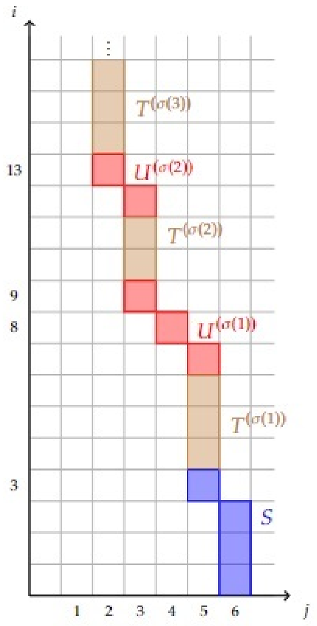
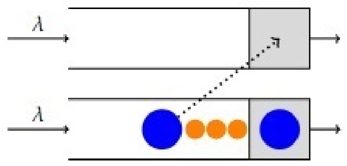
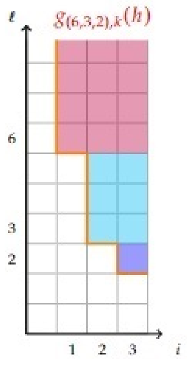
# Analysis of load balancing and scheduling policies in large-scale systems

Grzegorz Kielanski



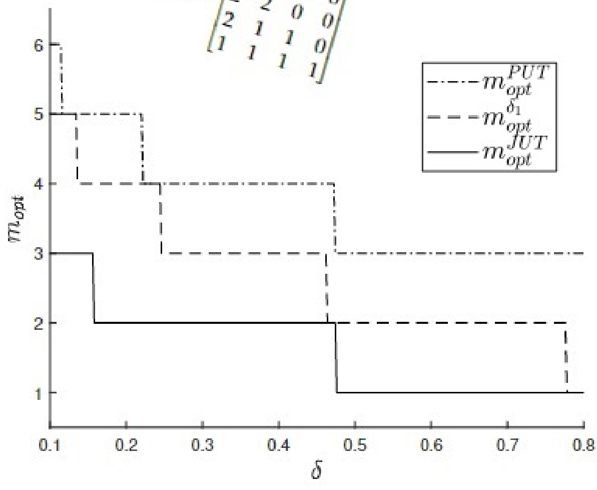
$$\xi_{1,k} = \lambda p_k + \frac{r\zeta_*}{r\zeta_* + \mu_1} \sum_{d=k+1}^{m} \phi_{d,d-k}\xi_{1,d}$$





$$E[R(m)] = 1 + \frac{\bar{\lambda} E[G^2]}{2(1-\bar{\lambda})} + \frac{\delta}{\bar{\lambda}} \frac{m(m-1)}{2(1-\bar{\lambda})}$$

$$P_4 = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$



$$T_\phi = \begin{bmatrix} 0_{mn_c+n_p,mn_c} & 0_{mn_c+n_p,(m+1)n_p} \\ \Phi \otimes (1_{n_p}\alpha^c) & 0_{mn_p,(m+1)n_p} \end{bmatrix}$$

Supervisor **prof. dr. Benny Van Houdt**

Thesis submitted in fulfilment of the requirements for the degree of doctor in science: Computer Science
**Faculty of Science | Antwerpen, 2023**

University of Antwerp

University of Antwerp

Faculty of Science

# Analysis of load balancing and scheduling policies in large-scale systems

Thesis submitted in fulfilment of the requirements for the degree of
doctor in science: Computer Science
at the University of Antwerp

**Grzegorz Kielanski**

Antwerpen, 2023

Supervisor
prof. dr. Benny Van Houdt

**Contact**
Grzegorz Kielanski

University of Antwerp
Faculty of Science
IDLab
Middelheimlaan 1, 2020 Antwerpen, België
M: Grzegorz.Kielanski@uantwerpen.be
T: 032653519

ISBN 000-00-00000-00-0

0 000000 000000

Dutch title:

# Analyse van werklastverdeling en planning in grootschalige systemen

*To my grandmother Stefania Rutka*

# Acknowledgments

Writing a PhD thesis is something that is never done overnight (or, at least, I hope so). This thesis is, in fact, based on three and a half years of my research. I really enjoyed doing research, although it had its challenging moments. This is why, I would first like to express my gratitude to the promoter of my PhD, prof. dr. Benny Van Houdt, for his leadership and for the hints he gave me when I really got stuck. Benny, you are an amazing boss and a great lecturer. I, once again, thank you for this opportunity!

I would also like to thank my parents, Barbara and Wiesław, for their continual support. You are always there for me and I know I'm extremely lucky to have you as my parents. Z całego serca Wam dziękuję!

Many thankful hugs go to my girlfriend, Emma, for her love. Your smile always lights up my day!

I thank Tim, Heda, Hadisha, Amina, Jessica, Thomas, Hazal, Berk, Tycho, Arben, Guner, Jolien, Vladomeare, Dominik, . . . for being great friends. I lift a glass to you!

I bow my head to my (ex-)colleagues: Tim, Rakshit and Lucas. I always enjoyed working and/or sharing work space with you!

Thank you to the students that attended my lectures. I hope you enjoyed my teaching as much as I enjoyed teaching you!

Last but not least, I would like to thank all the other people that have ever supported me.

*I wish good health to you all,*
*Grzegorz*

# Summary

Queueing theory plays a crucial role in modelling systems with congestion. It has been long applied in analyzing and improving the performance of communication systems. As modern communication systems often are composed of multiple heterogeneous resources, the analysis of such large-scale systems using traditional queueing theory can be prohibitive. When analyzing such systems exactly, one usually comes across the problem of the, so called, state space explosion.

Large-scale systems are therefore often studied through mean field analysis: if a system consists of a large number of queues, it can be approximated by a system with infinitely many queues. The analysis of the model of the latter system, the mean field model, is generally more straightforward, as it circumvents the problem of the state space explosion.

The goal of this thesis is to analyze and gain insight in the performance of existing and novel large-scale load balancing policies through the use of mean field modelling. Each chapter of this thesis contains the mean field analysis of a family of systems with these policies. In the analysis, techniques from dynamical systems, stochastic modelling, probability theory, numerical analysis and simulations are used.

The chapters are grouped into three parts. The first of these parts deals with monotone systems. These systems have an apparent ordering of states that is maintained over time. The next part deals with the analysis of systems with job stealing and multithreaded computing. In these systems parts of a job can be transferred between queues and can be thus worked on by different queues concurrently. In the last part several hyperscalable policies with a single dispatcher are studied. These are policies where a central dispatcher distributes the jobs to the queues, based on their estimated queue lengths. The policies are called hyperscalable if the message overhead between the queues and the dispatcher is less than one message per job on average. For the systems in the last two parts, simulations are performed for finite number of queues to measure the accuracy of the mean field approximation.

# Contents

# Part I

# Introductory Chapters

# Chapter 1
# Introduction

Imagine you find yourself in the following situation. You go to a supermarket. You search through the aisles for the products you want and when you are done you proceed to the cash registers to pay. Arriving at the cash registers, you notice that three of them are open. You see that there are customers queueing for each of these cash registers. Which of these queues do you choose to wait in? Without overthinking it, you simply count the number of customers in each queue and decide to join the one with the least customers. Assuming you want to wait the least amount of time, is this a good strategy?

Queueing theory is used to study queueing systems and tries to answer questions such as the one above. A queueing system is, roughly stated, any system where congestion occurs. Queueing systems consist of one or multiple queues, each queue further consisting of one or multiple service stations (servers) and a buffer. In this thesis, every queue has a single server. Every server provides service to the customers (jobs) that arrive to the server's queue. When studying a queueing system, we have to specify several things. First, the arrival process of the customers and the way they are distributed among the queues. The latter is called a "load balancing policy". One also has to specify in what manner do the customers queue (f.e. in what order the customers are serviced). This is called the "scheduling policy", also known as the "service discipline".

In the example above, the queueing system consists of three servers (cashiers), each cashier handling customers in the order of their arrival to the respective queue (the scheduling policy is "first come, first serve"). The load balancing policy is "join the shortest queue".

Continuing with the example, after you joined the shortest queue, it still turns out that the customers in the other queues get serviced faster. How is that possible? Maybe, your cashier works slower that the others... In terms of queueing theory, the servers possibly are not homogeneous (they are heterogeneous). Whether the servers are homogeneous or not is thus an important property of a queueing system.

Another possible explanation is that it just so happens that in your chosen queue, the customers have way more products that need to be scanned than in the other queues. In queueing theory this is modelled by assuming that the job sizes are random, this randomness can be described by specifying the job size distribution.

When modelling a queueing system we also have to make a bunch of other assumptions, for example:

- Is it possible that the service stations (in this case, the cash registers or the cashiers) break down?
- Do bored customers decide at some point to simply leave the system; here, the store (hopefully, without the products)?
- ...

Once all these assumptions are specified, the stability and the performance of the system can be studied. Stability means the following: if we let the system run infinitely long, does the number of customers in a queue stay finite or does it keep growing (on average). If the former is true for every queue of the system, the system is stable. The performance of a queuing system is usually measured through the "mean response time". This is the average time that a customer has to stay in the system (i.e. the sum of the time the customer has to wait in the queue and the time it takes to service the customer).

This thesis, that is the fruit of almost four years of my labour, deals with the analysis of the long term behaviour of several load balancing and scheduling policies in large scale systems. These are systems consisting of a large number of queues. The analysis deals both with well-known as well as novel policies. Each chapter of this thesis contains the analysis of a family of systems with these policies. In the analysis, techniques from dynamical systems, stochastic modelling, probability theory, numerical analysis and simulations are used.

In the remainder of the introduction, I shall give a brief explanation on the numbering used in the thesis and further on how the thesis is structured. When explaining the latter, I shall also suggest a reading order based on the reader's level of proficiency in queueing theory.

## 1.1 Structure of the thesis

### 1.1.1 Numbering used in the thesis

The thesis is made up of four parts followed by appendices. The four parts are numbered using roman numerals, while the different appendices use capital letters for labels. In the rest of the numbering explained further, Arabic numerals are used. Each part is split up into several numbered chapters (the numbering is not reset between parts). Every chapter consists of several sections, which in turn can further consist of subsections. These are numbered as *chapter.section* and *chapter.section.subsection* respectively. So, for example, you are now reading Part I, Chapter 1, Section 1.1 and Subsection 1.1.1. The numbering of sections and subsections is respectively reset between chapters and sections. For example, the first section of the next chapter is Section 2.1.

Sections can also contain "environments", namely: theorems, propositions, lemmas, definitions, remarks, examples and conjectures. These are numbered as *chapter.section.environment*, with the numbering being reset between sections, for example:

**Example 1.1.1.** This is an example.

Sometimes, references are made to expressions or equations. They are numbered as (*chapter.expression*), for example:

$$1 + 1 = 2. \tag{1.1}$$

This numbering is reset between chapters.

### 1.1.2 Chapters of the thesis and suggested reading order

As noted, the thesis is made up out of four parts and three appendices. The four parts are as follows:

- Part I: Introductory Chapters. This is the part you are reading now. It contains several introductory chapters to the thesis, namely: Chapter 2 on all important distributions used in this thesis, followed by a chapter with four elementary single server queueing systems (Chapter 3). There we also touch on several matrix identities used from Part III onward. The last introductory chapter, Chapter 4, contains an explanation on two techniques used to approximate systems with a large number of servers.
- Part II: Monotone Systems. The part consists of two chapters, each dealing with mean field models of a family of monotone systems. These are systems where the states show a certain ordering that is kept over time: if a state of the system is dominated w.r.t. this ordering by another state at some timestamp, it will stay dominated by the other system at all future times.
- Part III: Multithreaded Computing. The part consists of three chapters, each based on a published paper. It deals with systems where parts of a job can be concurrently worked on by different servers.
- Part IV: Hyperscalable Policies. In this part we study three hyperscalable policies. These are policies with communication overhead below 1. This means the following. We assume that there is a central dispatcher that assigns jobs to the queues based on information on the queue lengths that the dispatcher possesses. This information is updated through the messages sent between the dispatcher and the queues. We call a load balancing policy hyperscalable if, on average, less than one message per job is sent.

Note, that Parts II-IV contain original research for this thesis. The parts above are followed by the appendices containing a list of abbreviations (Appendix A) and a list of symbols (Appendix B). The final appendix (Appendix C) is a Dutch summary of the thesis.

I tried to make the thesis accessible to anybody with a basic understanding of probability theory, calculus, dynamical systems and matrices. To this end, I made the thesis as self-contained as possible. Below, I suggest a reading order of the chapters of this thesis, based on your level of experience with queueing theory.

If you, dear reader, have no prior experience with queueing theory, I suggest you first thoroughly read the introductory chapters (Chapters 2-4) in the presented order. If you have a basic understanding of queueing theory, I still suggest you read the introductory chapters. If you however are well versed in queueing theory, you may start reading Parts

II-IV right away. These parts can be read in any order and so can the chapters contained in them. I do however believe that the chapters of Parts III and IV can be best read in the order they are presented here.

As is customary in mathematical literature, I shall use the word “we” instead of “I” starting form the next chapter of the thesis. I hope you enjoy reading this book!

Chapter 2

# Frequently used distributions

When modelling any type of system that has randomness (aka a stochastic system), we usually have to specify what we mean by “random”. In mathematical terms, we have to describe the “randomness” by using random variable(s) and specify the distribution of said variable(s). In this chapter we therefore introduce all distributions used in this thesis, together with some important properties thereof.

The chapter is structured as follows. In Section 2.1, we define the exponential distribution, using the definition we provide a characterisation of the Poisson process. We also list several properties of this distribution and of the Poisson process. In Section 2.2, we define a generalization of the exponential distribution, namely (continuous) phase type distributions and list properties thereof used throughout the thesis. Several important examples of PH distributions, together with their properties are shown in Section 2.3. Section 2.4 deals with distributions that are not of the phase type but are used throughout the paper, namely discrete distributions and bounded Pareto distributions. When introducing the former of the two, we also define generating functions. Finally Section 2.5 contains the definition and several properties of Laplace-Stieltjes transforms.

## 2.1 Exponential distribution and the Poisson process

In this section, which is loosely based on [78], we introduce the exponential distribution and the Poisson process. There are multiple, different, equivalent ways for defining the latter of the two. However, for simplicity, we will introduce the Poisson process using the exponential distribution.
We also provide several useful properties of the two. These properties are part of the reason why exponential distributions and Poisson processes are widely used when modelling real life situations.

**Definition 2.1.1** (Exponential distribution)**.** We say that a positive valued random variable $X$ is exponentially distributed with parameter $\lambda$, $X \sim \exp(\lambda)$ for short, if it has a probability density function (pdf) given by $f_X(t) = \lambda e^{-\lambda t}$. In this case, its cumulative distribution function (cdf) is given by $F_X(t) = 1 - e^{-\lambda t}$ and its mean by $E[X] = 1/\lambda$.

In general, if a cdf $F$ is differentiable, then the pdf $f$ can be obtained as $f(t) = F'(t)$. For a simple explanation on cdfs see Subsection 2.4.1.

Suppose we wish to model the number of incoming calls to a call centre or the number of cars passing by a certain point on the road. If we assume that the average traffic intensity is constant, then we can use the Poisson process to this end.

**Definition 2.1.2** (Poisson (counting) process)**.** A counting process $(P(t))_{t\in[0,+\infty[}$, where counted occurrences will be called arrivals henceforth, is called Poisson with parameter/rate $\lambda$ if the following three conditions hold:

1. $P(0) = 0$, i.e. when we start counting we have observed no arrivals.
2. $P(t)$ has independent increments, i.e. the number of arrivals in non-overlapping time intervals is independent.
3. The interarrival times, that is the times between consecutive arrivals, are exponentially distributed with parameter $\lambda$.

We defined the Poisson process with rate $\lambda$ using one of equivalent definitions, namely through exponential interarrival times. An immediate consequence is that it takes on average $1/\lambda$ time units between arrivals.
The Poisson process can also be defined as a counting process where the number of arrivals in a time interval of length $t$ has a Poisson($\lambda t$) distribution or by giving the probability that an arrival occurs between time $t$ and $t + \Delta t$ [30, Section 1.7.].

We now list some properties of the exponential distribution and the Poisson process. The first one is the memoryless property of the exponential distribution [30, Section 1.8.].

**Theorem 2.1.3** (Memorylessness)**.** *The exponential distribution with parameter $\lambda$ is memoryless, meaning for $X \sim \exp(\lambda)$ and for $t, s \geq 0$ we have*

$$P(X > t + s|X > s) = P(X > t).$$

As interarrival times of a Poisson process are exponentially distributed, memorylessness implies that the timestamps of future arrivals are independent of those of previous arrivals. As such the memoryless property makes the exponential distribution an attractive tool in modelling stochastic systems. Another such property is the following:

**Theorem 2.1.4** (Minimum of two independent exponential distributions)**.** *Let $X_1$ and $X_2$ be independent exponentially distributed random variables with parameters $\lambda_1$ and $\lambda_2$, respectively. Then, $\min(X_1, X_2)$ is exponentially distributed with parameter $\lambda_1 + \lambda_2$. Further,*

$$P(X_1 < X_2) = \frac{\lambda_1}{\lambda_1 + \lambda_2}.$$

**Corollary 2.1.5.** *Let $X_1, X_2, \dots, X_n$ be independent exponentially distributed random variables with parameters $\lambda_1, \lambda_2, \dots, \lambda_n$, respectively. Then, $\min(X_1, X_2, \dots, X_n)$ is exponentially distributed with parameter $\lambda_1 + \lambda_2 + \cdots + \lambda_n$. Further,*

$$P(X_1 = \min(X_1, \dots, X_n)) = \frac{\lambda_1}{\lambda_1 + \cdots + \lambda_n}.$$

*Proof.* Both claims follow immediately from Theorem 2.1.4. □

In the next section we will make use of this Corollary when explaining phase type distributions. We now provide two important properties of the Poisson process.

**Theorem 2.1.6** (Poisson superposition). *The superposition of two independent Poisson processes with parameters $\lambda_1$ and $\lambda_2$ is a Poisson process with parameter $\lambda_1 + \lambda_2$. In other words, if arrivals can occur due to two independent Poisson processes with parameters $\lambda_1$ and $\lambda_2$, they in fact occur due to a Poisson process with parameter $\lambda_1 + \lambda_2$.*

*Proof.* Let $I_1 \sim \exp(\lambda_1)$ and $I_2 \sim \exp(\lambda_2)$ denote the interarrival times of the first and second Poisson process respectively. As interarrival times of both processes are memoryless (cf. Theorem 2.1.3), the interarrival time of the superposed process is distributed as $\min(I_1, I_2)$. Due to Theorem 2.1.4, the interarrival times of the superposed process are exponentially distributed with parameter $\lambda_1 + \lambda_2$. □

**Theorem 2.1.7** (Poisson random split). *If a Poisson process with parameter $\lambda$ is randomly split into two subprocesses with probability $p$ and $1 - p$, then the resulting processes are independent Poisson processes with parameters $\lambda p$ and $\lambda(1 - p)$.*

We reflect for a moment on Theorems 2.1.6 and 2.1.7. In this thesis we study large scale systems, that is, systems consisting of large number of queues. Let us denote this number by $N$. Due to the last two results, it is equivalent to say that

- each queue has its own Poisson arrival process with rate $\lambda$; and that
- the system as a whole has a Poisson arrival process with rate $N\lambda$ with each incoming arrival being assigned at random to one of these $N$ queues.

## 2.2 Phase type distributions

Suppose you wish to model the production time of some product that requires several production steps/phases. In many such cases the production time can be described using a distribution of the phase type. A phase-type distribution can be of either continuous or discrete time. As we only use the former in this thesis, we only introduce continuous phase type distributions, or PH distributions for short. As PH distributions can be defined through Markov chains (MCs for short), we will also take the opportunity to provide a very brief explanation on continuous time Markov chains (CTMCs).

**Definition 2.2.1** (Continuous time distribution of the phase type). A random variable $X$ is (continuous time) PH-distributed with representation $(\alpha, S)$, if $X$ is the time until absorption in state 0 of the CTMC characterised by the initial probability vector $[\alpha_0, \alpha]$ and rate matrix

$$Q = \begin{bmatrix} 0 & 0'_n \\ s^* & S \end{bmatrix},$$

where

- $\alpha_0 \geq 0$ and $\alpha$ is a positive valued row vector of length $n$, such that $\alpha 1_n \leq 1$ and $\alpha_0 + \alpha 1_n = 1$ (where $1_n$ is a column vector of ones of height $n$);

- $S$ is an $n \times n$ matrix, with negative diagonal entries, positive off-diagonal entries and with negative row sums[1];
- $s^*$ is a positive column vector of height $n$ given by $s^* = -S1_n$; and
- where the states of the CTMC are labelled as $0, 1, \ldots, n$.

In this case, we write $X \sim PH(\alpha, S)$, we call $\alpha$ the initial probability distribution, $S$ the phase matrix, $n$ the number of phases of the distribution and $(\alpha, S)$ a representation of $X$.

Let us unpack this definition. Suppose $X \sim PH(\alpha, S)$ is the distribution of the time that is needed to finish some job.

**The vector $[\alpha_0, \alpha]$:** This vector holds the probabilities that the job starts in a given phase, f.e. the third entry of $\alpha$ is the probability that the job is in phase 3 at time 0. Note, that $\alpha_0$ is the probability that the job is finished immediately upon starting, that is at time 0. As we always want jobs to carry some work in this thesis, we assume throughout the other sections that $\alpha_0 = 0$. As $\alpha_0$ and every entry of $\alpha$ holds some probability, we get that $\alpha_0$ and $\alpha$ must be positive valued. Note further, that $\alpha_0 + \alpha 1_n = 1$ comes from the fact that the job starts in some phase. This equation also explains why we do not need to note the value of $\alpha_0$ in the representation $(\alpha, S)$.

**The states of the CTMC:** As $S$ is an $n \times n$ matrix, $Q$ is an $(n+1) \times (n+1)$ matrix. The rows and columns of $S$ are labelled from 1 to $n$ while those of $Q$ are labelled from 0 to $n$. In the latter manner, we also label the states of the CTMC. As the name suggests, the state of the CTMC is used to denote the current state of the job. Note, that states 1 to $n$ of the CTMC are called "phases" of the distribution.

**The matrix $Q$:** The moment the job enters phase $i$, $n$ exponential timers get started. We label these timers from 0 to $n$ but skip label $i$. The 0-th, first, second,..., $(i-1)$-st, $(i+1)$-st, ..., $n$-th timer counts down time drawn at random from exponential distribution with parameter $Q_{i,0}, Q_{i,1}, \ldots, Q_{i,i-1}, Q_{i,i+1}, \ldots, Q_{i,n}$ respectively. Here $Q_{i,j}$ denotes the entry of $Q$ at row $i$ and column $j$ ($(i,j)$-th entry of $Q$ for short). Suppose that the $j$-th timer is the first to run out of time. At that moment the job changes phase from $i$ to $j$, or equivalently, the CTMC transitions from state $i$ to $j$. If $j = 0$, the job is finished. Notice how the first row of $Q$ only contains zeros, therefore, once the CTMC enters state 0 it cannot leave it anymore. This is why the 0-th state is called "absorbing". Note further, that if $Q_{i,j} = 0$, then the corresponding timer never runs out of time.

We can also characterise the probabilities that the CTMC in state $i$ will enter state $j$ next. By using Corollary 2.1.5, we know that the probability that $j$-th timer is the first one to run out is given by

$$\frac{Q_{i,j}}{Q_{i,0} + \cdots + Q_{i,i-1} + Q_{i,i+1} + \cdots + Q_{i,n}}$$

By using $s^* = -S1_n$, the above expression can be simplified to $-Q_{i,j}/Q_{i,i}$. It also follows that the mean time until some timer runs out is $-1/Q_{i,i}$. In fact, $-Q_{i,i}$ can be thought of as the rate at which the job leaves phase $i$ or, equivalently, as the rate at which the CTMC

[1]A matrix that has these properties is called a subgenerator matrix.

leaves state $i$. Note, that $i$-th entry of the vector $s^*$ thus contains the rate at which the CTMC goes from state $i$ to state 0, i.e. the rate at which the job gets finished given that it is in phase $i$.

Note further, that the transition probabilities to the next phase/state only depend on the current phase/state and not on previous phases nor on the number of previous transitions. This is in fact what is called the "Markov property".

Note finally, that for a PH distribution there exist infinitely many representations. As an example note, that a hyperexponential distribution (see Subsection 2.3.2), with all rates $\mu_i$ the same, is simply an exponential distribution.

We now wish to proceed with properties of PH distributions. To this end we need to be able to raise the Euler's number $e$ to an exponent that is a matrix. The next definition explains how to perform this operation.

**Definition 2.2.2.** Let $A$ be a square matrix. The exponential of $A$, denoted by $e^A$, is defined as

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \frac{A^4}{4!} + \dots,$$

where $k!$ denotes the factorial of $k$ given by $1 \cdot 2 \cdot 3 \cdot \ldots \cdot k$.

**Theorem 2.2.3.** *Let $X \sim PH(\alpha, S)$. The cumulative distribution function (cdf) of $X$ is given by*

$$F(t) = 1 - \alpha e^{-St}\mathbf{1}, \text{ for } t \geq 0,$$

*where $\mathbf{1}$ is a column vector of ones of the appropriate height, i.e. $\mathbf{1}$ has the same height as the number of phases of $PH(\alpha, S)$. Further, the probability density function (pdf) of $X$ is given by*

$$f(t) = \alpha e^{-St}s^*, \text{ for } t > 0$$

*and $f(0) = \alpha_0$.*

*Proof.* See f.e. [48, Theorem 2.4.1] or Lemma 2.2.2. and the subsequent remark in [60]. □

Note, $F(t)$ is the probability that the absorption in state 0 occurs before time $t$. One might ask: can we be certain that the absorption in state 0 will occur? This leads to the following definition.

**Definition 2.2.4.** We call a PH distribution nondefective if the absorption into state 0 occurs in finite time with probability 1, i.e. if $F(\infty) = 1$.

Another way of saying that a PH distribution is nondefective is calling the states that the CTMC can visit "transient". Simply put, a transient state is a state that gets visited only a finite number of times by the CTMC, which must be the case if the PH distribution is nondefective.
In case the phase matrix of a representation is non-singular, we get that the PH distribution is nondefective:

**Theorem 2.2.5.** *Consider a distribution with representation $PH(\alpha, S)$. Absorption into state 0 occurs with probability 1 from any phase if and only if the matrix $S$ is non-singular.*

*Proof.* See the proof of [48, Theorem 2.4.3] or [60, Lemma 2.2.1.]. □

Conversely, if a PH distribution is nondefective, then it has representations with non-singular phase matrices:

**Theorem 2.2.6.** *Let $X$ be a PH distributed random variable. Suppose that this PH distribution is nondefective. Then, there exist a vector $\alpha$ and an invertible matrix $S$ such that $X \sim PH(\alpha, S)$.*

*Proof.* See [48, Theorem 2.4.4]. □

Throughout this thesis, we not only assume that all used PH-distributions are non-defective, but also assume that they are described using a representation with non-singular/invertible phase matrix.

**Theorem 2.2.7.** *Let $X \sim PH(\alpha, S)$ and suppose $S$ is non-singular. Then $(-S^{-1})_{i,j}$ is the expected total time until absorption spent in phase $j$ given the distribution started in phase $i$. As a consequence, $(-S^{-1})_{i,j} \geq 0$. Further, the expected value of $X$, that is, the mean time until absorption in state 0 is given by*

$$E[X] = \alpha(-S)^{-1}\mathbf{1}.$$

*Proof.* The proof of the first statement can be found in [48, Theorem 2.4.3]. The second and the third statement follow immediately from the first. □

In this thesis we sometimes have to work with PH distributions that have mean 1, i.e. if $X \sim PH(\alpha, S)$, then $E[X] = 1$. In this case, the next proposition can be of use.

**Proposition 2.2.8.** *Let $X \sim PH(\alpha, S)$. Suppose that $S$ is non-singular and $E[X] > 0$. Then for $Y \sim PH(\alpha, S \cdot \alpha(-S)^{-1}\mathbf{1})$, we have $E[Y] = 1$.*

*Proof.* By Theorem 2.2.7, we have

$$E[Y] = \alpha\left[-S \cdot \alpha(-S)^{-1}\mathbf{1}\right]^{-1}\mathbf{1} = \alpha(-S)^{-1}\mathbf{1}/\alpha(-S)^{-1}\mathbf{1} = 1,$$

where in the last equality, we have used $E[X] = \alpha(-S)^{-1}\mathbf{1} > 0$. □

The following theorem states that the maximum and minimum of two PH distributions is once again a PH distribution.

**Theorem 2.2.9** (Maximum and minimum of two PH distributions)**.** *Let $X, Y$ be two PH variables with representations $(\alpha, P)$ and $(\beta, Q)$ respectively. Denote by $n_P$ and $n_Q$ respectively the number of phases of these representations. Denote further $p^* = -P1_{n_P}$ and $q^* = -Q1_{n_Q}$. Then:*

- $\max(X, Y)$ *is PH-distributed and has a representation $(\gamma, S)$ of $n_P n_Q + n_P + n_Q$ phases given by*

$$\gamma = [\alpha \otimes \beta, \beta_0 \alpha, \alpha_0 \beta],$$
$$S = \begin{bmatrix} P \otimes I_{n_Q} + I_{n_P} \otimes Q & I_{n_P} \otimes q^* & p^* \otimes I_{n_Q} \\ 0 & P & 0 \\ 0 & 0 & Q \end{bmatrix}.$$

- $\min(X, Y)$ *is PH-distributed and has a representation of* $n_P n_Q$ *phases given by* $(\gamma, S) = (\alpha \otimes \beta, P \otimes I_{n_Q} + I_{n_P} \otimes Q)$.

*Note,* $I_n$ *denotes the identity matrix of dimensions* $n \times n$*, while* $\otimes$ *denotes the Kronecker product, which is given in Definition 3.6.5.*

*Proof.* See [60, Theorem 2.2.9.]. □

The next theorem states that any positive valued distribution can be approximated arbitrarily close by a PH distribution.

**Theorem 2.2.10.** *The class of phase-type distributions is dense in the class of distributions on* $[0, +\infty[$.

*Proof.* A full proof for this result can be found, for example, in [6, Section 3.2.1]. □

In Definition 2.2.1, $\alpha, S$ and $s^*$ have some requirements. In the next definition, we define matrix exponential distributions, where these requirements get relaxed significantly. It then follows that matrix exponential distributions are a generalization of PH distributions.

**Definition 2.2.11** (Matrix exponential distribution)**.** A distribution is called matrix exponential if there exist

- a row vector $\alpha$ of length $n$,
- an $n \times n$ matrix $S$
- and column vector $s^*$ of height $n$,

such that the cdf of this distribution is given by

$$F(t) = 1 - \alpha e^{-St} s^*, \text{for } t \geq 0.$$

Although, in this thesis, we only need the definition of the matrix exponential distribution, we note that different properties of such distributions can be found for example in [32].

## 2.3 Examples of (acyclic) PH distributions

In this section we provide several examples of PH distributions. The distributions presented here are all used throughout the thesis. These distributions are not just any PH distributions but in fact acyclic PH distributions:

**Definition 2.3.1** (Acyclic phase type distribution)**.** A PH distribution is called acyclic if it has a representation $(\alpha, S)$, where $S$ is an upper (or a lower) triangular matrix.

The definition implies that if an acyclic PH distribution has left some phase, it can never return to that phase (as otherwise this would create a "cycle"). We now proceed with examples.

### 2.3.1 Exponential distribution

The exponential distribution with parameter $\lambda$ can be seen as the simplest example of a PH distribution, by setting $\alpha = 1$ and $S = -\lambda$. The mean of the distribution is given by $\alpha(-S)^{-1}\mathbf{1} = 1/\lambda$, in agreement with Definition 2.1.1. Further, the cdf and pdf in Theorem 2.2.3 clearly are a generalisation of those in Definition 2.1.1.

### 2.3.2 Hyperexponential distribution

Hyperexponential distributions are so called "mixtures" of exponential distributions. More precisely, a random variable is called hyperexponentially distributed if there exist $n$ exponential distributions with parameters $\mu_1, \mu_2, \ldots, \mu_n$, for some $n > 0$, such that with probability $\alpha_k$ the variable is distributed according to the $k$-th of these exponential distributions. Such hyperexponential distribution has a $PH(\alpha, S)$ representation of $n$ phases where

$$\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n],$$

$$S = \begin{bmatrix} -\mu_1 & & & \\ & -\mu_2 & & \\ & & \ddots & \\ & & & -\mu_n \end{bmatrix}.$$

We also write that this distribution has a representation $HypExp(\alpha, [\mu_1, \ldots, \mu_n])$. As the mean of the $k$-th exponential distribution is $1/\mu_k$, the mean of a $HypExp(\alpha, [\mu_1, \ldots, \mu_n])$ distributed variable is

$$\alpha(-S)^{-1}1_n = \sum_{k=1}^{n} \frac{\alpha_k}{\mu_k}.$$

Every hyperexponential distribution has a representation such that $\alpha_i \neq 0$ and $\mu_i \neq \mu_j$ for every $i, j \in \{1, \ldots, n\}$, with $i \neq j$. In this case we say that the hyperexponential distribution is "of order $n$" and write $HypExp(n)$ for short.

We note that a hyperexponential distribution of order 2 can be described using the parameters $E[X], SCV, f$, where $E[X]$ is the mean of the distribution, where $SCV$ is the squared coefficient of variation and where $f$ is the "shape parameter".

Using these parameters we can generate a $HypExp(2)$ distribution with representation $HypExp([\alpha_1, 1 - \alpha_1], [\mu_1, \mu_2])$, where

$$\mu_1 = \frac{SCV + (4f - 1) + \sqrt{(SCV - 1)(SCV - 1 + 8f(1 - f))}}{2E[X]f(SCV + 1)},$$

$$\mu_2 = \frac{SCV + (4(1 - f) - 1) - \sqrt{(SCV - 1)(SCV - 1 + 8f(1 - f))}}{2E[X](1 - f)(SCV + 1)}$$

and $\alpha_1 = E[X]\mu_1 f$ [80, Subsection 4.3].

We make several remarks on the parameters $E[X], SCV$ and $f$. First, we note that $SCV$ is defined as

$$\frac{Var[X]}{E[X]^2},$$

where $Var[X] = E[X^2] - E[X]^2$ is the variance of the distribution. Suppose we work on $HypExp(2)$ jobs one after another, then the shape parameter $f$ can be though of as the fraction of time we work on jobs in phase 1 and $E[X]$ as the mean time to finish a job. Note, that if $f = 1/2$, then the $HypExp(2)$ distribution is said to have "balanced means". In this case it is possible that we come across jobs of one phase more often than jobs of the other phase, but the total amount of work needed by jobs of phase 1 is the same as that of phase 2 jobs.

### 2.3.3 Coxian distribution

A random variable $X$ has a Coxian distribution of order $n$ if it has a $PH(\alpha, S)$ representation with $\alpha = e_1$ (with $e_1$ a row vector with 1 in its leftmost entry and 0's elsewhere) and

$$S = \begin{bmatrix} -\mu_1 & p_1\mu_1 & & & \\ & -\mu_2 & p_2\mu_2 & & \\ & & \ddots & \ddots & \\ & & & -\mu_{n-1} & p_{n-1}\mu_{n-1} \\ & & & & -\mu_n \end{bmatrix},$$

with $0 < p_i \leq 1$ for $i = 1, \ldots, n-1$ and $\mu_i > 0$ for $i = 1, \ldots, n$ [76, Section 2].

If a job is Coxian distributed, we can think about the job in the following way: The job starts in phase 1. After an exponentially distributed amount of time with parameter $\mu_1$, it goes to phase 2 with probability $p_1$ or is finished with probability $1 - p_1$. In the first case, it stays in phase 2 for an exponentially distributed amount of time with parameter $\mu_2$, then it proceeds into phase 3 with probability $p_2$ or finishes with probability $1 - p_2$. And so on... If the job reaches phase $n$, then it finishes after an exponentially distributed amount of time with parameter $\mu_n$. It follows that the probability that a job reaches phase $j$ is $p_1 p_2 \ldots p_j = \prod_{s=1}^{j-1} p_s$. Therefore

$$\alpha(-S)^{-1}1_n = \sum_{j=1}^{n} \frac{1}{\mu_j} \prod_{s=1}^{j-1} p_s$$

is the expected duration of the job.

We note that every hyperexponential distribution of order $n$ can be represented by a Coxian distribution of order $n$ [76, Section 2]. This implies that hyperexponential distributions form a subclass of Coxian distributions. As we explain below, another subclass of Coxian distributions is the class of Erlang distributions.

### 2.3.4 Erlang distribution

A random variable $X$ is Erlang distributed if it is the sum of $n$ identically distributed and independent exponential variables with parameter $\mu$, for some $n, \mu > 0$. In this case we write $X \sim Erlang(n, \mu)$. An $Erlang(n, \mu)$ distribution has the following PH representation: $\alpha = e_1$ and

$$S = \begin{bmatrix} -\mu & \mu & & & \\ & -\mu & \mu & & \\ & & \ddots & \ddots & \\ & & & -\mu & \mu \\ & & & & -\mu \end{bmatrix},$$

with $S$ an $n \times n$ matrix.

If a job has an $Erlang(n, \mu)$ distribution, it starts in phase 1, stays there for an exponentially distributed amount of time with parameter $\mu$, goes into phase 2, stays there for an exponentially distributed amount of time with parameter $\mu$, and so on, until it reaches phase $n$. Then, after an exponentially distributed amount of time with parameter $\mu$, it finishes. We thus must have that the average time to finish such job is $\alpha(-S)^{-1}1_n = n/\mu$. An $Erlang(n, n)$ distribution therefore has mean 1. In the remainder of the text we shall write $Erlang(n)$ for an $Erlang(n, n)$ distribution.

Note, that by setting $p_i = 1$ for $i = 1, \dots, n-1$ and $\mu_i = \mu$ for $i = 1, \dots, n$ in Subsection 2.3.3, we immediately get that Erlang distributions form a subclass of Coxian distributions.

### 2.3.5 Hyper-Erlang distribution

Similarly to Subsection 2.3.2, where we defined hyperexponential distributions using exponential distributions, we can define hyper-Erlang distributions using Erlang distributions.

Hyper-Erlang distributions are mixtures of Erlang distributions. More precisely, a random variable is hyper-Erlang distributed if there exist $n$ Erlang distributions with parameters $(m_1, \mu_1), (m_2, \mu_2), \dots, (m_n, \mu_n)$, such that the variable is distributed as $Erlang(m_k, \mu_k)$ with probability $p_k$. Let $S_k$ denote the phase matrix of the PH representation of the $Erlang(m_k, \mu_k)$ distribution for $k = 1, \dots, n$. Then the hyper-Erlang distribution has the following $PH(\alpha, S)$ representation:

$$\alpha = [p_1, 0'_{m_1-1}, p_2, 0'_{m_2-1}, \dots, p_n, 0'_{m_n-1}],$$

$$S = \begin{bmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_n \end{bmatrix}.$$

As the distribution is $Erlang(m_k, \mu_k)$ with probability $p_k$, we must have that

$$\alpha(-S)^{-1}\mathbf{1} = \sum_{j=1}^{n} p_j \frac{m_j}{\mu_j}.$$

## 2.4 Other distributions

In this section we present two classes of distributions used in the thesis that are not examples of continuous PH distributions, namely discrete distributions and truncated Pareto distributions. When introducing discrete distributions we also define generating functions.

### 2.4.1 Discrete distribution

Discrete distributions are used to describe random variables with a finite or countably infinite number of outcomes. Examples include surveying a random person on the street for this person's age in years, number of owned cars, number of children, ...

Let $X$ be a random variable that takes values in $\mathbb{Z}$, that is every possible outcome of $X$ is an integer. Then $X$ is called a "discrete random variable" and the corresponding distribution a "discrete distribution". We call the set of values that $X$ can take the "support" of (the distribution of) $X$, that is $\{i \mid P(X = i) \neq 0\}$ is the support of $X$.

The cumulative distribution function (cdf) of $X$ in $x$, denoted as $F_X(x)$, is the probability that $X$ is smaller than or equal to $x$, i.e. $F_X(x) = P(X \leq x)$. Sometimes we suppress the dependence on $X$ and simply write $F(x)$. In case of discrete distributions, we have $F_X(x) = \sum_{i \leq x} P(X = i)$. Note that the expected value/mean of $X$ is given by

$$E[X] = \sum_{i=-\infty}^{+\infty} iP(X = i).$$

In this thesis all discrete distributions have positive, finite support, that is the discrete variable $X$ can only take values in $\mathbb{N} = \{0, 1, 2, \dots\}$ and there are only finitely many numbers in $\mathbb{N}$ that $X$ can be. In fact, we have already seen several examples of discrete distributions with positive finite support, namely all initial probability vectors $\alpha$ from Section 2.3 are examples of discrete distributions.

We now introduce the generating functions, which concerns non-negative discrete distributions.

**Definition 2.4.1.** Let $X$ be a non-negative discrete random variable. The generating function of $X$ is defined as

$$P_X(z) = \sum_{n=0}^{\infty} P(X = n)z^n.$$

In Sections 3.5 and 10.4 we use the following properties of the generating functions:

**Theorem 2.4.2.** *Let $X$ be a non-negative discrete random variable. The generating function of $X$ has the following properties:*

$$P_X(1) = 1, \quad P'_X(1) = E[X] \quad \textit{and} \quad P''_X(1) = E[X(X-1)].$$

A thorough treatment on generating functions can be found in [29, Section 5], while [1, Section 2.2] contains a brief summary of the properties of generating functions.

### 2.4.2 Truncated Pareto distribution

In this thesis we use the truncated Pareto distribution in Chapter 10. In that chapter we developed formulas for general distributions, hence we also want to use them with a distribution that is not of the phase type. As the Pareto distributions are used to describe a wide variety of real-life phenomena, we chose to use those distributions.

The Pareto distribution is characterized by three values: $\alpha, L$ and $U$, with $0 < L < U \leq +\infty$. The value of $\alpha$ is called the shape parameter, while $L$ and $U$ respectively denote the lower and upper bound of the support of the distribution. Note that $U$ can be finite or infinite. In the former case the Pareto distribution is called "upper truncated" or, simply, "truncated". In this thesis we only consider truncated Pareto distributions. For $\alpha > 0$, the cdf of the truncated Pareto distribution is given by

$$F(x) = \begin{cases} \frac{1-(L/x)^\alpha}{1-(L/U)^\alpha}, & \text{if } L \leq x \leq U \\ 0 & \text{otherwise.} \end{cases}$$

For $\alpha \notin \{0, 1\}$, the mean of the truncated Pareto distribution is given by

$$E[X] = \frac{\alpha L}{\alpha - 1} \cdot \frac{1-(L/U)^{\alpha-1}}{1-(L/U)^\alpha} = \frac{\alpha L U}{\alpha - 1} \cdot \frac{U^{\alpha-1} - L^{\alpha-1}}{U^\alpha - L^\alpha}.$$

These and other properties of truncated Pareto distributions are summarized in [13].

## 2.5 Laplace-Stieltjes transforms

In this section we define the Laplace-Stieltjes transform of a random variable, which is used in Sections 3.5 and 10.4. We also remark on some properties of Laplace-Stieltjes transforms. We then illustrate how these properties can be used on an example.

**Definition 2.5.1.** Let $X$ be a random variable with $G$ as its distribution. The Laplace-Stieltjes transform (LST) of $G$ in $s$, denoted by $G^*(s)$, is defined as

$$G^*(s) = E\left[e^{-sX}\right] = \int_{x=0}^{\infty} e^{-sx} dF_X(x),$$

for $s \geq 0$, where $F_X$ is the cdf of $X$.

**Theorem 2.5.2.** *Let $X$ and $Y$ be random variables. Let $G$ and $H$ be the distribution of $X$ and $Y$ respectively and $G^*(s)$ and $H^*(s)$ their respective Laplace-Stieltjes transforms. Then*

$$G^{*\prime}(0) = -E[X] \quad \textit{and} \quad G^{*\prime\prime}(0) = E[X^2].$$

*If $Z$ is a random variable (with distribution $I$) that is equal to $X$ with probability $p$ and equal to $Y$ with probability $1-p$, then the LST of $I$ in $s$ is given by*

$$I^*(s) = pG^*(s) + (1-p)H^*(s).$$

*If $X$ and $Y$ are independent and $Z = X + Y$, then*

$$I^*(s) = G^*(s)H^*(s).$$

**Proposition 2.5.3.** *Let $X \sim \exp(\mu)$ and let $G$ denote the distribution of $X$. We have*

$$G^*(s) = \frac{\mu}{\mu + s}.$$

These and other properties of LSTs can be found in [1, Sections 2.3-2.4]. In the following proposition, we apply the properties of Theorem 2.5.2 and Proposition 2.5.3.

**Proposition 2.5.4.** *Let $0 \le p_1 \le 1$ and let $\mu_1, \mu_2 > 0$. Let $X \sim PH(\alpha, S)$ with $\alpha = (1,0)$ and*

$$S = \begin{bmatrix} -\mu_1 & p_1\mu_1 \\ 0 & -\mu_2 \end{bmatrix}.$$

*Denote $\tilde{p} = 1 - p_1\mu_1/(\mu_1 - \mu_2)$. Let $Y \sim PH(\tilde{\alpha}, \tilde{S})$, where $\tilde{\alpha} = (\tilde{p}, 1-\tilde{p})$ and*

$$\tilde{S} = \begin{bmatrix} -\mu_1 & 0 \\ 0 & -\mu_2 \end{bmatrix}.$$

*Then $X$ and $Y$ have the same distribution.*

*Proof.* We prove the claim by showing that the LSTs of the distributions of $X$ and $Y$ are the same. $X \sim \exp(\mu_1)$ with probability $1 - p_1$ and with probability $p_1$ it is the sum of an $\exp(\mu_1)$ and an $\exp(\mu_2)$ distributed variable (with the two random variables independent). The LST of the distribution of $X$ in $s$ is therefore given by

$$(1-p_1)\frac{\mu_1}{\mu_1 + s} + p_1\frac{\mu_1}{\mu_1 + s}\frac{\mu_2}{\mu_2 + s}. \tag{2.1}$$

$Y$ is exponentially distributed with parameter $\mu_1$ (respectively, parameter $\mu_2$) with probability $\tilde{p}$ (respectively $1 - \tilde{p}$). Its LST is therefore

$$\tilde{p}\frac{\mu_1}{\mu_1 + s} + (1-\tilde{p})\frac{\mu_2}{\mu_2 + s}. \tag{2.2}$$

Noting that both (2.1) and (2.2) can be rewritten as

$$\frac{\mu_1[\mu_2 + (1-p_1)s]}{(\mu_1 + s)(\mu_2 + s)}$$

finishes the proof. □

Chapter 3

# Some elementary single server queueing systems

In this chapter we present four important single server queueing systems, namely (in Kendall notation) the $M/M/1$, $M/PH/1$, $M/MAP/1$ and the $M/G/1$ queue. When presenting these queues we shall explain several core definitions and techniques used in this thesis. We also remark on several matrix identities that can be useful when dealing with the $M/PH/1$ queue and similar queues.

The rest of the chapter is structured as follows. We first explain the Kendall notation in Section 3.1. In Section 3.2 we describe the $M/M/1$ queue. There, we also touch on the concepts of irreducibility, positive recurrence, invariant distribution, balance equations and the response time. In Section 3.2, we also note two important results from queueing theory, namely the Little's law and the PASTA property. Section 3.3 deals with a generalization of $M/M/1$ queue, namely the $M/PH/1$ queue. There, we explain how the $M/PH/1$ queue can be modelled using a QBD MC and briefly touch on matrix analytic methods. In Section 3.4, we introduce the $M/MAP/1$ queue and provide a way of numerically determining its invariant distribution. The section also contains an explanation on how the $M/M/1$ and the $M/PH/1$ queues can be seen as an $M/MAP/1$ queue. A generalization of the three previous queues, the $M/G/1$ queue, is summarized in Section 3.5. In that section, we provide two ways of deriving the mean response time of the queue, namely through the use of mean value analysis and through generating functions. Finally, in Section 3.6, we remark on several important matrix identities used from Chapter 7 onward.

## 3.1 Kendall notation

The Kendall notation is used to classify elementary queueing systems. These usually consist of one or multiple queues each, with one or multiple service stations called servers. The Kendall notation was introduced in 1953 by David G. Kendall in [41]. Originally, it consisted of three symbols $A/S/N$, where $A$ describes the arrival process, $S$ is the service time distribution and $N$ denotes the number of servers of the queueing system.

$A$**, the arrival process.** The first of the three symbols is used to describe the arrival process, that is, $A$ describes in what way jobs arrive to the queueing system. In all queueing systems presented in this chapter, the arrival process is Poissonian (or equivalently the times between arrivals are exponential, cf. Section 2.1). This is denoted by $A = M$, where $M$ stands for "memoryless", as the exponential distribution is memoryless. In the next four sections, we thus assume that jobs arrive to the system according to a Poisson process with parameter $\lambda$.

Although not used in this thesis, we make note of some other arrival disciplines:

- $A = GI$, where $GI$ stands for "General Input" and means that the time between successive arrivals is distributed according to a general distribution. We can also specify this distribution, for example, when $A = E$, then the interarrival times are Erlang distributed. (This is also the reason why we write $A = M$ instead of $A = E$ for exponential interarrival times.)
- Batch arrivals $A^X$: every time an arrival event occurs, a batch of $X$ jobs is added to the system, where $X$ can be a natural number or a discrete random variable taking values in $\mathbb{N}$.

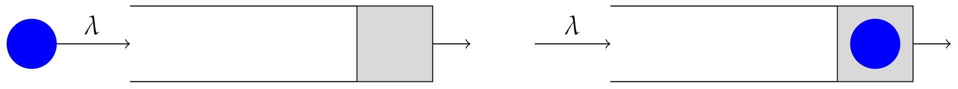
$S$**, the service time distribution.** The second parameter describes the distribution of the time that the server needs to finish a job. In the remainder of the chapter we consider the following cases:

- $S = M$, memoryless or Markovian service, i.e. the service time distribution is exponential (cf. Section 2.1).
- $S = PH$, the service time is distributed according to a distribution of the phase type.
- $S = MAP$, where $MAP$ stands for "Markovian arrival process". As the name suggests, in many cases, Markovian arrival processes are used as arrival disciplines. However, in Part III of this thesis we make use of queues with $MAP$s as a service discipline.
- $S = G$, general service times. The service time is distributed according to some (unknown) distribution $G$.

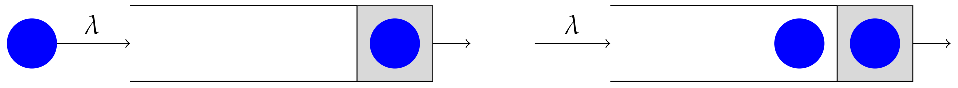
$N$**, the number of servers.** The third parameter denotes the number of servers, i.e. the number of service stations where jobs are served. $N$ can be any positive natural number or infinite. In the examples presented in the next sections we assume that there is only one server, i.e. $N = 1$.

**Other parameters.** We note that since its original version, the Kendall notation has been expanded to also include the buffer of the queue, the population of jobs that need to be served and the service discipline. In case of infinite buffer/population, these parameters are not included in the Kendall notation.

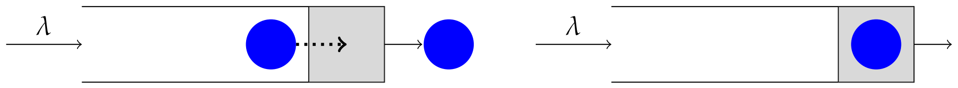
The buffer of the queue is simply the amount of jobs that can wait in the queue to be served. Think for example of a store that is full to the brim with people; no new customers



(a) A job enters the queue. As the server is not working on a job, the arriving job enters service.



(b) Another job enters the queue. As the server is busy, the job begins waiting in the queue.



(c) The server finishes working on the job in service. That job leaves the queue and the next job enters service.

Figure 3.1: A schematic representation of the $M/M/1$, $M/PH/1$, $M/MAP/1$ and $M/G/1$ queues.

can enter the store until some customers leave the store. In the queues presented in the next sections the buffer is assumed to be infinite.

As we are interested in the long term behaviour of queues (i.e. as time goes to infinity), we assume that the population of jobs needing service also is infinite. Otherwise, the queues have an infinite amount of time to process a finite number of jobs, which they will do in finite time. In this case the fraction of time that the servers are busy is 0.

The last of the other parameters is the service discipline (also called the "scheduling policy"), that is, in what way do the servers handle the jobs. Some important service disciplines include

- First-in-first-out (FIFO), also called first-come-first-served (FCFS): the jobs get served in the order of their arrival, that is, the oldest job gets processed first. In this case, the service discipline is not written in the Kendall notation. The next four sections thus deal with queuing systems with the FIFO service discipline.
- Last-in-first-out (LIFO): the newest job gets served first.
- Processor sharing (PS): the server processes all jobs in the queue at the same time, giving a fraction (1 divided by the number of jobs present in the queue) of its processing power to each of the jobs. In Chapter 5 we study a queueing system with processor sharing discipline.
- Random order of service (ROS): jobs are executed in a random order, irrespective of their arrival times.

