Master's Degree programme
in Computer Science

Final Thesis

A RCAT application and operational analysis on a fetching queueing network

Supervisor
Prof. Andrea Marin

Graduand
Diletta Olliaro
Matriculation number 855957

Academic Year
2020/2021

To my Mum and Sisters.

Acknowledgements

First of all, I would like to thank, my supervisor: Professor Andrea Marin, to whom I express my deepest sense of gratitude for having introduced me to this fascinating research field and for guiding me throughout this thesis work with patience and dedication.

I thank my favourite women in the world, the homeland of my heart: my mother and sisters. Firstly, I thank my mother, if it were not for her I would not even be studying computer science so, at this point, not only she gave me life but also a purpose and a passion to live for. Then, I thank Martina who brought me to my bachelor open day and who always had my back and has been the best guidance and bigger sister I could ask for. Finally, but not less important, I thank my little one: Carola, who has never stopped cheering me up and that, when needed, has experienced the pleasure of being my debugging rubber duck through my whole university years.

A special thought goes to the brother I wish I had, however we were lucky enough to find each other anyway: Davide, for the years of friendship and because he never stopped believing in me, not even for a second.

I wish to express my sincerest appreciation to Mauro, Gianluca and Giacomo, who have stood by me every step of the way, listening to me and being my most honest and supportive friends.

Last, but not least, I would love to thank the amazing people I have the privilege to call my friends: Rossana, Martina D.Z., Cecilia, Gianpietro, Lorenzo, Giulio, Fabio, Daniele, Shadow. Thank you for being fundamental in my academic path and the best friends I have ever had. We have met in the wonderful context I consider Ca' Foscari University to be and hopefully we will be part of each others lives for the years to come.

Thank you, I love you.

Abstract

In queueing networks, product-form solutions are of fundamental importance to retrieve performance metrics in complex models of multiprogrammed and time-shared computer systems. The product-form property states that the steady-state probabilities of the joint process can be expressed as the normalized product of the steady-state probabilities of its interacting components. This means that product-forms solutions allow us to factorize the system in different components, then studying each of them as they were isolated but also parametrizing and taking into account the effect of the whole system on each single component. Product-form solutions often lead to a complex and time-consuming computations as they require to verify balance equations to be proved. For this reason, researchers have been trying to apply different methods to derive the same results. The methodology we focused on is the one introduced by P.G. Harrison in [1] i.e., the Reversed Compound Agent Theorem, then extended by P.G. Harrison and A. Marin in [2, 3]. In this thesis, we started from the model proposed by D. Gates and M. Westcott in [4] for the replacement of train wheels and we transposed it in the context of queueing networks. From a practical point of view, we designed a discrete-event simulation representing the theoretical model, we validated the produced results and we conducted a sensitivity analysis in order to test the robustness of the model with respect to certain assumptions. Whereas, from a theoretical perspective we used the Reversed Compound Agent Theorem to retrieve the same result obtained by the authors in [4], without using the properties of dynamically reversed processes.

Contents

Introduction

In this thesis, we present a RCAT application and an operational analysis on a particular queueing network model. Starting from the model designed by D. Gates and M. Westcott in [4] for the replacement of train wheels we decided to transpose it into the computing systems context. We aim to provide an alternative path to prove the product-form solution of the equilibrium distribution of this model, possibly simplifying the existing one through the use of the Reversed Compound Agent Theorem (RCAT), introduced by P. G. Harrison in [1] and extended by P.G. Harrison and A. Marin in [2, 3].

Queueing networks are a mathematical model used to describe a wide range of real-world systems, for example, we can think about a car washing service, a post office but also about those computing systems structured in several components that communicate and coordinate in order to provide a service to a final user, these are commonly known as distributed systems. These models find a wide application in the analyses of computer systems since the pioneering work of L. Kleinrock [5, 6] who used these models to study and design the first packet-switching network.

In traditional queueing network models, jobs move along a set of stations and compete for the service resources they offer. This class of models became popular thanks to the product-form results that allow for an efficient solution of the stochastic process underlying the model. However, in many other contexts, such as manufacturing plants, the important goal is to maintain the servers busy and the jobs may be moved among the stations to this aim. Those networks, namely fetching queuing networks, are challenging to study and only few product-form results are known.

This mathematical model can be described considering the rules underlying the arrival of jobs or customers at a certain system, requiring a particular service. To obtain this service, jobs or customers may have to go through one or more of the basic units of this model: a queue. We say a customer arrives at a queue and waits for a certain period of time, after which he obtains the desired service and leaves that particular queue or the whole system. Processes describing arrivals and departures are subject to uncertainty, consequently, they are defined through stochastic processes.

We are particularly interested in stochastic models with underlying continuous-time Markov processes defined in a discrete state space. This is a special stochastic process that enjoys the so-called Markov property, meaning that the future behaviour of the process only depends upon the current state and not at all on the history of the previous states of the process itself. Mathematically, this translates into the fact that the process will change according to an exponential random variable. Specifying processes in a discrete state space means that the set of states i.e., the set of possible values that the random variable can assume, is finite or enumerable.

Queueing theory gives us the mathematical means for studying this type of systems, allowing us to retrieve some quantities of interest such as throughput, response time, etc. in order to achieve an accurate performance evaluation of the system itself. One of the results we are mostly interested in is given by the product-form solution for the equilibrium distribution of our systems. Product-form expressions are a powerful analytical tool that allows us to examine the different components of a system in isolation with the aim of obtaining exact analytical results for quantities of interest. Product-form solutions often lead to complex and time-consuming computations, this happens because their proof requires to verify global balance equations. The latter task becomes more and more difficult to tackle as the complexity of the system grows. This is why researchers have been trying to apply different methods to derive the same results.

As already anticipated, the approach we focused on is the one proposed by P.G. Harrison in [1], i.e., the Reversed Compound Agent Theorem that enables us, through the use of Stochastic Process Algebra (SPA), to model in a simpler way the interaction in a complex system among a finite set of components. SPAs provide several formalisms to describe in a precise and rigorous manner the behaviour of systems composed of interacting components. In [1], the author used the Performance Evaluation Process Algebra (PEPA, i.e., a SPA designed for modelling computer and communication systems by J. Hillston in [7]) syntax to derive a methodology to identify reversed processes from which product-form solutions can be immediately defined. A reversed process, as defined by F.P. Kelly in [8], is a process in which the state space is the same as the original one but the direction of time is reversed. Intuitively, it is as if we took a registration of the process behaviour, and then we looked at it backwards. Constructing reversed processes from transition graphs designed with the PEPA modelling technique, we are able to guess possible transition rates for the reversed process, which we will call reversed rates. The result will be a system of non-linear equations that uniquely determines the reversed rates. We are able to do so only if some structural conditions defined in RCAT are fulfilled and, if they actually are, we know that a product-form exists and that it is given by the stationary distribution of the single queues involved in which the unknown rates of the synchronizing actions[1] are replaced by the reversed rates of the corresponding synchronizing transitions.

Another important tool that we used in this thesis is the one provided by stochastic simulations. The majority of real-world systems are too complex for being evaluated analytically. Accordingly, we use stochastic simulations to evaluate models, collecting data in order to estimate the quantities of interest of the model. Even if a simulated model has already corresponding analytical results, by this mean we gain an in-depth view of the dynamic behaviour of the system and we can test the robustness of the model with respect to the violation of the assumptions required for its analytical tractability. Moreover, simulations provide us with an insight into the scalability of a system, as a matter of fact, we can observe the deterioration or the improvement of the system performance under specific inputs. Briefly, a stochastic simulation simulates the entities, and their respective behaviour, of a particular system, this is done using random variables that can properly represent the evolution of the system under study. We are particularly interested in simulations that are asymptotically stationary, meaning that after some simulated time they reach a point in which the collected statistics are independent of the time in which we collect them. In this way, the gathered statistics and the estimates computed on them will be reliable and non-dependent on the initial state that will almost surely cause some bias in the resulting analysis.

Contribution

In this thesis, we started from the model proposed by D. Gates and M. Westcott in [4] for the replacement of train wheels and we transposed it in the context of queueing networks.

Firstly, we designed a discrete-event simulation representing the above model. In order to do so, we used the Python library SimPy v. 4.0.2 which is actually a process-based discrete-event simulation framework. Secondly, we needed to be sure that the produced results were not biased towards the initial state set for the system; in order to do so we applied Welch graphical method and we detected the so-called warm-up period. The warm-up period is the initial period of a simulation execution, in which the collected statistics still depend upon the initial settings. A correctly specified warm-up period helps us to eliminate the initial bias of the gathered data; this is why, once we identified it, we performed our simulation runs starting to collect statistics after this period. Thirdly, we have to verify that our

[1]Synchronizing actions are those actions causing more than one change of state in different components of the system.

simulation is an accurate representation of the given model, to achieve this we computed confidence intervals on the gathered statistics and we compared them with the theoretical results presented by D. Gates and M. Westcott in [4]. For all the computed confidence intervals we checked that these theoretical results did belong to the intervals and since this was true we considered our simulation validated and reliable. Finally, we performed a sensitivity analysis to check the robustness of the model with respect to the exponential assumptions.

After this first approach to the model, we switched on the theoretical side and started to investigate the possibility of reproducing D. Gates and M. Westcott product-form using the Reversed Compound Agent Theorem instead of employing the properties of dynamically reversed processes. A dynamically reversed Markov process is one that is statistically indistinguishable from the reversed process modulo a certain renaming of the states. To handle this type of processes is a particularly challenging task as it requires to guess the rates of the reversed process and also the renaming of the states. The principal advantages of the RCAT methodology are the potential for simplification of the proof of the model product-form and of the deeper understanding of the underlying process of the model itself.

Structure of the Thesis

The thesis is organized as follows:

- in Chapter 1 we provide to the reader the necessary concepts of statistics which are used throughout this thesis;
- in Chapter 2 we present some basic notion on queueing theory;
- in Chapter 3 we illustrate the theoretical background needed to develop a discrete event simulation;
- in Chapter 4 we explain in detail the Reversed Compound Agent Theorem;
- in Chapter 5 we show our work starting from our theoretical approach and finally presenting our simulation program, its validation, and the sensitivity analysis;
- *Conclusions* completes this thesis and proposes some possible future work.

Chapter 1

Statistical background

In this chapter, we are going to analyse and describe the fundamental theoretical concepts used as basis for this thesis, we will present some essential elements of statistics [9] and of stochastic processes [8, 10]. We start from the concept of random variables, after a brief introduction we will present the most important families of discrete and continuous distributions. Then we will present in detail the concept of confidence interval and finally, we will describe Markov processes and the reversibility property.

1.1 Random Variables

A *random variable* is a function of an outcome of an experiment, meaning the elementary result of a random phenomenon. It can be expressed as

$$X = f(\omega).$$

The domain of a random variable is the sample space Ω whereas its range is given by the numerical set of possible values that the random variable can potentially take (e.g. real numbers $\mathbb{R}$, integers $\mathbb{Z}$, interval $(0, 1)$, etc.).
A random variable can be *discrete* if its range is finite or countable or *continuous* if it assumes a whole interval of values, this interval may be both bounded or unbounded and it can also be a union of several intervals.
The *probability distribution* for a random variable describes how the probabilities are distributed over the values of the range. For a discrete random variable, the probability distribution is defined by the probability mass function (pmf):

$$P(x) = \boldsymbol{P}\{X = x\}.$$

Then, we can also define the cumulative distribution function (cdf), defined as

$$F(x) = \boldsymbol{P}\{X \leq x\} = \sum_{y \leq x} \boldsymbol{P}(y).$$

We define the function above because in the case of a continuous random variable it is not meaningful to talk about the probability that a random variable will take a specific value; for this reason, we consider the probability that it will lie in a given interval. Thus, to define the counterpart of the pmf, we talk about the probability density function (pdf), which is defined as the derivative of the cdf, whenever it exists, $f(x) = F'(x)$ and we can write:

$$\boldsymbol{P}\{a < X < b\} = \int_a^b f(x)dx$$

as it provides the probability that the variable will take on a value within that interval.
We are usually interested in some specific measures computed on the values that random variables may assume. Among others, the measures we are going to use the most are the expected value or mean, the variance and the standard deviation. The first measure is basically a weighted average of the values the random variable may take, in the discrete case the weights are provided by the probability mass function while in the continuous one by the probability density function. Mean is usually denoted by the symbols $\mathbf{E}(X)$ or μ and the formulas to compute it are as follows:

$$\mathbf{E}(X) = \sum_x xP(x) \qquad \mathbf{E}(X) = \int xf(x)\ dx$$

for the discrete and the continuous case, respectively.

The variance (indicated as $\mathrm{Var}(X)$ or σ^2) is the expectation of the squared deviation of a random variable from its mean, mathematically it can be expressed as: $\mathrm{Var}(X) = \mathbf{E}(X - \mu)^2$. Variance is a very interesting measure as it indicates the spread between numbers in a dataset, however, we should notice that because of the squaring in the formula the variance is not in the same unit of measurement of the original data. For this reason, statisticians usually prefer to work with the standard deviation (denoted by σ) as it is the squared root of the variance, in this way the original unit of measurement is restored consequently standard deviation is easier to interpret. In fact, standard deviation is a measure of the amount of variation in a set of values, in practice a low standard deviation indicates that the values in the set tend to be close to the mean of the set, whereas an high one can be interpreted as a tendency to have values distant from the mean.
In the following, we are going to introduce some families of discrete and continuous distributions that are the ones most important for our purposes, in practice a family of distributions is a mathematical model that can adequately describe a wide range of different phenomena.

1.1.1 Bernoulli distribution

This is the simplest random variable as it can assume only two values. In practice, it describes experiments allowing only binary outcomes. The probability of success p is the parameter of the Bernoulli distribution. The idea is that we assume a binary outcome of an experiment and we associate a successful outcome to 1, which will have probability p, and a failure to 0, which will have probability q, with $q = (1-p)$. Accordingly, we will have:

$$P(x) = \begin{cases} q = p - 1 & \text{if } x = 0 \\ p & \text{if } x = 1 \end{cases} \qquad \mathbf{E}(X) = p \qquad \mathrm{Var}(X) = pq$$

1.1.2 Binomial distribution

Binomial distribution is used to describe the number of successes in a sequence of independent Bernoulli experiments. Parameter are given by n, the number of trials, and p, the probability of success. The characterising formulas of this distribution are as follows:

$$P(x) = \binom{n}{x} p^x q^{n-x} \qquad \mathbf{E}(X) = np \qquad \mathrm{Var}(x) = npq$$

$P(x)$ describes the probability of having exactly x success over n trials. Notice that each Bernoulli trial is associated with a Bernoulli variable that takes on value 0 in case of failure, and 1 in case of success. Consequently, the sum of these variables represents the number of successes; which means that any Binomial variable can be represented as the sum of independent Bernoulli variables.

1.1.3 Poisson distribution

This discrete distribution is extremely related with the concept of *rare* events. This means that two such events have an extremely low probability of occurring simultaneously or within a short period of time. The number of rare events occurring by a large population of independent individuals within a fixed period of time has Poisson distribution. A Poisson variable can assume any non-negative value as a matter of fact it can take value zero since there may not be any rare event in the considered period but also any positive integer as there is no limit in the number of rare events that there may be in that period. This distribution has only one parameter that we denote with $\lambda > 0$ and that represents the average number of rare events. Formulas of our measures of interest are given below:

$$P(x) = e^{-\lambda}\frac{\lambda^x}{x!},\ x = 0, 1, 2, \dots \qquad \mathbf{E}(X) = \lambda \qquad \mathrm{Var}(X) = \lambda$$

1.1.4 Uniform distribution

This continuous distribution describes an experiment that has a random outcome inside a specified interval. The extremes of this interval are the parameters of the distribution and they are denoted by a and b, for the minimum and the maximum values of the interval, respectively. This distribution assures equal preference to all values as a matter of fact it presents what is called the *uniform property*: the probability is only determined by the length of the interval. In fact, the characterising formulas of the distribution are the following:

$$f(x) = \frac{1}{b-a},\ a < x < b \qquad \mathbf{E}(X) = \frac{a+b}{2} \qquad \mathrm{Var}(X) = \frac{(b-a)^2}{12}$$

we can notice how they only depend on the width of the interval rather than on its location.

1.1.5 Exponential distribution

This continuous distribution is often used to model time. The parameter of the distribution is denoted again by λ and it stands for a frequency parameter, in practice it indicates the number of events per time unit. The formulas describing this distribution are the following:

$$f(x) = \lambda e^{-\lambda x},\ x > 0 \qquad \mathbf{E}(X) = \frac{1}{\lambda} \qquad \mathrm{Var}(X) = \frac{1}{\lambda^2}.$$

An appealing property of the exponential distribution is the *memoryless property.* This property states that a given probability distribution is independent of its history, for this reason if we interpret the values assumed by the random variables as times, any time may be considered as a starting point since past history does not help us understand present and future behaviour. Memoryless property formalise the fact that if something has or has not already happened, this does not actually help us to predict the future. Mathematically, consider an exponential variable T representing waiting time and consider t as the already elapsed portion of waiting time and x as the additional remaining time, for the memoryless property we have:

$$\boldsymbol{P}\{T > t + x | T > t\} = \boldsymbol{P}\{T > x\} \text{ for } t, x > 0.$$

This can be actually quite helpful since if observing a phenomenon in continuous time we realise it is memoryless then we immediately know it is described by an exponential random variable.

1.1.6 Normal distribution

The Normal distribution is likely the most important probability distribution in statistics since it has been found out it fits many natural phenomena such as blood pressure, measurement errors, etc. The two parameters describing it are the mean, which defines the location of the peak of the bell, and the standard deviation, which defines the width of the bell shape. Most values aggregate around the mean, whereas values further away from it stand on the so-called tails of the distribution. The formula characterising its pdf is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-(x-\mu)^2}{2\sigma^2}\right\}, \; -\infty < x < \infty$$

and its mean and variance are given by:

$$\mathbf{E}(X) = \mu \qquad \mathrm{Var}(X) = \sigma^2$$

The importance of this distribution mostly derives from the Central Limit Theorem (CLT). CLT states that if we have a population with mean μ and standard deviation σ, then the distribution of the sample means will be approximately normally distributed, provided that the sample size is sufficiently large (usually larger or equal to 30). This holds assuming that all samples are identical in size and regardless of the population distribution shape.

1.2 Confidence Intervals

An estimator is a sample statistic that estimates a population parameter, for example the sample mean is an estimator of the population mean. When providing an estimator we need to quantify its accuracy, as we know for sure we will estimate this sample statistic up to some error. This is because an estimator is computed on a sample, implying a certain amount of randomness in the measurements, rather than on the entire population, since the latter option is often not possible.
Confidence intervals are needed to quantify this amount of uncertainty and play a pivotal role in the validation of stochastic simulations. Usually, several executions of the simulation are run and from their results we estimate a population parameter, from which we build a confidence interval. Then, if the corresponding theoretical value is within the confidence interval computed from the simulation, the latter is considered to be valid for the parameter under analysis.
Notice that there exists a confidence interval for every summarized quantity such as median, mean, quartiles, standard deviation, probability of success, etc. In the following we are going to give precisely the definition of confidence interval, then we will present a general method to build it and finally we will describe the most used confidence intervals i.e. the ones computed for the mean.

Definition 1.1. An interval $[a, b]$ is a $(1-\alpha)100\%$ confidence interval for a certain parameter θ if it contains the parameter with probability $(1-\alpha)$, i.e.

$$\boldsymbol{P}\{a \leq \theta \leq b\} = 1 - \alpha.$$

The coverage probability $(1-\alpha)$ is also called a confidence level.

In practice an $(1-\alpha)100\%$ confidence interval is a range of values that we can be $(1-\alpha)100\%$ confident it contains our parameter; this does not mean that the confidence interval will contain the parameter with probability $(1-\alpha)100\%$, the parameter will either belong (with probability 1) or not to the interval. In fact, the confidence is in the method and not in a particular confidence interval; this indicates that if we repeat a certain sampling method several times, roughly the $(1-\alpha)100\%$ will cover the true population parameter, whereas the others will not.

The simplest method to construct a confidence interval requires a consistent[1] estimator $\hat{\theta}$ with an asymptotically normal distribution. This results to be true for sufficiently large sample sizes thanks to the central limit theorem discussed above. Then, a confidence interval of approximate level $(1-\alpha)$ for θ is

$$[a,b] = \hat{\theta} \pm z_{\alpha/2} \cdot \sigma(\hat{\theta}) \tag{1.1}$$

with $z_{\alpha/2}$ defined as $\boldsymbol{P}\{Z > z_{\alpha/2}\} = \alpha/2$. In the expression above Z is the standard normal variable i.e. the normal variable with parameters $\mu = 0$ and $\sigma = 1$. In fact, the previous confidence interval is built from the Z statistic

$$Z = \frac{\hat{\theta} - \theta}{\mathrm{SE}(\hat{\theta})}$$

under the assumed settings, Z is asymptotically distributed as a standard normal variable, so we have

$$\Pr\{-z_{\alpha/2} \le Z \le z_{\alpha/2}\} \approx 1 - \alpha.$$

To obtain the confidence interval showed in [1.1], we have:

$$\begin{aligned} 1-\alpha &\approx \Pr\{-z_{\alpha/2} \le Z \le z_{\alpha/2}\} \\ &= \Pr\left\{ -z_{\alpha/2} \le \frac{\hat{\theta}-\theta}{\mathrm{SE}(\hat{\theta})} \le z_{\alpha/2} \right\} \\ &= \Pr\{-z_{\alpha/2}\mathrm{SE}(\hat{\theta}) \le \hat{\theta} - \theta \le z_{\alpha/2}\mathrm{SE}(\hat{\theta})\} \\ &= \Pr\{\hat{\theta} - z_{\alpha/2}\mathrm{SE}(\hat{\theta}) \le \theta \le \hat{\theta} + z_{\alpha/2}\mathrm{SE}(\hat{\theta})\} \end{aligned}$$

Hence, a confidence interval of approximate level $(1-\alpha)$ is $[a,b]$ with

$$a = \hat{\theta} - z_{\alpha/2}\mathrm{SE}(\hat{\theta}) \text{ and } b = \hat{\theta} + z_{\alpha/2}\mathrm{SE}(\hat{\theta})$$

or in compact form $\hat{\theta} \pm z_{\alpha/2}\mathrm{SE}(\hat{\theta})$ as we stated above. What we have just presented will result more clear looking at figure 1.1.

[1]Consistency means that, as the sample size increases, the sampling distribution of the estimator becomes increasingly concentrated at the the true parameter value.

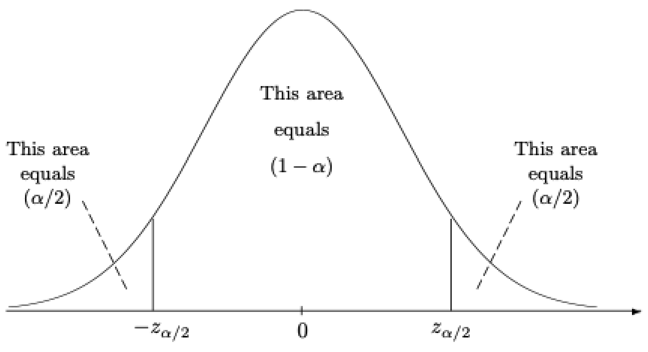

This area equals $(\alpha/2)$
This area equals $(1-\alpha)$
This area equals $(\alpha/2)$
$-z_{\alpha/2}$
0
$z_{\alpha/2}$

Figure 1.1: Standard normal quantiles $\pm z_{\alpha/2}$ and partition of the area under the density curve. This figure is taken from [9] page 249.

From the consideration done so far we can realize that the width of the confidence interval will decrease if the sample size increases and it will increase as standard deviation and confidence level increases.

