indeed, consider the case in which $\lambda \to 0$, i.e., we have rare arrivals. In this case, we should send all the requests to the fastest server, i.e., Q_2, since the probability of finding another job in service is very small, and hence the minimisation of the expected response time is clearly obtained in this way. Conversely, if we consider $\lambda \to \lambda_{\max}^-$ we have that the stability conditions become:

$$1 - \frac{\mu_2}{(\mu_1 + \mu_2)^-} < p < \frac{\mu_1}{(\mu_1 + \mu_2)^-}\,,$$

that becomes $(1/4)^- < p < (1/4)^+$, i.e., $p^* = 1/4$. Thus, we have that the optimal routing probability *depends* on the intensity of the arrival rate.

In order to be able to derive the optimal routing probability, we must understand what type of queues are Q_1 and Q_2. Indeed, they are (independent) $M/M/1$ queues, since the arrival process at the two queues are independent Poisson processes with intensity λp and $(1-\lambda)p$ by the property of slitting of Poisson processes. The expected number of jobs in the first queue is $E[N_1] = \rho_1/(1-\rho_1)$ and similarly in Q_2, $E[N_2] = \rho_2/(1-\rho_2)$, where $\rho_1 = \lambda p/\mu_1$ and $\rho_2 = \lambda(1-p)/\mu_2$. Thus, we can derive:

$$E[N] = E[N_1 + N_2] = E[N_1] + E[N_2] = \frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2}\,,$$

By Little's theorem, we have that:

$$E[R] = \frac{E[N]}{\lambda}$$

where we consider the two queues as a single system. Finally, we find the optimal value p^* by studying the roots of $\partial E[R]/\partial p$. We have:

$$\frac{\partial E[R]}{\partial p} = \frac{\mu_1}{(\mu_1 - \lambda p)^2} - \frac{\mu_2}{(\mu_2 - (1-p)\lambda)^2}$$

If we assume $\lambda = 0.5\lambda_{\max}$, then we have two roots of the equation $p = (2-\sqrt{3})/2$ and $p = (2+\sqrt{3})/2$, among which only the first is acceptable, i.e., $p^* \simeq 0.13$. The case of $\lambda = 0.9\lambda_{\max}$ produces two roots: $(6-\sqrt{3})/18 \simeq 0.24$ and $(6+\sqrt{3})/18 \simeq 0.43$. However, by Condition (2.7) we have $1/6 < p < 5/18$, which means that only the first root is valid.

2.3 Example (Expected conditioned response time)
In this example we want to derive the expected stationary response time of a job whose service time x is known and the probability that it has to wait for the service of at least 2 jobs. We consider a M/M/1 queueing system.

Solution. We use the PASTA property and conclude that, since the arrival process is Poisson, we have that a customer that arrives at the system in equilibrium sees a distribution of the stationary probability that is identical to that seen by a random observer. Recall that the expected response time is given by the sum of the service time and the waiting time. In the FCFS discipline of the M/M/1 queue, the new customer sees in the average $E[N] = \rho/(1-\rho)$ jobs in front of him, thus the waiting time, in the average is $E[N]/\mu$ (since μ^{-1} is the service rate for each customer). Therefore, we have:

$$E[R|S = x] = x + \frac{E[N]}{\mu}\,.$$

As sanity check, we may integrate this expression to obtain $E[R]$ by using the law of total probability:

$$E[R] = \int_0^\infty E[R|S = x]f(x)dx\,,$$

where $f(x)$ is the p.d.f. of the exponential distribution with parameter μ:

$$E[R] = \int_0^\infty \left(x + \frac{\rho}{1-\rho}\frac{1}{\mu}\right)\mu e^{-\mu x} = \left.\frac{e^{-\mu x}(\lambda x - \mu x - 1)}{\mu - \lambda}\right|_0^\infty = \frac{1}{\mu - \lambda}\,,$$

as expected. The probability of finding at least 2 jobs is $1 - \pi(0) - \pi(1) = 1 - (1-\rho) - (1-\rho)\rho = \rho^2$.

2.5.3 Response time distribution of M/M/1 queues

So far, we have derive the expected stationary indices of the M/M/1 queue. In this part, we propose a more detailed index, i.e., the distribution of the response time.

To derive the result, we need three ingredients: (i) the stationary distribution of a M/M/1 queue has the geometric expression given by Equation (2.6); (ii) PASTA property since we will reason on an arbitrary arriving customer; (iii) the exponential distribution is the only non-negative continuous distribution with the memoryless property.

Theorem 2.4 The stationary response time of a stable M/M/1 queue is exponentially distributed with parameter $\mu - \lambda$.

Proof. Let us consider an arbitrary customer arriving according to a Poisson process. This job sees the steady-state distribution of the system. For each job it finds in the queue at its arrival epoch, it has to wait an exponentially distributed random time and, of course, we have to take into account its own service time. Therefore, the response time is the sum of L independent exponentially distributed random variables with rate μ, where $Pr\{L = 1\} = \pi(0)$, $Pr\{L = 2\} = \pi(1)$, ... We can imagine the sum of a geometrically distributed number of independent random variables as the result of the following experiment: after summing each random variable, we flip a coin and with probability ρ we add another exponential random variable, with probability $1 - \rho$, we stop. This follows by the definition of the geometric random variable in terms of Benoulli experiments. Now, we can argue that this experiment is memoryless. Imagine that you have a timer that is exponentially distributed, call it X_1. When X_1 expires, you flip the coin and decide if starting another timer or returning X_1. In the first case, let X_2 be the second timer. When it expires you may return $X_1 + X_2$ or add another random variable. Let us consider an arbitrary instant t in this imaginary experiment, does the time that we have to wait until the end of the experiment depend on the history prior to t? The answer is negative. Indeed, the residual time of the exponential timer active at time t is exponentially distributed with rate μ by the memoryless property of the exponential distribution, and the number of other random variables that we are going to add is also geometrically distributed with parameter ρ. Thus, the distributing of the sum is memoryless, but the only memoryless random variable is the exponential one.

It remains to derive the parameter: we already know that the expected response time of the M/M/1 queue is $1/(\mu - \lambda)$, thus the parameter of the response time distribution must be $\mu - \lambda$.

2.6 M/G/1 queue

In the M/G/1 queue we have a Poisson arrival process, independent service times with identical general distribution a single server and a FCFS scheduling discipline. In this section, we present the derivation of the average stationary performance indices. We assume that the arrival process has intensity λ, while the service time distribution has expectation $1/\mu$ and second moment M_2. Once a job arrives at the queue, by the PASTA property, it sees the system with the same statistics observed by a random observer. When the job arrives at the system, it finds $N_w \geq 0$ jobs in the waiting room, where N_w is a random variable. Their service times are i.i.d. random variables denoted by $S_1, \ldots, S_{N_w}$. Moreover, we have to account for the residual time of the job in service, if any is present that we denote by the random variable ξ. Let us call W the random

time that the arrived job spends in the waiting room, then we have that:

$$\begin{aligned} E[W|N_w = n] &= E\left[\sum_{i=1}^{n} S_i\right] + E[\xi] = \sum_{i=1}^{n} E[S_i] + E[\xi] = \frac{n}{\mu} + E[\xi]\,, \quad n > 0 \\ E[W|N_w = 0] &= E[\xi] Pr\{\text{One job in service}|\text{empty waiting room}\} \end{aligned}$$

We can derive the unconditional expectation as:

$$E[W] = \sum_{n=1}^{\infty} \left(\frac{n}{\mu} + E[\xi]\right) Pr\{n \text{ jobs in the waiting room}\}$$

$$+E[\xi] Pr\{\text{One job is in the system}|\text{empty waiting room}\} Pr\{\text{empty waiting room}\}$$

$$= \frac{1}{\mu} \sum_{n=1}^{\infty} n Pr\{n \text{ jobs in the waiting room}\}$$

$$+E[\xi] \sum_{n=1}^{\infty} Pr\{n{+}1 \text{ jobs in the system}\} + E[\xi] Pr\{\text{Exactly 1 job in the system}\}\,.$$

In fact, we observe that the probability of having $n > 0$ jobs in the waiting room is identical to that of having $n + 1$ jobs in the system, since the service room cannot be empty if there are jobs in the waiting room. Moreover, the probability of having one job in service and the empty waiting room corresponds to the probability of having exactly one job in the system. Thus, we have:

$$E[W] = \frac{E[N_w]}{\mu} + E[\xi]\rho\,,$$

where $\rho = \lambda/\mu$ is the utilisation of the queue, i.e., the probability of finding the queue with one job in service (and an arbitrary number of jobs in the waiting room). Now, recall that Proposition 2.5 gives us the expected value of ξ, and by Little's theorem we have that $E[N_w] = \lambda E[W]$, thus:

$$E[W] = \frac{\lambda}{\mu} E[W] + \frac{M_2 \mu}{2} \frac{\lambda}{\mu}\,,$$

that, by solving on $E[W]$ gives us:

$$E[W] = \frac{M_2 \lambda}{2(1-\rho)}\,. \tag{2.8}$$

Equation (2.8) is known as the Pollaczek-Khinchine formula for the mean waiting time (P-K formula). It can be rewritten as:

$$E[W] = \frac{\rho + \lambda\mu\sigma^2}{2(\mu - \lambda)} \tag{2.9}$$

Starting from this relation we can obtain the mean number of jobs in the waiting room, by using Little's theorem:

$$E[N_w] = \lambda E[W]\,,$$

and the expected number of jobs in the system is:

$$E[N] = E[N_w] + \rho\,.$$

Finally, the expected response time can be derived as $E[R] = E[W] + \mu^{-1}$.

If we consult Equation (2.9) we notice that the expected waiting time depends on the mean and the variance of the service time. Specifically, larger variances for the service time imply higher response times.

2.4 Example
Let us compare the P-K formula for the expected response time with that of an M/M/1 queueing system. Recall that the variance of the exponential distribution with parameter μ is μ^{-2}. Thus, we can write:

$$E[R] = \frac{\lambda/\mu + \lambda\mu\mu^{-2}}{2(\mu-\lambda)} + \frac{1}{\mu} = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu} = \frac{1}{\mu-\lambda}\,,$$

as required.

2.5 Example (M/D/1 queue)
Let us consider the case in which we have a deterministic service time distribution μ^{-1}, i.e., $\sigma^2 = 0$. Notice that this is the case in which the expected response time is minimum. Thus, we can rewrite Equation (2.9) as:

$$E[R] = \frac{\lambda/\mu}{2(\mu-\lambda)} + \frac{1}{\mu} = \frac{2\mu-\lambda}{2\mu(\mu-\lambda)} = \frac{2-\rho}{2(\mu-\lambda)}\,.$$

By Little's theorem, we can derive the expected number of jobs in the queue:

$$E[N] = \frac{\lambda(2-\rho)}{2(\mu-\lambda)}\,.$$

2.6 Example
A hard disk serves the requests according to a FCFS order. The delay for retrieving some data consists of three parts:

- A rotation delay that is uniformly distributed between 0 and 6 milliseconds (we assume a HDD spindle speed of 10000 rotations per seconds);
- A seek time which is a uniform random variable between 0 and 16 milliseconds;
- A transfer time that is exponentially distributed with mean 10 milliseconds.

We ignore the possible effects of the cache and assume that the requests follow a Poisson process. We desire to find:

- The stability condition, i.e., the maximum amount of requests per second that the hard drive is able to serve.
- The expected response time for a request when the disk is working at 70% of utilisation.

Solution. Let us first determine the stability condition. The expected time to serve a request is the sum of the expected values of the three delays:

$$\mu^{-1} = \frac{6}{2} + \frac{16}{2} + 10 = 21ms\,.$$

Thus, the arrival rate must satisfy:

$$\lambda < \mu \quad \Longrightarrow \quad \lambda < 47.6\ req/s$$

Now, we need to compute the variance of the sum of the three random delays. This can be done in many ways, for instance one may compute the convolution of the three distributions. However, we recall that the convolution of three distributions is nothing more that the product of their Laplace transforms. Recall that the Laplace transform of the p.d.f. of the uniform distribution in $[a, b]$ is:

$$u(s) = \frac{e^{-sa} - e^{-sb}}{s(b-a)}\,,$$

while that of the exponential distribution with rate μ is:

$$e(s) = \frac{\mu}{\mu + s}\,.$$

The Laplace transform of the sum of independent random variables is the product of their Laplace transforms:

$$\hat{f}(s) = \frac{1 - e^{-6s}}{6s} \frac{1 - e^{-16s}}{16s} \frac{0.1}{s + 0.1}\,.$$

The variance can be computed as:

$$\sigma^2 = \left.\frac{\partial^2(\hat{f}(s))}{\partial s^2}\right|_{s=0} - \frac{1}{\mu^2} \simeq 124\ ms^2\,.$$

The computations can be easily performed with a symbolic computation tool. Now, we can apply the P-K formula with $\lambda = 0.7*47.6 = 33.2$ requests per second or, equivalently, 0.0332 requests per millisecond. Using the P-K formula (2.9) we obtain $E[R] \simeq 45$ milliseconds.

2.7 M/G/1/PS queue

In this section, we consider a M/G/1 queue with processor sharing discipline. Recall that this means that, once a job arrives at the queue, it immediately enters the service room but receives a service which is fairly divided by all the jobs that are in the queue. In order to analyse this queueing system, it is convenient to think of jobs with random sizes distributed according to independent random variables with general and identical distribution, and a server with constant unitary speed. The arrival process is a Poisson process with rate λ and the size of a job has c.d.f. $F(x)$ with mean μ^{-1}.

Suppose that $N(x)$ is the expected number of jobs in the system that have received at most x amount of service and let $T(x)$ be the expected time spent in the system by a job of size x. Now, we can imagine a special room of the system that contains all the jobs whose service exceeds x but is smaller than $x + \Delta x$ for some small Δx. The arrival rate at this room is λ multiplied by the fraction of jobs whose sizes exceed x, i.e., $\lambda(1 - F(x))$. If we imagine that Δx is small enough, there will not be any departure inside the box. Thus, we can apply Little's theorem for the fictitious room:

$$\underbrace{N(x+\Delta x) - N(x)}_{\text{Expected number of jobs in the room}} = \underbrace{(T(x+\Delta x) - T(x))}_{\text{Expected time spend in the room}} \cdot \lambda(1 - F(x))$$

Let us divide both hand sides by Δx and take the limit $\Delta x \to 0$. We recognise that

$$\lim_{\Delta x \to 0} \frac{N(x+\Delta x) - N(x)}{\Delta x} = \frac{\partial N(x)}{\partial x}\,.$$

Henceforth, we denote $\partial N(x)/\partial x$ by $n(x)$. A similar observation can be done for $T(x)$ obtaining:

$$n(x) = \lambda(1 - F(x))\frac{\partial T(x)}{\partial x}\,. \tag{2.10}$$

Now, we can derive another expression for $n(x)$. Recall that in the PS system all jobs receive the same intensity of work. Thus if we fix a certain attained service y, the fraction of jobs that need to receive y or more service is $Pr\{X > y\}$, where X is the r.v. modelling the job size. Thus, we must conclude that the amount of jobs in the queue that received a service lower or equal than x must be:

$$N(x) = K \int_0^x Pr\{X > y\} dy\,,$$

where X is the random variable that models a job size. To understand better this formula, assume that you increase x of a small quantity Δx. Clearly, the

amount of jobs that have received at most $x + \Delta x$ service is bigger than that for the jobs that have received at most x. How much bigger? Well, some jobs have left the system after x amount of service, so only those bigger than x can contribute to reach $x + \Delta x$ service.