Schematically, the $M/M/1$, $M/PH/1$, $M/MAP/1$ and $M/G/1$ queues can be thus represented as in Figure 3.1.

## 3.2 The $M/M/1$ queue

We first give an overview of the $M/M/1$ queue. This section is loosely based on [78]. Another overview of the $M/M/1$ queue can be found in [1, Chapter 4]. Suppose jobs arrive to the queue according to a Poisson process with rate $\lambda$. Suppose that the jobs have exponentially distributed service requirements with parameter $\mu$, that is, the time it takes for a single job to be processed by the server is $\exp(\mu)$ distributed.

We wish to describe the queue using a (continuous time) Markov chain. First we determine the "state space" of the CTMC. Throughout this thesis we will denote the state space by $\Omega$. Roughly stated, the state space is the set of all states the CTMC can be in. As the exponential distribution is memoryless, to describe the queue at a certain point in time, we only need to know the number of jobs in the queue. This implies that the state space of the $M/M/1$ queue is given by $\Omega = \mathbb{N}$.

We now describe the transitions between different states. At any time, the state can increase by one (at rate $\lambda$) due to an incoming arrival. If there are jobs present in the queue, the queue length decreases by one at rate $\mu$ due to jobs being finished by the server. This implies that the queue can be described using a CTMC with the rate matrix

$$Q = \begin{bmatrix} -\lambda & \lambda & & & \\ \mu & -\mu-\lambda & \lambda & & \\ & \mu & -\mu-\lambda & \lambda & \\ & & \ddots & \ddots & \ddots \end{bmatrix}.$$

This type of MC is an example of a "Birth-and-Death" MC. Note, that if $\mu, \lambda \neq 0$, then the MC can reach any given state from any other state. This is what is commonly referred to as "irreducibility".

Another important notion concerning MCs is that of recurrence. A state $i$ of a Markov chain is called "positive recurrent" if after leaving state $i$ the MC will return to state $i$ at some point with probability 1 and if the mean time that it takes for the MC to return to state $i$ is finite. If every state of a Markov chain is positive recurrent then we call the MC "positive recurrent". Note that an irreducible Markov chain with a finite state space is always positive recurrent.

In case of the $M/M/1$ queue, one can show that its CTMC is positive recurrent if and only if $\lambda/\mu < 1$. This inequality is intuitively clear, as it states that the rate at which jobs get completed by the server should be higher than the rate at which jobs arrive to the queue (otherwise, on average, the queue length would keep increasing as time goes on). The ratio $\lambda/\mu$ is referred to as the "load" of the queue and denoted by $\rho$. In this thesis, the load will be always equal to the arrival rate per queue multiplied with the average time needed to complete a job. The load is the fraction of time that the queue is busy (i.e. that the server is working on a job) as time becomes large.

As we are interested in the long term behaviour of the queueing systems, we would also like the know the probability that the CTMC is in certain state as time goes to infinity. Under certain assumptions, which we will note further, these probabilities can be studied though invariant distributions:

**Definition 3.2.1.** Let $Q$ be the rate matrix of a CTMC. We call a vector $\pi$ the "invariant distribution", "stationary distribution" or the "steady state probability vector" of the CTMC if $\pi Q = 0$ and $\pi\mathbf{1} = 1$ (the entries of $\pi$ sum to 1).

The equation $\pi\mathbf{1} = 1$ simply comes from the fact that we want $\pi$ to be a distribution. Note, that we will provide some intuition on the equation $\pi Q = 0$ in the proof of Theorem 3.2.3.

In case of CTMCs, irreducibility and positive recurrence are two properties we need for the existence of the invariant distribution $\pi$:

**Theorem 3.2.2.** *Let $Q$ be the rate matrix of an irreducible, positive recurrent CTMC with state space $\Omega$. The limit $\pi_j = \lim_{t\to\infty} p_{i,j}(t)$ exists, for every $i, j \in \Omega$, where $p_{i,j}(t)$ is the probability that the CTMC is in state $j$ at time $t$, given that it was in state $i$ at time 0. Further, the vector $\pi = (\pi_j)_{j\in\Omega}$ is the unique stationary distribution of the CTMC.*

*Proof.* A proof can be found in [14, Theorem 3.5.]. □

In case of the $M/M/1$ queue we have the following result for the invariant distribution:

**Theorem 3.2.3.** *For the $M/M/1$ queue with arrival rate $\lambda$ and service rate $\mu$, with $\lambda < \mu$, we have*

$$\pi_k = (1-\rho)\rho^k,$$

*for $k > 0$ and $\pi_0 = 1 - \rho$, where $\rho = \lambda/\mu$.*

*Proof.* We first provide some intuition behind Definition 3.2.1. If we write $\pi Q = 0$ out in its component form, then the equations we obtain are called "(global) balance equations", as they state that the rate at which the CTMC enters a certain state is the rate at which it leaves that state if the CTMC is in equilibrium. We also have the, so called, "detailed balance equations". They state that the rate at which the MC in equilibrium up-crosses a certain threshold state should be equal to the rate at which it down-crosses that threshold.

For the $M/M/1$ queue, the detailed balance equations are

$$\lambda\pi_{k-1} = \mu\pi_k,\ k \geq 1,$$

where we have chosen $k$ as the threshold state. This is equivalent to $\pi_k = \rho\pi_{k-1}$ which immediately implies that $\pi_k = \pi_0\rho^k$. From $\pi\mathbf{1} = 1$, one can then obtain $\pi_0 = 1 - \rho$ as needed.

Note, that this result is also an immediate consequence of the formula in [30, Section 2.1] or [48, Theorem 4.5.1]. □

So, for an $M/M/1$ queue we know $\pi_k$, the probability that the queue has $k$ jobs if observed at an arbitrary time instant. But what can we say about $\pi_k^a$, the probability that the queue has $k$ jobs given that an arrival just occurred? It turns out that for many stochastic systems these two probabilities are the same, that is $\pi_k = \pi_k^a$. These stochastic systems include those with Poissonian arrival processes:

**Theorem 3.2.4** (Poisson Arrivals See Time Averages, PASTA property)**.** *Let $\pi$ denote the invariant distribution of an irreducible, positive recurrent CTMC. Let $\pi_k^a$ denote the limiting probability that the CTMC is in state $k$ upon an arrival. If the number of future arrivals is independent of the current and past states of the CTMC, then $\pi_k = \pi_k^a$ for every $k \in \Omega$.*

*Proof.* A formal proof can be found in [84]. □

When studying the performance of a queueing system we are especially interested in the mean response time, with the response time being the amount of time that a job spends in the system. The mean response time can be obtained from the average queue length by using the following result:

**Theorem 3.2.5** (Little's law)**.** *Let*

- *$L$ be the average number of customers in the queue up to time $T$ as $T \longrightarrow \infty$,*
- *$\lambda$ be the average number of arrivals per time unit up to time $T$ as $T \longrightarrow \infty$, and*
- *$R$ be the average response time of the first $n$ jobs as $n \longrightarrow \infty$.*

*If $L, \lambda$ and $R$ are finite, then $L = \lambda R$.*

*Proof.* A proof can be found in [71]. □

Note, that Little's law can be also used on a part of a queueing system, which makes it extremely useful.

From the invariant distribution we can obtain the mean queue length as it is simply the expected value of $\pi$.

**Theorem 3.2.6.** *For the $M/M/1$ queue with arrival rate $\lambda$ and service rate $\mu$, with $\lambda < \mu$, the mean queue length and the mean response time are respectively given by*

$$E[Q] = \frac{\rho}{1-\rho} \text{ and } E[R] = \frac{1}{\mu - \lambda}.$$

*Proof.* The first claim can be proven by calculating $\sum_{k\in\mathbb{N}} k\pi_k$, the second follows from the first claim and Little's law. □

The response time can be split up in the waiting and service time. The first of these is the time that a job waits in the queue before entering service, while the second is the amount of time that a server works on a job until the job leaves the system. From the fact that the mean service time is $1/\mu$ it follows from the last result that the mean waiting time $E[W]$ in an $M/M/1$ queue is given by $E[W] = 1/(\mu - \lambda) - 1/\mu$.

## 3.3 The $M/PH/1$ queue

In this section we summarize the important results on the $M/PH/1$ queue. This queue is a generalization of $M/M/1$ queue (as PH distributions generalize the exponential distribution). Suppose jobs arrive to the queue according to a Poisson process with rate $\lambda$. Suppose further that the job size distribution has a PH representation of $n_p$ phases with parameters $(\alpha, S)$ and denote $s^* = -S\mathbf{1}$.

Similarly to the previous, section we can describe the $M/PH/1$ queue using a Markov chain. Knowing the number of jobs in the queue is not enough to describe the state of the $M/PH/1$ queue: we also need to know the phase of the job in service (when the queue is not empty). Hence, the state space of the queue is given by

$$\Omega = \{0\} \cup \{(i,j) | i = 1,2,\dots \text{ and } j = 1,\dots,n_p\},$$

where the first component of $(i,j)$ denotes the number of jobs in a busy queue and $j$ denotes the phase of the job in service. We use the lexicographical ordering on the states, that is, $(i,j) <_{lex} (i',j')$ if $i < i'$ or if $i = i'$ and $j < j'$. We will call $i$ the "level" and $j$ the "phase" of the queue. This way states representing the same number of jobs in the queue are grouped together.

We now describe the transitions between the states. When the queue is busy, phase changes of the job in service leave the level unchanged. This means that block-diagonal entries of the rate matrix $Q$ will consists of the matrix $S$ at every level, except on the diagonal. The level in the queue is increased by one due to arrivals, which occur at rate $\lambda$. If the queue is empty and an arrival occurs, the queue will jump from state 0 to $(1,j)$ with probability $\alpha_j$. Job completions from any state with phase $i$ occur at rate $s^*_i$ and decrease the level by 1. If the queue is non-empty after completion the next job immediately enters service and its initial phase is $j$ with probability $\alpha_j$. This implies that the rate matrix of the $M/PH/1$ queue is given by

$$Q = \begin{bmatrix} -\lambda & \lambda\alpha & & & & \\ s^* & S - \lambda I_{n_p} & \lambda I_{n_p} & & & \\ & s^*\alpha & S - \lambda I_{n_p} & \lambda I_{n_p} & & \\ & & s^*\alpha & S - \lambda I_{n_p} & \lambda I_{n_p} & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.1}$$

This type of MC is an example of a "Quasi-Birth-and-Death" MC, or QBD for short. Note that if $\lambda \neq 0$, then $(\alpha, S)$ can be chosen is such a way that the MC is irreducible. We assume that this is the case throughout the rest of this section.

The CTMC is positive recurrent if and only if the load is smaller than 1, that is if

$$\rho = \lambda\alpha(-S)^{-1}1_{n_p} < 1,$$

as $\alpha(-S)^{-1}1_{n_p}$ denotes the mean time to finish a job (cf. Theorem 2.2.7).

We now wish to describe the invariant distribution. We first have to introduce some notation. Suppose an invariant distribution $\pi$ of the $M/PH/1$ queue exists, we then set for $\ell = 1,2,\dots$

$$\pi_\ell = [\pi_{\ell,1}, \pi_{\ell,2}, \dots, \pi_{\ell,n_p}].$$

We have the following result:

**Theorem 3.3.1.** *For the $M/PH/1$ queue with arrival rate $\lambda$ and job size distribution $PH(\alpha, S)$, with $\rho < 1$, we have $\pi_0 = 1 - \rho$ and for $\ell \geq 1$ that*

$$\pi_\ell = (1-\rho)\alpha R^\ell,$$

*where the matrix $R$ is given by*

$$R = \lambda \left(\lambda I_{n_p} - \lambda 1_{n_p}\alpha - S\right)^{-1}.$$

*Proof.* See [27, Sections 1-2]. □

Note that $R$ is the smallest non-negative solution to the quadratic matrix equation $\lambda I_{n_p} + R(S - \lambda I_{n_p}) + R^2 s^* \alpha = 0$ [48, Section 6.4]. Throughout this thesis we will need to solve similar quadratic matrix equations for matrix $R$. In some cases, we will have explicit results for $R$, in other cases $R$ can only be found numerically. Note that this technique, where we have to find the matrix $R$ to determine $\pi$, is an example of a matrix analytic method. These methods are used to compute the invariant distribution when the rate matrix has a repeated structure (which clearly is the case for the $M/PH/1$ queue).

Similarly to Theorem 3.2.6 we have the following:

**Theorem 3.3.2.** *For the $M/PH/1$ queue with arrival rate $\lambda$ and job size distribution $PH(\alpha, S)$, with $\rho < 1$, we have*

$$E[R] = \alpha(-S)^{-1}1_{n_p} + \frac{\lambda}{1-\rho}\alpha(-S)^{-2}1_{n_p}.$$

*Proof.* The claim can be proven by first calculating $\sum_{\ell=1}^{\infty} \ell \pi_\ell 1_{n_p}$ (although this is not as simple as in the case of exponential job sizes) and by using Little's law. Alternatively, we can also use the formula from Theorem 3.5.1 by noting that the second moment of a PH distribution is given by $2\alpha(-S)^{-2}1_{n_p}$ and its mean service time by $\alpha(-S)^{-1}1_{n_p}$. □

As the mean service time is given by $\alpha(-S)^{-1}1_{n_p}$, the mean waiting time of an $M/PH/1$ queue is therefore given by $E[W] = \frac{\lambda}{1-\rho}\alpha(-S)^{-2}1_{n_p}$. Note that one can easily check that this formula and the formulas from Theorem 3.3.1 and 3.3.2 generalize those from Theorem 3.2.3 and 3.2.6.

## 3.4 The $M/MAP/1$ queue

We first give an intuitive explanation on the Markovian arrival process ($MAP$). Then, we describe the $M/MAP/1$ queue as a CTMC and show how we can numerically find its invariant distribution (Theorem 3.4.1). Finally, we explain how the $M/M/1$ and the $M/PH/1$ queues can be seen as examples of an $M/MAP/1$ queue (Examples 3.4.2 and 3.4.3 respectively).

We now give an explanation on *MAP*s. Note that other descriptions of the Markovian arrival process can be found, for example, in [53, Section 2.1.]. A *MAP* service is characterized by two $n \times n$ matrices $S_0$ and $S_1$. Denote $S = S_0 + S_1$. We assume that the following holds for the matrices $S_0$ and $S_1$:

- Every entry of $S_1$ is non-negative.
- The diagonal entries of $S_0$ are negative and non-diagonal entries of $S_0$ are non-negative.
- The row sums of $S = S_0 + S_1$ are 0, i.e. $S1_n = 0$. In other words, $S$ is a rate matrix of a CTMC.

Sometimes (f.e. in [53, 85]), the matrix $S$ is such that the CTMC with rate matrix $S$ is irreducible and positive recurrent. In this case, due to Theorem 3.2.2, there exists a unique invariant distribution, say $\gamma$, of the CTMC with rate matrix $S$, meaning $\gamma S = 0$ and $\gamma 1_n = 1$. In this case we call the *MAP* service "irreducible". As explained further, at the very least one has to make the following assumption on $S_0$ and $S_1$:

- The matrices $S_0$ and $S_1$ are such that for every state $i$ of the CTMC with rate matrix $S$, the CTMC can reach a state $j$, with the $j$-th row of $S_1$ containing a non-zero entry.

For ease of presentation, we shall henceforth refer to the states of the CTMC with rate matrix $S$ as the states of the *MAP*. We denote the Markovian arrival process characterized by the matrices $S_0$ and $S_1$ as $MAP(S_0, S_1)$.

We now explain how a $MAP(S_0, S_1)$ service works. Let $(S_k)_{i,j}$ denote the entry $(i, j)$ of the matrix $S_k$ for $k = 0, 1$. Suppose that the *MAP* service just entered state $i$. At this point $2n - 1$ exponential timers get started:

- The first, second,. . . , $(i - 1)$-st timer counts down time drawn at random from exponential distribution with parameter $(S_0)_{i,1}, (S_0)_{i,1}, \dots, (S_0)_{i,i-1}$ respectively.
- The $i$-th, $(i + 1)$-st,. . . , $(n - 1)$-st timer counts down time drawn at random from exponential distribution with parameter $(S_0)_{i,i+1}, (S_0)_{i,i+2}, \dots, (S_0)_{i,n}$ respectively.
- The $n$-th, $(n + 1)$-st,. . . , $(2n - 1)$-st timer counts down time drawn at random from exponential distribution with parameter $(S_1)_{i,1}, (S_1)_{i,2}, \dots, (S_1)_{i,n}$ respectively.

When one of the $2n - 1$ timers runs out of time, one of the following things happens, depending on which timer ran out of time:

- If it was one of the first $n - 1$ timers, say, the one corresponding to the entry $(S_0)_{i,j}$, then the state of the *MAP* changes from $i$ to $j$.
- If it was one of the latter $n$ timers, say, the one corresponding to the entry $(S_1)_{i,j}$, a service completion occurs. (This explains why we need the last assumption on $S_0$ and $S_1$.) If $i \neq j$, then the state of the *MAP* changes from $i$ to $j$. If $i = j$, the *MAP* stays in state $i$, yet the $2n - 1$ timers get reset.

Note, that if $(S_k)_{i,j} = 0$, then the corresponding timer never runs out of time.

We now proceed with the description of the $M/MAP/1$ queue. Jobs arrive to the queue according to a Poisson process, with parameter $\lambda > 0$. Note, that the arrival of a job does not change the state of the $MAP$. While there are jobs in the queue the $MAP$ runs as described above. Whenever a service completion occurs (with $MAP$ in state $j$), the first job in queue leaves service and the next one (if available) enters service (with $MAP$ in state $j$). Technically, to fully characterize the queue, we have to specify how many jobs the queue has at time 0 and in which state the $MAP$ is at time 0. This is however irrelevant for the long term behaviour of the queue, provided that the $MAP$ service is irreducible.

To know the state of the queue we only need to keep track of the number of jobs in the queue and the state of the $MAP$. In other words, the state space of the $M/MAP/1$ queue is given by

$$\Omega = \{(\ell, i) | \ell \in \mathbb{N}, i \in \{1, \dots, n\}\}.$$

Similarly to the previous section, we use the lexicographical ordering on the states.

Based on the discussion above, the $M/MAP/1$ queue can be described using the CTMC with rate matrix

$$Q = \begin{bmatrix} -\lambda I_n & \lambda I_n & & & \\ S_1 & S_0 - \lambda I_n & \lambda I_n & & \\ & S_1 & S_0 - \lambda I_n & \lambda I_n & \\ & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.2}$$

Similarly to (3.1), the rate matrix (3.2) has a QBD structure. It should be therefore possible to numerically find the invariant distribution of the queue (provided it exists). Similarly to the previous section, we introduce the following notation: suppose an invariant distribution $\pi$ of the $M/MAP/1$ queue exists, we then set for $\ell = 0, 1, \dots$

$$\pi_\ell = [\pi_{\ell,1}, \pi_{\ell,2}, \dots, \pi_{\ell,n}].$$

Suppose now that the CTMC with rate matrix $S$ is irreducible (and positive recurrent) and let $\gamma$ be its invariant distribution. The average rate at which jobs get completed is then $\gamma S_1 1_n$. It follows that the system is stable if $\lambda < \gamma S_1 1_n$. In this case, the load of the system is given by $\rho = \lambda/(\gamma S_1 1_n)$.

Under the above assumptions, we can numerically find the invariant distribution of the $M/MAP/1$ queue:

**Theorem 3.4.1.** *Suppose we have an $M/MAP/1$ queue with arrival rate $\lambda$ and irreducible $MAP(S_0, S_1)$ service, with $\rho = \lambda/(\gamma S_1 1_n) < 1$. Let the matrices $R$ and $G$ be the solutions to the equations*

$$\begin{aligned} \lambda I_n + R(S_0 - \lambda I_n) + R^2 S_1 &= 0 \\ S_1 + (S_0 - \lambda I_n)G + \lambda G^2 &= 0. \end{aligned}$$

*Then:*

- *The vector $\pi_0$ can be numerically obtained as a solution to the following set of equations*

$$\begin{cases} \pi_0(G - I_n) = 0, \\ \pi_0(I_n - R)^{-1} 1_n = 1. \end{cases}$$

- *For $\ell \geq 1$ we have $\pi_\ell = \pi_0 R^\ell$.*

*Proof.* See [48, Section 6.4]. □

Note that, in general, the matrices $R$ and $G$ from Theorem 3.4.1 can be numerically obtained by using the solver in [5]. Also note, that we have $\pi_0 1_n = 1 - \rho = 1 - \lambda/(\gamma S_1 1_n)$.

We finish this section by showing how the $M/M/1$ and the $M/PH/1$ queues can be seen as examples of the $M/MAP/1$ queue.

**Example 3.4.2** (The $M/M/1$ queue)**.** An $M/M/1$ queue with service rate $\mu$, can be seen as an $M/MAP/1$ queue with $MAP(-\mu, \mu)$ service. In this case $S = 0$, $n = 1$ and $\gamma = 1$.

**Example 3.4.3** (The $M/PH/1$ queue)**.** We describe an $M/PH/1$ queue with $PH(\alpha, \tilde{S})$ jobs of $n$ phases as an $M/MAP/1$ queue by setting $S_0 = \tilde{S}$ and $S_1 = s^*\alpha$ (with $s^* = -\tilde{S}1_n$) in (3.2). This can be easily seen by comparing the rate matrices (3.1) and (3.2): the only difference is that, when the queue is empty, in (3.1) the initial phase of the next job is chosen upon the arrival of said job, while in (3.2) that phase is already chosen at the time of completion of the previous job.

## 3.5 The $M/G/1$ queue

In this section we summarize important results on the $M/G/1$ queue. Suppose that jobs arrive to the queue according to a Poisson process with parameter $\lambda$ and suppose further that job sizes are distributed according to a general distribution $G$. Note that the load of the queue is then given by $\rho = \lambda E[G]$. Suppose that $\rho < 1$. Due to the generality of $G$, we cannot describe the $M/G/1$ queue using a CTMC, as we did for the $M/M/1$, $M/PH/1$ and $M/MAP/1$ queues. Despite this we still can provide formulas for the mean response time and mean waiting time of the queue.

A way of finding mean response and waiting times is through the use of the, so called, mean value analysis. Due to PASTA property the average waiting time of the queue is the same as the average waiting time that an arriving job experiences. This waiting time is 0 if the job arrives when the queue is empty, otherwise the job has to wait until all previous jobs get serviced. If the queue is busy, the incoming job also has to wait on the job in service to leave the system. Note that the latter wait is called the "residual service time". Note further, that the probability that the server is working on a job upon an arrival is $\rho$ (once again due to PASTA). Let $E[L_q]$ denote the average number of waiting jobs in the buffer of the queue (thus not counting the possible job in service). We then have that the mean waiting time equals

$$E[W] = E[L_q]E[G] + \rho E[\text{residual service time} \mid \text{a job is in service}].$$

By using Little's law on the buffer of the queue, we obtain $E[W] = \lambda E[L_q]$. Substituting this in the above equation and solving for $E[W]$ gives

$$E[W] = \frac{\rho}{1-\rho} E[\text{residual service time} \mid \text{a job is in service}].$$

It can be shown that the last expected value equals $E[G^2]/(2E[G])$ and we thus obtain:

**Theorem 3.5.1.** *For the $M/G/1$ queue with arrival rate $\lambda$ and job size distribution $G$, with $\rho < 1$, we have*

$$E[W] = \frac{\rho}{1-\rho} \cdot \frac{E[G^2]}{2E[G]} \text{ and } E[R] = E[G] + \frac{\rho}{1-\rho} \cdot \frac{E[G^2]}{2E[G]}.$$

The above derivation can be found for example in [30, Subsection 5.1.1] or in [1, Section 7.6]. Note, that we use a similar method in the second proof of the formula (10.6).

Another way of deriving this result is through the use of generating functions (cf. Definition 2.4.1). Let $\pi(z), \pi^a(z)$ and $\pi^d(z)$ denote the generating functions of: the invariant distribution, the distribution of the system observed upon an arrival and the distribution of the system observed upon a departure. Due to PASTA we must have $\pi(z) = \pi^a(z)$ for the $M/G/1$ queue. Due to the following result $\pi^d(z)$ also equals $\pi(z)$:

**Theorem 3.5.2.** *For the $M/G/1$ queue with arrival rate $\lambda$ and job size distribution $G$, with $\rho < 1$, we have $\pi(z) = \pi^d(z)$, meaning that the invariant distribution is the same as the distribution of the queue observed at departure times.*

*Proof.* See [30, Subsection 5.1.3]. □

The generating function $\pi(z)$ is given by:

**Theorem 3.5.3.** *For the $M/G/1$ queue with arrival rate $\lambda$ and job size distribution $G$, with $\rho < 1$, we have*

$$\pi(z) = \frac{(1-\rho)(1-z)G^*(\lambda - \lambda z)}{G^*(\lambda - \lambda z) - z},$$

*where $G^*(s)$ is the Laplace-Stieltjes transform of the job size distribution (cf. Definition 2.5.1).*

*Proof.* A proof can be found in [30, Subsections 5.1.2 and 5.1.5] or in [1, Section 7.2]. Note that this proof uses the result from Theorem 3.5.2. □

By using Theorem 3.5.3 together with Theorem 2.4.2 and Little's law, the mean response time of the $M/G/1$ queue can be now obtained as $E[R] = \pi'(1)/\lambda$. Note, that the formulas in Theorems 3.5.1 and 3.5.3 are commonly referred to as Pollaczek–Khinchine formulas.

Note further, that we shall write $\xi(z)$ for the generating function of the queue length of an $M/G/1$ queue from now on.

## 3.6 Frequently used matrix identities and operations

In this section we list several matrix identities used in this thesis from Chapter 7 onward. We start by displaying several inversion identities in Subsection 3.6.1. We then provide the definitions of the Kronecker product and the vector stacking operator in Subsection 3.6.2, together with a proposition concerning a connection between them.

### 3.6.1 Matrix inversion identities

When working with inverses of sums of matrices two identities can be very useful: Woodbury matrix identity and Sherman–Morrison formula [31, Section 18.2.d.]:

**Theorem 3.6.1** (Woodbury matrix identity). *Let $A, B, C$ and $D$ be matrices of dimensions $n \times n, n \times m, m \times m$ and $m \times n$ respectively. Suppose that $A$ and $C$ are invertible. Then $A + BCD$ is invertible if and only if $C^{-1} + DA^{-1}B$ is invertible. In this case, we have*

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}.$$

Sherman–Morrison formula can be seen as a special case of Woodbury matrix identity:

**Corollary 3.6.2** (Sherman–Morrison formula). *Let $A$ be an invertible $n \times n$ matrix and $u$ and $v$ column vectors of height $n$. Then $A + uv'$ is invertible if and only if $v'A^{-1}u \neq -1$. In this case, we have*

$$(A + uv')^{-1} = A^{-1} - \frac{A^{-1}uv'A^{-1}}{1 + v'A^{-1}u}.$$

*Proof.* This follows directly from Theorem 3.6.1 by setting $B = u, C = 1$ and $D = v'$. □

When studying QBDs one often comes across the geometric progression of matrices. The following properties can then be of use [31, Section 18.2.f.]:

**Theorem 3.6.3** (Geometric progression of matrices). *Let $A$ be a square matrix and $I$ the identity matrix of the same dimensions. For $n \geq 0$ denote $S_n = \sum_{k=0}^{n} A^k$. Then $\lim_{n\to\infty} S_n$ exists if and only if $\lim_{k\to\infty} A^k = 0$. In this case $I - A$ is invertible and we have*

$$S_n = \sum_{k=0}^{n} A^k = (I - A)^{-1}(I - A^{n+1})$$

*and thus*

$$\lim_{n\to\infty} S_n = \sum_{k=0}^{\infty} A^k = (I - A)^{-1}.$$

*Remark* 3.6.4. In Theorem 3.6.3, the condition $\lim_{k\to\infty} A^k = 0$ is equivalent to $sp(A) < 1$, i.e. for every eigenvalue $\lambda$ of $A$, we must have $|\lambda| < 1$.

Theorem 3.6.3 is a generalization of the geometric progression of numbers, where an infinite geometric series converges if and only if for $r$, the common ratio, $|r| < 1$ holds.

### 3.6.2 Matrix operations

We finish this chapter by defining two matrix operations and by presenting a connection between them.

**Definition 3.6.5** (Kronecker product)**.** Let $A$ and $B$ be $k\times\ell$ and $m\times n$ matrices respectively. The Kronecker product of $A$ and $B$, denoted by $A\otimes B$, is a $km\times\ell n$ matrix defined as follows

$$A\otimes B=\begin{bmatrix} a_{11}B & \dots & a_{1\ell}B \\ \vdots & \ddots & \vdots \\ a_{k1}B & \dots & a_{k\ell}B \end{bmatrix},$$

with $a_{ij}$ the $(i,j)$-th entry of $A$.

**Definition 3.6.6** (Vector stacking operator)**.** $vec\langle\cdot\rangle$, the vector stacking operator, is defined as follows: for an $m\times n$ matrix $A$, $vec\langle A\rangle$ is a column vector of height $mn$ consisting of the columns of $A$ stacked one under another (starting from the leftmost column). That is, if $A=[C_1,C_2,\dots,C_n]$, where $C_i$ denotes the $i$-th column of $A$, then

$$vec\langle A\rangle=\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{bmatrix}.$$

The following property, which is due to [65, Section 3.], shows a connection between $vec\langle\cdot\rangle$ and the Kronecker product.

**Proposition 3.6.7** (Roth's column lemma)**.** *Let $A$, $B$ and $C$ be $k\times\ell$, $\ell\times m$ and $m\times n$ matrices respectively. Then*

$$vec\langle ABC\rangle=(C'\otimes A)vec\langle B\rangle.$$

The last two definitions and the above proposition, along with some other properties, can be found for example in [37].

# Chapter 4

# Large-scale systems

## 4.1 Motivation

Simple queueing systems can often be studied directly by using Markov chains. These systems include a variety of systems where the number of servers $N$ is small. In Chapter 3, we have seen several examples of such queueing systems with $N = 1$. However, in this thesis, we are interested in studying large scale systems, i.e. systems where the number of servers $N$ is large. In case of large $N$, exact analysis is in most cases prohibitive.

One challenge we are faced with when trying to analyze an $N$-server queueing system, with $N$ large, is the "state space explosion". We illustrate this phenomenon with an example.

**Example 4.1.1.** Consider a system of $N$ queues, where each queue uses the FIFO discipline, each job has exponential service requirements with mean 1, and where every queue has a finite buffer consisting of $B$ slots (where we assume that the job in service occupies one of these slots). When a job tries to enter a queue that is full, the job is discarded from the system. Suppose jobs arrive to central dispatcher according to a Poisson process with rate $\lambda N$, for some $0 < \lambda < 1$. The dispatcher then distributes arriving jobs to the queues according to JSQ($d$) policy. JSQ($d$) stands for Join Shortest Queue out of $d$ selected queues. The policy is also referred to as power-of-$d$ policy. As the name suggests, the policy works as follows: upon an arrival, the central dispatcher chooses $d$ queues at random and one of those queues with the least amount of jobs gets assigned the arriving job (with ties between queues broken uniformly at random). Note that for $N = 1$, this system can be described in the Kendall notation as an $M/M/1/B$ queue.

Due to the memorylessness property of the exponential distribution (cf. Theorem 2.1.3), we only need to know the number of jobs in each queue to describe the state of the system. For the $N$-server system, we thus get $\Omega = \{0, 1, \dots, B\}^N$ as the state space, meaning that the number of states of the system is $(B + 1)^N$. The number of states thus quickly grows in function of $N$.

To circumvent the state space explosion we employ mean field modelling. We shall also refer to mean field modelling as "mean field method", simply "mean field", "cavity method" and "cavity queue". In this thesis, we employ mean field modelling in two manners, both are explained below. For ease of presentation we shall refer in this chapter to the first manner as "mean field method/model" and as "cavity method/queue" to the second. The first method focuses on studying the fraction of queues in a given state when $N \to \infty$. This method is explained in Section 4.2. The second method, focuses on studying a single queue from the system of infinitely many queues. We explain the cavity method in 4.3. Note, that the system of infinitely many queues, that is the system with $N \to \infty$, is referred to as the "large-scale limit". In Part II, we use the first method, in Part III both methods are used, while Part IV only uses the cavity method. Further, in Subsection 4.3.2, we remark on a connection between the two methods and why we can use the terms "mean field method" and "cavity method" interchangeably. In Section 4.4 we note the importance of performing simulations for finite $N$. There, we also explain how we calculate the relative error and the 95% confidence intervals for the simulations. Finally, in Section 4.5 we apply the cavity method to an example.

## 4.2 Mean field method

Mean field modelling is a popular tool and been used to study a large variety of systems, including systems with job stealing/sharing [25, 55, 72, 77, 79] and load balancing algorithms [2, 58, 80, 82].

The idea of the method is the following. We approximate the system of $N$ queues by a system with infinitely many queues. We then study the latter system, henceforth referred to as the "mean field model". When studying the mean field model, we do not focus on the state of each queue separately. Instead, we group the queues according to their state and focus on the fraction of queues in a state. Using a set of ODEs we can describe the evolution of these fractions as time progresses.

**Example 4.1.1** (continued)**.** Continuing with the example, we now give the mean field model, its state space and its set of ODEs. Since for every queue in the $N$-server system, we only need to know the queue length, we should group the queues according to their queue length. We can then describe the mean field model by the variables $x_i(t)$, where $x_i(t)$ denotes the fraction of queues in state $i$ at time $t$. The state space of this system is:

$$\Omega_\infty = \left\{ (x_i)_{0\le i\le B} \;\middle|\; \sum_{i=0}^{B} x_i = 1, \forall i \in \{0,\dots,B\} : 0 \le x_i \le 1 \right\}.$$

For ease of notation we set $x_{B+1}(t) = 0$, for all $t$.

The fraction of queues at time $t$ with $i$ or more jobs is $\sum_{k=i}^{B} x_k(t)$. The probability that if an arrival occurs at time $t$ a queue with $i$ jobs gets picked by the dispatcher (where $0 \le i < B$) is the same as the dispatcher choosing $d$ queues with $i$ or more jobs of which not all have at least $i+1$ jobs. This probability, which we will denote by $p_i(t)$, equals

$$p_i(t) = \left(\sum_{k=i}^{B} x_k(t)\right)^d - \left(\sum_{k=i+1}^{B} x_k(t)\right)^d .$$

The evolution of the mean field model can be then described using the following set of ODEs. For $0 < i \le B$, we have

$$\frac{dx_i(t)}{dt} = \lambda p_{i-1}(t)x_{i-1}(t) - 1[i < B]\lambda p_i(t)x_i(t) - x_i(t) + x_{i+1}(t), \tag{4.1}$$

where $1[P]$ is one if $P$ is true and is zero otherwise. The first two terms of Equation (4.1) are due to the dispatcher assigning arrivals to queues (of length $i-1$ and $i$ respectively), while the last two terms are due to completions (in queues with $i$ and $i+1$ jobs respectively). Similarly, we have

$$\frac{dx_0(t)}{dt} = -\lambda p_0(t)x_0(t) + x_1(t).$$

We thus have described the set of ODEs of the mean field.

In general, let $\Omega_\infty$ denote the state space of the mean field model. That is, if $i$ is a state in the system with a single queue ($N = 1$) and $x \in \Omega_\infty$ then $x_i$ denotes the fraction of queues in state $i$. The next step is showing that there exists a unique solution $x(t)$ for the set of ODEs, with $x(t) \in \Omega_\infty$ for every $t \geq 0$ and $x(0) = x_0$ for every $x \in \Omega_\infty$.

Without going too much into detail, this follows if we can show that the drift $dx(t)/dt$ is locally Lipschitz continuous and bounded on $\Omega_\infty$ for a suitable norm on $\mathbb{R}^{\mathbb{N}}$.

For ease of presentation, we assume throughout the rest of this section that queues have a finite buffer $B$. By using [4, Theorem 1.9.6] we can then almost immediatly show the existence of a fixed point. Note, that practically, there is very little difference between having a system with an infinite buffer and a huge finite buffer, provided that the system is stable.

Next, let $X^{(N)}(t)$ capture the system of $N$ queues at time $t$, such that $X^{(N)}(t)_i$ is the fraction of queues in state $i$ at time $t$ in the $N$-queue system. Let $\pi^{(N)}$ denote the invariant distribution of the process $X^{(N)}(t)$. Suppose that $X^{(N)}(0) \in \Omega_\infty$. Note, that in this case we call $X^{(N)}(t)$ a "sample path". Then, under suitable conditions [17, Chapter 11], we get

$$\lim_{N\to\infty} \sup_{t\leq T} ||X^{(N)}(t) - x(t)|| = 0. \tag{4.2}$$

(Note, that (4.2) is also easier to prove when $B < \infty$.) Equation (4.2) says that, over finite time scales, the sample paths of the $N$-server system converge (in probability) to the unique solution of the set of ODEs as the number of servers tends to infinity.

If we can then show that the set of ODEs has a fixed point $\pi$, which is a global attractor (meaning $\lim_{t\to\infty} x(t) = \pi$, for every $x \in \Omega_\infty$ with $x(0) = x$), then the invariant distributions $\pi^{(N)}$ converge weakly to the Dirac measure of $\pi$. Thus, if $\pi$ is a global attractor, we can approximate $\pi^{(N)}$ through (the Dirac measure of) $\pi$. In many cases (including the systems from Part II), proving global attraction is the most difficult step.

When proving global attraction, an essential step can be showing that the there exists a partial order relation on the state space $\Omega_\infty$ that is maintained over time.

**Definition 4.2.2** (Partial order)**.** Let $S$ be a set. We call $\preceq$ a partial order relation on the set $S$ if the following three properties hold:

- Reflexivity: for every $x \in S$, we have $x \preceq x$.
- Antisymmetry: for every $x, y \in S$, if $x \preceq y$ and $y \preceq x$, then $x = y$.
- Transitivity: for every $x, y, z \in S$, if $x \preceq y$ and $y \preceq z$, then $x \preceq z$.

The most known example of a partial order probably is the relation $\leq$ on the set of real numbers.

## 4.3 Cavity method

In the last section, we described the mean field method in case of systems that can be described using a set of ODEs as $N \to \infty$. This is true for all the systems in this thesis where this method is used. There exist systems, however, that cannot be described using a set of ODEs as $N \to \infty$ and are instead described by the, so called, differential inclusion [40]. In this case, the cavity method can be a more direct approach.

The cavity method has been long used in a large range of problems and fields. It was used, for example, in statistical physics [54] in 1987. In 2010, it was properly formalized and adapted to the study of large scale queueing systems in the paper [9]. The method has been used for the analysis of different queueing systems and policies, for example: [10,11,34,35].

We now provide the proper mathematical background of the method and afterwards provide an intuitive explanation behind the method. The modularized program from [9] consists of the following steps:

1. proving uniform stability;
2. given the existence and uniqueness of an invariant distribution of the large scale limit system, proving asymptotic independence; and
3. solving two fixed point equations.

We continue by explaining each of these steps in Subsections 4.3.1, 4.3.2 and 4.3.3 respectively.

### 4.3.1 Uniform stability

For $1 \leq i \leq N$, let $q_{i,N}(t)$ denote number of jobs at time $t$ in the $i$-th queue of the $N$-server system. Let $r_{i,N}(t)$ denote the column vector of residual service times in the $i$-th of the $N$ servers at time $t$, that is, the $k$-th entry of $r_{i,N}(t)$ is the remaining amount of service that the $k$-th job in $i$-th queue has to receive at time $t$. If the $i$-th queue has less than $k$ jobs at time $t$, i.e. $q_{i,N}(t) < k$, we let $k$-th entry of $r_{i,N}(t)$ be 0. Denote $Q_N(t) = (q_{1,N}(t), q_{2,N}(t), \dots, q_{N,N}(t))$ and $R_N(t) = (r_{1,N}(t), r_{2,N}(t), \dots, r_{N,N}(t))$. In this thesis we always assume that service requirements of different jobs are independent and identically distributed. In this case $(Q_N(t), R_N(t))$ forms a CTMC. Throughout the rest of the section we assume that this CTMC is irreducible for every $N \geq 1$.

**Definition 4.3.1** (Uniform stability). We say that uniform stability holds if

$$\sup_{N\geq 1} P\left(q_{1,N}(t) > M\right) \to 0 \tag{4.3}$$

and

$$\sup_{N\geq 1} P\left(\sum_{k\geq 1}(r_{1,N}(t))_k > M\right) \to 0, \tag{4.4}$$

as $M \to \infty$, for every $t$.

Uniform stability means that regardless of the number of queues $N$ and the time $t$, the probabilities that there are at least $M$ jobs in the first queue (4.3) and that there are at least $M$ units of work left in the first queue (4.4) approach 0 while $M$ keeps being increased.

Note, that uniform stability always holds if the queues have a finite buffer $B$. Indeed, for a finite buffer $B$ we have $P\left(q_{1,N}(t) > B\right) = 0$. Further, as jobs have finite mean, the average amount of work in total in a finite number of jobs is also finite. As there are at most $B$ jobs in the first queue, (4.4) therefore also holds.

### 4.3.2 Asymptotic independence

For the remaining steps the service discipline is assumed to be local:

**Definition 4.3.2.** A service discipline is said to be "local" if at every queue $i$ the service discipline only depends on $q_{i,N}(t)$ and on $r_{i,N}(t)$.

In this thesis all service disciplines are local.

Next step involves showing that the system of infinitely many queues has an invariant and ergodic measure $\Pi$ (on $\mathbb{N}^\infty$). One further has to show that $\Pi$ is unique.

If the $N$-server system is stable, then the CTMC $(Q_N(t), R_N(t))$ is positive recurrent and therefore has a unique invariant distribution $(\Pi_N, \Gamma_N)$. The next step is then showing that $\Pi_N \to \Pi$ as $N \to \infty$. Once this is done, we have to prove "asymptotic independence":

**Definition 4.3.3.** Let $\Pi^{(k)}$ denote the restriction of $\Pi$ to the first $k$ entries of $\Pi$. We say that a queueing system is asymptotically independent if for every $k = 1, 2, 3, \ldots$ we have

$$\Pi^{(k)} = \bigotimes_{i=1}^{k} \Pi^{(1)} \tag{4.5}$$

as $N \to \infty$, where $\bigotimes_{i=1}^{k} \Pi^{(1)}$ denotes the $k$-fold product space.

Let us reflect on the above definition for a moment. Asymptotic independence implies that in the large scale limit any finite number of queues become independent. Note, that

$$\Pi^{(1)}_\ell = \lim_{N\to\infty} \lim_{t\to\infty} P(q_{1,N}(t) = \ell)$$

is the probability that in the large scale limit the first queue, and therefore any queue, has $\ell$ jobs in equilibrium. This implies that $\Pi^{(1)}_\ell$ is also the fraction of queues with $\ell$ jobs in the large scale limit in equilibrium and $\Pi^{(1)}$ is thus a fixed point of the set of ODEs/the differential inclusion describing the large scale limit. This also explains the link between the two methods. Further, Equation (4.5) says the following: if we wish to calculate f.e. the probability that the first queue has 5 jobs and the second 2 in the large scale limit, we simply have to find $\Pi^{(1)}$ and calculate $\Pi^{(1)}_5 \Pi^{(1)}_2$. Finally, note that letting $N \to \infty$ is usually essential for asymptotic independence to hold.

Some important classes of systems where asymptotic independence has been shown to hold include:

- Systems with JSQ($d$) policy, FCFS service and job size distribution with decreasing hazard rate[1] [10].
- Systems using LL($d$) policy [10], where LL stands for "least loaded". These are the systems where the dispatcher gives an incoming job to the queue with the least remaining work out of $d$ randomly chosen queues.
- Systems using any convex combination of $LL(d, K)$ policies [67]. Note, that the $LL(d, K)$ policy is a generalization of the $LL(d)$ policy. Under $LL(d, K)$ policy, with $K \leq d$, $K$ jobs are assigned by a dispatcher to $K$ least loaded queues among $d$ selected queues.

### 4.3.3 Fixed point equations

As the final step, we actually calculate $\Pi^{(1)}$. This is done through two "fixed point equations".

For many load balancing policies, the rate at which jobs arrive to the first queue (or any other queue) in the large scale limit, depends on the number of jobs in said queue. If every arrival consists of a single job, let $\lambda_\ell$ denote the rate at which jobs arrive to the first queue in the large scale limit in equilibrium, provided that this queue has $\ell \geq 0$ jobs. Let $\Lambda$ denote the vector $(\lambda_\ell)_{\ell \in \mathbb{N}}$. If the system has batch arrivals, let $\lambda_{\ell,k}$ denote the rate at which a batch of $k \geq 1$ jobs arrives to the first queue in the large scale limit in equilibrium, provided that the queue has $\ell \geq 0$ jobs. In this case, let $\Lambda$ be the matrix $(\lambda_{\ell,k})_{\ell,k}$. As $\Pi^{(1)}_\ell$ is the fraction of queues in the large scale limit in equilibrium with $\ell$ jobs, $\Lambda$ depends on $\Pi^{(1)}$ and therefore, for some function $G$, we have

$$\Lambda = G(\Pi^{(1)}). \tag{4.6}$$

Conversely, the invariant measure $\Pi$, and therefore $\Pi^{(1)}$, clearly depends on the (batch) arrival rates $\Lambda$. Thus, there exists another function $F$ such that

$$\Pi^{(1)} = F(\Lambda). \tag{4.7}$$

Combining Equations (4.6) and (4.7) now yields

$$\Pi^{(1)} = F(G(\Pi^{(1)})),$$
$$\Lambda = G(F(\Lambda)).$$

Solving these two "fixed point equations" gives $\Pi^{(1)}$ and $\Lambda$.

Note, that in this thesis we are sometimes not only interested in the distribution of the number of jobs in the cavity queue in equilibrium, but also in the phase of the job in service. In this case let $\Pi^{(1)}_{\ell,k}$ be the probability that in the equilibrium the first queue has $\ell$ jobs and some phase $k$ and let $\Pi^{(1)}$ denote the matrix with $(\ell, k)$-th entry given by $\Pi^{(1)}_{\ell,k}$. Note, that $\Pi^{(1)}_\ell = \sum_k \Pi^{(1)}_{\ell,k} = \Pi^{(1)}_{\ell,k}\mathbf{1}$. Therefore, using the map $G'(\cdot) = G(\cdot\,\mathbf{1})$, Equation (4.6) can be rewritten as

$$\Lambda = G'(\Pi^{(1)}) = G(\Pi^{(1)}\mathbf{1}).$$

[1]A non-negative random variable with cdf $F(t)$ and pdf $f(t)$ is said to have decreasing hazard rate if the hazard function $h(t) = f(t)/(1 - F(t))$ is decreasing in function of $t$.

Proceeding similarly as above, we get two modified fixed point equations:

$$\Pi^{(1)} = F(G'(\Pi^{(1)})),$$
$$\Lambda = G'(F(\Lambda)).$$

By solving these equations, we can also obtain the distribution of the phase of the job in service.

In Parts III and IV, the fixed point equations are solved as follows. In Part III, we first find a way of calculating $\Lambda$. This requires solving several quadratic matrix equations. Based on this $\Lambda$ we can calculate $\Pi^{(1)}$. In Chapter 10, we find an explicit formula for $\Lambda$ and then an explicit formula for the generating function of $\Pi^{(1)}$. In Chapter 11, we use a bisection algorithm to numerically solve the fixed point equations.

## 4.4 Simulation results

Usually, we skip the steps described in Subsections 4.3.1 and 4.3.2 and we simply assume the following conjecture (referred to as the "Ansatz") to be true:

**Conjecture 4.4.1** (Ansatz)**.** *Consider a system of $N$ queues with a load smaller than* 1 *operating under some load balancing policy. Suppose that the system is uniformly stable and that the service discipline is local. Then, in the large scale limit, i.e. as $N \to \infty$, there exists a unique invariant measure. Further, in the large scale limit, the system exhibits asymptotic independence, i.e. any finite number of queues become independent.*

This allows us to immediately work on the analysis of the queue at the cavity and also gives the first reason for simulations: we examine whether simulation results support the Ansatz. Further, regardless of whether or not the Ansatz holds for a system, it is still worthwhile to simulate the system for various values of $N$. This is due to the fact that if the Ansatz holds, the cavity queue provides a correct approximation of the $N$-server system as $N$ grows large, however we do not know anything on the accuracy of the approximation in function of $N$. If the Ansatz does not hold, then the cavity queue/mean field model may still provide good approximate results for the system of $N$ queues.

When simulating a system of $N$ queues we usually measure the mean response time. In this thesis, for every simulated $N$-server system we run the simulation 20 times. For each of these 20 runs we note down the measured mean response time. The average of these measured mean response times (denoted as $\tilde{E}[R_N]$) should be approximately the theoretical mean response time of the $N$-server system. Let us denote the latter by $E[R_N]$. We are interested in the relative error of the mean response times $E[R_N]$ and $E[R_\infty]$, defined as

$$\frac{|E[R_N] - E[R_\infty]|}{E[R_\infty]}.$$

We can approximate this error by

$$\frac{|\tilde{E}[R_N] - E[R_\infty]|}{E[R_\infty]}.$$

In all simulations presented here, this approximation seems to be $O(1/N)$ accurate, similarly to the results presented in [24]. $O(1/N)$ accuracy means the following: if we increase the number of queues $N$ by a factor, then the relative error decreases by that factor. We also calculate the 95% confidence intervals for each set of 20 simulations. Let $\tilde{E}_i[R_N]$ denote the measured mean response time of the system of the $N$-server system in $i$-th simulation. This interval is then of the form $\left[\tilde{E}[R_N] - conf, \tilde{E}[R_N] + conf\right]$, with $conf$ given by

$$conf = C_T \sqrt{\frac{\sum_{i=1}^{n} \left(\tilde{E}[R_N] - \tilde{E}_i[R_N]\right)^2}{n(n-1)}}$$

for $n$ simulations, where $C_T$ is the so called critical $T$ value. For $n = 20$ and 95% confidence intervals, $C_T$ is (approximately) 2.093.

## 4.5 Queue at the cavity of Example 4.1.1

We finish this chapter by describing the queue at the cavity and the fixed point equations for the system in Example 4.1.1. Suppose the Ansatz holds for the system in the example.

Each of the queues, and therefore the cavity queue, of the large scale limit system has Poisson arrivals that depend on the number of jobs in the queue. Continuing with the notation from 4.3.3 we shall denote by $\lambda_\ell$ the rate at which jobs arrive to the cavity queue given that it has $\ell$ jobs. As the job sizes are exponential and the scheduling policy is FCFS, we only need to know the number of jobs in the cavity queue to describe its state. In other words, the state space of the cavity queue is given by $\Omega_{cav} = \{0, 1, \ldots, B\}$. The cavity queue then evolves as a CTMC with the rate matrix

$$Q(\Lambda) = \begin{bmatrix} -\lambda_0 & \lambda_0 & & & & \\ 1 & -1-\lambda_1 & \lambda_1 & & & \\ & 1 & -1-\lambda_2 & \lambda_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -1-\lambda_{B-1} & \lambda_{B-1} \\ & & & & 1 & -1 \end{bmatrix},$$

where we stress the dependence on $\Lambda$ by writing $Q(\Lambda)$. As $\lambda_\ell > 0$ for every $\ell = 0, 1, \ldots, B-1$, this CTMC is irreducible. As the state space is finite, the CTMC is also positive recurrent. Therefore, it has a unique invariant distribution $\Pi^{(1)}$ (cf. Theorem 3.2.2). This distribution can be found as the unique solution to the set of equations $\Pi^{(1)}Q = 0$ and $\Pi^{(1)}\mathbf{1} = 1$. This can be seen as the equivalent of Equation (4.7) as $\Pi^{(1)}$ depends on $Q(\Lambda)$, which in turn depends on $\Lambda$. On the other hand as $\Pi^{(1)}_\ell$ is the fraction of queues with $\ell$ jobs in equilibrium, $\lambda_\ell$ can be determined as follows. $\sum_{k=\ell}^{B} \Pi^{(1)}_k$ is the probability that a queue has at least $\ell$ job in equilibrium. Therefore

$$\left(\sum_{k=\ell}^{B} \Pi^{(1)}_k\right)^d - \left(\sum_{k=\ell+1}^{B} \Pi^{(1)}_k\right)^d$$

is the probability that the dispatcher assigns an arrival to a queue with $\ell$ jobs (the dispatcher chooses $d$ queues, each with $\ell$ or more jobs, but not all with at least $\ell+1$ jobs).

The rate at which jobs arrive to the queues with $\ell$ jobs is then

$$\lambda \left[ \left( \sum_{k=\ell}^{B} \Pi_k^{(1)} \right)^d - \left( \sum_{k=\ell+1}^{B} \Pi_k^{(1)} \right)^d \right].$$

As $\Pi_\ell^{(1)}$ is the fraction of queues with $\ell$ jobs in equilibrium, it follows that in equilibrium

$$\lambda_\ell = \frac{\lambda}{\Pi_\ell^{(1)}} \left[ \left( \sum_{k=\ell}^{B} \Pi_k^{(1)} \right)^d - \left( \sum_{k=\ell+1}^{B} \Pi_k^{(1)} \right)^d \right]$$

is the rate at which jobs get assigned to each queue (and therefore the cavity queue) with $\ell$ jobs.