Now, we focus in particular on confidence intervals constructed for the population mean. We know that the sample mean (denoted by $\bar{X}$) is a suitable estimator for the population mean; sample mean is a random variable and as other random variables it has a mean $\mathbf{E}(\bar{X})$ and a standard deviation $\sigma(\bar{X})$. Suppose random samples of size n are drawn from a population with mean μ and standard deviation σ it can be proved that the mean and the standard deviation of the sample mean satisfy the following relationships:

$$\mathbf{E}(\bar{X}) = \mu \text{ and } \sigma(\bar{X}) = \frac{\sigma}{\sqrt{n}}.$$

Accordingly, following the rule given by equation 1.1 we can write a confidence interval for the mean as:

$$\bar{X} \pm z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}.$$

Until now we have assumed the population standard deviation is known, unfortunately, this is rarely the case. For this reason, the standard deviation of a population is often estimated from a random sample drawn from the original population. This is called the sample standard deviation and it is defined as

$$s = \sqrt{\frac{\sum_{i=0}^{n}(x_i - \bar{X})^2}{n-1}}$$

where the $x_i,\ i = 0, \ldots, n$ are the realizations of the random variable X. It is important to underline the fact that everything we said holds if data follows a standard normal distribution or if the sample size is big enough for the Central

Limit Theorem to be applied. Otherwise, there are other random variables that can be used; in particular, if the variance in the data is unknown and sample size is smaller than 30 we can use the Student's t-distribution. This distribution is symmetric ans bell-shaped as the normal one, however it presents heavier tails, meaning that it is more likely to produce values far from its mean.

1.3 Stochastic Processes

Many physical systems, such as computers and communication networks, evolve in time. Stochastic processes provide the mathematical models for these systems, representing the sequences of states they enter as discussed in [8]. Formally,

Definition 1.2. a stochastic process X is a collection of random variables $X = \{X_t,\ t \in T\}$, indexed by $t \in T$ (they can also be denoted by the notation $X(t)$).

We are particularly interested in stochastic processes indexed by time and we distinguish between:

- Continuous-time processes, meaning that $T \subset \mathbb{R}$ and usually $T = [0, \infty)$.
- Discrete-time processes, meaning that $T \subset \mathbb{Z}$ and usually $T = \{0, 1, 2, \dots\}$.

We consider stochastic processes in which the support of the random variable X_t, i.e., the set of possible values it can assume, is the same for all $t \in T$. This is called the state space of the process and also here we can divide them in:

- Continuous state space, if X_t is a continuous random variable for each $t \in T$.
- Discrete state space, if X_t is a discrete random variable for each $t \in T$.

The state space of a stochastic process is usually denoted by $\mathcal{S}$ and each value composing it is called a state of the process. In practice, we will say that each $X(t)$ is the state of some system at time t and accordingly, to this interpretation we say that the system is in state x_i if at time t if $X(t) = x_i$. One of the properties that a stochastic process may or may not have is the one of stationarity.

Definition 1.3. A stochastic process is said to be stationary if the properties of the random variables remain the same over time. More precisely, a stochastic process is stationary if $X(t_0), X(t_1), \dots, X(t_n)$ and $X(t_0+\tau), X(t_1+\tau), \dots, X(t_n+\tau)$ have the same distribution for all $t_0, t_1, \dots, t_n, \tau \in T$.

In practice, this means that if a process X is stationary, shifting it by τ does not affect its distribution.

In the following, we are going to present a particular type of stochastic processes which is Markov processes, in particular we will focus on the simplest instances of this kind of processes that are birth-death and Poisson processes. Finally, we will present a property of particular interest for our work, that is reversibility.

1.3.1 Markov processes

A Markov process is a particular type of stochastic process that enjoys the memoryless property. In this context, this means that given a Markov process its future probabilities only depend upon the latest observed state and not on its previous history. Formally,

$$\begin{aligned}&\boldsymbol{P}\{X(t_{n+1}) = x_{n+1}|X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n\}\\ &\quad = \boldsymbol{P}\{X(t_{n+1}) = x_{n+1}|X(t_n) = x_n\} \quad \text{with } t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1}\end{aligned}$$

We distinguish between Markov processes, which definition is described above, and Markov chains that are the discrete-space version of Markov processes.

A Markov process is said to be time-homogeneous if the observed behaviour is independent of the epoch it is observed. Another interesting property is the one of irreducibility, we say that a Markov chain is irreducible if from any state we can reach any other state of the process by following a sequence of transitions. This means that if we draw the state transition diagram of a Markov chain, i.e., a graph where nodes represent the states of the process and the arcs stand for the transitions among the states themselves, for the process to be irreducible we want this graph to be strongly connected.
We say a particular state in a Markov process is *recurrent* if the probability that the process will return to that state is 1. If the expected number of steps required to return there, is less then infinity, then the state is said to be positive-recurrent. If all the states of the process are positive-recurrent and there is no periodicity, we say the Markov process enjoys the ergodicity property.
We now describe some quantities of interest. For a discrete time Markov chain, we can define the transition probability p_{ij} between two states x_i and x_j and we define it as

$$p_{ij} = \boldsymbol{P}\{X(t_n + 1) = x_j|X(t_n) = x_i\}$$

In practice, this is the probability that given that a transition out of state x_i will occur, it will be towards state x_j. For continuous time Markov chain we define the so called instantaneous transition rates between two states x_i and x_j to be:

$$q_{ij} = \lim_{\tau \to 0} \frac{\boldsymbol{P}\{X(t+\tau) = x_j|X(t) = x_i\}}{\tau} \qquad i \neq j$$

A Markov process remains in a state x_i for a period of time that is exponentially distributed with parameter the so called exit rate, defined as

$$q_i = \sum_{k \in \mathcal{S}, k \neq i} q_{ik}.$$

From this we can derive another formula for the probability with which the process will move from state x_i to another state x_j, that is:

$$p_{ij} = \frac{q_{ij}}{q_i}.$$

Each of these probabilities is the conditional probability of transitioning from state i into state j; each one is an element of the one-step transition probability matrix of the so called *embedded Markov chain* which is a regular discrete time Markov chain determined by the conditional probabilities defined above.
A process that satisfies the time homogeneous and irreducibility assumptions may possess an equilibrium distribution that is a collection of positive numbers $\pi(x_i)$, with x_i belonging to the state space (denoted by $\mathcal{S}$), summing to unity that satisfies the so called equilibrium equations, consequently for discrete time Markov chain we have:

$$\pi(x_i) = \sum_{k \in \mathcal{S}} \pi(x_k) p_{ki} \qquad i \in \mathcal{S}$$

If we can find such a collection of positive numbers and the process is ergodic then this collection can be normalised to produce an equilibrium distribution. When an equilibrium distribution exists it is unique and if

$$\lim_{n \to \infty} \boldsymbol{P}\{X(t_n) = x_i | X(t_0) = x_j\} = \pi(x_i)$$

then $\pi(x_i)$ is also the limiting distribution; this means that no matter which is the initial distribution the distribution over states converges to π as the time goes to infinity. Moreover, the proportion of time the process spends in state x_i converges to $\pi(x_i)$ as time goes to infinity. For ergodic processes the limiting or equilibrium distribution is also called *steady-state distribution.*
For continuous time Markov chain the equilibrium equations are given by:

$$\pi(x_i) \sum_{k \in \mathcal{S}} q_{ik} = \sum_{k \in \mathcal{S}} \pi(x_k) q_{ki} \qquad i \in \mathcal{S}$$

and an equilibrium distribution is a collection of positive numbers $\pi(x_i)$, $i \in \mathcal{S}$, summing to unity which satisfy the equilibrium equations. As for the discrete time case an equilibrium distribution is unique if it exists and then it is both the limiting and the stationary distribution. Notice also that the set of equations that characterize the equilibrium distribution of a Markov chain (when such a distribution exists) is called *global balance equations* (GBE).

1.3.2 Poisson processes

We now describe a particular type of Markov process. To deeply understand the nature of Poisson processes we first need to define what a counting process is.

Definition 1.4. A stochastic process $X(t)$ is said to be a counting process if:

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

Counting processes are actually concerned with counting the number of occurrences of something over time, for example the number of arrivals at a queue over a certain time interval. In fact, given two time two time epochs $t_1, t_2 \in T$ with $t_1 < t_2$ we have that $X(t_2) - X(t_1)$ is the number of events occurred in the interval $(t_1, t_2]$. If the number of events occurring in disjoint time intervals is independent then the counting process is said to possess *independent intervals.* Moreover, if the distribution of the number of events depends only on the length of the time interval then these increments are also said to be *stationary increments.*
Poisson processes are a special type of counting process enjoying some important properties. Notice that from now on we will use the term Poisson processes to refer to time-homogeneous Poisson processes, meaning that they possess independent and stationary increments. We now give three definitions of Poisson process, these three definitions can be proved to be equivalent but they highlight some important aspects of this process.

Definition 1.5. A Poisson process $X(t)$ with rate $\lambda \in \mathbb{R}$ is a continuous time counting process (meaning that $t \in \mathbb{R}$) that satisfies the following properties:

1. $X(0) = 0$.
2. for any pair of disjoint 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, formally:
$$\boldsymbol{P}\{X(t+s) - X(t) = x\} = \frac{(\lambda s)^x \mathrm{e}^{-\lambda s}}{x!}$$
which is exactly the probability distribution of a Poisson random variable with mean λs.

Notice that there is no t in the formula above and this is because, as we anticipated before, the number of events occurring in an interval depends only on the length of the interval itself and not on where it starts. Accordingly, if we observe a sufficiently large interval we will observe a probability distribution concentrated around the mean.

Definition 1.6. A Poisson process $X(t)$ with rate $\lambda \in \mathbb{R}^+$ 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. $\boldsymbol{P}\{X(h) = 1\} = \lambda h + o(h)$, where λ is the intensity of the process, h is the width of the interval[2]. This means that taking an interval of small width h the probability of observing one event is approximately λh.

4. $\boldsymbol{P}\{X(h) \geq 2\} = o(h)$. This means that the probability of observing two or more events in the same small interval is very low (lower than the width of the interval). In fact, we can consider that the probability of observing two simultaneous events is 0.

Definition 1.7. A Poisson process $X(t)$ with rate $\lambda \in \mathbb{R}^+$ is a continuous time counting process that satisfies the following property:

1. $X(0) = 0$.

2. Let $S_i = \inf\{t : X(t) \geq i\}$ be the time epoch in which the process jumps from $i-1$ to i, with $i = 1, 2, 3, \ldots$ and $S_0 = 0$. Then $Y_i = S_{i+1} - S_i$ are independent and identically distributed (i.i.d) exponential random variables with rate λ. This means that the time between one increment and the following one is described by an i.i.d exponential random variable with parameter λ. Formally we have:
$$S_i \approx \exp(\lambda).$$

From these definitions, we can notice some appealing properties of Poisson processes. Firstly, we have that
$$\mathbf{E}[X(s)] = \lambda s$$
where λ is the number of events per unit of time, e.g. we can interpret it as the number of arrivals per second and s is the length of the observed interval of time. Secondly, it has only one parameter which means that, for example, we only need one value to describe the workload, moreover λ is also the parameter of the exponential distribution modelling inter-arrival time; in fact, it can be proved

[2]We say that a function f is $o(h)$ if
$$\lim_{h \to 0} \frac{f(h)}{h} = 0$$
and when writing $o(h)$ we mean that this quantity can be ignored since it grows as fast as h.

that in a sequence of rare events, when the number of events is described by a Poisson process, the time between events is exponential. Finally, we have a linear relationship between the number of arrivals and the length of the interval.

Other than this, we have other two very important properties of this type of processes: superposition and splitting.

Superposition. Let $X_1(t)$ and $X_2(t)$ be two independent Poisson processes with intensities λ_1 and λ_2, respectively. Then, process $Y(t) = X_1(t) + X_2(t)$ defined as the superposition of the two Poisson processes is still a Poisson process with intensity $\lambda_1 + \lambda_2$.

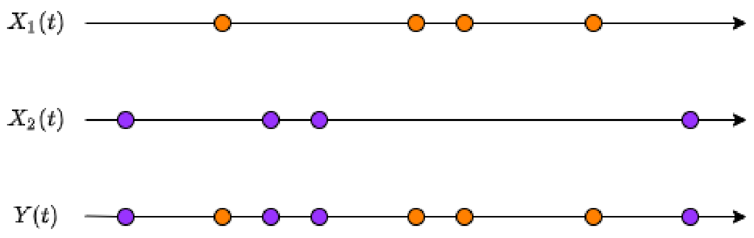


Figure 1.2: Superposition property of Poisson processes.

Splitting. Let $X(t)$ be a Poisson process and $Y_1(t)$ be the process obtained by keeping each arrival event with probability p and by discarding it with probability $(1-p)$ and $Y_2(t)$ being the counting process of the discarded arrivals. Assume that these choices are independent of the original process. Then $Y_1(t)$ and $Y_2(t)$ are independent Poisson processes with intensity λp and $\lambda(1-p)$, respectively.

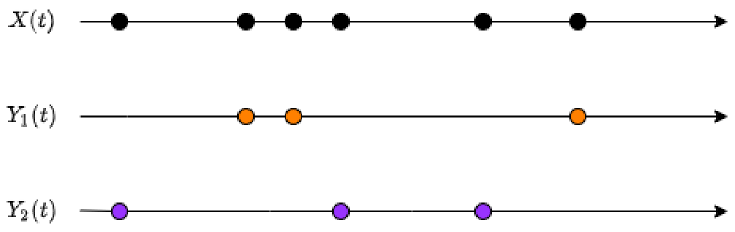


Figure 1.3: Splitting property of Poisson processes.

1.3.3 The Law of Rare Events

The following theorem basically states that when the size of the population, n, is very large and the probability p of a certain event is very small then the binomial random variable with parameters n and p can be approximated by a Poisson random variable.

Theorem 1.1 (The Law of Rare Events)**.** For each $n \geq 1$, let B_n be a Binomial random variable with parameters n and $p = \lambda/n$. Then, for any $k \geq 0$:

$$\lim_{n\to\infty} \boldsymbol{P}\{X_n = k\} = \mathrm{e}^{-\lambda}\frac{\lambda^k}{k!}$$

Accordingly, when n goes to infinity we have a Poisson process with parameter λ.

For example, if we have a population of n independent individuals that generate an arrival to our system rarely, meaning with low probability p. Thanks to the law of rare events, we know that if n is large enough the observed arrival process to our system is a Poisson process.

1.4 Reversibility for Markov Processes

In this section, we will study an important and very useful property of stochastic processes: reversibility. We say that a stochastic process $X(t)$ is reversible if, when the direction of time is reversed, the behaviour of the reversed process $X^R(t)$ remains the same. Formally,

Definition 1.8. A stochastic process $X(t)$ is *reversible* if $(X(t_1), X(t_2), \ldots, X(t_n))$ has the same distribution as $(X(\tau - t_1), X(\tau - t_2), \ldots, X(\tau - t_n)$ for all $t_1, t_2, \ldots, t_n, \tau \in T$

The first thing we can say about this type of processes is that every reversible process is also stationary by definition of stationarity. Moreover, we can give some necessary and sufficient conditions for reversibility in terms of equilibrium distribution and transition rates. These conditions are called *detailed balance conditions* and are presented in the theorem below.

Theorem 1.2. A stationary Markov process is reversible if and only if there exists a collection of positive numbers $\pi(x_i)$ $x_i \in \mathcal{S}$, summing to unity that satisfies the detailed balance conditions

$$\pi(x_i)q_{ij} = \pi(x_j)q_{ji} \qquad x_i, x_j \in \mathcal{S}$$

When there exists such a collection, $\pi(x_i)$ is the equilibrium distribution of the process.

In fact, a collection of positive numbers satisfying the detailed balance conditions whose sum is finite can of course be normalized to produce an equilibrium distribution. However, notice that, as we pointed out above, if the process is not

stationary then it is neither reversible even if the detailed balance conditions are satisfied.
The term $\pi(x_i)q_{ij}$ is called *probability flux* from state x_i to state x_j. The detailed balance conditions presented above, in practice, require that for each pair of states x_i, x_j, the probability flux out of state x_i to state x_j is equal to the probability flux out of state x_j to state x_i. Notice that these conditions are different with respect to the so called full balance conditions which require to satisfy the balance equations that follow:

$$\pi(x_i) \sum_{x_j \in \mathcal{S}} q_{ij} = \sum_{x_j \in \mathcal{S}} \pi(x_j) q_{ji} \qquad x_i \in \mathcal{S} \wedge i \neq j.$$

These equations basically require that the probability flux out of state x_i equals the probability flux into state x_i, moreover these equations are usually called *global balance equations.*

1.4.1 Birth-Death Processes

Birth and Death processes are a special case of continuous-time Markov processes and they are also the simplest instance of reversible processes. The state space $\mathcal{S}$ is composed by $\{0, 1, \ldots, K\}$ with K possibly infinite. State transition can be only of two types: a birth which increases the state variable by one, so have a transition from x_i to x_{i+1}, and a death which decreases the state variable by one, consequently we witness a transition from state x_i to state x_{i-1}. A birth-death process can be represented by its transition diagram as follows.

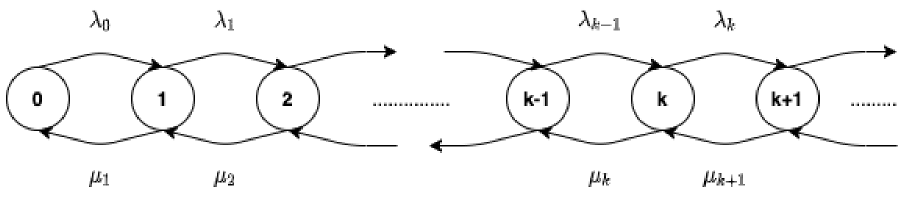


Figure 1.4: Birth-Death process state transition diagram.

The detailed balance condition state that the equilibrium distribution of a stationary birth and death process satisfies

$$\pi(x_i) q_{i,i-1} = \pi(x_{i-1}) q_{i-1,i}$$

hence it is reversible. Moreover, it can be proved that if the state transition diagram of a Markov process is a tree then the process is time reversible. From this, it follows that all Markov processes of the birth-death type are time reversible.

1.4.2 Kolmogorov's Criterion

Detailed balance conditions allow us to decide whether a stationary Markov process is reversible or not, using its equilibrium distribution and its transition rates. As the equilibrium distribution is retrieved starting from the transition rates, one could wonder if it is possible to say if a process is reversible or not directly from its transition rates, without first computing the equilibrium probabilities. Kolmogorov's criteria allow us to do exactly this.

Theorem 1.3 (Kolmogorov's Criterion)**.** Given a closed cycle $i_1, i_2, \dots, i_n, i_1$ in the state transition diagram, stationary Markov process is reversible if and only if its transition rates satisfy

$$q_{i_1 i_2} q_{i_2 i_3} \cdots q_{i_n i_1} = q_{i_1 i_n} q_{i_n i_{n-1}} \cdots q_{i_2 i_1}$$

for any finite sequence of states $i_1, \dots, i_n \in \mathcal{S}$

In practice, the product of the transition rates round the cycle needs to be the same in both directions. It can be proved that the Kolmogorov's criterion is equivalent to the detailed balance conditions consequently this criterion gives a necessary and sufficient condition for reversibility. Furthermore, since a tree-structured transition diagram does not have any cycles, Kolmogorov's criterion is always satisfied and as we have already said it follows that any Markov process represented by a tree is time reversible.

1.4.3 Reversed process and ρ-reversibility

If $X(t)$ is a reversible Markov process then $X^R(t)$ is also a Markov process since it is statistically indistinguishable from $X(t)$. We now characterize more precisely the reversed process [8, 10].

Theorem 1.4. If $X(t)$ is a stationary Markov process with transition rates q_{ij} with $x_i, x_j \in \mathcal{S}$ and equilibrium distribution $\pi(x_i)$ with $x_i \in \mathcal{S}$ then the reversed process $X^R(t)$ is a stationary Markov process with transition rates

$$q^R_{ij} = \frac{\pi(x_j) q_{ji}}{\pi(x_i)}, \qquad x_i, x_j \in \mathcal{S}$$

and the same equilibrium distribution.

The converse of the above theorem also holds.

Theorem 1.5. Let $X(t)$ be a stationary Markov process with transition rates q_{ij}, $x_i, x_j \in \mathcal{S}$ if we can find a collection of numbers q^R_{ij} $x_i, x_j \in \mathcal{S}$ such that

$$q^R_i = q_i$$

and a collection of positive numbers $\pi(x_i), x_i \in \mathcal{S}$ summing to unity such that

$$\pi(x_i)q_{ij} = \pi(x_j)q^R_{ji} \qquad x_i, x_j \in \mathcal{S} \tag{1.2}$$

then q^R_{ij} are the transition rates of the reversed process $X^R(t)$ and $\pi(x_i)$ is the equilibrium distribution of both processes.

The similarity of equations [1.2] to the detailed balance conditions should be noticed. In fact, a generalization of Kolmogorov's criterion can also be obtained and it states what follows.

Theorem 1.6. Let $X(t)$ be a stationary Markov process with transition rates q_{ij}, $x_i, x_j \in \mathcal{S}$. If there exist transition rates q^R_{ij}, $x_i, x_j \in \mathcal{S}$ such that

$$q_{i_1 i_2} q_{i_2 i_3} \cdots q_{i_n i_1} = q^R_{i_1 i_n} q^R_{i_n i_{n-1}} \cdots q^R_{i_2 i_1}$$

for any finite sequence of states $x_{i_1}, x_{i_2}, \ldots, x_{i_n} \in \mathcal{S}$ and $q^R_i = q_i$ for every state $i \in \mathcal{S}$ then q^R_{ij} are the transition rates of the reversed Markov process $X^R(t)$.

We may encounter stationary Markov processes for which the reversed process, even if not statistically indistinguishable from the original one, it would be if a *state renaming function* ρ was applied to the state space $\mathcal{S}$. The intuition behind is that we are not able to distinguish between $X(t)$ and $X^R(t)$ once that a state renaming function ρ (a bijection from $\mathcal{S}$ to $\mathcal{S}$) is applied to rename the states. In the following, we give a formal definition of what we have just presented; notice that given a Markov process $X(t)$ we denote by $\rho(X)(t)$ the same process where the state names are changed according to ρ.

Definition 1.9. Let $X(t)$ be a stationary Markov process with state space $\mathcal{S}$ and ρ be a renaming on $\mathcal{S}$. $X(t)$ is said to be $\rho-$reversible if for all $t_1, t_2, \ldots, t_n, \tau \in T$, $(X(t_1), X(t_2), \ldots, X(t_n))$ has the same stationary distribution as $(\rho(X)(\tau - t_1), \rho(X)(\tau - t_2), \ldots, \rho(X)(\tau - t_n))$

Since $X(t)$ and $X^R(t)$ have the same stationary distribution π it follows that if $X(t)$ is $\rho-$reversible then

$$\pi(x_i) = \pi(\rho(x_i)) \qquad \forall x_i \in \mathcal{S}.$$

Necessary and sufficient conditions for $\rho-$reversibility can be provided in two ways: by the solution of the linear system of $\rho-$detailed balance equations or by an extended formulation of Kolmogorov's criterion. Both alternatives are presented below.

Theorem 1.7. Given a stationary Markov process $X(t)$ with state space $\mathcal{S}$, if there exists a collection of positive real numbers π summing to unity and a bijection ρ from $\mathcal{S}$ to $\mathcal{S}$ such that

$$q_i = q_{\rho(i)} \qquad \forall x_i \in \mathcal{S} \tag{1.3}$$

$$\pi(x_i)q_{ij} = \pi(\rho(x_j))q_{\rho(j)\rho(i)} \forall x_i, x_j \in \mathcal{S}, x_i \neq x_j \tag{1.4}$$

then $X(t)$ is $\rho - reversible$ and $\pi(x_i)$ is its steady-state distribution.

Equation [1.3] imposes that the residence time of a state and its renaming must be the same. Whereas the set of equations [1.4] are the so called $\rho-$detailed balance equations; in case the renaming function ρ is known they can be used, as an alternative to the more complex global balance equations, to compute the steady-state distribution of the process. The latter consideration is basically the reason why reversibility is such an important property for stochastic processes.

Theorem 1.8. Given a stationary Markov process $X(t)$ with state space $\mathcal{S}$ and let ρ be a renaming on $\mathcal{S}$. $X(t)$ is $\rho-$reversible if and only if for any finite sequence of states $x_{i_1}, x_{i_2} \dots, x_{i_n} \in \mathcal{S}$

$$q_{i_1 i_2} q_{i_2 i_3} \cdots q_{i_n i_1} = q_{\rho(i_1)\rho(i_n)} q_{\rho(i_n)\rho(i_{n-1})} \cdots q_{\rho(i_2)\rho(i_1)}$$

and equation [1.2] holds for all $x_i \in \mathcal{S}$.