Taking the derivative on both hand-sides, we have:

$$n(x) = KPr\{X > x\} = K(1 - F(x)) .$$

By equating this expression of $n(x)$ and that of Equation (2.10), we have:

$$\frac{\partial T(x)}{\partial x} = \frac{K}{\lambda} ,$$

and hence, integrating both hand-sides:

$$T(x) = \frac{K}{\lambda} x + c . \tag{2.11}$$

To determine constant c, we can observe that in PS queue (but not in a FCFS!) $T(0) = 0$, that immediately allows us to conclude that $c = 0$. We still need to derive the expression for the constant K. In order to derive this, let us assume that a very large job x ($x \to \infty$) enters the queue. This job has an infinite response time, however it will surely depart after all the other jobs that arrive during his sojourn have left. Therefore, we can derive its residence time by slightly changing the discipline and letting the queue work faster for all the small jobs (and it will finish them quicker) and when the processor is free, it will devote all its service power to the huge job. Since the processor is busy for a fraction $\rho = \lambda/\mu$ of the time, it can work on the huge job for $1 - \rho$. Then we have that:

$$\lim_{x\to\infty} T(x) = \lim_{x\to\infty} \frac{x}{1-\rho} .$$

Let us compare this latter relation with that of Equation (2.11). Clearly, we must have:

$$\frac{\lim_{x\to\infty} T(x)}{\lim_{x\to\infty} x/(1-\rho)} = \lim_{x\to\infty} \frac{T(x)}{x/(1-\rho)} = 1 .$$

Using Equation (2.11):

$$\lim_{x\to\infty} \frac{K/\lambda x}{x/(1-\rho)} = \frac{K(1-\rho)}{\lambda} = 1 ,$$

i.e., $K = \lambda/(1 - \rho)$, and, more importantly, that:

$$T(x) = \frac{x}{1-\rho} . \tag{2.12}$$

This is an important relation for the PS queueing system that gives the expected response time conditioned to the job size. In order to get the unconditioned expected response time, we apply the law of total probability:

$$E[R] = \int_0^\infty T(x) f(x) dx = \frac{1}{1-\rho} \int_0^\infty x f(x) dx = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu - \lambda} .$$

This is a surprising result: the expected response time of a M/G/1/PS queue is the same of a M/M/1 queue and does not depend on the second (or further) moments of the service time distribution. This fact is known as the *insensitivity property* or the M/G/1/PS queue, i.e., the expected performance indices depend only on the expectation of the service time and not on its distribution.

2.7 Example
Let us consider a queueing system with a uniform service time distribution in the interval $[a, b]$, with $a \leq b$. Give the conditions on a, b under which the FCFS discipline has an expected response time lower than the PS discipline.

Solution. Recall that for the uniform distribution, we have:

$$\mu^{-1} = \frac{a+b}{2}, \quad \sigma^2 = \frac{1}{12}(b-a)^2 .$$

The solution of the problem is based on the following reasoning. We know that the expected response time in strictly monotonically increasing with respect to the variance of the service time distribution, as shown by Equation (2.9). Moreover, we know that once $\sigma^2 = 1/\mu^2$, the M/G/1 queue has the same expected response time of the M/M/1 queue (see Example 2.4). Finally, we notice that for any service time distribution, the M/G/1/PS queue has the same expected response time of the M/M/1 queue. Thus, if the distribution of the service time has a variance which is lower than that of the exponential distribution, then FCFS is more convenient in terms of the minimisation of the expected response time.

In conclusion, we need to solve the following inequality:

$$\frac{1}{12}(b-a)^2 < \left(\frac{a+b}{2}\right)^2 ,$$

with the condition $0 \leq a \leq b$. The solution shows that this is true for all the valid values of a and b, i.e., for the uniform distribution it is always more convenient (in terms of minimisation of the expected response time) to adopt a FCFS scheduling over the PS.

2.8 Exercises with solutions

Exercise 1 We model a web server by a M/M/1 queue with service rate $\mu = 8$ jobs/second. Let the arrival rate be λ:

1. Give the stability condition for the system
2. Compute the expected number of customers in the system in stability given $\lambda = 4$ jobs/second
3. Compute the expected response time in stability given $\lambda = 4$ jobs/second

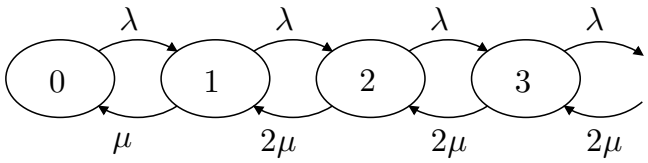


Figure 2.5: CTMC underlying a M/M/2 queueing system.

Solution. The stability condition for this type of queue is $\lambda < \mu$, i.e., $\lambda < 8$ jobs per second. If $\lambda = 4$, the expected number of customer $\overline{N}$ is given by:

$$\overline{N} = \frac{\lambda/\mu}{1 - \lambda/\mu} = 1 \text{ job}\,.$$

The expected response time can be easily obtained as:

$$\overline{R} = \frac{1}{\mu - \lambda} = \frac{1}{8 - 4} = 0.25 \text{ s}\,.$$

2.9 M/M/m queue and the Erlang formula

In this section, we consider the case of a queueing system with FCFS discipline and multiple identical servers with exponential service time. In practice, this system may be useful to model multi-core systems but also call centers, telephone switches and so on. In these systems, each server can work at most on a job. As a consequence, even if we have m servers, when the number of jobs n in the system is lower than m, we are unable to exploit the whole service power because some servers will remain idle. Notice that since both the inter-arrival and the service times are independent and exponentially distributed, the stochastic process underlying this queue is a CTMC. We first derive the stationary distribution for the M/M/2 queueing system and then we give the general formula.

2.9.1 M/M/2

The Markov chain underlying a M/M/2 system is shown in Figure 2.5. Since we have a generalised birth&death process, we may easily derive the expression for the stationary distribution. Fix state 0 as a reference state, then we have:

$$\pi(n) = \pi(0) \left(\frac{\lambda}{\mu}\right)^n \frac{1}{2^{n-1}}\,, \quad n > 0.$$

Let $\rho = \lambda/\mu$. We may derive the $\pi(0)$ by normalising the probabilities, i.e., $\pi(0)$ must satisfy:

$$\sum_{n=0}^{\infty} \pi(n) = \pi(0) + \sum_{n=1}^{\infty} \pi(0)\rho^n \frac{1}{2^{n-1}} = \pi(0) \left(1 + \frac{2\rho}{2 - \rho}\right)\,,$$

where the last step follows by noting that the series is geometric and converges if $\lambda < 2\mu$. This stability condition corresponds to the intuition that the intensity of the arrival process must be strictly lower than the total computational power. We can derive $\pi(0)$ as:

$$\pi(0) = \frac{2-\rho}{2+\rho}$$

Thus, we can obtain the expected number of jobs in the system:

$$\overline{N} = \sum_{n=1}^{\infty} n\pi(n) = \sum_{n=1}^{\infty} n \left(\frac{2-\rho}{2+\rho}\right) \rho^n \frac{1}{2^{n-1}} = \frac{4\rho}{4-\rho^2}\,.$$

At this point, we can use Little's law to obtain the expected response time:

$$\overline{R} = \frac{\overline{N}}{\lambda} = \frac{4\mu}{4\mu^2 - \lambda^2} = \frac{1}{2\mu - \lambda}\frac{4\mu}{2\mu + \lambda}\,.$$

It may be instructive to compare this result with the response time of a M/M/1 queue with service rate 2μ, i.e.:

$$\overline{R}' = \frac{1}{2\mu - \lambda}$$

Since $\lambda < 2\mu$ for the stability condition, we know that fraction $4\mu/(2\mu + \lambda)$ is strictly larger than 1, this the expected response time of the $M/M/2$ queue is strictly larger than that of a M/M/1 queue with a double speed server. Notice that when $\lambda \to 0^+$, i.e., we are in low load condition, $4\mu/(2\mu + \lambda) \to 2$, i.e., the M/M/2 queue has an expected response time which is double of the M/M/1. However, when $\lambda \to 2\mu$, i.e., the maximum arrival rate, we have $4\mu/(2\mu+\lambda) \to 1$, and hence in heavy load the expected response time of the M/M/2 queue tends to that of the M/M/1.

2.9.2 General formula for the M/M/m queueing system

In this queueing system, we model the availability of multiple servers with a load-dependent service rate, thus we have:

$$\mu(n) = \begin{cases} n\mu & \text{if } n \leq m \\ m\mu & \text{otherwise} \end{cases}$$

where $\mu(n)$ denotes the system service rate when there are n jobs. The steady-state distribution is:

$$\pi(n) = \begin{cases} \pi(0)\frac{1}{n!}\rho^n & \text{if } n \leq m \\ \pi(0)\frac{1}{m!}\rho^n \frac{1}{m^{n-m}} & \text{if } n > m \end{cases}$$

with $\rho = \lambda/\mu$, where $\pi(0)$ can be determined as:

$$\pi(0) = \left(\sum_{j=0}^{m-1} \rho^j \frac{1}{j!} + \rho^m \frac{1}{m!} \frac{1}{1 - \frac{\lambda}{m\mu}} \right)^{-1}\,.$$

In the analysis of the M/M/m queueing system, the Erlang-C formula plays a pivotal role. It gives the probability of finding all the m servers busy, i.e.:

$$C(m,\rho) = \sum_{j=m}^{\infty} \pi(j)\,.$$

This can be computed by avoiding the infinite sum as follows:

$$C(m,\rho) = \left[1 + \left(1 - \frac{\rho}{m}\right)\left(\frac{m!}{\rho^m}\right)\sum_{k=0}^{m-1}\frac{\rho^k}{k!}\right]^{-1}. \tag{2.13}$$

As sanity check, one may verify that $C(1,\rho) = \rho$ as expected from the analysis of the M/M/1 queueing system. In conclusion, we have that the expected number of jobs in the system is given by:

$$\overline{N} = \frac{\rho/m}{1-\rho/m}C(m,\rho) + \rho\,,$$

and, by Little's law, we have:

$$\overline{R} = \frac{C(m,\rho)}{m\mu - \lambda} + \frac{1}{\mu}\,.$$

2.10 G/G/1 queueing system

In this queueing system, both the arrival process and the service times are general and independent. Unfortunately, most of the techniques that we have seen so far do not apply to the G/G/1 system since the queue is not Markovian (i.e., its underlying stochastic process is not a Markov chain) and we cannot use the PASTA result.

G/G/1 queue can be studied thanks to the Lidley's equation. However, we propose a simple approximation of the mean waiting time that allows us to approximate also the other mean performance indices thanks to Little's law.

The approximation that we present is called Kingman's formula:

$$\overline{W} = \left(\frac{\rho}{1-\rho}\right)\left(\frac{c_a^2 + c_s^2}{2}\right)\frac{1}{\mu}\,, \tag{2.14}$$

where, as usual, $\rho = \lambda/\mu$ and λ, μ are the arrival and service rates, respectively. c_a and c_s are the coefficient of variations for inter-arrival and service times, respectively. Kingsman's approximation is accurate in heavy-load but it is acceptable also for moderate load analysis.

Notice that, if the inter-arrival times are exponentially distributed $c_a = 1$ and Kingman's formula gives the waiting time of the M/G/1 (to see this, recall that $c_s = \mu\sigma$).

What does Kingman's formula tell us? We know from the P-K formula that the higher is the variance of the service time distribution the higher is

the expected waiting time for the M/G/1 queue. Now we discover that both the variance of the inter-arrival process and that of the service time negatively affect the waiting time of the queue. Indeed, consider the particular case of a D/D/1 queue in stability. What is the waiting time? Clearly, 0 (and this is what Kingman's formula actually predict since $c_a = c_v = 0$). In contrast, as the coefficients of of the inter-arrival and service times increase, the expected waiting time also increases.

Exercises

Exercise 1 Derive the expression for the mean waiting time for the M/G/1 queue (2.9) from that of Equation (2.8).

Exercise 2 Consider again Example (2.2). Prove that the optimal value of p under Condition (2.7) is unique for any value of $\lambda < \lambda_{\max}$.

Exercise 3 Generalise the result of Example 2.3 to find the probability that a customer arriving at the queue in stability finds more than n jobs.

Exercise 4 Can you give an example of a queueing system with $\lambda < \mu$ but for which Limit (2.4) does not exist? *Hint: think to the periodicity...*

Chapter 3

Queueing networks

When we model a complex system, it may be useful to combine different servers each of which is equipped with its queue and discipline. Let us make an example. Consider a web server with a CPU which is assigned according to a PS discipline and a disk which serves its jobs according to a FCFS discipline. Suppose that we wish to model a system in which, once a new customer arrives, it requires an average of μ_{CPU}^{-1} seconds to the CPU and μ_{DISK}^{-1} seconds to the disk. The disk is visited twice (for instance once for the database access and once for retrieving some multimedia files), while the CPU once. The distribution of the service time at the CPU and at the disk can be arbitrary. Actually, we will see that we can also distinguish the service time distribution at the disk depending on the fact that the customer is at its first or second visit. We depict the model of this system in Figure 3.1. In this setting, customers arrive from the outside, receive the service after the visit to the CPU and disk stations, and finally, they leave the system. It is clear that, similarly to what happens for single queueing systems, the intensity of the workload is given by the intensity of the arrival process, i.e., the expected number of jobs that arrive at the system per unit of time. If the system is stable, we can also observe that the intensity of the departure process, i.e., the throughput of the system, must be equal to the arrival rate.

When we consider queueing networks, we may have a different type of sys-

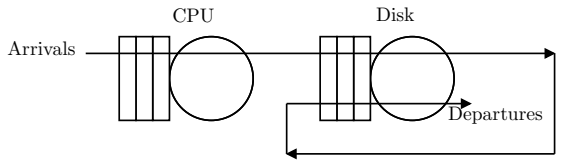


Figure 3.1: Example of open queueing network consisting of two resources: CPU and Disk. Each request makes one visit at the CPU station and two at the disks.