# Part II

# Monotone Systems

# Chapter 5 Supermarket Model in Processor Sharing Systems

*This Chapter is based on the paper* [43]*, titled "On the Asymptotic Insensitivity of the Supermarket Model in Processor Sharing Systems". This was the first paper I worked on that got accepted to a conference, namely to ACM SIGMETRICS '21. The conference was planned to take place in Beijing, China. However due to the COVID-19 pandemic, it was changed into a fully virtual event. My presentation of the paper can be freely viewed at* `https://www.youtube.com/watch?v=uG41LRnvpyU`. *The paper was subsequently published in Proceedings of the ACM on Measurement and Analysis of Computing Systems.*

## 5.1 Introduction

The supermarket model refers to a popular load balancing model consisting of $N$ homogeneous servers and Poisson arrivals with rate $\lambda N$, where for each incoming job $d$ servers are selected at random and the job joins the queue with the fewest jobs. In the seminal papers [58,82] it was shown that when job sizes are exponential with mean 1 the probability of having $k$ or more jobs in a server converges to $\lambda^{(d^k-1)/(d-1)}$ as the number of servers $N$ tends to infinity. Hence the queue length decays doubly exponential as soon as $d > 1$, demonstrating the power of having $d$ choices. While the authors considered First-Come-First-Served (FCFS) servers, the result also applies to processor sharing (PS) servers as both systems are equivalent when the job sizes are exponential.

A modularized program to study the supermarket model with non-exponential job sizes was proposed in [9] for both FCFS and PS servers. The program relies on an ansatz that asserts that, for a randomized load balancing scheme *in equilibrium*, any fixed number of queues become independent of one another as the number of servers tends to infinity. Using this ansatz hypothesis the limiting steady-state queue length distribution and other performance measures of interest can be computed by studying the queue at the cavity (e.g., [33,34]).

For the supermarket model with processor sharing the steady-state queue length distribution of the queue at the cavity is that of an $M/G/1/PS$ queue with arrival rates that depend on the queue length. As the queue length distribution of such a queue is known to be insensitive to the job size distribution (meaning only the mean job size matters) [12], this naturally leads to the conjecture that the queue length distribution becomes insensitive to the job size distribution as the number of servers tends to infinity, which we refer to as *asymptotic insensitivity*. In [9] the authors noted that for the supermarket model with a finite fixed number of servers $N$ with PS service, the queue length distribution is not insensitive, meaning only as $N$ tends to infinity the sensitivity vanishes. In addition, the authors showed that the cavity map that is used to compute the limiting steady-state queue length distribution in case of PS servers has a unique fixed point that corresponds to the same distribution as in the exponential case, yielding further support for the conjecture of asymptotic insensitivity. While the ansatz was proven in [10, 66] for various load balancing policies and proving the ansatz for the supermarket model with PS servers would settle the conjecture, it is still an open problem.

In [81] the authors also considered the supermarket model with PS service and general service times. The authors used measure-valued processes and martingale techniques to show that the limit of the empirical distributions satisfies a set of partial differential equations (PDEs). These PDEs correspond to the transient behavior of an $M/G/1/PS$ queue with a queue length and *time* dependent arrival rate. The authors further showed that this set of PDEs has a unique fixed point, which is in agreement with the result in [9]. However, as stated after listing their main contributions, in order to prove asymptotic insensitivity of the limit of the stationary measures, global attraction of the fixed point must be proven. Instead of providing such a proof, the authors present simulation results supporting asymptotic insensitivity.

A major challenge in proving asymptotic insensitivity for the the supermarket model with PS servers lies in overcoming the apparent lack of monotonicity in such systems. In this chapter we show that monotonicity arguments can still be leveraged if we restrict ourselves to the class of hyperexponential distributions of order 2. More specifically, we prove that the limiting steady-state queue length distribution of the supermarket model with PS servers and order 2 hyperexponential job sizes is the same as the limiting distribution for exponential job sizes. In other words we prove *asymptotic insensitivity of the limiting steady-state queue length distribution within the class of hyperexponential distributions of order* 2. The class of hyperexponential distributions of order 2 is often used in performance modeling as it can be regarded as a mixture of long and short jobs and can be used to match any squared coefficient of variation (SCV) larger than one.

It is worth noting that convergence of the steady state measures has been established in some specific cases even for systems that are not monotone. For instance, in [51] it is shown that various load balancing policies for FCFS servers achieve vanishing delays in the heavy traffic regime when the load equals $1 - N^{-\alpha}$, for $0 < \alpha < 0.5$, when the job sizes have general order-2 Coxian distributions. The supermarket model is one of the policies considered in [51], but in this case $d$ scales as $O(N^{\alpha}\log(N))$, whereas in this chapter $d$ is a constant independent of $N$, which implies that delays do not vanish, and servers use PS instead of FCFS.

The approach taken in this chapter is as follows. For job sizes with an order 2 hyperexponential distribution, the sample paths of the stochastic process of the supermarket model consisting of $N$ servers converge to the solution of a set of ordinary differential equations (ODEs) that is shown to have a unique fixed point. The main step to establish asymptotic insensitivity then exists in showing that the fixed point of the set of ODEs is a global attractor. We show that the set of ODEs is monotone by using a Coxian representation of the hyperexponential distribution and defining a suitable state space and partial order, from which global attraction follows without much effort. The work in this chapter is in this regard somewhat similar to [76], where a Coxian representation and a suitable state space and partial order was also used to prove global attraction of some load balancing systems. However the systems considered in [76] are restricted to FCFS servers, which simplifies the set of ODEs and especially the proof that the set of ODEs is monotone. Moreover, the result in [76] applies to hyperexponential distributions of any order, while for PS servers the approach appears to be limited to order 2 distributions (see Section 5.8). Similar to [76], we also present our global attraction result in such a manner that it can be used to prove global attraction for models with PS servers other than the supermarket model and we demonstrate this for the traditional push strategy (see Section 5.9). To avoid some technical issues, we assume that the buffer size at each server is finite.

The main contributions of the chapter are as follows:

- We prove asymptotic insensitivity of the limiting steady-state queue length distribution within the class of hyperexponential distributions of order 2 for the supermarket model with PS servers.
- We present our global attraction result used to prove asymptotic insensitivity in such a manner that it may also be leveraged for other load balancing policies with PS servers and demonstrate this using the traditional push algorithm.

The chapter is structured as follows. In Section 5.2 we discuss the model under consideration and present the Coxian representation. In Section 5.3 we introduce the set of ODEs describing the mean field limit, while the state space and partial order are presented in Section 5.4. Our global attraction result is stated and proven in Section 5.5. In Section 5.7 we show that the assumptions of our global attraction result are satisfied for the supermarket model and prove that the set of ODEs has a unique fixed point that corresponds to the same queue length distribution as in the exponential case. The asymptotic insensitivity result is presented in Section 5.8, while in Section 5.9 we demonstrate that our results are not limited to the supermarket model. Finally conclusions are drawn in Section 5.10.

## 5.2 Model Description

We focus on the supermarket model, also known as the JSQ($d$) load balancing policy, with processor sharing servers. In this model, arrivals occur according to a Poisson process with rate $\lambda N$, we have a set of $N$ servers that use processor sharing and each incoming job is immediately assigned to a server by selecting a server with the least number of jobs among a set of $d$ random servers (with ties being broken at random). We assume the processing speed of a server equals 1 and when $n$ jobs are present in a server, each job receives an equal share $1/n$ of the processing speed of the server. We further assume each server has a finite buffer of size $B$, meaning an incoming job is lost if $d$ servers with a full buffer are selected. The finiteness of the buffer allows us to avoid certain technical issues and is not an uncommon assumption in mean field modeling [22,25]. Further in a real system all buffers are finite and there is hardly any difference between having a huge finite buffer or an infinite buffer (as long as the system is stable).

We consider order 2 hyperexponential job sizes with a mean equal to 1, meaning $\lambda < 1$ suffices for the system to be stable. More specifically, with some probability $\tilde{p}$ jobs have an exponential size with mean $1/\mu_1$ and with probability $1-\tilde{p}$ jobs have an exponential size with mean $1/\mu_2$ with $\mu_1 > \mu_2$, such that $\tilde{p}/\mu_1 + (1-\tilde{p})/\mu_2 = 1$. While the standard phase-type representation $(\tilde{\alpha}, \tilde{S})$ of a hyperexponential distribution is given by $\tilde{\alpha} = (\tilde{p}, 1-\tilde{p})$ and

$$\tilde{S} = \begin{bmatrix} -\mu_1 & 0 \\ 0 & -\mu_2 \end{bmatrix},$$

hyperexponential distributions also have a Coxian representation, see [76, Proposition 1], which in case of 2 phases corresponds to a representation $(\alpha, S)$ with $\alpha = (1, 0)$ and

$$S = \begin{bmatrix} -\mu_1 & p_1\mu_1 \\ 0 & -\mu_2 \end{bmatrix},$$

where $p_1 = (1-\tilde{p})(1-\mu_2/\mu_1)$ and $(1-p_1)\mu_1 > \mu_2$. In fact one can readily check (by computing the Laplace Stieljes transform) that any distribution with an order 2 Coxian representation and $(1-p_1)\mu_1 > \mu_2$, is a hyperexponential distribution with $\tilde{p} = 1 - p_1\mu_1/(\mu_1 - \mu_2)$ (cf. Proposition 2.5.4). For further use we denote $\nu_1 = \mu_1(1-p_1)$ and $\nu_2 = \mu_2$. To establish monotonicity we formulate the mean field limit using the Coxian representation $(\alpha, S)$.

The main challenge to prove global attraction using monotonicity arguments, is to pick a set of variables that capture the system state such that the set of ODEs that describes the dynamics of the mean field model in terms of these variables is monotone with respect to some partial order on the associated state space. When the job sizes are exponential, [82] showed that the set of ODEs given by

$$\frac{d}{dt}h_j(t) = \lambda(h_{j-1}^d(t) - h_j^d(t)) - (h_j(t) - h_{j+1}(t)),$$

where the variables $h_j(t)$ represent the fraction of the servers with $j$ or more jobs at time $t$ is monotone with respect to the pointwise partial order.

When the servers are FCFS servers as in [76] and the job sizes are hyperexponential of order 2, then it suffices to use a set of variables that represent the fraction of servers with $j$ or more jobs (denoted as $h_{j,1}$ in [76]) and a set of variables for the fraction of servers with $j$ or more jobs for which the server is in phase 2 (denoted as $h_{j,2}$ in [76]). In this case a stronger partial order $\leq_C$ is required to get a monotone system. This order is such that $h \leq_C \tilde{h}$ if

$$h_{j_1,1} - h_{j_1,2} + h_{j_2,2} \leq \tilde{h}_{j_1,1} - \tilde{h}_{j_1,2} + \tilde{h}_{j_2,2},$$

for all $j_1 \geq j_2 \geq 1$. Note that $h_{j_1,1} - h_{j_1,2} + h_{j_2,2}$ is the fraction of servers with $j_1$ or more jobs in service phase 1 (given by $h_{j_1,1} - h_{j_1,2}$) plus the fraction of servers with $j_2$ or more jobs in service phase 2 (given by $h_{j_2,2}$).

For PS servers the system state is clearly more complex as we need to keep track of the number of jobs in service phase 1 and phase 2. Therefore a more complex set of variables denoted as $h_{i,j}$ is required, where $h_{i,j}$ represents the fraction of servers with at least $i+j$ jobs of which at least $j$ jobs are in phase 2, for $i, j \geq 0$. As a result, the partial order in case of PS servers is more involved as is the set of ODEs that describe the evolution of the mean field limit. This implies that proving monotonicity requires different arguments and is more challenging in case of PS servers. Indeed, the monotonicity proof in case of

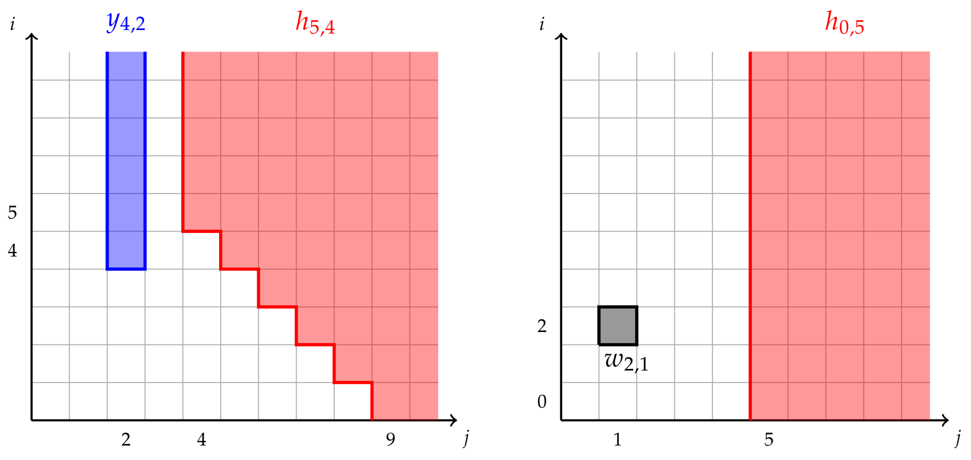


Figure 5.1: Illustration of the variables $h_{i,j}$, $y_{i,j}(h) = h_{i,j} - h_{i-1,j+1}$ and $w_{i,j}(h) = y_{i,j}(h) - y_{i+1,j}(h)$.

FCFS servers given in [76, Proposition 6] is fairly straightforward compared to the proof of Proposition 5.5.2 in this chapter.

## 5.3 The Set of ODEs

In this section we introduce the set of ODEs that describes the mean field limit when the servers use processor sharing, have a finite buffer of size $B$ and the order 2 hyper-exponential job sizes are represented in Coxian form. Let $h_{i,j}(t)$, for $i, j \geq 0$, denote the fraction of servers with at least $i + j$ jobs of which at least $j$ jobs are in service phase 2 at time $t \geq 0$. We set $h_{0,0}(t) = 1$ and $h_{i,j}(t) = 0$, if $i + j > B$. We define $1[P]$ to be 1 if the property $P$ holds and 0 otherwise. To ease the presentation we define

- $y_{i,j}(h(t)) = h_{i,j}(t) - h_{i-1,j+1}(t)$, for $i > 0, j \geq 0$,
- $y_{0,j}(h(t)) = h_{0,j}(t) - h_{0,j+1}(t)$, for $j \geq 0$,
- $w_{i,j}(h(t)) = y_{i,j}(h(t)) - y_{i+1,j}(h(t))$, for $i, j \geq 0$,

Note that $y_{i,j}(h(t))$ represents the fraction of the queues with at least $i + j$ jobs of which exactly $j$ jobs are in service phase 2 at time $t \geq 0$, $w_{i,j}(h(t))$ is the fraction of the queues with exactly $i$ jobs in service phase 1 and exactly $j$ jobs in service phase 2 at time $t \geq 0$. An illustration of these variables can be seen in Figure 5.1. While the notation may appear a bit heavy, its usefulness becomes apparent in the next section.

In order to understand the set of ODEs that is presented next, we make the following observations:

- Phase changes increase $h_{i,j}(t)$ if and only if such a phase change happens in a server with exactly $j - 1$ jobs in phase 2 and exactly $k + 1$ jobs in phase 1, for $k \geq i$. This happens at rate $p_1\mu_1$ times the fraction $(k+1)/(k+j)$ of jobs that are in phase 1. This explains the appearance of the first sum in (5.1).

- Service completions decrease $h_{i,j}(t)$ if a service completion happens in a server with exactly $i+j$ jobs. These service completions occur at rate $(\nu_1(i-k)+\nu_2(j+k))/(i+j)$ if $i-k$ of the $i+j$ jobs are in phase 1, which explains the second sum in (5.1).
- Service completions also decrease $h_{i,j}(t)$ if a service completion of a job in phase 2 occurs in a server with $k+j > i+j$ jobs, of which exactly $j$ jobs are in phase 2. These service completions occur at rate $\nu_2 j/(j+k)$ when there are $j+k$ jobs. This yields the third sum in (5.1).

Let $f_{i,j}(h(t))$ capture the changes due to other events, such as arrivals (specified later on), then the system of ODEs is given by:

$$\frac{d}{dt}h_{i,j}(t) = f_{i,j}(h(t)) + 1[j \geq 1]p_1\mu_1 \sum_{k=i}^{\infty} w_{k+1,j-1}(h(t))\frac{k+1}{k+j}$$
$$- \sum_{k=0}^{i} w_{i-k,j+k}(h(t))\frac{\nu_1(i-k)+\nu_2(j+k)}{i+j} - \nu_2 \sum_{k=i+1}^{\infty} w_{k,j}(h(t))\frac{j}{k+j} \qquad (5.1)$$

for all $i, j \geq 0$ with $(i,j) \neq (0,0)$ and $i+j \leq B$. Note that $w_{i,j}(h(t)) = 0$ whenever $i+j > B$.

*Remark* 5.3.1. In this chapter we show that this set of ODEs has a unique fixed point that is a global attractor using monotonicity arguments. The monotonicity is proven by separately showing that $f_{i,j}(h(t))$ is monotone and that all the remaining terms are monotone. This implies that our result can also be used to prove global attraction for other systems of ODEs as long as they have the above form and $f_{i,j}(h(t))$ is monotone, as demonstrated in Section 5.9.

## 5.4 State space and partial order

We define the state space $\Omega_B$ of the mean field model in terms of the variables $h_{i,j}$ as follows

$$\Omega_B = \Big\{(h_{i,j})_{i,j\geq 0,\,(i,j)\neq(0,0)} \Big| \, 0 \leq h_{i,j} \leq 1, h_{i,j} = 0 \text{ for } i+j > B, h_{i,j} \geq h_{i+1,j},$$
$$h_{i,j} \geq 1[i \geq 1]h_{i-1,j+1}, (h_{i+1,j} - h_{i,j+1}) - (h_{i+2,j} - h_{i+1,j+1}) \geq 0\Big\}.$$

The last condition states that $w_{i+1,j} \geq 0$, where $w_{i,j}$ and $y_{i,j}$ is defined analogue to $w_{i,j}(h(t))$ and $y_{i,j}(h(t))$, respectively. Note that from the last two conditions we get

$$h_{i,j} \geq h_{i,j+1}.$$

We now define the variables $g^{j}_{i_1,\ldots,i_s}(h)$ illustrated in Figure 5.1(right) that will be used to define the partial order.

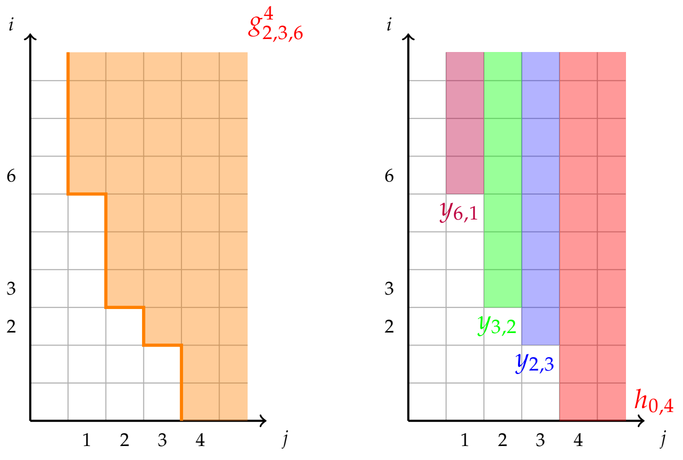


Figure 5.2: Illustration of the variables $g^j_{i_1,\ldots,i_s}(h)$ from Definition 5.4.1.

**Definition 5.4.1.** For $h \in \Omega_B$, $j \geq 1$, $0 \leq s \leq j$ and $0 = i_0 < i_1 < i_2 < \ldots < i_s$, we set

$$g^j_{i_1,\ldots,i_s}(h) = h_{0,j} + \sum_{k=1}^{s} y_{i_k,j-k}(h).$$

As an example, we show $g^4_{2,3,6}(h)$ in Figure 5.2.

**Definition 5.4.2** (Partial order $\leq_C$ on $\Omega_B$)**.** Let $h, \tilde{h} \in \Omega_B$. We state that $h \leq_C \tilde{h}$ if

$$g^j_{i_1,\ldots,i_s}(h) \leq g^j_{i_1,\ldots,i_s}(\tilde{h}) \tag{5.2}$$

for all $j \geq 1$, $0 \leq s \leq j$ and $0 = i_0 < i_1 < i_2 < \ldots < i_s$.

*Remark* 5.4.3. By noting that $g^{i+j}_{1,2,\ldots,i}(h) = h_{i,j}$, (5.2) implies that $h_{i,j} \leq \tilde{h}_{i,j}$ for all $i, j$.

*Remark* 5.4.4. To see why the pointwise partial order does not suffice, consider $h, \tilde{h} \in \Omega_B$ with $w_{0,1}(h) = w_{2,0}(h) = 1/2$ and $w_{1,1}(\tilde{h}) = w_{1,0}(\tilde{h}) = 1/2$. In other words, in state $h$ half of the servers contain a single job in phase 2 and the other servers contain 2 jobs both in phase 1, while in state $\tilde{h}$ half of the servers contain a single job in phase 1 and the remaining servers contain a phase 1 and phase 2 job. It is easy to check that $h_{i,j} \leq \tilde{h}_{i,j}$ for all $i, j \geq 0$ (as $h_{1,0} = \tilde{h}_{1,0} = 1$, $h_{2,0} = \tilde{h}_{2,0} = 1/2$, $h_{0,1} = \tilde{h}_{0,1} = 1/2$, $h_{1,1} = 0$, $\tilde{h}_{1,1} = 1/2$ and $h_{i,j} = \tilde{h}_{i,j} = 0$ for all other $i, j$). Hence, $h$ is smaller than $\tilde{h}$ in the pointwise order. However, idle servers are created at rate $\nu_2/2$ in state $h$ and at rate $\nu_1/2$ in state $\tilde{h}$. As $\nu_1 > \nu_2$, this implies that $\tilde{h}_{1,0}$ decreases faster than $h_{1,0}$, meaning the system is not monotone with respect to the pointwise partial order. Note that $h \not\leq_C \tilde{h}$ as $g^1_2(h) = 1$ and $g^1_2(\tilde{h}) = 1/2$, so the fact that idle servers are created at a higher rate from state $\tilde{h}$ than state $h$ does not violate monotonicity with respect to the order $\leq_C$.

*Remark* 5.4.5. Consider two sets $\mathcal{A}$ and $\tilde{\mathcal{A}}$ of $N$ servers and let $h$ and $\tilde{h}$ be their corresponding states in $\Omega_B$. The intuition behind the order $\leq_C$ is that it should be such that $h \leq_C \tilde{h}$ implies that there exists a mapping $m : \mathcal{A} \to \tilde{\mathcal{A}}$ such that both the total number

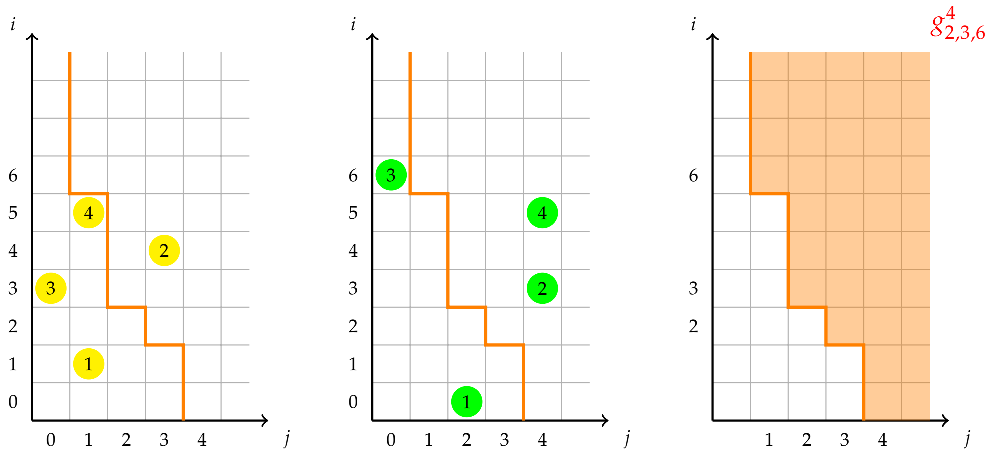


Figure 5.3: Illustration of the intuition from Remark 5.4.5, with $N = 4$. The yellow system is dominated by the green one.

of jobs as well as the number of jobs in phase two for any server $a \in \mathcal{A}$ is dominated by the corresponding quantities of server $m(a) \in \tilde{\mathcal{A}}$. Remark 5.4.4 shows that this is not the case for the pointwise order.

We illustrate the intuition in Figure 5.3. The Figure shows two systems with $N = 4$, where each of the servers is depicted by a numbered, coloured ball. The yellow system is clearly dominated by the green one ($m$ can be chosen to map the $k$-th server from the yellow system to the $k$-th server from the green one). In terms of the variables $g^j_{i_1,\dots,i_s}(h)$, the dominance can be seen as follows: for every $j \geq 1, 0 \leq s \leq j$ and $0 = i_0 < i_1 < i_2 < \dots < i_s$, if we draw the border of of $g^j_{i_1,\dots,i_s}(h)$ and compare the number of coloured balls to the right of the border, then the number of green balls should be at least as high as the number of yellow balls. In Figure 5.3 this is shown for $g^4_{2,3,6}(h)$.

For further use we remark that for $j \geq s \geq 1$:

$$g^j_{i_1,\dots,i_s}(h) = g^j_{i_1,\dots,i_{s-1}}(h) + y_{i_s,j-s}(h) \tag{5.3}$$

and

$$w_{i_s,j-s}(h) = g^j_{i_1,\dots,i_s}(h) - g^j_{i_1,\dots,i_{s-1},i_s+1}(h), \tag{5.4}$$

$$w_{0,j}(h) = g^j(h) - g^{j+1}_1(h). \tag{5.5}$$

To simplify the notation we set $g^j_{i_1,\dots,i_j,k}(h) = g^j_{i_1,\dots,i_j}(h)$ if $k \geq 1$, i.e., additional indices after position $j$ have no impact. Also note that for $i_s > B - (j - s)$ we have

$$g^j_{i_1,\dots,i_{s-1},i_s}(h) = g^j_{i_1,\dots,i_{s-1}}(h), \tag{5.6}$$

as $y_{i_s,j-s}(h) = 0$ for $i_s + j - s > B$. The next two Lemmas are used further on to prove monotonicity of the set of ODEs in (5.1) with respect to the order $\leq_C$.

**Lemma 5.4.6.** *Let $j, s, i_1, \dots, i_s$ be as in Definition 5.4.2. Let $c_1, \dots, c_s \in \mathbb{R}$ and $c_0 = 0$. We then have*

$$\sum_{k=1}^{s} w_{i_k,j-k}(h)c_k = g^j_{i_1,\dots,i_s}(h)c_s - \sum_{k=0}^{s-1} g^j_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1}(h)(c_{k+1} - c_k). \tag{5.7}$$

*Proof.* First, repeatedly using (5.3) and (5.4) gives

$$\sum_{k=1}^{s} w_{i_k,j-k}(h) = g^j_{i_1,\dots,i_s}(h) - g^j_{i_1+1,\dots,i_s+1}(h),$$

which yields that the left hand side of (5.7) can be rewritten as:

$$g^j_{i_1,\dots,i_s}(h)c_1 - g^j_{i_1+1,\dots,i_s+1}(h)c_1 + \sum_{k=2}^{s} w_{i_k,j-k}(h)(c_k - c_1). \tag{5.8}$$

Applying (5.3) and (5.4) implies that (5.8) is equivalent to

$$\begin{aligned}
&g^j_{i_1,\dots,i_s}(h)c_1 - g^j_{i_1+1,\dots,i_s+1}(h)c_1 - \sum_{k=2}^{s}(-g^j_{i_1,\dots,i_k}(h) + g^j_{i_1,\dots,i_{k-1},i_k+1}(h))(c_k - c_1)\\
&= g^j_{i_1,\dots,i_s}(h)c_1 - g^j_{i_1+1,\dots,i_s+1}(h)c_1\\
&- \sum_{k=2}^{s}(-g^j_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1}(h) + g^j_{i_1,\dots,i_{k-1},i_k+1,\dots,i_s+1}(h))(c_k - c_1).
\end{aligned} \tag{5.9}$$

Rearranging the terms in (5.9) (and noting that the sum can start in $k = 1$), we conclude that the left hand side of (5.7) can be written as

$$\begin{aligned}
&g^j_{i_1,\dots,i_s}(h)c_1 - g^j_{i_1+1,\dots,i_s+1}(h)c_1 + g^j_{i_1,\dots,i_s}(h)(c_s - c_1) - 1[s \geq 2]g^j_{i_1,i_2+1,\dots,i_s+1}(h)(c_2 - c_1)\\
&- \sum_{k=2}^{s-1} g^j_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1}(h)(c_{k+1} - c_k) = g^j_{i_1,\dots,i_s}(h)c_s - \sum_{k=0}^{s-1} g^j_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1}(h)(c_{k+1} - c_k).
\end{aligned} \tag{5.10}$$

This finishes the proof. □

**Lemma 5.4.7.** *Let $j, s, i_1, \dots, i_s$ be as in Definition 5.4.2. If $s > \tilde{s}$, then*

$$\frac{i_s}{i_s + j - s} > \frac{i_{\tilde{s}}}{i_{\tilde{s}} + j - \tilde{s}}. \tag{5.11}$$

*Proof.* We have that

$$\frac{i_{k+1}}{i_{k+1} + j - (k+1)} > \frac{i_k}{i_k + j - k} \tag{5.12}$$

is equivalent to

$$(i_{k+1} - i_k)(j - k) > -i_k, \tag{5.13}$$

which is true, as the right hand side is negative. The statement then follows by repeatedly using (5.12) $s - \tilde{s}$ times. □

## 5.5 Global Attraction

In this section, we define empty summations to be 0. We now state three assumptions for $f_{i,j}(h)$ that suffice for the set of ODEs in (5.1) to have a global attractor in $\Omega_B$. We prove that these assumptions hold for the supermarket model in Section 5.7.

**Assumption 5.1.** *The functions $f_{i,j}(h) : \Omega_B \to \mathbb{R}$ are such that for any $h_0 \in \Omega_B$, the set of ODEs given by* (5.1) *has a unique solution $h(t) : [0, \infty) \to \mathbb{R}$ with $h(0) = h_0$.*

**Definition 5.5.1.** For $h \in \Omega_B$, $j \geq 1$, $0 \leq s \leq j$ and $0 = i_0 < i_1 < i_2 < \ldots < i_s$, we set

$$F^j_{i_1,\ldots,i_s}(h) = f_{0,j}(h) + \sum_{k=1}^{s}(f_{i_k,j-k}(h) - f_{i_k-1,j-k+1}(h)).$$

**Assumption 5.2.** *The functions $f_{i,j}(h) : \Omega_B \to \mathbb{R}$ are such that for all $j \geq 1$ and all $i_k$'s as in (5.2) we have*

$$F^j_{i_1,\ldots,i_s}(h) \leq F^j_{i_1,\ldots,i_s}(\tilde{h}) \tag{5.14}$$

*if $h \leq_C \tilde{h}$ and $g^j_{i_1,\ldots,i_s}(h) = g^j_{i_1,\ldots,i_s}(\tilde{h})$.*

**Assumption 5.3.** *The functions $f_{i,j}(h) : \Omega_B \to \mathbb{R}$ are such that the set of ODEs given by (5.1) has a unique fixed point $\pi$ in $\Omega_B$.*

Under the first two assumptions we prove that the partial order $\leq_C$ is preserved over time.

**Proposition 5.5.2.** *Assume that Assumptions 5.1-5.2 hold and let $h_0, \tilde{h}_0 \in \Omega_B$. Let $h(t)$ and $\tilde{h}(t)$ be the unique solution of* (5.1) *with $h(0) = h_0$ and $\tilde{h}(0) = \tilde{h}_0$, respectively. If $\nu_1 = \mu_1(1 - p_1) > \mu_2 = \nu_2$ and $h_0 \leq_C \tilde{h}_0$ then $h(t) \leq_C \tilde{h}(t)$ for any $t \geq 0$.*

*Proof.* The proof is long and technical, therefore we present it in Section 5.6. The main idea is to prove that $\frac{d}{dt} g^j_{i_1,\ldots,i_s}(h(t))$ is non-decreasing in $g^{j'}_{i'_1,\ldots,i'_{s'}}(h(t))$ for all sets of indices $(j', i'_1, \ldots, i'_{s'})$, as in (5.2), different from $(j, i_1, \ldots, i_s)$. □

**Theorem 5.5.3** (Global attraction)**.** *Assume Assumptions 5.1-5.3 hold and $\nu_1 > \nu_2$, then $\pi$ is a global attractor of the set of ODEs given by* (5.1), *meaning $h(t)$ converges to $\pi$ as $t$ tends to infinity for any $h_0 \in \Omega_B$ with $h(0) = h_0$.*

*Proof.* The proof is similar to [40, Theorem 4]. Define $(h^{(u)})_{i,j} = 1$ and $(h^{(\ell)})_{i,j} = 0$ for $0 < i + j \leq B$, then $h^{(\ell)} \leq_C h \leq_C h^{(u)}$ for all $h \in \Omega_B$. By Proposition 5.5.2 it suffices to show that $h(t)$ converges to $\pi$ when $h(0) = h^{(\ell)}$ and when $h(0) = h^{(u)}$ as the trajectories of other initial states $h_0$ must remain between these two trajectories.

We prove the convergence when $h(0) = h^{(\ell)}$, the proof for $h^{(u)}$ is analogous. First note that $h(0) = h^{(\ell)} \leq_C h(t - s)$ holds for $0 < s < t$, as $h^{(\ell)} \leq_C h$ for all $h \in \Omega_B$. Hence, by Proposition 5.5.2 $h(s) \leq_C h(t)$ for $0 < s < t$, as $h(t)$ is the state at time $s$ if we start in state $h(t-s)$. The theory of monotone dynamical systems (see [38, Theorem 1.4 ]) now implies that $h(t)$ converges to a fixed point as $\Omega_B$ is a compact set. Due to Assumption 5.3, $h(t)$ must converge to $\pi$ when $h(0) = h^{(\ell)}$. □

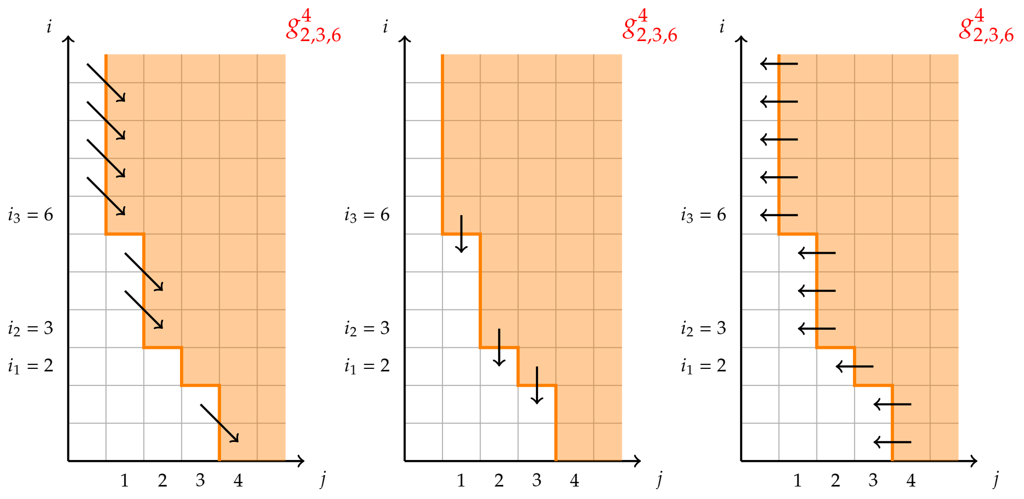


Figure 5.4: Illustration of change to $g^4_{2,3,6}(h(t))$ due to phase changes (left), service completions in phase 1 (middle) and service completions in phase 2 (right).

## 5.6 Proof of Proposition 5.5.2

We show that (5.2) is retained over time. Suppose that $g^j_{i_1,\ldots,i_s}(h(t)) = g^j_{i_1,\ldots,i_s}(\tilde{h}(t))$ for some $j \geq 1$ and $0 < i_1 < \ldots < i_s$, with $s \leq j$. We need to show that

$$\frac{d}{dt} g^j_{i_1,\ldots,i_s}(h(t)) \leq \frac{d}{dt} g^j_{i_1,\ldots,i_s}(\tilde{h}(t)), \tag{5.15}$$

as otherwise (5.2) is violated at some time greater than $t$. Hence, it suffices to show the claim that $\frac{d}{dt} g^j_{i_1,\ldots,i_s}(h(t))$ is non-decreasing in $g^{j'}_{i'_1,\ldots,i'_{s'}}(h(t))$ for all sets of indices $(j', i'_1, \ldots, i'_{s'})$, as in (5.2), different from $(j, i_1, \ldots, i_s)$. Due to Assumption 5.2, this holds for the terms associated to $f_{i,j}(h(t))$ and it suffices to show that this claim also holds for the remaining terms corresponding to phase changes and service completions.

The rest of the proof is structured as follows. In Subsection 5.6.1 we prove that the claim holds in the terms associated with phase changes, while Subsections 5.6.2-5.6.4 show that the claim holds in the terms associated with completions: Subsection 5.6.2 deals with completions from phase 1, Subsection 5.6.3 with completions from phase 2 in case where $i_{k+1} - i_k \geq 2$ for every $k$ and Subsection 5.6.4 with the general case of completions from phase 2. To ease presentation, we suppress the dependence on $h(t)$ throughout the rest of the proof.

### 5.6.1 Drift due to the phase changes

The drift of $g^j_{i_1,\ldots,i_s}$ due to phase changes can be written as

$$\begin{aligned}1[j \geq 1]p_1\mu_1\Bigg(&\sum_{v=1}^{s}\sum_{k=i_{v-1}+1}^{i_v-1} w_{k,j-v}\frac{k}{k+j-v}\\ &+1[j\geq s+1]\sum_{k=i_s+1}^{\infty} w_{k,j-s-1}\frac{k}{k+j-(s+1)}\Bigg),\end{aligned} \tag{5.16}$$

as illustrated in Figure 5.4(left).

For ease of notation set $i_{s+1}=\infty$. Using (5.4), we have that (5.16) is equal to

$$\begin{aligned}1[j \geq 1]p_1\mu_1\Bigg(&\sum_{v=1}^{s}\sum_{k=i_{v-1}+1}^{i_v-1} (g^j_{i_1,\ldots,i_{v-1},k}-g^j_{i_1,\ldots,i_{v-1},k+1})\frac{k}{k+j-v}\\ &+1[j\geq s+1]\sum_{k=i_s+1}^{\infty} (g^j_{i_1,\ldots,i_s,k}-g^j_{i_1,\ldots,i_s,k+1})\frac{k}{k+j-(s+1)}\Bigg).\end{aligned}$$

Due to (5.3), this is the same as

$$\begin{aligned}1[j \geq 1]p_1\mu_1\Bigg(&\sum_{v=1}^{s}\sum_{k=i_{v-1}+1}^{i_v-1} (g^j_{i_1,\ldots,i_{v-1},k,i_{v+1},\ldots,i_s}-g^j_{i_1,\ldots,i_{v-1},k+1,i_{v+1},\ldots,i_s})\frac{k}{k+j-v}\\ &+1[j\geq s+1]\sum_{k=i_s+1}^{\infty} (g^j_{i_1,\ldots,i_s,k}-g^j_{i_1,\ldots,i_s,k+1})\frac{k}{k+j-(s+1)}\Bigg).\end{aligned} \tag{5.17}$$

Given $i_1,\ldots,i_s$ (with $i_0=0$ and $i_{s+1}=\infty$), we now define $a_i$ as follows:

$$a_i = \frac{i}{i+j-v}-\frac{i-1}{i+j-v-1}\geq 0,$$

for $v=1,\ldots,s+1$ and $i_{v-1}+2\leq i\leq i_v-1$. Using (5.6), (5.17) can be written as

$$\begin{aligned}1[j \geq 1]p_1\mu_1\Bigg(&\sum_{v=1}^{s}\Bigg(1[i_v-i_{v-1}\geq 2]g^j_{i_1,\ldots,i_{v-1},i_{v-1}+1,i_{v+1},\ldots,i_s}\frac{i_{v-1}+1}{i_{v-1}+1+j-v}\\ &+\sum_{k=i_{v-1}+2}^{i_v-1} g^j_{i_1,\ldots,i_{v-1},k,i_{v+1},\ldots,i_s}a_k - 1[i_v-i_{v-1}\geq 2]g^j_{i_1,\ldots,i_s}\frac{i_v-1}{i_v-1+j-v}\Bigg)\\ &+1[j\geq s+1]\Bigg(g^j_{i_1,\ldots,i_s,i_s+1}\frac{i_s+1}{i_s+j-s}+\sum_{k=i_s+2}^{\infty} g^j_{i_1,\ldots,i_s,k}a_k - g^j_{i_1,\ldots,i_s}\Bigg)\Bigg),\end{aligned} \tag{5.18}$$

which shows that if $g^j_{i_1,\ldots,i_s}(h(t)) = g^j_{i_1,\ldots,i_s}(\tilde{h}(t))$ with $h(t)\leq_C \tilde{h}(t)$, then the drift of $g^j_{i_1,\ldots,i_s}(\tilde{h}(t))$ due to the phase changes is at least as large as the drift of $g^j_{i_1,\ldots,i_s}(h(t))$ due to the phase changes as only the $g^j_{i_1,\ldots,i_s}$ terms have a negative coefficient in the above expression.

### 5.6.2 Drift due to job completions from phase 1

Job completions decrease $g^j_{i_1,\dots,i_s}$ in the following two ways as illustrated in Figure 5.4(middle) and (right):

- when there is a job completion of a job in phase 1 in a server with exactly $i_k$ jobs in phase 1 and exactly $j-k$ jobs in phase 2 for some $k \in \{1,\dots,s\}$;
- when there is a job completion of a job in phase 2 in a server with between $i_k$ and $i_{k+1}-1$ jobs in phase 1 and exactly $j-k$ jobs in phase 2 for some $k \in \{0,\dots,s-1[j=s]\}$.

Set $i_{s+1}=\infty$. The change to $g^j_{i_1,\dots,i_s}$ due to service completions can therefore be written as

$$-\nu_1 \sum_{k=1}^{s} w_{i_k,j-k} \frac{i_k}{i_k+j-k} \tag{5.19}$$

$$-\nu_2 \sum_{k=0}^{s-1[j=s]} \sum_{i=i_k}^{i_{k+1}-1} w_{i,j-k} \frac{j-k}{i+j-k}. \tag{5.20}$$

Note, that we can drop $-1[j=s]$ from (5.20) as $j-k=0$ in such case. We now rewrite both these expressions to show that combined they are such that only the $g^j_{i_1,\dots,i_s}$ terms have negative coefficients.

Using Lemma 5.4.6, we have that (5.19) is equal to

$$-\nu_1 g^j_{i_1,\dots,i_s} \frac{i_s}{i_s+j-s} + \nu_1 \sum_{k=0}^{s-1} g^j_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1} \left( \frac{i_{k+1}}{i_{k+1}+j-k-1} - \frac{i_k}{i_k+j-k} \right). \tag{5.21}$$

Note that the coefficients appearing in the sum are positive due to (5.12).

### 5.6.3 Drift due to job completions from phase 2: the easy case

We now proceed with (5.20). For ease of presentation we assume we have $i_{k+1}-i_k \geq 2$ as the general case is tedious. The full proof can be found in Subsection 5.6.4. So suppose $i_{k+1}-i_k \geq 2$ for all $k \in \{0,\dots,s-1\}$. We can reorder the terms in (5.20) as

$$-\nu_2 \left( w_{0,j} + \sum_{k=0}^{s} \sum_{i=i_k+1}^{i_{k+1}-2} w_{i,j-k} \frac{j-k}{i+j-k} \right.$$
$$\left. + \sum_{k=0}^{s-1} w_{i_{k+1}-1,j-k} \frac{j-k}{i_{k+1}-1+j-k} + \sum_{k=1}^{s} w_{i_k,j-k} \frac{j-k}{i_k+j-k} \right),$$

by making use of the fact that $i_{s+1}=\infty$. By means of (5.5) and (5.4), this equals

$$-\nu_2 \left( (g^j - g^{j+1}_1) + \sum_{k=0}^{s} \sum_{i=i_k+1}^{i_{k+1}-2} (g^{j+1}_{i_1,\dots,i_k,i} - g^{j+1}_{i_1,\dots,i_k,i+1}) \frac{j-k}{i+j-k} \right.$$

$$+\sum_{k=1}^{s}\left(w_{i_k-1,j-k+1}\frac{j-k+1}{i_k+j-k}+w_{i_k,j-k}\frac{j-k}{i_k+j-k}\right)\Bigg). \tag{5.22}$$

As

$$w_{i_k-1,j-k+1}+w_{i_k,j-k}=g^{j+1}_{i_1,\dots,i_{k-1},i_k-1,i_k}-g^{j+1}_{i_1,\dots,i_k,i_k+1},$$

we get that (5.22) is equal to

$$-\nu_2\Bigg((g^j-g_1^{j+1})+\sum_{k=0}^{s}\sum_{i=i_k+1}^{i_{k+1}-2}(g^{j+1}_{i_1,\dots,i_k,i}-g^{j+1}_{i_1,\dots,i_k,i+1})\frac{j-k}{i+j-k}$$
$$+\sum_{k=1}^{s}\left(g^{j+1}_{i_1,\dots,i_{k-1},i_k-1,i_k}-g^{j+1}_{i_1,\dots,i_k,i_k+1}\right)\frac{j-k+1}{i_k+j-k}-\sum_{k=1}^{s}w_{i_k,j-k}\frac{1}{i_k+j-k}\Bigg).$$

This can be restated using (5.3) as

$$-\nu_2\Bigg((g^j_{i_1,\dots,i_s}-g^{j+1}_{1,i_1,\dots,i_s})+\sum_{k=0}^{s}\sum_{i=i_k+1}^{i_{k+1}-2}(g^{j+1}_{i_1,\dots,i_k,i,i_{k+1},\dots,i_s}-g^{j+1}_{i_1,\dots,i_k,i+1,i_{k+1},\dots,i_s})\frac{j-k}{i+j-k}$$
$$+\sum_{k=1}^{s}(g^{j+1}_{i_1,\dots,i_{k-1},i_k-1,i_k,\dots,i_s}-g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s})\frac{j-k+1}{i_k+j-k}-\sum_{k=1}^{s}w_{i_k,j-k}\frac{1}{i_k+j-k}\Bigg).$$

By adding and subtracting two sums we find

$$-\nu_2\Bigg((g^j_{i_1,\dots,i_s}-g^{j+1}_{1,i_1,\dots,i_s})$$
$$+\sum_{k=0}^{s}\sum_{i=i_k+1}^{i_{k+1}-1}g^{j+1}_{i_1,\dots,i_k,i,i_{k+1},\dots,i_s}\frac{j-k}{i+j-k}-\sum_{k=0}^{s}\sum_{i=i_k}^{i_{k+1}-2}g^{j+1}_{i_1,\dots,i_k,i+1,i_{k+1},\dots,i_s}\frac{j-k}{i+j-k}$$
$$-\left(\sum_{k=1}^{s}g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s}\frac{j-k+1}{i_k+j-k}-\sum_{k=0}^{s}g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s}\frac{j-k}{i_k+j-k}\right)$$
$$-\left(\sum_{k=0}^{s}g^{j+1}_{i_1,\dots,i_k,i_{k+1}-1,i_{k+1},\dots,i_s}\frac{j-k}{i_{k+1}-1+j-k}-\sum_{k=1}^{s}g^{j+1}_{i_1,\dots,i_{k-1},i_k-1,i_k,\dots,i_s}\frac{j-k+1}{i_k+j-k}\right)$$
$$-\sum_{k=1}^{s}w_{i_k,j-k}\frac{1}{i_k+j-k}\Bigg).$$

Combining both double sums and keeping in mind that $i_{s+1}=\infty$, the above expression is equivalent to

$$-\nu_2 g^j_{i_1,\dots,i_s}+\nu_2\sum_{k=0}^{s}\sum_{i=i_k+1}^{i_{k+1}-1}g^{j+1}_{i_1,\dots,i_k,i,i_{k+1},\dots,i_s}\left(\frac{j-k}{i+j-k-1}-\frac{j-k}{i+j-k}\right)$$
$$+\nu_2\sum_{k=1}^{s}g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s}\left(\frac{j-k+1}{i_k+j-k}-\frac{j-k}{i_k+j-k}\right)$$

$$+\,\nu_2 \sum_{k=1}^{s} g^{j+1}_{i_1,\dots,i_{k-1},i_k-1,i_k,\dots,i_s} \left(\frac{j-k+1}{i_k+j-k} - \frac{j-k+1}{i_k+j-k}\right) + \nu_2 \sum_{k=1}^{s} w_{i_k,j-k} \frac{1}{i_k+j-k},$$

$$= -\nu_2 g^{j}_{i_1,\dots,i_s} + \nu_2 \sum_{k=0}^{s} \sum_{i=i_k+1}^{i_{k+1}-1} g^{j+1}_{i_1,\dots,i_k,i,i_{k+1},\dots,i_s} \left(\frac{j-k}{i+j-k-1} - \frac{j-k}{i+j-k}\right)$$

$$+\,\nu_2 \sum_{k=1}^{s} g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s} \frac{1}{i_k+j-k} + \nu_2 \sum_{k=1}^{s} w_{i_k,j-k} \frac{1}{i_k+j-k}, \tag{5.23}$$

We still need to deal with the term $\nu_2 \sum_{k=1}^{s} w_{i_k,j-k} \frac{1}{i_k+j-k}$. Using Lemma 5.4.6, we find that this term equals

$$\nu_2 g^{j}_{i_1,\dots,i_s} \frac{1}{i_s+j-s} - \nu_2 \sum_{k=0}^{s-1} g^{j}_{i_1,\dots,i_k,i_{k+1}+1,\dots,i_s+1} \left(\frac{1}{i_{k+1}+j-k-1} - \frac{1}{i_k+j-k}\right),$$

which has the same form as (5.21). This shows that the term associated with the service completions from phase 2 is also monotone when $i_{k+1} - i_k \geq 2$ for $k = 0, \dots, s$. Note that we did not rely on the fact that $\nu_1 > \nu_2$, however, this requirement is necessary for the full proof in the next Subsection where we may have $i_{k+1} = i_k + 1$ for some $k$ values.

## 5.6.4 Drift due to job completions from phase 2: the general case

The expression in (5.23) for (5.20) is only valid in case $i_{k+1} \geq i_k + 2$, for $k = 0, \dots, s$. In this Subsection we derive a general expression for (5.20), where $i_{k+1} = i_k + 1$ for some $k$ values is allowed. This is for instance needed for $g^{j+1}_{i_1,\dots,i_k,i_k+1,i_{k+1},\dots,i_s}$ to be well defined. We combine this expression with (5.21) (which was shown to be equivalent to (5.20)), to conclude that the sum of the terms corresponding to service completions in phase 1 and 2 together are monotone when $\nu_1 > \nu_2$.

For given $j, s, i_1, \dots, i_s$ as in Definition 5.4.2, with $i_s < \infty$ and $i_{s+1} = \infty$, we define inductively $d_1 = 1$ and $d_k = 1 + 1[i_k - i_{k-1} = 1]d_{k-1}$ for $k = 2, \dots, s$. We further define an injection $\sigma : \{1, \dots, \tilde{s}\} \to \{1, \dots, s\}$ as follows: $\sigma(\kappa)$ is the $\kappa$-th index $k$ such that $i_{k+1} - i_k \geq 2$, not counting whether or not $i_1 \geq 2$. As $i_{s+1} = \infty$, we have $\tilde{s} \geq 1$ and $\sigma(\tilde{s}) = s$. We also set $\sigma(0) = 0$. We now prove three lemmas which are combined afterwards.