Notice that if ρ is the identity then $X(t)$ is simply reversible, whereas if ρ is an involution then we say that $X(t)$ is *dynamically reversible*. An involution is a function that is its own inverse, meaning that it returns the identity when applied to itself. In practice, this means that even if the reversible process is not statistically indistinguishable from the original one, it would be if some of the states were interchanged. In particular, to make the concept of *dynamic reversibility* more clear suppose that to each state $x_i \in \mathcal{S}$ there corresponds a conjugate state $x_i^+ \in \mathcal{S}$ with $(x_i^+)^+ = x_i$. Then process $X(t)$ is said to be dynamically reversible if it is statistically indistinguishable from $[X^R(t)]^+$. Formally,

Theorem 1.9. A stationary Markov process with $q_i = q_{i^+}$, $x_i \in \mathcal{S}$ is dynamically reversible if and only if there exist a collection of positive numbers $\pi(x_i), x_i \in \mathcal{S}$ summing to unity that satisfy

$$\pi(x_i) = \pi(x_i^+) \qquad x_i \in \mathcal{S}$$

and

$$\pi(x_i)q_{ij} = \pi(x_j^+)q_{j^+,i^+} \qquad x_i, x_j \in \mathcal{S}$$

when there exists such a collection $\pi(x_i), x_i \in \mathcal{S}$ it is the equilibrium distribution of the process.

Chapter 2

An Overview on Queueing Theory

Queueing networks are a mathematical model used to describe a wide range of real-world systems, these models find a wide application in the analyses of computer systems. In traditional queueing network models, jobs move along a set of stations and compete for the service resources they offer. To obtain this service, jobs or customers may have to go through one or more of the basic units of this model: a queue. We say a customer arrives at a queue and waits for a certain period of time, after which he obtains the desired service and leaves that particular queue or the whole system. Processes describing arrivals and departures are subject to uncertainty, consequently they are defined through stochastic processes. We are particularly interested in stochastic models with underlying continuous-time Markov processes defined in a discrete state space.
Queueing theory [5, 11] gives us the mathematical means for studying this type of systems, allowing us to retrieve some quantities of interest such as throughput, response time, etc. in order to achieve an accurate performance evaluation of the system itself. One of the results we are more interested in is given by the product-form solution for the equilibrium distribution of our systems. Product-form expressions are a powerful analytical tool that allows us to examine the different components of a system in isolation with the aim of obtaining exact analytical results for quantities of interest.
In this chapter, we are going to present several aspects of queuing theory necessary to understand the model we have worked on and the project we have developed.

2.1 Queueing Theory Basics

In computer systems, jobs can compete for resources, use them, synchronize with each other, etc. In fact, any queueing system is can be subdivided into three parts (as represented in image 2.1): an arrival process that describes the way customers arrive at the system, a waiting room where they wait for service and a service room where they are actually served.

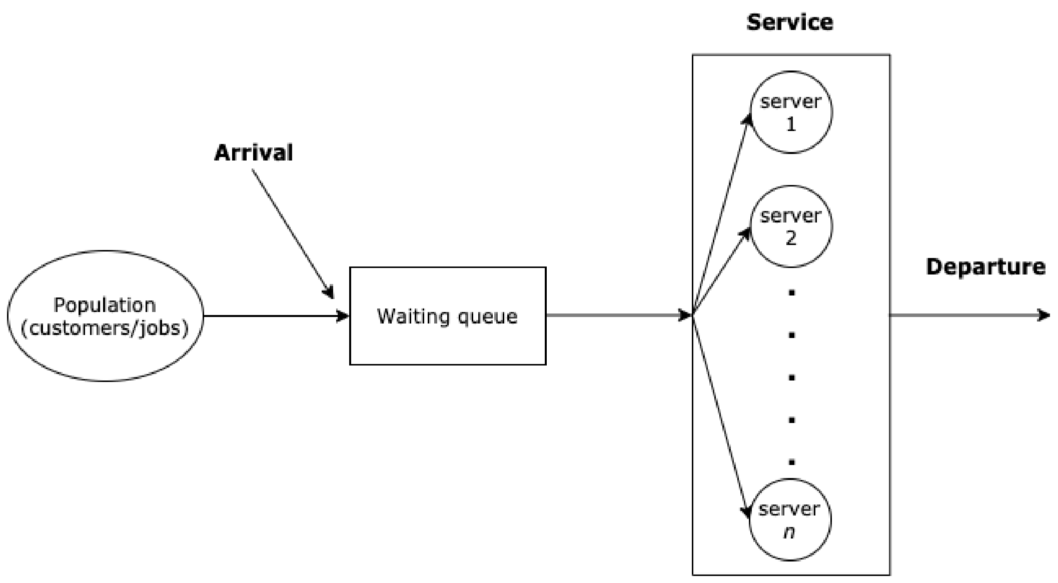


Figure 2.1: Graphic representation of a generic queueing system.

In this section, we will first present queuing notation and in particular Kendall's notation, then we will report some fundamental results of queueing theory and finally, we will show an operational analysis and an analytical study of the stochastic process underlying one of the simplest queueing models: the M/M/1 queue.

2.1.1 Queueing and Kendall's Notation

In a queueing system, we serve customers or jobs. If jobs arrive at time $t_1, t_2, \ldots t_n$, then the times $T_i = t_i - t_{i-1}$, with $i \in \{1, \ldots, n\}$ are called inter-arrivals times. We usually assume T_i are independent and identically distributed random variables; these form what we call the *arrival process*. The most common arrival process is the Poisson arrival process so that inter-arrival times are all independent and identically distributed exponential random variables.
We usually do not know the size of the jobs and consequently neither how long they will be needing service. Accordingly, we model the job service times as independent random variable which will give the *service time distribution*. When all the jobs are statistically identical we say that the system has a *single class* of jobs; conversely, if we can cluster the jobs into classes in which the elements are statistically identical we say the system is *multi-class*. When analysing a system we are also interested in the *system capacity* which is the maximum number of

jobs that the system can contain and in the *population size* that is the maximum number of jobs actually willing to enter the system.
Another important aspect of queueing systems is the *scheduling* or *queueing discipline* that is the policy adopted by the system to assign the resource(s) to the jobs waiting to be served. These policies can be *preemptive* or *non-preemptive* whether the job in service can be put again in the waiting room because of some event or never leave the service room unless its service is finished. A preemptive policy can either be *with resume*, if when the job that was preempted return in service continuing from the point it was left, or *with restart* if when a job is preempted the work done up to that moment is lost.
We usually want a queuing discipline to be *work-conserving.*

Definition 2.1. A queueing discipline is work-conserving if

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

Queuing systems are classified according to Kendall's notation; it is based on the type of arrival process, service time and other characteristics of the queueing system. According to this approach, we describe a queueing system 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 time, respectively. They are replaced by letters describing these distributions. The most used ones are:

  M denotes the exponential distribution, notice that if $A = M$ then the arrival process is a Poisson process.

  D denotes the deterministic distribution.

  E_k denotes the Erlang distribution with k phases of service

  PH indicates phase-type distributions i.e. defined in terms of the time to absorption of a continuous-time Markov chain.

  G denotes the general distribution, sometimes indicated with GI to stress the fact that it is independent of any other process characterising the queue and that inter-arrival times are independent.

  MAP stands for Markovian Arrival Process, which allows the arrival process to have interdependency among 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 the system.
- D is the scheduling discipline.

In general K and P are omitted, when they are they usually need to be intended as ∞. Notice that when D is omitted, the queueing discipline is to be implicitly intended as *First Come First Served* (FCFS). To understand better how this notation works, we propose some examples.

M/M/1 stands for infinite capacity and population, Poisson arrival process and exponentially distributed service times; queueing discipline is FCFS.

G/M/1 characterizes a queueing system with an arrival process with general inter-arrival times, exponentially distributed service times and FCFS queueing discipline; population and buffer size are infinite.

M/M/3/20/1500/FCFS denotes a queueing system with Poisson arrival process and exponentially distributed service times, 3 servers, a maximum of 20 jobs in the system, a population size of 1500 and FCFS queueing discipline.

2.1.2 Performance indexes

Analysing the queueing system components and their behaviour we can retrieve some performance of interest. Below, we present a list of the indexes we are more interested in. Starting from what we have already said we can easily imagine we will have to consider the waiting time, that is the time a customer actually spends waiting for service, and service time, that is the time that elapses from when a customer begins to be served to the moment the customer leaves service and the next customer is called in for service. Moreover, we usually know or can retrieve the arrival and service rate, the first one denotes the rate at which customers arrive requiring service, whereas the latter is the rate at which customers are served.

- Utilization, this is the portion of time spent by a server actually serving a customer instead of idling.
- Response time, this is the sum of the time spent by a customer waiting and in service for one visit to a server.
- Residence time, this is the total response time for a customer if it visits a server multiple times.
- Throughput, this is the at which customers are served, so it is basically the speed the system can provide for serving customers.

When designing a system it would be desirable to achieve the majority of the following goals. We aim at minimizing waiting, service, response and residence time and at obtaining the highest possible throughput. Furthermore, knowing the average arrival rate can lead to an appropriate designing of a system in such a way that it would never find itself overloaded or underloaded.
Finally, notice that we say a system is *unstable* if the number of jobs grows continuously and becomes infinite. Otherwise, we say the system is *stable.* We usually require that the maximum throughput of the system is lower than the arrival rate. In practice, this means that defining the system load intensity as

$$\rho = \frac{\lambda}{m\mu}$$

we want this quantity to be lower than one, where m is the number of available parallel servers. If this condition is not satisfied then the system will start to malfunction, which will lead to a likely crash because of the fact the load exceeds the system capacity. Conversely, if the condition is fulfilled it means the system meets the requirement for stability i.e., the system is designed properly to maintain the steady-state operating condition.

2.2 Queuing Systems: properties and theorems

In this section, we are going to present several important results that have important applications in the analysis of queueing systems. We will see that the majority of the results we will talk about actually have wide applications thanks to the loose assumptions needed for them to be applied.

2.2.1 Little's Law

Little's law relates the expected number of jobs, the expected response time and the throughput of a queueing system. Assume to observe a continuous time system in an interval $[0, t]$, then let us define the following quantities: $A(t)$ as the number of arrivals and $C(t)$ as the number of departures, both observed in the interval $[0, t]$.
Therefore, $N(t) = A(t) - C(t)$ is the number of customers in the system at time t, assuming that at time 0 the system is empty. $N(t)$ is clearly a stochastic process as we are not usually able to predict how many jobs there will be in the system at time t, however we may be able to give a probabilistic characterisation of this quantity. We are particularly interested in the random variable N that is the distribution of $N(t)$ when $t \to \infty$. Formally,

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

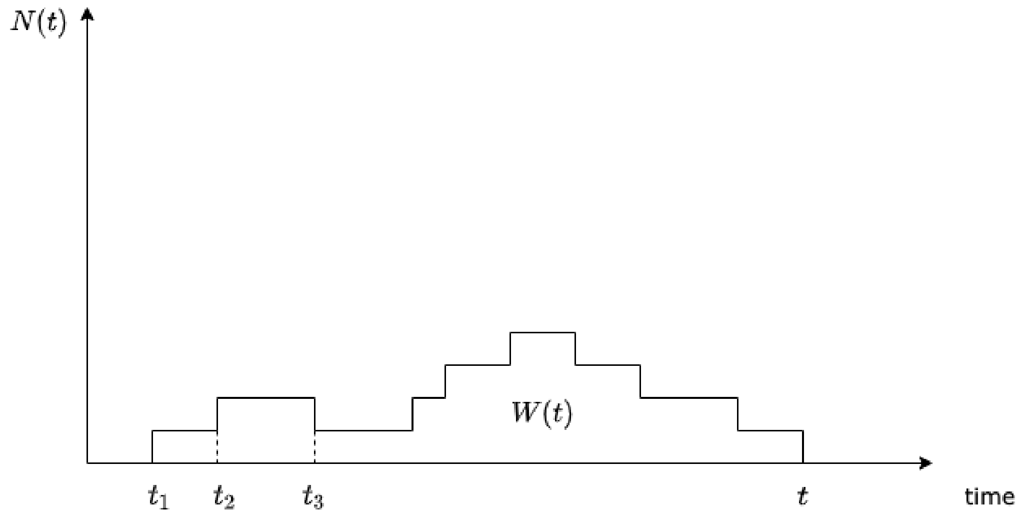


Figure 2.2: Example of realisation of the stochastic process associated with $N(t)$. At time $t_1$we have a first arrival, followed by a second one at time t_2; then at time t_3 we observe a first departure, and so on.

whenever this limit exists. In some cases, we are able to retrieve the distribution of N but we are usually more interested in its expectation, denoted by $\bar{N}$. Now, we define a quantity called *cumulative work* ($W(t)$), it measures the amount of pending work that has been stored in the system in the interval $[0, t]$. Formally,

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

In figure 2.2 below we show a possible realisation of the stochastic process $N(t)$ The expected number of jobs in $[0, t]$ is then given by the following formula:

$$\bar{N}(t) = \frac{W(t)}{t}$$

At this point, we use the quantities derived so far to retrieve the expected response time of the jobs in $[0, t]$, that we denote as $\bar{R}(t)$. Notice that for the moment we assume that both at time 0 and t the system is empty. Therefore, we had $C(t)$ jobs completions in $[0, t]$ and we want to find the cumulative time spent in the system by all the jobs. This is nothing more than $W(t)$ accordingly we can write:

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

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

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

This relation is known as Little's law. Notice that the reasoning that we have shown is rather informal and relies on the assumption that the system is empty both at time 0 and t. However, Little's law holds also without this assumption. So we have what follows.

Theorem 2.1 (Little's Law)**.** *Given a queueing system without internal loss or generation of jobs, we have that the following relations hold for any finite time horizon t.*

$$\bar{N}(t) = \bar{R}(t)X(t)$$

Notice that the assumptions of Little's law are loose as a matter of fact we require neither a specific arrival process nor a special service time distribution. Little's theorem can be derived from 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 jobs and assume that the following limits exist and are finite:*

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t} \tag{2.1}$$

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

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

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:

$$\bar{N} = \lambda\bar{R} \tag{2.4}$$

An interesting fact about the theorem above is that we are no longer using the throughput as in Little's law but the intensity of the arrival process, this step is particularly relevant and it follows from the existence of the limits. In fact, if limits [2.1] are well defined and finite then the queueing system is stable consequently for $t \to \infty$ the throughput is balanced with the intensity of the arrival process. Moreover, notice 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.

2.2.2 Utilisation Law

Consider a single server queue with a general arrival process and general service time distribution. We assume that jobs have an expected size of μ^{-1} and that 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 fraction of time 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 in the system. Thus, we have the following proposition.

Theorem 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.

2.2.3 PASTA Property and Inspection Paradox

PASTA (Poisson Arrivals See Time Averages) property is widely used for the analysis of queueing systems, this is in particular because it allows a simple derivation of some performance indexes.

Theorem 2.4 (PASTA property)**.** 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.

Now we want to characterize the residual service time seen by a customer at its arrival epoch conditioned to the fact that a job is actually being served.

Definition 2.2 (Residual Life of a Job)**.** 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 distribution we know that in queues with this particular type of service time distribution, the residual life is identical to the original service time. However, this does not hold in general. 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 as the residual life of a random observer. To proceed with a formal derivation we start with three observations:

1. The probability of finding a job of size x (i.e. with service time x) is directly proportional to x.

2. The probability of finding a job of size x is proportional to the probability that the job is actually requesting that service time (i.e. to the probability it does exist).

3. We intuitively support the idea (for now) that the random observer does see half of the residual service time (in average) of a generic job because at its arrival time it has more probability to see a large job in service than a small one. Notice, in fact, that sampling randomly the system to estimate the average service time is wrong because we would not consider the fact that it is more probable that we sample in the middle of a big job, consequently there would be a risk of overestimating.

Let $f(x)$ be the probability density function the service time distribution, then $f(x)dx$ is the probability that a job of size x is actually requesting that service time. Therefore, if we call $g(x)$ the service time distribution seen by a random observer, we have:

$$g(x)dx = Kf(x)dx$$

where K has to be determined. Since $g(x)$ must be a probability density function the following relation must hold:

$$\int_0^\infty g(x)dx = 1 \quad \text{i.e.} \quad \int_0^\infty Kf(x)dx = 1 \qquad \Rightarrow \; K = \mu$$

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

$$\boldsymbol{P}\{Y \leq y | X = z\} = \frac{y}{x} \qquad 0 \leq y \leq x$$

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

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

with $0 \leq y \leq x$. Thus, we have the unconditional probability density function of Y that is:

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

where $f_Y(y)$ is the residual service time.

As an example, consider the case in which $F(x) = 1 - \mathrm{e}^{-\mu x}$ i.e. the service time is exponentially distributed. Then, we have

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

that is exactly what we would be expecting in the case of exponentially distributed service times with rate μ.
We are now able to derive the following important result:

Theorem 2.5 (Average Residual Life)**.** The average residual life for a service with cumulative density function $F(x)$ is:

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

where $M_2 = \int_0^{\infty} x^2 f(x)dx$ is the second moment of the service time distribution.

In conclusion, we can highlight that the expected residual time does not depend only on the mean but also on the variance, a high variance means high expected residual life whereas low variance means lower expected residual life.

2.2.4 Stability of a Queue

We are usually interested in studying the behaviour of queueing systems when $t \to \infty$ i.e. what we call the queueing systems *long run* behaviour. We now try to give a characterization of the notion of stability in a queueing system exactly when $t \to \infty$. We already know $N(t)$ is the stochastic process underlying the population of the system, for any finite t, in principle we can derive $\boldsymbol{P}\{N(t) = n\}$ i.e.the probability that at time t the system contains exactly n jobs. However, if $t \to \infty$, the limit

$$\lim_{t \to \infty} N(t) \tag{2.5}$$

cannot be finite, because as t grows more and more jobs are expected to be found in the system.

Definition 2.3 (Stability of a Queue)**.** We say a queue is *stable* if and only if, in the long run (i.e. $t \to \infty$) we have a finite expected number of jobs in the system and the expected response time for the jobs is also finite.

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

- 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 [2.5] exists and has a finite mean;
- If the queue has an underlying Markov chain, then this is ergodic.

2.3 M/M/1 Queue

The M/M/1 queueing system is composed 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 First Come, First Served. The analysis of this type of queue widely exploits the memoryless property of both inter-arrival and service times, plus of course their independence.

2.3.1 Operational Analysis

Given the intensity of the arrival process λ and the service rate μ we assume $\lambda < \mu$ for stability so we know that $X = \lambda$ and $U = \dfrac{\lambda}{\mu}$ since we have one single server. $\bar{N}$ is the expected number of jobs in the system, seen by a random observer. For the PASTA property we have that:

$$\bar{R} = \underbrace{\bar{N} \cdot \frac{1}{\mu}}_{\text{waiting time}} + \underbrace{\frac{1}{\mu}}_{\text{service time}}$$

This comes from the fact that when I arrive I see $\bar{N}$ jobs in the system in average, however there is one in service, because of the memoryless property of the exponential distribution we do not have to account for the residual service time of the job in the service room, i.e. its expected service time is equal to its mean consequently to $\dfrac{1}{\mu}$. At this point, from Little's law we know $\bar{N} = \lambda\bar{R}$, consequently we have

$$\bar{R} = \lambda \bar{R} \cdot \frac{1}{\mu} + \frac{1}{\mu}$$

$$\bar{R}\left(1 - \frac{\lambda}{\mu}\right) = \frac{1}{\mu}$$

$$\bar{R} = \frac{1}{\mu}\left(\frac{\mu}{\mu - \lambda}\right) = \frac{1}{\mu - \lambda}$$

accordingly, we can also obtain

$$\bar{N} = \frac{\lambda}{\mu - \lambda}$$

2.3.2 Study of the Underlying Stochastic Process

The states of the queue are described by the number of jobs in the queue and can range from 0 to infinity in the natural numbers meaning that the state space can be formally described as $\mathcal{S} = \mathbb{N}$. An M/M/1 queue can be described as a Birth-Death process, meaning that at a certain epoch 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 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, 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 this is the residual residence time in state i. We conclude that $n(t) \in \mathbb{N}$ (i.e. the number of jobs in the system at a certain time t) is a continuous time Markov chain (as we can see in fig.2.3(a)) and this is because for each state i and each time epoch $t \in \mathbb{R}$, the current state of the process contains all the information for the probabilistic characterization of its future evolution.

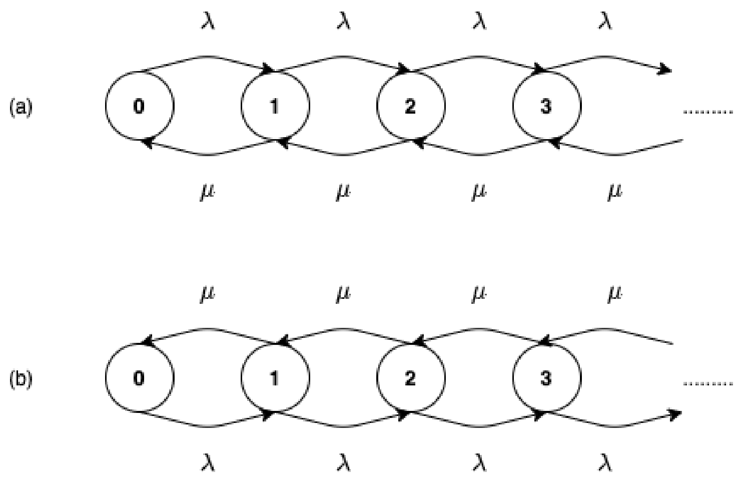


Figure 2.3: Continuous Time Markov Chain underlying the M/M/1 queue. Forward process (a) and Reversed process (b).

We now show that the Markov chain underlying the M/M/1 queue is reversible which will yield a relation that can be used to compute the equilibrium distribution by induction. The Markov chain underlying the M/M/1 queue is showed in fig. 2.3(a), we now construct the reversed process. For Kolmogorov's generalized criteria we have that for every state of the forward process the residence times should be the same as the ones of the reversed process; so, for state 0 of the reversed process we have that the transition rate of the outgoing arc will be λ. Again, for Kolmogorov's generalized criteria the product of the transition rates of a cycle in the forward process should be the same of the reversed one consequently the incoming arc of state 0 will have transition rate μ. For state 1 the reasoning is the same: in the forward process the residence time of state 1 is given by $\lambda \cdot \mu$ consequently the outgoing arc from state 1 in the reversed process will have transition rate λ, and for the product of the transition rates rule we will have that the rate of the incoming arc of state 1 in the reversed process will be μ. Following with this reasoning for all the other states we obtain the reversed process in fig.2.3(b). We observe that it is statistically indistinguishable with respect to the forward process consequently the process is reversible. The detailed balance conditions states that the equilibrium distribution satisfies

$$\pi(i)q_{i,k} = \pi(j)q_{j,i} \qquad i,j \in \mathcal{S}$$

which in this particular case, as we have only transitions from state i to state $i-1$ ($i > 0$) or to state $i+1$ ($i \geq 0$) for all $i \in \mathcal{S}$, becomes

$$\pi(i)q_{i,i-1} = \pi(i-1)q_{i-1,i}$$

and hence the equilibrium distribution is given by

$$\pi(j) = \pi(0) \prod_{i=1}^{j} \frac{q_{i-1,i}}{q_{i,i-1}} \tag{2.6}$$

In stability, this equation gives the stationary distribution of the queue length as a function of $\pi(0)$. In order to retrieve $\pi(0)$ we impose the normalising condition of the probability distribution, i.e.

$$\sum_{i=0}^{\infty} \pi(i) = 1$$

Consequently, in the M/M/1 case, with the given transition rates we have that if the arrival rate λ is less than the service rate μ the process has an equilibrium distribution which is, from equation 2.6:

$$\pi(i) = \pi(0)\rho^i$$

where $\rho = \dfrac{\lambda}{\mu}$ (ρ is also called the *load factor*) and $i > 0$.
In order to retrieve $\pi(0)$ we impose the normalising condition of the probability distribution and for the geometric series we have that, if $\rho < 1$ (i.e. $\lambda < \mu$), $\sum_{i=0}^{\infty} \rho^i$ converges to $\dfrac{1}{1-\rho}$, consequently we obtain

$$\pi(0) = 1 - \rho$$

therefore

$$\pi(i) = (1-\rho)\rho^i \qquad \text{with } i \in \mathbb{N}.$$

Notice that this algebraic reasoning is coherent with the stability condition for a system with infinite capacity and without costumer rejection, i.e. the arrival rate must be lower than the service rate. At this point we can derive the utilisation of the queue, given by

$$U = \sum_{i=1}^{\infty} = 1 - \pi(0) = \rho.$$

Considering $\pi(i)$ as the probability that the queue contains i jobs we have that the expected number of jobs is:

$$\bar{N} = \sum_{i=1}^{\infty} i\pi(i) = \sum_{i=0}^{\infty} i(1-\rho)\rho^i = (1-\rho)\sum_{i=1}^{\infty} i\rho^i = \frac{\rho}{1-\rho}$$

and by Little's theorem we obtain the expected response time,

$$\bar{R} = \frac{\bar{N}}{\lambda} = \frac{\frac{\lambda}{\mu}}{1-\frac{\lambda}{\mu}} \cdot \frac{1}{\lambda} = \frac{1}{\mu - \lambda}$$

2.4 Queueing Networks

Queueing networks are used to model complex systems as a matter of fact queueing networks are composed of several single queues interconnected among them thanks to a routing network. Usually, we represent servers with circles and queues by rectangles as we have already seen; moreover, we denote the routing network with arrows (an example is given in fig.2.4).