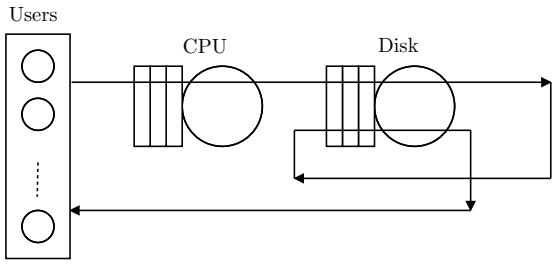


Figure 3.2: Example of closed queueing network consisting of two resources: CPU and Disk. The left most rectangle denotes the thinking time phase of the jobs.

tems in which the intensity of the workload is set with the *level of multiprogramming*. Let us imagine that the web server described above has a certain number of customers that interact with it. Each customer sends a request for a page, then this is processed by the CPU and the disk, and the answer is sent back. At this point the customer consults the answer for a certain amount of time (*thinking time*) and then sends another request to the web server. Clearly, while the requests compete for the use of the resources (namely, the CPU and the disk), the thinking times are performed in parallel by the customers. This is an example for closed queueing network where the workload is given by the number of customers that interact with the system. The model is depicted in Figure 3.2.

In this chapter, we classify the queueing networks and introduce some operational analysis rules. This approach to the analysis of queueing networks provides bounds on their performance indices rather than the exact results, but it is very important because it does not require any assumption on the distributions of the inter-arrival times or service times aside from the knowledge of their first moment.

3.1 Characterisation of queueing networks

We already saw by an example that queueing networks can be classified in two types: open and closed. In this section, we review the characteristics of these systems and their performance indices.

3.1.1 Open networks

Open networks are characterised by the fact that there are one or more arrival streams of jobs from the outside and one or more departure streams. The queueing network of Figure 3.1 is an example of open queueing network. Clearly, the queueing systems of Chapter 2 are special cases of open networks with a single station. Each station in the queueing network can have arrivals coming

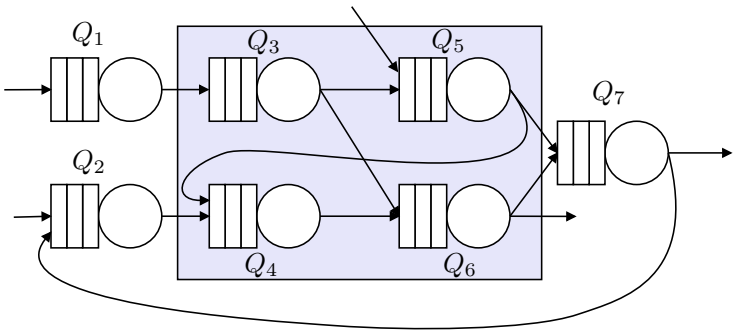


Figure 3.3: Example of open queueing network.

from the outside or from other stations. We stress from now that, requiring that the arrivals from the outside occur according to a certain process, e.g., a Poisson process, does not mean that each station of the queueing network will see that type of arrival process. Indeed, feedbacks and the dependency introduced by the routing can drastically change the arrival processes seen by queues in the network. In the next chapter, we will see some cases in which an open queueing network under Poisson arrival process has stations whose arrival processes (in steady-state) are Poisson arrival processes. This result is called Burke's theorem and imposes strong assumptions on the network.

In general, when we talk about open queueing networks, we implicitly assume that the capacity of each queue is infinite. Handling finite capacity queues may be hard in queueing networks, and some results are presented in [1].

Performance indices

In order to introduce the performance indices for an open queueing network, we use the running example shown in Figure 3.3. It consists of 7 queueing stations interconnected by a routing scheme. While we discuss the most common types of routing in Section 3.1.4, here it is sufficient to say that if a station has two or more outgoing arcs, it means that the customers, once they leave that station, can choose (probabilistically or deterministically) one among the target stations or the outside as next phase. For example, once a customers finishes its work at station Q_6, it may either continue to Q_7 *or* leave the system, while once it leaves station Q_3 it may proceed either to Q_5 or to Q_6. In this book, we are interested in the steady-state behaviour of queueing networks, hence the first problem that we address is that of defining when we say that a queueing network is unstable. For general queueing networks with priority scheduling the decision on the stability of a queueing network can be hard, but these examples fall outside the objectives of this work. Definition 3.1 says that an open queueing network is unstable if for some initial state the system tends to accumulate more and more jobs as time passes.

Definition 3.1 (Instability of an open queueing network) An open queueing network is unstable if, for some initial state, the number of jobs in the network will, with positive probability, go to infinity as time $t \to \infty$.

For a stable open queueing network, all the stations are also stable. Let $N(t)$ be the total number of jobs in the queueing network at time t, then we can define the stationary expected number of jobs in the system as:

$$\overline{N} = \lim_{t \to \infty} \frac{N(t)}{t} ,$$

whenever this limit exists. This definition can be extended to consider a particular station and we will use $\overline{N}_1, \ldots, \overline{N}_7$ or a subset of the network. For example, we may be interested in the expected number of jobs in the grey sub-network of Figure 3.3. We write:

$$\overline{N}_{3,4,5,6} = \overline{N}_3 + \overline{N}_4 + \overline{N}_5 + \overline{N}_6 .$$

The throughput of a certain station will be denoted by X_i and corresponds to the expected number of job completions per unit of time. The utilisation is the fraction of time a single server is busy (for the multi server cases, other definitions are applicable). The definitions of these performance indices are analogue to those proposed in Chapter 2. Notice that, while the utilisation of a sub-network seems difficult to interpret, we can easily consider the throughput of a sub-network. For example, consider the grey sub-network shown in Figure 3.3, the throughput of that sub-network is the sum of the throughput seen on the connections between Q_5 and Q_7, between Q_6 and Q_7 and from Q_6 to the outside. Notice that internal flows, e.g., that from Q_5 to Q_4 are not taken into account.

At this point, the reader will not be surprised in discovering that in a stable open queueing network, for any sub-network, the total flow incoming must be equal its throughput. The is no exception if the sub-network is the whole queueing network! Thus, an important rule of open networks is that, in stability, the throughput is equal to the intensity of the arrival process and is independent of the service rates of the stations. Clearly, the server rates have an important impact on the response times, waiting times and network population.

3.1.2 Closed networks

Closed queueing networks are characterised by a fixed number of customers that move among the system's stations. There is not any arrival stream or departure. Therefore, it is interesting to understand how we can define the system's throughput. To this aim, we introduce two further classifications of closed networks:

- Interactive systems or terminal-driven systems
- Batch systems

In **interactive systems**, each customer alternates a *thinking state* and a *submitted state*. Thus we can see the system consisting of two main components: the terminals where the customers in the thinking state reside, and the central subsystem where the computations take place. During the thinking state, whose duration Z is called *thinking time*, the customer consumes the output of the central subsystem and prepares the next request. These operations occur in parallel for all the jobs in the thinking state. In contrast, during the submitted state, the customer waits for an answer from the central subsystem. All the customers in the submitted state compete for the resources of the central subsystem. The duration of the submitted state R is the response time of the system. The number of customers that are in the system is called *level of multiprogramming* of the system. Figure 3.2 shows an example of an interactive system where the CPU and the disk queueing stations form the central subsystem. We finally introduce the system time, which is the sum of the response time R and the thinking time Z. Using the averages:

$$\overline{T} = \overline{R} + \overline{Z}\,.$$

When we desire to emphasise that the performance indices depend on the network population, we write $\overline{T}(N)$ or $\overline{R}(N)$, while $\overline{Z}$ is independent of N.

The throughput of the system is given by the number of job completions performed by the central subsystem. Suppose that we have N statistically identical customers and that customers have an expected system time $\overline{T}(N)$. If $N = 1$ it is easy to see that the system throughput $X(1) = \overline{T}^{-1}(1)$, since $\overline{T}(1)$ is the expected time required by a customer to make a whole cycle in the system. But since we have N customers, we have:

$$X(N) = \frac{N}{\overline{T}(N)}\,. \tag{3.1}$$

These relations are very important and will be widely used in the following sections. We summarise them in the following theorem.

Theorem 3.1 For an interactive system, it holds that:

$$\overline{R}(N) = \frac{N}{X(N)} - \overline{Z}\,.$$

Here, we have an important observation. Intuitively, R depends on N, i.e., larger values of N imply larger response times R because there is more competition for the central subsystem resources. On the other hand, Z does not depend on N because the thinking times occur in parallel. Anyway, T depends on N and, clearly, on the choice of the service rates in the central subsystem. Therefore, Equation (3.1) suggests two observations:

- In contrast with open networks, in closed queueing networks the throughput depends both on the service rates and on the system population.

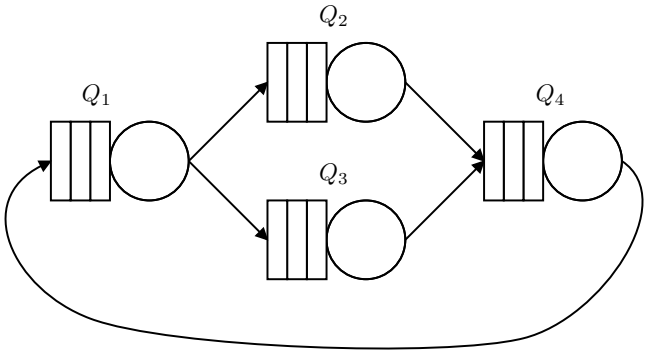


Figure 3.4: Example of closed queueing network modelling a batch system.

- Equation (3.1) may wrongly suggest that when $N \to \infty$ we have $X \to \infty$. This would be really counter-intuitive! Indeed, we have to take into account the fact that T depends on N and, in general, also $T \to \infty$ when $N \to \infty$. Thus, the computation of the throughput X requires some attention when we desire to consider the asymptotic case $N \to \infty$ since Equation (3.1) brings to an undetermined form. We will study these cases in the following sections.

Some of the observations that we have done for open queueing networks, regarding the performance measures of single stations and of sub-networks, can be straightforwardly reformulated for closed systems. Regarding the stability, since the population in closed networks is fixed, then we have that the network cannot be unstable.

In **batch systems**, we have a constant number of jobs that are continuously being processed. As soon as a job is completed, a new one is begun. Informally, it can be seen as an interactive system where the thinking time Z is deterministically set to 0. Figure 3.4 shows an example of a queueing network modelling a batch system. In this type of systems, we usually choose one reference station, the throughput of that station will be considered as the system throughput. Ideally, this station should see all the customers that have completed their work or all the customers which are entering the system. In the network of Figure 3.4, stations Q_4 and Q_1 are candidate for this aim. Notice that the relations derived for the interactive systems are still valid for the batch systems with the assumption $Z = 0$.

3.1.3 Classification of the customers

In the simplest queueing networks, all the customers are statistically identical. This means that when a customer arrives at a queueing station, its service time has the same distribution of all the other customers. In general, we assume that these distributions are independent of each others. We call this type of queueing

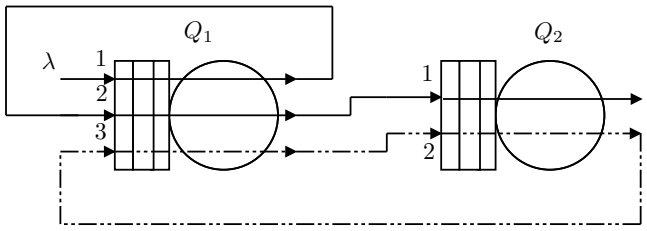


Figure 3.5: Example of mixed queueing network with multiple classes and multiple chains of customers.

networks *single-class queueing networks*.

However, in more sophisticated models, we can distinguish the customers and change the way they behave (the routing or the service time). We have two ways to classify the customers: the class and the chain. In the literature, several definitions of classes and chains have been proposed, and, at the moment, there is not a uniform accepted nomenclature. In general, a class is a temporary classification of a customer while a chain is a permanent classification. For example, in an interactive system we may have N_1 jobs that represent the web server users and N_2 that are the administrators. Clearly, the type of operations performed by administrative and ordinary users differ by typology and resource requirements. Therefore, we say that the queueing network has two chains: one for the ordinary users and one for the administrators. In contrast, classes may model a certain phase of the computation of a customer and hence can change with time. For example, consider the case of Figure 3.1 where each job visits the disk station exactly twice. We can model this behaviour by imagining that the first time the job arrives at the disk station, it has a certain class c but, as soon as it leaves, it performs a *class switching* to d and re-enter the same station. Since the station can distinguish customers with class c and d, we can have different class-dependent service time distributions and we are sure that once class d jobs leave the station, they will not enter the same station again but will leave the system. In this textbook, we interpret the classes with a *local meaning*, i.e., associated with the queueing station. In other words, the class of a customer is a pair formed by a label $i \in \mathbb{N}$ and the name of the station.

Let us consider the example of Figure 3.5 that shows a queueing network with 5 classes of customers and 2 chains. The classes are $(1, Q_1)$, $(2, Q_1)$, $(3, Q_1)$, $(1, Q_2)$ and $(2, Q_2)$ read as *class i of station Q_j*. Therefore, each station can specify a different service time distribution for each of these classes. When a queueing network has multiple classes at the stations, we say that it is a *multi-class queueing network*. The network consists of two chains, one open, depicted with solid line, and one closed, depicted in dashed line. Clearly, the network is neither open nor closed and for this reason we say that it is mixed. Indeed, when we have multi-chain models, it is better to use open and closed as properties of the chains rather than of the network itself. Thus, for multi-class networks we

can have either single chain models or multi-chain models. Observe that, since class-switching is allowed but no chain switching is possible, the chains cluster the classes into disjoint sets, i.e., they for a partition of the customer classes.

3.1.4 Routing

Routing in queueing networks plays a pivotal role. It describes the way the jobs or customers move among the stations of the network. Routing policies can be arbitrary complicated and are objective of research. For example, cloud systems can consist of thousands of computers interconnected. The routing policy decides which computer will process a request coming from the outside or which computer will process a request after the end of processing at another station. The decision may be taken based on the information that the dispatcher has (e.g., does it know the current utilisation of all the servers?), the constraints of the network (can any computer perform all the operations or are there specialisations?) and the goal that the designer wants to reach (do we wish to minimise the response time, the energy consumption or a mix of the two indices?).

In queueing networks, if the routing depends on the state of the systems, we say that it is *state dependent*, otherwise, we say that it is *state independent*. State dependent routing policies are usually quite difficult to study, and are outside the scope of this book.