**Lemma 5.6.1.** *Define the following formulas, these are illustrated in Figure 5.5*

$$S = \sum_{k=0}^{\sigma(1)-1} \sum_{i=i_k}^{i_{k+1}-1} w_{i,j-k} \frac{j-k}{i+j-k} + w_{i_{\sigma(1)},j-\sigma(1)} \frac{j-\sigma(1)}{i_{\sigma(1)}+j-\sigma(1)},$$

*for* $\kappa \in \{1, \dots, \tilde{s}\}$*:*

$$T^{(\sigma(\kappa))} = \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-2} w_{i,j-\sigma(\kappa)} \frac{j-\sigma(\kappa)}{i+j-\sigma(\kappa)},$$

*and for* $\kappa \in \{1, \dots, \tilde{s}-1\}$*:*

$$U^{(\sigma(\kappa))} = w_{i_{\sigma(\kappa)+1}-1,j-\sigma(\kappa)} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)}$$

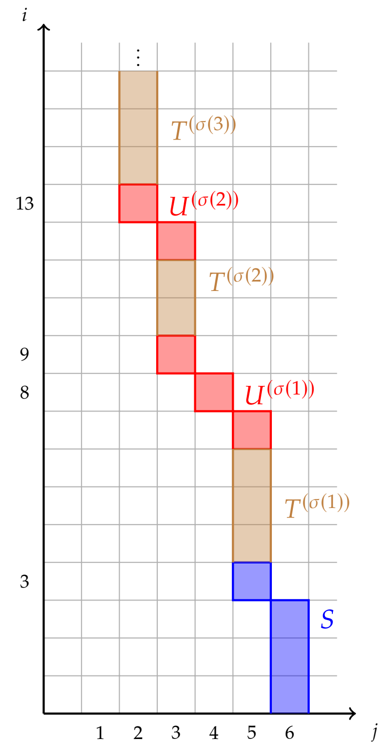


Figure 5.5: Illustration of the formulas defined in Lemma 5.6.1 for $g^6_{3,8,9,13}(h(t))$, we have $\sigma(1) = 1$, $\sigma(2) = 3$, $\sigma(3) = 4$.

$$+ \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)-1} \sum_{i=i_k}^{i_{k+1}-1} w_{i,j-k} \frac{j-k}{i+j-k} + w_{i_{\sigma(\kappa+1)},j-\sigma(\kappa+1)} \frac{j-\sigma(\kappa+1)}{i_{\sigma(\kappa+1)}+j-\sigma(\kappa+1)}.$$

*For ease of notation set for $k \notin \sigma(\{1,\ldots,\tilde{s}\})$:*

$$T^{(k)} = 0,$$

*and for $k \notin \sigma(\{1,\ldots,\tilde{s}-1\})$:*

$$U^{(k)} = 0.$$

*Then:*

$$-\nu_2 \sum_{k=0}^{s} \sum_{i=i_k}^{i_{k+1}-1} w_{i,j-k} \frac{j-k}{i+j-k} = -\nu_2 (S + \sum_{k=1}^{s} T^{(k)} + \sum_{k=1}^{s} U^{(k)}), \tag{5.24}$$

*further,*

$$S = g^j_{i_1,\ldots,i_s} - \sum_{i=1}^{i_1-1} g^{j+1}_{i,i_1,\ldots,i_s} \left( \frac{j}{i+j-1} - \frac{j}{i+j} \right)$$

$$- g^{j+1}_{i_1,\dots,i_{\sigma(1)},i_{\sigma(1)}+1,i_{\sigma(1)+1},\dots,i_s} \frac{j}{i_1 - 1 + j} - \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k + j - k},$$

*for* $\kappa \in \{1, \dots, \tilde{s}\}$*:*

$$\begin{aligned} T^{(\sigma(\kappa))} = {} & g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)}+1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)} + j - \sigma(\kappa)} \\ & - \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-1} g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i,i_{\sigma(\kappa)+1},\dots,i_s} \left( \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa) - 1} - \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa)} \right) \\ & - 1[\kappa < \tilde{s}] g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)+1} - 1 + j - \sigma(\kappa)}, \end{aligned}$$

*and for* $\kappa \in \{1, \dots, \tilde{s} - 1\}$*:*

$$\begin{aligned} U^{(\sigma(\kappa))} = {} & g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)+1} - 1 + j - \sigma(\kappa)} \\ & - g^{j+1}_{i_1,\dots,i_{\sigma(\kappa+1)},i_{\sigma(\kappa+1)}+1,i_{\sigma(\kappa+1)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)+1} - 1 + j - \sigma(\kappa)} - \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)} w_{i_k,j-k} \frac{d_k}{i_k + j - k}. \end{aligned}$$

*Proof.* By writing

$$\begin{aligned} S + \sum_{k=1}^{s} T^{(k)} + \sum_{k=1}^{s} U^{(k)} &= S + \sum_{\kappa=1}^{\tilde{s}} T^{(\sigma(\kappa))} + \sum_{\kappa=1}^{\tilde{s}-1} U^{(\sigma(\kappa))} \\ &= S + T^{(\sigma(1))} + U^{(\sigma(1))} + \cdots + T^{(\sigma(\tilde{s}-1))} + U^{(\sigma(\tilde{s}-1))} + T^{(\sigma(\tilde{s}))}, \end{aligned}$$

the first claim clearly holds. We have for $\kappa \in \{1, \dots, \tilde{s}\}$

$$\begin{aligned} T^{(\sigma(\kappa))} &= \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-2} \left( g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i} - g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i+1} \right) \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa)} \\ &= \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-2} \left( g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i,i_{\sigma(\kappa)+1},\dots,i_s} - g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i+1,i_{\sigma(\kappa)+1},\dots,i_s} \right) \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa)}, \end{aligned}$$

where we relied on (5.4) for the first equality and (5.3) for the second. By adding and subtracting zero to the sums, we find that this is equal to (where the indicator function is due to $i_{\sigma(\tilde{s})+1} = i_{s+1} = \infty$)

$$\begin{aligned} & g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)}+1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)} + j - \sigma(\kappa)} - \sum_{i=i_{\sigma(\kappa)}}^{i_{\sigma(\kappa)+1}-2} g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i+1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa)} \\ & + \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-1} g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i + j - \sigma(\kappa)} - g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{1[\kappa < \tilde{s}](j - \sigma(\kappa))}{i_{\sigma(\kappa)+1} - 1 + j - \sigma(\kappa)} \\ & = g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)}+1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j - \sigma(\kappa)}{i_{\sigma(\kappa)} + j - \sigma(\kappa)} \end{aligned}$$

$$
\begin{aligned}
&- \sum_{i=i_{\sigma(\kappa)}+1}^{i_{\sigma(\kappa)+1}-1} g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i,i_{\sigma(\kappa)+1},\dots,i_s} \left( \frac{j-\sigma(\kappa)}{i+j-\sigma(\kappa)-1} - \frac{j-\sigma(\kappa)}{i+j-\sigma(\kappa)} \right) \\
&- 1[\kappa < \tilde{s}] g^{j+1}_{i_1,\dots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\dots,i_s} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)}.
\end{aligned}
$$

This proves the expression for $T^{(\sigma(\kappa))}$. Suppose $i_1 = 1$, then by the definition of $\sigma$,

$$
S = \sum_{k=0}^{\sigma(1)} w_{i_k,j-k} \frac{j-k}{i_k+j-k}.
$$

The definition of $\sigma$ implies that $i_k = k = d_k$ for $k = 1, \dots, \sigma(1)$ when $i_1 = 1$. By using (5.5) and then (5.3), we get

$$
\begin{aligned}
S &= g^j - g_1^{j+1} + \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{j-k}{i_k+j-k} = g^j - g_1^{j+1} + \sum_{k=1}^{\sigma(1)} w_{i_k,j-k}(1 - \frac{k}{i_k+j-k}) \\
&= g^j_{i_1,\dots,i_{\sigma(1)}} - g^{j+1}_{1,i_1+1,\dots,i_{\sigma(1)}+1} - \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k} \\
&= g^j_{i_1,\dots,i_s} - g^{j+1}_{i_1,\dots,i_{\sigma(1)},i_{\sigma(1)}+1,i_{\sigma(1)+1},\dots,i_s} - \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k}.
\end{aligned}
$$

Suppose now $i_1 \geq 2$, then similarly as $i_k - k = i_1 - 1$ and $d_k = k$ for $k = 1, \dots, \sigma(1)$

$$
\begin{aligned}
S &= \sum_{i=0}^{i_1-1} w_{i,j} \frac{j}{i+j} + \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{j-k}{i_k+j-k} \\
&= g^j - g_1^{j+1} + \sum_{i=1}^{i_1-1} w_{i,j} \frac{j}{i+j} + \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{j-k}{i_k+j-k} \\
&= g^j_{i_1,\dots,i_s} - g^{j+1}_{1,i_1,\dots,i_s} + \sum_{i=1}^{i_1-2} \left( g^{j+1}_{i,i_1,\dots,i_s} - g^{j+1}_{i+1,i_1,\dots,i_s} \right) \frac{j}{i+j} \\
&+ \left( g^{j+1}_{i_1-1} - g^{j+1}_{i_1} \right) \frac{j}{i_1-1+j} + \sum_{k=1}^{\sigma(1)} w_{i_k,j-k}( \frac{j}{i_1-1+j} - \frac{k}{i_k+j-k}) \\
&= g^j_{i_1,\dots,i_s} - g^{j+1}_{1,i_1,\dots,i_s} + \sum_{i=1}^{i_1-2} \left( g^{j+1}_{i,i_1,\dots,i_s} - g^{j+1}_{i+1,i_1,\dots,i_s} \right) \frac{j}{i+j} \\
&+ \left( g^{j+1}_{i_1-1,i_1,\dots,i_{\sigma(1)}} - g^{j+1}_{i_1,i_1+1,\dots,i_{\sigma(1)}+1} \right) \frac{j}{i_1-1+j} - \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k}.
\end{aligned}
$$

Similarly to the proof of $T^{(k)}$, we find that $S$ is equal to

$$
g^j_{i_1,\dots,i_s} - g^{j+1}_{1,i_1,\dots,i_s} + g^{j+1}_{1,i_1,\dots,i_s} - g^{j+1}_{i_1-1,i_1,\dots,i_s} \frac{j}{i_1-1+j}
$$

$$
\begin{aligned}
&- \sum_{i=1}^{i_1-1} g^{j+1}_{i,i_1,\ldots,i_s} \left( \frac{j}{i+j-1} - \frac{j}{i+j} \right) + \left( g^{j+1}_{i_1-1,i_1,\ldots,i_s} - g^{j+1}_{i_1,i_1+1,\ldots,i_{\sigma(1)}+1,i_{\sigma(1)+1},\ldots,i_s} \right) \frac{j}{i_1-1+j} \\
&- \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k} \\
&= g^{j}_{i_1,\ldots,i_s} - \sum_{i=1}^{i_1-1} g^{j+1}_{i,i_1,\ldots,i_s} \left( \frac{j}{i+j-1} - \frac{j}{i+j} \right) \\
&- g^{j+1}_{i_1,\ldots,i_{\sigma(1)},i_{\sigma(1)}+1,i_{\sigma(1)+1},\ldots,i_s} \frac{j}{i_1-1+j} - \sum_{k=1}^{\sigma(1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k}.
\end{aligned}
$$

Finally, noting that $i_{k+1} - 1 = i_k$ for $k = \sigma(\kappa)+1, \ldots, \sigma(\kappa+1)-1$, $i_k - k = i_{\sigma(\kappa)+1} - \sigma(\kappa) - 1$ and $d_k = k - \sigma(\kappa)$ for $k = \sigma(\kappa)+1, \ldots, \sigma(\kappa+1)$, we find similarly to the proof of $S$ with $i_1 = 1$ for $\kappa \in \{1, \ldots, \tilde{s}-1\}$:

$$
\begin{aligned}
U^{(\sigma(\kappa))} &= w_{i_{\sigma(\kappa)+1}-1,j-\sigma(\kappa)} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} + \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)} w_{i_k,j-k} \frac{j-k}{i_k+j-k} \\
&= \left( g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1} - g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa)+1}} \right) \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} \\
&+ \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)} w_{i_k,j-k} \left( \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}+j-\sigma(\kappa)-1} + \frac{\sigma(\kappa)-k}{i_k+j-k} \right) \\
&= g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\ldots,i_{\sigma(\kappa+1)}} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} \\
&- g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa)+1},i_{\sigma(\kappa)+1}+1,\ldots,i_{\sigma(\kappa+1)}+1} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} \\
&- \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k} \\
&= g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa)},i_{\sigma(\kappa)+1}-1,i_{\sigma(\kappa)+1},\ldots,i_s} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} \\
&- g^{j+1}_{i_1,\ldots,i_{\sigma(\kappa+1)},i_{\sigma(\kappa+1)}+1,i_{\sigma(\kappa+1)+1},\ldots,i_s} \frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}-1+j-\sigma(\kappa)} \\
&- \sum_{k=\sigma(\kappa)+1}^{\sigma(\kappa+1)} w_{i_k,j-k} \frac{d_k}{i_k+j-k}.
\end{aligned}
$$

This finishes the proof of the first lemma. □

**Lemma 5.6.2.** *We have for* $\kappa = 0, \ldots, \tilde{s}-1$

$$
\frac{j-\sigma(\kappa)}{i_{\sigma(\kappa)+1}+j-(\sigma(\kappa)+1)} \geq \frac{j-\sigma(\kappa+1)}{i_{\sigma(\kappa+1)}+j-\sigma(\kappa+1)}.
$$

*Proof.* As $i_{\sigma(\kappa+1)} - \sigma(\kappa+1) \geq i_{\sigma(\kappa)+1} - (\sigma(\kappa)+1)$, the claim holds if

$$\frac{j-\sigma(\kappa)}{i_{\sigma(\kappa+1)}+j-\sigma(\kappa+1)} \geq \frac{j-\sigma(\kappa+1)}{i_{\sigma(\kappa+1)}+j-\sigma(\kappa+1)}.$$

But this is true as $\sigma(\kappa+1) > \sigma(\kappa)$. □

**Lemma 5.6.3.** *We have for every* $k = 1, \ldots, s-1$*: if* $\frac{d_{k+1}}{i_{k+1}+j-(k+1)} - \frac{d_k}{i_k+j-k} > 0$ *then*

$$\frac{d_{k+1}}{i_{k+1}+j-(k+1)} - \frac{d_k}{i_k+j-k} = \frac{i_{k+1}}{i_{k+1}+j-(k+1)} - \frac{i_k}{i_k+j-k}. \tag{5.25}$$

*Proof.* Suppose $d_{k+1} = 1$, i.e. $i_{k+1} - i_k \geq 2$. Then $i_k + j - k < i_{k+1} + j - (k+1)$ and thus

$$\begin{aligned}\frac{d_{k+1}}{i_{k+1}+j-(k+1)} - \frac{d_k}{i_k+j-k} &< \frac{d_{k+1}}{i_{k+1}+j-(k+1)} - \frac{d_k}{i_{k+1}+j-(k+1)} \\ &= \frac{1-d_k}{i_{k+1}+j-(k+1)} \leq 0.\end{aligned}$$

Hence, it suffices prove the claim for the case where $d_{k+1} = 1 + d_k$, i.e. where $i_{k+1} - i_k = 1$. (5.25) is then equivalent to

$$\frac{d_{k+1}}{i_k+j-k} - \frac{d_k}{i_k+j-k} = \frac{i_{k+1}}{i_k+j-k} - \frac{i_k}{i_k+j-k},$$

which is true. □

We now combine these Lemmas. We can rewrite (5.24) using Lemma 5.6.1. Note that the last term in $T^{(\sigma(\kappa))}$ cancels the first term in $U^{(\sigma(\kappa))}$, while the first term in $T^{(\sigma(\kappa))}$ can be combined with the last term in $U^{(\sigma(\kappa-1))}$ for $\kappa > 1$ and with $S$ for $\kappa = 1$. Then, by Lemma 5.6.2 all these terms have the desired sign and only the following sum remains to be dealt with

$$\nu_2 \sum_{k=1}^{s} w_{i_k,j-k} \frac{d_k}{i_k+j-k}. \tag{5.26}$$

Using Lemma 5.4.6, we can rewrite (5.26) as

$$\begin{aligned}\nu_2 \sum_{k=1}^{s} w_{i_k,j-k} \frac{d_k}{i_k+j-k} &= \nu_2 g^{j}_{i_1,\ldots,i_s} \frac{d_s}{i_s+j-s} \\ &- \nu_2 \sum_{k=0}^{s-1} g^{j}_{i_1,\ldots,i_k,i_{k+1}+1,\ldots,i_s+1} \left( \frac{d_{k+1}}{i_{k+1}+j-k-1} - \frac{d_k}{i_k+j-k} \right).\end{aligned}$$

The coefficients in this sum may not have the desired sign, however, if this is the case they can be combined with the expression that was derived for (5.19), that is,

$$\begin{aligned}-\nu_1 \sum_{k=1}^{s} w_{i_k,j-k} \frac{i_k}{i_k+j-k} &= -\nu_1 g^{j}_{i_1,\ldots,i_s} \frac{i_s}{i_s+j-s} \\ &+ \nu_1 \sum_{k=0}^{s-1} g^{j}_{i_1,\ldots,i_k,i_{k+1}+1,\ldots,i_s+1} \left( \frac{i_{k+1}}{i_{k+1}+j-k-1} - \frac{i_k}{i_k+j-k} \right).\end{aligned}$$

The proof of Proposition 5.5.2 is now finished using Lemma 5.6.3 and $\nu_1 > \nu_2$.

## 5.7 The Supermarket Model

In this section we show that Assumptions 5.1-5.3 hold for the supermarket model with processor sharing. We start by describing the expressions for $f_{i,j}(h(t))$. We first note that $f_{0,j}(h(t)) = 0$ as new jobs start service in phase 1 (that is, $\alpha = (1, 0)$ when using the Coxian representation). So suppose $i \geq 1$. The probability that all $d$ chosen servers have at least $i + j - 1$ jobs but not all have at least $i + j$ jobs is $h^d_{i+j-1,0}(t) - h^d_{i+j,0}(t)$. In other words, this is the probability that at least one chosen server has exactly $i + j - 1$ jobs. As

$$\frac{h_{i-1,j}(t) - h_{i,j}(t)}{h_{i+j-1,0}(t) - h_{i+j,0}(t)}$$

is the probability that a server with exactly $i + j - 1$ jobs has at least $j$ jobs in phase 2, the arrival terms are given by

$$\begin{aligned} f_{i,j}(h(t)) &= \lambda \left(h^d_{i+j-1,0}(t) - h^d_{i+j,0}(t)\right) \frac{h_{i-1,j}(t) - h_{i,j}(t)}{h_{i+j-1,0}(t) - h_{i+j,0}(t)} \\ &= \lambda \left(\sum_{\ell=0}^{d-1} h^\ell_{i+j-1,0}(t) h^{d-1-\ell}_{i+j,0}(t)\right)(h_{i-1,j}(t) - h_{i,j}(t)), \end{aligned} \tag{5.27}$$

where we have used the identity $(a^d - b^d)/(a - b) = \sum_{\ell=0}^{d-1} a^\ell b^{d-1-\ell}$. Define $\Psi_{i,j}(h(t)) = \frac{d}{dt}h_{i,j}(t)$ and $\Psi(h) = \left[\Psi_{i,j}(h)\right]_{i,j=0}^\infty$. We first show that $\Psi$ is Lipschitz continuous on $\Omega_B$. We use the supremum metric on $\Omega_B$:

$$\mathbf{d}(h, \tilde{h}) = \sup_{i,j=0}^\infty |h_{i,j} - \tilde{h}_{i,j}|. \tag{5.28}$$

**Proposition 5.7.1.** *The drift $\Psi$ is Lipschitz continuous on $\Omega_B$, meaning Assumption 5.1 is met when $f_{i,j}(h(t))$ is defined as in* (5.27).

*Proof.* Let $h, \tilde{h} \in \Omega_B$ and let $w, \tilde{w}$ be the corresponding vectors as defined in Section 5.3. As $w_{i,j} = (h_{i,j} - h_{i-1,j+1}) - (h_{i+1,j} - h_{i,j+1})$ for $i \geq 1$ and $w_{0,j} = h_{0,j} - h_{1,j}$, we have

$$\sup_{i,j=0}^\infty |w_{i,j} - \tilde{w}_{i,j}| \leq 4\mathbf{d}(h, \tilde{h}). \tag{5.29}$$

We have

$$\begin{aligned} \nu_2 \sup_{i,j=0}^\infty \left| \sum_{k=i+1}^\infty (w_{k,j} - \tilde{w}_{k,j}) \frac{j}{k+j} \right| &\leq \nu_2 \sup_{i,j=0}^\infty \sum_{k=i+1}^\infty \left| w_{k,j} - \tilde{w}_{k,j} \right| \\ &\leq \nu_2(B+1) \sup_{i,j=0}^\infty \left| w_{i,j} - \tilde{w}_{i,j} \right| \leq 4\nu_2(B+1)\mathbf{d}(h, \tilde{h}), \end{aligned}$$

where we have used (5.29) in the last inequality. Proceeding similarly we get

$$\begin{aligned} \mathbf{d}(\Psi(h), \Psi(\tilde{h})) &\leq 4(B+1)(\mu_1 + 2\mu_2)\mathbf{d}(h, \tilde{h}) \\ &+ \lambda \sup_{i,j=0}^\infty \left| (h_{i,j} - h_{i+1,j}) \sum_{\ell=0}^{d-1} h^\ell_{i+j,0} h^{d-1-\ell}_{i+j+1,0} - (\tilde{h}_{i,j} - \tilde{h}_{i+1,j}) \sum_{\ell=0}^{d-1} \tilde{h}^\ell_{i+j,0} \tilde{h}^{d-1-\ell}_{i+j+1,0} \right|. \end{aligned}$$

We now use the inequality $|a_1^{m_1}a_2^{m_2} - b_1^{m_1}b_2^{m_2}| \le m_1|a_1 - b_1| + m_2|a_2 - b_2|$, for $0 \le a_1, a_2, b_1, b_2 \le 1$ and $m_1, m_2 \in \mathbb{N} \setminus \{0\}$, to find that

$$\begin{aligned}
&\sup_{i,j=0}^{\infty}\left|(h_{i,j} - h_{i+1,j})\sum_{\ell=0}^{d-1} h_{i+j,0}^{\ell} h_{i+j+1,0}^{d-1-\ell} - (\tilde{h}_{i,j} - \tilde{h}_{i+1,j})\sum_{\ell=0}^{d-1} \tilde{h}_{i+j,0}^{\ell}\tilde{h}_{i+j+1,0}^{d-1-\ell}\right| \\
&\le \sup_{i,j=0}^{\infty}|h_{i,j} - h_{i+1,j} - (\tilde{h}_{i,j} - \tilde{h}_{i+1,j})| + \sup_{i,j=0}^{\infty}\left|\sum_{\ell=0}^{d-1}(h_{i+j,0}^{\ell}h_{i+j+1,0}^{d-1-\ell} - \tilde{h}_{i+j,0}^{\ell}\tilde{h}_{i+j+1,0}^{d-1-\ell})\right| \\
&\le 2\mathbf{d}(h,\tilde{h}) + \sup_{i,j=0}^{\infty}\sum_{\ell=0}^{d-1}\left|h_{i+j,0}^{\ell}h_{i+j+1,0}^{d-1-\ell} - \tilde{h}_{i+j,0}^{\ell}\tilde{h}_{i+j+1,0}^{d-1-\ell}\right|.
\end{aligned}$$

Using the above mentioned inequality once more, we get

$$\begin{aligned}
&\sup_{i,j=0}^{\infty}\sum_{\ell=0}^{d-1}\left|h_{i+j,0}^{\ell}h_{i+j+1,0}^{d-1-\ell} - \tilde{h}_{i+j,0}^{\ell}\tilde{h}_{i+j+1,0}^{d-1-\ell}\right| \\
&\le \sup_{i,j=0}^{\infty}\sum_{\ell=0}^{d-1}\Big(\ell|h_{i+j,0} - \tilde{h}_{i+j,0}| + (d-1-\ell)|h_{i+j+1,0} - \tilde{h}_{i+j+1,0}|\Big) \\
&\le 2d^2\mathbf{d}(h,\tilde{h}).
\end{aligned}$$

To conclude, we have

$$\mathbf{d}(\Psi(h),\Psi(\tilde{h})) \le \Big(4(B+1)(\mu_1 + 2\mu_2) + 2\lambda(d^2+1)\Big)\,\mathbf{d}(h,\tilde{h}).$$

□

We now proceed with Assumption 5.2. Let $j, s, i_1, \ldots, i_s$ be as in Definition 5.4.2. Define

$$b_k = \lambda\sum_{\ell=0}^{d-1} h_{i_k+j-k-1,0}^{\ell}(t)h_{i_k+j-k,0}^{d-1-\ell}(t),$$

to simplify the notation. Note, that $b_k > b_{k+1}$ if $i_{k+1} - i_k \ge 2$ and $b_k = b_{k+1}$ if $i_{k+1} - i_k = 1$. We then have that $F_{i_1,\ldots,i_s}^{j}(h(t))$, as defined in Definition 5.5.1, is given by

$$\begin{aligned}
F_{i_1,\ldots,i_s}^{j}(h(t)) &= f_{0,j}(h(t)) + \sum_{k=1}^{s}(f_{i_k,j-k}(h(t)) - f_{i_k-1,j-k+1}(h(t))) \\
&= \lambda\sum_{k=1}^{s}\Big(h_{i_k+j-k-1,0}^{d}(t) - h_{i_k+j-k,0}^{d}(t)\Big) \\
&\quad\cdot\frac{(h_{i_k-1,j-k}(t) - h_{i_k,j-k}(t)) - 1[i_k \ge 2](h_{i_k-2,j-k+1}(t) - h_{i_k-1,j-k+1}(t))}{h_{i_k+j-k-1,0}(t) - h_{i_k+j-k,0}(t)} \\
&= \sum_{k=1}^{s} b_k w_{i_k-1,j-k}(h(t)).
\end{aligned}$$

The change due to arrivals is illustrated in Figure 5.6.

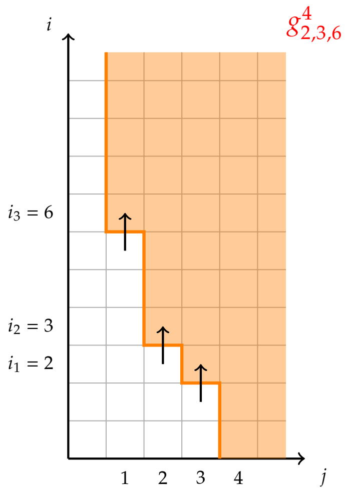


Figure 5.6: Illustration of change to $g^4_{2,3,6}(h(t))$ due to arrivals with JSQ($d$).

**Proposition 5.7.2.** *Assumption 5.2 holds for the system of ODEs* (5.1) *with* $f_{i,j}(h(t))$ *specified by* (5.27).

*Proof.* It suffices to show that $F^j_{i_1,\ldots,i_s}(h(t))$ is non-decreasing in $g^{j'}_{i'_1,\ldots,i'_{s'}}(h(t))$ for all sets of indices $\{j', i'_1, \ldots, i'_{s'}\}$, as in (5.2), different from $\{j, i_1, \ldots, i_s\}$. Suppose first that $i_1 \geq 2$. For ease of notation set $b_0 = b_{s+1} = 0$. By using Lemma 5.4.6, we find

$$\begin{aligned}
F^j_{i_1,\ldots,i_s}(h(t)) &= \sum_{k=1}^{s} b_k w_{i_k-1,j-k}(h(t)) \\
&= g^j_{i_1-1,\ldots,i_s-1}(h(t))b_s - \sum_{k=0}^{s-1} g^j_{i_1-1,\ldots,i_k-1,i_{k+1},\ldots,i_s}(h(t))\Big(b_{k+1} - b_k\Big) \\
&= -g^j_{i_1,\ldots,i_s}(h(t))b_1 + \sum_{k=1}^{s} g^j_{i_1-1,\ldots,i_k-1,i_{k+1},\ldots,i_s}(h(t))\Big(b_k - b_{k+1}\Big).
\end{aligned}$$

Suppose now $i_1 = 1$. We have

$$F^j_{i_1,\ldots,i_s}(h(t)) = b_1 w_{0,j-1}(h(t)) + \sum_{k=2}^{s} b_k w_{i_k-1,j-k}(h(t)).$$

Using (5.5) and (5.4), this is equal to

$$b_1(g^{j-1}(h(t)) - g^j_1(h(t))) + \sum_{k=2}^{s} b_k(g^{j-1}_{i_2-1,\ldots,i_k-1}(h(t)) - g^{j-1}_{i_2-1,\ldots,i_{k-1}-1,i_k}(h(t))).$$

By (5.3), we can write this as

$$b_1(g^{j-1}_{i_2,\ldots,i_s}(h(t)) - g^j_{i_1,\ldots,i_s}(h(t))) + \sum_{k=2}^{s} b_k(g^{j-1}_{i_2-1,\ldots,i_k-1,i_{k+1},\ldots,i_s}(h(t)) - g^{j-1}_{i_2-1,\ldots,i_{k-1}-1,i_k,\ldots,i_s}(h(t)))$$

$$= -b_1 g^j_{i_1,\ldots,i_s}(h(t)) + \sum_{k=1}^{s} g^{j-1}_{i_2-1,\ldots,i_k-1,i_{k+1},\ldots,i_s}(h(t))\,(b_k - b_{k+1})\,.$$

This finishes the proof. □

We now present a fixed point $\pi$ of the system of the ODEs given in (5.1) when $f_{i,j}(h(t))$ is given by (5.27) and then prove that it is the unique fixed point. As

$$h_{i,j}(t) = \sum_{k \ge i+j} \sum_{\ell=j}^{k} w_{k-\ell,\ell}(h(t)),$$

finding a fixed point of (5.1) is equivalent to finding a fixed point $\pi^w$ of the corresponding set of equations for $\frac{d}{dt} w_{i,j}(h(t))$. We have

$$\begin{aligned}
\frac{d}{dt} w_{i,j}(h(t)) &= 1[i \ge 1]\lambda \left(h^d_{i+j-1,0}(t) - h^d_{i+j,0}(t)\right) \frac{w_{i-1,j}(h(t))}{h_{i+j-1,0}(t) - h_{i+j,0}(t)} \\
&- \lambda \left(h^d_{i+j,0}(t) - h^d_{i+j+1,0}(t)\right) \frac{w_{i,j}(h(t))}{h_{i+j,0}(t) - h_{i+j+1,0}(t)} \\
&+ 1[j \ge 1] p_1 \mu_1 w_{i+1,j-1}(h(t)) \frac{i+1}{i+j} - \mu_1 w_{i,j}(h(t)) \frac{i}{i+j} - \mu_2 w_{i,j}(h(t)) \frac{j}{i+j} \\
&+ \mu_2 w_{i,j+1}(h(t)) \frac{j+1}{i+j+1} + (1-p_1)\mu_1 w_{i+1,j}(h(t)) \frac{i+1}{i+j+1}, \qquad (5.30)
\end{aligned}$$

for $i, j \ge 0$ and $i + j \le B$, where $w_{i,j}(h(t)) = 0$ for $i + j > B$. Consider the set of ODEs for $i = 1, \ldots, B$ given by

$$\frac{d}{dt} \hat{h}_i(t) = \lambda(\hat{h}_{i-1}(t)^d - \hat{h}_i(t)^d) - (\hat{h}_i(t) - \hat{h}_{i+1}(t)) \qquad (5.31)$$

and set $\hat{h}_0(t) = 1$ and $\hat{h}_i(t) = 0$, for $i > B$. Notice this set of ODEs corresponds to the mean field limit of the supermarket model with FCFS or processor sharing service, exponential job sizes and a finite buffer of size $B$ [58].

**Proposition 5.7.3.** *The set of ODEs given by* (5.31) *has a unique fixed point* $\hat{\pi}$. *Further* $\hat{\pi}_i \in (0, \lambda^{(d^i-1)/(d-1)})$ *and* $\hat{\pi}_i > \hat{\pi}_{i+1}$.

*Proof.* Summing the equations in (5.31) from $i = k$ to $B$ yields that any fixed point $\hat{\pi}$ satisfies

$$\hat{\pi}_k = \lambda(\hat{\pi}^d_{k-1} - \hat{\pi}^d_B),$$

for $k = 2, \ldots, B$ and $\hat{\pi}_1 = \lambda(1 - \hat{\pi}^d_B)$. Define $H_1(x) = \lambda(1 - x^d)$ and $H_k(x) = \lambda(H_{k-1}(x)^d - x^d)$ for $k = 2, \ldots, B$, then $\hat{\pi}_k = H_k(\hat{\pi}_B)$. Using induction on $k$ one now readily shows that $H_k(0) = \lambda^{1+d+\ldots+d^{k-1}} = \lambda^{(d^k-1)/(d-1)} > 0$.

We first consider the case where $d$ is odd. As $H_{k-1}(x)^{d-1} \ge 0$ for $d$ odd and $H'_k(x) = \lambda d H_{k-1}(x)^{d-1} H'_{k-1}(x) - \lambda d x^{d-1}$, we immediately have by induction on $k$ that $H_k(x)$ is decreasing on $[0, 1]$ with $H_k(1) \le 0$. As $\hat{\pi}_B = H_B(\hat{\pi}_B)$, this implies that there exists a unique solution for $\hat{\pi}_B \in (0, \lambda^{(d^B-1)/(d-1)})$ and therefore at most one fixed point as

$\hat{\pi}_k = H_k(\hat{\pi}_B)$. Further, we see that $\hat{\pi}_k \le \lambda^{(d^k-1)/(d-1)}$ as $H_k(0) = \lambda^{(d^k-1)/(d-1)}$ and $H_k(x)$ is decreasing. Finally, $\hat{\pi}_k < \hat{\pi}_{k-1}$ as $\hat{\pi}_k = \lambda(\hat{\pi}_{k-1}^d - \hat{\pi}_B^d)$ with $\hat{\pi}_B$ positive.

Now assume $d$ is even. $H_1(x)$ is clearly decreasing and positive on $[0,1]$, therefore $H_2(x)$ is decreasing on $[0,1]$ and $H_2(1) = -\lambda < 0$. Let $\xi_2$ be the unique root of $H_2(x)$ in $(0,1)$. For $x \in (\xi_2, 1]$ we have $-x < H_2(x) < 0$. We now find using induction on $k$ that for $k = 3, \dots, B$

1. $H_k(x)$ is decreasing on $[0, \xi_{k-1})$ as $H'_k(x) = \lambda d H_{k-1}(x)^{d-1} H'_{k-1}(x) - \lambda d x^{d-1}$ and $H_{k-1}(x)$ is positive and decreasing on $[0, \xi_{k-1})$,
2. $H_k(\xi_{k-1}) = -\lambda \xi_{k-1}^d < 0$ and $-x < -\lambda x^d < H_k(x) < 0$ for $x \in [\xi_{k-1}, 1]$, where we use (i) that $H_{k-1}(x)^d \ge 0$ for $d$ even to find that $-\lambda x^d < H_k(x)$ and (ii) that $-x < H_{k-1}(x)$ yields $x^d > H_{k-1}(x)^d$ for $d$ even, that is, $H_k(x) < 0$.
3. $H_k(x)$ has a unique root $\xi_k$ on $[0,1]$ with $\xi_k < \xi_{k-1}$.

The proof now proceeds as in the case with $d$ odd. □

We now define $\pi^w$ as

$$\pi^w_{i,j} = (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \binom{i+j}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^j, \tag{5.32}$$

and show that it is a fixed point of (5.30). Note that $\pi^w$ depends on $\lambda$ via $\hat{\pi}$. If we then define $\pi_{i,j} = \sum_{k \ge i+j} \sum_{\ell=j}^k \pi^w_{k-\ell,\ell}$, then $\pi$ is a fixed point of (5.1) and

$$\pi_{i,0} = \sum_{k \ge i} \sum_{\ell=0}^{k} \pi^w_{k-\ell,\ell} = \sum_{k \ge i} (\hat{\pi}_k - \hat{\pi}_{k+1}) \left(\frac{1}{\mu_1} + \frac{p_1}{\mu_2}\right)^k = \hat{\pi}_i,$$

as the mean job size equals one, that is, $1/\mu_1 + p_1/\mu_2 = 1$. Hence the probability of having $i$ or more jobs in the fixed point $\pi$ is the same as in the exponential case.

**Proposition 5.7.4.** *$\pi^w$ is a fixed point of* (5.30).

*Proof.* $\pi^w$ is a fixed point of (5.30) if

$$\begin{aligned} 0 = {} & 1[i \ge 1]\lambda \left(\hat{\pi}^d_{i+j-1} - \hat{\pi}^d_{i+j}\right) \frac{\pi^w_{i-1,j}}{\hat{\pi}_{i+j-1} - \hat{\pi}_{i+j}} - \lambda \left(\hat{\pi}^d_{i+j} - \hat{\pi}^d_{i+j+1}\right) \frac{\pi^w_{i,j}}{\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}} \\ & + 1[j \ge 1] p_1 \mu_1 \pi^w_{i+1,j-1} \frac{i+1}{i+j} - \mu_1 \pi^w_{i,j} \frac{i}{i+j} - \mu_2 \pi^w_{i,j} \frac{j}{i+j} \\ & + \mu_2 \pi^w_{i,j+1} \frac{j+1}{i+j+1} + (1-p_1)\mu_1 \pi^w_{i+1,j} \frac{i+1}{i+j+1}. \end{aligned}$$

By using the definition of $\pi^w$, this is equivalent to

$$0 = 1[i \ge 1]\lambda(\hat{\pi}^d_{i+j-1} - \hat{\pi}^d_{i+j}) \binom{i+j-1}{i-1} \left(\frac{1}{\mu_1}\right)^{i-1} \left(\frac{p_1}{\mu_2}\right)^j$$

$$
\begin{aligned}
&- \lambda(\hat{\pi}^d_{i+j} - \hat{\pi}^d_{i+j+1}) \binom{i+j}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^j \\
&+ 1[j \geq 1] p_1 \mu_1 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \binom{i+j}{i+1} \left(\frac{1}{\mu_1}\right)^{i+1} \left(\frac{p_1}{\mu_2}\right)^{j-1} \frac{i+1}{i+j} \\
&- \mu_1 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \binom{i+j}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^j \frac{i}{i+j} \\
&- \mu_2 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \binom{i+j}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^j \frac{j}{i+j} \\
&+ \mu_2 (\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2}) \binom{i+j+1}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^{j+1} \frac{j+1}{i+j+1} \\
&+ (1-p_1)\mu_1 (\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2}) \binom{i+j+1}{i+1} \left(\frac{1}{\mu_1}\right)^{i+1} \left(\frac{p_1}{\mu_2}\right)^j \frac{i+1}{i+j+1}.
\end{aligned}
$$

We can rewrite this as

$$
\begin{aligned}
0 = {} & \lambda(\hat{\pi}^d_{i+j-1} - \hat{\pi}^d_{i+j}) \frac{\mu_1 i}{i+j} - \lambda(\hat{\pi}^d_{i+j} - \hat{\pi}^d_{i+j+1}) \\
&+ \mu_2 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \frac{j}{i+j} - \mu_1 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \frac{i}{i+j} \\
&- \mu_2 (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) \frac{j}{i+j} + p_1 (\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2}) + (1-p_1)(\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2}),
\end{aligned}
$$

or, equivalently,

$$
0 = (\lambda(\hat{\pi}^d_{i+j-1} - \hat{\pi}^d_{i+j}) - (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1})) \frac{\mu_1 i}{i+j} - (\lambda(\hat{\pi}^d_{i+j} - \hat{\pi}^d_{i+j+1}) - (\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2})). \tag{5.33}
$$

But this is true by definition of $\hat{\pi}$. □

**Proposition 5.7.5.** *$\pi^w$ is the unique fixed point of* (5.30) *and therefore Assumption 5.3 holds for the set of ODEs in* (5.1) *with $f_{i,j}(h(t))$ specified by* (5.27).

*Proof.* The proof proceeds similar to [81, Theorem 3]. We first argue that any fixed point $\theta$ must have the same form as in (5.32), that is, it can be written as

$$
\theta_{i,j} = (\hat{\theta}_{i+j} - \hat{\theta}_{i+j+1}) \binom{i+j}{i} \left(\frac{1}{\mu_1}\right)^i \left(\frac{p_1}{\mu_2}\right)^j. \tag{5.34}
$$

Let

$$
\lambda_{i+j} = \lambda(\theta^d_{i+j,0} - \theta^d_{i+j+1,0})/(\theta_{i+j,0} - \theta_{i+j+1,0})
$$

and replace $\lambda\left(h^d_{i+j-1,0}(t) - h^d_{i+j,0}(t)\right)/(h_{i+j-1,0}(t) - h_{i+j,0}(t))$ in the set of ODEs given by (5.30) by $\lambda_{i+j-1}$ and $\lambda\left(h^d_{i+j,0}(t) - h^d_{i+j+1,0}(t)\right)/(h_{i+j,0}(t) - h_{i+j+1,0}(t))$ by $\lambda_{i+j}$. Then, $\theta$ is also a fixed point of this new set of ODEs. However this system of ODEs corresponds an $M/PH/1$ queue with a pre-specified arrival rate that depends on the queue length and processor sharing service. As such a queue is insensitive to the job size distribution [12], it has a unique fixed point of the form given in (5.34). It now suffices to argue that $\hat{\theta}_j = \hat{\pi}_j$, where $\hat{\pi}$ is the unique solution of (5.31).

As $\theta$ has the form given in (5.34), we can repeat the proof of the previous theorem to show that (5.33) holds with $\hat{\pi}$ replaced by $\hat{\theta}$ for any $i$ and $j$ and therefore also for $i = 0$, which means $\hat{\theta}$ is a fixed point of (5.31). The proof is then completed due to Proposition 5.7.3. □

## 5.8 Asymptotic Insensitivity

We now present our main result, having established global attraction of the fixed point $\pi$, the proof is quite standard. Let $\pi^{(N)}$ be the stationary measure associated to the Markov chain $X^{(N)}(t)$ that captures the fraction of the servers with $i+j$ or more jobs, where at least $j$ of these jobs are in phase 2 in a system with $N$ servers and initial state $X^{(N)}(0) \in \Omega_B$.

**Theorem 5.8.1.** *The limiting queue length distribution of the supermarket model with processor sharing service is insensitive to the job size distribution within the class of hyperexponential distributions of order* 2*, that is, the sequence $\pi^{(N)}$ converges weakly to the Dirac measure associated with the fixed point $\pi$. In other words, the limiting queue length distribution is given by the unique fixed point $\hat{\pi}$ of* (5.31).

*Proof.* By the Lipschitz continuity and Kurtz' theorem [17, Chapter 11] the sample paths $X^{(N)}(t)$ of the stochastic system consisting of $N$ servers converge in probability to the unique solution of the set of ODEs given by (5.1) with $f_{i,j}(h(t))$ specified by (5.27) over any finite time scale $(0, T]$, that is,

$$\lim_{N\to\infty} \sup_{t \le T} ||X^{(N)}(t) - h(t)|| = 0,$$

in probability if $\lim_{N\to\infty} X^{(N)}(0) = h_0$, where $h(0) = h_0$.

As $\Omega_B$ is compact the sequence of stationary measures $\pi^{(N)}$ is tight. Hence, Prokhorov's theorem implies that any subsequence of the sequence $\pi^{(N)}$ has a further subsequence that converges to some measure on $\Omega_B$. Now [25, Theorem 4], implies that *any* limit point of the these further subsequences of $\pi^{(N)}$ has support on the compact closure of the set of accumulation points of the set of ODEs for all initial conditions $h_0 \in \Omega_B$. By Theorem 5.5.3 the only accumulation point is the fixed point $\pi$, which proves that all these further subsequences converge to the same limit point, being the Dirac measure $\delta_\pi$ of the fixed point $\pi$. This implies that the sequence of measures $\pi^{(N)}$ converges weakly to the Dirac measure $\delta_\pi$. □

An interesting question at this stage is whether this result can be generalized easily to hyperexponential distributions of order $r > 2$. We now argue that this does not appear to be the case. Assume we have 3 phases, then the state would be captured by the variables $h_{i,j,k}(t)$ that represent the fraction of the servers with at least $i + j + k$ jobs, of which at least $j + k$ are in phase 2 or 3 and at least $k$ are in phase 3 at time $t$. Similarly, define $w_{i,j,k}(t)$ as the fraction of the servers with exactly $i$ jobs in phase 1, $j$ in phase 2 and $k$ in phase 3. Now consider the state $h(0)$ where $w_{0,1,0}(t) = 1$, meaning all servers contain 1 job and this job is in phase 2. Further, consider the state $\tilde{h}(t)$ with $\tilde{w}_{99,1,0}(t) = 1$, meaning all servers contain 100 jobs, 99 in phase 1 and 1 in phase 2. If we generalize the partial order $\leq_C$ presented in this chapter in the obvious manner, then clearly $h(0) \leq_C \tilde{h}(0)$. However, $h(\epsilon) \leq_C \tilde{h}(\epsilon)$ may not hold for $\epsilon$ small, meaning the system does not appear to be monotone. To understand this, note that from state $h(0)$ servers are created that contain at least 1 job in phase 3 at a rate $\mu_2 p_2$ as the full server capacity is devoted to a single job in phase 2 in state $h(0)$, while from state $\tilde{h}(0)$ jobs in phase 3 are only created at a rate $\mu_2 p_2/100$, thus $h_{0,0,1}(\epsilon)$ will exceed $\tilde{h}_{0,0,1}(\epsilon)$ for $\epsilon$ small enough.

## 5.9 Traditional push

In this section we illustrate that Theorem 5.5.3 can also be used to prove asymptotic insensitivity of other systems with PS servers. More specifically we focus on the traditional push strategy studied for FCFS servers in [16] and [55, Section VI.A]. We will argue that Assumptions 5.1 to 5.3 hold for this strategy in case of PS servers, which allows us to establish asymptotic insensitivity within the class of order-2 hyperexponential distributions by using the same arguments as in the proof of Theorem 5.8.1.

We consider a system consisting of $N$ servers, each subject to its own local Poisson arrival process with rate $\lambda < 1$. When a job arrives in server $n$ and server $n$ is busy, a single random server $n'$ is probed and the incoming job is immediately transferred to server $n'$ provided that it is idle. Otherwise the job is executed on server $n$. Note that although the servers are PS servers, a job is fully executed on a single server under this strategy. This is in contrast to the traditional pull or the rate-based strategies in [55], where transferred jobs are always partially executed on one server before being transferred to another PS server. In fact, such partial executions imply that asymptotic insensitive is lost despite the PS service discipline.

We first define the drift terms $f_{i,j}(h)$ corresponding to job arrivals and transfers for the traditional push strategy. As jobs always start service in phase 1, we have $f_{0,j}(h) = 0$ as in the supermarket model. When $i > 0$ and $i + j > 1$, then $h_{i-1,j} - h_{i,j}$ is the fraction of the queues with an exact queue length of $i + j - 1$ with at least $j$ jobs in phase 2. Arrivals in such a queue increase the queue length to $i + j$ provided that the probed server is busy, which occurs with probability $h_{1,0}$. Hence, $\lambda(h_{i-1,j} - h_{i,j})h_{1,0}$ is the rate at which $h_{i,j}$ increases due to arrivals that are not transferred (for $i > 0$ and $i + j > 1$). When $i = 1$ and $j = 0$, $h_{i,j} = h_{1,0}$ is the fraction of busy servers and this fraction increases at rate $\lambda(1 - h_{1,0})$, due to local arrivals in idle servers, plus $\lambda h_{1,0}(1 - h_{1,0})$, due to arrivals in busy servers that are immediately transferred to an idle server. Hence, we have

$$\begin{aligned} f_{i,j}(h) &= 1[i > 0, i + j > 1]\lambda(h_{i-1,j} - h_{i,j})h_{1,0} \\ &\quad + 1[i = 1, j = 0]\lambda((1 - h_{1,0}) + h_{1,0}(1 - h_{1,0})) \\ &= 1[i > 0]\lambda(h_{i-1,j} - h_{i,j})h_{1,0} + 1[i = 1, j = 0]\lambda(1 - h_{1,0}). \end{aligned} \tag{5.35}$$

In particular, we have

$$f_{1,0}(h) = \lambda(1 - (h_{1,0})^2). \tag{5.36}$$

Define $\Xi_{i,j}(h(t)) = \frac{d}{dt}h_{i,j}(t)$ and $\Xi(h) = \left[\Xi_{i,j}(h)\right]_{i,j=0}^{\infty}$. The next two results show that Assumptions 5.1 and 5.2 hold.

**Proposition 5.9.1.** *The drift $\Xi$ is Lipschitz continuous on $\Omega_B$, meaning Assumption 5.1 is met when $f_{i,j}(h)$ is defined as in* (5.35).

*Proof.* Let $h, \tilde{h} \in \Omega_B$. Proceeding similarly to Proposition 5.7.1, we get

$$\mathbf{d}(\Xi(h), \Xi(\tilde{h})) \leq 4(B+1)(\mu_1 + 2\mu_2)\mathbf{d}(h, \tilde{h}) + \sup_{i,j=0}^{\infty} \left| f_{i,j}(h) - f_{i,j}(\tilde{h}) \right|.$$

We have

$$\sup_{i,j=0}^{\infty} \left| f_{i,j}(h) - f_{i,j}(\tilde{h}) \right| \leq \lambda \mathbf{d}(h, \tilde{h}) + \lambda \sup_{i,j=0}^{\infty} \left| (h_{i-1,j} - h_{i,j})h_{1,0} - (\tilde{h}_{i-1,j} - \tilde{h}_{i,j})\tilde{h}_{1,0} \right|. \tag{5.37}$$

We now use the inequality $|a_1a_2 - b_1b_2| \leq |a_1 - b_1| + |a_2 - b_2|$, for $0 \leq a_1, a_2, b_1, b_2 \leq 1$ on (5.37), to find that

$$\sup_{i,j=0}^{\infty} \left| (h_{i-1,j} - h_{i,j})h_{1,0} - (\tilde{h}_{i-1,j} - \tilde{h}_{i,j})\tilde{h}_{1,0} \right| \leq \left| h_{1,0} - \tilde{h}_{1,0} \right| + 2\mathbf{d}(h, \tilde{h}) \leq 3\mathbf{d}(h, \tilde{h}).$$

To conclude, we have

$$\mathbf{d}(\Xi(h), \Xi(\tilde{h})) \leq (4(B+1)(\mu_1 + 2\mu_2) + 4\lambda)\, \mathbf{d}(h, \tilde{h}).$$

□

**Proposition 5.9.2.** *Assumption 5.2 holds for the system of ODEs* (5.1) *with $f_{i,j}(h)$ specified by* (5.35).

*Proof.* By (5.36), $F_1^1(h)$ is only decreasing in $g_1^1 = h_{1,0}$. So we may assume that the indices $\{j, i_1, \ldots, i_s\}$ are different from $j = 1, i_1 = 1$. We then have

$$\begin{aligned} F_{i_1,\ldots,i_s}^{j}(h) &= f_{0,j}(h) + \sum_{k=1}^{s} (f_{i_k,j-k}(h) - f_{i_k-1,j-k+1}(h)) \\ &= \lambda h_{1,0} \sum_{k=1}^{s} w_{i_k-1,j-k}(h). \end{aligned}$$

The arrivals can be visualised in the same way as those for the supermarket model (see Figure 5.6). If $s = 0$ then there is nothing to show, so suppose $s \geq 1$. We need to check two cases: $i_1 = 1$ and $i_1 > 1$. Set $c_k = \lambda h_{1,0} = \lambda g_1^1(h)$ for $k = 1, \ldots, s$ and $c_0 = c_{s+1} = 0$. Suppose first $i_1 > 1$. By using Lemma 5.4.6, we get

$$\begin{aligned} F_{i_1,\ldots,i_s}^{j}(h) &= \sum_{k=1}^{s} c_k w_{i_k-1,j-k}(h) \\ &= g_{i_1-1,\ldots,i_s-1}^{j}(h)c_s - \sum_{k=0}^{s-1} g_{i_1-1,\ldots,i_k-1,i_{k+1},\ldots,i_s}^{j}(h)\Big(c_{k+1} - c_k\Big) \end{aligned}$$

$$= \lambda g_1^1(h)(g^j_{i_1-1,\dots,i_s-1}(h) - g^j_{i_1,\dots,i_s}(h)).$$

If $h, \tilde{h} \in \Omega_B$ such that $h \leq_C \tilde{h}$ and $g^j_{i_1,\dots,i_s}(h) = g^j_{i_1,\dots,i_s}(\tilde{h})$, then $g_1^1(h) \leq g_1^1(\tilde{h})$ and $g^j_{i_1-1,\dots,i_s-1}(h) \leq g^j_{i_1-1,\dots,i_s-1}(\tilde{h})$ such that

$$\lambda g_1^1(h)(g^j_{i_1-1,\dots,i_s-1}(h) - g^j_{i_1,\dots,i_s}(h)) \leq \lambda g_1^1(\tilde{h})(g^j_{i_1-1,\dots,i_s-1}(\tilde{h}) - g^j_{i_1,\dots,i_s}(\tilde{h})).$$

Suppose now $i_1 = 1$. Proceeding similarly as in Proposition 5.7.2, we get

$$\begin{aligned}
F^j_{i_1,\dots,i_s}(h) &= \sum_{k=1}^{s} c_k w_{i_k-1,j-k}(h) \\
&= -c_1 g^j_{i_1,\dots,i_s}(h) + \sum_{k=1}^{s} g^{j-1}_{i_2-1,\dots,i_k-1,i_{k+1},\dots,i_s}(h)\,(c_k - c_{k+1}) \\
&= \lambda g_1^1(h)(g^{j-1}_{i_2-1,\dots,i_s-1}(h) - g^j_{i_1,\dots,i_s}(h)).
\end{aligned}$$

If $h, \tilde{h} \in \Omega_B$ such that $h \leq_C \tilde{h}$ and $g^j_{i_1,\dots,i_s}(h) = g^j_{i_1,\dots,i_s}(\tilde{h})$, then $g_1^1(h) \leq g_1^1(\tilde{h})$ and $g^{j-1}_{i_2-1,\dots,i_s-1}(h) \leq g^{j-1}_{i_2-1,\dots,i_s-1}(\tilde{h})$ such that

$$\lambda g_1^1(h)(g^{j-1}_{i_2-1,\dots,i_s-1}(h) - g^j_{i_1,\dots,i_s}(h)) \leq \lambda g_1^1(\tilde{h})(g^{j-1}_{i_2-1,\dots,i_s-1}(\tilde{h}) - g^j_{i_1,\dots,i_s}(\tilde{h})).$$

This finishes the proof. □

We now proceed by arguing that we have a unique fixed point in $\Omega_B$. As $h_{i,j} = \sum_{k\geq i+j}\sum_{\ell=j}^{k} w_{k-\ell,\ell}$, finding a fixed point of (5.1) is equivalent to finding a fixed point $\pi^w$ of the corresponding set of equations for $\frac{d}{dt}w_{i,j}(h(t))$. As there is no ambiguity here, we denote $w(h(t))$ simply as $w(t)$. For the traditional push we have, similar to (5.30),

$$\begin{aligned}
\frac{d}{dt}w_{i,j}(t) &= 1[i \geq 1]\lambda(w_{i-1,j}(t) - w_{i,j}(t))(1 - w_{0,0}(t)) + 1[i = 1, j = 0]\lambda w_{0,0}(t) \\
&\quad - 1[i = j = 0]\lambda w_{0,0}(t) - 1[i = 0]\lambda w_{0,j}(t)(1 - w_{0,0}(t)) \\
&\quad + 1[j \geq 1]p_1\mu_1 w_{i+1,j-1}(t)\frac{i+1}{i+j} - \mu_1 w_{i,j}(t)\frac{i}{i+j} - \mu_2 w_{i,j}(t)\frac{j}{i+j} \\
&\quad + \mu_2 w_{i,j+1}(t)\frac{j+1}{i+j+1} + (1-p_1)\mu_1 w_{i+1,j}(t)\frac{i+1}{i+j+1},
\end{aligned} \tag{5.38}$$

for $i, j \geq 0$ and $i + j \leq B$, where $w_{i,j}(t) = 0$ for $i + j > B$.