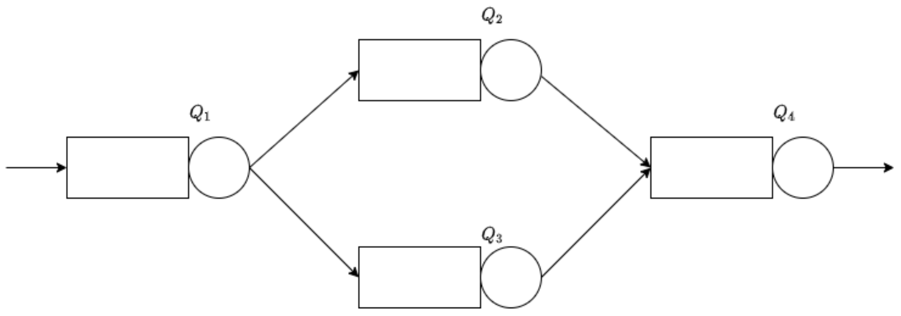


Figure 2.4: Example of an open queueing network.

Notice that in this setting what happens is that jobs departing from one queue arrive at another (or maybe the same) queue, consequently we are interested in studying the performance of the whole system and not of a single queue.
Queueing networks can be classified in two different classes: open or closed queueing networks. In the first case we have one or more inputs coming from the outside and one or more external destinations, whereas closed queueing networks are characterized by a fixed number of customers that circulate continuously among the queues, never leaving the system.
Considering an open network we have that each queue in the system can have an input stream coming from the outside or from another component of the system. This is crucial to understand because it means that knowing the arrival process to the system does not directly provide us also with the knowledge of the input streams to each queue. In fact, feedbacks and dependencies introduced by the (deterministic or probabilistic) routing network can considerably change the arrival process seen by each system component.
As before, we are interested in the steady-state behaviour of the network. This is why we introduce the following definition.

Definition 2.4. We say 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 $t \to \infty$.

In fact, for an open queueing network to be stable we require each station of the network to be stable. If we let $N(t)$ be the total number of jobs in the queueing network at time t we can define the stationary expected number of jobs in the system as

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

whenever this limit exists. In open queueing networks we have that in stability the throughput is equal to the total incoming flow and independent to the single service rates of the stations.
We are now going to present some fundamental results.

2.4.1 Burke's Theorem

Burke's theorem (presented in [12]) is a very well-known fundamental result in queueing theory. The theorem states what follows.

Theorem 2.6. For an $M/M/1$, $M/M/m$ or $M/M/\infty$ queue in the steady-state

1. the departure process is Poisson with rate equal to the rate of the arrival process.

2. at time t the number of customers in the system is independent of the sequence of departure time prior t.

Burke provided a first proof showing that the distance in time between a departure and the following one is independent from the others time intervals and exponentially distributed with parameter equal to the arrival rate; meaning, in practice that from the point of view of an external observer it is impossible to distinguish the departure process from the arrival one as they look exactly the same.
Another interesting way to prove the same result consists of exploiting the reversibility property. In fact, statement 1 of theorem 2.6 follows from the fact that the arrival process is Poisson with rate λ and its reverse is also Poisson and with the same rate. Moreover, statement 2 of theorem 2.6 follows from the fact that the departures prior to t in the reversed system is the same process as the arrival process after t in the reversed process. It is clear that the number in the queue is independent of the arrivals after that point in a Poisson system. We can state what we have just presented above because the stochastic processes underlying the queues cited in 2.6 can be modelled by Birth-Death processes which we have already seen to be reversible.

This theorem forms the basis for the theory of *product-form* networks. It actually allows us to study the queues in isolation which greatly simplifies analysis. In fact, we can state that the steady-state distribution of a system in which this theorem can be applied is given by the product of the marginal probabilities of the single queues i.e., of their stationary distributions. Notice that product-form expressions provide us exact analytical results for the retrieval of the performance indexes we are interested in.

Example. Consider the following queueing network (fig. 2.5) that we can define as an $M/M/1$ tandem queue system composed of two queues.

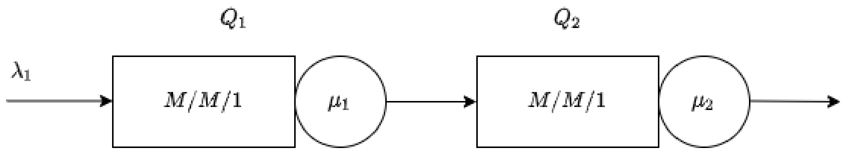


Figure 2.5: $M/M/1$ tandem queue network.

Thanks to Burke's theorem we can analyse the two queues in isolation, consequently, recalling the $M/M/1$ results, we can write

$$\pi_1(i) = (1-\rho_1)\rho_1^i \qquad \rho_1 = \frac{\lambda_1}{\mu_1}$$
$$\pi_2(i) = (1-\rho_2)\rho_2^i \qquad \rho_2 = \frac{\lambda_1}{\mu_2}$$

and since the processes are independent of the number of jobs in the queue we can write the following product-form result

$$\pi(i,j) = (1-\rho_1)(1-\rho_2)\rho_1^i\rho_2^j = \pi_1(i)\pi_2(j)$$

which gives the stationary probability of serving i jobs in Q_1 and j jobs in Q_2. This result can easily be expanded by induction to an arbitrary number K of queues, leading to the following more general result:

$$\pi(n_1,\dots n_K) = \prod_{i=1}^{K} \pi_i(n_i) \qquad \text{where } \pi_i(n_i) = (1-\rho_i)\rho_i^{n_i}, \quad i = 1,\dots,K$$

2.4.2 Jackson's Theorem

Jackson's theorem (presented in [13, 14]) is one of the most significant developments in queueing theory as it extends Burke's theorem and allows a wider range of networks to have a product-form solution. This theorem can be applied to Jackson's networks i.e. open networks of K $\bullet/M/m$ queues where jobs arrive from the outside according to independent Poisson processes with rate λ_i (with $i = 1,\dots,K$) to one or more queues and are then probabilistically routed from

one queue to another, or possibly the same, until they eventually leave the system. Notice that departures may take place from one or more queues and the fact we allow for feedback in the network configuration is exactly the reason why we indicate our queues as $\bullet/M/m$; as a matter of fact even if the arrival process is Poisson, because of cycles we cannot be sure that the flow arriving at each queue is still a Poisson process.
One of the requirements for Jackson's networks is what we call *probabilistic routing*, it means that jobs travel from one station to another following a fixed routing matrix P; i.e., after being served at queue Q_i a job moves to queue Q_j (we may have also $i = j$) with probability p_{ij}. We denote the probability that a job leaves the system from queue Q_i with p_{i0}. Moreover it must always hold that

$$\sum_{j=0}^{K} p_{ij} = 1 \qquad \forall\ x_i \in \mathcal{S}.$$

Jackson empirically discovered that even if we do not know what kind of process arrives at Q_i, whatever that is it has an effect on Q_i which is the same that would have a Poisson process with the same intensity. This means that even if the arrival process is not Poisson, it behaves as if it was Poisson.
The arrival rate at each queue can be computed through *traffic equations*; in fact, these equations describe the mean arrival rate of traffic. Let e_i denote the total intensity of the incoming traffic at each station Q_i from the outside or from any other station of the network. Moreover, suppose to have the vector $(\lambda_1, \dots, \lambda_K)$ that represents the arrival rates at each station; then, we have:

$$e_i = \lambda_i + \sum_{j=1}^{K} e_j p_{ji} \tag{2.7}$$

this holds in stability because we know that in an open network considered in steady-state the throughput is equal to the total incoming traffic flow. Furthermore, if the routing of the network is irreducible then the *system of traffic equations* [2.7] admits a unique solution that provides us with the throughput and the incoming traffic intensity at each station. Now we can formally present Jackson's theorem.

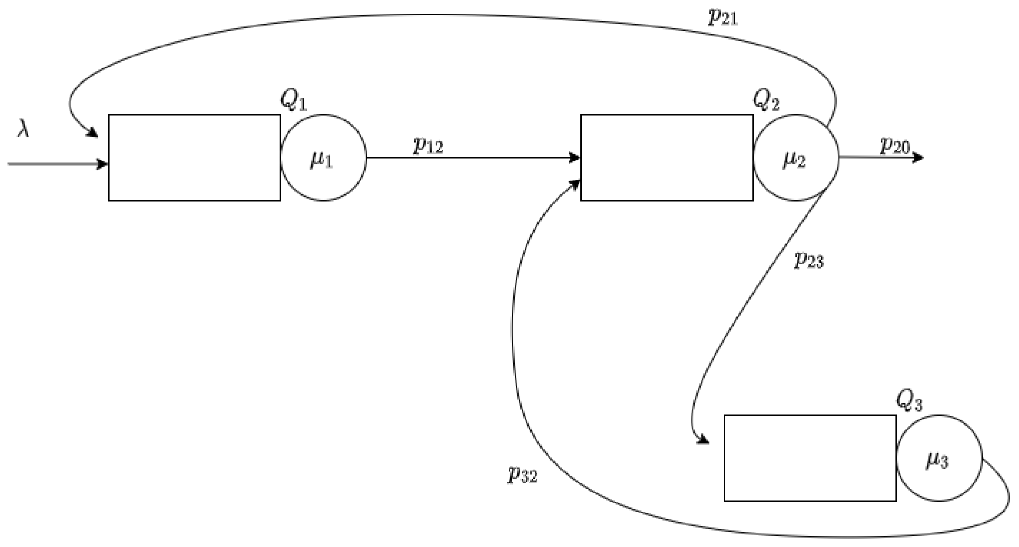
Theorem 2.7 (Jackson's Theorem). Provided that the arrival rate at each queue is such that in equilibrium exists (i.e. load factor of each queue is less than 1) the probability of the overall system state $\mathbf{n} = (n_1, \dots, n_K)$ for K queues is given by the product-form expression

$$\pi(\mathbf{n}) = \prod_{i=1}^{K} \pi_i(n_i)$$

where $\pi_i(n_i)$ is the steady-state distribution of station i *as if* it were subject to a

Poisson arrival process with intensity e_i.

Example Consider the following queueing network



where $\mu_1 = 3$ jobs/s, $\mu_2 = 3$ jobs/s, $\mu_3 = 2.5$ jobs/s, $\lambda = 2$ jobs/s and

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0.2 & 0 & 0.1 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} p_{10} \\ p_{20} \\ p_{30} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.7 \\ 0 \end{bmatrix}$$

Below we write the system of traffic equations for this particular network

$$\begin{cases} e_1 = e_2 p_{21} + \lambda \\ e_2 = e_1 + e_3 \\ e_3 = e_2 p_{23} \end{cases}$$

Accordingly we have

$$e_1 = \frac{\lambda \cdot (1 - p_{23})}{p_{20}} = 2.57 \qquad e_2 = \frac{\lambda}{p_{20}} = 2.86 \qquad e_3 = \frac{\lambda \cdot p_{23}}{p_{20}} = 0.29$$

By Jackson's theorem we have

$$\pi_i(n_i) = (1 - \rho_i)\rho_i^{n_i} \qquad \text{with } \rho_i = \frac{e_i}{\mu_i}$$

If we want to know which is the expected number of jobs in the system we can compute what follows:

$$\bar{N} = \bar{N}_1 + \bar{N}_2 + \bar{N}_3 = \sum_{i=1}^{3} \frac{\rho_i}{1 - \rho_i}$$

and if we are interested in the expected response time we can apply Little's theorem: $\bar{R} = \dfrac{\bar{N}}{\lambda}$.

Chapter 3

Stochastic Simulations

In this chapter, we present the main aspects of stochastic simulations so that the reader will be able to understand in detail the simulator we built and that will be shown later. A stochastic simulation [15, 16] is nothing more than a computer experiment where the real environment is replaced with a program execution; in this experiment variables change stochastically with individual probabilities. Simulations are a tool that helps us to design or study complex systems. Attention must be paid in modelling the system randomness, in the statistical analysis of the results and also in the validation of the simulation itself. This tool offers both advantages and drawbacks.
Drawbacks reside in the fact that, usually, simulations can be expensive and time-consuming to develop, moreover for each run of the simulation model we only obtain estimates of a model's real characteristics consequently different independent runs will be needed. Furthermore, if we do not verify carefully that the simulation is a fair reproduction of the real system or of a model the obtained results are completely useless.
Nevertheless, they also offer several advantages such as the possibility of approaching also complex, real-world systems that we would not be able to approach accurately with mathematical models, the fact that simulations allow us to estimate performance indexes of an existing system under different experimental circumstances, over which we have clearly more control than in the case we were dealing with the real system. Finally, notice that simulations are also often used to compare different systems designs.

One of the first things we need to understand when talking about simulations is the difference between *real* time and *simulated* time. In a simulation the flow of time is controlled by a computer: it will be necessary to deal with a certain parallelism as in real systems more actions can happen at the exact same time whereas in the simulated program they will be serialized. This is done taking into

account the simulated time, i.e. the time in which the event should happen in the real system. Every action is then decomposed into instantaneous events (for example, the arrival of a customer to the system) e and we assume that two events cannot happen at the exact same time.
The real time of a simulation depends instead on the performance of the computer we are using to execute the simulation (e.g. speed of the processor, amount of memory available) and on the level of optimization of the simulation program. We now need to define some basic concepts:

- A *system* is a collection of components that interact among them in order to satisfy some requirements. At every moment in time, the system is in a particular situation that we call *state*; this is determined by the instantaneous conditions of all the system components. Notice that a system evolving in time is described from the state history.

- A *model* of a system is its mathematical representation, as a matter of fact, the model of a real system offers a certain abstraction level. It is important to remember that the model by itself does not represent the history of the system as a matter of fact a model state is always associated with a state of the real system.

- A model presents different types of variables.
    - *Exogenous* variables depend on the environment, if we can control them we call them *parameters*.
    - *Endogenous* variables depend on the model itself, in practice they are the output results.
    - *State* variables describe the state of the system in a certain moment and they vary in time interacting with endogenous and exogenous variables; remember that these interactions are defined by the characteristics of operations.

To simulate a model means to produce the history of the states of the model and then interpret it as the history of the states of the system. The simulated model is said to be the *simulation model* whereas the simulation on the computer is called *digital simulation*. The suitability of a system model is decided according to the study goals.

3.1 Types of Simulation

There are different types of simulation, usually the following classification is used.

Deterministic/Stochastic. A deterministic simulation does not have random components, this type of simulation is used when we want to test some property of a system for which the environment is completely known, for example to test the robustness of a certain implementation. In the majority of cases however, this type of simulation is not sufficient nor available (in fact, it is not easy to know everything about a system) as a matter of fact random components help us to better model the system environment; a direct consequence of stochastic simulations is that also the output will be random and this will allow us to gather statistics and perform data analysis.

Terminating/Non-terminating. A simulation is said to be *terminating* if it terminates when a particular condition happens; for example, if we want to evaluate the execution time of a sequence of operations in a certain well-defined environment, it is possible to execute the sequence with the simulator and keep record of the simulated time. Typically, we use terminating simulations when we are interested in the lifetime of a certain system or when the system inputs depend on time.

Asymptotically stationary/Non-stationary. The stationary condition can be verified only in case of non-terminating stochastic simulations as stationarity is a property that needs to be evaluated over time. Often, the simulation state depends on the initial conditions and it may be difficult to find favorable initial conditions; for example, if we simulate an information server probably we would start with an empty buffer but this hypothesis is actually too optimistic because in a real system executed for a certain amount of time there will be data structures that will not be empty.
Stationarity is a solution to this problem, as a matter of fact a stationary simulation, i.e. that has a unique stationary regime, is such that we do not retrieve any useful information from its initial state as its state distribution becomes independent of the initial conditions.
Unfortunately, in practice a simulation is rarely exactly stationary however it can be asymptotically stationary; meaning that after some simulated time, the simulation will get to a stationary state. More precisely, we have that a simulation with time independent input can always be thought of as a Markov chain. Remember that a Markov chain is a generic stochastic process such that to simulate the future after a certain time t, the only information needed is the state of the system at time t (*memoryless property*). Markov chain theory states that the simulation can converge or diverge to a stationary behaviour. If our aim is to measure some performance indexes of the system under study we are probably interested in its stationary behaviour. Different reasons could cause the absence

of an asymptotically stationary regime, for example:

- Unstable models: in queueing systems where the arrival rate is greater than the service capacity, in this case the longer the execution time of the simulation, the longer will be the length of the queue, i.e. it will grow indefinitely without reaching a stationary state,

- *Freezing simulation*: the simulated system does not converge to a stationary regime and instead freezes becoming slower and slower. This behaviour is usually caused by some rare event that drastically alters the system's functioning. The probability of observing a rare event is proportional to the length of the simulation time. If, by chance, the simulation has some regenerative state (i.e. a point in which the system reaches a clean state, e.g. if it becomes empty again) then the simulation freezes if the interval of time between two regenerative points has an infinite average.

- Models with inputs depending on the simulation time, in this type of models is exactly this dependence to prevent the system from the achievement of a stationary regime; an example of this situation may be a the internet traffic that grows every month and that in some moments of the day is more intense.

Notice, in particular, that even if the initial state does not influence the stationary behaviour it does influence the time required to reach the stationary regime. Moreover, it is important to underline that in stationarity the state of the model does not become constant but it is the probability of observing a certain state that stabilizes.

In the majority of cases when we execute a non-terminating simulation it is necessary to be sure that this reaches a stationary regime; otherwise, the simulation output and the consequent analysis will depend on the initial condition and on the length of the simulation itself.

3.1.1 Simulation Techniques

Here, we present the three main classes of simulators.

Discrete Event Simulation - DES. In this case, we model a set of entities which state evolves in time, these entities interact in order to compete for resources or to synchronize on some particular events. The crucial point in this type of simulation is the maintenance of the list of events that need to be processed. In the next section, we will describe in detail this type of simulation as it is the one that we used for our project.

System Dynamics - SD. With respect to the previous type, the approach is top-down as a matter of fact we model the entire system instead of the singular entities; to model the connection among state variables, differential equations are used. This technique is used when the system has a big number of identical or very similar entities and consequently it is more appropriate to consider their multiplicity rather than their identity.

Agent Based Simulation - ABS. This technique consists of simulating the decisional process of every single agent that is part of the system, the final goal usually is to observe the behaviour of the system emerging as the result of agents interaction.

3.2 Random Number Generator

A simulation of a system in which we recognize random components clearly requires some method of generating numbers that are random. We know computers are deterministic machines, exception made for some particular kind of hardware that is able to generate only numbers that are actually pseudo-random and not completely random. It is possible to limit the problem to the generation of random numbers in the interval $(0, 1)$ with uniform distribution, from which then other distribution are generated. Random number generators (RNG) must have the following properties:

- the generated number must belong to the interval $(0, 1)$, they must be uniformly distributed and there shall be no correlation among the various extractions
- they should be fast and they should not occupy too much memory
- extractions should be reproducible
- different streams of random numbers should be available
- the random number generator should be portable

The generation of (pseudo)random numbers is a very delicate topic as a lot of generators available in software or libraries are not adequate to be used in simulations. Now, we give a formal definition [17] of (pseudo)random number generator and then we will present the one we implicitly used in our simulator i.e. the *Marsenne Twister* generator [18].

Definition 3.1 (Random Number Generator)**.** Mathematically, a (pseudo)random number generator can be defined as a quintuple (S, μ, f, U, g), where S is a finite set of states, μ is a probability distribution on S used to select the initial state s_0, called *seed*, $f : S \rightarrow S$ is the transition function, $U = [0, 1]$ is the output set and finally $g : S \rightarrow U$ is the output function.

The state evolves according to the following recurrence

$$s_i = f(s_{i-1}) \qquad \text{for } i \geq 1$$

and the output of step i is:

$$u_i = g(s_i) \in U$$

these u_i are the so called pseudo(random) number produced by the generator. Since set S is finite sooner or later the generator will come back to a previously visited state, i.e.

$$s_{l+j} = s_l \quad \text{for some } l \geq 0 \text{ and } j > 0$$

consequently

$$s_{i+j} = s_i \text{ and } u_{i+j} = u_i \ \forall i \geq l.$$

The smallest value of $j > 0$ for which this happens is called *the length of the period*, denoted by the Greek letter ρ. Clearly, this value cannot be greater than S, in particular if b bits are used to represent a state then we will have $\rho \leq 2^b$. The best (pseudo)random number generators are built in such a way that the length of their period gets as close as possible to that limit.

3.2.1 Marsenne-Twister Random Number Generator

Marsenne-Twister is a (pseudo)random number generator, is the *general-purpose* generator most used. It was developed in 1997 by Makoto Matsumoto and Takuji Nishimura. The algorithm of this generator generates an optimal set of (pseudo)random numbers, this is because the creators tried to fix the known issues of the generators existing at the time when they were creating their own generator.
The name of the generator derives from the fact that the length of its period is chosen so that it is a Marsenne prime number. In mathematics, a Marsenne prime number satisfies the following property

$$M_p = 2^p - 1$$

with p that is a positive prime integer.
Among the many advantages that this generator offers we recall the following:

- it has a huge period that equals $2^{19937} - 1$ and this was proved by the creators of the algorithm
- it passed different randomness statistical tests

- it turned out to be more efficient than other algorithms that do not even get close to it as far as quality is concerned
- it is extremely portable, as a matter of fact it is included in different programming languages and libraries.

3.3 Discrete-Event Simulation

The majority of communications systems are simulated using discrete-event simulations. To deeply comprehend this technique we must keep in mind the concepts of entity, i.e. the elements that interact, and of event, intended as the change of state of one or more entities. The most important aspects of this methodology reside in keeping track of a global variable that will have the name `currentTime` and in the presence of an event scheduler.
Events are objects that represent different transitions, every event has its associated time (i.e. `timestamp`); an event scheduler is nothing more than an ordered list organized in ascending order according to `timestamp`. What happens is that the simulation program takes the first event from this list, moves `currentTime` to the `timestamp` of the picked event and executes the event. The execution of this event may cause the scheduling of new events with a grater `timestamp` with respect to `currentTime` and it may also cause the order change or the cancellation of some events that were already in the event list. Remember that the global variable `currentTime` should never be modified by an event except for the fact that as events progressively take place `currentTime` jumps from a `timestamp` value to another. This is exactly why it is called discrete event simulation as the events that change the system states form a discrete set. Notice also that event not only gave to simulate the logic of the system but they also have to keep updated some counters that will be needed in order to perform output data analysis.

In practice, a simulator event oriented is realised as follows; we maintain a calendar of events that we call *Future Event List* (FEL), from which the event with the smaller `timestamp` is extracted. A subroutine manages the event modifying the state of all the entities influenced by the event and eventually modifying the value of the endogenous variables (output), the same subroutine also updates the calendar of events. We show two diagrams in fig. 3.1 that present the functioning of discrete event simulations intuitively.