A type of routing which is widely used in queueing networks is the *probabilistic routing*. We first describe the probabilistic routing for single-class queueing networks. Whenever a customer leaves a certain station, it chooses among all the possible destinations (including the outside) in a probabilistic way. This probability does not change with time or the state of the network, and is independent of all the other processes of the model. If the network consists of the station $Q_1, Q_2, \ldots, Q_K$, then we can specify the probabilistic routing with a routing matrix $\mathbf{P}$ whose dimension is $K \times K$ and the element p_{ij} denotes the probability that, after the service at station i a customer will go to station j. $\mathbf{P}$ is stochastic (i.e., the sum of the elements of each row is equal to 1) if the network is closed, and is sub-stochastic (i.e., the sum of the elements of each row is lower equal to 1) if it is open. In fact, in the latter case, we may have the probability that a job leaves a station after its service and this is not represented in the routing matrix but can be derived by difference. For example, the probability that a customer leaves the system after the service at station i is $1 - \sum_{j=1}^{K} p_{ij}$.

What changes in multi-class and multi-chain queueing networks? First of all, the probabilistic routing occur not between stations but between classes. Recall that, since we used the notion of classes in a local way, this also includes the station. Hence the probabilistic routing specifies the probability for a customer leaving a certain station Q_j as class (c, Q_j) to enter station Q_i as class (d, Q_i). We have a routing matrix for each chain, and these matrices are either stochastic (for closed chains) or sub-stochastic (for open chains).

3.1 Example
Consider again the queueing network of Figure 3.5. Let us order the classes by using first the number of the queue and then the number of the class. The routing matrix for the chain depicted in solid line (let us call it A), is:

$$\mathbf{P}^A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} .$$

For the second chain (let us call it B), we have:

$$\mathbf{P}^B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

As we can see, although we call this type of routing with the adjective *probabilistic*, the (independent) deterministic routing is just a special case.

3.2 Example
Let us consider a second example consisting of a model for an interactive system with an application server and two databases: a master and a slave. 60% of the requests are sent to the master and include all the *write requests* and some *read* requests. The remaining read requests are sent to the slave. Let us assume that the write requests are the 12% of the total. The write requests, after being processed by the master database but be processed also by the slave database in order to maintain the coherency. The queueing network model is shown in Figure 3.6 in which Q_1 represents the thinking states of the jobs, Q_2 the application servers and Q_3, Q_4 the primary and secondary databases, respectively. The central subsystems consists of Q_2, Q_3 and Q_4. According to the description of the system, we have $p_{23} = 0.6$ and $p_{24} = 0.4$. We can determine p_{34} as follows: Q_3 receives 0.6 of the total requests, containing 0.12 of writes, i.e., we have 0.48 of the total reads. Once a request leaves Q_3, what is the probability that it is a write request? Clearly we have $p_{24} = 0.12/0.6 = 0.2$ while $p_{31} = 0.8$. The routing matrix is stochastic:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 \\ 0.8 & 0 & 0 & 0.2 \\ 1 & 0 & 0 & 0 \end{pmatrix} .$$

In a single-class closed queueing network, we say that the routing is *irreducible* if each station can be reached by any other station after one or more job completions. For single-class open queueing networks, the irreducibility is more complicated. We introduce a virtual station representing the outside and

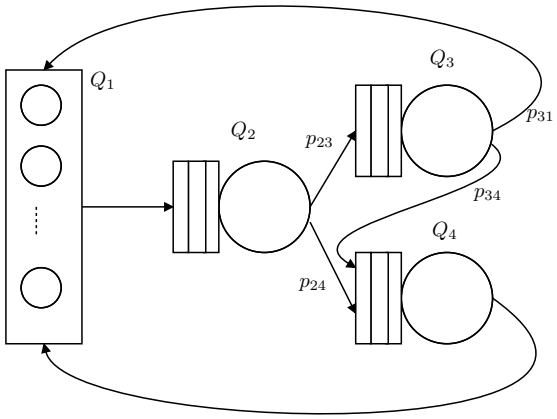


Figure 3.6: Interactive system with an application server and two databases.

all the flows outgoing the network becoming incoming flows to this station and vice versa. Then, we say that the routing is irreducible if the closed queueing network obtained in this way has irreducible routing.

3.1.5 Finite capacity queueing networks

Simonetta: lasciamo questo paragrafo? vuoi mettere tu qualcosa?

3.2 Operational analysis and asymptotic

In this section, we study some general laws for queueing networks. These laws are very powerful because they do not depend on the distribution of the service times at the stations but they are still capable of providing bounds on important performance indices. Particularly, we focus on single-class networks, although the literature addresses the operational laws also for multi-class and multi-chain models.

3.2.1 Little's law and theorem for queueing networks

Let us consider an open queueing network and let us assume that the total intensity of the arrivals at the network is λ. We also assume that the system is stable, so we have a stationary throughput $X = \lambda$. This is a system with an input stream and an output stream as those studied in Section 2.2, thus Little's law and theorem apply. We can include in this class also sub-networks. For example, consider again the grey subnetwork shown in Figure 3.3: if we know the expected number of customers in the whole subnetwork, i.e.:

$$\overline{N}_{3,4,5,6} = \overline{N}_3 + \overline{N}_4 + \overline{N}_5 + \overline{N}_6 \,,$$

and the throughput of that sub-network, then we can compute the expected response time limited to that portion of the queueing network thanks to Little's theorem as:

$$\overline{R}_{3,4,5,6} = \frac{\overline{N}_{3,4,5,6}}{X_{3,4,5,6}} .$$

The same reasoning holds for sub-networks of closed queueing networks. For example, consider the queueing network depicted in Figure 3.2. If we consider just the central subsystem consisting of the CPU and Disk station, then we can related the expected response time $\overline{R}$, the system throughput X and the expected number of jobs in the submitted state $\overline{N}_s$ as:

$$\overline{N} = X\overline{R} .$$

What happens if we consider the whole queueing network of Figure 3.2? This is the model of an interactive system, hence we know that the throughput of the central subsystem is what we consider to be the total throughput of the network. However, now the total number of customers in the queueing network is fixed, i.e., N is not a random variable but a constant. Therefore, we can obtain the total expected system time (i.e., the sum of the expected response time and that of the expected thinking time) as:

$$\overline{T} = \frac{N}{X} .$$

Observe that this is nothing more than Equation (3.1) that we derived before.

The flexibility of Little's results now appears clearly. They can be used to single queues (embedded in a network or isolated), to sub-networks of closed and open queueing networks and to open or closed networks.

3.3 Example
Let us consider the model of Figure 3.2 with $N = 12$ jobs. Suppose that thanks to some measurements at the terminals we know that an average of 3 jobs/s enter the submitted state. Moreover we know that the expected duration of the thinking time is 1s. We want to computed the expected response time of the system.

First of all, we can compute the expected system time as $\overline{T} = \overline{N}/X = 12/3 = 4$s. In fact, if the arrival stream at the central subsystem is 3 jobs/s, its throughput must be the same. The expected response time can be derived as

$$\overline{R} = \overline{T} - \overline{Z} = 4 - 1 = 3\text{s} .$$

3.4 Example (Derivation of the utilisation law)
The utilisation law states that, in a single server queueing system in stability, when the arrival rate is λ and the service rate is μ, the utilisation is $\rho = \lambda/\mu$, regardless to the characteristics of the arrival or service pro-

cesses. We prove this result by applying Little's theorem. Consider the system formed by two elements: the waiting room and the service room. Let us focus on the service room. The service room can contain either 1 or 0 jobs, i.e., the expected number of jobs in the waiting room corresponds to the probability of finding the server busy. So, the utilisation ρ has this additional interpretation. Since the queueing system is stable, the throughput of the waiting room must be equal to the arrival rate λ. What is the expected response time of a job entering the service room? Since, there is no queue in the service room, the expected response time is μ^{-1}. Thus we can apply Little's theorem and obtain:

$$\rho = \lambda \cdot \left(\frac{1}{\mu}\right) = \frac{\lambda}{\mu}\,,$$

as required.

3.2.2 The forced flow law

In this section, we learn an important law of queueing networks that relates the throughput of the stations. This is known as *forced flow law* and will be very useful. We begin by introducing an important quantity in queueing networks, i.e., the *relative visit ratios*. Let us consider a queueing network (open or closed) consisting of K stations, and let, without loss of generality, station Q_1 be the reference station. The reference station can be chosen arbitrary and the results of the operational analysis do not change according to that choice. However, for interactive systems, a common choice is the terminal stations. Let $\overline{V}_i$ be the expected number of visits at station Q_i for each visit at the reference station that a customer does, for $i = 1, \ldots, K$. Clearly, $\overline{V}_1 = 1$ by definition. Then, we have:

Theorem 3.2 (Forced flow law) For each queueing station of a network with K stations and station 1 as reference station, we have:

$$X_i = \overline{V}_i X_1\,.$$

The law of forced flow states that if we know $\overline{V}_i$ and the throughput of the reference station, then we can derive the throughput of all the other stations. Similarly to Little's law, also the forced flow law is a theorem and it may be proved. However, we provide an intuitive argument that supports Theorem 3.2 rather than a formal proof. If the throughput at station Q_1 is X_1, and, on average, for each job completion at Q_1 we observe $\overline{V}_i$ completions of the same job at Q_i, then it must be that the throughput of Q_i is exactly $\overline{V}_i$ times the throughput of Q_1.

Computation of the throughput in open queueing networks

Let us consider an open queueing network with K stations and probabilistic routing defined my the routing matrix $\mathbf{P}$. Moreover, let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ be the vector of the arrival rates at each station. Let e_i denote the total intensity of the arrivals at station i from the outside or any other station of the network. Since we assume that the network is stable, we have $X_i = e_i$ for every $i = 1, \ldots, K$. Thus the intensity of the arrivals seen by station i is:

$$e_i = \lambda_i + \sum_{k=1}^{K} X_k p_{ki} \,,$$

i.e.:

$$e_i = \lambda_i + \sum_{k=1}^{K} e_k p_{ki} \,. \tag{3.2}$$

The set of equations (3.2) for $i = 1, \ldots, K$ is call *system of traffic equations* and if the routing is irreducible it admits a unique solution which gives the throughput at each station. Recall that, in open queueing networks, the throughput does not depend on the service rates!

Now, we can apply the forced flow law and obtain the relative visit ratios for the reference station:

$$\overline{V}_i = \frac{X_i}{X_1} = \frac{e_i}{e_1} \,.$$

If the open network does not have a probabilistic routing, or this is unknown, we may take the opposite approach. Indeed, we can measure the relative visit ration $\overline{V}_i$ and the intensity of the arrivals. Then, thanks for the forced flow law we can derive the throughput at each station.

Computation of the throughput in closed queueing networks

Let us consider a closed queueing network with probabilistic routing. By following the same reasoning shown for open queueing networks, we have:

$$X_i = \sum_{k=1}^{K} X_k p_{ki} \,. \tag{3.3}$$

However, even if the routing is irreducible, this linear system is undetermined. Indeed, we know that in closed systems the throughput depends on the network population and on the service rates which do not appear in Equation (3.3). Therefore, for closed queueing networks, we proceed as follows. Thanks to the forced flow law, we can write:

$$\overline{V}_i X_1 = \sum_{k=1}^{K} \overline{V}_k X_1 p_{ki} \,.$$

Dividing both hand sides by X_1 (which is strictly positive when the queueing network has at least one customer and the routing is irreducible), leads to:

$$\begin{cases} \overline{V}_i = \sum_{k=1}^{K} \overline{V}_k p_{ki} & \text{for } i = 2, \ldots, K \\ \overline{V}_1 = 1 \end{cases} \tag{3.4}$$

We can see that the traffic equations for closed queueing networks define a relation among the relative visit ratios and not among the throughput of the stations. This difference is crucial for the analysis of these type of systems.

Clearly, given the solution of the system of Equations (3.4) and the throughput of one station, we can derive the throughput of all the other stations of the queueing networks. In general, this is useful in benchmarking experiments of interactive systems where we can measure the throughput of the terminal station and somehow compute the relative visit ratios of the components of the central subsystems. This allows us to derive the throughout at each station.

3.5 Example
We perform a benchmarking experiment of a web applications. The system consists of a benchmarking machine simulating the terminals of an interactive system, an application server and two databases. The system is depicted in Figure 3.6. In order to serve $1,000$ requests in an experiment of 20 minutes and with a thinking time of 20s on average, we measure that:

- The application server (Q_2) has served $1,000$ requests
- The master database (Q_3) has served 720 requests
- The slave database (Q_4) has served 405 requests

We want to compute the throughput, and the expected response time of the central subsystem knowing that the benchmark is simulating the behaviour of 20 customers.

First of all, we can compute the expected relative visit ratios, assuming the terminals (Q_1) as reference station:

$$\overline{V}_1 = 1\,, \quad \overline{V}_2 = 1\,, \quad \overline{V}_3 = 0.72\,, \quad \overline{V}_4 = 0.405\,.$$

The system throughput is $X = X_1 = 1000/(20{*}60) = 0.83$ jobs per second. Now, we can derive the system response time by Little's theorem:

$$\overline{T} = \frac{20}{0.83} = 24.1 \text{ s.}$$

Therefore, the expected response time can be derived as:

$$\overline{R} = \overline{T} - \overline{Z} = 24.1 - 20 = 4.1 \text{ s}$$

The throughput of each station can be derived by the forced flow law:

$$X_2 = X_1\overline{V}_2 = 0.83 \text{ jobs/s}\,, \quad X_3 = X_1\overline{V}_3 = 0.60 \text{ jobs/s}\,,$$

$$X_4 = X_1\overline{V}_4 = 0.336 \text{ jobs/s}\,.$$

3.2.3 The service demand and the bottleneck law

In this section, we introduce an assumption which is not required by the previous results. Indeed, we assume that the service time of a job is independent of the number of visits it performs to a certain station of the networks. For the probabilistic routing case, this condition is trivially satisfied because the routing is independent of all the other stochastic processes by definition. However, the results that we propose here are applicable for a wider class of routing schemes than the probabilistic one.

The service demand is the total amount of service required by a customer to each station for each visit it does at a reference station. Thanks to the independence assumption states before, we can write:

$$\overline{D}_i = \overline{V}_i \frac{1}{\mu}$$

since μ^{-1} is the expected service time for each visit. By the utilisation law and the forced flow law, we can write:

$$\rho_i = \frac{X_i}{\mu} = \frac{X_1\overline{V}_i}{\mu} = X_1\overline{D}_i\,.$$

The last equality is the *bottleneck law* that relates the service demand to the system's throughput and the queues' load factors. Recall that the utilisation law holds only for single server queues with constant speed, and so also the bottleneck law.

3.6 Example
A web server receives on average 4 requests per second. Each request generates an average of 10 disk accesses, each of each has an average service time of 10ms. We want to compute the utilisation of the disk.

We first derive the service demand of the disk $\overline{D} = \overline{V}\mu^{-1} = 10 \cdot 10 \cdot 10^{-3} = 0.1$s. Then, we apply the bottleneck law and obtain:

$$\rho = X\overline{D} = 0.4\,.$$

3.2.4 Asymptotic bounds for closed queueing networks

In this section, we use the operational laws stated before to derive some important bounds on the performance measure of a queueing network.