We first consider the set of ODEs for the traditional push strategy in case of exponential job sizes (which has the same form as in [55, Section VI.A] for FCFS servers). Let $k_i(t)$ be the fraction of servers with $i$ or more jobs at time $t$, then

$$\frac{d}{dt}k_i(t) = \lambda(k_{i-1}(t) - k_i(t))k_1(t) - (k_i(t) - k_{i+1}(t)) + 1[i = 1]\lambda(1 - k_1(t)), \tag{5.39}$$

for $i = 1, \dots, B$ and set $k_0(t) = 1$ and $k_i(t) = 0$, for $i > B$.

**Proposition 5.9.3.** *The set of ODEs given by* (5.39) *has a unique fixed point $\hat{\pi}$. Further $\hat{\pi}_i \in (0, \lambda^{2i-1})$ and $\hat{\pi}_i > \hat{\pi}_{i+1}$.*

*Proof.* Summing the equations in (5.39) from $i = k$ to $B$ yields that any fixed point $\hat{\pi}$ satisfies

$$\hat{\pi}_k = \frac{\lambda^2}{1+\lambda\hat{\pi}_B}(\hat{\pi}_{k-1} - \hat{\pi}_B),$$

for $k = 2, \ldots, B$ and $\hat{\pi}_1 = \lambda/(1+\lambda\hat{\pi}_B)$. Define $V_1(x) = \lambda/(1+\lambda x)$ and $V_k(x) = \frac{\lambda^2}{1+\lambda x}(V_{k-1}(x) - x)$ for $k = 2, \ldots, B$, then $\hat{\pi}_k = V_k(\hat{\pi}_B)$. It is easy to see that $V_k(0) = \lambda^{2k-1} > 0$. By noting that $(V_{i+1}(x) - V_i(x)) = \frac{\lambda^2}{1+\lambda x}(V_i(x) - V_{i-1}(x))$, we find,

$$\begin{aligned} V_k(x) &= \sum_{i=1}^{k-1}(V_{i+1}(x) - V_i(x)) + V_1(x) = \sum_{i=1}^{k-1}\left(\frac{\lambda^2}{1+\lambda x}\right)^{i-1}(V_2(x) - V_1(x)) + V_1(x) \\ &= -\underbrace{\frac{1-\left(\frac{\lambda^2}{1+\lambda x}\right)^{k-1}}{1-\frac{\lambda^2}{1+\lambda x}}}_{>0}\underbrace{\frac{\lambda(\lambda^2x^2 - \lambda^2 + 2\lambda x + 1)}{(1+\lambda x)^2}}_{>0} + \frac{\lambda}{1+\lambda x}, \end{aligned}$$

for $x \in [0,1]$. Hence, $V_k(x) < V_{k-1}(x)$ for $k = 2, \ldots, B$ and $x \in [0,1]$. One easily checks that $V_2'(x) < 0$ on $[0,1]$ and $V_2(1) = -\lambda^2/(1+\lambda)^2 < 0$, meaning $V_2(x)$ has a unique root $\xi_2$ on $[0,1]$. We now complete the proof by showing by induction that $V_k(x)$ is decreasing on $[0, \xi_{k-1}]$ and $V_k(x) < 0$ on $[\xi_{k-1}, 1]$, which implies that $V_k(x)$ has a unique root $\xi_k < \xi_{k-1}$ on $[0,1]$ (as $V_k(0) = \lambda^{2k-1}$) and $V_k(x)$ is negative on $(\xi_k, 1]$.

By definition of $V_k(x)$, we have

$$V_k'(x) = \frac{\lambda^2}{1+\lambda x}V_{k-1}'(x) - \frac{\lambda^3}{(1+\lambda x)^2}V_{k-1}(x) - \frac{\lambda^2}{(1+\lambda x)^2}.$$

Therefore, $V_k'(x)$ is negative if $V_{k-1}'(x) \leq 0$ and $V_{k-1}(x) \geq 0$. By induction this is the case on $[0, \xi_{k-1}]$, meaning $V_k(x)$ is decreasing on $[0, \xi_{k-1}]$ and negative on $[\xi_{k-1}, 1]$ as $V_k(x) < V_{k-1}(x)$. □

**Proposition 5.9.4.** *Let $\hat{\pi}$ be the unique fixed point of* (5.39) *and define $\pi^w$ as in* (5.32)*, then $\pi^w$ is a fixed point of* (5.38).

*Proof.* The proof is nearly identical to the proof of Proposition 5.7.4, that is, by replacing $w_{i,j}(t)$ by $\pi^w_{i,j}$ in (5.38) with the left hand side set equal to zero, one obtains

$$\begin{aligned} 0 &= \left(\lambda(\hat{\pi}_{i+j-1} - \hat{\pi}_{i+j})\hat{\pi}_1 - (\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1}) + 1[i+j=1]\lambda(1-\hat{\pi}_1)\right)\frac{\mu_1 i}{i+j} \\ &\quad - \left(\lambda(\hat{\pi}_{i+j} - \hat{\pi}_{i+j+1})\hat{\pi}_1 - (\hat{\pi}_{i+j+1} - \hat{\pi}_{i+j+2}) + 1[i+j=0]\lambda(1-\hat{\pi}_1)\right). \end{aligned} \tag{5.40}$$

for $i \geq 0$, which holds as $\hat{\pi}$ is the unique fixed point of (5.39). □

**Proposition 5.9.5.** *$\pi^w$ is the unique fixed point of* (5.38) *and therefore Assumption 5.3 holds for the set of ODEs in* (5.1) *with $f_{i,j}(h)$ specified by* (5.35).

*Proof.* We can repeat the same arguments as in the proof of Proposition 5.7.5, except that we define $\lambda_{i+j} = \lambda(1-\theta_{0,0})$ for $i+j > 0$, $\lambda_0 = \lambda + \lambda(1-\theta_{0,0})$ and rely on Proposition 5.9.3. Note that the $M/PH/1$ queue with pre-specified arrival rates has arrival rate $\lambda_0$ when the queue is empty and $\lambda_1$ when the queue is busy. □

Having established Assumptions 5.1 to 5.3, global attraction of the unique fixed point follows from Theorem 5.5.3 and asymptotic insensitivity within the class of hyperexponential distributions of order 2 follows for the traditional push strategy by repeating the arguments of in the proof of Theorem 5.8.1.

## 5.10 Conclusions

In this chapter we established an asymptotic insensitivity result for the supermarket model with processor sharing servers for the class of hyperexponential distributions of order 2. To the best of our knowledge, it is the first result of its kind for systems with PS service. More specifically, we showed that the weak limit of the stationary distributions as the number of servers tends to infinity is given by the Dirac measure of a fixed point, the queue length distribution of which is insensitive to the job size distribution. The main step in proving this result is showing that the set of ODEs describing the evolution of the mean field limit, has a global attractor. We also demonstrated, using the traditional push strategy in distributed systems, that our results can be of use to prove asymptotic insensitivity results beyond the supermarket model.

Chapter 6

# Global attraction of ODE-based mean field models of heterogeneous systems with hyperexponential job sizes

*The first results I obtained in my research were a generalization of the results in* [76, Sections 3-8]. *These account for about half of the results presented in this chapter. The rest of the chapter's results I have found towards the the end of my PhD. This means that this chapter contains both some of the first and some of the last results I discovered for my thesis.*

## 6.1 Introduction

Mean field modelling is a popular tool in studying large scale systems. The tool has been used in systems with job stealing/sharing in homogeneous [55,76,77] and heterogeneous [25,72,79,80] setting, load balancing algorithms [2,58,76,80,82] and for other problems (see [76, Section 1]).

In particular, in [76] the mean field models of a family of homogeneous systems with hyperexponential job sizes were studied. Using monotonicity arguments, it was shown that under several assumptions the set of ODEs describing such mean field model has a unique fixed point which is a global attractor. It was further proven that these assumptions hold for several policies, including JSQ($d$) and pull and push strategies. In [76, Section 9], the author shared his belief that the same arguments could be leveraged in case of heterogeneous queues. In this chapter we generalize the results and the assumptions from [76, Sections 3-8] to heterogeneous systems, thus confirming the author's belief. As many of the proofs are analogous to those from [76], we omit some of them and instead provide intuition or describe the idea behind omitted proofs. This generalization is the main contribution of this chapter.

As examples, we show that the assumptions hold for JSQ($d$) policy and a policy which we call split-JSQ($d$). The latter policy works as follows: the central dispatcher first determines to which type of server a job is assigned and then uses power-of-$d$ policy on the servers of that type. This policy was studied in [80] for jobs of the phase type, however global attraction was not proven there.

The rest of the chapter is structured as follows. In Section 6.2, we describe the system and the job size distribution. We model the system using a set of ODEs as the number of queues $N$ tends to infinity in Section 6.3. The state space of this set of ODEs and a partial order on the states are introduced in Section 6.4. Using this partial order, global attraction is proven (under certain assumptions) in Section 6.5. In Sections 6.6 and 6.7, we present two examples of load balancing policies for which these assumptions hold, respectively the JSQ($d$) and the split-JSQ($d$) policies. Finally, in Section 6.8 we make some concluding remarks.

## 6.2 System description

### 6.2.1 The queues

The system consists of $N$ queues, each comprising of a server and an infinite buffer. Note, that from the practical point of view, there is little difference between having an infinite buffer and a huge finite one. All queues handle jobs according to the FCFS policy. When an idle queue receives a job, then that job immediately enters service. Upon finishing the job, the next job enters service immediately (if available). We suppose there are $1 \leq T < \infty$ types of servers and we denote by $q_k$ the fraction of queues with server of type $k$, for $k = 1, \ldots, T$. Thus $\sum_{k=1}^{T} q_k = 1$ and we suppose that $q_k > 0$ for every $k = 1, \ldots, T$. We denote by $s_k$ the "strength" of a server of type $k$, that is the amount of work per time unit that a type-$k$ server can process. We assume for every $k$ that $s_k > 0$ and that the average strength of the servers is 1, i.e. $\sum_{k=1}^{T} q_k s_k = 1$. Note, that if $s_k = s_\ell$ for every $k, \ell = 1, \ldots, T$ or if $T = 1$, then the queues are homogeneous. We shall henceforth refer to queues that have servers of type $k$ as queues of type $k$.

### 6.2.2 The distribution of the job sizes

Service requirements of jobs are distributed according to a distribution from the class $C_0$. $C_0$ consists of Coxian distributions with some restrictions. We first present a definition of a Coxian distribution and then introduce the class $C_0$ below.

A phase type distribution with parameters $(\alpha, S)$ and $n$ phases is called Coxian if $\alpha = [1, 0, \ldots, 0]$ and

$$S = \begin{bmatrix} -\mu_1 & p_1\mu_1 & & & \\ & -\mu_2 & p_2\mu_2 & & \\ & & \ddots & \ddots & \\ & & & -\mu_{n-1} & p_{n-1}\mu_{n-1} \\ & & & & -\mu_n \end{bmatrix},$$

with $\mu_i \geq 0$, for every $i = 1, \ldots, n$ and with $0 < p_i \leq 1$ for every $i = 1, \ldots, n-1$. We set $p_n = 0$ and denote $\nu_i = \mu_i(1 - p_i)$ for $i = 1, \ldots, n$. We say that a Coxian distribution belongs to the class $C_0$ if and only if $\nu_i = \mu_i(1 - p_i)$ is decreasing in $i$. Note, that for a distribution in $C_0$, we must have $p_i \neq 1$ for $i = 1, \ldots, n-1$ due to $\mu_n > 0$ and $p_n = 0$. Note further, that $C_0$ contains all hyperexponential distributions [76, Theorem 1.]. We assume that jobs have mean 1, that is $\alpha(-S)^{-1}1_{n_p} = 1$, or equivalently:

$$\sum_{j=1}^{n} \frac{1}{\mu_j} \prod_{s=1}^{j-1} p_s = 1. \tag{6.1}$$

Let $R_{i,k}$ be the expected remaining time of a job in phase $i$ in a server of type $k$. Clearly, $R_{n,k} = \frac{1}{s_k\mu_n}$ and $R_{i-1,k} = \frac{1}{s_k\mu_{i-1}} + p_{i-1}R_{i,k}$ for $i = 2, \ldots, n$. As we assume that the mean service requirement of jobs equals 1, we must have $R_{1,k} = \frac{1}{s_k}$ (as all jobs start in phase 1). We thus have

$$s_k p_{i-1}\mu_{i-1}R_{i,k} = s_k\mu_{i-1}R_{i-1,k} - 1. \tag{6.2}$$

**Lemma 6.2.1.** *If $\mu_i(1-p_i)$ is decreasing, we have $R_{i,k} > R_{i-1,k}$ for $i = 2, \ldots, n$.*

*Proof.* The proof is analogous to that of [76, Lemma 2.]. From the definition of $R_{i,k}$ we get

$$R_{i,k} = \frac{1}{s_k\mu_i} + \sum_{j=i}^{n-1}\left(\prod_{k=i}^{j} p_k\right)\frac{1}{s_k\mu_{j+1}},$$

as $\prod_{k=i}^{j} p_k$ is the probability that a job in phase $i$ will reach phase $j+1$. This implies

$$R_{i,k} - R_{i-1,k} = \frac{1}{s_k}\left[\frac{1-p_{i-1}}{\mu_i} + \sum_{j=i}^{n-1}\left(\prod_{k=i}^{j} p_k\right)\frac{1-p_{i-1}}{\mu_{j+1}} - \frac{1}{\mu_{j-1}}\right]. \tag{6.3}$$

Hence, it suffices to show that RHS of Equation (6.3) is greater than 0. This can be done analogously to [76, Appendix A]. □

We still have not specified how jobs arrive and are distributed among the servers, and if jobs can be transferred between servers. This is because we wish that the arguments in the following sections apply to a large family of policies. In Sections 6.6 and 6.7 we will present examples of these policies.

## 6.3 The form of the ODEs

We wish to define a set of ODEs to model the system as $N \to \infty$. To this end we define the variables $h_{\ell,i,k}(t)$ as the fraction of servers at time $t$ with at least $\ell$ jobs in the queue, in phase $\geq i$ and of type $k$.

For ease of notation, we set $h_{\ell,n+1,k}(t) = 0$, $h_{0,1,k}(t) = q_k$ and, for $i > 1$, $h_{0,i,k}(t) = h_{1,i,k}(t)$. Then $h_{\ell,i,k}(t) - h_{\ell,i+1,k}(t)$ denotes the fraction of queues with at least $\ell$ jobs, job in service in phase exactly $i$ (and of type $k$) and $h_{\ell,i,k}(t) - h_{\ell+1,i,k}(t)$ the fraction of queues with exactly $\ell$ jobs, with job in service in phase $\geq i$ (and of type $k$). Similarly,

$$(h_{\ell,i,k}(t) - h_{\ell,i+1,k}(t)) - (h_{\ell+1,i,k}(t) - h_{\ell+1,i+1,k}(t))$$

is the fraction of queues with exactly $\ell$ jobs, leading job in phase $i$ and of server type $k$. We also introduce the variables $f_{\ell,i,k}(h(t))$. These capture all events that are not phase changes or service completions, e.g. job arrivals, transfers,...

By using the variables $h_{\ell,i,k}(t)$ and $f_{\ell,i,k}(h(t))$ we can describe the evolution of the system using a set of ODEs. For the case of exponential job sizes we get:

$$\frac{d}{dt}h_{\ell,1,k}(t) = f_{\ell,1,k}(h(t)) - s_k(h_{\ell,1,k}(t) - h_{\ell+1,1,k}(t)). \tag{6.4}$$

For the case of $C_0$ distributions we make some remarks analogous to the homogeneous case. Service completion in a queue of length $\geq \ell$ always decreases $h_{\ell,i,k}(t)$ for $i \geq 2$, as the next job starts service in phase 1, while a completion decreases $h_{\ell,1,k}(t)$ if the queue length is exactly $\ell$. The set of ODEs from (6.4) thus generalizes as:

$$\begin{aligned}\frac{d}{dt}h_{\ell,1,k}(t) &= f_{\ell,1,k}(h(t)) \\ &\quad - s_k\sum_{j=1}^{n}[(h_{\ell,j,k}(t) - h_{\ell,j+1,k}(t)) - (h_{\ell+1,j,k}(t) - h_{\ell+1,j+1,k}(t))]\nu_j \end{aligned} \tag{6.5}$$

and for $i = 2, \dots, n$ as

$$\begin{aligned}\frac{d}{dt}h_{\ell,i,k}(t) &= 1[\ell > 1]f_{\ell,i,k}(h(t)) + s_k(h_{\ell,i-1,k}(t) - h_{\ell,i,k}(t))p_{i-1}\mu_{i-1} \\ &\quad - s_k\sum_{j=i}^{n}(h_{\ell,j,k}(t) - h_{\ell,j+1,k}(t))\nu_j, \end{aligned} \tag{6.6}$$

where for a statement $P$, $1[P]$ is one if $P$ is true and 0 otherwise. Note that the only way that queues of type $\tilde{k}$ can influence those of type $k$, for $k \neq \tilde{k}$, is through $f_{\cdot,\cdot,k}(h(t))$.

## 6.4 State space and partial order

For the non-homogeneous exponential case we have the state space:

$$\Omega^{nh}_{expo} = \{(h_{\ell,1,k})^{k\in\{1,\dots,T\}}_{\ell>0} | \forall k \in \{1,\dots,T\} : 0 \leq h_{\ell,1,k} \leq q_k, h_{\ell+1,1,k} \leq h_{\ell,1,k}, \sum_{\ell} h_{\ell,1,k} < \infty\}.$$

A partial order relation on $\Omega^{nh}_{expo}$ is given by the componentwise order. For the case of $C_0$ distributed job sizes we define:

$$\begin{aligned}\Omega^{nh} &= \{(h_{\ell,1,k})^{k\in\{1,\dots,T\}}_{\ell>0,i\in\{1,\dots,n\}} | \forall k \in \{1,\dots,T\} : 0 \leq h_{\ell,i,k} \leq q_k, h_{\ell,i+1,k} \leq h_{\ell,i,k}, \\ & h_{\ell+1,i,k} \leq h_{\ell,i,k}, h_{\ell,i,k} + h_{\ell+1,i+1,k} \geq h_{\ell+1,i,k} + h_{\ell,i+1,k}, \sum_{\ell} h_{\ell,1,k} < \infty\}. \end{aligned} \tag{6.7}$$

The first three conditions of (6.7) are obvious. The fourth condition comes from the inequality $h_{\ell,i,k} - h_{\ell+1,i,k} \geq h_{\ell,i+1,k} - h_{\ell+1,i+1,k}$. This inequality states that the fraction of queues with queue length $\ell$ with leading job in phase $\geq i$ (and of type $k$) should be equal or greater to the fraction of queues with queue length $\ell$ with leading job in phase $\geq i + 1$ (and of type $k$). The last condition says that the system should be stable.

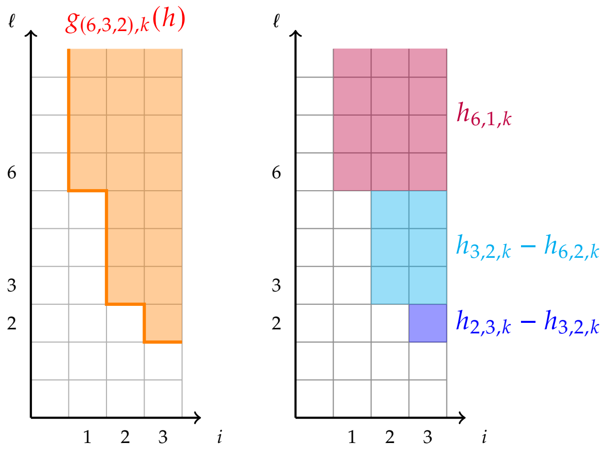


Figure 6.1: Illustration of condition (6.9) in terms of variables $g_{(\ell_1,\ldots,\ell_n),k}(h)$.

**Definition 6.4.1** (Partial order $\leq_C$)**.** Let $h, \tilde{h} \in \Omega^{nh}$. We state that $h \leq_C \tilde{h}$ if and only if

$$h_{\ell,i,k} \leq \tilde{h}_{\ell,i,k}, \tag{6.8}$$

for all $\ell, i, k$, and

$$h_{\ell_1,1,k} + \sum_{i=2}^{n}(h_{\ell_i,i,k} - h_{\ell_{i-1},i,k}) \leq \tilde{h}_{\ell_1,1,k} + \sum_{i=2}^{n}(\tilde{h}_{\ell_i,i,k} - \tilde{h}_{\ell_{i-1},i,k}), \tag{6.9}$$

for all $k$ and any set $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_n \geq 1$ with $\ell_1 > \ell_n$ and $\ell_1, \ldots, \ell_n \in \mathbb{N} \cup \{\infty\}$.

**Definition 6.4.2.** For any $\ell_1, \ldots, \ell_n \in \mathbb{N} \cup \{\infty\}$ with $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_n \geq 1$, $\ell_1 > \ell_n$ and any $k \in \{1, \ldots, T\}$, define $g_{(\ell_1,\ldots,\ell_n),k}$ and $f_{(\ell_1,\ldots,\ell_n),k}$ as a function from $\Omega^{nh}$ to $\mathbb{R}$ such that

$$g_{(\ell_1,\ldots,\ell_n),k}(h) = h_{\ell_1,1,k} + \sum_{i=2}^{n}(h_{\ell_i,i,k} - h_{\ell_{i-1},i,k}), \tag{6.10}$$

and

$$f_{(\ell_1,\ldots,\ell_n),k}(h) = f_{\ell_1,1,k}(h) + \sum_{i=2}^{n}(f_{\ell_i,i,k}(h) - f_{\ell_{i-1},i,k}(h)). \tag{6.11}$$

Condition (6.9) can thus be restated as $g_{(\ell_1,\ldots,\ell_n),k}(h) \leq g_{(\ell_1,\ldots,\ell_n),k}(\tilde{h})$. We illustrate (6.9) in Figure 6.1. We now provide some intuition behind the partial order.

*Remark* 6.4.3. Consider two sets $\mathcal{A}$ and $\tilde{\mathcal{A}}$ of $N$ queues and let $h$ and $\tilde{h}$ be their corresponding states in $\Omega_B$. The intuition behind the order $\leq_C$ is that it should be such that $h \leq_C \tilde{h}$ implies that there exists a mapping $m : \mathcal{A} \to \tilde{\mathcal{A}}$ such that every queue $a$ is mapped to a queue $m(a)$ of the same type with at least the same amount of jobs as $a$ and with the phase of the job in service in queue $m(a)$ being at least that of the job in service in queue $a$.

We illustrate the intuition in Figure 6.2. The Figure shows two systems with $N = 4$ and $T = 1$, where every queue is depicted by a numbered, coloured ball. The yellow

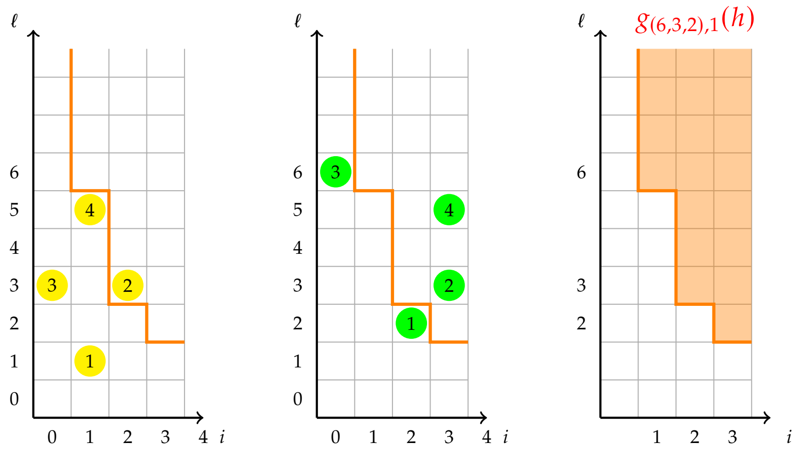


Figure 6.2: Illustration of the intuition from Remark 6.4.3, with $N = 4$ and $T = 1$. The yellow system is dominated by the green one.

system is clearly dominated by the green one ($m$ can be chosen to map the $k$-th queue from the yellow system to the $k$-th queue from the green one). In terms of the variables $g_{(\ell_1,\ldots,\ell_n),k}(h)$, the dominance can be seen as follows: for every set of integers $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_n \geq 1$ with $\ell_1 > \ell_n$, if we draw the border of of $g_{(\ell_1,\ldots,\ell_n),k}(h)$ and compare the number of coloured balls to the right of the border, then the number of green balls should be at least as high as the number of yellow balls, for every $k = 1, \ldots, T$. In Figure 6.2 this is shown for $g_{(6,3,2),1}(h)$ and $T = 1$.

**Proposition 6.4.4.** *For every fixed point $\pi$ of* (6.5)-(6.6) *we have $\forall k \in \{1, \ldots, T\}$:*

$$\pi_{1,i,k} = \pi_{1,1,k} \sum_{j=i}^{n} \frac{1}{\mu_j} \prod_{s=1}^{j-1} p_s, \tag{6.12}$$

*for $i = 1, \ldots, n$, where $\mu_i$ and $p_s$ are the parameters of the Coxian representation.*

*Proof.* This Proposition can be proven exactly as [76, Proposition 5]. Instead we provide some intuition on equality (6.12). $\pi_{1,i,k}/\pi_{1,1,k}$ should be the fraction of time that a server of type $k$ works on a job in phase $\geq i$ provided that the server is busy. Note that we say "should be" instead of "is", as $\pi$ is a fixed point of set of ODEs and at this point we do not know whether it is an invariant distribution of the cavity queue corresponding to the system.

When a job enters phase $j$, it takes on average $1/s_k\mu_j$ time to leave it. The probability that a job reaches phase $j$ is given by $\prod_{s=1}^{j-1} p_s$. Therefore

$$\frac{1}{s_k} \sum_{j=i}^{n} \frac{1}{\mu_j} \prod_{s=1}^{j-1} p_s$$

is the average amount of time that a job spends in phases $\geq i$. This implies that the following relation should hold:

$$\frac{\pi_{1,i,k}}{\pi_{1,1,k}} = \frac{\frac{1}{s_k}\sum_{j=i}^{n}\frac{1}{\mu_j}\prod_{s=1}^{j-1}p_s}{\frac{1}{s_k}\sum_{j=1}^{n}\frac{1}{\mu_j}\prod_{s=1}^{j-1}p_s}.$$

Due to (6.1) and $s_k > 0$, this simplifies to $\pi_{1,i,k}/\pi_{1,1,k} = \sum_{j=i}^{n}\frac{1}{\mu_j}\prod_{s=1}^{j-1}p_s$. □

## 6.5 Global attraction

In this section we prove that there exists a unique fixed point $\pi$ of the set of ODEs from (6.5)-(6.6) and that this fixed point is a global attractor. We do this under certain assumptions. In the next sections we consider certain policies for which we show that the assumptions hold. We make the following assumptions:

**Assumption 6.1.** *The functions $f_{\ell,i,k}(h) : \Omega^{nh} \to \mathbb{R}$ are such that for any $h \in \Omega^{nh}$, the set of ODEs given by* (6.5) - (6.6) *has a unique solution $h(t) : [0,\infty) \to \Omega^{nh}$ with $h(0) = h$ and there exists a fixed point $\pi$ in $\Omega^{nh}$.*

**Assumption 6.2.** *The functions $f_{\ell,i,k}(h) : \Omega^{nh} \to \mathbb{R}$ are non-decreasing in $h_{\ell',i',k'}$ for any $(\ell',i',k') \neq (\ell,i,k)$.*

**Assumption 6.3.** *The functions $f_{(\ell_1,\ldots,\ell_n),k}(h) : \Omega^{nh} \to \mathbb{R}$ are such that*

$$f_{(\ell_1,\ldots,\ell_n),k}(h) \leq f_{(\ell_1,\ldots,\ell_n),k}(\tilde{h})$$

*for any $h, \tilde{h} \in \Omega^{nh}$ with $h \leq_C \tilde{h}$ and $g_{(\ell_1,\ldots,\ell_n),k}(h) = g_{(\ell_1,\ldots,\ell_n),k}(\tilde{h})$.*

**Assumption 6.4.** *Suppose that for every $h \in \Omega^{nh}$ the set of ODEs given by* (6.5) - (6.6) *has a unique solution $h(t) : [0,\infty) \to \Omega^{nh}$ with $h(0) = h$. The first derivative of $h_{\ell,i,k}(t)$ exists and is bounded for every $(\ell,i,k)$ and every $h \in \Omega^{nh}$, where $h(0) = h$.*

Note, that assuming that $dh_{\ell,i,k}(t)/dt$ is bounded is equivalent to assuming that $f_{\ell,i,k}(h(t))$ is bounded.

**Assumption 6.5.** *The functions $f_{\ell,1,k}(h)$ are such that for any fixed point $\pi$ and $L \geq 1$ we have*

$$\sum_{k=1}^{T}\sum_{\ell\geq L}(f_{\ell,1,k}(h) - f_{\ell,1,k}(\pi)) = \sum_{k=1}^{T}\sum_{\ell=1}^{L-1}\sum_{j=1}^{n} b_{L,\ell,j,k}(h)(h_{\ell,j,k} - \pi_{\ell,j,k}) - a_{L,\pi}(h),$$

*for some bounded functions $b_{L,\ell,j,k}(h)$ on $\Omega^{nh}$ and functions $a_{L,\pi}(h)$ for which $a_{L,\pi}(h) \geq 0$ if $\pi \leq_C h$ and $a_{L,\pi}(h) \leq 0$ if $h \leq_C \pi$.*

Note that in Assumption 6.5, we have $a_{L,\pi}(\pi) = 0$ for any fixed point $\pi$.

The first step in proving global attraction is showing "monotonicity", that is: if at time $t$ a system state is dominated by another state w.r.t. the partial order $\leq_C$, it will stay dominated at time $t + s$, for every $s \geq 0$.

**Proposition 6.5.1** (Monotonicity). *Assume that Assumptions 6.1-6.3 hold and let $h, \tilde{h} \in \Omega^{nh}$. Let $h(t)$ and $\tilde{h}(t)$ be the unique solution of (6.5) - (6.6) with $h(0) = h$ and $\tilde{h}(0) = \tilde{h}$, respectively. If $v_i$ is decreasing in $i$ and $h \leq_C \tilde{h}$ then $h(t) \leq_C \tilde{h}(t)$ for any $t > 0$.*

*Proof.* The proof is analogous to [76, Proposition 6]. The idea of the proof is that if at some time $t$ we have $h_{\ell,i,k}(t) = \tilde{h}_{\ell,i,k}(t)$ for some $(\ell, i, k)$, then we need to show

$$\frac{d}{dt} h_{\ell,i,k}(t) \leq \frac{d}{dt} \tilde{h}_{\ell,i,k}(t).$$

Due to $h(t) \leq_C \tilde{h}(t)$ it suffices to show that $\frac{d}{dt} h_{\ell,i,k}(t)$ is non-decreasing in $h_{\ell',i',k'}(t)$ for all $(\ell', i', k') \neq (\ell, i, k)$. Once this is shown, we know that condition (6.8) stays true over time and using a similar idea we can prove that so does condition (6.9). □

**Proposition 6.5.2.** *Let $h \in \Omega^{nh}$ and assume $v_i$ is decreasing in $i$, then the trajectory $h(t)$ starting in $h(0) \in \Omega^{nh}$ at time 0 converges to $\pi$ provided that for any $\tilde{h} \in \Omega^{nh}$ with $\tilde{h} \leq_C \pi$ or $\pi \leq_C \tilde{h}$, $h(t)$ with $h(0) = \tilde{h}$ converges pointwise to $\pi$.*

*Proof.* The proof is completely analogous to the proof of [76, Proposition 7]. The idea is to define for every state $h \in \Omega^{nh}$ some states $h^d, h^u \in \Omega^{nh}$ such that $h^d \leq_C \pi$, $h^d \leq_C h$ and $\pi \leq_C h^u$, $h \leq_C h^u$. The claim then follows from Proposition 6.5.1. Usually, we can simply pick $h^d$ to be the zero state, while more care is needed in choosing $h^u$. □

Note, that proof of Proposition 6.5.2 uses that the inequality

$$g_{(\ell_1,\ldots,\ell_n),k}(h) = h_{\ell_1,1,k} + \sum_{i=2}^{n} (h_{\ell_i,i,k} - h_{\ell_{i-1},1,k}) \leq h_{\ell_n,1,k} \tag{6.13}$$

holds for any $k \in \{1, \ldots, T\}$ and $h \in \Omega^{nh}$. We illustrate this inequality in Figure 6.3 and note that it can be proven similarly to [76, (17)].

For the proof of the global attraction we need the following two lemmas.

**Lemma 6.5.3.** *Assume that Assumptions 6.1-6.4 hold and let $h \in \Omega^{nh}$. Let $h(t)$ be the unique solution of (6.5) - (6.6) with $h(0) = h$. Let $\pi$ be a fixed point of (6.5) - (6.6). Suppose that $v_i$ is decreasing in $i$. Let $L \geq 1$ and $j \in \{1, \ldots, n\}$ be arbitrary. If $\pi \leq_C h(0)$, then*

$$\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k (h_{L,j,k}(t) - \pi_{L,j,k}) dt < \infty$$

*implies that for every $k = 1, \ldots, T$, $\lim_{t\to\infty} h_{L,j,k}(t) = \pi_{L,j,k}$. If $h(0) \leq_C \pi$, then*

$$\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k (\pi_{L,j,k} - h_{L,j,k}(t)) dt < \infty$$

*implies that for every $k = 1, \ldots, T$, $\lim_{t\to\infty} h_{L,j,k}(t) = \pi_{L,j,k}$.*

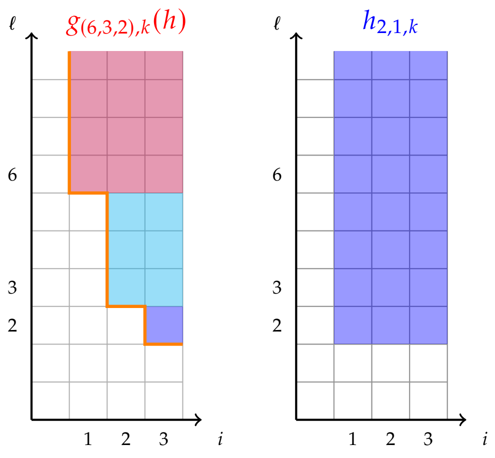


Figure 6.3: Illustration of Inequality (6.13). We clearly have $g_{(6,3,2),k}(h) \leq h_{2,1,k}$.

*Proof.* We prove the case where $\pi \leq_C h(0)$. The case where $h(0) \leq_C \pi$ can be proven analogously. So suppose $\pi \leq_C h(0)$. Assume that $\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k(h_{L,j,k}(t) - \pi_{L,j,k})dt < \infty$ is true. Due to Proposition 6.5.2, we have for every $t \geq 0$ that $\pi \leq_C h(t)$, which implies for every $k \in \{1, \dots, T\}$ that $\pi_{L,j,k} \leq h_{L,j,k}(t)$. As $s_k > 0$, we have for every $k$ that $0 \leq \int_{t=0}^{\infty}(h_{L,j,k}(t) - \pi_{L,j,k})dt < \infty$. Due to Assumption 6.4, $dh_{L,j,k}(t)/dt$ exists and is bounded. Similarly to the proof of [82, Lemma 7.(c)], we now get for every $k$ that $h_{L,j,k}(t) \longrightarrow \pi_{L,j,k}$ as needed. □

**Lemma 6.5.4.** *Define* $z_{1,L}(h(t)) = \sum_{k=1}^{T} \sum_{\ell \geq L} h_{\ell,1,k}(t)$ *and*

$$z_2(h(t)) = \sum_{k=1}^{T} s_k \sum_{i=2}^{n} h_{1,i,k}(t)(R_{i,k} - R_{i-1,k}).$$

*Then*

$$\frac{d}{dt} z_{1,L}(h(t)) = \sum_{k=1}^{T} \sum_{\ell \geq L} f_{\ell,1,k}(h(t)) - \sum_{k=1}^{T} s_k \sum_{j=1}^{n} (h_{L,j,k}(t) - h_{L,j+1,k}(t))\nu_j \tag{6.14}$$

*and*

$$\frac{d}{dt} z_2(h(t)) = -\sum_{k=1}^{T} s_k h_{1,1,k}(t) + \sum_{k=1}^{T} s_k \sum_{j=1}^{n} (h_{1,j,k}(t) - h_{1,j+1,k}(t))\nu_j. \tag{6.15}$$

*Proof.* The proof is analogous to the one of [76, Lemma 3]. The expression for $\frac{d}{dt} z_{1,L}(h(t))$ follows directly from Equation (6.5). For $\frac{d}{dt} z_2(h(t))$ we can make use of Equation (6.6) and obtain (after exchanging the order of the sums):

$$\frac{d}{dt}\left(\sum_{i=2}^{n} h_{1,i,k}(t)(R_{i,k} - R_{i-1,k})\right) = \sum_{i=2}^{n} (h_{1,i-1,k}(t) - h_{1,i,k}(t)) s_k p_{i-1} \mu_{i-1} R_{i,k}$$

$$
\begin{aligned}
&- \sum_{i=2}^{n} (h_{1,i-1,k}(t) - h_{1,i,k}(t)) s_k p_{i-1} \mu_{i-1} R_{i-1,k} \\
&- \sum_{j=2}^{n} \left[ \sum_{i=2}^{j} (R_{i,k} - R_{i-1,k}) \right] (h_{1,j,k}(t) - h_{1,j+1,k}(t)) s_k \nu_j .
\end{aligned}
$$

Using (6.2) on the first sum, we can rewrite this as

$$
\begin{aligned}
\frac{d}{dt} \left( \sum_{i=2}^{n} h_{1,i,k}(t)(R_{i,k} - R_{i-1,k}) \right) &= \sum_{i=2}^{n} (h_{1,i-1,k}(t) - h_{1,i,k}(t)) s_k \nu_{i-1} R_{i-1,k} \\
&- \sum_{i=2}^{n} (h_{1,i-1,k}(t) - h_{1,i,k}(t)) \\
&- \sum_{j=2}^{n} (h_{1,j,k}(t) - h_{1,j+1,k}(t)) s_k \nu_j R_{j,k} \\
&+ \sum_{j=2}^{n} (h_{1,j,k}(t) - h_{1,j+1,k}(t)) s_k \nu_j R_{1,k} \\
&= (h_{1,1,k}(t) - h_{1,2,k}(t)) s_k \nu_1 R_{1,k} - h_{1,1,k}(t) + h_{1,n,k}(t) \\
&- h_{1,n,k}(t) s_k \nu_n R_{n,k} + \sum_{j=2}^{n} (h_{1,j,k}(t) - h_{1,j+1,k}(t)) s_k \nu_j R_{1,k} .
\end{aligned}
$$

The result follows by noting that $R_{1,k} = \frac{1}{s_k}$, $R_{n,k} = \frac{1}{s_k \mu_k}$ and $p_n = 0$. □

*Remark* 6.5.5. From the definitions of $z_{1,L}(h(t))$ and $z_2(h(t))$ follows immediately that $\frac{d}{dt} z_{1,L}(\pi) = 0$ and $\frac{d}{dt} z_2(\pi) = 0$.

**Proposition 6.5.6.** *Assume that Assumptions 6.1 - 6.5 hold and that $\mu_i(1 - p_i)$ is decreasing in i. For any $h(0) \in \Omega^{nh}$ with $h(0) \leq_C \pi$ or $\pi \leq_C h(0)$, $h(t)$ converges pointwise to $\pi$.*

*Proof.* The proof is similar to the one of [76, Proposition 8]. We assume $\pi \leq_C h(0)$, the proof for $h(0) \leq_C \pi$ follows similarly. We first show that $h_{1,1,k}(t)$ converges to $\pi_{1,1,k}$ for every $k \in \{1, \dots, T\}$. Due to Lemma 6.5.3, with $L, j = 1$, it suffices to show that $\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k (h_{1,1,k}(t) - \pi_{1,1,k}) dt < \infty$. Let $z(h) = z_{1,1}(h) + z_2(h)$. By Lemma 6.5.4 and Assumption 6.5 with $L = 1$ we then have

$$
\frac{d}{dt} z(h(t)) = \sum_{k=1}^{T} \sum_{\ell \geq 1} f_{\ell,1,k}(\pi) - \sum_{k=1}^{T} s_k h_{1,1,k}(t) - a_{1,\pi}(h(t)).
$$

Further, due to Remark 6.5.5 we have $\frac{d}{dt} z(\pi) = 0$, which implies $\sum_{k=1}^{T} \sum_{\ell \geq 1} f_{\ell,1,k}(\pi) = \sum_{k=1}^{T} s_k \pi_{1,1,k}$ as $a_{1,\pi}(\pi) = 0$. We thus get

$$
\begin{aligned}
\int_{t=0}^{\tau} \sum_{k=1}^{T} s_k (h_{1,1,k}(t) - \pi_{1,1,k}) dt &= - \int_{t=0}^{\tau} \frac{d}{dt} z(h(t)) dt - \int_{t=0}^{\tau} a_{1,\pi}(h(t)) dt \\
&\leq z(h(0)) - z(h(\tau)) \leq z(h(0)),
\end{aligned}
$$

as $a_{1,\pi}(h(t)) \geq 0$ and $z(h(\tau)) \geq 0$ for $\tau \geq 0$ (due to Lemma 6.2.1). Hence $\int_{t=0}^{\tau} \sum_{k=1}^{T} s_k(h_{1,1,k}(t) - \pi_{1,1,k})dt$ is uniformly bounded in $\tau \geq 0$, meaning $\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k(h_{1,1,k}(t) - \pi_{1,1,k})dt < \infty$.

We now show that $h_{1,j,k}(t)$ converges to $\pi_{1,j,k}$ for $j = 2, \dots, n$ and for $k = 1, \dots, T$, by arguing that $\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k(h_{1,j,k}(t) - \pi_{1,j,k})dt < \infty$. This is sufficient due to Lemma 6.5.3, with $L = 1$. As $\nu_j$ is decreasing in $j$, we have $\nu_{j-1} - \nu_j > 0$ and it suffices to show that

$$\int_{t=0}^{\tau} \sum_{k=1}^{T} s_k \sum_{j=2}^{n} (h_{1,j,k}(t) - \pi_{1,j,k})(\nu_{j-1} - \nu_j)dt$$

is uniformly bounded in $\tau \geq 0$. As $z_2(h(\tau)) \geq 0$ (thanks to Lemma 6.2.1), we have $z_2(h(0)) \geq -\int_{t=0}^{\tau} \frac{dz_2(h(t))}{dt}dt$ and Lemma 6.5.4 implies

$$\begin{aligned}
z_2(h(0)) &\geq \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k h_{1,1,k}(t)dt - \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k \sum_{j=1}^{n} (h_{1,j,k}(t) - h_{1,j+1,k}(t))\nu_j dt \\
&= \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k h_{1,1,k}(t)(1 - \nu_1)dt + \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k \sum_{j=2}^{n} h_{1,j,k}(t)(\nu_{j-1} - \nu_j)dt \\
&\geq \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k(h_{1,1,k}(t) - \pi_{1,1,k})(1 - \nu_1)dt \\
&+ \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k \sum_{j=2}^{n} (h_{1,j,k}(t) - \pi_{1,j,k})(\nu_{j-1} - \nu_j)dt,
\end{aligned}$$

where the last inequality is due to $\pi \leq_C h(0)$ and Proposition 6.5.1. This shows the uniform boundedness in $\tau$ as $0 \leq \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k(h_{1,1,k}(t) - \pi_{1,1,k})dt$.

We complete the proof by showing that $(h_{L,j,k}(t) - h_{L,j+1,k}(t))$ converges to $(\pi_{L,j,k} - \pi_{L,j+1,k})$, for $L > 1$ and $j = 1, \dots, n$. Note that $(h_{L,j,k}(t) - h_{L,j+1,k}(t))$ is not necessarily larger than $(\pi_{L,j,k} - \pi_{L,j+1,k})$ when $\pi \leq_C h(t)$. We do however have $h_{L,j,k}(t) + h_{L-1,j+1,k}(t) - h_{L,j+1,k}(t) \geq \pi_{L,j,k} + \pi_{L-1,j+1,k}(t) - \pi_{L,j+1,k}$ due to (6.9). Using a similar argument as in Lemma 6.5.3, we get that showing

$$\int_{t=0}^{\infty} \sum_{k=1}^{T} s_k[(h_{L,j,k}(t) + h_{L-1,j+1,k}(t) - h_{L,j+1,k}(t)) - (\pi_{L,j,k} + \pi_{L-1,j+1,k}(t) - \pi_{L,j+1,k})]dt < \infty$$

is a sufficient condition for $h_{L,j,k}(t) + h_{L-1,j+1,k}(t) - h_{L,j+1,k}(t)$ to converge to $\pi_{L,j,k} + \pi_{L-1,j+1,k}(t) - \pi_{L,j+1,k}$ for every $k$. Thus, using induction on $L$ it suffices to show that

$$\begin{aligned}
\Psi_{L,\tau} = \int_{t=0}^{\tau} \sum_{k=1}^{T} s_k \sum_{j=1}^{n} ((h_{L,j,k}(t) + h_{L-1,j+1,k}(t) - h_{L,j+1,k}(t)) \\
- (\pi_{L,j,k} + \pi_{L-1,j+1,k} - \pi_{L,j+1,k}))\nu_j dt
\end{aligned}$$

is uniformly bounded in $\tau \geq 0$. As $z_{1,L}(h(\tau)) \geq 0$ for $\tau \geq 0$, we have $z_{1,L}(h(0)) \geq -\int_{t=0}^{\tau} \frac{dz_{1,L}(h(t))}{dt}dt$ and Lemma 6.5.4 implies for $L \geq 1$

$$z_{1,L}(h(0)) \geq -\int_{t=0}^{\tau} \sum_{k=1}^{T} \sum_{\ell \geq L} f_{\ell,1,k}(h(t))dt + \sum_{k=1}^{T} s_k \int_{t=0}^{\tau} \sum_{j=1}^{n} (h_{L,j,k}(t) - h_{L,j+1,k}(t))\nu_j dt$$

$$= -\int_{t=0}^{\tau}\sum_{k=1}^{T}\sum_{\ell\geq L}(f_{\ell,1,k}(h(t)) - f_{\ell,1,k}(\pi))dt$$
$$+\int_{t=0}^{\tau}\sum_{k=1}^{T} s_k \sum_{j=1}^{n}((h_{L,j,k}(t) - h_{L,j+1,k}(t)) - (\pi_{L,j,k} - \pi_{L,j+1,k}))\nu_j dt,$$

where the last equality holds thanks to Lemma 6.5.4 and Remark 6.5.5. Therefore, we find

$$z_{1,L}(h(0)) + \int_{t=0}^{\tau}\sum_{k=1}^{T} s_k \sum_{j=1}^{n}(h_{L-1,j+1,k}(t) - \pi_{L-1,j+1,k})\nu_j dt$$
$$\geq -\int_{t=0}^{\tau}\sum_{k=1}^{T}\sum_{\ell\geq L}(f_{\ell,1,k}(h(t)) - f_{\ell,1,k}(\pi))dt + \Psi_{L,\tau}.$$

By relying on Assumption 6.5 with $L > 1$ this is equivalent to

$$z_{1,L}(h(0)) + \int_{t=0}^{\tau}\sum_{k=1}^{T} s_k \sum_{j=1}^{n}(h_{L-1,j+1,k}(t) - \pi_{L-1,j+1,k})\nu_j dt$$
$$+\int_{t=0}^{\tau}\sum_{k=1}^{T}\sum_{\ell=1}^{L-1}\sum_{j=1}^{n} b_{L,\ell,j,k}(h(t))(h_{\ell,j,k}(t) - \pi_{\ell,j,k})dt$$
$$\geq \Psi_{L,\tau} + \int_{t=0}^{\tau} a_{L,\pi}(h(t))dt.$$

As $b_{L,\ell,j,k}(h)$ is bounded on $\Omega^{nh}$, the left hand side is uniformly bounded in $\tau$ by induction on $L$ and therefore so are the (positive) integrals on the right hand side. □

We need Assumption 6.5 and functions $z_{1,L}(h(t))$ and $z_2(h(t))$ to prove Proposition 6.5.6. This is not the case if the buffer is assumed to be finite. Having a finite buffer also simplifies finding the state $h^u$ in Proposition 6.5.2.

Essentially, we have now proven the main theorem of this chapter:

**Theorem 6.5.7** (Global Attraction). *Consider the set of ODEs given by* (6.5) - (6.6). *Assume that Assumptions 6.1 - 6.5 hold and $\mu_i(1-p_i)$ is decreasing in i. Then, for any $h(0) \in \Omega^{nh}$, $h(t)$ converges pointwise to the unique fixed point $\pi \in \Omega^{nh}$ as t tends to infinity.*

*Proof.* This follows from Propositions 6.5.2 and 6.5.6. □

## 6.6 Example: JSQ($d$)

JSQ($d$) stands for Join Shortest Queue out of $d$ selected queues. The policy works as follows: jobs arrive to the central dispatcher according to Poisson process with parameter $\lambda N$, upon an arrival the dispatcher chooses $d$ queues at random and assigns the job to a queue with the least number of jobs (with ties between queue lengths broken uniformly at random). We assume that the dispatcher distributes jobs instantaneously, i.e. no time passes between the dispatcher receiving a job and the job being assigned to a queue.

An advantage of the policy is that the dispatcher does not have to know all the queue lengths, which can be problematic for large values of $N$. As such, the policy is often used in cloud load balancing setting (where there are possibly multiple dispatchers and heterogeneous servers). Among popular load balancers, NGINX [23] and HAProxy [73] can use the JSQ(2) policy. The policy has been studied for example in [58,81,82].

In this section we remark on the stability of the system and then show that the Assumptions 6.1-6.5 are satisfied for the JSQ($d$) policy.

We first note that having $\lambda < 1$ is not enough for the system to be stable. Indeed, consider the following example:

**Example 6.6.1.** Suppose that we have two types of servers ($T = 2$) and that $q_1 = q_2 = 1/2$, i.e. half of the system consists of servers of type 1 and the other half of servers of type 2. Suppose further that $s_1 = 1.8$ and $s_2 = 0.2$, such that the average strength of servers is $s_1/2 + s_2/2 = 1$ as required. Assume that the system handles the JSQ(2) policy. In this case, on average, for every fourth arrival two queues of type 2 get chosen. This implies that queues of type 2 get at least one fourth of the incoming jobs. The average number of jobs that every queue of type 2 receives per time unit is therefore at least $\lambda/(4q_2) = \lambda/2$. Hence, even if $\lambda < 1$, it is possible that $\lambda/2 \geq 0.2$ and the system is unstable.

The example above shows that we need a stronger stability condition than $\lambda < 1$. It also shows that the condition should depend on the fractions and strengths of servers of given types. The needed stability condition is given in [59, Section 2.4]:

$$\lambda < \min_{k=1}^{T} \left( \frac{\sum_{\ell=1}^{k} q_\ell s_\ell}{\left(\sum_{\ell=1}^{k} q_\ell\right)^d} \right), \tag{6.16}$$

where we assume, without loss of generality, that $s_1 \leq s_2 \leq \cdots \leq s_T$. Note that [59, Section 2.4] deals with heterogeneous processor sharing systems with exponential job sizes. The stability condition is however the same for any service policy and job size distribution (of the same mean). Note further, that if $T = 1$, then (6.16) simplifies to $\lambda < 1$.