As far as the FEL management is concerned some basic operations are required such as: event insertion, extraction of the next event to be processed, deleting of an arbitrary event (i.e. not necessarily the event with the smaller `timestamp`). The advancement of the simulated time, as we have seen proceeds jumping from one timestamp event to another. This advancement can proceed

- with fixed intervals, meaning that time advances of Δt at each step and all the events happening in that interval of time must be simulated (these events will be considered simultaneous), with this option some problems should be taken care of such as how to treat the multiple events and how to choose Δt
- with the events i.e. the time advances of a quantity equal to the time needed for the next event in the FEL to happen.

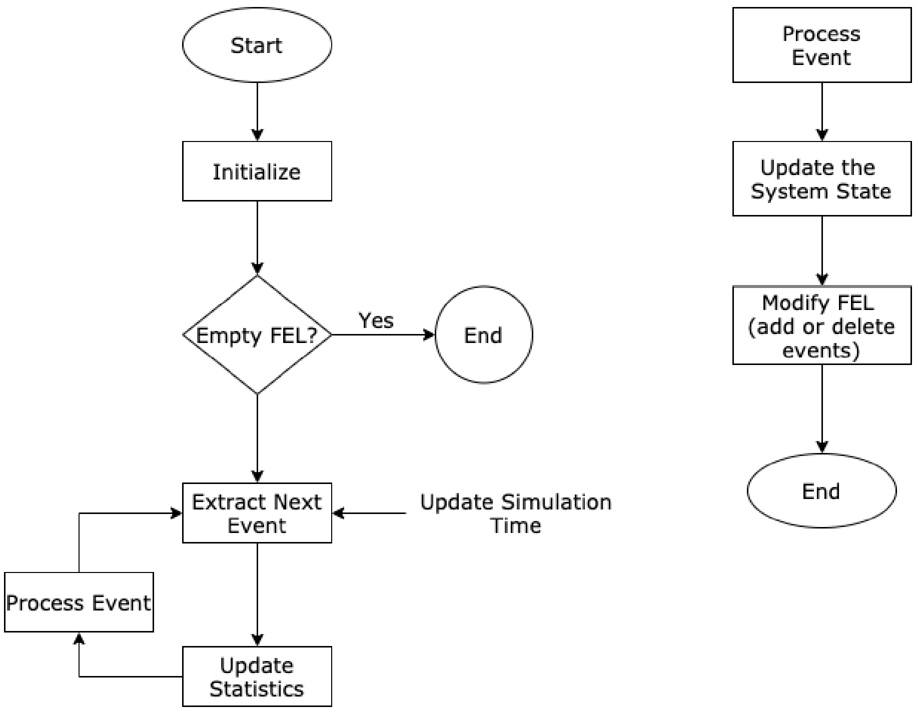


Figure 3.1: On the left the fundamental cycle of a discrete event simulation and on the right the processing of an event.

3.4 Output Analysis

When dealing with simulation output it is important to analyse it appropriately. This is mainly because each run of a simulation offers us a possible realization of the random variables underlying the simulation. Considered this, it is easy to understand why one run of arbitrary length of the simulation program is not enough to consider the retrieved results as the real characteristics of the system under study. In general, we produce n independent runs, where independent means that for each run we change the seed for the generation of the needed random numbers. Notice, that the initial settings and the configuration of the simulation are identical for all the independent runs. Furthermore, as initial conditions usually do not represent fairly the conditions we would find the system in, in an arbitrary moment we consider, for the gathering of the statistics of interest, the steady-state behaviour of the stochastic process underlying the simulation. In fact, we do not want our results to be biased towards the initial settings of the simulation.

3.4.1 Validation

The main goal of validation is to produce a simulation model that represents accurately enough the real system behaviour so that the simulation can be used to experiment the system under different circumstances, to analyse its behaviour and predict its performance. Validation is defined in [19] as determining whether the model's output behaviour has a satisfactory range of accuracy for the model's intended purpose over the domain of the model's intended applicability.

The level of accuracy required is usually defined by the range within which the difference between a model's output variable and the corresponding system output variable must be contained. This range is commonly known as the model *acceptable range of accuracy.* When the variables of interest are random variables, these are usually of primary interest and they are usually the quantities used to determine the model validity.

To obtain a high degree of confidence in a simulation model and its results, comparisons of the model's and system's output behaviours are usually required. However, often the situation is such that the system is not observable consequently it is usually not possible to obtain a high degree of confidence in the model. In this situation, the usual approach is to carefully analyse the model output behaviour and to make some comparisons with other valid models (also theoretical ones) whenever possible.

One of the most known approaches used in comparing the simulation model output behaviour to either the system output behaviour or another (validated) model output behaviour is the use of confidence intervals as they allow us to make an objective decision. Notice that this approach is not always available as it is not frequent that the statistical assumptions required can be satisfied or in any case it is not an easy approach to use as sometimes we satisfy these assumptions only with great difficulty (required assumptions are usually data independence and normality). Another issue may be a highly non-stationary behaviour or also the quantity of data available, as a matter of fact it may be expensive and not efficient at all to retrieve the desired dataset. Nevertheless, confidence intervals can be obtained for the differences between means, variances, and distributions of different output variables of a simulation model and a system for each set of experimental conditions. These intervals can be used as the model range of accuracy for model validation, where the model range of accuracy is the confidence interval or region around the estimated difference between some function (e.g., the mean) of the model and system output variable being evaluated.

3.4.2 Warm-up Period and Independent Replications

As far as non-terminating simulations are concerned it is necessary to distinguish two phases, a transient phase (initial phase or *warm-up* phase) and a stationary phase that is generally reached, as explained above, for $t \to \infty$ where t represents the simulated time.
Statistics gathered during the transient phase depend on the initial state therefore they are of no interest for the statistical analysis of the obtained results. One of the main issues of non-terminating simulations is exactly when to end them, this is because in practice we cannot rely on the stationarity condition $t \to \infty$ and it is necessary to understand when the stationary regime was reached because it is there that we will collect the statistics of interest. In order to identify the warm-up period and consequently the beginning of the stationary regime i.e. the point from which we will start to gather data, we can use Welch's graphical method. This method is presented below.

Welch's Graphical Method. Suppose we want to estimate the mean of a sequence of observations to be intended as random variables $Y_1, Y_2, Y_3, \dots$ in the stationary regime, in practice we want to estimate

$$\nu = \lim_{i \to \infty} \mathbf{E}[Y_i]$$

consequently, we will have a finite sequence of observations Y_i with $i \in [1, m]$ (where m is the number of gathered observations) and we will look for a value l that allow us to define:

$$\bar{Y}_l = \frac{\sum_{i=l}^{m} Y_i}{m - l}$$

where $\bar{Y}_l$ is the average of the random variables of observations from $l + 1$ to m, the hope is to obtain $\nu \sim \bar{Y}_l$. Notice that it is important to choose l accurately because if it is too small then $\bar{Y}_l$ will be influenced by the initial observations that are not stationary, whereas if it is too big (so maybe near to m) the number of random variables used to compute $\bar{Y}_l$ would be too small and the accuracy of ν estimation would be too low. To choose l we do what follows.

We process n pilot executions (with n possibly greater or equal to 5), each execution will gather m observations, so that Y_{ji} will be the random variable associated with the i-th observation of execution j with $1 \leq i \leq m$ and $1 \leq j \leq n$.

Now, we define the sequence made up of the averages,

$$\bar{Y}_i = \frac{\sum_{j=1}^{n} Y_{ji}}{n}$$

this will reduce the variance of the process. At this point, we create a window of width $w \leq \lfloor m/4 \rfloor$ and we construct a sequence of $m - w$ random variables defined as the averages of the w previous $\bar{Y}_i$, the current one and the w following $\bar{Y}_i$:

$$\bar{Y}_i' = \frac{\sum_{s=i-w}^{i+w} \bar{Y}_s}{2w+1}$$

Notice that for random variables from 1 to w the average is computed as

$$\bar{Y}_i' = \frac{\sum_{s=1}^{2i-1} \bar{Y}_s}{2i-1}$$

In practice we are computing a moving average on the process composed by the random variables given by the mean of a same observation over different runs. Finally, we draw a graph of the $\bar{Y}_i'$ observations for $i = 1, \ldots, m - w$ and we choose a value for l, beyond which the series of $\bar{Y}_i$ seems to be more stable.

As anticipated before we know that non-terminating simulations do not have a natural stopping point. However, it is necessary to decide when to interrupt the simulation and with it the collection of statistics. Intuitively, what would be ideal is to interrupt the simulation when the quality of the obtained estimates is reasonably good. In practice, the basic idea to evaluate the quality of the estimates consists of executing independent runs and then compare their estimates. The empirical rule states to make the duration of a simulation lasts the time sufficient to do so that the number of repeated execution give estimates that together approximately represent a random Gaussian variable. But it is necessary to pay attention not to extend excessively a simulation as this might provoke inconsistency in the obtained results because of time and resources consumption, of numeric instability of floating-point and also because of the periodicity of the random number generators.

Summarizing, to avoid initialization bias we use Welch's graphical method, then in order to produce estimates of quantities of interest and their respective confidence intervals we can use the method of independent replications. Basically, it consists of computing the average over the runs of the observations for each quantity of interest, excluded the ones cut off with Welch's graphical method, and then we use the results to construct confidence intervals.

Chapter 4

Deriving Product-Form Solutions

As we have already anticipated in the previous chapters, when studying a queueing system, we are particularly interested in its product-form expression, if one exists. Unfortunately, product-form solutions often lead to complex and time-consuming computations. In fact, historically, in order to retrieve a product-form expression one typically had to guess that such an expression existed and then verify that the Kolmogorov's equations of the underlying Markov process were satisfied. This task becomes more and more difficult to tackle as the complexity of the system grows. For this exact reason, researchers have been trying to apply different methods to derive the same result.
The approach we are going to present in this chapter is the one developed by P.G. Harrison in [1]; the approach is based on the so called *Reversed Compound Agent Theorem* (RCAT). First, we are going to present what is a Stochastic Process Algebra and in particular, we are going to present the main features of PEPA (*Performance Evaluation Process Algebra*). Secondly, we are going to show RCAT itself and one of its extensions that were useful to our aim and finally we will propose an example of use.

4.1 Performance Evaluation Process Algebra

PEPA is a stochastic process algebra. Process algebras [20] emerged firstly as a modelling technique for the functional analysis of concurrent systems. Stochastic process algebra was then proposed as a tool for performance and dependability analysis and modelling. The main attractive feature of SPA was the fact that compositionality was explicit. Compositionality means that a complex expression is actually determined by the meanings of its constituent expressions and the rules used to combine them, in practice it is the ability to model a system as the interaction of its subsystems. In fact, this principle was applied to systems and this resulted particularly useful because when a system is composed of interacting components we are now able to model interactions and components separately.

PEPA is a stochastic process algebra designed for modelling computer and communication systems by J. Hillston as presented in [7]. This formalism can be used for studying quantitative and qualitative properties of the model under analysis. In particular, it provides an elegant syntax for expressing continuous-time Markov processes in a compositional way. In fact, PEPA was the first language to be developed with the explicit intention of generating Markov processes that could be solved numerically for performance evaluation. PEPA peculiarity, with respect to other stochastic process algebras, consists in the fact that it associates a random variable, that will represent duration, with every action. These random variables are assumed to be exponentially distributed and this is the connection between this process algebra and Markov processes.

4.1.1 Syntax

PEPA is based on three main ingredients: *components* that are the active units within the system, *activities* that capture the actions of those units and *cooperation* that expresses the interaction between components.

Models are constructed from components. These components perform activities and, as we have already said, each activity has an associated duration described by an exponentially distributed random variable. Moreover, each activity has an *action type* α and an *activity rate* r. Each system action is uniquely typed and there is a countable set $\mathcal{A}$ of all possible types, activities with the same action type are different instances of the same action by the system. Since an exponential distribution is uniquely determined by its parameter, the duration of an activity is represented by a single real number parameter, the so-called *activity rate*. This rate may assume the value of any positive real number of the distinguished symbol $\top$ that has to be read as *unspecified*. We now present some notations, also summarizing the concepts presented so far.

- $\mathcal{A}$ is the set of all action types (τ included, that denotes the type *unknown*)
- $\mathcal{R}^+$ is the set of all positive real numbers, including $\top$
- $\mathcal{A}ct = \mathcal{A} \times \mathcal{R}^+$ is the set of all activities

Therefore, an activity is denoted as

$$a = (\alpha, r)$$

with $a \in \mathcal{A}ct, \alpha \in \mathcal{A}$ and $r \in \mathcal{R}^+$. Typically, components are denoted by upper-case letters, activities by lower-case letters and action types by Greek letters.

The combinators of the language allow expressions, or terms, to be constructed defining the behaviour of the components via the activities they undertake and the interactions between them. The behaviour of each component and consequently of the whole system is described with the following semantic.

- *Prefix*

  $$C = (\alpha, r).S$$

  This means that component C carries out activity (α, r) and then it subsequently behaves as component S. A delay is thus inherent in each activity in the model and the timing behaviour of the system is captured, moreover since the duration is a random variable also the uncertainty on how long an action will take is represented.

- *Choice*

  $$C = S_1 + S_2$$

  This means that component C represent a system which may either behave as component S_1 or as component S_2. In any case, C enables all the current activities both of S_1 and of S_2 but only the activity that completes first will decide the behaviour of C and the other component of the choice will be discarded. Two facts need to be highlighted: the first one is that the continuous nature of the probability distributions ensures that the probability that S_1 and S_2 both complete an activity at the exact same time is 0; the second one is that the *choice* combinator implicitly assume that S_1 and S_2 are competing for the same (implicit) resource therefore, the combinator $+$ represent the competition between components.

- *Cooperation*

  $$C = C_1 \bowtie C_2$$

  This is an *indexed family of combinators*, one for each possible set of action types $L \subseteq \mathcal{A}$. L is the so-called *cooperation set* and defines the action types on which the components involved must synchronize. $\bowtie$ assumes that each component proceeds independently with any activities whose types do not occur in L. Whereas, activities with action types in L require the simultaneous involvement of both components in an activity of that type (notice that $\tau \notin L$); accordingly one component may become blocked waiting for the other component to be ready to participate. The cooperation, in practice, forms a new shared activity with the same action type as the two cooperating activities and with rate reflecting the rate of the *slower* participant. If one of the two cooperating activities has an unspecified rate in a component, then the component is said to be passive with respect to that action type.

- *Parallel Composition*

$$C = C_1 || C_2$$

This represents exactly cooperation when $L = \emptyset$ which means that components are allowed to proceed concurrently without interactions between them. It exactly as if we wrote $C_1 \underset{\emptyset}{\bowtie} C_2$.

- *Assignment*

$$A \overset{def}{=} P$$

This means that a *constant* component A behaves as component P.

A race condition governs the dynamic behaviour of a model whenever more than one activity is enabled, this means that when many activities try to proceed only the fastest will succeed. Consequently, the probability that a particular activity completes will be given by the ratio of the activity rate of that activity to the sum of the activity rates of all the enabled activities.

As already anticipated, a component may be passive with respect to an action type; this means that all the activities of that type enabled by the component will have an unspecified rate $\top$. These activities must be shared with another component so that the other component can determine the rate of this shared activity. If more than one activity of a given passive type can be simultaneously enabled by a component, each unspecified activity rate must be assigned a weight; weights are natural numbers used to determine the relative probabilities of the possible outcomes of the activities of that action type. If no weights are assigned we assume that multiple instances have equal probabilities of occurring.

When we have $P = (\alpha, r).S$ we say S is a derivative of P if, as in this case, there only one activity between the two then we say that S is a *one-step derivative* of P. The *derivative set* of a PEPA component P, denoted $ds(P)$ is defined as the smallest set of components such that if $P \overset{def}{=} P_0$ then $P_0 \in ds(P)$ and if $P_i \in ds(P) \wedge \exists\ a \in \mathcal{A}ct(P_i)$ such that $P_i = a.P_j$ then $P_j \in ds(P)$. In practice, the derivative set of a component P is the set of all PEPA components representing all the reachable states of the system by P.

Given a PEPA component and its derivative set, its derivation graph is the labelled directed multigraph whose set of nodes is the derivative set of the component and whose set of arcs is defined by the activities connecting the states of the graph.

4.2 Reversed Compound Agent Theorem

The reversed compound agent theorem enables us, through the use of PEPA, to model and interpret in a simpler way the interactions and synchronization actions among different components of a system model. Using this approach, we are able to retrieve product-form solutions for the system under study without the need of solving the, computationally heavy, global balance equations. In this section we are going to present the first version of this theorem omitting the details of the proof, then we are going to show some of the extensions developed for this approach and finally we are going to show an operational example in which we try to obtain the same product-form for a simple queueing system using Jackson's theorem and RCAT.

RCAT exploits an abbreviated PEPA syntax in which we consider the prefix combinator, the cooperation between two agents (components) and the constant agent defined by the assignment combinator. As we have seen in standard PEPA the shared actions of a cooperation occur at the rate of the slowest component; nevertheless, now, we require that for each action type in the cooperation set, exactly one agent is *passive* and concretely its synchronising action has rate $\top = \infty$. This, basically means that the passive agent actually waits for the other one. The set of actions, which an agent P may next engage in, is called the set of *current actions* and when the system is behaving as agent P these are the actions that are *enabled*. In addition, the derivation graph, formed by syntactic PEPA terms at the nodes, with arcs representing the transition between them, determines the underlying Markov process of an agent P. The transition rate between two agents, C_i and C_j, denoted $q(C_i, C_j)$, is the sum of the action rates labelling arcs connecting C_i and C_j.

Here, we also need to introduce relabelling, which preserves the semantic but will be useful to define the reversed process of cooperations: $P\{x \leftarrow y\}$. describes agent P in which all occurrences of symbol y have been replaced by x; y may be an action type or also a rate. Notice that from now on we will refer to all agents as simple if they are defined only through prefixes and assignments, whereas we will say an agent is compound if it contains at least one instance of the cooperation combinator.

4.2.1 Rates of the Reversed Actions

It is rather simple to find a PEPA agent definition $\bar{X}$ that has the derivation graph with arrows in the opposite direction with respect to those of a given agent X. What it is not trivial is to find the appropriate rates of the reversed actions to make $\bar{X}$ the actual reversed process of X as defined in section 1.4.

For simple agents, we can directly analyse the state transition graph of the Markov process and after determining the reversed graph $\bar{G}$, we proceed as follows:

1. We use the conservation of outgoing rate to equate $q_i^R = q_i$ for all states $i \in \mathcal{S}$

2. We find a covering set of cycles, we may choose the set of all cycles or the set of all minimal cycles in G. If $\mathcal{S}$ is infinite then the set of cycles is also infinite but in practice cycles often repeat in a parametrised way so that the number of cycles to be considered will be finite.

3. For each (parametrised) cycle, we apply Kolmogorov's criteria i.e. we equate the known product of the rates around the cycle in G with the symbolic product of the rates around the reversed cycle in $\bar{G}$. Some of the reversed rates may be already known or imposed by the previous steps or from previously considered cycles in this step, which simplifies the resulting equations.

The result will be a system of possibly non-linear equations that uniquely determines the reversed rates in $\bar{G}$. This must be so because of the necessity and sufficiency of Kolmogorov's criteria and the uniqueness of the equilibrium state probabilities in an ergodic Markov chain. Notice that one of the reasons for analysing simple agents is to provide base cases for a compositional analysis of larger Markov chains. In fact, any continuous Markov chain can be described using only simple agents, however, the fact that an agent can perform multiple actions leading to the same derivative causes multiple arcs in the derivation graph and consequently also between two states in the transition graph of the underlying Markov chain. This does not create any issue as we can always determine the total reversed rate between any two states with multiple arcs between them, using one of the known methods. However, we need to consider multiple actions individually in cooperations. This is because an agent may have several actions leading to the same derivative that synchronise with different actions in a cooperating component. For example, this can happen when a queue witnesses a departure of a job, this job may join other queues or also leave the system.

In the reversed cooperation, the portion of the total reversed rate allocated to each individual reversed arc is crucial therefore a rule is needed. The rule we use is the following:

Definition 4.1. The reversed actions of multiple actions (a_i, λ_i) for $1 \leq i \leq n$ that an agent P can perform, which lead to the same derivative Q, are respectively

$$\left(\bar{a}_i, \left(\frac{\lambda_i}{\lambda}\right)\bar{\lambda}\right)$$

where $\lambda = \lambda_1 + \cdots + \lambda_n$ and $\bar{\lambda}$ is the reversed rate of the one-step, composite transition with rate λ in the Markov chain, corresponding to all the arcs between P and Q.

In other words, the total reversed rate is distributed amongst the reversed arcs in proportion to the forward transition rates.

4.2.2 Compound Agents

Under appropriate conditions, the reversed agent of a cooperation between two agents P and Q is a cooperation between the reversed agents of P and Q, after some reparametrisation. Before formally showing this result (4.1), we first need to define some new notation.

Definition 4.2. The subset of actions types in a set L which are *passive* with respect to a process P (i.e. are of the form $(a, \top)$ in P) is denoted by $\mathcal{P}_P(L)$. The set of the corresponding active action types is denoted by $\mathcal{A}_P(L) = L \setminus \mathcal{P}_P(L)$.

Last thing to do before considering the following theorem we need to syntactically transform the agent under analysis so that every occurrence of a passive action $(a, \top)$ is relabelled as $(a, \top_a)$; this guarantees that every passive action rate is uniquely identified with exactly one action type.

Theorem 4.1 (Reversed Compound Agent Theorem)**.** Suppose that the cooperation $P \underset{L}{\bowtie} Q$ has a derivation graph with an irreducible subgraph G. Given that

1. every passive action type in $\mathcal{P}_P(L)$ or $\mathcal{P}_Q(L)$ is always enabled in P or Q respectively (i.e. enabled in all states of the transition graph);
2. every reversed action of an active action type in $\mathcal{A}_P(L)$ or $\mathcal{A}_Q(L)$ is always enabled in $\bar{P}$ or $\bar{Q}$, respectively;
3. every occurrence of a reversed action of an active action type in $\mathcal{A}_P(L)$ (respectively, $\mathcal{A}_Q(L)$) has the same rate in $\bar{P}$ (respectively, $\bar{Q}$)

then the reversed agent $\overline{P \underset{L}{\bowtie} Q}$ with derivation graph containing the reversed subgraph $\bar{G}$, is

$$\bar{R}\{(\bar{a}, \overline{p_a}) \leftarrow (\bar{a}, \top) | a \in \mathcal{A}_P(L)\} \underset{L}{\bowtie} \bar{S}\{(\bar{a}, \overline{q_a}) \leftarrow (\bar{a}, \top) | a \in \mathcal{A}_Q(L)\},$$

where

$$R = P\{\top_a \leftarrow x_a | a \in \mathcal{P}_P(L)\},$$
$$S = Q\{\top_a \leftarrow x_a | a \in \mathcal{P}_Q(L)\},$$

$\{x_a\}$ are the solutions (for $\{\top_a\}$) of the equations

$$\begin{aligned} \top_a &= \overline{q_a}, \quad a \in \mathcal{P}_P(L) \\ \top_a &= \overline{p_a}, \quad a \in \mathcal{P}_Q(L) \end{aligned}$$

and $\overline{p_a}$ (respectively, $\overline{q_a}$) is the symbolic rate of action type $\bar{a}$ in $\bar{P}$ (respectively $\bar{Q}$)

As far as conditions 1. and 2. are concerned it is rather straightforward to check if the passive actions are enabled in every state of the two agents P and Q and if the reversed active actions are also always enabled. Condition 3. may require a little more work if the rates of the reversed processes are not known. Nevertheless, we can assume that condition 3. holds and solve the equations to find the reversed rates (which will be precisely the traffic equations for the internal flow butt we will refer to them as *rate equations* because we may deal with more general models than queueing networks) and if these equations have a solution, we see that the condition is satisfied. The existence of a rate equations solution ensures the product-form of the model which follows naturally once the reversed process and its rates have been defined.