Let us consider an interactive system, single-class, where the reference station is Q_1 and is also the terminal station. Suppose that we have a level of multiprogramming $N \to \infty$. We further assume that all the stations are single server with constant speed. What happens when $N \to \infty$? The common, but *wrong*, answer is that all the queues tend to grow in population size proportionally to their load factor. Let us try to understand why this is wrong. When N grows, the utilisation of the queues grows but at a certain point there will be a certain station Q_b whose load factor approaches 1 before the others. This station is the *bottleneck* station of the queueing network. If $\rho_b \to 1$ when $N \to \infty$, we have that the throughput $X_b \to \mu_b$, because the throughput of station Q_b cannot exceed its service rate and the utilisation is close to 1. Now, if we apply the forced flow law, we obtain the throughput of the system $X_1 = X_{max} = \mu_b / \overline{V}_b$. This means that, *when there are many customers in the system, the maximum throughput of the bottleneck determines the maximum total throughput of the system*! What happens to the load factor of the stations which are not the bottleneck? We can consider the bottleneck law, stating that $\rho_i = X\overline{D}_i$. Now, we know that $X\overline{D}_b = \rho_b \to 1$. By the same reasoning before, we obtain that:

- $\overline{D}_i$ cannot be larger than $\overline{D}_b$ (otherwise $\rho_b > 1$ which is impossible);
- The bottleneck is the station with the larger service demand;
- The load factor of all the non-bottleneck stations must be lower than 1, therefore their expected population is finite;

It is interesting to note that when a closed queueing network goes to saturation, the jobs tend to accumulate in the bottleneck! Thus, we know that $X \leq 1/\overline{D}_b$ must hold. Let us use this result to bound the expected response time. By Theorem 3.1, we have that:

$$\overline{R} = \frac{N}{X} - \overline{Z} \geq \frac{N}{X_{max}} - \overline{Z} = N\overline{D}_b - \overline{Z}\,.$$

Another lower bound for the expected response time can be derived by considering the situation of $N = 1$ customers in the network. Clearly, in this case, the customer never experiences waiting time at the stations and by increasing the multiprogramming level the performance can only get worse. Therefore, the expected response time of the central subsystem (when there is only one job) is just the sum of the service demands. Hence, in general it holds that:

$$\overline{R} \geq \sum_{i=2}^{K} \overline{D}_i\,.$$

By using this observation, we can resort again to Theorem 3.1 to obtain:

$$X = \frac{N}{\overline{R} + \overline{Z}} \leq \frac{N}{\sum_{i=2}^{K} \overline{D}_i + \overline{Z}}\,.$$

In conclusion, we can summarise these results as follows.

Theorem 3.3 (Asymptotic bounds for interactive systems) Given a closed interactive queueing networks with N jobs, it holds that:

$$X \leq \min\left(\frac{N}{\overline{D}+\overline{Z}}, \frac{1}{\overline{D}_b}\right) \tag{3.5}$$

and

$$\overline{R} \geq \max\left(\overline{D}, N\overline{D}_b - \overline{Z}\right) , \tag{3.6}$$

where

$$\overline{D} = \sum_{j=2}^{K} \overline{D}_j ,$$

and $\overline{D}_b$ is the service demand of the bottleneck, i.e., $\overline{D}_b = \max_{i=2,\ldots,K}(\overline{D}_i)$.

3.7 Example
Consider again the network of Figure 3.6 with the following parameterisation: $\mu_2^{-1} = 3$ s, $\mu_3^{-1} = 4$ s, $\mu_4^{-1} = 3$ s and with the following relative visit ratios: $\overline{V}_2 = 1$, $\overline{V}_3 = 0.6$ and $\overline{V}_4 = 0.5$. The expected thinking time is $\overline{Z} = 40$ s. Assume that all the service times are exponentially distributed with FCFS scheduling discipline. Under these assumptions, we can compute the throughput and the expected response time exactly (the details will be explained in Chapter 4). However, in this example, we aim to compare the throughput and the expected response time as function of the multiprogramming level N.

In order to derive the asymptotic bounds we need to compute

1. the service demand at the bottleneck station. The service demands are:
$$\overline{D}_2 = 3\text{s}\,,\ \overline{D}_3 = 0.6 \cdot 4 = 2.4\text{s}\,, \overline{D}_4 = 0.5 \cdot 3 = 1.5\text{s}\,;$$
thus Q_2 is the bottleneck and its service demand is $\overline{D}_b = 3$s.

2. the service demand of the central subsystem:
$$\overline{D} = \overline{D}_2 + \overline{D}_3 + \overline{D}_4 = 6.9 \text{ s}\,.$$

We can now derive the bounds of Theorem 3.3. In Figure 3.7 we show the plots of the throughput and expected response time. It is worth of notice that that while the bounds are the same regardless to the assumptions on the queueing disciplines and the service time distributions, the correct answer is valid only under the assumptions of exponentially distributed service times and FCFS scheduling disciplines. In other words, if we change these assumptions, the line may change but will surely remain within the same bounds.

One important aspect of operational analysis for interactive systems is that we have a practical way to determine the maximum level of multiprogramming which is acceptable before compromising the expected response time without gaining more throughput. The inspection of the plots of Figure 3.7 reveals that the intersection point of the two asymptotes is a good approximation of the optimal working point of the system. Thus we can solve the equation in N:

$$\frac{N}{\overline{D} + \overline{Z}} = \frac{1}{\overline{D}_b} ,$$

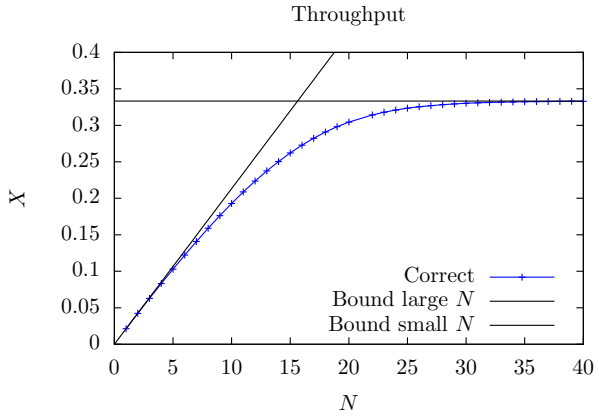
i.e.:

$$N_{opt} = \frac{\overline{D} + \overline{Z}}{\overline{D}_b} . \tag{3.7}$$

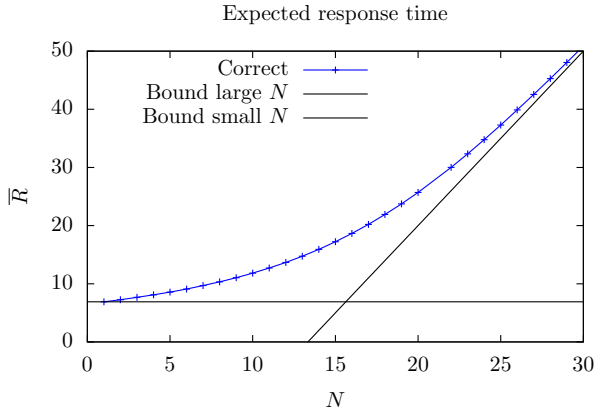
3.8 Example
Let us determine the optimal level of multiprogramming for the model of Example 3.7. By using Equation (3.7) we have:

$$N_{opt} = \frac{6.9 + 40}{3} = 15.63 ,$$

which can be approximated to 15 or 16.



(a) Throughput: plot of the bounds and the exact solution.



(b) Expected response time: plot of the bounds and exact solution.

Figure 3.7: Plots of the performance indices of Example 3.7.

Chapter 4

Product-form queueing networks

4.1 Why do we need product-form queueing networks?

Single queueing systems may be challenging to study analytically and this becomes even worse when we consider network of queues. Indeed, in Chapter 3, we saw some operational and asymptotic analysis of queueing networks. If these approaches are quite general, we observe that they give bounds on the average performance indices rather than their exact values. Moreover, detailed information like the stationary probability distribution of the number of customers in a queue cannot be obtained.

Product-form queueing networks (PFQN) are a particular class of queueing networks whose exact probabilistic analysis is doable in many cases. By 'doable', we mean that we can provide algorithms (and sometimes closed-form expressions) for the computation of many interesting indices which are numerically stable and with a low time and space computational complexity. Indeed, if a queueing network admits an underlying Markov chain, one may in principle generate this Markov chain and solve the global balance equations to obtain the stationary probability distribution. Then, the expected performance indices should be derived. Unfortunately, this approach is not viable. In fact, the cardinality of the Markov chain's state spaces can be so high that they cannot be stored in the memory of computers, and when this is possible, the algorithms for the solution of the linear system of global balance equations (together with the normalising condition) is too expensive to run and becomes numerically unstable.

The trade-off with respect to the systems studied in Chapter 3 is that we must introduce new assumptions, e.g., on the distribution of the service time or on the scheduling disciplines. However, product-form networks either on

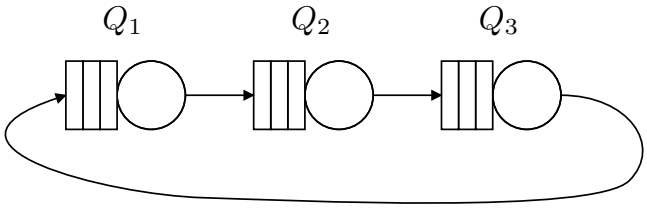


Figure 4.1: Example of simple queueing network with 3 stations.

their open or closed formulation, proved to be extremely important tools for the performance evaluation of real-world systems. Beside, product-form theory is a very fascinating topic that has not been entirely understood, yet. In fact, while the proofs of the results are usually algebraic, a purely probabilistic interpretation of the results presented in this chapter is still to be formulated.

4.2 Markovian queueing networks

We say that a queueing network is Markovian if its underlying stochastic process is a Markov chain, i.e., on a discrete state space and it enjoys the memoryless property. In this section, we study the Markovian representation of queueing networks.

4.2.1 Modelling the service time

Let us begin with the closed queueing network shown in Figure 4.1 under the assumption that the service times are independent and exponentially distributed with parameters μ_1, μ_2 and μ_3 for Q_1, Q_2 and Q_3, respectively. The scheduling discipline is FCFS for all the stations.

If we have only one job in the network, we can show that we can define the CMTC underlying the system with only 3 states . Recall that, in order to have a proper Markovian representation, the state must encode all the information needed to probabilistically characterise the future. The first information that we need is the position of the job. At a given epoch, the job can stay in one of the three queues. The second information should be the residual service of the job but, thanks to the memoryless property of the exponential distribution, we may simplify our representation and ignore it. By contrast, assume that the service times are all independent and distributed according to uniform distributions in the interval $(1, 2)$ seconds. A state representation that only considers the position of the job is not sufficiently rich to probabilistically characterise the future events of the system. Indeed, the situation in which the job has been in Q_1 for 0.5 seconds and that in which it has been in Q_1 for 1.9 seconds are very different. In the former case, we are sure that after 0.2 seconds the job will still be in Q_1 while in the latter we are sure that the job after 0.2 seconds will be in Q_2! The network of Figure 4.1 with uniform service distributions is *not*

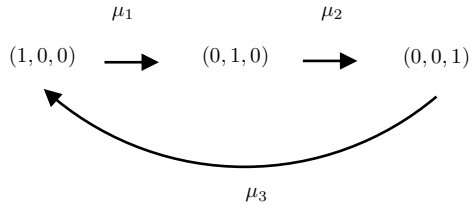


Figure 4.2: Markov chain underlying the network of Figure 4.1 with $N = 1$ job.

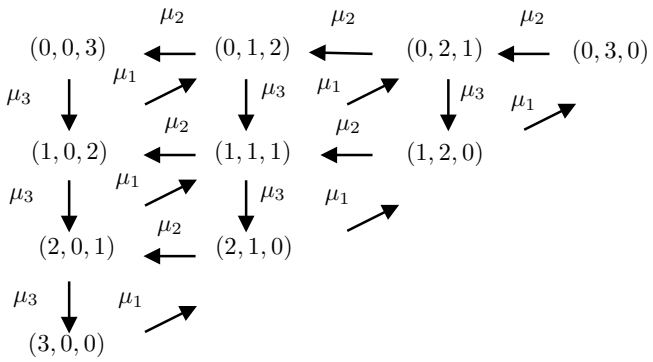


Figure 4.3: Markov chain underlying the network of Figure 4.1 with $N = 3$ jobs.

Markovian. In Figure 4.2, we show the representation of the CTMC underlying the network of Figure 4.1 when there is one job in the system.

If the number of jobs is higher than one and the service times are exponentially distributed, we can encode the state by counting the number of jobs at each station. Also in this case, the residual time of the jobs in service can be ignored thanks to the memoryless property of the exponential distribution. The CTMC underlying the network with 3 jobs is shown in Figure 4.3. The Markov chain has 10 states. Can we say how many states has a closed queueing network with M stations and N jobs assuming that every distribution of the jobs in the stations is possible? This is a simple combinatorial problem whose solution is:

$$|\mathcal{S}| = \binom{N+M-1}{M},$$

where $|\mathcal{S}|$ denotes the cardinality of the state space of the Markov chain. This observation is rather important. In fact, suppose that we have 5 stations and 40 jobs. Although this does not seem a huge systems, if we want to generate the underlying Markov chain we need:

$$|\mathcal{S}| = \binom{44}{5} > 10^6,$$

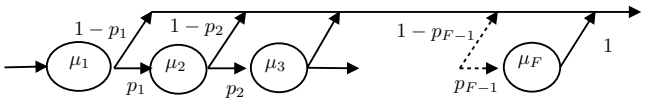


Figure 4.4: Coxian service time distribution.

i.e., more than a million of states! This tells us that having a Markovian representation of a QN is not sufficient to state its numerical tractability. Product-form queueing networks will allow us to study much larger networks than those that can be studied by a direct inspection of its underlying process.

Now, we may be lead to think that Markovian queues must be associated with exponential service time distributions. This is not true. In fact, we may represent any service time consisting of exponential phases. In particular, Coxian service time is very important because with the right combination of parameters, Coxian random variables can approximate arbitrarily well any distribution. It is worth of notice that this result is extremely important, but should be considered carefully. For example, the number of phases required to accurately approximate some distributions can be extremely high (e.g, the uniform distribution).

Figure 4.4 illustrates how we can visualise a Coxian distributed service time. Jobs arrive at the service room and enter the first phase of service whose length is exponentially distributed with rate μ_1. Then, with probability p_1 the job moves to the second phase of service, and with probability $1-p_1$ the job leaves the service room. All the phases of service have independent random durations but they are all exponentially distributed with a certain rate. The number of phases must be finite.