We now derive a formula for $f_{\ell,i,k}(h(t))$. First note that $\sum_{m=1}^{T} h_{\ell,1,m}(t)$ is the fraction of queues with at least $\ell$ jobs at time $t$. It follows that

$$\left(\sum_{m=1}^{T} h_{\ell-1,1,m}(t)\right)^d - \left(\sum_{m=1}^{T} h_{\ell,1,m}(t)\right)^d$$

is the probability that the dispatcher chooses a queue with exactly $\ell - 1$ jobs upon an arrival at time $t$, as this simply is the probability of choosing $d$ queues with at least $\ell - 1$ jobs of which not all have $\ell$ or more jobs. Further, the probability that a queue with $\ell - 1$ jobs is of type $k$ at time $t$ equals

$$\frac{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)}{\sum_{m=1}^{T} (h_{\ell-1,1,m}(t) - h_{\ell,1,m}(t))}.$$

It now follows that for $\ell \geq 1$

$$f_{\ell,1,k}(h(t)) = \lambda \left[ \left( \sum_{m=1}^{T} h_{\ell-1,1,m}(t) \right)^{d} - \left( \sum_{m=1}^{T} h_{\ell,1,m}(t) \right)^{d} \right] \frac{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)}{\sum_{m=1}^{T} \left( h_{\ell-1,1,m}(t) - h_{\ell,1,m}(t) \right)} \tag{6.17}$$

$$= \lambda \left( h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t) \right) \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{\ell-1,1,m}(t) \right)^{j} \left( \sum_{m=1}^{T} h_{\ell,1,m}(t) \right)^{d-1-j}, \tag{6.18}$$

where we used

$$\frac{a^d - b^d}{a-b} = \sum_{j=0}^{d-1} a^j b^{d-1-j} \tag{6.19}$$

in the second equality. As

$$\frac{h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t)}{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)}$$

is the probability that a queue of type $k$ with $\ell - 1$ jobs at time $t$ has the leading job in phase $\geq i$, we further get for $\ell > 1$ and $i \geq 2$ that

$$f_{\ell,i,k}(h(t)) = f_{\ell,1,k}(h(t)) \cdot \frac{h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t)}{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)}$$

$$= \lambda \left( h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t) \right) \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{\ell-1,1,m}(t) \right)^{j} \left( \sum_{m=1}^{T} h_{\ell,1,m}(t) \right)^{d-1-j}. \tag{6.20}$$

Define $\Psi_{\ell,i,k}(h(t)) = \frac{d}{dt} h_{\ell,i,k}(t)$ and $\Psi(h) = [\Psi_{\ell,i,k}(h)]_{\ell,i,k}$. We call $\Psi$ the drift. Clearly, the drift is bounded on $\Omega^{nh}$. We now show that $\Psi$ is Lipschitz continuous on $\Omega^{nh}$ with respect to the supremum metric on $\Omega^{nh}$, with the supremum metric given by

$$\mathbf{d}(h, \tilde{h}) = \sup_{\ell=1}^{\infty} \sup_{i=1}^{n} \sup_{k=1}^{T} \left| h_{\ell,i,k} - \tilde{h}_{\ell,i,k} \right|,$$

where $h, \tilde{h} \in \Omega^{nh}$. It will then follow that for every $h \in \Omega^{nh}$ the set of ODEs (6.5)-(6.6) has a unique solution $h(t)$ with $h(0) = h$.

**Proposition 6.6.2.** *The drift $\Psi$ is Lipshitz continuous on $\Omega^{nh}$.*

*Proof.* Let $h, \tilde{h} \in \Omega^{nh}$. We have

$$\sup_{\ell \neq 0,i,k} \left| f_{\ell,i,k}(h) - f_{\ell,i,k}(\tilde{h}) \right| = \lambda \sup_{\ell \neq 0,i,k} \left| (h_{\ell-1,i,k} - h_{\ell,i,k}) \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{\ell-1,1,m} \right)^{j} \left( \sum_{m=1}^{T} h_{\ell,1,m} \right)^{d-1-j} \right.$$
$$\left. - \left( \tilde{h}_{\ell-1,i,k} - \tilde{h}_{\ell,i,k} \right) \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} \tilde{h}_{\ell-1,1,m} \right)^{j} \left( \sum_{m=1}^{T} \tilde{h}_{\ell,1,m} \right)^{d-1-j} \right|.$$

We now use the inequality

$$|a_1^{m_1} a_2^{m_2} - b_1^{m_1} b_2^{m_2}| \leq m_1 |a_1 - b_1| + m_2 |a_2 - b_2|, \tag{6.21}$$

for $0 \leq a_1, a_2, b_1, b_2 \leq 1$ and $m_1, m_2 \in \mathbb{N} \setminus \{0\}$, to find that

$$\sup_{\ell \neq 0, i, k} \left| f_{\ell,i,k}(h) - f_{\ell,i,k}(\tilde{h}) \right| \leq \lambda \sup_{\ell \neq 0, i, k} \left| h_{\ell-1,i,k} - h_{\ell,i,k} - \tilde{h}_{\ell-1,i,k} + \tilde{h}_{\ell,i,k} \right| \tag{6.22}$$

$$+ \lambda \sup_{\ell=1}^{\infty} \sum_{j=0}^{d-1} \left| \left( \sum_{m=1}^{T} h_{\ell-1,1,m} \right)^j \left( \sum_{m=1}^{T} h_{\ell,1,m} \right)^{d-1-j} - \left( \sum_{m=1}^{T} \tilde{h}_{\ell-1,1,m} \right)^j \left( \sum_{m=1}^{T} \tilde{h}_{\ell,1,m} \right)^{d-1-j} \right|. \tag{6.23}$$

Note that due to $h_{0,i,k} = h_{1,i,k}$ if $i > 1$ and $h_{0,i,k} = q_k$ if $i = 1$ (for every $h \in \Omega^{nh}$), we get that (6.22) is smaller than or equal to $2\lambda \mathbf{d}(h, \tilde{h})$. Using (6.21) inside the sum over $j$'s in (6.23), we further get

$$\begin{aligned} &\sup_{\ell \neq 0, i, k} \left| f_{\ell,i,k}(h) - f_{\ell,i,k}(\tilde{h}) \right| \\ &\leq 2\lambda \mathbf{d}(h, \tilde{h}) + \lambda \sup_{\ell=1}^{\infty} \sum_{j=0}^{d-1} \left| j \sum_{m=1}^{T} (h_{\ell-1,1,m} - \tilde{h}_{\ell-1,1,m}) + (d-1-j) \sum_{m=1}^{T} (h_{\ell,1,m} - \tilde{h}_{\ell,1,m}) \right| \\ &\leq 2\lambda \mathbf{d}(h, \tilde{h}) + \lambda d(d-1) \sup_{\ell=1}^{\infty} \left| \sum_{m=1}^{T} (h_{\ell-1,1,m} - \tilde{h}_{\ell-1,1,m}) \right| \\ &\leq 2\lambda \mathbf{d}(h, \tilde{h}) + \lambda d(d-1) T \mathbf{d}(h, \tilde{h}), \end{aligned}$$

where in the last inequality we have used $h_{0,1,m} = q_m$, for every $h \in \Omega^{nh}$. It now follows that

$$\begin{aligned} \mathbf{d}(\Psi(h), \Psi(\tilde{h})) \leq{} & \lambda(2 + d(d-1)T) \mathbf{d}(h, \tilde{h}) + 4n \max_{i=1}^{n} \nu_i \max_{k=1}^{T} s_k \mathbf{d}(h, \tilde{h}) \\ & + 2 \max_{i=1}^{n} p_i \mu_i \max_{k=1}^{T} s_k \mathbf{d}(h, \tilde{h}), \end{aligned}$$

which finishes the proof. □

We only prove the existence of a fixed point in case of finite buffers. So suppose that every queue has a buffer of size $B < \infty$. To make sure that queue lengths never exceed $B$ in our model, we can simply set $f_{\ell,i,k}(h) = 0$ for $\ell > B$. Then, $\Omega^{nh}$ is a compact, positively invariant subset of $\mathbb{R}^{BnT}$. Further, the set $\Omega^{nh}$ is clearly homeomorphic to the closed unit ball in $\mathbb{R}^{BnT}$. [4, Theorem 1.9.6] now implies that there exists a fixed point in $\Omega^{nh}$. This, together with Proposition 6.6.2, proves that Assumption 6.1 holds for JSQ($d$) policy (with finite buffers).

We now show that Assumptions 6.2-6.5 hold for any buffer size (including infinite buffers). Looking at Equations (6.18) and (6.20), it may look like Assumption 6.2 does not hold due to the terms $-h_{\ell,1,k}(t)$ and $-h_{\ell,i,k}(t)$ respectively. However, as $h_{\ell-1,i,k}(t) \geq h_{\ell,i,k}(t)$ for any $\ell \geq 1$ and any $i$ and $k$, the Assumption 6.2 does hold. Assumption 6.3 can be shown analogously to [76, Section 7.1]: for $(\ell_1, \ldots, \ell_n)$ with $\ell_1 > \cdots > \ell_n$, we have

$$f_{(\ell_1,\ldots,\ell_n),k}(h) = \lambda \sum_{i=1}^{n} \Bigg[ \left( g_{(\ell_1,\ldots,\ell_{i-1},\ell_i - 1[\ell_i > 1],\ell_{i+1},\ell_n),k}(h) - g_{(\ell_1,\ldots,\ell_n),k}(h) \right)$$

$$\cdot \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{\ell_i-1,1,m} \right)^j \left( \sum_{m=1}^{T} h_{\ell_i,1,m} \right)^{d-1-j} \Bigg].$$

For the general case assume that $\tilde{\ell}_1 > \cdots > \tilde{\ell}_{n'}$ are the unique values appearing in the sequence $\ell_1 \geq \cdots \geq \ell_n$ and let $\ell_{j_i}$ be the first element in this sequence equal to $\tilde{\ell}_i$ for $i = 1, \ldots, n$. We then have

$$f_{(\ell_1,\ldots,\ell_n),k}(h) = \lambda \sum_{i=1}^{n'} \Bigg[ 1[\tilde{\ell}_i > 1] \left( g_{(\ell'_1,\ldots,\ell'_n),k}(h) - g_{(\ell_1,\ldots,\ell_n),k}(h) \right)$$
$$\cdot \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{\tilde{\ell}_i-1,1,m} \right)^j \left( \sum_{m=1}^{T} h_{\tilde{\ell}_i,1,m} \right)^{d-1-j} \Bigg],$$

where $\ell'_m = \ell_m - 1$ for $j_i \leq m \leq j_{i+1}$ and $\ell'_m = \ell_m$ otherwise. Assumption 6.3 thus also holds.

Looking at (6.18) and (6.20), Assumption 6.4 clearly holds, as $0 \leq h_{\ell,j,k}(t) \leq 1$ for every $t$ and every $(\ell, j, k)$.

For every $h \in \Omega^{nh}$, we have $\sum_{k=1}^{T} \sum_{\ell \geq 1} f_{\ell,1,k}(h) = \lambda$. Therefore, for Assumption 6.5 with $L = 1$, we can simply set $a_{1,\pi}(h) = 0$. Suppose $L > 1$. By using formula (6.17), we get that $\sum_{k=1}^{T} \sum_{\ell \geq L} (f_{\ell,1,k}(h) - f_{\ell,1,k}(\pi))$ equals

$$\lambda \sum_{\ell \geq L} \left[ \left( \sum_{m=1}^{T} h_{\ell-1,1,m} \right)^d - \left( \sum_{m=1}^{T} h_{\ell,1,m} \right)^d - \left( \sum_{m=1}^{T} \pi_{\ell-1,1,m} \right)^d + \left( \sum_{m=1}^{T} \pi_{\ell,1,m} \right)^d \right].$$

Working out the sum over $\ell$, this simplifies to

$$\lambda \left[ \left( \sum_{m=1}^{T} h_{L-1,1,m} \right)^d - \left( \sum_{m=1}^{T} \pi_{L-1,1,m} \right)^d \right],$$

which is equal to

$$\lambda \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{L-1,1,m} \right)^j \left( \sum_{m=1}^{T} \pi_{L-1,1,m} \right)^{d-1-j} \sum_{k=1}^{T} (h_{L-1,1,k} - \pi_{L-1,1,k}),$$

due to Equation (6.19). For every $k \in \{1, \ldots, T\}$, we can thus set

$$b_{L,L-1,1,k}(h) = \lambda \sum_{j=0}^{d-1} \left( \sum_{m=1}^{T} h_{L-1,1,m} \right)^j \left( \sum_{m=1}^{T} \pi_{L-1,1,m} \right)^{d-1-j} \leq \lambda d,$$

$b_{L,\ell,j,k}(h) = 0$ for $(\ell, j) \neq (L-1, 1)$ and $a_{L,\pi}(h) = 0$, which shows that Assumption 6.5 holds for $L > 1$.

## 6.7 Example: split-JSQ($d$)

In this section we introduce a variant of the JSQ($d$) policy, which we call split-JSQ($d$). The split-JSQ($d$) policy works as follows. Jobs arrive to the central dispatcher according to a Poisson process with parameter $\lambda N$. Upon an arrival, the dispatcher first picks a type of queue according to a general distribution. Let $\rho_k$ denote the probability that the dispatcher will choose type $k$, for $k = 1, \dots, T$. We call these probabilities the "routing probabilities". The dispatcher then uses JSQ($d$) policy on the queues of the chosen type, i.e. it picks $d$ queues of that type at random and it assigns the job to a queue with the lowest number of jobs among the chosen queues, with ties broken randomly. We assume that the dispatcher distributes jobs instantaneously.

This policy was studied in [80] for jobs of the phase type, however the authors did not prove global attraction in their paper. The policy was also studied for exponential job sizes in [59, Section 2.5] under the name hybrid SQ($d$), where the author provided a way of finding the optimal routing probabilities.

We note, that the existence of a fixed point and global attraction for the split-JSQ($d$) follow immediately if we have existence of a fixed point and global attraction in each subsystem consisting servers of type $k$. Thus, if each subsystem is stable, we can simply use [76, Section 7.1] and existence of a fixed point and global attraction follow.

We now determine the conditions needed for stability. As the average queue strength and mean job size are both 1, the first condition for stability is $\lambda < 1$. We also need stability for each type of queues. Hence, we need for every $k = 1, \dots, T$, that $\lambda\rho_k < s_k q_k$ as $\lambda\rho_k/q_k$ is the average amount of jobs that a queue of type $k$ gets per time unit and $s_k$ is the strength of a type $k$ server. For the remainder of this section we therefore assume that $\lambda < 1$ and that $\lambda\rho_k < s_k q_k$ holds for every $k = 1, \dots, T$.

As explained above, the existence of a fixed point and global attraction now follow. However, as an example we analyze the policy in more detail. We first determine the terms $f_{\ell,i,k}(h(t))$. We then verify Assumptions 6.1-6.5, except Assumption 6.3, where we simply refer to [76].

We will now derive a formula for $f_{\ell,i,k}(h(t))$. The probability that at time $t$ a queue has at least $\ell$ jobs provided that it is of type $k$ is $h_{\ell,1,k}(t)/q_k$. Therefore,

$$\frac{1}{q_k^d}\left(h_{\ell-1,1,k}(t)^d - h_{\ell,1,k}(t)^d\right)$$

is the probability that a dispatcher chooses $d$ queues of which at least one has at least $\ell - 1$ jobs and not all queues have $\ell$ or more jobs, provided that the dispatcher chooses type $k$ queues. As $\lambda\rho_k/q_k$ is the average amount of jobs that a queue of type $k$ gets per time unit, it follows that

$$\begin{aligned} f_{\ell,1,k}(h(t)) &= \frac{\lambda\rho_k}{q_k} \cdot q_k \cdot \frac{1}{q_k^d}\left(h_{\ell-1,1,k}(t)^d - h_{\ell,1,k}(t)^d\right) \\ &= \frac{\lambda\rho_k}{q_k^d}\left(h_{\ell-1,1,k}(t)^d - h_{\ell,1,k}(t)^d\right) \end{aligned} \tag{6.24}$$

for $\ell \geq 1$. As

$$\frac{h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t)}{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)}$$

is the probability that a queue of type $k$ has $\ell - 1$ jobs with leading job in phase $\geq i$, we further get for $\ell > 1$ and $i \geq 2$ that

$$\begin{aligned} f_{\ell,i,k}(h(t)) &= \frac{\lambda \rho_k}{q_k^d} \left( h_{\ell-1,1,k}(t)^d - h_{\ell,1,k}(t)^d \right) \frac{h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t)}{h_{\ell-1,1,k}(t) - h_{\ell,1,k}(t)} \\ &= \frac{\lambda \rho_k}{q_k^d} \left( h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t) \right) \sum_{j=0}^{d-1} h_{\ell-1,1,k}(t)^j h_{\ell,1,k}(t)^{d-1-j}, \end{aligned} \tag{6.25}$$

where we have used Equation (6.19) to obtain (6.25).

We now verify Assumptions 6.1-6.5 for the split-JSQ($d$) policy. We first show that Assumption 6.1 holds for the split-JSQ($d$) policy. Clearly, if we focus on queues of type $k$, we get the following subsystem: on average we have $\lambda \rho_k / q_k$ arrivals per queue per time unit, these arrivals get distributed according to JSQ($d$) policy and the service of a job takes $1/s_k$ time units on average. We can re-scale the time by a factor of $1/s_k$ and we get that the subsystem is the system from [76, Section 4.1]. The subset of the set ODEs (6.5)-(6.6) concerning the queues of type $k$, then has a unique solution, for every $k = 1, \ldots, T$. The whole set of ODEs then also has a unique solution $h(t)$. Further, the subsystem of queues of type $k$ has a fixed point, which we will denote by $\pi^{(k)}$. We now get a fixed point needed for Assumption 6.1 by setting $\pi_{\ell,i,k} = \pi^{(k)}_{\ell,i}$ for every $(\ell, i, k)$.

Assumption 6.2 clearly holds if we look at (6.24). At first glance it may look like Assumption 6.2 does not hold for (6.25) for the terms inside the sum due to $-h_{\ell,i,k}(t)$ before the sum. However, as $h_{\ell-1,i,k}(t) - h_{\ell,i,k}(t) \geq 0$, Assumption 6.2 also holds for (6.25). Further, Assumption 6.3 can be shown analogously to Assumption A2 in [76, Section 7.1].

Looking at (6.24) and (6.25), Assumption 6.4 clearly holds, as $0 \leq h_{\ell,j,k}(t) \leq 1$ for every $t$ and every $(\ell, j, k)$ as $q_k > 0$ for every $k$.

For Assumption 6.5, we note the following. For $L = 1$, we have $\sum_{k=1}^{T} \sum_{\ell \geq 1} f_{\ell,1,k}(h) = \sum_{k=1}^{T} \lambda \rho_k = \lambda$ for every $h \in \Omega^{nh}$. Therefore, we can set $a_{1,\pi}(h) = 0$. For $L > 1$, we have

$$\sum_{\ell \geq L} \left( f_{\ell,1,k}(h) - f_{\ell,1,k}(\pi) \right) = \frac{\lambda \rho_k}{q_k^d} \left( h^d_{\ell-1,1,k} - \pi^d_{\ell-1,1,k} \right),$$

which means that we can choose $a_{L,\pi}(h) = 0$ and due to Equation (6.19)

$$b_{L,L-1,1,k}(h) = \frac{\lambda \rho_k}{q_k^d} \sum_{j=0}^{d-1} h^j_{L-1,1,k} \pi^{d-1-j}_{L-1,1,k} \leq \frac{\lambda \rho_k d}{q_k^d}.$$

Finally, we can set $b_{L,\ell,j,k}(h) = 0$ for $(\ell, j) \neq (L-1, 1)$.

## 6.8 Conclusion

In this chapter we were able to generalize the results from [76]. We have namely shown that monotonicity arguments can be used to show global attraction in mean field models for a large class of heterogeneous systems with hyperexponential job sizes (whereas [76] dealt with the homogeneous case). This confirmed a part of the author's belief in [76, Section 9].

As examples, we have shown that the global attraction holds in case of two heterogeneous variants of the JSQ($d$) policy, which we called JSQ($d$) and split-JSQ($d$). For the latter of these global attraction follows almost immediately from [76, Section 7.1]. We expect that global attraction also holds in case of heterogeneous variants of other policies, such as pull and push policies.

We share the belief from [76, Section 9], that a similar monotonicity argument can be applied to systems where every queue uses FCFS discipline and has $C > 1$ servers. This, however, is a subject of future work.

# Part III

# Multithreaded Computing

Jobs in multithreaded computing systems consist of several threads [7,83]. Upon starting the execution a main thread (which we call a parent job) several other threads are spawned (which we call child jobs). These spawned child jobs are initially stored locally, but can be redistributed at a later stage. One way of redistributing jobs is called "randomized work stealing": processors that become empty start probing other processors at random (uniformly) and if the probed processor has pending jobs, some of its jobs are transferred to the probing processor [7,16]. Another option is to make use of "randomized work sharing", where servers that have pending jobs probe others to offload some of their work to other servers.

Work stealing solutions have been studied by various authors and are often used in practice. They have been implemented for example in Cilk programming language [8,18], Intel TBB [64], Java fork/join framework [49], KAAPI [26] and .NET Task Parallel Library [50]. Some early studies on work sharing and stealing include [16,57,69]. In [16] the performance of work stealing and sharing is compared for homogenous systems with exponential job sizes. Using similar techniques the work in [16] was generalized to heterogeneous systems in [57]. The key takeaway from these papers is that work stealing clearly outperforms work sharing in system with high load.

More recent work includes [25,55,56,77,79]. In [25] the authors analyse the system consisting of several homogeneous clusters with exponential job sizes and where half of the jobs are transferred when a probe is successful. A fair comparison between stealing and sharing strategies is given for homogeneous networks and exponential job sizes in [55,56] and for non-exponential job sizes in [77]. Further, the comparison in [55] is extended to heterogeneous networks in [79]. The key difference with the systems in this part is that in these prior works jobs are considered to be sequential and are always executed as a whole on a single server.

In this part we study multithreaded computing through the use of large scale homogeneous systems with randomized work stealing. We namely let empty servers send probes, if a server with pending jobs or pending parts of a job is probed, some work is transferred to the probing server. We assume that each job (parent job) spawns several other jobs (child jobs) upon the beginning of its service. Note, that the child jobs can have other distribution than the parent. We refer to the parent job together with its children as a job.

The part is structured as follows. It consists of three chapters (Chapters 7-9), each based on a published paper (papers [68], [44] and [45] respectively). In Chapter 7 we make a comparison of two systems: in the first one only a single parent can be transferred at once, in the second one one child get transferred per successful steal. In Chapter 8 we analyze systems where if a probed server has pending child jobs, a number of them is transferred to the probing queue. Otherwise, a pending parent job is transferred (if available). The number of transferred children follows a general distribution. This allows us to investigate and compare different stealing strategies. In Chapters 7 and 8 both parent and child job service requirements are assumed to be exponentially distributed. In Chapter 9 we generalize the analysis of Chapter 8 to parent and child jobs of the phase type.

As all these systems consist of multiple queues that can influence each other in a non-trivial way, it is impossible to study these systems exactly for $N$, the number of servers, large. We instead opt to analyze the systems with $N \to \infty$ through mean field techniques.

For each of the systems we analyze a QBD Markov chain describing the cavity queue. We also model each system using a set of ODEs and show that the unique fixed point of this set of ODEs coincides with the invariant distribution of the cavity queue of the corresponding system. As these mean field models are approximations of the $N$-server systems, we validate our analysis using simulation in each of the three chapters. Finally, for each system we perform various numerical experiments.

Chapter 7

# Randomized work stealing: a comparison of two systems

*This Chapter contains the paper* [68], *titled "Performance Analysis of Work Stealing in Large-scale Multithreaded Computing". The writing of this paper started in late 2019. This was the only time I got the chance to work with a researcher from abroad (Nikki Sonenberg) on site. The paper was published in June 2021 in ACM Transactions on Modeling and Performance Evaluation of Computing Systems.*

## 7.1 Introduction

In this chapter we analyze the performance of randomized work stealing on a large system of servers in the context of dynamic multithreaded computations. To the best of our knowledge we are the first to develop a mean field model for work stealing in this setting. Following [7,83], a multithreaded computation is composed of a set of threads, each of which has a sequential ordering of tasks. During execution, a thread, a sequence of instructions executed within the context of a job, may spawn new child threads and a scheduling algorithm determines which processors execute which threads.

We suppose there is a system of homogeneous processors operating with randomized work stealing and consider the problem of balancing workloads over the processors where jobs arriving to a processor can spawn child jobs that can feasibly be executed in parallel with the parent job. Child jobs are generated upon a parent job beginning service, and these child jobs are then initially stored locally. We consider two work stealing protocols: one where only child jobs are able to be migrated across servers; and one where parent jobs can be migrated across servers, but must be migrated together with their child jobs.

We define mean field models for both stealing strategies and validate the models using simulation. We prove the existence of a unique fixed point for each model and use this to study the performance of these strategies in terms of the response time distribution for the job. While we do not present convergence proofs for the stationary measures, we believe such proofs may be constructed using existing mean field results [24,47]. In this chapter we make the following contributions:

- We present two mean field models for work stealing in multithreaded computations.
- We prove that these models have a unique fixed point that can be computed efficiently using matrix analytic methods (by solving a single Quasi-Birth-Death Markov chain). This is the main technical contribution of the chapter.
- We indicate how to compute the response time distribution of a job for both strategies.
- For both strategies we illustrate the effect on mean response time of varying probe rate, load and child job size distributions. In selected scenarios, we show that with high probe rate and low loads, child stealing achieves a lower mean response time; but the parent stealing strategy clearly performs better under low probe rate and high loads.
- The developed model and methods provide a foundation for the performance analysis of more general systems with similar features.

The rest of this chapter is organised as follows. In Section 7.2 we describe the system and the work stealing strategies considered. The mean field model is introduced in Section 7.3. The model of a single server is introduced in Section 7.4 and used to compute the fixed point in Section 7.5. We present results for the response time distribution in Section 7.6 and in Section 7.7 present explicit results when the probe rate tends to infinity. We validate the mean field model using simulation in Section 7.8. The performance of the stealing strategies with numerical examples is presented in Section 7.9 and Section 7.10 contains some concluding remarks.

## 7.2 System description and strategies

We consider a system with the following characteristics:

i. $N$ homogeneous servers each with an infinite buffer to store jobs.

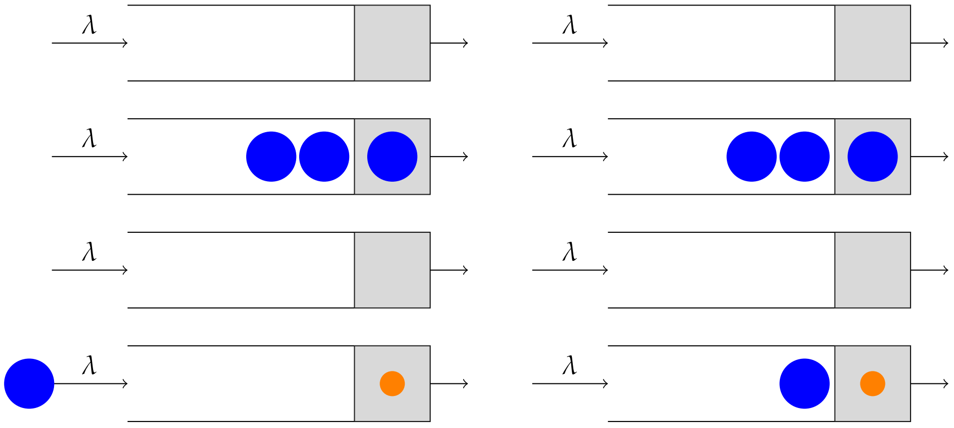
ii. Each server is subject to its own local Poisson arrival process with rate $\lambda$. Arriving jobs are referred to as parent jobs.

iii. Upon a parent entering service, the parent job spawns $i \in \{0, 1, \ldots, m\}, m \geq 1$, child jobs at that server, the number of which follows a general distribution with finite support, $\check{p} = \{p_i\}$. We refer to a *job* as a parent job and its spawned child jobs.

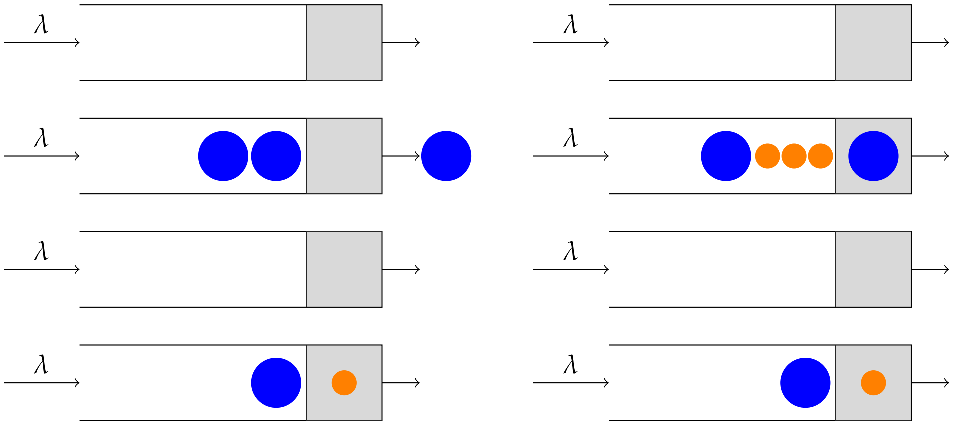
iv. Child jobs spawned at a server are served before any waiting parent jobs are served.

v. It is assumed that parent and child jobs have exponentially distributed service requirements with rates $\mu_1$ and $\mu_2$, respectively.

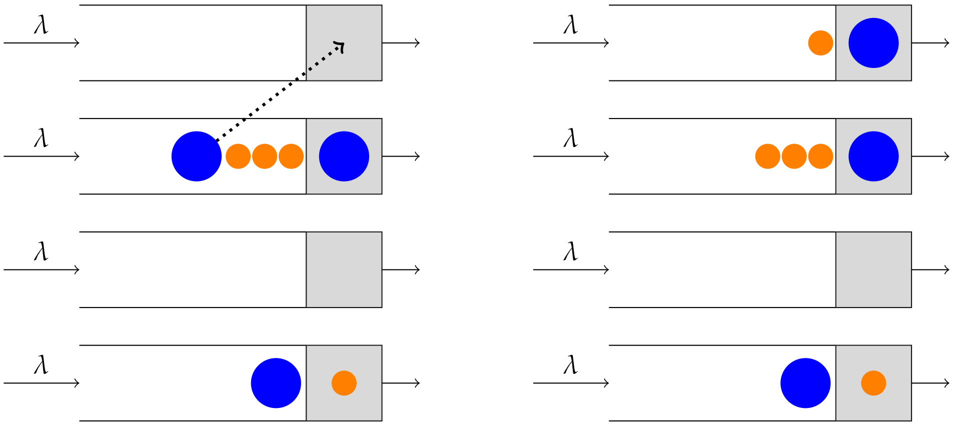
In this chapter we study the performance of rate based work stealing strategies [55,77] in our model of multithreaded computations. More specifically we consider the following two randomized work stealing protocols:



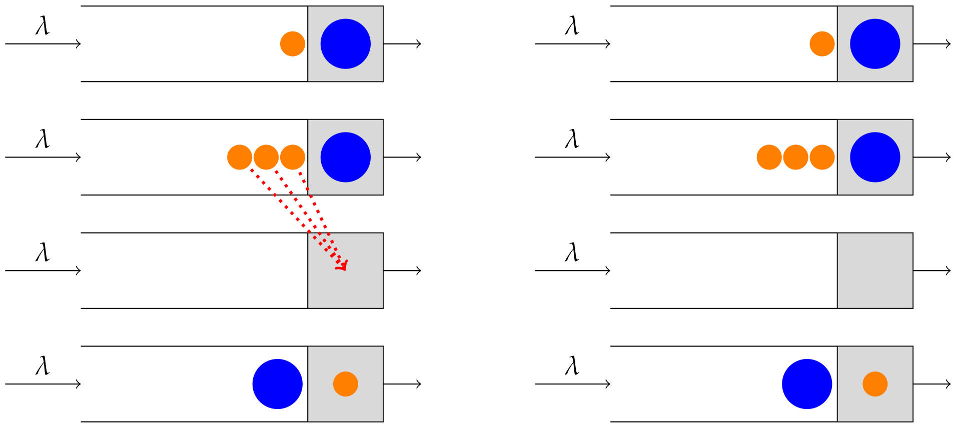
(a) A parent job enters fourth queue and starts waiting.



(b) A parent completes service in the second queue and the next parent enters service, spawning three children.



(c) The second queue is probed by the first and the waiting parent is transferred, starts service in the the first queue and spawns a child.



(d) The third queue probes the second. As the latter has no pending parents, no transfer occurs.

Figure 7.1: Example of a parent stealing system with $N = 4$. Blue and orange dots depict parent and child jobs respectively.

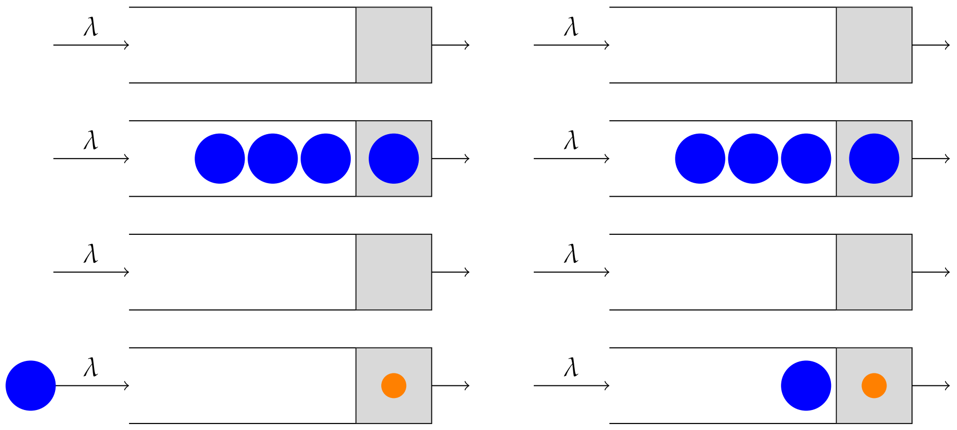
- *Parent job stealing*. When a server is idle, it generates probe messages at rate $r$. As long as the server remains idle, probes are sent according to Poisson process with rate $r$. This process is interrupted whenever the server becomes busy. The probed server is selected at random and a probe is successful if there are parent jobs waiting to be served. We assume the policy is to always steal the oldest parent job, that is, the head-of-the-line parent job in the waiting room.
- *Child job stealing*. Idle servers again probe at rate $r$, but a probe is only successful if there is at least one child job waiting to be served. In this case a single child job is transferred to the idle probing server. Extending the model such that multiple child jobs can be stolen at once is non trivial and subject to future work.

The two systems are illustrated for $N = 4$ in Figures 7.1 and 7.2 respectively.

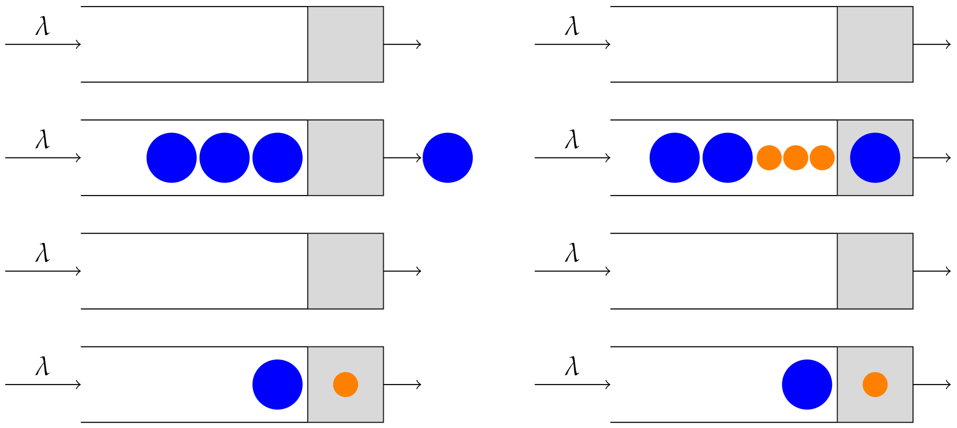
We compare the performance of these two strategies, noting that when a parent job is transferred to an idle server, its service immediately starts and its child jobs are spawned at this server. Probes and job transfers are assumed to be instantaneous. Another interpretation of the rate based probing is that it takes an exponentially distributed amount of time with mean $1/r$ to probe another server (and to transfer the job if the probe is successful) and steal attempts are executed sequentially.

## 7.3 Mean field model

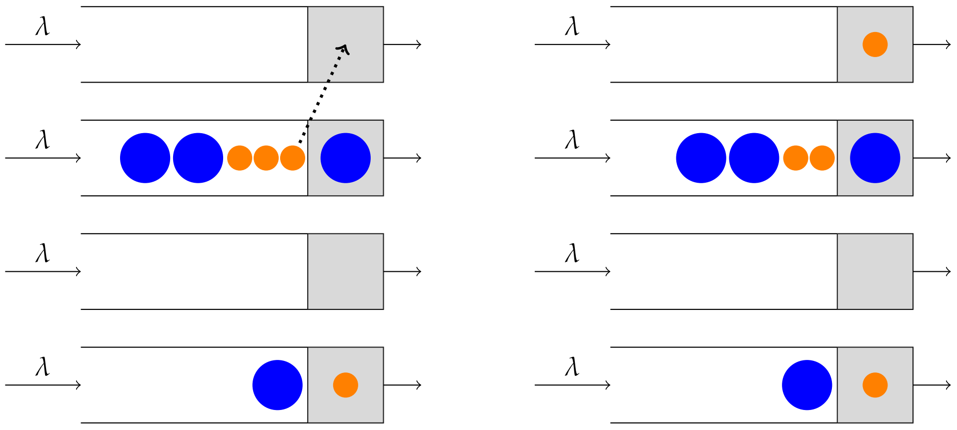
We use a mean field model to describe the system with $N \to \infty$ servers. The infinite system is defined by a set of ODEs and we use the superscript $(c)$ when referring to the system where child job stealing is allowed and superscript $(p)$ where parent job stealing is allowed.



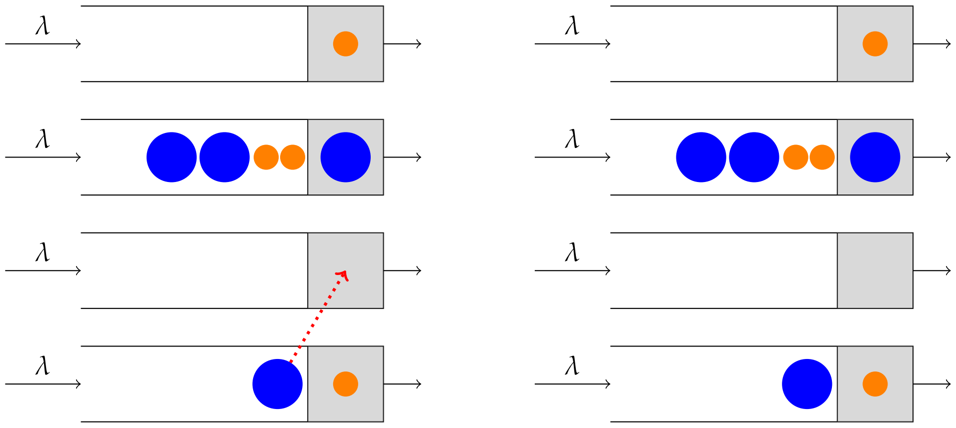
(a) A parent job enters fourth queue and starts waiting.



(b) A parent completes service in the second queue and the next parent enters service, spawning three children.



(c) The first of the waiting children in the second queue gets transferred to the first queue and begins service.



(d) The fourth queue is probed by the third. As the fourth queue has no pending child jobs, no transfer occurs.

Figure 7.2: Example of a child stealing system with $N = 4$. Blue and orange dots depict parent and child jobs respectively.

For $i \in \{(c),(p)\}$, denote by $f^i_{\ell,j,k}(t)$ the fraction of servers with $\ell$ parent jobs in waiting in the queue, $j \in \{0,1,\dots,m\}$ child jobs in the queue and $k \in \{0,1\}$ describing whether a parent is in service ($k = 1$) or not ($k = 0$) at time $t$. Note that $\ell$ does not count parent jobs in service, whereas $j$ counts child jobs waiting and in service. Let $f^i_*(t)$ be the fraction of idle queues at time $t$, that is when $\ell, j, k = 0$. Let $1[A]$ be equal to one if $A$ is true and zero otherwise.

### 7.3.1 Child job stealing

For $\ell \geq 0$ and $j + k \geq 1$,

$$\begin{aligned}
\frac{d}{dt} f^{(c)}_{\ell,j,k}(t) = {} & \lambda f^{(c)}_{\ell-1,j,k}(t)1[\ell \geq 1] + \lambda p_j f^{(c)}_*(t)1[\ell = 0, k = 1] \\
& - \lambda f^{(c)}_{\ell,j,k}(t) + \mu_1 f^{(c)}_{\ell,j,k+1}(t)1[k = 0] \\
& + \mu_1 p_j f^{(c)}_{\ell+1,0,k}(t)1[k = 1] - \mu_1 f^{(c)}_{\ell,j,k}(t)1[k = 1] \\
& + \mu_2 f^{(c)}_{\ell,j+1,k}(t)1[j \leq m-1, k = 0] \\
& + \mu_2 p_j f^{(c)}_{\ell+1,1,k-1}(t)1[k = 1] - \mu_2 f^{(c)}_{\ell,j,k}(t)1[k = 0] \\
& + r f_*(t) f^{(c)}_{\ell,j+1,k}(t)1[j \leq m-1] \\
& - r f_*(t) f^{(c)}_{\ell,j,k}(t)1[j + k > 1] \\
& + r f^{(c)}_*(t) \sum_{\substack{\ell' \geq 0, \\ j'+k' > 1}} f^{(c)}_{\ell',j',k'}(t)1[\ell = 0, j = 1, k = 0]
\end{aligned}$$

and for, $\ell, j, k = 0$,

$$\frac{d}{dt} f_*^{(c)}(t) = -\lambda f_*^{(c)}(t) + \mu_1 f_{0,0,1}^{(c)}(t) + \mu_2 f_{0,1,0}^{(c)}(t) - r f_*^{(c)}(t) \sum_{\substack{\ell \geq 0,\\ j+k>1}} f_{\ell,j,k}^{(c)}(t).$$

The first three terms of the drift of $f_{\ell,j,k}^{(c)}(t)$ correspond to arrivals of parent jobs, the following three terms correspond to service completions of a parent job, the following three terms correspond to service completions of a child job and the remaining terms correspond to job transfers. Similarly for $f_*^{(c)}(t)$, the first term is due to arrivals of parent jobs, the second and third due to parent and child job completions, respectively and the last term is due to child job transfers. Note that

$$\sum_{\ell \geq 0, j+k>1} f_{\ell,j,k}^{(c)}(t) = 1 - f_*^{(c)}(t) - \sum_{\ell \geq 0} (f_{\ell,1,0}^{(c)}(t) + f_{\ell,0,1}^{(c)}(t)),$$

which equals the probability that a probe transmitted at time $t$ succeeds in stealing a child job.

We now rewrite these equations in matrix form, using the vectors below, where $0_i$ is a column vector of zeroes of length $i$, $e_i$ is the $i$-th row of the unit matrix and $\mathbf{1}$ a column vector of ones:

$$\begin{aligned}
f_\ell^{(c)}(t) &= \left(f_{\ell,1,0}^{(c)}(t), \dots, f_{\ell,m,0}^{(c)}(t), f_{\ell,0,1}^{(c)}(t), \dots, f_{\ell,m,1}^{(c)}(t)\right),\\
\alpha &= \begin{bmatrix} 0_m' & p_0 & p_1 & \dots & p_m \end{bmatrix},\\
\mu &= \begin{bmatrix} \mu_2 & 0_{m-1}' & \mu_1 & 0_m' \end{bmatrix}',\\
v_0 &= \begin{bmatrix} 1 & 0_{m-1}' & 1 & 0_m' \end{bmatrix}',
\end{aligned}$$

where $f_\ell^{(c)}(t)$ and $\alpha$ are row vectors of size $2m+1$, while $\mu$ and $v_0$ are column vectors of size $2m+1$. Note that $v_0$ marks the states where $j+k=1$, which are the states where there are no child jobs waiting for service.

We then have for $\ell \geq 0$,

$$\begin{aligned}
\frac{d}{dt} f_\ell^{(c)}(t) &= \lambda f_{\ell-1}^{(c)}(t) 1[\ell \geq 1] - \lambda f_\ell^{(c)}(t) + \lambda f_*^{(c)}(t) \alpha 1[\ell = 0]\\
&\quad + f_\ell^{(c)}(t) S^{(c)}(r,t) + f_{\ell+1}^{(c)}(t) \mu \alpha\\
&\quad + r f_*^{(c)}(t) \sum_{\ell' \geq 0} f_{\ell'}^{(c)}(t)(\mathbf{1} - v_0) e_1 1[\ell = 0], \qquad (7.1)
\end{aligned}$$

and

$$\begin{aligned}
\frac{d}{dt} f_*^{(c)}(t) &= -\lambda f_*^{(c)}(t) + f_0^{(c)}(t)\mu\\
&\quad - r f_*^{(c)}(t) \sum_{\ell \geq 0} f_\ell^{(c)}(t)(\mathbf{1} - v_0). \qquad (7.2)
\end{aligned}$$

The size $(2m+1) \times (2m+1)$ matrix $S^{(c)}(r,t)$ is defined as

$$S^{(c)}(r,t) = \begin{bmatrix} S_{11}^{(c)}(r,t) & S_{12} \\ S_{21} & S_{22}^{(c)}(r,t) \end{bmatrix} \qquad (7.3)$$

with

$$S_{11}^{(c)}(r,t) = \begin{bmatrix} -\mu_2 & & & \\ \mu_2 + rf_*(t) & -(\mu_2 + rf_*(t)) & & \\ & \mu_2 + rf_*(t) & -(\mu_2 + rf_*(t)) & \\ & & & \ddots \end{bmatrix},$$

$$S_{21} = \begin{bmatrix} 0 & \dots & & \\ \mu_1 & & & \\ & \mu_1 & & \\ & & \mu_1 & \\ & & & \ddots \end{bmatrix},$$

$$S_{22}^{(c)}(r,t) = \begin{bmatrix} -\mu_1 & & & \\ rf_*(t) & -(\mu_1 + rf_*(t)) & & \\ & rf_*(t) & -(\mu_1 + rf_*(t)) & \\ & & & \ddots \end{bmatrix},$$

and $S_{12} = 0$ is an $m \times (m+1)$ matrix.

### 7.3.2 Parent job stealing

The ODE model for parent job stealing can be constructed in a very similar fashion as the child job stealing model and we omit the details. Note that

$$\sum_{\ell' \geq 1} f_{\ell'}^{(p)}(t)\mathbf{1} = 1 - f_*^{(p)}(t) - f_0^{(p)}(t)\mathbf{1}$$

equals the probability that a probe transmitted at time $t$ succeeds in stealing a parent job. Hence for $\ell \geq 0$,

$$\begin{aligned} \frac{d}{dt} f_\ell^{(p)}(t) = {} & \lambda f_{\ell-1}^{(p)}(t) 1[\ell \geq 1] - \lambda f_\ell^{(p)}(t) + \lambda f_*^{(p)}(t)\alpha 1[\ell = 0] \\ & + f_\ell^{(p)}(t) S^{(p)} + f_{\ell+1}^{(p)}(t)\mu\alpha + rf_*^{(p)}(t) f_{\ell+1}^{(p)}(t) \\ & - rf_*^{(p)}(t) f_\ell^{(p)}(t) 1[\ell \geq 1] \\ & + rf_*^{(p)}(t)\left(1 - f_*^{(p)}(t) - f_0^{(p)}(t)\mathbf{1}\right)\alpha 1[\ell = 0], \end{aligned} \tag{7.4}$$

and

$$\frac{d}{dt} f_*^{(p)}(t) = -\lambda f_*^{(p)}(t) + f_0^{(p)}(t)\mu - rf_*^{(p)}(t)\left(1 - f_*^{(p)}(t) - f_0^{(p)}(t)\mathbf{1}\right), \tag{7.5}$$

with

$$S^{(p)} = \begin{bmatrix} S_{11}^{(p)} & S_{12} \\ S_{21} & S_{22}^{(p)} \end{bmatrix} \tag{7.6}$$

and

$$S_{11}^{(p)} = \begin{bmatrix} -\mu_2 & & & \\ \mu_2 & -\mu_2 & & \\ & \mu_2 & -\mu_2 & \\ & & & \ddots \end{bmatrix},$$

$$S_{22}^{(p)} = -\mu_1 I,$$

This mean field model is a special case of the one described in [77] by focusing on the phase type distribution characterized by $(\alpha, S^{(p)})$. There are two minor differences in [77]: $f_*^{(p)}(t)$ and $f_\ell^{(p)}(t)$ are denoted as $f_0(t)$ and $f_{\ell+1}(t)$ and the mean service time of a job is assumed to be 1. Note that the latter assumption can be made without loss of generality by rescaling time.

## 7.4 QBD description

The sets of ODEs given by (7.1)-(7.2) and (7.4)-(7.5) describe the transient evolution of the infinite system for the child and parent stealing models, respectively. We now introduce two Quasi-Birth-Death (QBD) Markov chains and show further on that their unique stationary distributions corresponds to the unique fixed points of these two mean field models.