Propagation of Instantaneous Transitions. In [2] the authors have proved that RCAT can be applied also to models that involve propagating synchronizations. To understand what we mean we briefly need to discuss the concept of G-network. G-networks [21] are a class of product-form queueing networks in which both positive and negative customers are allowed; the first ones behave as we are used to in traditional queueing networks whether the negative ones at arrival to a station delete a positive customer, if any is present, or vanishes otherwise. In [22] it was shown that these negative customers may act as triggers, meaning that they can move a customer from a non-empty queue to another one. The class of G-networks has been then further investigated to comprehend chains of instantaneous state changes that can be modelled as the propagation of instantaneous transition as shown in [23, 24]. Accordingly, we write

$$P = (a \to b, \top).Q$$

to denote a passive action with type a that takes process P to Q and instantaneously synchronizes as active on type b. The rate at which the transition synchronizes on type b is well-defined and equal to the reversed rate of the passive action with type a. As we shall see in the next chapter, this property will be extremely useful for the modelling of the system we studied.

Example. We now give an example of an RCAT application, showing also that the result obtained through this alternative method is exactly the same if we were

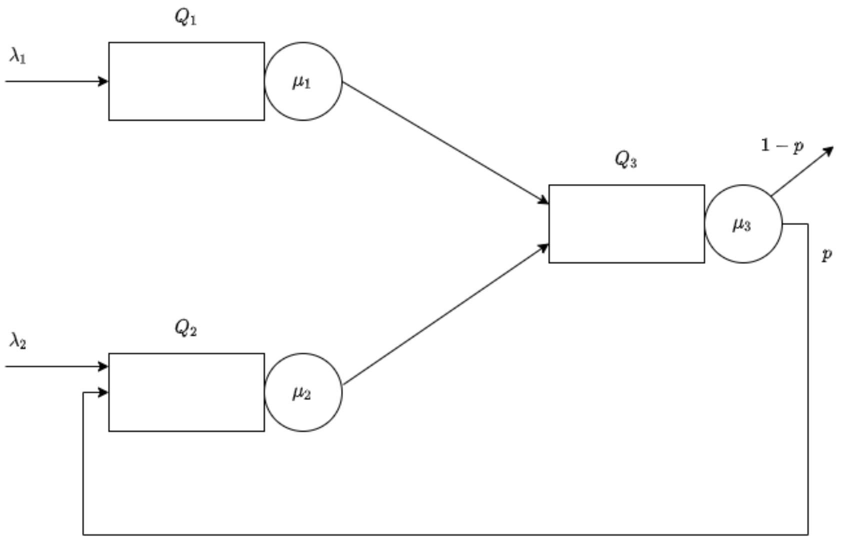


Figure 4.1: Queueing Network under study.

applying Jackson's theorem. The system we are going to analyse is the queueing network showed in fig. 4.1 and its PEPA specification is as follows.

$$\begin{aligned}
&Q_{1,n} = (a_{01}, \lambda_1).Q_{1,n+1} \quad n \geq 0\\
&Q_{1,n} = (a_{13}, \mu_1).Q_{1,n-1} \quad n > 0\\
&Q_{2,n} = (a_{32}, \top).Q_{2,n+1} \quad n \geq 0\\
&Q_{2,n} = (a_{02}, \lambda_2).Q_{2,n+1} \quad n \geq 0\\
&Q_{2,n} = (a_{23}, \mu_2).Q_{2,n-1} \quad n > 0\\
&Q_{3,n} = (a_{23}, \top_{23}).Q_{3,n+1} \quad n \geq 0\\
&Q_{3,n} = (a_{13}, \top_{13}).Q_{3,n+1} \quad n \geq 0\\
&Q_{3,n} = (a_{32}, \mu_3 p).Q_{3,n-1} \quad n > 0\\
&Q_{3,n} = (a_{30}, \mu_2(1-p)).Q_{3,n-1} \quad n > 0
\end{aligned}$$

with $(W_0 \underset{a_{13}}{\bowtie} W_3)||(W_1 \underset{a_{23},a_{32}}{\bowtie} W_3)$.

In order to verify that our model satisfies the three structural conditions of RCAT we draw part of the Markov chain underlying each queue, highlighting the different transitions between states (fig. 4.2).

At this point we can state that the first two structural conditions are satisfied as all passive actions are enabled in every state and the reversed active actions are also enabled. In fact, by inspection of the image above we can see there is an outgoing arc for every state for every passive action and an incoming arc for every state for every reversed active action. To verify the third condition we need

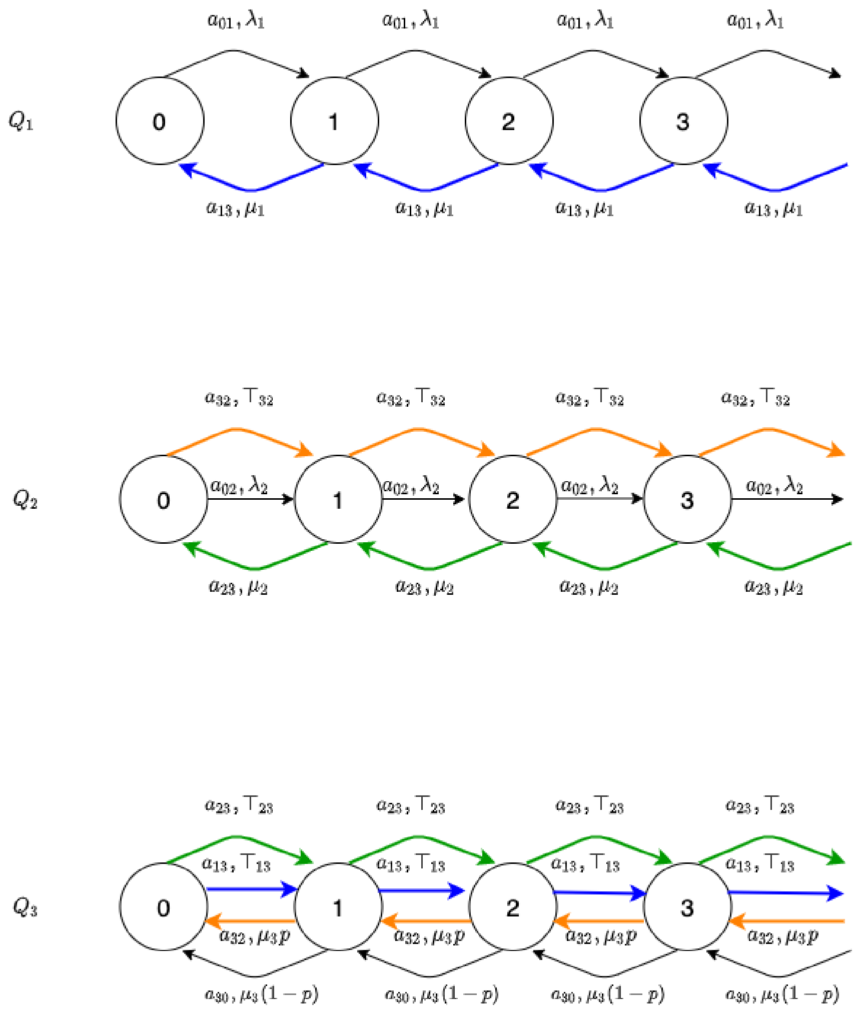


Figure 4.2: Graphical representation of the PEPA model of the system showed in fig. 4.1

to retrieve traffic conditions in order to check that the rate of each reversed active action is constant. Considering the x_i with the same subscripts as action labels as the rates of those actions, we have the following traffic equations:

$$\begin{cases} x_{13} = \lambda_1 \\ x_{23} = \lambda_2 + x_{32} \\ x_{32} = (x_{13} + x_{23})\dfrac{\mu_3 p}{\mu_3 p + \mu_3(1-p)} \end{cases} \rightarrow \begin{cases} x_{13} = \lambda_1 \\ x_{23} = \lambda_2 + x_{32} \\ x_{32} = (x_{13} + x_{23})p \end{cases}$$

these will give the following results:

$$x_{13} = \lambda_1, \quad x_{23} = \frac{p\lambda_1 + \lambda_2}{(1-p)}, \quad x_{32} = \frac{\lambda_1 + \lambda_2}{(1-p)}$$

Since the reversed rates have been found and they are constant, RCAT tells us not only a product form exists but also which one it is; as a matter of fact RCAT states that the product form is given by the stationary distribution at the queues where all the $\top_i$ have been replaced by the correspondent reversed rates. In fact, we have:

$$\pi_1(n_1) = \left(\frac{\lambda_1}{\mu_1}\right)^{n_1}$$

$$\pi_2(n_2) = \left(\frac{p\lambda_1 + \lambda_2}{(1-p)\mu_2}\right)^{n_2}$$

$$\pi_3(n_3) = \left(\frac{\lambda_1 + \lambda_2}{(1-p)\mu_3}\right)^{n_3}$$

Now, to check our result is true we retrieve the product form using Jackson's theorem, meaning again traffic equations, so we will have:

$$\begin{cases} e_1 = \lambda_1 \\ e_2 = \lambda_2 + e_3 p \\ e_3 = e_1 + e_2 \end{cases}$$

which results will be

$$e_1 = \lambda_1, \quad e_2 = \frac{p\lambda_1 + \lambda_2}{(1-p)}, \quad e_3 = \frac{\lambda_1 + \lambda_2}{(1-p)}$$

. Given that by Jackson's theorem we would have

$$\boldsymbol{\pi}(n_1, n_2, n_3) = \left(\frac{e_1}{\mu_1}\right)^{n_1}\left(\frac{e_2}{\mu_2}\right)^{n_2}\left(\frac{e_3}{\mu_3}\right)^{n_3}$$

Substituting the just obtained results we will obtain the same result as before, meaning we did right.

Chapter 5

The model

Using the Reversed Compound Agent Theorem several well-known product-form have been proved again in a modular and compact way. This is exactly what we have done and we are going to present our work in this chapter. We are going to present the model devised by D. Gates and M. Westcott in [25, 4] that represents our starting point, then we will show our derivation of its product-form expression. Finally, we will present the simulation we built upon this model and how we validated it.

5.1 The Starting Point

In [25, 4], the authors devised a particular Markov process for modelling the process of wheel replacement and restoration on the trains of Queensland Railways. Because of continuous use trains' wheels get worn, when the wear becomes excessive the wheel is removed and replaced from stock, whereas worn wheels get reprofiled by grinding and go into stock so that they can be used again in trains that need wheels with a smaller diameter. Once a wheel becomes too small it is discarded. Notice that reprofiling leads to wheels of various sizes both on trains and in stock. The aim of the original project was to optimize the management of the system given the associated costs, in this section we will discuss the model itself and its probabilistic aspects. In fact, notice that randomness resides in the random wear process of the wheels that creates a random demand for wheels from stock. The following assumptions were made about the problem:

1. There are M size classes of wheel, labelled $1, \ldots, M$. Type 1 wheels are the new ones,

2. Any train can have only a single class of wheel on it, so there are effectively M types of train as well, with a type i train ($i = 1, \ldots, M$) having only size

i wheels for all time.

3. The minimum grinding needed to reprofile a worn size i wheel reduces its size by one size class, though we can choose to reduce it further. So size M wheels cannot be ground down further, consequently they are discarded once worn.

4. There is an unlimited supply of new wheels, which are not held in stock (meaning we always have $x_1 > 0$), and unlimited storage for the stock of other wheels.

5. Worn wheels occur on type i trains in a Poisson process of rate q_i, independently for each type of train.

6. A train cannot be left idle (since the number of trains is just adequate for peak hour demand), so must have its worn wheels replaced immediately.

The stock level at time t is a random process $\boldsymbol{X}(t) = (x_2(t), \ldots, x_M(t))$, where $x_i(t) \quad (i = 2, \ldots, M)$ is the number of size i wheels in stock at time $t > 0$. Changes of state are caused by the arrival of a train with worn wheels.

- If this train is of type i, it will discard a worn size i wheel which is ground to a size $i+1$ wheel; thus $x_{i+1}(t) \to x_{i+1}(t) + 1$.

- Provided $x_i(t) > 0$, a size i wheel will be taken from stock and put on the train; thus $x_i(t) \to x_i(t) - 1$.

The interesting problem in building the model is: what happens if $x_i(t) = 0$? The authors have chosen the following strategy. The smallest wheel in stock greater than size i is taken and it is ground it down to a size i wheel. If there are no wheels in stock of sizes $2, \ldots, i-1$ then a new wheel is taken and ground it down to size i. Notice that there is one case in which on the arrival of a train with worn wheels no state changes take place and this is the case in which the incoming train has worn wheels of size M and there are no size M wheels to use for replacement so new ones must be taken. In chapter 2 and 6 of [8] the author describes some queueing and migration models that may resemble the process under study, nevertheless some important differences must be underlined:

1. transition rates depend on the state of the system not only through the number at the origin and at the destination but also through all the states between them.

2. transition rates effectively depend on the destination rather than the origin, unlike most queueing models. In fact, one could think of the system as one of a series of queues, where queue k is generating its own input at rate q_{k-1} and taking it from customers waiting at some earlier queue.

The authors were particularly interested in the equilibrium distribution, if any, of $\boldsymbol{X}(t)$. Their strategy was to dynamically reverse $X(t)$ and to apply a generalization of the classical Kolmogorov cycle. Once proved that the generalized Kolmogorov criteria held for $\boldsymbol{X}(t)$ i.e. the process was effectively dynamically reversible they deduced the following result.

Theorem 5.1. The Markov process $\boldsymbol{X}(t)$ on $\mathbb{Z}^{M-1}$ with infinitesimal transition rates defined as follows[1]:

$$q_{jk} = q_{k-1} \cdot I(x_j > 0, x_{j+1} = \cdots = x_{k-1} = 0)$$

has an equilibrium distribution if ans only if $\rho_i \equiv q_i/q_M < 1 \quad (i = 1, \ldots, M-1)$, given by

$$\pi_X(\boldsymbol{x}) = \prod_{i=2}^{M} (1 - \rho_{i-1}) \rho_{i-1}^{x_i} \tag{5.1}$$

The derivation of this product-form is rather complicated, it is enough to consider the fact one should exploit and prove to fulfill the detailed balance conditions and Kolmogorov criteria. This is the reason why in the following section we propose an alternative approach.

5.2 An Alternative Approach

In this section, we revise the model of Westcott and Gates and provide an alternative proof of its product-form solution. First, we give an alternative and more general description of the model, basically changing only some notation. We can think of this model as if it was implementing a sender-based policy for migration strategy in a distributed system. The model is defined as follows.
We consider a system with K warehouses numbered from 1 to K and an always full warehouse denoted by the number 0. The space state of the warehouses is $\mathcal{S}_K = \mathbb{N}^K$, with $K \in \mathbb{N}$, $K \geq 1$ and the state of the system can be described by vector $\boldsymbol{n} = (n_1, \ldots, n_K) \in \mathcal{S}_K$. With an abuse of notation we define vector e_i, $i = 1, \ldots, K$ as the vector filled with zeros except for a 1 in position i. Each warehouse is associated with a server that behaves as follows:

- If warehouse i is non-empty then, after an exponentially distributed time with rate μ_i, it sends to another warehouse, say warehouse j, the object it is using with a probability p_{ij} specified by the routing probabilities. After the disposal of the object, it retrieves a new one from its own warehouse. Therefore, the state changes from $\mathbf{n}$ to $\mathbf{n} - e_i + e_j$ unless the object is disposed instead of being sent to warehouse j; in this case state changes from $\mathbf{n}$ to

[1]Notice that $I(A)$ is the indicator function of event A.

$\mathbf{n} - e_i$. Notice that we say server i is a provider of server j if $p_{ij} > 0$ and the probability of disposal is given by $1 - \sum_{j=1}^{K} p_{ij}$.

- If warehouse i is empty (notice that this can never be the case of warehouse 0) then after disposing the object it is using to say warehouse j, the server contacts one of its providers and tries to fetch a new object from it. If the contacted warehouse is empty, then its server applies again the same policy to its provider, this process terminates when a non-empty provider is found (eventually warehouse 0). If $k > 0$ is the provider with the non-empty warehouse then the state changes from $\mathbf{n}$ to $\mathbf{n} - e_k + e_j$, whereas if the non-empty warehouse is the never empty one then the state changes from $\mathbf{n}$ to $\mathbf{n} + e_j$. In both cases, warehouse j is chosen as above.

Notice that there may be no state change if the object from server i is disposed and the new one is taken from warehouse 0. Server 0 fetches objects from its always full warehouse and sends the one that has received the service needed to one of the other K warehouses with rate μ_0.
In figure5.1 we show the network configuration we started from, then we will give its PEPA specification and we will also show the graphical representation of the PEPA model underlying the processes in figure 5.2 so that we will be ready to apply RCAT. Notice that, for the moment, we consider the case of three warehouses but the reasoning can be extended to a tandem of queues with arbitrary finite length.

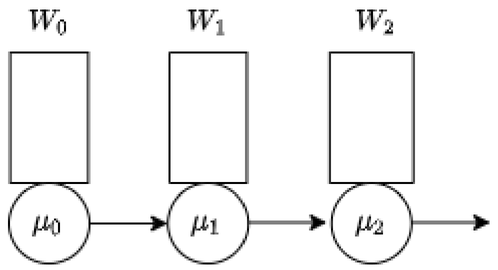


Figure 5.1: Representation of the system under study.

The PEPA specification of the system in fig.5.1 is as follows.

$$\begin{aligned}
W_0 &= (a_{01}, \top_{01}).W_0 \\
W_{1,n} &= (a_{01}, \mu_0).W_{1,n+1} \\
W_{1,n} &= (a_{12}, \top_{12}).W_{1,n-1} \\
W_{1,0} &= (a_{12}, \top_{12}).W_{1,0} \to (a_{01}, x_{12}).W_{1,0} \\
W_{2,n} &= (a_{12}, \mu_1).W_{2,n+1} \\
W_{2,n} &= (a_{20}, \mu_2).W_{2,n-1} \\
W_{2,0} &= (a_{12}, \mu_2).W_{2,0}
\end{aligned}$$

with $W_0 \underset{\{a_{01}\}}{\bowtie} W_1 \underset{\{a_{12}\}}{\bowtie} W_2$. The graphical representation of the PEPA model underlying this system is shown in figure 5.2.

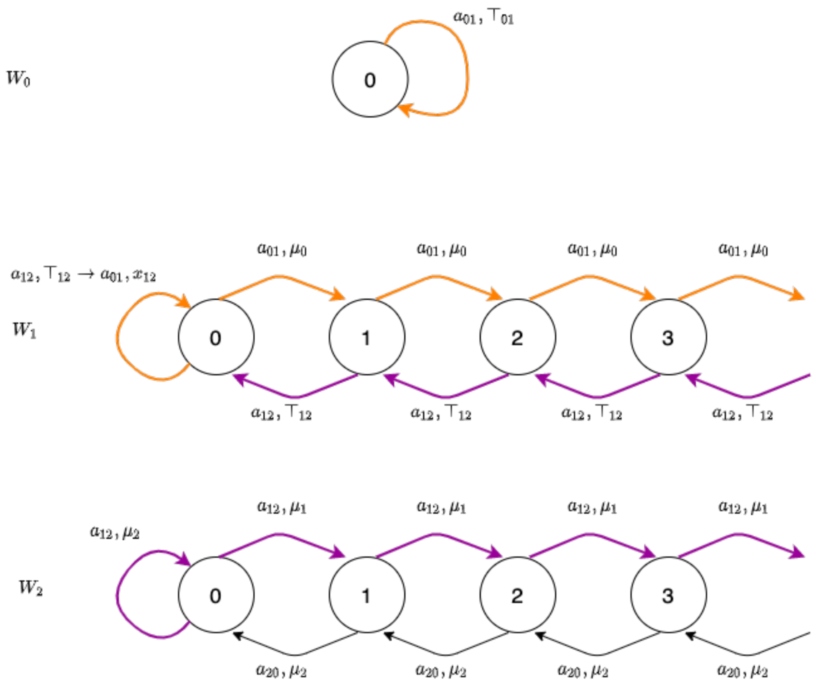


Figure 5.2: Graphical representation of the PEPA model underlying the queues of the system shown in fig. 5.1

By inspection of the diagrams in fig.5.2 we can state that the structural conditions of RCAT are satisfied as

- a_{01} is passive in W_0 so we need that every state of the process has an outgoing arc labelled a_{01} and this is true for W_0.

- a_{01} is active in W_1 so we need that every state of the process has an incoming arc labelled a_{01} and this is true for W_1 thanks to the transition propagating from state 0 into state 0.
- a_{12} is passive in W_1 so we need that every state of the process has an outgoing arc labelled a_{12} and this is true for W_1.
- a_{12} is active in W_2 so we need that every state of the process has an incoming arc labelled a_{01} and this is true for W_2.

Accordingly, we have that every passive action type in the cooperation set is always enabled in the processes W_i and every reversed action of an active action type in the cooperation set is always enabled in the reversed processes $\overline{W}_i$, as requested by the structural conditions. Now, we need to find exactly the reversed rates in order to verify the *rate condition*, i.e., we want every occurrence of a reversed action of an active action type in the cooperation set to have constant rate. First, we focus on x_{12} the reversed rate of action type a_{12}, remember that for Kolmogorov's criteria the product of the rates for every cycle in the reversed process must be the same of the product of the rates for every cycle in the forward process. Accordingly, if we consider the reversed process of W_2 we can state $x_{12} = \mu_2$; moreover since the reverse rate of a self-loop is equal to the forward rate we can safely replace the occurrences of $\top_{12}$ in process W_1 with the rate x_{12}. We notice that every occurrence of active transition labelled a_{12} has the same reversed rate. Then, to determine x_{01}, the reversed rate of action type a_{01}, we follow the same reasoning as above, so if we consider the reversed process of W_1 we can state $x_{01} = x_{12}$ and we replace $\top_{01}$ by x_{12}. Consequently, we obtain the graphical representation shown in figure 5.3.

From which we can derive the following product form

$$\pi(n_1, n_2) \propto \left(\frac{\mu_0}{x_{12}}\right)^{n_1} \left(\frac{\mu_1}{x_{12}}\right)^{n_2} = \left(\frac{\mu_0}{\mu_2}\right)^{n_1} \left(\frac{\mu_1}{\mu_2}\right)^{n_2} = \prod_{i=1}^{K} \rho_{i-1}^{n_i}$$

with $K = 2$ and where $\rho_i = \mu_i/\mu_K$. Adding the normalising constant we obtain exactly product-form 5.1, i.e.

$$\pi(\boldsymbol{n}) = \pi(n_1, n_2) = \prod_{i=1}^{2} (1 - \rho_{i-1})\rho_{i-1}^{n_i}.$$

To show the product-form holds true also for $K \geq 2$ we prove it by induction in the following.

Proof. We prove by induction that for all $K \in \mathbb{N}$

$$\pi(n_1, \ldots, n_K) = \prod_{i=1}^{K} (1 - \rho_{i-1})\rho_{i-1}^{n_i} \tag{5.2}$$

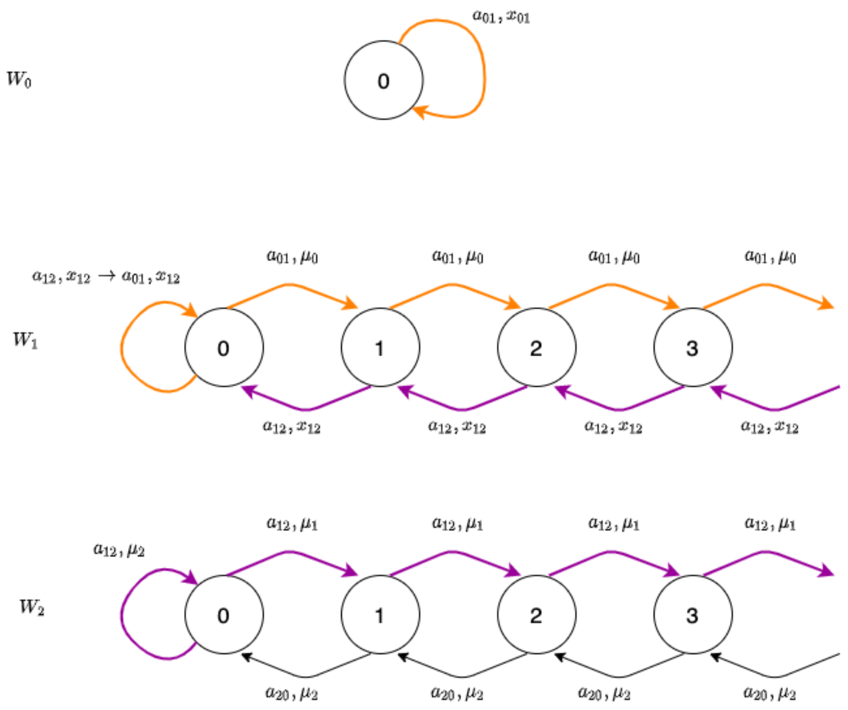


Figure 5.3: Graphical representation of the PEPA specification underlying the queues of the system shown in fig. 5.1, after the application of RCAT

Base case. When $K = 1$ we have $\pi(n_1) = (1 - \rho_0)\rho_0^{n_1}$ and when $K = 2$ we have $\pi(n_1, n_2) = (1-\rho_0)\rho_0^{n_1}(1-\rho_1)\rho_1^{n_2}$ and we proved this above through the use of the Reversed Compound Agent Theorem.