The property of Coxian random variables allow us to conclude that we can approximate arbitrary well queueing networks with any service time distribution (see Remark 4.1). However, when we give the Markovian representation of the queueing network, we have to embed in the state representation the phase of service reached by the job in service. This complicates the state representation and emphasises the problem of the growth of the state space previously explained. For example, if the queueing network with 40 jobs and 5 stations would have one station with a 2 phases Coxian service time, then the number of states of the underlying CTMC exceeds $2 \cdot 10^6$ and if we assume that a second station has a 3 phases Coxian service time, then the number of states would exceed $6 \cdot 10^6$! We will see that product-form queueing networks, for certain service disciplines, allow us to avoid this further growth of the state space.

Remark 4.1 We should pay attention when we say that the behaviour of a queueing network with arbitrary service time can be approximated arbitrary well by Coxian service time. For example, let us consider a queueing network consisting of two stations Q_1 and Q_2 whose service time is deter-

ministic and fixed to 1. The output of Q_1 is Q_2 and vice versa. We assume the presence of 1 job.

This queueing network does *not* have a steady-state behaviour because of its periodic services. In fact, the steady-state distribution must be independent of the initial state of the system. However, if we approximate the service time with a Coxian r.v. with any finite number of phases, we have a network with a well-defined steady-state behaviour. To see this, consider that the CTMC underlying the queueing network with Coxian service times has a finite number of states and is irreducible. Hence, it admits a unique long-term behaviour, the steady-state.

In conclusion, we have to keep in mind that an 'arbitrary close' approximation of a queueing network requires attention, e.g., checking that the original queueing network admits a steady-state behaviour.

4.2.2 Modelling the arrival processes

In open queueing networks, we need to model the arrivals of customers from the outside. Although the system can have multiple arrival streams of customers, we first focus on a single stream. If the inter-arrival times are independent and exponentially distributed, then we have a Poisson arrival process. This is simple to encode in the state of the CTMC, since we do not have to keep track of the residual life of the random variable modelling the time to the next arrival. Clearly, this follows from the memoryless property of the exponential distribution.

In case of multiple arrival streams, it is easy to have a Markovian encoding when they are mutually independent and form Poisson processes. All the delays to the next arrivals enjoy the memoryless property.

We can model arrival processes that are not Poisson with a similar technique adopted for the non-exponential service times, i.e., with the method of phases, and obtain a Markovian queueing network. However, at the moment, general product-form results for such queueing networks are not known, and hence we omit the discussion.

4.1 Example

In this example, we consider the open queueing network depicted in Figure 4.5. It shows a tandem of two queues where arrivals occur at both stations. We assume that the service times of the two queues are independent and exponentially distributed with rates μ_1 and μ_2, respectively. Arrivals at the first and second stations follow two independent Poisson processes with intensities λ_1 and λ_2, respectively. Notice that we may even see the model consisting of a single Poisson arrival process with intensity $\lambda_1 + \lambda_2$ and then with a probabilistic routing. Customers go to the first queue with probability $\lambda_1/(\lambda_1 + \lambda_2)$ and to the second queue with probability $\lambda_2/(\lambda_1 + \lambda_2)$.

The Markov chain underlying the queueing network is shown in Fig-

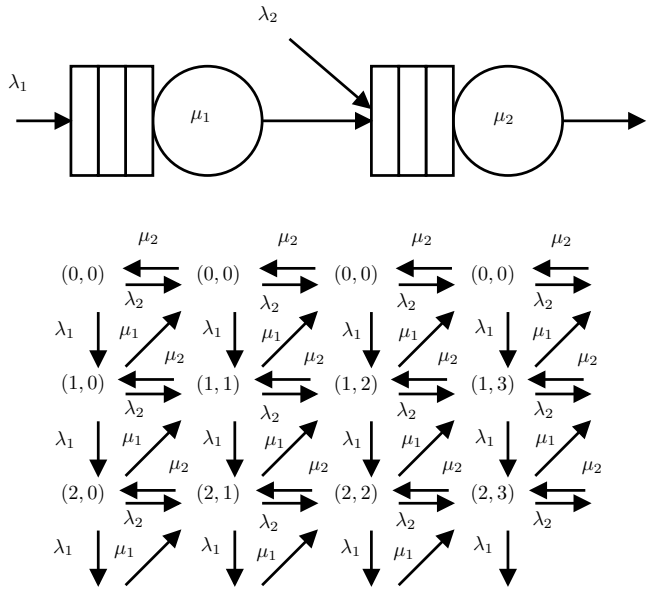


Figure 4.5: Open queueing network and its underlying Markov chain.

ure 4.5. The number of customers in the first queue ranges from 0 to infinity, and the same happens for the second queue. The joint process in the Cartesian product $[0, \infty) \times [0, \infty)$. The graphical representation of the states shows the regularity of the Markov chain. The stability condition is $\lambda_1 < \mu_1$ and $\lambda_1 + \lambda_2 < \mu_2$. Notice that, in general, the arrival process at the second queue is not a Poisson process (although we will see soon that it is a Poisson process when the system is in steady-state) but its intensity must surely be $\lambda_1 + \lambda_2$.

4.2.3 Modeling the independent probabilistic routing

In this section, we focus on how we can model the independent probabilistic routing in queueing networks when the service time distributions of the stations are independent and exponential. Suppose we have an exponentially distributed random variable X with p.d.f.:

$$f(t) = \mu e^{-\mu t}\,.$$

Moreover, we have a Bernoulli random variable B, independent of X, that takes value 0 with probability p and 1 with probability $1 - p$. The outcome of our experiment consists of the pair with the observation of the duration of X and the outcome of the Bernoulli experiment (x, B). This experiment is stochastically

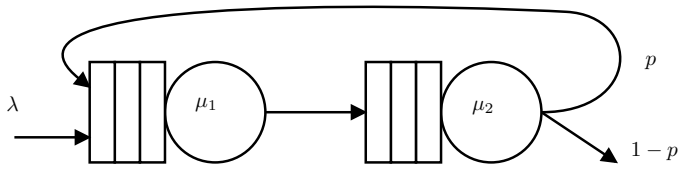


Figure 4.6: Network with probabilistic routing and exponential service time and Poisson arrival stream.

indistinguishable for another experiment in which we consider two independent exponential random variables X_1 and X_2 with rates $p\mu$ and $(1-p)\mu$, respectively, and we choose to observe the r.v. C defined as:

$$C = \begin{cases} 0 & \text{if } X_1 < X_2 \\ 1 & \text{otherwise} \end{cases}$$

Hence, we have the pair $(\min(X_1, X_2), C)$. This claim can be proved thanks to the properties of the exponential distribution.

From this observation, we can easily map the independent probabilistic routing to the underlying Markov chain. We basically split the transition modelling the service into two or more transitions according to the probabilistic routing.

4.2 Example
Let us consider the network of Figure 4.6 and focus, for simplicity, on the single state n, m with n, m strictly positive. Here, n denotes the number of jobs in the first queue, and m the number of jobs in the second queue. Let us consider which states are reachable from n, m:

- $(n+1, m)$ due to an arrival at the first queue with rate λ;
- $(n-1, m+1)$ due to a job completion at the first queue with rate μ_1;
- $(n+1, m-1)$ due to a job completion at the second queue and a probabilistic choice of routing to the first station. This happens with probability p, thus the rate of this transition is $\mu_2 p$;
- $(n, m-1)$ due to a job completion at the second station and a departure with probability $1-p$. The transition rate is $\mu_2(1-p)$.

Not all the states have these outgoing transitions. For example, if we consider state $(0, 0)$ the only state that can be reached is state $(1, 0)$ with rate λ.

4.3 Burke's theorem

The first result on product-form is due to Paul J. Burke [2] that in 1956 studies the output process of a M/M/m queueing system where m ranges from 1 to

∞. This will allow us to study the composition of queueing systems in a simple way.

Theorem 4.1 (Burke's theorem) Given a M/M/m queueing system in steady-state with arrival intensity λ and service rate μ, the following propositions hold:

1. The departure process is Poisson with intensity λ
2. At time t, the number of customers in the queue is independent of the sequence of departure times prior to t.

Proof. The simplest proof of the theorem is not that proposed in the seminal work by Burkes and relies on the observations that a birth&death process is time-reversible. Recall that a stationary CTMC is reversible if $X(t)$ and $X(\tau - t)$ have the same statistics from all t, τ. Thus, the use of the reversibility argument is allowed by the hypothesis that the chain is in steady-state.

In our case, notice that a transition from state n to state $n+1$ in the forward chain corresponds to a transition from $n+1$ to n in the backward chain. In other words, the instants of arrival in the forward chain correspond to the instants of departures in the reversed chain. Thus, since the arrival process is Poisson with intensity λ, also departure process is Poisson with intensity λ and this concludes the proof of the first bullet.

As for the second bullet, consider the backward process. Clearly, the state at time t is independent of the arrival times that will occur after time t (by the memoryless property of the exponential distribution and independence of the inter-arrival and service times). This means that in the forward process, the state at time t is independent of the departure times prior to time t, as required. □

Burke's theorem is somehow counter-intuitive. Consider the claim that the output process is Poisson. The time between two consecutive job departures is exponentially distributed with rate μ when there are at least two jobs in the queue, but when there is only one job, after the first departure we have to wait an arrival of another job and its service to see a second departure. It is surprising to see that the compound process is Poisson! Indeed, we must recall that Burke's theorem holds only if the queue is in steady-state.

The second bullet of Theorem 4.3 is as important as the first. Consider the tandem system of Figure 4.7 assuming that Q_1 is a M/M/1 and Q_2 has independent exponential service times with single server. Since the output of Q_1, in steady-state, is Poisson with intensity λ, and clearly the service process of Q_2 is independent of its arrival process, then we have that Q_2 is also M/M/1 queue. We can write:

$$\pi_1(n_1) = \left(1 - \frac{\lambda}{\mu_1}\right)\left(\frac{\lambda}{\mu_1}\right)^{n_1},$$

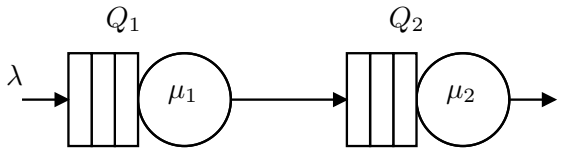


Figure 4.7: Burke's theorem applied to a tandem of two queues.

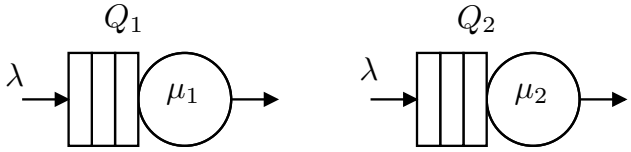


Figure 4.8: Two independent M/M/1 queues.

and

$$\pi_2(n_2) = \left(1 - \frac{\lambda}{\mu_2}\right)\left(\frac{\lambda}{\mu_2}\right)^{n_2} .$$

What can we say about $\pi(n_1, n_2) = Pr\{N_1 = n_1, N_2 = n_2\}$, i.e., the joint distribution of Q_1 and Q_2? The number of jobs N_2 at time t depends on the departure times from Q_1 prior to t and, of course, on the service times in Q_2 (which are independent of what happens in Q_1). However, by Burke's theorem, the number of jobs N_1 in Q_1 at time t is independent of the sequence of departure times prior to t, and hence N_1 is independent of N_2, in steady-state. Thus we have:

$$\pi(n_1, n_2) = \pi_1(n_1) \cdot \pi_2(n_2) . \tag{4.1}$$

This is extremely surprising! Although Q_1 and Q_2 are not independent, since Q_1 sends its jobs to Q_2, the joint stationary probability of observing n_1 jobs in Q_1 and n_2 in Q_2 is in *product-form*. What is the difference between independence and product-form? For example, Equation (4.1) holds also for the system of Figure 4.8 where the two queues are clearly independent. However, Equation (4.1) holds for the independent queues without the assumption of being in steady-state, while the product-form of the system of Figure 4.7 requires the system to be in steady-state.

4.3 Example
Given the queueing network of Figure 4.9 where the arrival streams at Q_1 and Q_2 are independent and Poisson and all the servers are single, independent and their service times are exponentially distributed. The routing between Q_1 and Q_2, Q_3 is probabilistic and independent with probabilities p_12 and p_13, with $p_{12} + p_{13} = 1$.

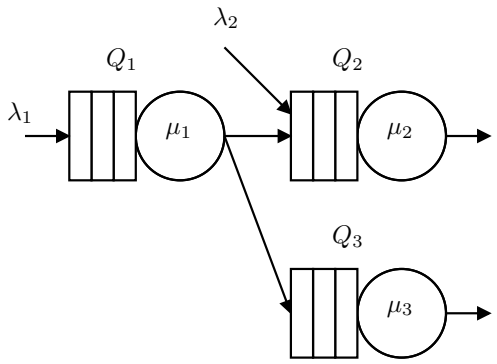


Figure 4.9: Example of tree-structured queueing network.