For $i \in \{(c), (p)\}$, we define the QBD process $\{X_t^i(r), Y_t^i(r)\ Z_t^i(r) : t \geq 0\}$, where the *level* is given by $X^i$ and the *phase* is given by $(Y^i, Z^i)$ with generator $Q^i(r)$. Denote by $X^i \geq 0$ the number of parent jobs waiting, $Y^i \in \{0, 1, \ldots, m\}$ the number of child jobs in the queue and $Z^i = \{0, 1\}$ where $Z^i = 1$ if a parent is currently in service and $Z^i = 0$ if not. Define

$$\pi_*^i(r) = \lim_{t\to\infty} P[X_t^i(r) = 0, Y_t^i(r) = 0, Z_t^i(r) = 0],$$

and for $\ell \geq 0$,

$$\pi_\ell^i(r) = \left(\pi_{\ell,1,0}^i(r), \ldots, \pi_{\ell,m,0}^i(r), \pi_{\ell,0,1}^i(r), \ldots, \pi_{\ell,m,1}^i(r)\right),$$

where

$$\pi_{\ell,j,k}^i(r) = \lim_{t\to\infty} P[X_t^i(r) = \ell, Y_t^i(r) = j, Z_t^i(r) = k].$$

### 7.4.1 Child job stealing

The QBD for the child stealing model is very similar to a simple M/PH/1 queue with arrival rate $\lambda$ and phase-type service time characterized by $(\alpha, S^{(c)}(r))$, except that we also have additional job arrivals at some rate $\lambda_c(r)$ (defined later) when the server is idle and these additional arrivals have an exponential service time with parameter $\mu_2$. The subgenerator matrix $S^{(c)}(r)$ is identical to $S^{(c)}(r,t)$ defined by (7.3), if we replace $f_*(t)$ by $q = 1 - \rho$ with

$$\rho = \lambda \left( \frac{1}{\mu_1} + \frac{\sum_{n=1}^m n p_n}{\mu_2} \right).$$

**Proposition 7.4.1.** *The mean $m_{PH} = \alpha(-S^{(c)}(r))^{-1}\mathbf{1}$ of the phase-type distribution characterized by $(\alpha, S^{(c)}(r))$ can be written as*

$$m_{PH} = \frac{\rho}{\lambda} - \frac{1}{\mu_2}\left[\sum_{j=1}^{m}\tilde{p}_j\left(\frac{rq}{rq+\mu_1}\right)^j + \frac{rq}{rq+\mu_2}\sum_{j=2}^{m}\tilde{p}_j\left(1-\left(\frac{rq}{rq+\mu_1}\right)^{j-1}\right)\right], \qquad (7.7)$$

*where $\tilde{p}_j = \sum_{n\geq j} p_n$.*

*Proof.* Using blockwise inversion, $(-S^{(c)}(r))^{-1}$ equals

$$\begin{bmatrix} (-S_{11}^{(c)}(r))^{-1} & 0 \\ (-S_{22}^{(c)}(r))^{-1}S_{21}(-S_{11}^{(c)}(r))^{-1} & (-S_{22}^{(c)}(r))^{-1} \end{bmatrix},$$

where

$$S_{11}^{(c)}(r) = \begin{bmatrix} -\mu_2 & & \\ \mu_2+rq & -\mu_2-rq & \\ & \ddots & \ddots \end{bmatrix},$$

$$S_{22}^{(c)}(r) = \begin{bmatrix} -\mu_1 & & \\ rq & -\mu_1-rq & \\ & \ddots & \ddots \end{bmatrix}.$$

For $i \in \mathbb{N}$, define $d_i = \frac{(rq)^i}{(\mu_1+rq)^{i+1}}$. We then have

$$(-S_{11}^{(c)}(r))^{-1} = \begin{bmatrix} \frac{1}{\mu_2} & & & \\ \vdots & \frac{1}{\mu_2+rq} & & \\ \vdots & \vdots & \ddots & \\ \frac{1}{\mu_2} & \frac{1}{\mu_2+rq} & \cdots & \frac{1}{\mu_2+rq} \end{bmatrix},$$

$$(-S_{22}^{(c)}(r))^{-1} = \begin{bmatrix} \frac{\mu_1+rq}{\mu_1}d_0 & & & \\ \frac{\mu_1+rq}{\mu_1}d_1 & d_0 & & \\ \frac{\mu_1+rq}{\mu_1}d_2 & d_1 & d_0 & \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix},$$

and thus

$$(-S_{22}^{(c)}(r))^{-1}S_{21}(-S_{11}^{(c)}(r))^{-1}$$
$$= \mu_1\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{1}{\mu_2}d_0 & 0 & & \vdots \\ \frac{1}{\mu_2}(d_0+d_1) & \frac{1}{\mu_2+rq}d_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \frac{1}{\mu_2}\sum_{k=0}^{m-1}d_k & \frac{1}{\mu_2+rq}\sum_{k=0}^{m-2}d_k & \cdots & \frac{1}{\mu_2+rq}d_0 \end{bmatrix}.$$

Using the identity that $\mu_1 \sum_{k=0}^{s} d_k = 1 - \left(\frac{rq}{rq+\mu_1}\right)^{s+1}$, we find that $(-S_{22}^{(c)}(r))^{-1}\mathbf{1} = \mathbf{1}/\mu_1$ and

$$(-S_{22}^{(c)}(r))^{-1}S_{21}(-S_{11}^{(c)}(r))^{-1}\mathbf{1} = \begin{bmatrix} 0 \\ \frac{1}{\mu_2}\left(1-\frac{rq}{rq+\mu_1}\right) \\ \frac{1}{\mu_2}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^2\right) + \frac{1}{\mu_2+rq}\left(1-\frac{rq}{rq+\mu_1}\right) \\ \frac{1}{\mu_2}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^3\right) + \frac{1}{\mu_2+rq}\sum_{j=1}^{2}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^j\right) \\ \vdots \\ \frac{1}{\mu_2}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^m\right) + \frac{1}{\mu_2+rq}\sum_{j=1}^{m-1}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^j\right) \end{bmatrix}.$$

As $1/(rq+\mu_2) = \frac{1}{\mu_2}\left(1-\frac{rq}{rq+\mu_2}\right)$, the $(n+1)$-st entry of the vector $(-S_{22}^{(c)}(r))^{-1}S_{21}(-S_{11}^{(c)}(r))^{-1}\mathbf{1}$ can be written as $1/\mu_2$ times

$$\begin{aligned} n - \left(\frac{rq}{rq+\mu_1}\right)^n - (n-1)\frac{rq}{\mu_2+rq} - \left(1-\frac{rq}{\mu_2+rq}\right)\sum_{j=1}^{n-1}\left(\frac{rq}{rq+\mu_1}\right)^j \\ = n - \sum_{j=1}^{n}\left(\frac{rq}{rq+\mu_1}\right)^j - \frac{rq}{\mu_2+rq}\sum_{j=1}^{n-1}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^j\right). \end{aligned}$$

We may therefore conclude that the mean $\alpha(-S^{(c)}(r))^{-1}\mathbf{1}$ equals

$$\begin{aligned} \frac{1}{\mu_1} + \frac{1}{\mu_2}\sum_{n=1}^{m} np_n - \frac{1}{\mu_2}\sum_{n=1}^{m} p_n \sum_{j=1}^{n}\left(\frac{rq}{rq+\mu_1}\right)^j \\ - \frac{1}{\mu_2}\frac{rq}{\mu_2+rq}\sum_{n=2}^{m} p_n \sum_{j=2}^{n}\left(1-\left(\frac{rq}{rq+\mu_1}\right)^{j-1}\right), \end{aligned}$$

which clearly equals (7.7). □

Note that the mean of the phase-type distribution $(\alpha, S^{(c)}(r))$ is upper bounded by $\rho/\lambda$ (and only equal for $r = 0$). This implies that the load of the queue (when ignoring the additional arrivals when the server is idle) is upper bounded by $\rho$. As such it is obvious that this queueing system is stable for all $r \geq 0$ if $\rho < 1$. For completeness we provide a formal proof in Proposition 7.4.2. The possible transitions for this QBD for $i = (c)$ are listed in Table 7.1: 1. a parent job arriving at an idle queue and proceeding directly into service, where any child jobs generated join the queue, 2. a parent arriving to a non-idle queue, 3. completion of a parent in service, not succeeded by another parent job, 4. child service completion, succeeded by either another child job or no job, 5. child service completion, succeeded by a parent job that enters service and any child jobs generated join the queue, 6. parent service completion, succeeded by a parent job that enters service and any child jobs generated join the queue, 7. arrival of a child job due to work stealing and 9. negative arrivals due to work stealing elsewhere.

Table 7.1: Transitions for the QBDs in Section 7.4

| $i \in \{(c),(p)\}$ | From | Rate | For |
|---|---|---|---|
| 1. $(c),(p)$ | $(0,0,0) \to (0,j,1)$ | $\lambda p_j$ | $j = 0,1,\dots m,$ |
| 2. $(c),(p)$ | $(X^i,Y^i,Z^i) \to (X^i+1,Y^i,Z^i)$ | $\lambda$ | $Y^i \geq 1, Z^i = 0$ or $Z^i = 1,$ |
| 3. $(c),(p)$ | $(X^i,Y^i,1) \to (X^i,Y^i,0)$ | $\mu_1$ | $Y^i \geq 1$ or $X^i = 0, Y^i = 0,$ |
| 4. $(c),(p)$ | $(X^i,Y^i,0) \to (X^i,Y^i-1,0)$ | $\mu_2$ | $Y^i \geq 2$ or $X^i = 0, Y^i = 1,$ |
| 5. $(c),(p)$ | $(X^i,1,0) \to (X^i-1,j,1)$ | $\mu_2 p_j$ | $X^i \geq 1, j = 0,1,\dots m,$ |
| 6. $(c),(p)$ | $(X^i,0,1) \to (X^i-1,j,1)$ | $\mu_1 p_j$ | $X^i \geq 1, j = 0,1,\dots m,$ |
| 7. $(c)$ | $(0,0,0) \to (0,1,0)$ | $\lambda_c(r)$ | |
| 8. $(p)$ | $(0,0,0) \to (0,j,1)$ | $\lambda_p(r)p_j$ | $j = 0,1,\dots m,$ |
| 9. $(c)$ | $(X^i,Y^i,Z^i) \to (X^i,Y^i-1,Z^i)$ | $rq^i$ | $Y^i \geq 2, Z^i = 0$ or $Y^i \geq 1, Z^i = 1.$ |
| 10. $(p)$ | $(X^i,Y^i,Z^i) \to (X^i-1,Y^i,Z^i)$ | $rq^i$ | $X^i \geq 1$ |

The generator of the QBD Markov chain has the following form:

$$Q^{(c)}(r) = \begin{bmatrix} -\lambda_0^{(c)}(r) & \lambda_c(r)e_1 + \lambda\alpha & & \\ \mu & A_0^{(c)}(r) & A_1 & \\ & A_{-1}^{(c)} & A_0^{(c)}(r) & A_1 \\ & & \ddots & \ddots \end{bmatrix}.$$

with $\lambda_0^{(c)}(r) = \lambda_c(r) + \lambda$. The size $2m+1$ matrix $A_0^{(c)}(r)$ contains the transitions between states belonging to the same level and is given by

$$A_0^{(c)}(r) = S^{(c)}(r) - \lambda I,$$

where $S^{(c)}(r)$ is identical to $S^{(c)}(r,t)$ with $f_*^{(c)}(t) = q$. Similarly we define $S_{11}^{(c)}(r)$ and $S_{22}^{(c)}(r)$.

The matrices $A_{-1}^{(c)}(r)$ and $A_1$ record the transitions for which the level is decreased and increased by one, respectively. We have

$$A_{-1}^{(c)} = \mu\alpha, \tag{7.8}$$

and

$$A_1 = \lambda I.$$

Denote by $A^{(c)}(r) = A_{-1}^{(c)} + A_0^{(c)}(r) + A_1$, the generator of the phase process, then

$$A^{(c)}(r) = S^{(c)}(r) + \mu\alpha.$$

The physical interpretation of this generator describes that the phase, which captures the mixture of the number of children present in the queue and the type of job in service, can change due to the completion of the current job in service or when a child job is stolen which can only occur when $j + k > 1$.

Due to the QBD structure [60], we have

$$\pi_0^{(c)}(r) = \pi_*^{(c)}(r)R_0^{(c)}(r) \tag{7.9}$$

and for $\ell \geq 1$,

$$\pi_\ell^{(c)}(r) = \pi_0^{(c)}(r) R^{(c)}(r)^\ell, \tag{7.10}$$

where $R^{(c)}(r)$ is a $(2m+1) \times (2m+1)$ matrix and by [48, Proposition 6.4.2] the smallest nonnegative solution to

$$A_1 + R^{(c)}(r) A_0^{(c)}(r) + R^{(c)}(r)^2 A_{-1}^{(c)} = 0.$$

Also,

$$\lambda_c(r) e_1 + \lambda \alpha + R_0^{(c)}(r) A_0^{(c)}(r) + R_0^{(c)}(r) R^{(c)}(r) A_{-1}^{(c)} = 0$$

and

$$A_1 G^{(c)}(r) = R^{(c)}(r) A_{-1}^{(c)},$$

where $G^{(c)}(r)$ is the smallest nonnegative solution to

$$A_{-1}^{(c)} + A_0^{(c)}(r) G^{(c)}(r) + A_1 G^{(c)}(r)^2 = 0.$$

Then

$$R_0^{(c)}(r) = -\left(\lambda_c(r) e_1 + \lambda \alpha\right) \left(A_0^{(c)}(r) + \lambda G^{(c)}(r)\right)^{-1}, \tag{7.11}$$

where $\left(A_0(r) + \lambda G^{(c)}(r)\right)$ is a subgenerator[1] matrix and is therefore invertible. We note that $R^{(c)}(r)$ and $G^{(c)}(r)$ are independent of $\lambda_c(r)$.

The physical interpretation of matrix $G^{(c)}(r)$ is that the $(i,j)$-th entry of the matrix $G^{(c)}(r)$ is the probability that the QBD will first enter level $\ell - 1$ in phase $j$, given that it starts in phase $i$ of level $\ell$. Due to this interpretation we clearly have

$$G^{(c)}(r) = G^{(c)} = \mathbf{1}\alpha.$$

To fully characterize the QBD in terms of $\lambda, \mu_1, \mu_2$ and the probabilities $p_i$, we still need to specify $\lambda_c(r)$. The steal rate $\lambda_c(r)$ is defined as

$$\lambda_c(r) = \frac{\lambda}{q} \left[ \sum_{j=1}^{m} \tilde{p}_j \left( \frac{rq}{rq + \mu_1} \right)^j + \frac{rq}{rq + \mu_2} \sum_{j=2}^{m} \tilde{p}_j \left( 1 - \left( \frac{rq}{rq + \mu_1} \right)^{j-1} \right) \right], \tag{7.12}$$

where $\tilde{p}_j = \sum_{i \geq j} p_i$ is the probability that there are $j$ jobs to be stolen. Note that the expression between brackets is identical to the expression appearing in (7.7). Using probabilistic arguments one finds that the first sum in this expression corresponds to the mean number of child jobs stolen during the service of a parent job, while the second term is the mean number of child jobs that is stolen while a child is in service. The second expression relies on the fact that the number of child jobs stolen after the parent finishes its service has a binomial distribution with parameters $(k-1, rq/(rq+\mu_2))$ if there were $k$ child jobs left when the service of the parent ended.

[1]A matrix is a subgenerator if its diagonal entries are negative, its off-diagonal entries are non-negative and its row sums are negative.

**Proposition 7.4.2.** *The QBD process* $\{X_t^{(c)}(r), Y_t^{(c)}(r), Z_t^{(c)}(r) : t \geq 0\}$ *has a unique stationary distribution for any* $r \geq 0$ *if* $\rho < 1$.

*Proof.* If suffices to check the drift condition for QBD processes [60], which states that the process is positive recurrent if $\theta^{(r)} A_{-1}^{(c)}(r)\mathbf{1} > \theta^{(r)} A_1 \mathbf{1}$, where $\theta^{(r)}$ is the such that $\theta^{(r)} A^{(c)}(r) = 0$ and $A^{(c)}(r) = A_{-1}^{(c)}(r) + A_0^{(c)}(r) + A_1$. Denote

$$\theta^{(r)} = (\theta^{(r)}_{(0,1)}, \ldots, \theta^{(r)}_{(0,m)}, \theta^{(r)}_{(1,0)}, \theta^{(r)}_{(1,1)}, \ldots, \theta^{(r)}_{(1,m)}), \tag{7.13}$$

then

$$\theta^{(r)}_{(0,1)} = \frac{1}{\mu_2} \sum_{j=1}^{m} p_j \left(1 - \left(\frac{rq}{rq + \mu_1}\right)^j\right), \tag{7.14}$$

$$\theta^{(r)}_{(0,i)} = \frac{1}{rq + \mu_2} \sum_{j=i}^{m} p_j \left(1 - \left(\frac{rq}{rq + \mu_1}\right)^{j-i+1}\right), \tag{7.15}$$

for $i = 2, \ldots, m$ and

$$\theta^{(r)}_{(1,0)} = \frac{1}{\mu_1} \sum_{j=0}^{m} p_j \left(\frac{rq}{rq + \mu_1}\right)^j, \tag{7.16}$$

$$\theta^{(r)}_{(1,i')} = \frac{1}{rq + \mu_1} \sum_{j=i'}^{m} p_j \left(\frac{rq}{rq + \mu_1}\right)^{j-i'}, \tag{7.17}$$

for $i' = 1, \ldots, m$. It can be readily verified that $\theta^{(r)} A^{(c)}(r) = 0$. Using these expressions one finds that

$$\theta^{(r)} A_{-1}^{(c)} \mathbf{1} = \theta^{(r)} \mu = 1.$$

By (7.16) and (7.17) we find $\sum_{i=0}^{m} \theta^{(r)}_{(1,i)} = \frac{1}{\mu_1}$, while combining (7.14) and (7.15) yields

$$\begin{aligned}
\sum_{i=1}^{m} \theta^{(r)}_{(0,i)} &= \frac{1}{rq + \mu_2} \sum_{j=2}^{m} j p_j + \frac{1}{\mu_2} p_1 \left(1 - \frac{rq}{rq + \mu_1}\right) \\
&\quad - \frac{1}{\mu_1} \underbrace{\left(\frac{rq + \mu_1}{rq + \mu_2} - \frac{\mu_1}{\mu_2}\right)}_{\geq 0} \sum_{j=2}^{m} p_j \left(1 - \left(\frac{rq}{rq + \mu_1}\right)^j\right) \\
&\leq \frac{1}{\mu_2} \left(\sum_{j=2}^{m} j p_j\right) + \frac{1}{\mu_2} p_1.
\end{aligned}$$

As $A_1 = \lambda I$, this shows that the upward drift $\theta^{(r)} A_1 \mathbf{1}$ is at most $\rho$. □

**Proposition 7.4.3.** *We have* $\pi_*^{(c)}(r) = q$.

*Proof.* Due to (7.9) and (7.10) we have

$$\pi_*^{(c)}(r) = \frac{1}{1 + \sum_{\ell \geq 0} R_0^{(c)}(r)(R^{(c)}(r))^\ell \mathbf{1}}. \tag{7.18}$$

By Proposition 7.4.2 and [48, Proposition 6.4.2], we have $sp(R^{(c)}(r)) < 1$ and $R^{(c)}(r) = A_1(-U(r))^{-1}$ with $U(r) = A_0^{(c)}(r) + A_1 G^{(c)}$. We therefore have

$$\begin{aligned}
\sum_{\ell \geq 0} R_0^{(c)}(r)(R^{(c)}(r))^\ell &= (\lambda_c(r)e_1 + \lambda\alpha)(-U(r))^{-1} \sum_{\ell \geq 0} \lambda^\ell (-U(r))^{-\ell} \\
&= \frac{\lambda_c(r)e_1 + \lambda\alpha}{\lambda} \sum_{\ell \geq 0} \lambda^\ell (-U(r))^{-\ell} - \frac{\lambda_c(r)e_1 + \lambda\alpha}{\lambda} \\
&= \frac{\lambda_c(r)e_1 + \lambda\alpha}{\lambda} (I + \lambda U(r)^{-1})^{-1} - \frac{\lambda_c(r)e_1 + \lambda\alpha}{\lambda},
\end{aligned}$$

where $U(r) = A_0^{(c)}(r) + \lambda \mathbf{1}\alpha$. Using the Woodbury matrix identity [28] we get:

$$(I + \lambda U(r)^{-1})^{-1} = I - \lambda(U(r) + \lambda I)^{-1}$$

and thus:

$$\sum_{\ell \geq 0} R_0^{(c)}(r)(R^{(c)}(r))^\ell = -(\lambda_c(r)e_1 + \lambda\alpha)(U(r) + \lambda I)^{-1}. \tag{7.19}$$

Employing the Sherman-Woodbury formula and the fact that $U(r) + \lambda I = S^{(c)}(r) + \lambda \mathbf{1}\alpha$, we further have

$$-(U(r) + \lambda I)^{-1} = (-S^{(c)}(r))^{-1} + \frac{\lambda(-S^{(c)}(r))^{-1}\mathbf{1}\alpha(-S^{(c)}(r))^{-1}}{1 - \lambda\alpha(-S^{(c)}(r))^{-1}\mathbf{1}}.$$

Let $m_{PH} = \alpha S^{(c)}(r))^{-1}\mathbf{1}$, this implies

$$\begin{aligned}
-\alpha(U(r) + \lambda I)^{-1}\mathbf{1} &= m_{PH} + \frac{\lambda m_{PH}^2}{1 - \lambda m_{PH}} = \frac{m_{PH}}{1 - \lambda m_{PH}}, &(7.20)\\
-e_1(U(r) + \lambda I)^{-1}\mathbf{1} &= \frac{1}{\mu_2} + \frac{1}{\mu_2}\frac{\lambda m_{PH}}{1 - \lambda m_{PH}}, &(7.21)
\end{aligned}$$

as $e_1(-S^{(c)}(r))^{-1}\mathbf{1} = 1/\mu_2$. Combining (7.19), (7.20) and (7.21) yields

$$\begin{aligned}
\sum_{\ell \geq 0} R_0^{(c)}(r)(R^{(c)}(r))^\ell \mathbf{1} &= \frac{\lambda_c(r)}{\mu_2}\left(1 + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}}\right) + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}} \\
&= \frac{\lambda}{q}\left(\frac{\rho}{\lambda} - m_{PH}\right)\left(1 + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}}\right) + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}} \\
&= \frac{\rho}{q}\left(1 + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}}\right) - \frac{\lambda}{q}\frac{m_{PH}}{1 - \lambda m_{PH}} + \frac{\lambda m_{PH}}{1 - \lambda m_{PH}} \\
&= \frac{1 - q}{q},
\end{aligned}$$

where the second equality follows from (7.12) and (7.7). The result now follows from (7.18). □

By using (7.11) and $G^{(c)} = \mathbf{1}\alpha$, we now also get an explicit formula for $\pi_0^{(c)}(r)$.

### 7.4.2 Parent job stealing

The QBD process for the system with parent job stealing is nearly identical to the one used in [77]. Compared to the QBD for the system with child job stealing, this queue is not similar to an M/PH/1 queue. Instead it corresponds to an M/PH/1 queue subject to negative arrivals (when the queue has pending jobs) and these correspond to parent jobs that are stolen.

The generator of the process $\{X_t^{(p)}(r), Y_t^{(p)}(r), Z_t^{(p)}(r)\}$ is

$$Q^{(p)}(r) = \begin{bmatrix} -\lambda_0^{(p)}(r) & (\lambda + \lambda_p(r))\alpha & & \\ \mu & B_0^{(p)} & A_1 & \\ & A_{-1}^{(p)}(r) & A_0^{(p)}(r) & A_1 \\ & & \ddots & \ddots \end{bmatrix},$$

with $\lambda_0^{(p)}(r) = \lambda + \lambda_p(r)$,

$$A_{-1}^{(p)}(r) = A_{-1}^{(c)} + rqI,$$

where $A_{-1}^{(c)}$ is given in Equation (7.8) and

$$A_0^{(p)}(r) = S^{(p)} - \lambda I - rqI,$$

where $S^{(p)}$ is given by (7.6) and

$$B_0^{(p)} = S^{(p)} - \lambda I.$$

We have

$$\pi_0^{(p)}(r) = \pi_*^{(p)}(r) R_0^{(p)}(r)$$

and for $\ell \geq 1$,

$$\pi_\ell^{(p)}(r) = \pi_0^{(p)}(r) R^{(p)}(r)^\ell,$$

where

$$R_0^{(p)}(r) = -(\lambda + \lambda_p(r))\alpha \left(B_0^{(p)} + A_1 G^{(p)}(r)\right)^{-1},$$

and $R^{(p)}(r)$ is the smallest nonnegative solution to

$$A_1 + R^{(p)}(r) A_0^{(p)}(r) + R^{(p)}(r)^2 A_{-1}^{(p)} = 0.$$

As in [77] we still need to set the value of $\lambda_p(r)$ to fully characterize the QBD. This value is determined by demanding that $\pi_*^{(p)}(r) = q = 1 - \rho$, which yields

$$\lambda_0^{(p)}(r) = \frac{\rho}{q\alpha(\lambda(I - G^{(p)}(r)) - S^{(p)})^{-1}(I - R^{(p)}(r))^{-1}\mathbf{1}} - \lambda,$$

while the steady state probabilities of this system are given by

$$\pi_*^{(p)}(r) = q,$$
$$\pi_\ell^{(p)}(r) = \rho \frac{\alpha(\lambda(I - G^{(p)}(r)) - S^{(p)})^{-1} R^{(p)}(r)^\ell}{\alpha(\lambda(I - G^{(p)}(r)) - S^{(p)})^{-1}(I - R^{(p)}(r))^{-1}\mathbf{1}},$$

for $\ell \geq 0$. Contrary to the case with child job stealing, the matrices $R^{(p)}(r)$ and $G^{(p)}(r)$ must be determined numerically by solving a non-linear matrix equation.

While there is no explicit expression for $\pi_\ell(r)$ as it is a function of $R(r)$ (or $G(r)$), these matrices can be computed very efficiently using matrix analytic methods [5].

## 7.5 Stationary behaviour

In this section we show that the stationary distribution of the child stealing QBD in Section 7.4 corresponds to the unique fixed point $\zeta^i$ of the set of ODEs. For the case of parent job stealing, we illustrate how the results by [77] can be modified to obtain the desired result. Define for $i \in \{C, P\}$, $\zeta^i = (\zeta^i_*, \zeta^i_0, \zeta^i_1, \dots)$ with $\zeta^i_* + \sum_{\ell \geq 0} \zeta^i_\ell \mathbf{1} = 1$.

### 7.5.1 Child job stealing

**Lemma 7.5.1.** *For any fixed point $\zeta^{(c)} = (\zeta^{(c)}_*, \zeta^{(c)}_0, \zeta^{(c)}_1, \dots)$ with $\zeta^{(c)}_* + \sum_{\ell \geq 0} \zeta^{(c)}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (7.1)-(7.2) *we have*

$$\lambda = \lambda \zeta^{(c)}_* + \sum_{\ell \geq 1} \zeta^{(c)}_\ell \mu. \tag{7.22}$$

*Proof.* As $\frac{d}{dt} f^{(c)}_\ell(t) = 0$ in a fixed point we can show that

$$\sum_{\ell \geq 0} \zeta^{(c)}_\ell \begin{pmatrix} 0_m \\ \mu_1 \\ 0_m \end{pmatrix} + \sum_{\ell \geq 0} \zeta^{(c)}_\ell \begin{pmatrix} \mu_2 \\ 0_m \\ 0_m \end{pmatrix} = \lambda + r\zeta^{(c)}_* \sum_{\ell \geq 0} \zeta^{(c)}_\ell (\mathbf{1} - v_0), \tag{7.23}$$

by using the equality $\sum_{\ell \geq 0} (\ell + 1) \frac{d}{dt} f^{(c)}_\ell(t) \mathbf{1} = 0$. (7.22) now follows by combining this equality with $\frac{d}{dt} f^{(c)}_*(t) = 0$. □

**Lemma 7.5.2.** *For any fixed point $\zeta^{(c)} = (\zeta^{(c)}_*, \zeta^{(c)}_0, \zeta^{(c)}_1, \dots)$ with $\zeta^{(c)}_* + \sum_{\ell \geq 0} \zeta^{(c)}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (7.1)-(7.2) *we have for $1 \leq k \leq m$*

$$r\zeta^{(c)}_* \sum_{\ell \geq 0} \zeta^{(c)}_\ell \begin{pmatrix} 0_{m+k} \\ 1_{m-k+1} \end{pmatrix} = \lambda \sum_{j=k}^{m} \tilde{p}_j \left( \frac{r\zeta^{(c)}_*}{r\zeta^{(c)}_* + \mu_1} \right)^{j-k+1}, \tag{7.24}$$

*where $1_i$ denotes a column vector of i ones.*

*Proof.* We use backward induction on $k$ to prove this result. By demanding that

$$\sum_{\ell\geq 0} \frac{d}{dt} f_\ell^{(c)}(t) \begin{pmatrix} 0_{m+k} \\ 1_{m-k+1} \end{pmatrix} = 0$$

for any $k \in \{1, \ldots, m\}$, we find due to Lemma 7.5.1 that

$$\lambda \tilde{p}_k = r\zeta_*^{(c)} \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+k} \\ 1 \\ 0_{m-k} \end{pmatrix} + \mu_1 \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+k} \\ 1_{m-k+1} \end{pmatrix},$$

which is equivalent to (7.24) when $k = m$. For $k < m$ we can rewrite the above as

$$\lambda \tilde{p}_k = (r\zeta_*^{(c)} + \mu_1) \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+k} \\ 1_{m-k+1} \end{pmatrix} - r\zeta_*^{(c)} \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+k+1} \\ 1_{m-k} \end{pmatrix},$$

and use induction on the second term. This yields

$$\lambda \tilde{p}_k = (r\zeta_*^{(c)} + \mu_1) \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+k} \\ 1_{m-k+1} \end{pmatrix} - \lambda \sum_{j=k+1}^{m} \tilde{p}_j \left( \frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)} + \mu_1} \right)^{j-k}.$$

This expression is clearly equivalent to (7.24). □

**Proposition 7.5.3.** *For any fixed point $\zeta^{(c)} = (\zeta_*^{(c)}, \zeta_0^{(c)}, \zeta_1^{(c)}, \ldots)$ with $\zeta_*^{(c)} + \sum_{\ell\geq 0} \zeta_\ell^{(c)} \mathbf{1} = 1$ of the set of ODEs in Equations* (7.1)-(7.2) *we have*

$$\zeta_*^{(c)} = q, \tag{7.25}$$

$$r\zeta_*^{(c)} \sum_{\ell\geq 0} \zeta_\ell^{(c)} (\mathbf{1} - v_0) = \zeta_*^{(c)} \lambda_c(r), \tag{7.26}$$

*where $\lambda_c(r)$ was defined in* (7.12).

*Proof.* We denote 1:$i$ for the column vector of length $i$ with the $j$-th entry equal to $j$. To establish that $\zeta_*^{(c)} = q$ it suffices to establish the following two identities:

$$\sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_m \\ 1_{m+1} \end{pmatrix} = \frac{\lambda}{\mu_1}, \tag{7.27}$$

$$\sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 1_m \\ 0_{m+1} \end{pmatrix} = \frac{\lambda}{\mu_2} \left( \sum_{i=1}^{m} i p_i \right). \tag{7.28}$$

As $\sum_{\ell\geq 0} \frac{d}{dt} f_\ell^{(c)}(t) \begin{pmatrix} 1_m \\ 0_{m+1} \end{pmatrix} = 0$, we find

$$\sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} \mu_2 \\ 0_m \\ 0_m \end{pmatrix} = r\zeta_*^{(c)} \sum_{\ell\geq 0} \zeta_\ell^{(c)} (\mathbf{1} - v_0) + \sum_{\ell\geq 0} \zeta_\ell^{(c)} \begin{pmatrix} 0_{m+1} \\ \mu_1 1_m \end{pmatrix}. \tag{7.29}$$

Combining (7.23) and (7.29) yields (7.27). From $\sum_{\ell\geq 0}\frac{d}{dt}f_\ell^{(c)}(t)[(1{:}m)'\ 0\ (1{:}m)']'=0$ and Lemma 7.5.1, one can show that

$$\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}\mu_2 1_m\\ 0_{m+1}\end{pmatrix}=\lambda\left(\sum_{i=1}^m ip_i\right),$$

which is equivalent to (7.28).

Proving (7.26) requires more work. We prove the following two equalities that together provide us with the required result:

$$r\zeta_*^{(c)}\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}0_{m+1}\\ 1_m\end{pmatrix}=\lambda\sum_{j=1}^m\tilde{p}_j\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^j,$$
$$(r\zeta_*^{(c)}+\mu_2)\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}0\\ 1_{m-1}\\ 0_{m+1}\end{pmatrix}=\lambda\sum_{j=2}^m\tilde{p}_j\left(1-\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^{j-1}\right). \tag{7.30}$$

The first is immediate from Lemma 7.5.2 if we set $k=1$. To establish the second equality, we first note that $\sum_{\ell\geq 0}\frac{d}{dt}f_\ell^{(c)}(t)[0\ (1{:}(m-1))'\ 0'_{m+1}]'=0$ allows us to show that

$$(r\zeta_*^{(c)}+\mu_2)\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}0\\ 1_{m-1}\\ 0_{m+1}\end{pmatrix}=\mu_1\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}0_{m+2}\\ 1{:}(m-1)\end{pmatrix}. \tag{7.31}$$

Clearly, we have

$$\begin{pmatrix}0_{m+2}\\ 1{:}(m-1)\end{pmatrix}=\sum_{k=2}^m\begin{pmatrix}0_{m+k}\\ 1_{m-k+1}\end{pmatrix}.$$

Combining this with (7.31) and Lemma 7.5.2 for $k=2$ to $m$ we find

$$\begin{aligned}(r\zeta_*^{(c)}+\mu_2)\sum_{\ell\geq 0}\zeta_\ell^{(c)}\begin{pmatrix}0\\ 1_{m-1}\\ 0_{m+1}\end{pmatrix}&=\frac{\lambda\mu_1}{r\zeta_*^{(c)}}\sum_{k=2}^m\sum_{j=k}^m\tilde{p}_j\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^{j-k+1}\\
&=\frac{\lambda\mu_1}{r\zeta_*^{(c)}}\sum_{j=2}^m\tilde{p}_j\sum_{s=1}^{j-1}\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^s\\
&=\frac{\lambda\mu_1}{r\zeta_*^{(c)}+\mu_1}\sum_{j=2}^m\tilde{p}_j\frac{1-\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^{j-1}}{1-\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)}\\
&=\lambda\sum_{j=2}^m\tilde{p}_j\left(1-\left(\frac{r\zeta_*^{(c)}}{r\zeta_*^{(c)}+\mu_1}\right)^{j-1}\right),\end{aligned}$$

which proves (7.30). □

**Theorem 7.5.4.** *The stationary distribution $\pi^{(c)}(r)$ of the QBD Markov chain characterized by $Q^{(c)}(r)$ is the unique fixed point $\zeta^{(c)}$ of the set of ODEs in Equations* (7.1)-(7.2).

*Proof.* Using Proposition 7.5.3 we show that the fixed point equations $\frac{d}{dt}f_\ell^{(c)}(t) = 0$ are equivalent to the balance equations of the QBD Markov chain characterized by $Q^{(c)}(r)$. The uniqueness of the fixed point the follows from the uniqueness of the stationary distribution of the Markov chain.

For $\ell \geq 1$, $\frac{d}{dt}f_\ell^{(c)}(t) = 0$ can be written as

$$0 = \zeta_{\ell-1}^{(c)}(\lambda I) + \zeta_\ell^{(c)}\left(S^{(c)}(r,t) - \lambda I\right) + \zeta_{\ell+1}^{(c)}\mu\alpha,$$

which is exactly the balance equations of $Q^{(c)}(r)$ for $\ell \geq 1$ as $\zeta_*^{(c)} = q$ due to Proposition 7.5.3. This implies that $\zeta_\ell^{(c)} = \zeta_0^{(c)}R(r)^\ell$, for all $\ell \geq 1$ for any fixed point.

For $\ell = 0$, $\frac{d}{dt}f_\ell^{(c)}(t) = 0$ implies

$$0 = \zeta_0^{(c)}(S(t) - \lambda I) + \zeta_1\mu\alpha + \lambda\zeta_*^{(c)}\alpha + r\zeta_*^{(c)}\sum_{\ell' \geq 0}\zeta_{\ell'}^{(c)}(\mathbf{1} - v_0)e_1.$$

Due to Proposition 7.5.3 we can rewrite this as

$$0 = \zeta_0^{(c)}A_0(r) + \zeta_1^{(c)}A_{-1} + q\left(\lambda_c(r)e_1 + \lambda\alpha\right).$$

This indicates that $\frac{d}{dt}f_\ell^{(c)}(t) = 0$ corresponds to the balance equation for $\ell = 0$. Finally one readily checks that $\frac{d}{dt}f_*^{(c)}(t) = 0$ is equivalent to the first balance equation due to (7.26). □

## 7.5.2 Parent job stealing

**Theorem 7.5.5.** *The stationary distribution $\pi^{(p)}(r)$ of the QBD Markov chain characterized by $Q^{(p)}(r)$ is the unique fixed point $\zeta^{(p)}$ of the set of ODEs in Equations* (7.4)-(7.5).

*Proof.* Define $\theta^{(p)} = (\theta_1^{(p)}, \ldots, \theta_{2m+1}^{(p)})$ with $\theta_j^{(p)} = \frac{1}{\mu_2}\tilde{p}_j$ for $j = 1, \ldots, m$ and $\theta_j^{(p)} = \frac{1}{\mu_1}p_{j-m-1}$ for $j = m+1, \ldots, 2m+1$. We then have $\theta^{(p)}(S^{(p)} + \mu\alpha) = 0$. Therefore

$$\beta^{(p)} = \frac{1}{\sum_{j=1}^{2m+1}\theta_j^{(p)}}\theta^{(p)} = \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\sum_{j=1}^m jp_j\right)^{-1}\theta^{(p)}$$

is the stationary distribution of the service phase given the server is busy. We also have

$$\beta^{(p)}\mu = \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\sum_{j=1}^m jp_j\right)^{-1}.$$

One can now make the same calculations as in [77, Proposition 1] to conclude that

$$\sum_{\ell \geq 0}\zeta_\ell^{(p)} = \lambda\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\sum_{j=1}^m jp_j\right)\beta^{(p)} = \rho\beta^{(p)}.$$

Table 7.2: Non-zero entries of the rate matrix $\tilde{S}^{(c)}(r)$

| | From | Rate | For |
|---|---|---|---|
| 1. | $(Y^{(c)}, Z^{(c)}, \tilde{Y}^{(c)}) \rightarrow (Y^{(c)}-1, Z^{(c)}, \tilde{Y}^{(c)})$ | $\mu_2$ | $Y^{(c)} \geq 1, Z^{(c)} = 0$ |
| 2. | $(Y^{(c)}, Z^{(c)}, \tilde{Y}^{(c)}) \rightarrow (Y^{(c)}, Z^{(c)}-1, \tilde{Y}^{(c)})$ | $\mu_1$ | $Z^{(c)} = 1$ |
| 3. | $(Y^{(c)}, Z^{(c)}, \tilde{Y}^{(c)}) \rightarrow (Y^{(c)}, Z^{(c)}, \tilde{Y}^{(c)}-1)$ | $\mu_2\tilde{Y}^{(c)}$ | $\tilde{Y}^{(c)} \geq 1$ |
| 4. | $(Y^{(c)}, Z^{(c)}, \tilde{Y}^{(c)}) \rightarrow (Y^{(c)}-1, Z^{(c)}, \tilde{Y}^{(c)}+1)$ | $rq^{(c)}$ | $Y^{(c)} + Z^{(c)} \geq 2$ |

As $\beta^{(p)}$ is a stochastic vector, we get

$$\zeta_*^{(p)} = 1 - \rho\beta^{(p)}\mathbf{1} = q.$$

The rest of the proof is identical to the one of [77, Theorem 1], except with $q$ instead of $1-\lambda$. □

## 7.6 Response time distribution

Define $T^i(r)$ as the response time for a job with probe rate $r$, in a system where only type $i \in \{(p),(c)\}$ jobs can be transferred. The response time is the interval of time between the arrival epoch of a parent job and the instant at which the parent and all of its child jobs have completed service. This can be expressed as

$$T^i(r) = W^i(r) + J^i(r),$$

where $W^i(r)$ is waiting time defined as the interval between the arrival epoch of a parent job and instant at which it moves into service. In the case that parent jobs are stolen, we assume that the oldest waiting parent job is stolen (as this should be best to reduce the variability of the waiting time). The service time $J^i(r)$ is defined as the time between the start of service of the parent job and the first point in time in which both the parent and all of its children completed service. Clearly $W^i(r)$ and $J^i(r)$ are independent. Note that $W^i(r)$ is harder to compute in case parent jobs are stolen, while $J^i(r)$ is more demanding when child jobs can be transferred.

### 7.6.1 Waiting time distribution

We present a unified analysis for both models. Due to the PASTA property we have $P[W^i(r) = 0] = q$. To compute $P[W^i(r) > t]$ we employ the approach taken in Ozawa [61] and Horvath *et al.* [39]. Ozawa [61] studied FIFO queues defined by a QBD Markov chain where transitions that increase/decrease the level are regarded as arrivals/departures. Ozawa showed that the sojourn time distribution (the time between an arrival and its departure) of a queue defined by a QBD has a matrix exponential form (of order $n^2$ if we have $n$ phases per level). A similar result is presented below for the waiting time $W^i(r)$.

**Theorem 7.6.1.** *For $i \in \{(p),(c)\}$, the distribution of the waiting time is given by*

$$P[W^i(r) > t] = (\mathbf{1}' \otimes \pi_0^i(I - R^i(r))^{-1})e^{\mathbb{W}^i t}vec\langle I\rangle \tag{7.32}$$

*with* $\mathbb{W}^i = ((A_0^i(r) + A_1)' \otimes I) + ((A_{-1}^i(r))' \otimes R^i(r))$ *and where* $vec\langle\cdot\rangle$ *is the column stacking operator. The mean waiting time is*

$$\begin{aligned} E\left[W^i(r)\right] &= \int_0^\infty P\left[W^i(r) > t\right] dt \\ &= (\mathbf{1}' \otimes \pi_0^i(I - R^i(r))^{-1})(-\mathbb{W}^i)^{-1} vec\langle I\rangle. \end{aligned} \tag{7.33}$$

*Proof.* Let $(N^i(k,t))_{j,j'}$ be the probability that we have exactly $k$ transitions that decrease the level by one in $(0,t)$ and the phase at time $t$ equals $j'$ for the QBD $Q^i$ given that the level never decreased below 1 and the phase was $j$ at time 0. Due to the PASTA property we have

$$P[W^i(r) > t] = \sum_{n=1}^{\infty} \pi_{n-1}^i \sum_{k=0}^{n-1} N^i(k,t)\mathbf{1},$$

as $(\pi_{n-1}^i)_j$ is the probability that a tagged parent job is the $n$-th parent job waiting in the queue immediately after it arrived and the service phase equals $j$. In such case there can be at most $n-1$ events that decrease the level otherwise $W^i(r) < t$. Thus,

$$\begin{aligned} P[W^i(r) > t] &= \sum_{k=0}^{\infty} \pi_0^i \sum_{n=k+1}^{\infty} (R^i(r))^{n-1} N^i(k,t)\mathbf{1}, \\ &= \pi_0^i (I - R^i(r))^{-1} \sum_{k=0}^{\infty} (R^i(r))^k N^i(k,t)\mathbf{1}. \end{aligned}$$

Using the same arguments as in [61] or [39] one finds that

$$vec\langle \sum_{k=0}^{\infty} (R^i(r))^k N^i(k,t)\rangle = e^{\mathbb{W}^i t} vec\langle I\rangle.$$

The proof is completed by noting that $vec\langle ABC\rangle = (C' \otimes A) vec\langle B\rangle$ (cf. Proposition 3.6.7). □

### 7.6.2 Service distribution

When parent jobs are stolen, a parent job and all its child jobs are executed on the same server. Hence, the service time $J^{(p)}$ has a phase type distribution with parameters $(\alpha, S^{(p)})$:

$$P[J^{(p)} < t] = 1 - \alpha e^{-S^{(p)} t}\mathbf{1},$$

and $E\left[J^{(p)}\right] = \alpha\left(-S^{(p)}\right)^{-1}\mathbf{1}$. We have from (7.32) that the waiting time distribution $W^i(r)$ follows a matrix exponential distribution with parameters $(\mathbf{1}' \otimes \pi_0^i(I - R^i(r))^{-1}, \mathbb{W}^i, (-\mathbb{W}^i) vec\langle I\rangle)$. Therefore due to [6, Theorem 4.4.2] the convolution of the waiting time and service time can be expressed as

$$P[T^{(p)}(r) > t] = \begin{bmatrix} \mathbf{1}' \otimes \pi_0^{(p)}(I - R^{(p)}(r))^{-1} & q\alpha \end{bmatrix} e^{\mathbb{T}^{(p)} t} (-\mathbb{T}^{(p)})^{-1} \begin{pmatrix} 0_{2m+1} \\ \mu \end{pmatrix},$$

where

$$\mathbb{T}^{(p)} = \begin{bmatrix} \mathbb{W}^{(p)} & (-\mathbb{W}^{(p)})vec\langle I\rangle\alpha \\ 0 & S^{(p)} \end{bmatrix},$$

and

$$E[T^{(p)}(r)] = \left[\mathbf{1}' \otimes \pi_0^{(p)}(I - R^{(p)}(r))^{-1} \quad q\alpha\right](\mathbb{T}^{(p)})^{-2}\begin{pmatrix} 0_{2m+1} \\ \mu \end{pmatrix}.$$

Note, that we can also simply calculate $E[T^{(p)}(r)]$ as the sum of (7.33) and $\alpha\left(-S^{(p)}\right)^{-1}\mathbf{1}$.

When child jobs are stolen, we need to keep track of the number of transferred child jobs as such a child job may be the last to complete service. To this end, we define the phase process $\{Y_t^{(c)}(r), Z_t^{(c)}(r), \tilde{Y}_t^{(c)}(r)\}_{t\geq 0}$ where $Y_t^{(c)}(r), Z_t^{(c)}(r)$ are defined as before and $\tilde{Y}_t^{(c)} \in \{0, 1, \ldots, m\}$ is the number of transferred child jobs still in service.

The service time $J^{(c)}(r)$ of a parent job with $n$ child jobs therefore equals the time needed for the phase process to go from phase $(n, 1, 0)$ to $(0, 0, 0)$. In other words $J^{(c)}(r)$ can be represented as a phase-type distribution with parameters $(\tilde{\alpha}^{(c)}, \tilde{S}^{(c)}(r))$. As $Z_t^{(c)}(r) \in \{0, 1\}$, $0 \leq \tilde{Y}_t^{(c)}(r) + Y_t^{(c)}(r) \leq m$ and $(0, 0, 0)$ is the absorbing state $\tilde{S}^{(c)}(r)$ is a size $d = 2\sum_{k=1}^{m+1} k - 1 = m^2 + 3m + 1$ matrix. Its non-zero entries are listed in Table 7.2. The vector $\tilde{\alpha}^{(c)}$ equals $p_n$ in the position corresponding to phase $(n, 1, 0)$ and equals zero for any other phase. Then

$$P[J^{(c)}(r) < t] = 1 - \tilde{\alpha}^{(c)}e^{-\tilde{S}^{(c)}(r)t}\mathbf{1},$$

and $E\left[J^{(c)}(r)\right] = \tilde{\alpha}^{(c)}\left(-\tilde{S}^{(c)}(r)\right)^{-1}\mathbf{1}$. Let $\tilde{\mu}$ be a column vector of height $d$, with $\mu_2$ in entries corresponding to states $(1, 0, 0)$ and $(0, 0, 1)$, with $\mu_1$ in entry corresponding to state $(0, 1, 0)$ and with all other entries zero. The convolution of $W^{(c)}(r)$ and $J^{(c)}(r)$ can be then computed in a similar manner as in the parent job stealing case. We namely have:

$$P[T^{(c)}(r) > t] = \left[\mathbf{1}' \otimes \pi_0^{(c)}(I - R^{(c)}(r))^{-1} \quad q\tilde{\alpha}^{(c)}\right]e^{\mathbb{T}^{(c)}t}(-\mathbb{T}^{(c)})^{-1}\begin{pmatrix} 0_{2m+1} \\ \tilde{\mu} \end{pmatrix},$$

where

$$\mathbb{T}^{(c)} = \begin{bmatrix} \mathbb{W}^{(c)} & (-\mathbb{W}^{(c)})vec\langle I\rangle\tilde{\alpha}^{(c)} \\ 0 & \tilde{S}^{(c)} \end{bmatrix},$$

and

$$E[T^{(c)}(r)] = \left[\mathbf{1}' \otimes \pi_0^{(c)}(I - R^{(c)}(r))^{-1} \quad q\tilde{\alpha}^{(c)}\right](\mathbb{T}^{(c)})^{-2}\begin{pmatrix} 0_{2m+1} \\ \tilde{\mu} \end{pmatrix}.$$

## 7.7 Probe rate $r \longrightarrow \infty$

**Child job stealing** Taking $r \longrightarrow \infty$, we define $\lambda_c = \lim_{r\to\infty}\lambda_c(r)$, then by Equation (7.12),

$$\lambda_c = \frac{\lambda}{q}\sum_{j=1}^{m} jp_j.$$

The resulting process is given by the QBD $\{X^{(c)}, Z^{(c)}\}$ as defined in Section 7.4, with $X \geq 0$ and $Z \in \{0,1\}$, noting the empty boundary state is distinct from the state $(X^{(c)}, Z^{(c)}) = (0,0)$, in which a child job is in service. The rate matrix is

$$Q_{\infty}^{(c)} = \begin{bmatrix} -\lambda - \lambda_c & \lambda_c e_1 + \lambda e_2 & & \\ \mu_\infty & A_0^\infty & A_1^\infty & \\ & A_{-1}^\infty & A_0^\infty & A_1^\infty \\ & & \ddots & \ddots \end{bmatrix},$$

where $\mu_\infty = [\mu_2, \mu_1]'$, $A_{-1}^\infty = [0_2, \mu_\infty]$, $A_0^\infty = -\mathrm{diag}([\mu_2 + \lambda, \mu_1 + \lambda])$ and $A_1^\infty = \lambda I$. Proceeding similarly to section 7.4.1, we find

$$\begin{aligned} G_\infty^{(c)} &= \begin{bmatrix} 0_2 & 1_2 \end{bmatrix}, \\ R_{0,\infty}^{(c)} &= \begin{bmatrix} \frac{\lambda_c}{\mu_2+\lambda} & \frac{\lambda}{\mu_1}\left(1 + \frac{\lambda_c}{\mu_2+\lambda}\right) \end{bmatrix}. \end{aligned}$$

Using [48, Proposition 6.4.2], we get

$$R_\infty^{(c)} = \lambda \begin{bmatrix} \frac{1}{\mu_2+\lambda} & \frac{\lambda}{\mu_1(\mu_2+\lambda)} \\ 0 & \frac{1}{\mu_1} \end{bmatrix}. \tag{7.34}$$

Note that $(I - R_\infty^{(c)})$ is invertible as $\lambda < \mu_1$. (Note that this can be shown algebraically: $(I - R_\infty^{(c)})$ is not invertible only when $\lambda = \mu_1$.) Directly from (7.34), we get for $k \geq 1$

$$\pi_{k,0} = \pi_{0,0}\left(\frac{\lambda}{\mu_2 + \lambda}\right)^k.$$

According to the PASTA property,

$$\begin{aligned} E[W^{(c)}] &= \frac{1}{\mu_2}\sum_{k=0}^{\infty}\pi_{k,0} + \frac{1}{\mu_1}\sum_{k=0}^{\infty}\pi_{k,1} + \frac{1}{\mu_1}\sum_{k=1}^{\infty}k(\pi_{k,0} + \pi_{k,1}), \\ &= \left(\frac{1}{\mu_2} - \frac{1}{\mu_1}\right)\pi_{0,0}\sum_{k=0}^{\infty}\left(\frac{\lambda}{\mu_2+\lambda}\right)^k + \frac{1}{\mu_1}\sum_{k=0}^{\infty}(k+1)[\pi_{k,0}\ \pi_{k,1}]\mathbf{1}, \\ &= \left(\frac{1}{\mu_2} - \frac{1}{\mu_1}\right)\pi_{0,0}\frac{\mu_2+\lambda}{\mu_2} + \frac{1}{\mu_1}[\pi_{0,0}\ \pi_{0,1}]\sum_{k=0}^{\infty}(k+1)\left(R_\infty^{(c)}\right)^k\mathbf{1}, \\ &= \left(\frac{1}{\mu_2} - \frac{1}{\mu_1}\right)\pi_{0,0}\frac{\mu_2+\lambda}{\mu_2} + \frac{1}{\mu_1}[\pi_{0,0}\ \pi_{0,1}]\left(I - R_\infty^{(c)}\right)^{-2}\mathbf{1}. \end{aligned}$$

With $\pi_0 = \pi_* R_{0,\infty}^{(c)} = q R_{0,\infty}^{(c)}$,

$$E[W^{(c)}] = \left(\frac{1}{\mu_2} - \frac{1}{\mu_1}\right)\frac{q\lambda_c}{\mu_2} + \frac{q}{\mu_1}R_{0,\infty}^{(c)}\left(I - R_\infty^{(c)}\right)^{-2}\mathbf{1}, \tag{7.35}$$

where

$$R_{0,\infty}^{(c)}\left(I - R_\infty^{(c)}\right)^{-2} = \begin{bmatrix} \frac{\lambda_c(\mu_2+\lambda)}{(\mu_2)^2} & \frac{\lambda_c\lambda^2}{(\mu_2)^2(\mu_1-\lambda)} + \frac{\lambda_c\lambda^2\mu_1 + \lambda\mu_2\mu_1(\mu_2+\lambda+\lambda_c)}{\mu_2(\mu_1-\lambda)^2(\mu_2+\lambda)} \end{bmatrix}.$$

Thus limit of the waiting time depends on the mean of $\check{p}$ via $\lambda_c$. The limit of the mean response time is $E[T^{(c)}] = E[W^{(c)}] + \tilde{\alpha}^{(c)}(-\tilde{S}^{(c)})^{-1}e$, where $\tilde{S}^{(c)} = \lim_{r\to\infty} \tilde{S}^{(c)}(r)$.

In the limit, the service time is the maximum of set of exponentials and the expected service time is

$$E[J^{(c)}] = \sum_{k=0}^{m} p_k J_k,$$

where $J_k$, $k = 0, \dots, m$, as the average time it takes for a parent job that will spawn $k$ child jobs to be processed. Obviously, we have $J_0 = \frac{1}{\mu_1}$. For $J_k$, with $k \geq 1$, let $J^p \sim \exp(\mu_1)$, $J^1, \dots, J^m \sim \exp(\mu_2)$ be independent random variables. We will provide a recursive formula for $J_k$ using the following facts:

$$E[\min(J^p, J^1, \dots, J^k)] = \frac{1}{\mu_1 + k\mu_2},$$
$$P[J^p < \min(J^1, \dots, J^k)] = \frac{\mu_1}{\mu_1 + k\mu_2}.$$

We now define recursively, for $k \geq 1$:

$$\begin{aligned} J_k &= \frac{1}{\mu_1 + k\mu_2} + \frac{\mu_1}{\mu_1 + k\mu_2} E[\max(J^1, \dots, J^k)] + \frac{k\mu_2}{\mu_1 + k\mu_2} E[\max(J^p, J^1, \dots, J^{k-1})], \\ &= \frac{1}{\mu_1 + k\mu_2} + \frac{\mu_1}{\mu_1 + k\mu_2}\frac{1}{\mu_2}\sum_{j=1}^{k}\frac{1}{j} + \frac{k\mu_2}{\mu_1 + k\mu_2} J_{k-1}. \end{aligned} \tag{7.36}$$

**Proposition 7.7.1.** *The worst case for the service time is for a fixed number of child jobs* $d = 1, 2 \dots,$ *as*

$$J_d \geq \sum_{n=0}^{m} p_n J_n, \tag{7.37}$$

*such that* $\sum_{n=1}^{m} n p_n = d$ *and for all* $n$, $\max(\{p_n | n = 0, \dots, m\}) < 1$.