Induction hypothesis. Let $k \geq 2$ be given and suppose 5.2 is true for $K = k$, then by definition of product form we have what follows.

Induction step.

$$\begin{aligned}\pi(n_1, \dots, n_{k+1}) &= \pi(n_1, \dots, n_k)(1 - \rho_k)\rho_k^{n_{k+1}} \\ &= \prod_{i=1}^{k}(1 - \rho_{i-1})\rho_{i-1}^{n_i} \cdot (1 - \rho_k)\rho_k^{n_{k+1}} \qquad \text{by induction hypothesis} \\ &= \prod_{i=1}^{k+1}(1 - \rho_{i-1})\rho_{i-1}^{n_i}\end{aligned}$$

Thus, 5.2 holds for $K = k+1$ and the proof of the induction step is complete.

The conclusion is that the product-form solution proposed in 5.2 holds true for all $K \in \mathbb{N}$, with $K \geq 1$. □

Proof. We apply induction also from a modelling perspective.

Base case. When $K = 2$ RCAT holds as we have showed before.

Induction hypothesis. Let $k > 2$ be given and suppose RCAT holds for $K = k$ and we show the graphical representation of the PEPA specification underlying the k queues in fig.5.4.

Induction step. In order to show that RCAT still holds true for $K = k+1$, we add the station W_{k+1}, modify the diagram of W_k to allow the synchronization between the two queues and verify that structural conditions and the rate condition still hold. The graphical representation of $k + 1$ queues is given in fig. 5.5.

Now considering the graphical representation of W_{k+1} we observe that $a_{k,k+1}$ is active so we need that every state of the process has an incoming arc labelled $a_{k,k+1}$ and this is true; as $a_{k,k+1}$ is the only synchronizing action in W_{k+1} we state structural conditions hold. As far as the rate condition is concerned, we want every occurrence of a reversed action of an active action type in the cooperation set to have constant rate. For Kolmogorov's criteria the product of the rates for every cycle in the reversed process must be the same of the product of the rates for every cycle in the forward process. Considering the reversed process underlying W_{k+1} we can state

$$x_{k,k+1} = \mu_{k+1}$$

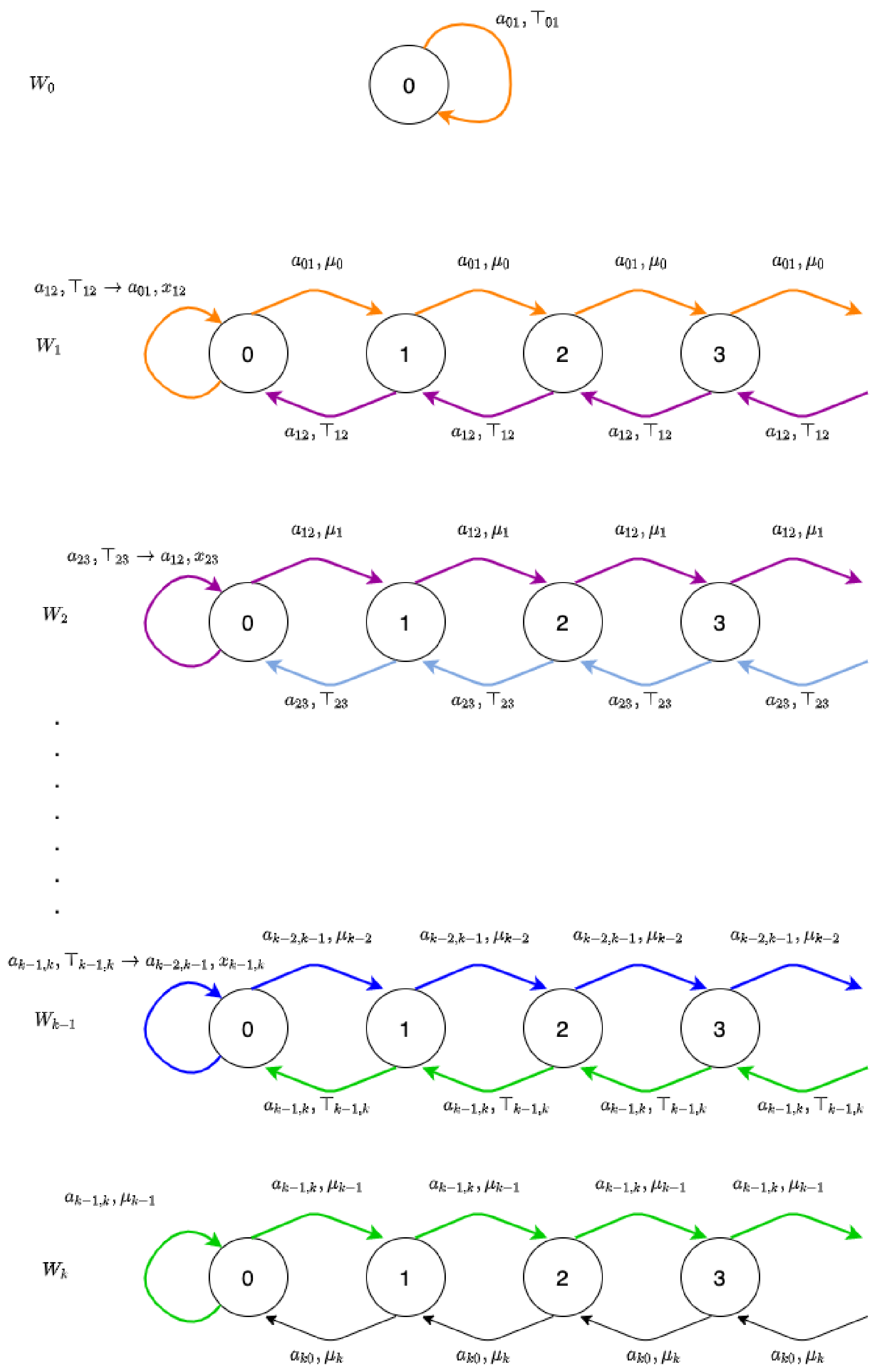


Figure 5.4: Graphical representation of the PEPA specification underlying the queues of the model under study with k stations.

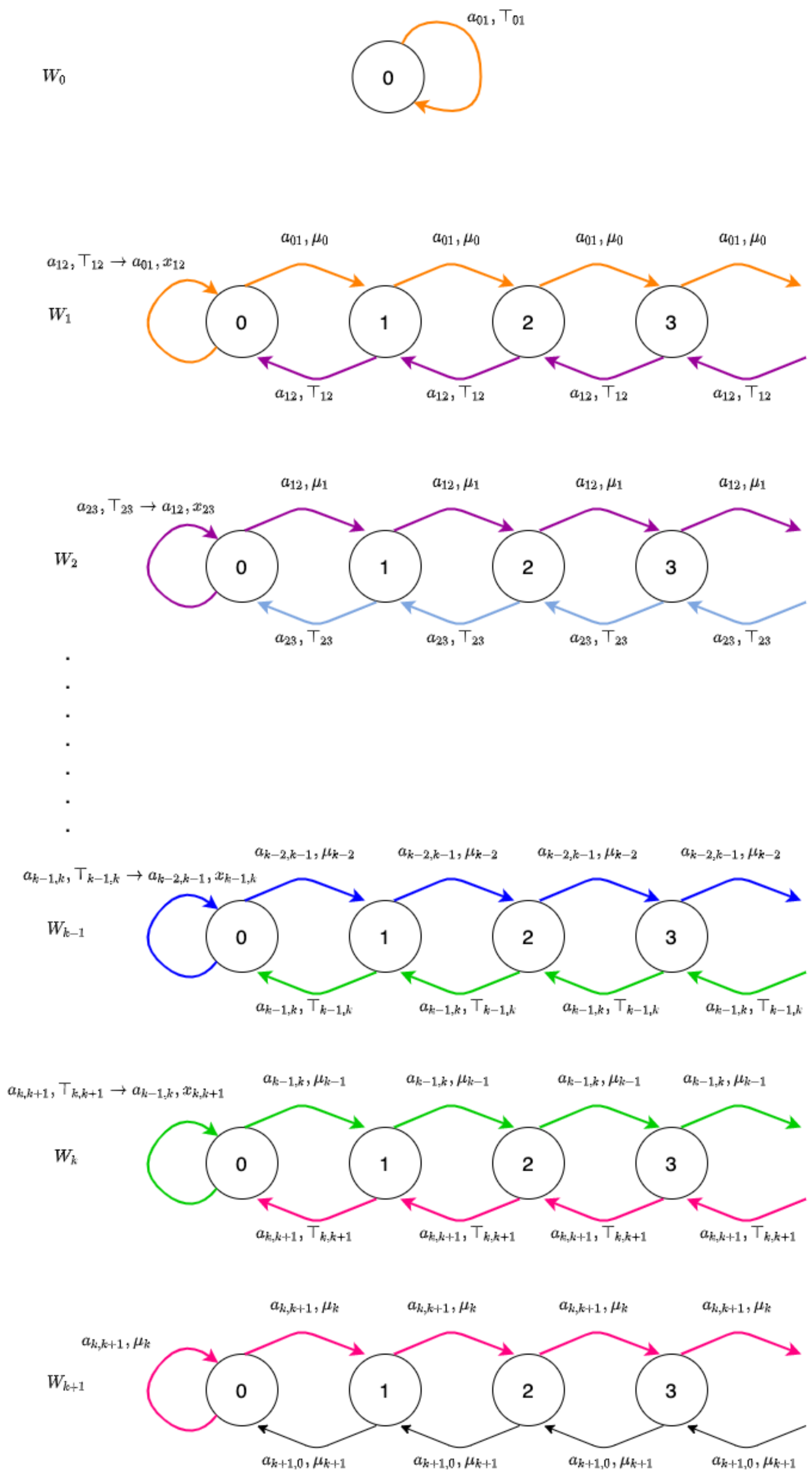

$a_{01}, \top_{01}$
W_0
0
a_{01}, μ_0
$a_{12}, \top_{12} \rightarrow a_{01}, x_{12}$
W_1
0 1 2 3
$a_{12}, \top_{12}$
a_{12}, μ_1
$a_{23}, \top_{23} \rightarrow a_{12}, x_{23}$
W_2
$a_{23}, \top_{23}$
$a_{k-2,k-1}, \mu_{k-2}$
$a_{k-1,k}, \top_{k-1,k} \rightarrow a_{k-2,k-1}, x_{k-1,k}$
W_{k-1}
$a_{k-1,k}, \top_{k-1,k}$
$a_{k-1,k}, \mu_{k-1}$
$a_{k,k+1}, \top_{k,k+1} \rightarrow a_{k-1,k}, x_{k,k+1}$
W_k
$a_{k,k+1}, \top_{k,k+1}$
$a_{k,k+1}, \mu_k$
W_{k+1}
$a_{k+1,0}, \mu_{k+1}$

Figure 5.5: Graphical representation of the PEPA specification underlying the queues of the model under study with $k+1$ stations.

and since the reverse rate of a self loop is equal to the forward rate we can replace all the occurrences of $\top_{k,k+1}$ in the diagram of W_k with rate $x_{k,k+1}$. Consequently, also the rate condition still holds true. Thus, RCAT holds also for $K = k + 1$ and the proof of the induction step is complete.

The conclusion is that RCAT holds true for all $K \in \mathbb{N}$, with $K \geq 1$, from which the product form showed before follows. □

5.3 The Simulation

The implemented project is an event-oriented simulation. We choose to develop the simulation program with the Python programming language. In particular, we used the SimPy library (version 4.0.1) [26] that is a process-based discrete-event simulation framework based on standard Python. The simulation is designed to be adaptable to different tests, in order to achieve this we devised a good parametrization of the input parameters. In fact, the system configuration can be completely defined by the user. This means we can arbitrarily specify the number of warehouses and corresponding servers, the rate of the exponentially distributed service times at each station and the probability routing matrix. In addition, we can specify the number of runs to be executed (by adding or subtracting seeds from the `seeds` vector) and also the run time (meaning simulated time) of the program. Finally, we can also choose if terminate the simulation after a certain amount of simulated time or after a certain number of events have been processed.

5.3.1 Implementation Details

The code has three main components.

- In the`__name__ == '__main__'` block we iterate over the list of seeds we defined as global variable. In the loop that represents the block of code in which we set and launch the replicated independent runs of the simulation we do what follows.
  - ➤ We create the array `state` representing the state of the warehouses, notice that the first position is always initialized to 100 and we will see later that it never decreases.
  - ➤ We define two important variables that we use for gathering statistics: `lastChange`, a list where we record the time of the last change of state for each warehouse and `timings`, a list of dictionaries (one for every warehouse) in which we record for how long each warehouse contains a

certain number of objects; for each one of these dictionaries the number of objects the is the key and the time is the value).

- We set a seed for this particular simulation.
- In the variable `env` we create the environment that will manage the simulation time as well as the scheduling and processing of events.
- We create a list of server objects (entities) that will be initialised with their own parameters (index and routing probabilities towards different servers).
- `env.process(stats_collector(...))` schedules the event represented by the function in input `stats_collector` that records the instantaneous value of the `state` variable. Notice that through the use of the keyword `yield` this event will be scheduled every second.
- Through the iteration of function `start_delayed(env, servers[i].behaviour(), delay)` for each server an event `behaviour` is scheduled in the environment `env` with an exponentially distributed delay [2].
- `env.run(until=end)` starts the simulation environment `until` a variable `end` that can be either an integer representing the units of time or an event.
- Finally we call the function `times_collector(...)` that deals with producing (for each run) a number of `.csv` files and plots (`.png` files) containing the comparison of the simulation results with the theoretical ones as far as the proportion of time spent in each state is concerned.

- The block concerning the definition of class `Server` which presents the following functions:
  - `behaviour(self)` that is the process execution method of each entity i.e. of each server. It has an infinite while loop, that can be changed, if needed, to stop the simulation after a certain amount of events. In the loop two methods are called `dispatch()` and `fetch()` and then through the use of the `yield` keyword another `behaviour(self)` event is scheduled after an exponentially distributed time.
  - function`fetch(self)` is the method in which the server picks its provider (its warehouse or another one) with the policy previously presented and then if the warehouse is not warehouse 0 its state is decreased by one

[2]previously computed with `random.expovariate(mean)`, where `mean` is the frequency parameter previously defined among the global variables.

and if the simulation time has passed the warmup interval it starts updating statistics.

- function `dispatch()` dispatch an object to another warehouse according to the routing probability consequently the state of the destination gets increased by 1, unless the object was not disposed.
- function `find_provider()` is called in `fetch()` to find a provider in case its own warehouse is empty. This is a recursive function that goes backwards on the warehouses until it finds a non-empty one (eventually warehouse 0)

• Among the global variables we can set we have:

  - `means`, the list of frequency parameter for each server.
  - `routing_probs`, the list of lists containing a list for each server that represents the routing probability from itself towards the other stations.
  - `warmup`, the variable representing the time after which the warm-up interval ended.
  - `seeds`, the list of seeds.

The topology of the simulated system is shown in figure 5.6.

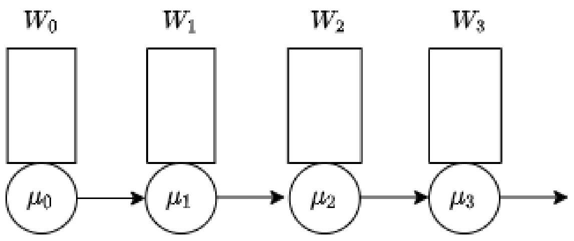


Figure 5.6: Tandem Configuration

with $\mu_0 = 1.0, \mu_1 = 2.0, \mu_2 = 3.0$ and $\mu_3 = 4.0$.

5.3.2 Warm-up Detection

In order to perform a steady-state analysis, we need to determine the warm-up phase of our simulation, as anticipated before, this is done to be able to gather statistics independent from the initialization bias.
The kind of data we are collecting up to now allows the simulation to step through events using simulated time, meaning we perform the simulation as fast as possible, not considering the wall-clock time.

To detect the warm-up period we used Welch graphical method. During the simulation we gather the state of the system, i.e. the number of objects present in each warehouse. A snapshot of the system is taken every second. On these output data we performed Welch graphical method.
Since we know that the initial state does not influence the existence of a steady-state for the simulation, but it does influence the time employed to reach it we decided to perform Welch method on the data gathered when starting from state

- $[100, 0, 0, 0]$ meaning W_0 has some jobs in it (by model definition we always have at least one job in W_0 and its state never actually changes) and W_1, W_2, W_3 are empty. For simplicity we will call this starting point "empty initial state".

- $[100, 100, 100, 100]$ meaning at the beginning of the simulation all the warehouses have some objects stored. For simplicity we will call this starting point "full initial state".

To briefly recap, Welch's graphical method consists of:

1. making n replications with run length m

2. determining the ensemble averages over replications for each observation

3. defining a Moving Average to smooth oscillations with high frequency; choose a window approx $w = \lfloor m/4 \rfloor$

4. plotting the moving average and choosing L to be the value beyond which our distribution seems to be in steady-state

We decided to perform 20 independent runs ($n = 20$), each run during a simulated hour ($m = 3600\ s$), meaning that the size of the window to define the moving averages was chosen to be $w = 900$. In case of empty initial state the resulting plot is the one of figure 5.7 and we identified the warm-up period in the first $800s$; whereas in case of full initial state the plot is presented in figure 5.8 and the warm-up period was identified in the first $1000s$.

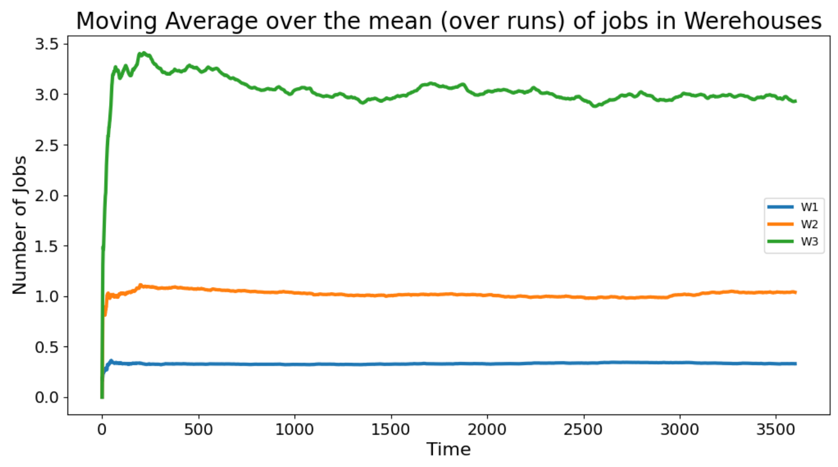


Figure 5.7: *Empty Initial State*, warm-up interval: first 800s.

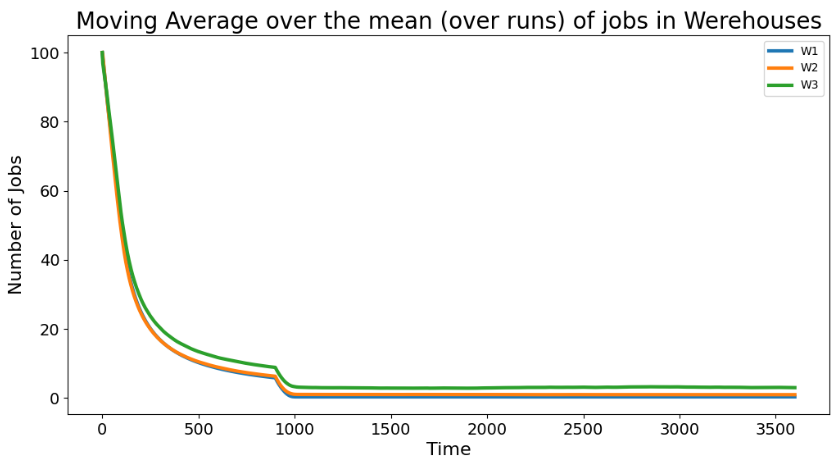


Figure 5.8: *Full Initial State*, warm-up interval: first 1000s.

After this brief analysis we decided to proceed using the initial state that caused the shorter warm-up phase and consequently to modify our simulation code so that it collected statistics only after the end of the warm-up interval.

5.3.3 Confidence Interval For Mean Number of Jobs

We know that from a theoretical point of view computing the mean number of jobs in each station can be done using the formula:

$$\overline{N}_i = \frac{\rho_{i-1}}{1-\rho_{i-1}} \qquad i = 1, \dots, K$$

where K is the number of warehouses. What we will show in this section is the theoretical results for each station, the results retrieved from the simulation and the corresponding confidence intervals. To obtain an average of the number of jobs in each station for each run we took a photograph of the system every second and we then computed the mean over the overall number of observations. To construct the confidence intervals we used the classical method for confidence intervals for the mean, using as estimate the mean over all the runs of our simulation. To show this data we will use some tables[3] for simplicity and because often plots were not clear enough, being the values so small.

Station	Analytical Res.	Sim. Estimate	Conf. Int.	Check
1	0.3333	0.3331	[0.3329, 0.3334]	IN
2	1.0	0.9993	[0.9982, 1.0005]	IN
3	3.0	2.9988	[2.9940, 3.0036]	IN

Table 5.1: Confidence Intervals for the mean number of jobs in each station.

5.3.4 Confidence Interval For Probability of being in a Certain State

We use the product form retrieved from this model as theoretical result and we use our simulation to retrieve the proportion of time spent in a certain state and construct on this data a confidence interval.
To retrieve data from the simulator we counted how much time a station was in a certain state and then we computed the ratio on the overall simulation time (minus the warm-up time). Then in order to construct confidence intervals we used the formula for confidence interval for the mean.