1. Find the stability condition
2. Determine the stationary probability that both Q_2 and Q_3 are empty
3. Determine the stationary probability that neither Q_2 nor Q_3 are empty
4. Find the expected stationary response time of a job arriving either at Q_1 or Q_2

Solution. The stability condition requires that the load factor of the three queues is strictly lower than 1:

$$\lambda_1 < \mu_1 \ \wedge \ \lambda_1 p_{12} + \lambda_2 < \mu_2 \ \wedge \ \lambda_1 p_{13} < \mu_3 \, .$$

Q_1 is a M/M/1 queue. For Q_2 and Q_3, we use Burke's theorem to study the systems in isolation. In steady-state, the output of Q_1 is a Poisson process, and by the property of splitting of Poisson processes, the arrivals at Q_2 from Q_1 follows a Poisson process with intensity $\lambda_1 p_{12}$ and the arrivals at Q_2 follows a Poisson process with intensity $\lambda_1 p_{13}$. The compound arrival process seen by Q_2 is the superposition of the arrivals from Q_1 and those from the outside. The former is Poisson in steady-state, the latter is always Poisson, so in steady-state we can use the result on the superposition of independent Poisson processes and conclude that Q_2 sees an arrival stream that follows a Poisson process with intensity $\lambda_1 p_{12} + \lambda_2$. Thus, the probability of Q_2 and Q_3 being empty is, by the product-form result, the product of the probabilities of finding Q_2 and Q_3 empty, i.e.:

$$\Pr\{N_2 = 0, N_3 = 0\} = \left(1 - \frac{\lambda_1 p_{12} + \lambda_2}{\mu_2}\right)\left(1 - \frac{\lambda_1 p13}{\mu_3}\right) \, .$$

Analogously, since the probability of being non-empty for a M/M/1 queue is its load factor, we have:

$$\Pr\{N_2 > 0, N_3 > 0\} = \frac{\lambda_1 p_{12} + \lambda_2}{\mu_2} \cdot \frac{\lambda_1 p13}{\mu_3}\,.$$

Finally, in order to find the expected response time, we apply Little's law. The expected number of jobs in the system is the sum of the expected number of jobs in each queue:

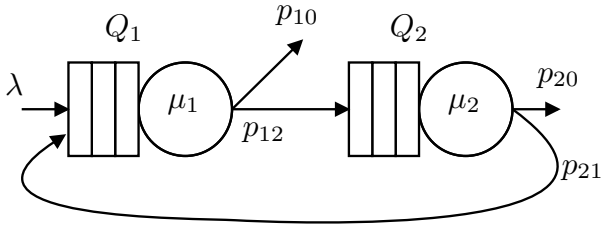
$$\overline{N} = \overline{N}_1 + \overline{N}_2 + \overline{N}_3 = \frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2} + \frac{\rho_3}{1-\rho_3}\,,$$

where $\rho_1 = \lambda_1/\mu_1$, $\rho_2 = (\lambda_1 p_{12} + \lambda_2)/\mu_2$, $\rho_3 = \lambda p_{13}/\mu_3$. Thus, we have:

$$\overline{R} = \frac{\overline{N}}{\lambda_1 + \lambda_2}\,.$$

4.4 Jackson networks

Burke's theorem is a very important result, with a clear probabilistic interpretation. However, the drawback is that it imposes constraints on the network topology that, for example, must not have cycles. Consider the queueing network in Figure 4.10: the output of Q_2 is sent back to Q_1 with probability p_{21}. When we are in steady-state, it can be seen that the aggregated arrival process at Q_1 is *not* a Poisson process. However, Jackson'n theorem states that the network is still in product-form and hence easily tractable.



$$p_{12} + p_{10} = 1$$
$$p_{21} + p_{20} = 1$$

Figure 4.10: Network with exponential queues and cycle. Burke's theorem cannot be applied.

Theorem 4.2 (Jackson's theorem) Consider an open queueing network of N stations with the following assumptions:

Station types: All the stations have exponential service time distribution with rate μ_i and an arbitrary positive number m_i of servers (possibly infinite), with $i = 1, \ldots, N$. The scheduling discipline is FCFS for all stations.

Routing: The routing is independent and probabilistic according to matrix $\mathbf{P}$ where entry p_{ij} denotes the probability of moving from station i to station j. Hence, $\sum_{j=1}^{N} p_{ij} \leq 1$.

Arrivals: The arrival stream from the outside are independent Poisson processes with intensity $\lambda_i \geq 0$ at station i.

Traffic: The system of traffic equations:

$$e_i = \lambda_i + \sum_{j=1}^{N} e_j p_{ji} \tag{4.2}$$

has a unique solution.

Then, the following propositions hold true:

- The network is stable if for all $i = 1, \ldots, N$ we have $e_i/(m_i\mu_i) < 1$.
- The stationary distribution of state $\mathbf{n} \in \mathbb{N}^N$ is:

$$\pi(\mathbf{n}) = \prod_{i=1}^{N} g_i(n_i)\,, \tag{4.3}$$

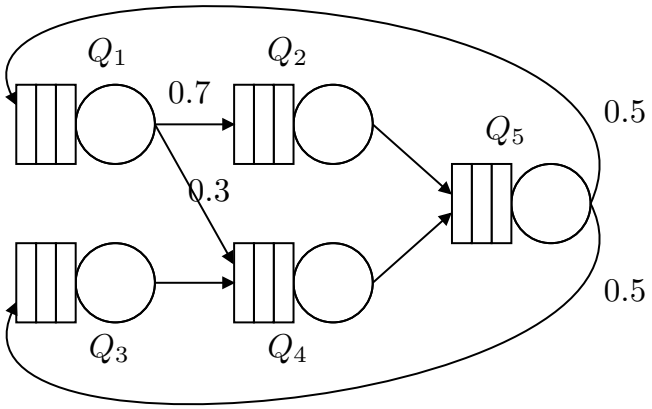
where $g_i(n_i)$ is the steady-state probability of a $M/M/m_i$ queue with service rate μ_i and arrival intensity e_i.

Notice that, it is easy to state if the system of traffic equations has a unique solution. If we consider the outside as a virtual station and imagine that a job that leaves the system visits the outside-station and then returns the queueing network, the traffic equation system has a unique solution if and only if this customer can move from any station to any other station.

Jackson's theorem is quite different from Burke's theorem in the sense that it does not state any property of the the arrival or departure process at the stations, aside from their intensity. However, it states that, even if the arrival process at a station may be *not* a Poisson process (even in steady-state), the marginal stationary distribution of that station is the same we would observe if the arrival stream were Poisson. Moreover, the product-form of Equation (**??** holds.

4.5 Exercises with solution

Exercise 1. Given the following Gordon-Newell queueing in which all the stations are single server with service rates $\mu_1 = 1.6$ $\mu_2 = 2$, $\mu_3 = 1.5$, $\mu_4 = 2$, $\mu_5 = 1.5$ jobs per second. Answer the following questions:



- Which station is the bottleneck of the system?
- Compute the maximum throughput at Q_3
- Assume there is only one customer in the system, compute the throughput at Q_3
- Use MVA to compute the expected number of customers in the bottleneck when the system has 3 customers

Solution. Recall that, since the queueing network belongs to the class of Gordon-Newell networks, the service times are independent and exponentially distributed and the scheduling disciplines are FCFS.

In order to determine the bottleneck, we need to compute the station with the highest utilisation. Thus, we proceed with the determination of the relative visit ratios to the stations by solving the system of traffic equations:

$$\begin{cases} e_1 = 0.5e_5 \\ e_2 = 0.7e_1 \\ e_3 = 0.5e_5 \\ e_4 = 0.3e_1 + e_3 \\ e_5 = e_2 + e_4 \end{cases}$$

Recall that, since the routing is ergodic and the queueing network is closed, thus system admits infinite solutions that differ by a positive real constant. Let us choose a reference station, for example Q_5, i.e., let us impose $e_5 = 1$. We

immediately have, $e_1 = 0.5$ and $e_3 = 0.5$. From the second and forth traffic equations we finally obtain $e_2 = 0.35$ and $e_4 = 0.65$. Recall that e_i represents the expected number of visits at station i for every visit at the reference station. Now, we have:

$$\frac{e_1}{\mu_1} = \frac{0.5}{1.6} = 0.3125\,, \quad \frac{e_2}{\mu_2} = \frac{0.35}{2} = 0.1750\,, \quad \frac{e_3}{\mu_3} = \frac{0.5}{1.5} = 0.3333\,,$$

$$\frac{e_4}{\mu_4} = \frac{0.65}{2} = 0.3250\,, \quad \frac{e_5}{\mu_5} = \frac{1}{1.5} = 0.6667\,.$$

Thus, the bottleneck is station Q_5.

The maximum throughput in is obtain when the population of the network K tends to infinity. Once this happens, the throughput of the bottleneck is its service rate, in our case $X_5 = 1.5$ jobs per second. Since $e_3 = 0.5e_5$, we have that the throughput at Q_3 is 0.75 jobs per second (clearly lower than its service rate).

If there is only one customer in the queueing network, we have that the system works at its lowest capacity, i.e., we obtain its minimum throughput (excluding the case of the empty system). For this type of analysis, we exploit the fact that there is no queueing at the stations, and hence the service times and the response times are identical. Specifically, the expected response time at station i is $1/\mu_i$. If we divide the values $e_1, \ldots, e_5$ by e_3, we obtain the expected number of visits at station Q_i for each visit at station Q_3, i.e.:

$$e'_1 = \frac{0.5}{0.5} = 1\,, \quad e'_2 = \frac{0.35}{0.5} = 0.7\,, \quad e'_3 = \frac{0.5}{0.5} = 1\,,$$

$$e'_4 = \frac{0.65}{0.5} = 1.3\,, \quad e'_5 = \frac{1}{0.5} = 2$$

Thus, the customer, once it exists from station Q_3, has an expected return time $\overline{Ret}$ that can be computed as:

$$\overline{Ret} = \sum_{i=1}^{5} \frac{e'_i}{\mu_i} = 3.625 \text{ s}\,.$$

Since there is only one customer, the throughput of Q_3 is:

$$X_3^1 = \frac{1}{\overline{Ret}} = \frac{1}{3.625} = 0.276 \text{ jobs/s}\,.$$

Now, we have to apply MVA to compute the throughput at the bottleneck when we have three customers in the system. We recall that MVA is a recursive algorithm in the number of customers in the networks. Let us begin with $K = 0$. In this case, we have $\overline{N}_i^0 = 0$ and $X_i^0 = 0$ for all $i = 1, \ldots, 5$. Now we apply the arrival theorem, and we recall that:

$$\overline{R}_i^{k+1} = \frac{1 + \overline{N}_i^k}{\mu_i}\,,$$

and we obtain:

$$\overline{R}_1^1 = \frac{1}{\mu_1} = 0.625\,, \quad \overline{R}_2^1 = \frac{1}{\mu_2} = 0.5\,, \quad \overline{R}_3^1 = \frac{1}{\mu_3} = 0.667$$

$$\overline{R}_4^1 = \frac{1}{\mu_4} = 0.5\,, \quad \overline{R}_5^1 = \frac{1}{1.5} = 0.667\,.$$

Recall that our reference station is Q_5 (which is also the bottleneck). We can compute the throughput at Q_5 as follows:

$$X_5^k = \frac{k}{\sum_{j=1}^{5} e_j \overline{R}_j^k}\,.$$

Thus, we have $X_5^1 = 0.552$ jobs/s. We can obtain the throughput of all the stattions by noting that $X_i^k = e_i X_5^k$ since the throughput must be proportional to the relative visit ratios. Then, we have:

$$X_1^1 = 0.267\,, \quad X_2^1 = 0.193\,, \quad X_3^1 = 0.276\,, \quad X_4^1 = 0.359\,.$$

Finally, we conclude the case $k = 1$ by using Little's theorem to derive the expected number of jobs in each station, i.e., $\overline{N}_i^k = R_i^k X_i^k$.

$$\overline{N}_1^1 = 0.169\,, \quad \overline{N}_2^1 = 0.0965\,, \quad \overline{N}_3^1 = 0.184\,, \quad \overline{N}_4^1 = 0.180\,, \quad \overline{N}_5^1 = 0.368\,.$$

Notice that the sum of the expected jobs in the queues is 1 (aside for the numerical approximation). We now consider the case $k = 2$ and we use again the arrival theorem, obtaining:

$$\overline{R}_1^2 = 0.731\,, \quad \overline{R}_2^2 = 0.548\,, \quad \overline{R}_3^2 = 0.789\,, \quad \overline{R}_4^2 = 0.590\,, \quad \overline{R}_5^2 = 0.912\,.$$

At this point, we may again compute the throughput at the reference station and we obtain $X_5^2 = 0.890$ jobs per second, and we derive all the others by using the relative visit ratios:

$$X_1^2 = 0.445\,, \quad X_2^2 = 0.332\,, \quad X_3^2 = 0.445\,, \quad X_4^2 = 0.579\,, \quad X_5^2 = 0.890\,.$$

Finally, we apply Little's law and conclude the case $k = 2$:

$$\overline{N}_1^2 = 0.325\,, \quad \overline{N}_2^2 = 0.182\,, \quad \overline{N}_3^2 = 0.351\,, \quad \overline{N}_4^2 = 0.342\,, \quad \overline{N}_5^2 = 0.812\,.$$

The case $k = 3$ follows analogously:

$$\overline{R}_1^3 = 0.828\,, \quad \overline{R}_2^3 = 0.591\,, \quad \overline{R}_3^3 = 0.901\,, \quad \overline{R}_4^3 = 0.671\,, \quad \overline{R}_5^3 = 1.208\,.$$

To conclude the exercise, we just need to compute the throughput at our reference station:

$$X_5^3 = 1.105 \text{ jobs per second}\,.$$

Exercise 2. A system processes the jobs in two phases whose durations are exponentially distributed and independent. We want to compare three different architectures in terms of average response time:

- One single server with processor sharing discipline capable of serving each phase with an expected time of $(2\mu)^{-1}$
- One single server with FCFS discipline capable of serving each phase with an expected time of $(2\mu)^{-1}$
- Two servers in tandem each of which takes care of serving one phase with an expected time of μ^{-1}

Assume the jobs arrive at system according to a Poisson process of intensity λ jobs per second.

Hint. The variance of a random variables with two exponential stages with rate 2μ is $(2\mu^2)^{-1}$.

Solution. Let us consider the first configuration. Since the discipline is PS and the arrival follows a Poisson process, we now that the expected response time is that of a M/M/1 queue by the insensitivity property of this discipline. The expected service time is $2(2\mu)^{-1}$, i.e., μ^{-1}, and hence the service rate is μ. Therefore, the expected response time is:

$$\overline{R}_1 = \frac{1}{\mu - \lambda}\,.$$

Let us consider the second configuration. Since the queueing discipline is FCFS, we must treat the queue as a M/G/1 system and we need to compute the variance of the service time distribution. In our case the, we are dealing with an Erlang random variable with two stages with rate (2μ) and we know its variance is $(2\mu^2)^{-1}$. Thus, by using the P-K formula for the expected indices, we have:

$$\overline{R}_2 = \frac{\lambda/\mu + \lambda\mu(2\mu^2)^{-1}}{2(\mu - \lambda)} = \frac{1 - 4\mu}{4\lambda\mu - 4\mu^2}\,.$$

We notice that the variance of the service time is lower than that of an exponential distribution and hence the FCFS discipline is preferable with respect to the PS.

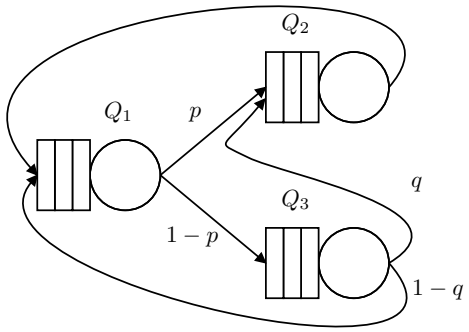
The third case consists of a tandem of two exponential queues, i.e., a Jackson's network. Both queues are identical and, thanks to the product-form, we can say the expected total number of jobs in the system is $2\rho/(1-\rho)$, i.e., the sum of the expected number of jobs in the first and second queue. Thus, the expected response time, by Little's theorem, is:

$$\overline{R}_3 = \frac{2}{\mu - \lambda}\,,$$

which results to be the configuration with the worst performance.

Exercise 3. Let us consider the system depicted in the figure below. The jobs in the system are firstly pre-processed by Q_1 and then a database request is formulated. The system has two databases: a master (Q_2) and a slave (Q_3). The master serves the write requests which are 40% of the total ($p = 0.4$). The read requests are sent to Q_3, but they may find a valid copy of the data with probability $1 - q = 0.9$. When data are not found, the request in sent to Q_2 to be served. Q_1 can serve 6 requests per second, Q_2 and Q_3 can serve 3 requests per second.

All the stations has a First Come First Served queueing discipline. Answer



to the following questions:

1. What is the bottleneck of the system?
2. Consider Q_1 as reference station. What is the maximum and minimum throughput of the system?
3. What are the conditions for the product form of the system?
4. Henceforth, assume the conditions stated before. If the system has infinite jobs, what is the expected number of jobs in the three queues?
5. If you can replace the database with higher load factor, what should be the speed of the new one in order to have the same load factor of the other?

Solution. In order to identify the bottleneck, we need to solve the system of traffic equations:

$$\begin{cases} e_1 = e_2 + e_3(1-q) \\ e_2 = e_1 p + e_3 q \\ e_3 = e_1(1-p) \end{cases}$$

Since the queueing networks is closed, we have to fix a reference station, e.g., station Q_1, and set $e_1 = 1$. Thus, we have:

$$e_1 = 1\,, \quad e_2 = p + q - pq\,, \quad e_3 = 1 - p\,.$$

Recalling that $p = 0.4$ and $q = 0.1$, we have:

$$e_1 = 1\,, \quad e_2 = 0.46\,, \quad e_3 = 0.6\,.$$

Recall that e_i represents the expected number of visits performed by a job at station Q_i for each visit at the reference station Q_1. Thus, we have:

$$\frac{e_1}{\mu_1} = 0.167\,, \quad \frac{e_2}{\mu_2} = 0.153\,, \quad \frac{e_3}{\mu_3} = 0.2\,.$$

Therefore, the bottleneck is Q_3. Since Q_3 is the bottleneck, its maximum throughput corresponds to its service rate, i.e., $X_3^{\max} = 3$. By recalling that the throughputs of the stations are proportional to the relative visit ratios, we have:

$$X_1^{\max} = \frac{X_3^{\max}}{e_3} = 5 \text{ jobs/s}\,.$$

The conditions required for the Gordon-Newell product-form of the system are:

- Independent and exponentially distributed service times.
- FCFS discipline.
- Independent probabilistic routing.

If the number of jobs tends to infinity, the expected number of jobs in the bottleneck station will be also infinite. Whereas, the expected number of jobs in the other queues can be seen as open stations whose arrival rate corresponds to their maximum throughput. For example, for Q_1 we have:

$$\overline{N}_1^{\max} = \frac{X_1^{\max}/\mu_1}{1 - X_1^{\max}/\mu_1} = 5 \text{ jobs}\,.$$

Finally, the database with highest load factor is Q_3 and hence we need to solve the following equation in x:

$$\frac{e_3}{x} = \frac{e_2}{\mu_2}\,,$$

which gives $x = \mu_2 e_3/e_2 = 3.91$ jobs per second.

4.6 Other exercises

Exercise 1 Consider the queueing network of Figure 4.7 and the product-form solution obtained by Burke's theorem. In this exercise, you are asked to verify that the product-form solution obtained by Burke's theorem satisfies the system of global balance equations associated with the CTMC underlying the queueing network.

Chapter 5

Performance testing

Performance testing on real systems is a crucial aspect in the life-cycle of hardware and software architectures. While the design of a product is in general driven by models and their analysis, once this is completed or partially completed, performance tests are carried out. In this chapter, we focus on the assessment of the quantitative properties of a software architecture. We can identify four types of experiments:

1. Performance regression testing,
2. Performance optimisation testing,
3. Performance benchmarking testing,
4. Scalability testing.

In the development of a software project, functional regression testing is an important procedure adopted by software engineers. It consists in the verification of the functional properties of the software under development after some changes in the code. *Performance regression testing* plays a similar role for the quantitative properties of the software architecture. Therefore, the performance regression testing aims at checking if the performance of the software has been degraded, possibly below the acceptable levels, by the changes introduced in the source code. Indeed, even small modifications to the source code may cause severe performance degradation. For example, if the string associated with a query to a database is slightly changed by requiring the introduction of a condition on a field which does not have an associated index, this may turn to be a serious problem for the user's experience. In practice, performance regression testing are done less frequently than functional regression testing, but this does not mean that they are less important. From a methodological point view, performance regression testing consists in the comparison of two systems: the system that we had *before* the changes and the system that we have *after* the changes.

The *Performance optimisation testing* aims at finding the best software configuration for achieving the highest performance in the most likely contexts. By software configuration, we mean two aspects:

- What are the best configuration parameters of the software itself or the components it relies on? For example, suppose that the software under development uses a web server such as Apache to perform some operations. What is the correct level of multiprogramming for the configuration of Apache?
- Sometimes, we can adapt the code of the software under development to achieve better performance, although the functional properties are not changed. For instance, this is the case when we choose a certain data structures to perform a certain operation, e.g., we choose to use a dictionary instead of a linked list to store certain informations. Another interesting example can be the use of lazy loading in Object-Relational-Mapping (ORM) systems (see Example 5.1).

In this case, we need to test different systems, each of which has its own configuration. In most of the practical cases, the number of parameters to configure can be high, and hence finding the optimal configuration can be a time consuming work.

5.1 Example (Lazy loading in ORM)
The ORM is a programming technique that relies on complex frameworks that allows the conversion of data between incompatible type systems and is based on object oriented programming languages. Specifically, ORM allows for an elegant integration of object oriented code with database system. For instance, assume that we desire to retrieve a record from a database in a C# program. We may proceed, without ORM, as follows:

```
1 var query = "SELECT id, model, engine_type, power, color,
      brand, nationality
2              FROM cars NATURAL JOIN brands
3              WHERE id = 10";
4 var results = context.Cars.FromSqlRaw(query).ToList();
5 var model = result[0]["model"];
```

Notice that we have embedded a language (SQL) as string into another language (C#). The solution that makes use of ORM would look like the following:

```
1 var car = repository.GetCar(10);
2 var model = person.GetModel();
```

In the ORM example, we should decide if the join with the table *brands* should be done when we retrieve the car or later, only in the event that we need to access to the information stored in that table. This latter approach is called *lazy loading*. It should be clear that the abstraction mechanism provided by the ORM framework guarantees that the functional behaviour

in case of lazy or non-lazy operations is the same, but the performance may drastically change. According to the following code and program behaviour, lazy loading may be or may be not convenient. Usually, lazy loading is controlled by some annotations to the code that the ORM framework interpret to takes the appropriate decisions. These annotations are decided by the software developers and should be consequence of the outcomes of the performance optimisation testing phase.

The *Performance benchmarking testing* is a set of experiments designed to describe to the end users the performance of a software. The objective is more related to communication and marketing rather than to enhancing of the quantitative aspects of the product. Therefore, in this phase, the focus of the software engineers is more on how to describe the (hopefully high) performance of the software rather than in improving it. For this purpose, the companies usually refer to standard programs to generate the workload and measure the performance. In this way, the software can be compared to analogue ones in a fair way. These testing programs are known as benchmarks and are important to compare different products in a meaningful way. The Standard Performance Evaluation Corporation (SPEC) is a 'non-profit corporation formed to establish, maintain and endorse standardized benchmarks and tools to evaluate performance and energy efficiency for the newest generation of computing systems'.

The last type of testing that we consider is the *scalability test*. This aims at estimating the maximum number of simultaneous users that the system can support or, in other words, its maximum workload. The maximum workload is usually obtained by investigating the point at which a certain level of expected performance is not met any more. As we saw in Chapter 3, in open systems, the intensity of the workload is measured in number of requests per unit of time, while in closed systems the intensity of the workload is measured with the number of simultaneous interactive users.

5.1 Tools for measuring software performance

When we want to measure the performance of a certain software product, we use a software that emulates the behaviour of the users and records the performance indices. The system that we aim to study is usually known as the *System Under Test* (SUT) while the software that we use to study its performance is called *benchmark*. We distinguish two types of benchmarks:

- *Competitive benchmarks* which are standardised programs aimed at assessing the performance of a certain software or hardware equipment. These standardised benchmarks are used to compare different systems with the same or similar functional behaviours. For instance, we may compare the performance of two database management systems, like Oracle or PostgreSql. These benchmarks are often developed by independent organisations like SPEC[1] or by programmer teams.

[1] `https://www.spec.org`

- *Research benchmarks* which are tools developed ad-hoc for measuring the performance of a system. In this case, the tool is more focused on the improvement of the performance of the system during its development or its maintenance, and the competitive aspects are less important. While in principle it is possible to develop a research benchmark tool in any programming language, in many cases we resort to tools that are developed to this aim and are flexible enough to meet our needs. For example, *httperf*[2] is an open source tool designed to measure the performance of web servers, and *Tsung*[3] is a tool that allows for measuring the performance of web servers, databases and other types of service. There are several advantages in using these tools rather than developing new tools from scratch. These tools are optimised to handle the concurrency efficiently, and thus they use the resources of the machine that performs the measurements in an efficient way. Indeed, notice that the emulation of several customers is usually implemented with a massive use of threads, but this can be extremely inefficient and resource demanding. Moreover, tools like Tsung and httperf are equipped with nice functionalities that allow the engineer to easily obtain the statistics on the executed test. Therefore, we recommend that, whenever it is possible, these tools should be adopted. In some cases, the peculiarities of the SUT can make these tools unusable and hence the programmers have to develop a benchmarking tool tailored to their specific needs. In the development of ad-hoc tools, the use of the resources and in particular of the concurrency requires a lot of attention. Indeed, we do not want our benchmark tool to become the system bottleneck! In these cases our experiments measure the performance of the benchmark rather than those of the SUT.

Furthermore, we can characterise the benchmarks into the following categories:

Synthetic : Synthetic benchmarks generate a fictitious workload for the SUT that should be as similar as possible to what will be the real workload. The computations or the services produced by the SUT are meaningless in this context. Most of benchmarks that we will see are synthetic. In many cases, the engineers that want to test the SUT do not even have the users' application and hence need to resort to this measurement method. Clearly, it is important that the synthetic benchmarks behave, from the point of view of the SUT, in a way that is almost indistinguishable from what will be the real system.

Micro : These benchmarks are small programs that test only one aspect of the SUT, independently of the rest of the system. For example, in a software architecture, we may decide to check if the response time of the database is below our expectations and hence we develop a test which concentrates only on its quantitative properties.

[2] https://github.com/httperf/httperf
[3] http://tsung.erlang-projects.org

Kernel : In many cases, during the development of a software, the engineers know which portion of the system will require the largest effort. Thus, they may decide to test the performance of just that part of the code under the assumption that when this has good performance, the whole system is expected to behave well.

Application : The application benchmarks are entire applications which are designed to be representative of the whole set of applications. For example, Ghostscript is an interpreter for the PostScript language and for PDF. For its intrinsic nature, Ghostscript is often used as application benchmark. In these cases, the choice of the representative application requires attention: its work has to be meaningful for the SUT characteristics that we desire to understand and it must be easy to obtain the statistics recorded during the experiment.

5.2 Designing a benchmarking experiment

Let us consider the generic *System Under Test* (SUT) depicted in Figure 5.1. There are two possible ways to configure the benchmarking experiment: the first consists in setting up an open system in which we generate the arrival process, the second is based on the idea of interactive system, i.e., we use a fixed number of customers.

In general, the main concerns when one designs a benchmarking experiment is the proper characterisation of the workload. While there are entire volumes and research areas devoted to the problem of the workload characterisation (and we refer to the bibliography section at the end of this chapter), in this book we just state the guidelines for this important phase of the benchmarking experiments.

Workloads can be clustered into two classes:

- *Executable workload* that can be directly executed on the SUT;
- *Non-executable workload* that can be used in analytical models such as those seen in Chapters 2 and 3.

Another classification of the workload models is the following:

- *Natural workload.* This type of workload is taken from measurements or programs that are present in real systems.
- *Synthetic workload.*

i

5.2.1 Open system tests

In the benchmark experiments based on an open system, the main configuration parameter is the workload, i.e., the characterisation of the requests that arrive

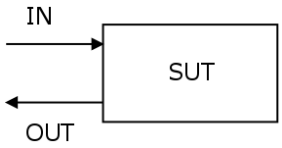


Figure 5.1: Graphical representation of the System Under Test (SUT).

at the SUT. When we talk about workload characterisation, we do not mean just the intensity of the arrival process but also its other properties. For instance, do we want to test a bursty or a smooth workload? In general, applications work much better with smooth workloads than bursty ones, hence the results may vary based on this choice.

5.2.2 Open-loop benchmarking

5.2.3 Closed-loop benchmarking

5.3 Designing your own benchmark

5.4 Using a benchmark tool

5.4.1 SPEC benchmarks

5.4.2 Tsung benchmark

5.5 Determining the accuracy of an experiment

Literature review

The classification of benchmarks proposed in this book is based on [7]. The book contains also more important information on how to conduct a proper benchmarking experiment. The classification of the workload models is inspired by the book [8]. Several books and works are devoted to the workload characterisation problem. A recent survey on modern scientific techniques for workload characterisation is [3]. The book [5] is an important contribution for understanding the workload characterisation problem and address it in a practical way. Specifically, the book explains how to statistically characterise a workload from sample data. There are also several tools that help in the workload characterisation task. Among these, we point out PhFit [6] which allows the engineer to fit the inter-arrival times (assumed to be independent) into a phase-type random variable and KPC-Toolbox [4] that takes into account also the possible auto-correlation of the data.

Bibliography

[1] S. Balsamo, V. de Nitto Persone, and R. Onvural. *Analysis of Queueing Networks with Blocking*. International Series in Operations Research & Management Science. Springer, 2001.

[2] P. J. Burke. The output of a queuing system. *Operations Research*, 4(6):699–704, 1956.

[3] M.C. Calzarossa, L. Massari, and D. Tessera. Workload Characterization: A Survey Revisited. *ACM Comput. Surv.*, 48(3):48:1–48:43, 2016.

[4] G. Casale, E. Z. Zhang, and E. Smirni. KPC-Toolbox: Best recipes for automatic trace fitting using Markovian Arrival Processes. *Perform. Eval.*, 67(9):873–896, 2010.

[5] D. G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, `https://www.cse.huji.ac.il/~feit/wlmod/`, 2014.

[6] András Horváth and Miklós Telek. Phfit: A general phase-type fitting tool. In *Proceedings of Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings*, pages 82–91, 2002.

[7] D. Lilja. *Measuring Computer Performance: A Practitioner's guide*. Cambridge University Press, 2008.

[8] D. Menasce, V. A. F. Almeida, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, 2004.