State	Analytical Res.	Sim. Estimate	Confidence Interval	Check
0	7.5e-01	7.500746e-01	[7.499319e-01, 7.502172e-01]	IN
1	1.875e-01	1.874754e-01	[1.873589e-01, 1.875919e-01]	IN
2	4.6875e-02	4.683089e-02	[4.677716e-02, 4.688463e-02]	IN
3	1.171875e-02	1.172061e-02	[1.168234e-02, 1.175888e-02]	IN
4	2.929688e-03	2.9248e-03	[2.906185e-03, 2.943415e-03]	IN
5	7.324219e-04	7.306659e-04	[7.232551e-04, 7.380766e-04]	IN
6	1.831055e-04	1.826603e-04	[1.779957e-04, 1.873248e-04]	IN
7	4.577637e-05	4.564159e-05	[4.392343e-05, 4.735975e-05]	IN

[3]In the tables headers we use the following abbreviations to make them fit into the page: "Analytical Res." for analytical result, "Sim. Estimate" for simulation estimate, "Conf. Int.".) for confidence interval

8	1.144409e-05	1.124516e-05	[1.019574e-05, 1.229458e-05]	IN
9	2.861023e-06	2.531976e-06	[2.021230e-06, 3.042721e-06]	IN
10	7.152557e-07	6.162049e-07	[3.874343e-07, 8.449755e-07]	IN
11	1.788139e-07	1.818219e-07	[4.135830e-08, 3.222856e-07]	IN
12	4.470348e-08	4.79766e-08	[-2.066563e-09, 9.801976e-08]	IN
13	1.117587e-08	6.50212e-08	[-4.986180e-08, 1.799042e-07]	IN
14	2.793968e-09	1.308815e-08	[-1.256463e-08, 3.874093e-08]	IN
15	6.984919E-10	9.416356E-09	[-9.039702e-09, 2.787241e-08]	IN

Table 5.2: Confidence intervals for the mean probability of station 1 being in a particular state

State	Analytical Res.	Sim. Estimate	Confidence Interval	Check
0	5e-01	5.000796e-01	[4.998506e-01, 5.003086e-01]	IN
1	2.5e-01	2.500617e-01	[2.499668e-01, 2.501566e-01]	IN
2	1.25e-01	1.249862e-01	[1.248986e-01, 1.250739e-01]	IN
3	6.25e-02	6.246857e-02	[6.238527e-02, 6.255188e-02]	IN
4	3.125e-02	3.121386e-02	[3.115366e-02, 3.127407e-02]	IN
5	1.5625e-02	1.561345e-02	[1.557336e-02, 1.565354e-02]	IN
6	7.8125e-03	7.791495e-03	[7.752171e-03, 7.830819e-03]	IN
7	3.90625e-03	3.890731e-03	[3.862091e-03, 3.919371e-03]	IN
8	1.953125e-03	1.944338e-03	[1.924479e-03, 1.964197e-03]	IN
9	9.765625e-04	9.784595e-04	[9.666310e-04, 9.902881e-04]	IN
10	4.882812e-04	4.870465e-04	[4.800364e-04, 4.940566e-04]	IN
11	2.441406e-04	2.446756e-04	[2.393640e-04, 2.499872e-04]	IN
12	1.220703e-04	1.198068e-04	[1.163265e-04, 1.232872e-04]	IN
13	6.103516e-05	5.848656e-05	[5.582695e-05, 6.114616e-05]	IN
14	3.051758e-05	2.988362e-05	[2.835481e-05, 3.141243e-05]	IN
15	1.525879e-05	1.500602e-05	[1.336080e-05, 1.665125e-05]	IN
16	7.629395e-06	8.331384e-06	[7.300057e-06, 9.362711e-06]	IN
17	3.814697e-06	4.318951e-06	[3.570253e-06, 5.067649e-06]	IN
18	1.907349e-06	2.111977e-06	[1.544334e-06, 2.679621e-06]	IN
19	9.536743e-07	1.125475e-06	[6.725728e-07, 1.578378e-06]	IN
20	4.768372e-07	5.271561e-07	[3.021374e-07, 7.521747e-07]	IN
21	2.384186e-07	1.262716e-07	[6.618201e-09, 2.459251e-07]	IN
22	1.192093e-07	5.300295e-08	[-1.230042e-09, 1.072360e-07]	NOT IN
23	5.960464e-08	5.975658e-09	[-3.568474e-09, 1.551979e-08]	NOT IN

Table 5.3: Confidence intervals for the mean probability of station 2 being in a particular state.

State	Analytical Res.	Sim. Estimate	Confidence Interval	Check
0	2.5e-01	2.501416e-01	[2.499188e-01, 2.503645e-01]	IN
1	1.875e-01	1.876205e-01	[1.874508e-01, 1.877901e-01]	IN
2	1.40625e-01	1.40628e-01	[1.405299e-01, 1.407262e-01]	IN
3	1.054688e-01	1.054363e-01	[1.053705e-01, 1.055022e-01]	IN
4	7.910156e-02	7.905655e-02	[7.898594e-02, 7.912716e-02]	IN
5	5.932617e-02	5.929747e-02	[5.920879e-02, 5.938615e-02]	IN
6	4.449463e-02	4.446399e-02	[4.437527e-02, 4.455271e-02]	IN
7	3.337097e-02	3.334719e-02	[3.326754e-02, 3.342684e-02]	IN
8	2.502823e-02	2.498615e-02	[2.491311e-02, 2.505920e-02]	IN
9	1.877117e-02	1.874255e-02	[1.867688e-02, 1.880822e-02]	IN
10	1.407838e-02	1.406218e-02	[1.401332e-02, 1.411104e-02]	IN
11	1.055878e-02	1.053843e-02	[1.050180e-02, 1.057505e-02]	IN
12	7.919088e-03	7.888453e-03	[7.852407e-03, 7.924500e-03]	IN
13	5.939316e-03	5.921576e-03	[5.883531e-03, 5.959620e-03]	IN
14	4.454487e-03	4.442649e-03	[4.416016e-03, 4.469282e-03]	IN
15	3.340865e-03	3.335884e-03	[3.308099e-03, 3.363670e-03]	IN
16	2.505649e-03	2.513033e-03	[2.490875e-03, 2.535190e-03]	IN
17	1.879237e-03	1.883832e-03	[1.863443e-03, 1.904220e-03]	IN
18	1.409428e-03	1.419882e-03	[1.404671e-03, 1.435092e-03]	IN
19	1.057071e-03	1.063748e-03	[1.048375e-03, 1.079121e-03]	IN
20	7.92803e-04	8.013267e-04	[7.870798e-04, 8.155736e-04]	IN
21	5.946022e-04	6.076284e-04	[5.945254e-04, 6.207314e-04]	IN
22	4.459517e-04	4.525175e-04	[4.418390e-04, 4.631960e-04]	IN
23	3.344638e-04	3.367014e-04	[3.268752e-04, 3.465276e-04]	IN
24	2.508478e-04	2.527254e-04	[2.427817e-04, 2.626692e-04]	IN
25	1.881359e-04	1.927272e-04	[1.850548e-04, 2.003996e-04]	IN
26	1.411019e-04	1.441917e-04	[1.374834e-04, 1.509001e-04]	IN
27	1.058264e-04	1.08553e-04	[1.033015e-04, 1.138045e-04]	IN
28	7.936982e-05	8.122699e-05	[7.701107e-05, 8.544291e-05]	IN
29	5.952736e-05	5.96537e-05	[5.599948e-05, 6.330793e-05]	IN
30	4.464552e-05	4.442939e-05	[4.133876e-05, 4.752002e-05]	IN
31	3.348414e-05	3.304209e-05	[3.025337e-05, 3.583081e-05]	IN
32	2.511311e-05	2.438208e-05	[2.182315e-05, 2.694100e-05]	IN
33	1.883483e-05	1.75217e-05	[1.515439e-05, 1.988900e-05]	IN

34	1.412612e-05	1.344964e-05	[1.157648e-05, 1.532281e-05]	IN
35	1.059459e-05	1.017461e-05	[8.630080e-06, 1.171914e-05]	IN
36	7.945944e-06	7.517311e-06	[6.139295e-06, 8.895328e-06]	IN
37	5.959458e-06	5.923425e-06	[4.623264e-06, 7.223586e-06]	IN
38	4.469593e-06	4.397712e-06	[3.174556e-06, 5.620867e-06]	IN
39	3.352195e-06	3.125398e-06	[2.152052e-06, 4.098744e-06]	IN
40	2.514146e-06	2.267176e-06	[1.344097e-06, 3.190256e-06]	IN
41	1.88561e-06	1.581048e-06	[9.216680e-07, 2.240428e-06]	IN
42	1.414207e-06	1.15398e-06	[6.158855e-07, 1.692074e-06]	IN
43	1.060655e-06	8.967196e-07	[5.039647e-07, 1.289475e-06]	IN
44	7.954916e-07	5.634597e-07	[2.558737e-07, 8.710457e-07]	IN
45	5.966187e-07	5.276168e-07	[1.950425e-07, 8.601911e-07]	IN
46	4.47464e-07	4.161931e-07	[1.233407e-07, 7.090454e-07]	IN
47	3.35598e-07	3.968342e-07	[1.056640e-07, 6.880045e-07]	IN
48	2.516985e-07	2.885574e-07	[6.617765e-08, 5.109371e-07]	IN
49	1.887739e-07	1.729103e-07	[2.494236e-08, 3.208781e-07]	IN
50	1.415804e-07	2.252675e-07	[1.183776e-08, 4.386973e-07]	IN
51	1.061853e-07	1.788831e-07	[1.958468e-08, 3.381814e-07]	IN
52	7.963898e-08	5.769006e-08	[-2.868590e-08, 1.440660e-07]	IN
53	5.972924e-08	2.681123e-08	[-2.573878e-08, 7.936125e-08]	IN
54	4.479693e-08	8.054352e-09	[-7.732178e-09, 2.384088e-08]	NOT IN

Table 5.4: Confidence intervals for the mean probability of station 3 being in a particular state

For all the computed confidence intervals we computationally checked that our theoretical results do belong to the intervals and this result is shown in the column we called "check". The fact that three theoretical values do not belong to the computed confidence intervals is caused by the smaller and smaller probability of being in a certain state. Notice also that for the construction of confidence intervals we executed 30 runs, processing 8 million events for each run, this was done in order to try to satisfy the normal approximation assumption. By this set of simulations, we have shown that the analytical model accurately approximates the performance obtained with the simulation, consequently we consider the simulation program validated.

5.4 Sensitivity Analysis

Sensitivity analysis is used to determine how output variables (i.e. the results of a simulation) are affected by changes applied to the input random variables. In practice, we violate some model assumptions to see how much these assumptions influence the outcomes. We will state that the model is robust with respect to the assumptions we will violate if the obtained results will be not so distant from the original output. Notice that, in this case we do not expect theoretical results to be inside the confidence intervals built from the simulation results but we hope to observe a similar behaviour in the distributions and similar values as far as mean indexes are concerned. On the contrary, if the results are completely different we will say that the model is not robust with respect to the violation of the assumptions.
We are particularly interested in testing the sensitivity of the model under study with respect to a departure from the classical exponential assumption of the service times. In order to do this, we repeated the experiments described in the previous section for two other distinctive distributions describing service times; we used the lognormal and the uniform distributions. For both of them, we kept the same means we used for the exponential one. Then, we applied a variation coefficient of approximately 0.58 to the uniform distribution, meaning we have a smaller variance with respect to the exponential one[4]. In order to do so, we set 0 and $2\overline{X}$ as the parameters of the uniform distribution, where $\bar{X}$ is the mean of the exponential distribution. As far as the lognormal distribution is concerned, we applied a variation coefficient of approximately 1.22 using a standard deviation parameter defined by the following formula: $\sqrt{\frac{1}{2\overline{X}} + \frac{1}{\overline{X}}}$, meaning we added half of the variance to the variance of the exponential distribution to obtain the variance of the lognormal one, then we retrieved the standard deviation squaring the obtained values. We performed the experiments both in case of low load and of high load (in this case we have $\mu_0 = 1.9$, $\mu_1 = 2.0$, $\mu_2 = 2.1$, $\mu_3 = 2.2$). Below we show the obtained results.

First we present three plots, corresponding to the three warehouses, showing the proportion of time spent in each state for each distribution and according to the theoretical result. Then, we present a table containing the mean number of jobs in a warehouse, for each warehouse and for each distribution. These plots and the table are shown first in the low load case and then in the high load case.

[4]The variation coefficient is given by the standard deviation divided by the mean; notice that the exponential distribution has variation coefficient equal to 1. A bigger variation coefficient means we have a greater variance and conversely a variation coefficient smaller than 1 means we have a lower variance

Low load.

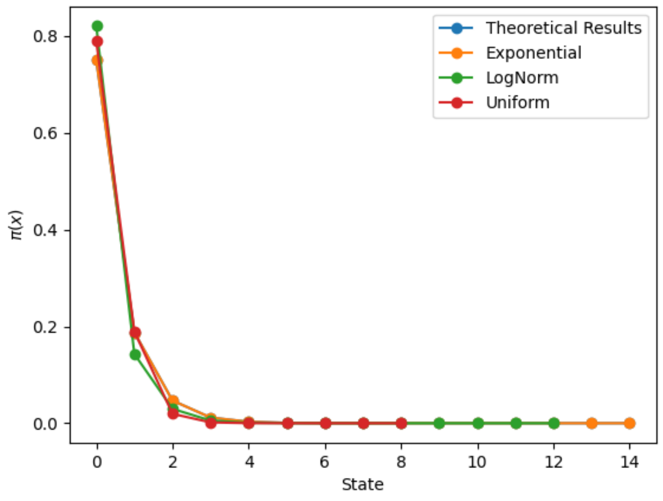


Figure 5.9: Warehouse 1.

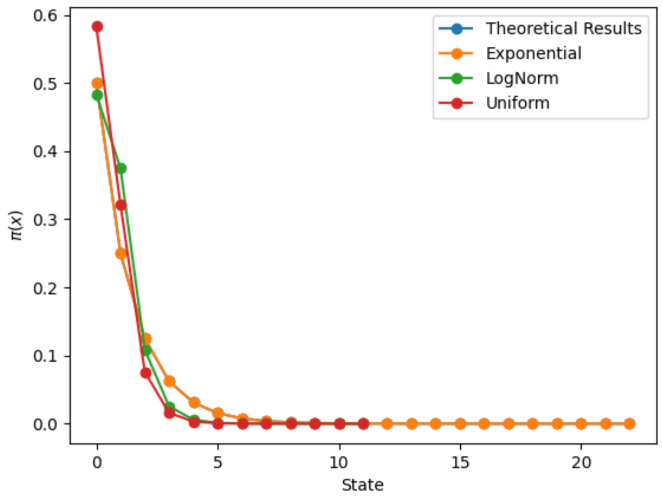


Figure 5.10: Warehouse 2.

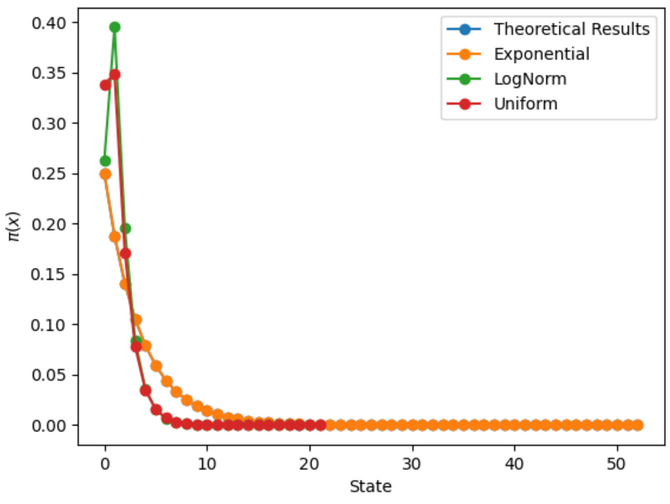


Figure 5.11: Warehouse 3.

Station	Analytical Res.	Exponential	Uniform	Lognormal
1	0.3333	0.3331	0.2340	0.2239
2	1.0	0.9993	0.5382	0.7014
3	3.0	2.9988	1.2319	1.3391

Table 5.5: Mean number of jobs in each station according to the analytical results and according to the simulation estimates provided respectively by the exponential, uniform and lognormal distribution.

High load

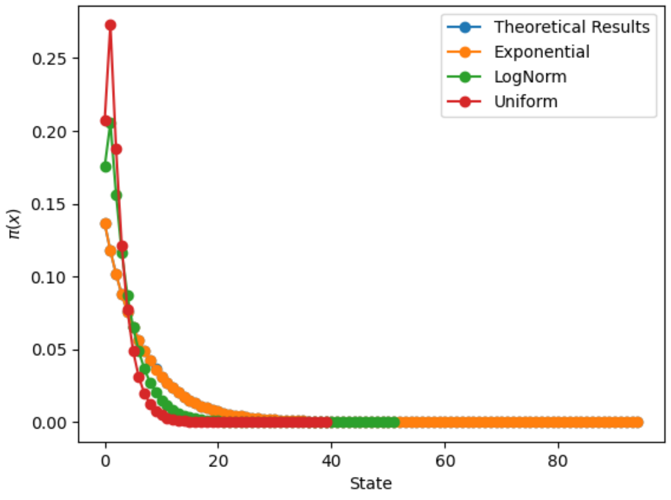


Figure 5.12: Warehouse 1.

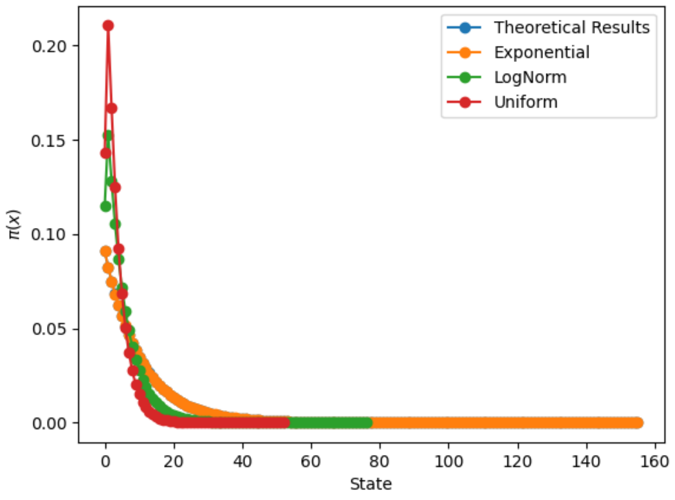


Figure 5.13: Warehouse 2.

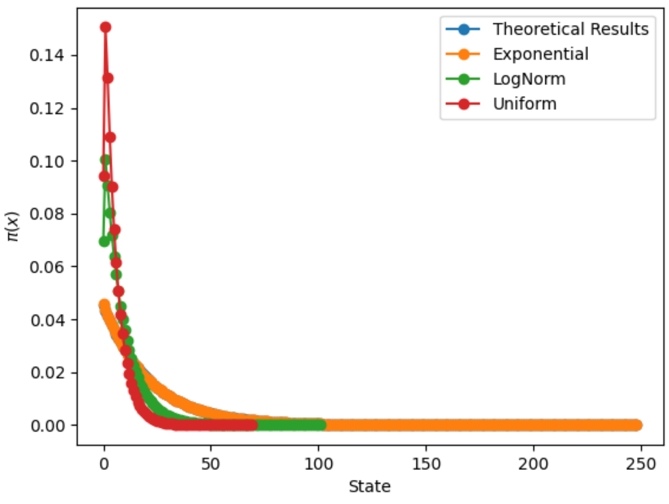


Figure 5.14: Warehouse 3.

Station	Analytical Res.	Exponential	Uniform	Lognormal
1	6.3333	6.3388	2.2212	3.2738
2	10.0	10.0182	3.3338	5.0855
3	21.0	20.9659	5.2288	8.5578

Table 5.6: Mean number of jobs in each station according to the analytical results and according to the simulation estimates provided respectively by the exponential, uniform and lognormal distribution.

From the results produced above we can state that in low load, violating the exponential assumption of service times, the system does not behave exactly as predicted by the analytical solution. Nevertheless, the mean number of jobs for each warehouse does not exhibit a huge effect caused by the changing of the distributions. As we can see through a quick inspection of the results in the high load case, the values obtained using the lognormal or the uniform distribution get more and more distant from our original outcomes.

Conclusions and Future Works

In this thesis, we have presented the theoretical background needed to understand the different steps of our work. We showed the model at the basis of our study, then we approached to it from a theoretical point of view in order to simplify the proof of its product-form solution. This, in fact, was initially proved by the authors of [4] employing the properties of dynamically reversed processes, which can result complicated and computationally heavy. The main problem with this kind of proof is that we have to guess both the reversed process and the renaming function of the states, this can certainly be very difficult to achieve, as a matter of fact the literature has few example of models whose product-form solution was derived in this way.
Using the Reversed Compound Agent Theorem we were able to prove the same result; this theorem enabled us, through the use of the PEPA stochastic process algebra, to model and interpret in a simpler way the interactions and synchronization actions among different components of a system model. In fact, RCAT offers an alternative approach to derive the equilibrium state probabilities without the need of solving balance equations. Moreover, it allows us to mechanise the methodology used for generating reversed process using only the instantaneous transition rates and then a product-form solution follows from specifications. Thus, the related effort for the computation of product form solutions is drastically reduced.
We approached the model also from a more practical perspective; as a matter of fact we built and validated a simulation program simulating our model, this was done to gain an in-depth knowledge of the model, to carry out a performance assessment and also a comparison with the obtained analytical results; moreover, we also used it to test if the obtained results are invariant also in case of violation of the model assumptions.

In conclusion we underline that until this moment the model is assumed to have only tandem configuration, meaning we are considering a system composed of a certain number of queues where a departure from one of them means an arrival to the following queue. This is valid unless the departure is taking place in the last queue of this tandem, in this case the departure is to be intended as a departure from the system. An implied property of this configuration is that it does not admit feedback. However, in other real-world applications we could think of a scenario in which feedback is needed, this is why we would like to expand this model to admit them and in general to extend it to other topologies. To do so we intend to use the presented tools in order to give a theoretical proof on a possible reliable product-form and also an empirical validation of these results.

Bibliography

[1] P. G. Harrison, "Turning back time in markovian process algebra," *Theoretical Computer Science*, vol. 290, pp. 1947–1986, 2003.

[2] P. G. Harrison and A. Marin, "Deriving the rate equations characterising product-form models and application to propagating synchronisations," *6th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 107–116, 2012.

[3] P. Harrison and A. Marin, "Product-forms in multi-way synchronizations," *The Computer Journal*, vol. 57, pp. 1693–1710, 2013.

[4] D. Gates and M. Westcott, "Replacement of train wheels: An application of dynamic reversal of a markov process," *Journal of Applied Probability*, vol. 31, pp. 1–8, 1994.

[5] L. Kleinrock, *Queueing Systems, Volume I: Theory.* Wiley-Interscience, 1975.

[6] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications.* Wiley-Interscience, 1976.

[7] J. Hillston, *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[8] F. Kelly, *Reversibility and Stochastic Networks.* Probability and Statistics Series, J. Wiley, 1979.

[9] M. Baron, *Probability and Statistics for Computer Scientists.* 5 ed., 2018.

[10] A. Marin and S. Rossi, "On discrete time reversibility modulo state renaming and its applications," *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014*, p. 225–232, 2014.

[11] H. H. Liu, *Software Performance and Scalability: A Quantitative Approach.* Wiley, 2008.

[12] P. J. Burke, “The output of a queuing system,” *Operations Research*, vol. 4, pp. 699–704, 1956.

[13] J. R. Jackson, “Networks of waiting lines,” *Operations Research*, vol. 5, pp. 518–521, 1957.

[14] J. R. Jackson, “Jobshop-like queueing systems,” *Management Science*, vol. 10, pp. 131–142, 1963.

[15] J. Y. L. Boudec, *Performance Evaluation of Computer and Communication Systems*. EPFL Press, 2011.

[16] A. M. Law, *Simulation Modeling & Analysis*. McGraw-Hill, 5 ed., 2015.

[17] P. L’Ecuyer, “Software for uniform random number generation: Distinguishing the good and the bad,” *Winter Simulation Conference Proceedings*, vol. 1, pp. 95 – 105, 2001.

[18] M. Matsumoto and T. Nishimura, “Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Trans. Model. Comput. Simul.*, vol. 8, p. 3–30, 1998.

[19] R. G. Sargent, D. M. Goldsman, and T. Yaacoub, “A tutorial on the operational validation of simulation models,” *Proceedings of the 2016 Winter Simulation Conference*, p. 163–177, 2016.

[20] A. Clark, S. Gilmore, J. Hillston, and M. Tribastone, “Stochastic process algebras,” *The Computer Journal*, vol. 55, pp. 132–179, 2007.

[21] E. Gelenbe, “Product-form queueing networks with negative and positive customers,” *Journal of Applied Probability*, vol. 28, pp. 656–663, 1991.

[22] E. Gelenbe, “G-networks with triggered customer movement,” *Journal of Applied Probability*, vol. 30, p. 742–748, 1993.

[23] S. Balsamo, P. G. Harrison, and A. Marin, “A unifying approach to product-forms in networks with finite capacity constraints,” *SIGMETRICS Perform. Eval. Rev.*, vol. 38, p. 25–36, 2010.

[24] P. Harrison, “Compositional reversed markov processes, with applications to g-networks,” *Performance Evaluation*, vol. 57, pp. 379–408, 2004.

[25] M. Westcott, “Modelling and Analysis of Wheel Replacement and Restoration,” *Proceedings of the 1990 Mathematics-in-Industry Study Group*, pp. 30–44, 1990.

[26] S. Team, “Simpy v. 4.0.1 documentation.” `https://simpy.readthedocs.io/en/4.0.1/`.