*Proof.* It suffices to argue that

$$J_n - J_{n-1} \geq J_{n+1} - J_n.$$

The result then follows immediately by concavity. Due to (7.36) with $k = n + 1$, we have that $2J_n \geq J_{n+1} + J_{n-1}$ can be written as

$$(2\mu_1 + (n+1)\mu_2)J_n \geq 1 + \frac{\mu_1}{\mu_2}\sum_{j=1}^{n+1}\frac{1}{j} + (\mu_1 + (n+1)\mu_2)J_{n-1}.$$

By (7.36) with $k = n$, we get that this is equivalent to

$$\frac{2\mu_1 + (n+1)\mu_2}{\mu_1 + n\mu_2}\left(1 + \frac{\mu_1}{\mu_2}\sum_{j=1}^{n}\frac{1}{j} + n\mu_2 J_{n-1}\right) \geq 1 + \frac{\mu_1}{\mu_2}\sum_{j=1}^{n+1}\frac{1}{j} + (\mu_1 + (n+1)\mu_2)J_{n-1}$$

that is,

$$\frac{\mu_1+\mu_2}{\mu_1+n\mu_2}\left(1+\frac{\mu_1}{\mu_2}\sum_{j=1}^{n}\frac{1}{j}+n\mu_2 J_{n-1}\right) \geq \frac{1}{n+1}\frac{\mu_1}{\mu_2}+(\mu_1+\mu_2)J_{n-1}.$$

By multiplying with $\frac{\mu_1+n\mu_2}{\mu_1+\mu_2}$, we get

$$1+\frac{\mu_1}{\mu_2}\sum_{j=1}^{n}\frac{1}{j}+n\mu_2 J_{n-1} \geq \frac{1}{n+1}\frac{\mu_1+n\mu_2}{\mu_1+\mu_2}\frac{\mu_1}{\mu_2}+(\mu_1+n\mu_2)J_{n-1}.$$

Which, after dividing by $\mu_1$, is equivalent to

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{1}{n+1}\right) \geq \frac{n-1}{n+1}\frac{1}{\mu_1+\mu_2}+J_{n-1}. \tag{7.38}$$

By using the definition of $J_{n-1}$ and multiplying by $\mu_1+(n-1)\mu_2$, we get

$$\begin{aligned}1+(n-1)\frac{\mu_2}{\mu_1}+(n-1)\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{1}{n+1}\right)+\frac{\mu_1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{1}{n+1}\right)\\ \geq \frac{n-1}{n+1}+\frac{n-1}{n+1}\frac{(n-2)\mu_2}{\mu_1+\mu_2}+1+\frac{\mu_1}{\mu_2}\sum_{j=1}^{n-1}\frac{1}{j}+(n-1)\mu_2 J_{n-2},\end{aligned}$$

which after simplification and division by $(n-1)\mu_2$ is equivalent to

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{2}{n+1}\right)+\frac{1}{n-1}\frac{\mu_1}{\mu_2^2}\left(\frac{1}{n}-\frac{1}{n+1}\right) \geq \frac{n-2}{n+1}\frac{1}{\mu_1+\mu_2}+J_{n-2}. \tag{7.39}$$

Which holds if

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{2}{n+1}\right) \geq \frac{n-2}{n+1}\frac{1}{\mu_1+\mu_2}+J_{n-2}. \tag{7.40}$$

Doing similar steps as between (7.38) and (7.39), we get that (7.40) is equivalent to

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{3}{n+1}\right)+\frac{1}{n-2}\frac{\mu_1}{\mu_2^2}\left(\frac{1}{n-1}+\frac{1}{n}-\frac{2}{n+1}\right) \geq \frac{n-3}{n+1}\frac{1}{\mu_1+\mu_2}+J_{n-3},$$

which is true if

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{3}{n+1}\right) \geq \frac{n-3}{n+1}\frac{1}{\mu_1+\mu_2}+J_{n-3},$$

and so on. In the end we get that $J_n - J_{n+1} \geq J_{n-1} - J_n$ holds if

$$\frac{1}{\mu_1}+\frac{1}{\mu_2}\left(\sum_{j=1}^{n}\frac{1}{j}-\frac{n}{n+1}\right) \geq J_0,$$

which is true as $J_0 = \frac{1}{\mu_1}$ and $\frac{1}{j} \geq \frac{1}{n+1}$. □

**Parent job stealing** Taking $r \to \infty$, the effect on the mean response and waiting times in the limit is shown in Figure 7.7. The expected waiting time $E[W^{(p)}]$ goes to zero and the mean response time $E[T^{(p)}]$ goes to the mean service time, $E[J^{(p)}]$, i.e.

$$\lim_{r \to \infty} E[T^{(p)}(r)] = E[J^{(p)}]. \tag{7.41}$$

## 7.8 Model validation

Mean field models are intended to capture the system behavior as the number of servers in the systems tends to infinity. In this section we use simulation experiments to indicate that the system performance for a large finite system is very close to the fixed point of the mean field models. It may be possible to formally show that the sequence of the stationary measures of the finite systems weakly converge towards to the Dirac measure of the fixed point using the methodology in [24]. In fact, if we truncate the queues to some large finite size $B$, proving the convergence of the sample paths over finite time scales towards the solution of the set of ODEs is fairly straightforward using Kurtz's theorem [47]. To show that the convergence can be extended to the stationary regime one also needs to establish global attraction of the fixed point. Global attraction is often proven using monotonicity arguments, but it seems doubtful that such an argument can be leveraged for our multithreading models.

We consider different scenarios for varying probe rates and the two stealing strategies, for $N = 500$ with $\mu_1 = 1, \mu_2 = 2$ and child job distribution $\check{p} = [5,4,3,2,1]/15 \approx [0.33, 0.27, 0.20, 0.13, 0.07]$. Figure 7.3 shows the calculated waiting and response times for the mean field models (solid lines) and the simulations (dotted lines) for two different probe rates, $r = 1, 10$. The simulation started from an empty system and the system was simulated for $T = 10^5$ with a warm-up period of 33%. The 95% confidence intervals were computed based on 5 runs.

We see that there is an excellent agreement between the ODE and simulation times for all settings, and that the ODE plots consistently lie close to the displayed confidence intervals. As expected, the response times for $r = 10$ are less than for $r = 1$, for both strategies.

Table 7.3 shows the mean field value and the relative errors obtained when comparing the mean response rate in a finite system with $N$ servers, under both stealing strategies with $r \in \{1, 10\}$, $\rho \in \{0.75, 0.85\}$ based on 20 runs, with a runtime of $T = 2 \cdot 10^5$ and warm-up period of 33%. Overall the relative error tends to increase with $\rho$ and $r$ and decreases in $N$. Similar to the results in [24], the expected response times appear to be $1/N$ accurate, which means that doubling $N$ should approximately halve the relative error. For all cases the relative error is below 1%.

## 7.9 Numerical experiments

In this section we consider the performance, i.e., the mean response time of a job for each policy, followed by a comparison of the two.

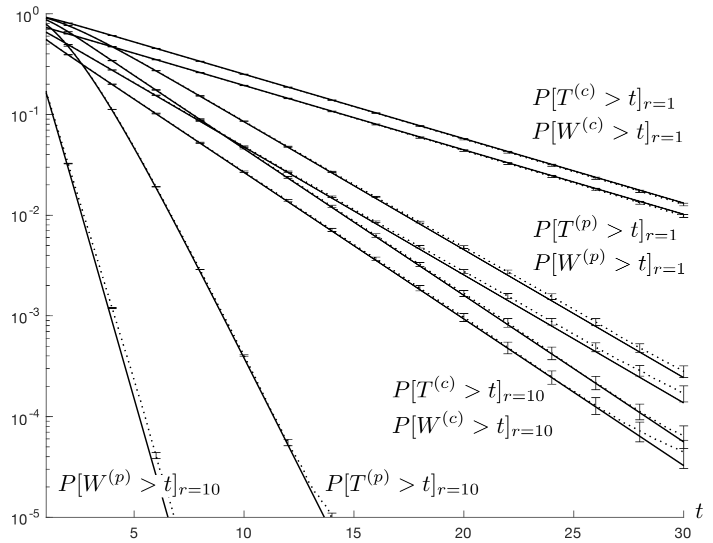


Figure 7.3: Waiting and response times from the ODE and simulations.

### 7.9.1 Performance of child job stealing

**Example 7.9.1.** For the parameter set $(\rho, \mu_1, \mu_2, \check{p}) = (0.85, 1, 2, U(2,4))$, with $r \longrightarrow \infty$ we illustrate this result in Equation (7.35) in Figure 7.4 where for increasing $r$ we see the mean waiting and response times approach $E[W^{(c)}] = 0.9015$ and $E[T^{(c)}] = 2.2706$. In this system, setting a probe rate $r \sim 10^2$ would provide performance of $r = \infty$.

**Example 7.9.2.** Consider the three child job distributions with mean 3, $\check{p}_1 \sim D(3)$ and $\check{p}_2 \sim U(0,6)$ and $\check{p}_3$ defined with $p(m = 1) = 5/7$ and $p(m = 8) = 2/7$. The variances are 0 and 4, and 16 respectively. For $\rho = 0.75, 0.85, 0.95$, we illustrate the mean response times in Figure 7.5 $(\mu_1, \mu_2) = (1, 2)$. We note that the dimension of the system grows with $m$. The three dashed lines correspond to the proportional increase in mean response time from $\check{p}_1$ to $\check{p}_3$ and the three sold lines correspond to this for $\check{p}_1$ to $\check{p}_3$. Under small probe rates, $r < 10^1$, there is a significant impact of the variability on performance under child job stealing, where the most variable distribution $\check{p}_3$ performs worse than the less variable distribution $\check{p}_2$.

Further, the performance is worsened at a higher load, in the given scenarios the largest increase of 0.279 occurs for $r = 1$ and load $\rho = 0.95$. This figure also illustrates that for increasing $r$, system performance is similar under different loads and child job size variability. We note that for $r \longrightarrow \infty$, the child job distributions with positive variance perform 2 to 3% better than the deterministic child job distribution. This is due to the result in in Equation (7.37), where for $\check{p}_1 : E[J^{(c)}] = 1.3738$, $\check{p}_2 : E[J^{(c)}] = 1.3684$, $\check{p}_3 : E[J^{(c)}] = 1.3072$.

**Example 7.9.3.** We consider the effect on performance of changing the ratio $c = \mu_1/\mu_2$, under a fixed load $\rho$ and arrival rate $\lambda$. Given $c$, we have $\mu_2 = \frac{\lambda}{\rho}(\frac{1}{c} + \sum_{n=1}^{m} np_n)$ and

Table 7.3: Relative error of simulation results for $E[T^i]$ for $i \in \{(c),(p)\}$, based on 20 runs

| | | $\rho = 0.75$ | | $\rho = 0.85$ | |
|---|---|---|---|---|---|
| | $N$ | sim. ± conf. | rel.err.% | sim. ± conf. | rel.err.% |
| $r = 1$ | | | | | |
| $(c)$ | 250 | 4.6027 ± 2.71e-03 | 0.0702 | 7.3763 ± 6.57e-03 | 0.0994 |
| | 500 | 4.6026 ± 1.58e-03 | 0.0669 | 7.3718 ± 4.99e-03 | 0.0384 |
| | 1000 | 4.5999 ± 1.18e-03 | 0.0084 | 7.3683 ± 3.81e-03 | 0.0095 |
| | 2000 | 4.6001 ± 7.46e-04 | 0.0140 | 7.3707 ± 2.84e-03 | 0.0226 |
| | 4000 | 4.6000 ± 8.81e-04 | 0.0105 | 7.3685 ± 1.85e-03 | 0.0065 |
| | $\infty$ | 4.5995 | | 7.3690 | |
| $(p)$ | 250 | 3.3085 ± 1.26e-03 | 0.2634 | 4.6953 ± 3.05e-03 | 0.3715 |
| | 500 | 3.3038 ± 1.10e-03 | 0.1217 | 4.6858 ± 1.93e-03 | 0.1682 |
| | 1000 | 3.3022 ± 5.66e-04 | 0.0726 | 4.6830 ± 1.77e-03 | 0.1081 |
| | 2000 | 3.3006 ± 4.79e-04 | 0.0249 | 4.6799 ± 1.20e-03 | 0.0422 |
| | 4000 | 3.3002 ± 3.12e-04 | 0.0129 | 4.6788 ± 8.40e-04 | 0.0203 |
| | $\infty$ | 3.2998 | | 4.6779 | |
| $r = 10$ | | | | | |
| $(c)$ | 250 | 2.7648 ± 2.45e-04 | 0.3382 | 3.7295 ± 5.75e-04 | 0.6935 |
| | 500 | 2.7601 ± 8.17e-04 | 0.1684 | 3.7160 ± 3.30e-03 | 0.3297 |
| | 1000 | 2.7577 ± 5.30e-04 | 0.0781 | 3.7098 ± 1.32e-03 | 0.1624 |
| | 2000 | 2.7564 ± 3.11e-04 | 0.0339 | 3.7074 ± 9.00e-04 | 0.0981 |
| | 4000 | 2.7561 ± 2.44e-04 | 0.0272 | 3.7048 ± 5.88e-04 | 0.0283 |
| | $\infty$ | 2.7555 | | 3.7038 | |
| $(p)$ | 250 | 1.9529 ± 3.32e-04 | 0.4146 | 2.2031 ± 6.53e-04 | 0.9520 |
| | 500 | 1.9488 ± 1.68e-04 | 0.2039 | 2.1929 ± 3.45e-04 | 0.4848 |
| | 1000 | 1.9468 ± 1.60e-04 | 0.1013 | 2.1875 ± 3.20e-04 | 0.2369 |
| | 2000 | 1.9458 ± 1.14e-04 | 0.0533 | 2.1848 ± 1.93e-04 | 0.1142 |
| | 4000 | 1.9453 ± 9.28e-05 | 0.0267 | 2.1836 ± 1.63e-04 | 0.0611 |
| | $\infty$ | 1.9448 | | 2.1823 | |

$\mu_1 = \frac{\lambda}{\rho}(1 + c\sum_{n=1}^{m} np_n)$. As $c$ is increased, the mean job size remains constant, the mean parent job size decreases and the mean child job size increases, and $\lim_{c\to\infty} 1/\mu_1 = 0$, $\lim_{c\to\infty} 1/\mu_2 = \rho(\lambda\sum_{n=1}^{m} np_n)^{-1}$. Figure 7.6 illustrates the mean waiting and response times for $(\lambda,\rho,r)$= $(0.4, 0.95, 10)$ for $\check{p}_1 \sim U(2,4)$ and $\check{p}_2 \sim U(4,5)$.

An improvement in performance, due to the reduction in waiting time, can be seen for $c$ increasing to 1. For $c > 1$, the mean child job size has approached its limit and no further performance improvements are obtained. As expected, when the mean number of child jobs is decreased, from $\check{p}_2$ to $\check{p}_1$, i.e., the mean size of a child job is decreased we see the performance improve.

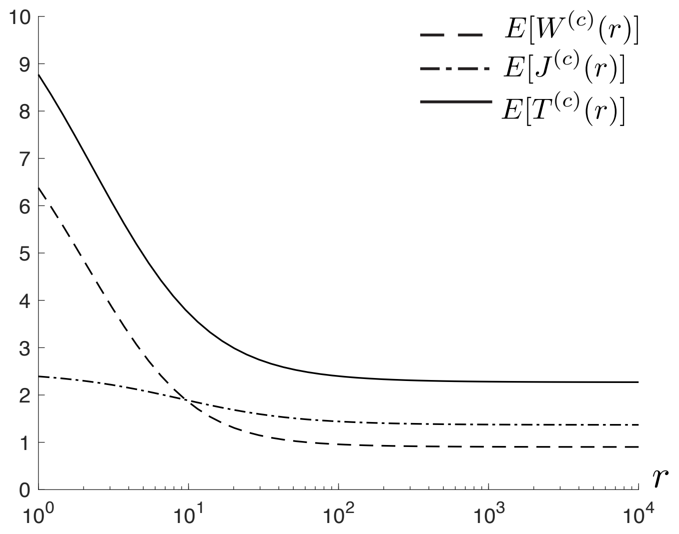


Figure 7.4: Example 7.9.1. Probe rate $r \longrightarrow \infty$

### 7.9.2 Performance of parent job stealing

**Example 7.9.4.** For the parameter set $(\rho, \mu_1, \mu_2, \check{p})$= $(0.85, 1, 2, U(2, 4))$, we illustrate the result in Equation (7.41) in Figure 7.7. For $r \sim 10^2$ the waiting time is close to zero and the mean response time approaches $E[T^{(p)}] = 2.5$.

**Example 7.9.5.** Using the distributions and parameters as Example 7.9.2, we display the mean response times for parent stealing in Figure 7.8. For the scenarios considered, the most significant impact on the mean response time is seen from $\check{p}_3$ with a proportional increase of 0.208 of $\check{p}_1$ under load $\rho = 0.95$ and $r = 1$. This impact on the mean response

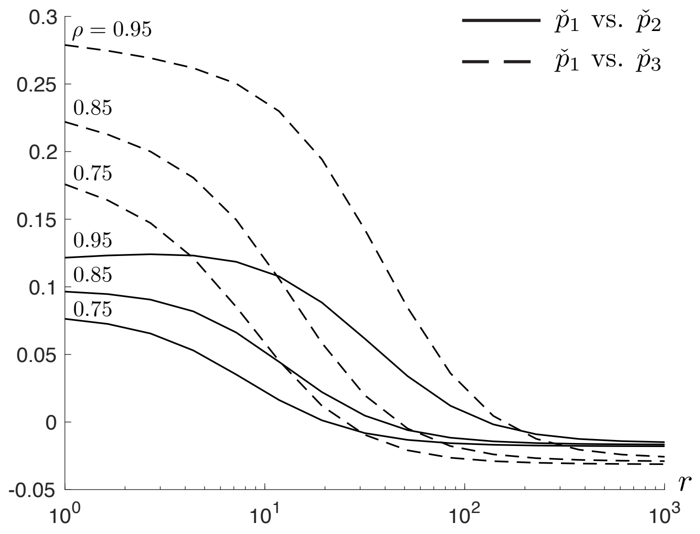


Figure 7.5: Example 7.9.2. Proportional increase in $E[T^{(c)}(r)]$

time reduces with an increased probe rate, a reduced load and a smaller variance in child job size distribution.

**Example 7.9.6.** Under parent job stealing, changing the proportion of the workload between parent and child jobs $c = \mu_1/\mu_2$ does not impact the performance of the system, as the job, the parent and any child jobs, is stolen together following a successful probe.

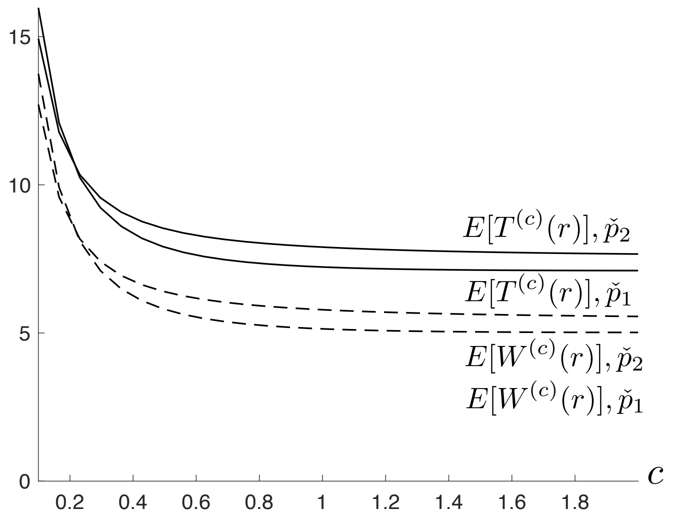


Figure 7.6: Example 7.9.3. Parent to child job service ratio $c = \mu_1/\mu_2$

### 7.9.3 Child versus parent job stealing

**Example 7.9.7.** For varying probe rates $r$ and loads $\rho$ we illustrate the proportional difference in the mean response times for child job stealing compared to parent job stealing in Figure 7.9 for child job size distributions $\check{p} = D(2), D(4), D(6), D(8)$, with $(\mu_1, \mu_2) = (1, 2)$. Under high loads and with small probe rates, parent job stealing shows the most significant benefit in performance. Increasing the number of child jobs in the system increases the orange region in which child stealing performs best. The white region illustrates the parameter set in which the two strategies perform within ±6.7% of

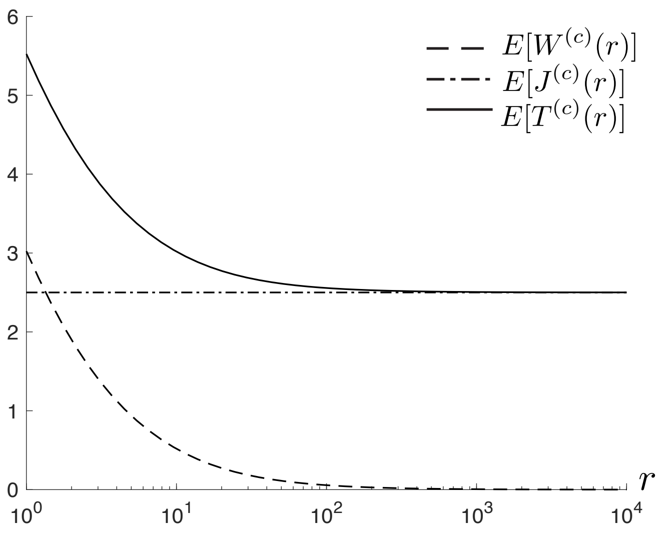


Figure 7.7: Example 7.9.4. Probe rate $r \longrightarrow \infty$

each other. In Figure 7.9(d), for $\rho = 0.5$ and $r = 20$, child stealing performs $\sim$ 50% better than parent job stealing. Whereas at $\rho = 0.95$ and $r = 1$ parent stealing performs $\sim$ 100% better than child job stealing. We note that we obtain similar insights for other child job size distributions.

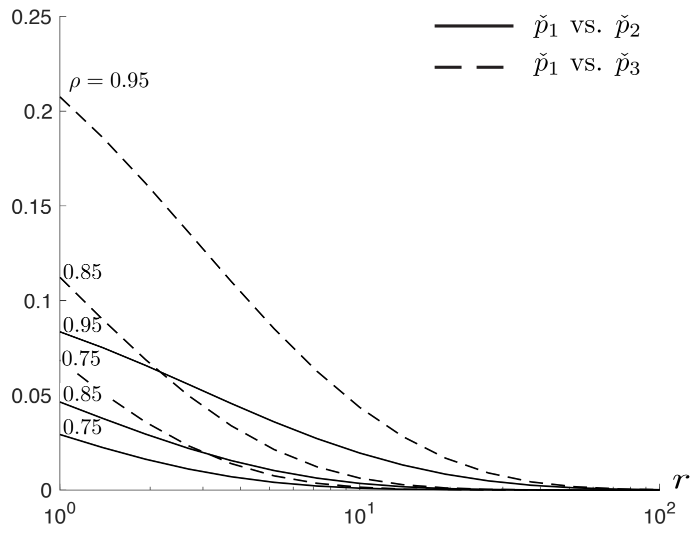


Figure 7.8: Example 7.9.5. Proportional increase in $E[T^{(p)}(r)]$

## 7.10 Conclusions and further work

We introduced two mean field models for randomized work stealing in multithreaded computations in large systems, where parent jobs spawn child jobs. We proved the existence of a unique fixed point and showed that this fixed point can be computed easily using matrix analytic methods (by solving a single quasi-birth-death Markov chain). The accuracy of these models was illustrated using simulation experiments.

The two models correspond to two stealing strategies: one that involves the transfer of

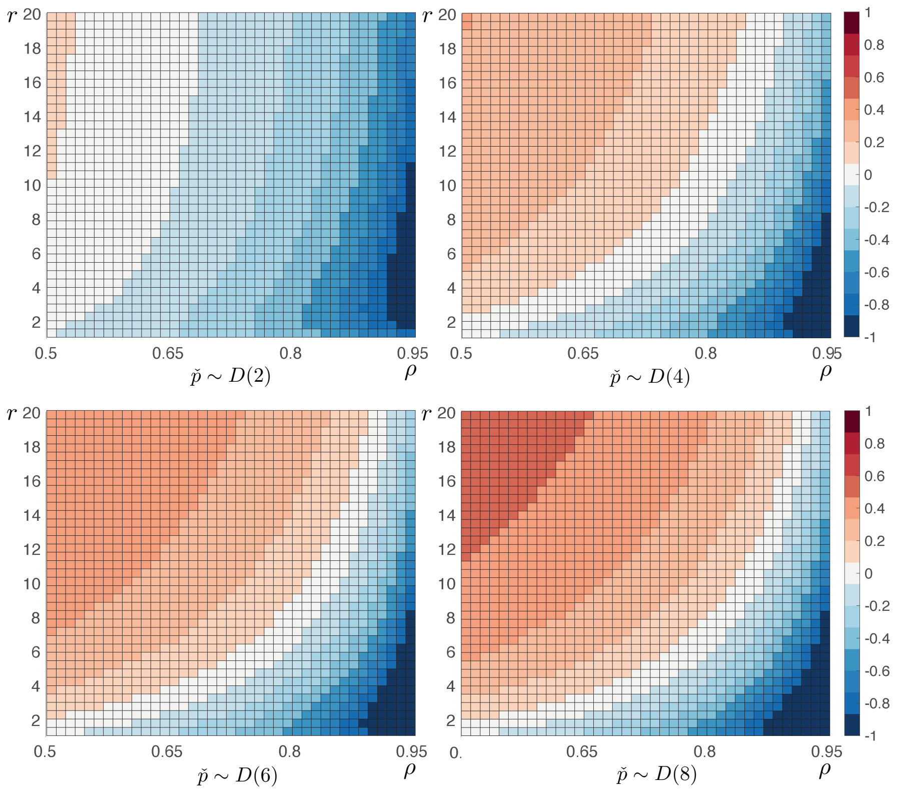


Figure 7.9: Example 7.9.7. Proportional difference in mean response time of $(c)$ compared to $(p)$. $(E[T^{(p)}] - E[T^{(c)}])/E[T^{(p)}]$

child jobs across servers; the other where parent jobs are transferred (together with the child jobs that they spawn). Having derived expressions for the response time distributions for each strategy, we investigated the impact of the probe rates, load, and child job size variability on performance with respect to the individual stealing strategies. We also studied the effect of changing the ratio of parent to child service rates and identified scenarios (low probe rate, high load) where parent stealing significantly outperformed child stealing, and scenarios (high probe rate, low load) where child stealing achieved a lower mean response time.

Future work exists in considering extensions of the model in which multiple child jobs can be transferred after a successful probe, or multigenerational multithreading, that is where child jobs can generate their own offspring jobs.

Chapter 8

# Analysis of work stealing strategies in large scale multi-threaded computing

*This Chapter contains the paper* [44] *of the same title, together with results that were originally omitted due to the constraint on the number of pages. This paper was the second paper I presented at a conference (namely at QEST 2021) and it was subsequently published in the conference journal. The conference was supposed to take place in Paris, France. However, due to the COVID-19 pandemic it was held online instead.*
*My presentation can be freely viewed at* `https://www.youtube.com/watch?v=468aPFBOysc?t=2653`*. At the conference, the paper won the best paper award. As a consequence, I was asked to write an extended version of the paper, which is contained in the next chapter.*

## 8.1 Introduction

In this chapter, we consider a system of homogeneous processors that uses a randomized work stealing policy. We consider a set of policies where if a server with pending child jobs is probed by an idle server, some of its child jobs are transferred. When a server is probed that does not have any pending child jobs, a pending parent job is transferred instead (if available). The work presented in this chapter is closely related to Chapter 7, where two systems are considered: one system where parent jobs can be stolen and the other system where child jobs can be stolen one at a time. In the current chapter we allow that several child jobs can be stolen at once and the main objective is to provide insights on how to determine the number of child jobs that should be transferred in such an event. When several child jobs can be stolen at once, child jobs may be transferred several times before being executed and this considerably complicates the analysis compared to the one in Chapter 7.

The main contributions of the chapter are the following:

1. We introduce a Quasi-Birth-Death (QBD) Markov chain describing a single server queueing system with negative arrivals that is used to approximate the performance of the work stealing system. We present simulation results that suggest that as the number of servers becomes large, the approximation error tends to zero.

2. We prove that this QBD has a unique stationary distribution for which we provide formulas for the waiting, service, mean waiting and mean service time. These are the main technical results of the chapter.

3. We write down a set of ODEs which captures the transient evolution of the system when the number of servers tends to infinity. We show that this set of ODEs has a unique fixed point which coincides with the stationary distribution. This contribution was originally omitted from [44] due to the constraint on the number of pages.

4. We compare the performance of several stealing strategies. Our main insight is that the strategy of stealing half of the child jobs performs well for low loads and/or high probe rates and stealing all child jobs is a good heuristic when the load is high and/or the probe rate is low.

The rest of this chapter is organized as follows. In Section 8.2 we describe the system while the Quasi-Birth-Death (QBD) Markov chain is introduced in Section 8.3 and the response time distribution is analyzed in Section 8.4. In Section 8.5 we present explicit results when the probe rate tends to infinity. In Section 8.6 we describe the work stealing strategies considered and present the performance of these strategies using numerical examples. The QBD approximation is validated using simulation in Section 8.7. In Section 8.8 we define a set of ODEs that captures the behaviour of the system when the number of servers tends to infinity and we show that this set of ODEs has a unique fixed point coinciding with the stationary distribution of the QBD. Section 8.9 contains some concluding remarks and possible future work.

## 8.2 System description and strategies

We consider a system with $N$ homogeneous servers each with an infinite buffer to store jobs. Parent jobs arrive in each server according to a local Poisson arrival process with rate $\lambda$. Upon entering service a parent job spawns $i \in \{0, 1, \dots, m\}, m \geq 1$, child jobs, the number of which follows a general distribution with finite support $p_i$ (i.e., $p_i \geq 0$ for every $i$ and $\sum_{i=0}^{m} p_i = 1$). These child jobs are stored locally and have priority over any parent jobs (either already present or yet to arrive), while the spawning parent job continues service. Thus, when a (parent or child) job completes service the server first checks to see whether it has any waiting child jobs, if so it starts service on a child job. If there are no child jobs present, service on a waiting parent job starts (if any are present). We assume that parent and child jobs have exponentially distributed service requirements with rates $\mu_1$ and $\mu_2$ respectively.

When a server is idle, it probes other servers at random at rate $r > 0$, where $r$ is a system parameter. Note that $r$ determines the amount of communication between the servers and increasing $r$ should improve performance at the expense of a higher communication overhead. When a server is probed (by an idle server) and it has waiting (parent or child) jobs, we state that the probe is successful. When a successful probe reaches a server without waiting child jobs, a parent job is transferred to the idle server. Note that such a transferred parent job starts service and spawns its child jobs at the new server.

When a successful probe reaches a server with pending/waiting child jobs, several child jobs can be transferred at once. If the probed server is serving a parent job and there are $i$ child jobs in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\phi_{i,j}$ (i.e., for every $i$ we have $\sum_{j=1}^{i} \phi_{i,j} = 1$). On the other hand if a child job is being processed by the probed server and there are $i$ child jobs waiting in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\psi_{i,j}$ (i.e., for every $i$ we have $\sum_{j=1}^{i} \psi_{i,j} = 1$). For ease of notation we set $\phi_{i,j} = \psi_{i,j} = 0$ if $j > i$ and further $\phi_{i,j} = \psi_{i,j} = 0$ if $i$ or $j$ is 0. Probes and job transfers are assumed to be instantaneous. In Figure 8.1, we visualize the working of a system with $N = 4$.

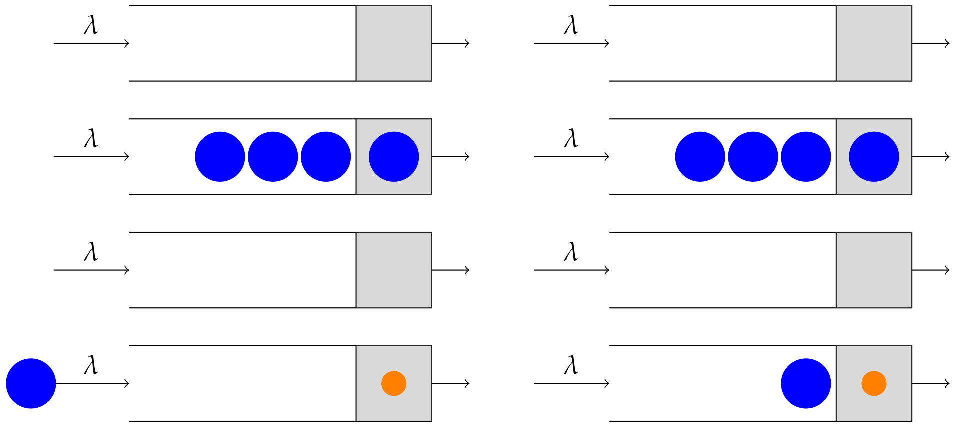
Note, that we make the distinction between the probabilities $\phi_{i,j}$ and $\psi_{i,j}$ because it is possible that good a stealing strategy has different stealing probabilities depending on whether or not a parent is in service. For example: suppose we have a system where parent jobs take a long time to process, while children get processed quickly. Suppose further that the arrival rate $\lambda$ is small such that the chance that there exist pending parent jobs in the system is small. Then one can expect that the following is a good stealing strategy: if a parent is in service in the probed queue steal most or all of the child jobs (as it will probably take a long time before the parent exits service); if there is a child in service steal half of the pending children.

The main objective of this chapter is to study how the probabilities $\phi_{i,j}$ and $\psi_{i,j}$ influence the response time of a job, where the response time is defined as the time between the arrival of a parent job and the completion of the parent and all its spawned child jobs. Given the above description, it is clear that we get a Markov process if we keep track of the number of parent and child job in each of the $N$ servers. This Markov process however does not appear to have a product form, making its analysis prohibitive.

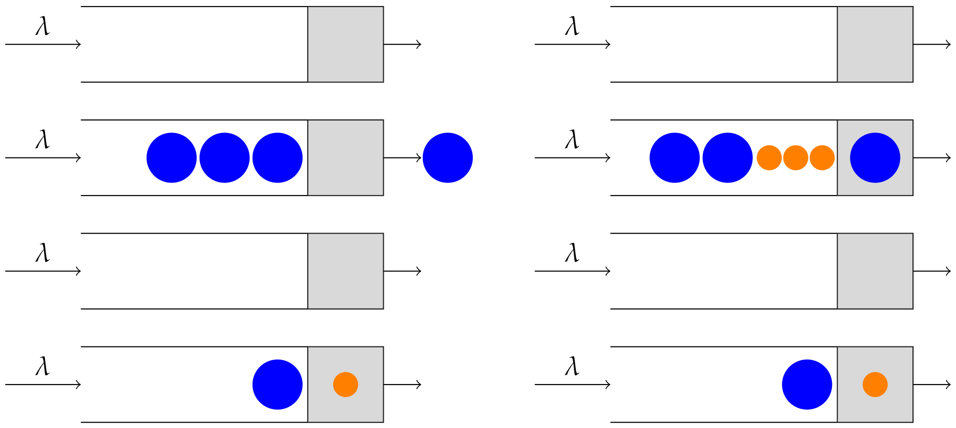
Instead we use an approximation method, the accuracy of which is investigated in Section 8.7. The idea of the approximation exists in focusing on a single server and assuming that the queue lengths at any other server are independent and identically distributed as in this particular server. Within the context of load balancing, this approach is known as the cavity method (see [9] or Section 4.3). In fact all the analytical models used in [16, 25, 55–57, 68, 69, 77, 79] can be regarded as cavity method approximations. A common feature of such an approximation is that it tends to become more accurate as the number of servers tends to infinity, as we demonstrate in Section 8.7 for our model. The cavity method typically involves iterating the so-called cavity map (see [9] or Section 4.3). However, in our case the need for such an iteration is avoided by deriving expressions for the rates at which child and parent jobs are stolen.

## 8.3 Quasi-Birth-Death Markov chain

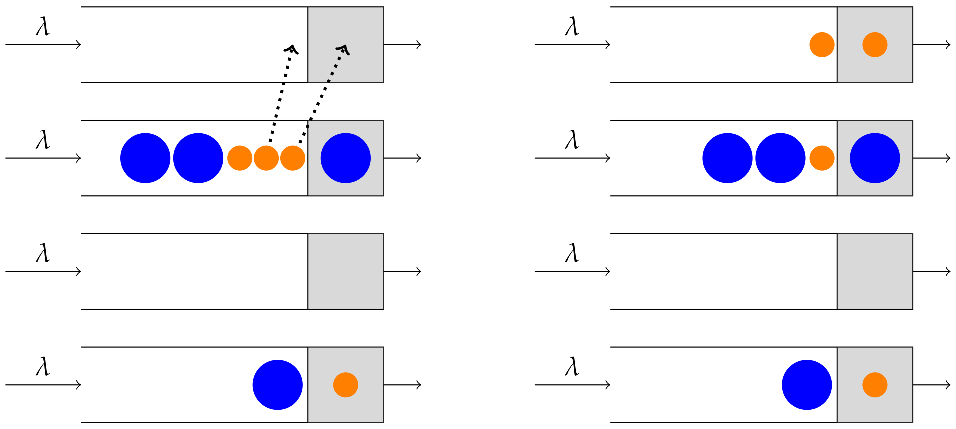
In this section we introduce a Quasi-Birth-Death (QBD) Markov chain to approximate the system from the viewpoint of a single server. Let $\lambda_p(r)$ denote the rate at which parent jobs are stolen when the server is idle. Let $\lambda_{c,1}(r), \ldots, \lambda_{c,m}(r)$ denote respectively the rates at which $1, \ldots, m$ child jobs are stolen. We provide formulas for these rates further on. The evolution of a single server has the following characteristics, where the negative arrivals correspond to steal events:



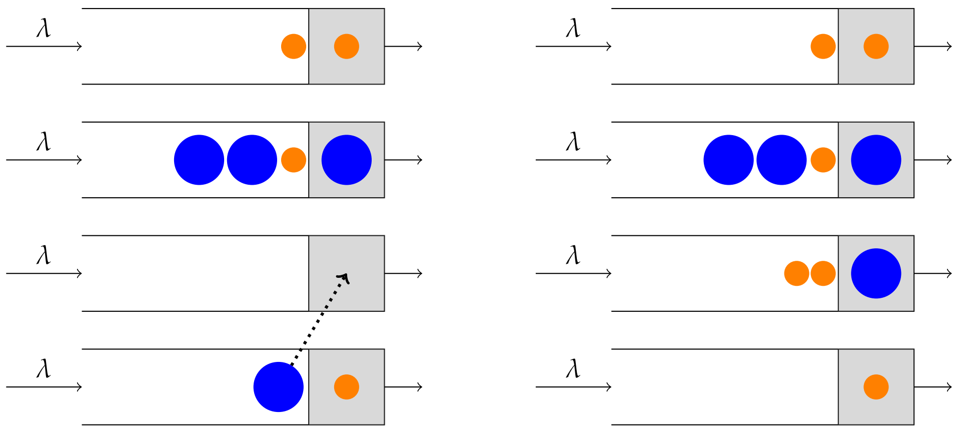
(a) A parent job enters fourth queue and starts waiting.



(b) A parent completes service in the second queue and the next parent enters service, spawning three children.



(c) The first two waiting children in the second queue get transferred to the first queue. The first of these child jobs starts service.



(d) Waiting parent job gets transferred from the fourth to first queue. The parent enters service and spawns two jobs.

Figure 8.1: Example of a system with $N = 4$. Blue and orange dots depict parent and child jobs respectively.

1. When the server is busy, arrivals of parent jobs occur according to a Poisson process with rate $\lambda$. When the server is idle, parent jobs arrive at the rate $\lambda + \lambda_p(r)$, while a batch of $i$ child jobs arrives at rate $\lambda_{c,i}(r)$ for $i = 1, \dots, m$.
2. Upon entering service, a parent job spawns $i \in \{0, 1, \dots, m\}, m \geq 1$, child jobs with probability $p_i$. Child jobs are stored locally.
3. Child jobs have priority over any parent jobs *waiting* in the queue and are thus executed immediately after their parent job when executed on the same server.
4. Parent and child jobs have exponentially distributed service requirements with rates $\mu_1$ and $\mu_2$, respectively.
5. If there are parent jobs and no child jobs waiting in the buffer of the server then a negative parent arrival occurs at the rate $rq$, where $q = 1 - \rho$ is the probability that a queue is idle (where $\rho$ is defined in (8.1)).
6. If a parent job is in service and there are $i \in \{1, \dots, m\}$ child jobs in the buffer of the server, a batch of $j$ negative child job arrivals occurs at the rate $rq\phi_{i,j}$, for all $j \in \{1, \dots, i\}$.
7. If a child job is in service and there are $i \in \{1, \dots, m-1\}$ child jobs pending in the buffer of the server, a batch of $j$ negative child job arrivals occurs at the rate $rq\psi_{i,j}$, for all $j \in \{1, \dots, i\}$.

Note that the load of the system can be expressed as

$$\rho = \lambda \left( \frac{1}{\mu_1} + \frac{\sum_{n=1}^{m} n p_n}{\mu_2} \right). \tag{8.1}$$

Denote by $X \geq 0$ the number of parent jobs waiting, by $Y \in \{0, 1, \dots, m\}$ the number of child jobs in the server (either in service or waiting), and by $Z \in \{0, 1\}$ whether a parent

Table 8.1: Transitions for the QBD in Section 8.3

| | From → To | Rate | For |
|---|---|---|---|
| 1. | $(0,0,0) \to (0,j,0)$ | $\lambda_{c,j}(r)$ | $j = 1,\dots,m,$ |
| 2. | $(0,0,0) \to (0,j,1)$ | $(\lambda + \lambda_p(r))p_j$ | $j = 0,1,\dots,m,$ |
| 3. | $(X,Y,Z) \to (X+1,Y,Z)$ | $\lambda$ | $X+Y+Z \geq 1,$ |
| 4. | $(X,Y,1) \to (X,Y,0)$ | $\mu_1$ | $X \geq 0, Y \geq 1$ or $X = 0, Y = 0,$ |
| 5. | $(X,Y,0) \to (X,Y-1,0)$ | $\mu_2$ | $X \geq 0, Y \geq 2$ or $X = 0, Y = 1,$ |
| 6. | $(X,1,0) \to (X-1,j,1)$ | $\mu_2 p_j$ | $X \geq 1, j = 0,1,\dots,m,$ |
| 7. | $(X,0,1) \to (X-1,j,1)$ | $\mu_1 p_j$ | $X \geq 1, j = 0,1,\dots,m,$ |
| 8. | $(X,Y,Z) \to (X-1,Y,Z)$ | $rq$ | $X \geq 1, Y+Z = 1,$ |
| 9. | $(X,Y,1) \to (X,Y-j,1)$ | $rq\phi_{Y,j}$ | $X \geq 0, Y \geq j, j = 1,\dots,m,$ |
| 10. | $(X,Y,0) \to (X,Y-j,0)$ | $rq\psi_{Y-1,j}$ | $X \geq 0, Y \geq j+1, j = 1,\dots,m-1.$ |

job is currently in service ($Z = 1$) or not ($Z = 0$). The possible transitions of the QBD Markov chain are listed in Table 8.1, corresponding to: 1. a batch of $j$ child jobs arriving at an idle queue and the first child job proceeding directly into service, 2. a parent job arriving at an idle queue and proceeding directly into service, spawning $j$ child jobs, 3. a parent arriving to a non-idle queue, 4. completion of a parent in service, not succeeded by another parent job, 5. child service completion, succeeded by either another child job or no job, 6. child service completion, succeeded by a parent job that enters service and spawns $j$ child jobs, 7. parent service completion, succeeded by a parent job that enters service and spawns $j$ child jobs, 8. negative parent job arrival, 9. a parent is in service and a batch of negative child job arrivals occurs, 10. a child job is in service and a batch of negative child job arrivals occurs.

The three dimensional process $\{X_t(r), Y_t(r), Z_t(r) : t \geq 0\}$ is an irreducible, aperiodic Quasi-Birth-Death process, where the *level* $\ell = *$ when the chain is in state $(0,0,0)$ and equals $\ell \geq 0$ when the chain is in a state with $X = \ell$ (different from $(0,0,0)$). When the level $\ell \geq 0$, the *phase* of the QBD is two dimensional and given by $(Y,Z)$. The $2m+1$ phases of level $\ell \geq 0$ are ordered such that the $j$-th phase corresponds to $(Y,Z) = (j,0)$, for $j = 1,\dots,m$ and phase $m+1+j$ to $(Y,Z) = (j,1)$ for $j = 0,\dots,m$.

As explained below, the generator of the process is

$$Q(r) = \begin{bmatrix} -\lambda_0(r) & \sum_{j=1}^m \lambda_{c,j}(r)e_j + (\lambda + \lambda_p(r))\alpha & & & \\ \mu & B_0(r) & A_1 & & \\ & A_{-1}(r) & A_0(r) & A_1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

with $\lambda_0(r) = \sum_{j=1}^m \lambda_{c,j}(r) + \lambda + \lambda_p(r)$, with $e_j$ a row vector with 1 in its $j$-th entry and zeros elsewhere. The initial probability vector $\alpha$ records the distribution of child jobs upon a parent job entering service and is given by $\alpha = \begin{bmatrix} 0'_m & p_0 & p_1 & \dots & p_m \end{bmatrix}$, where $0_i$ is a column vector of zeros of length $i$. Indeed, at rate $\lambda_{c,j}(r)$ a batch of $j$ child jobs arrives in an idle server, causing a jump to level 1 and phase $j$, while at rate $\lambda + \lambda_p(r)$ a parent job arrives that spawns $j$ child jobs with probability $p_j$ causing a jump to phase $m+1+j$ of level 1.

For further use, define

$$S(r) = \begin{bmatrix} S_{00}(r) & 0 \\ S_{10} & S_{11}(r) \end{bmatrix},$$

where $S_{00}(r)$ is an $m \times m$ matrix and $S_{11}(r)$ is an $(m+1) \times (m+1)$ matrix,

$$S_{00}(r) = rq \begin{bmatrix} & & & \\ \psi_{1,1} & & & \\ \vdots & \ddots & & \\ \psi_{m-1,m-1} & \cdots & \psi_{m-1,1} & \end{bmatrix} + \begin{bmatrix} -\mu_2 & & & \\ \mu_2 & \ddots & & \\ & \ddots & \ddots & \\ & & \mu_2 & -\mu_2 \end{bmatrix},$$

$$S_{10} = \begin{bmatrix} 0 & \cdots & \\ \mu_1 & & \\ & \mu_1 & \\ & & \ddots \end{bmatrix}, \quad S_{11}(r) = rq \begin{bmatrix} & & & \\ \phi_{1,1} & & & \\ \vdots & \ddots & & \\ \phi_{m,m} & \cdots & \phi_{m,1} & \end{bmatrix} + \begin{bmatrix} -\mu_1 & & \\ & \ddots & \\ & & \ddots & \\ & & & -\mu_1 \end{bmatrix}.$$

The matrix $A_0(r)$ contains the possible transitions for which the level $\ell > 0$ remains unchanged, this is when child jobs are stolen, or when a waiting child moves into service. Hence

$$A_0(r) = S(r) - \lambda I - rqI.$$

Note that even when there are no child jobs waiting, the rate $rq$ appears on the main diagonal due to the negative parent arrivals. When $\ell = 0$ there are no parent jobs waiting and therefore the negative parent arrivals that occur in phase 1 and $m+1$ have no impact. This implies that

$$\begin{aligned} B_0(r) &= A_0(r) + rqV_0 \\ &= S(r) - \lambda I - rq(I - V_0), \end{aligned}$$

where $V_0 = \mathrm{diag}(\begin{bmatrix} 1 & 0'_{m-1} & 1 & 0'_m \end{bmatrix})$. The level $\ell$ can only decrease by one due to a service completion from a phase with no pending child jobs, that is, from phase 1 and $m+1$. To capture these events define $\mu = \begin{bmatrix} \mu_2 & 0'_{m-1} & \mu_1 & 0'_m \end{bmatrix}'$. The level can also decrease due to a negative parent arrival when $\ell > 0$. The matrix $A_{-1}(r)$ records the transitions for which the level decreases and therefore equals

$$A_{-1}(r) = \mu\alpha + rqV_0.$$

Finally, parent job arrivals always increase the level by one:

$$A_1 = \lambda I.$$

Denote by $A(r) = A_{-1}(r) + A_0(r) + A_1$, the generator of the phase process, then

$$A(r) = S(r) + \mu\alpha - rq(I - V_0).$$

Define

$$\pi_*(r) = \lim_{t \to \infty} P[X_t(r) = 0, Y_t(r) = 0, Z_t(r) = 0],$$

and for $\ell \geq 0$,

$$\pi_\ell(r) = (\pi_{\ell,1,0}(r), \dots \pi_{\ell,m,0}(r), \pi_{\ell,0,1}(r), \dots, \pi_{\ell,m,1}(r))$$

where

$$\pi_{\ell,j,k}(r) = \lim_{t\to\infty} P[X_t(r) = \ell, Y_t(r) = j, Z_t(r) = k].$$

Due to the QBD structure [60], we have

$$\pi_0(r) = \pi_*(r)R_0(r), \tag{8.2}$$

where $R_0(r)$ is a row vector of size $2m + 1$ and for $\ell \geq 1$,

$$\pi_\ell(r) = \pi_0(r)R(r)^\ell, \tag{8.3}$$

where $R(r)$ is a $(2m + 1) \times (2m + 1)$ matrix and by [48] the smallest nonnegative solution to

$$A_1 + R(r)A_0(r) + R(r)^2A_{-1}(r) = 0.$$

Also, due to the balance equations with $\ell = 0$, we have

$$\sum_{j=1}^{m} \lambda_{c,j}(r)e_j + (\lambda + \lambda_p(r))\alpha + R_0(r)B_0(r) + R_0(r)R(r)A_{-1}(r) = 0$$

and due to [48, Chapter 6]

$$A_1G(r) = R(r)A_{-1}(r),$$

where $G(r)$ is the smallest nonnegative solution to

$$A_{-1}(r) + A_0(r)G(r) + A_1G(r)^2 = 0.$$

Combining the above yields the following expression:

$$R_0(r) = -\left(\sum_{j=1}^{m} \lambda_{c,j}(r)e_j + (\lambda + \lambda_p(r))\alpha\right)(B_0(r) + \lambda IG(r))^{-1}, \tag{8.4}$$

where $B_0(r) + \lambda IG(r)$ is a subgenerator matrix and is therefore invertible. We note that $R(r)$ and $G(r)$ are independent of $\lambda_{c,1}(r), \dots, \lambda_{c,m}(r)$ and $\lambda_p(r)$ and can be computed easily using the toolbox presented in [5]. To fully characterize the QBD in terms of $\lambda, \mu_1, \mu_2$ and the probabilities $p_i$, $\phi_{i,j}$ and $\psi_{i,j}$, we need to specify $\lambda_{c,1}(r), \dots, \lambda_{c,m}(r)$ and $\lambda_p(r)$.

To determine these rates we use the following observation: $q = 1 - \rho$ should be the probability that the QBD is in state $(0, 0, 0)$ and in this state batches of $j$ child jobs arrive at rate $\lambda_{c,j}(r)$. Therefore $q\lambda_{c,j}(r)$ should equal the parent arrival rate $\lambda$ times the expected number of times that a batch of $j$ child jobs is stolen per parent job. The main difficulty in using this equality lies in the fact that we must also take into account that a child job can be stolen several times before it is executed.

To this end and as a preparation for Proposition 8.3.1, we define recursively $p_{0,i}(r)$, $i = 1, \dots, m$, as the probability that the QBD visits phase $(i, 0)$ during the service of a job and similarly $p_{1,i'}(r)$, $i' = 0, \dots, m$ that phase $(i', 1)$ is visited. By conditioning on whether we first have a service completion or steal event, we have

$$p_{1,m}(r) = p_m,$$

$$p_{1,i}(r) = p_i + \frac{rq}{rq+\mu_1} \sum_{j>i} p_{1,j}(r)\phi_{j,j-i},$$

for $i \in \{0, \dots, m-1\}$, and

$$p_{0,i}(r) = \frac{\mu_1}{rq+\mu_1} p_{1,i}(r) + \frac{\mu_2}{rq+\mu_2} p_{0,i+1}(r) + \frac{rq}{rq+\mu_2} \sum_{j>i} p_{0,j}(r)\psi_{j-1,j-i},$$

for $i \in \{1, \dots, m\}$, with $p_{0,m+1} = 0$. Note that

$$p_{1,0}(r) + p_{0,1}(r) = 1, \tag{8.5}$$

as phase $(1,0)$ or $(0,1)$ is visited before any job completes service.
We also define $p_i^j(r)$, for $1 \le i \le j \le m$, as the probability that the QBD visits phase $(0,i)$ given that it is in the phase $(0,j)$ before a job completes service. We have

$$\begin{aligned} p_j^j(r) &= 1, \\ p_i^j(r) &= \frac{\mu_2}{rq+\mu_2} p_{i+1}^j(r) + \frac{rq}{rq+\mu_2} \sum_{k=i+1}^{j} \psi_{k-1,k-i} p_k^j(r), \end{aligned}$$

for $i \in \{1, \dots, j-1\}$. Note that we have $p_1^j(r) = 1$, for $1 \le j \le m$, as the QBD visits phase $(0,1)$ before completing service if it is in phase $(0,j)$. We are now in a position to define $\lambda_{c,i}(r)$ recursively as:

$$\begin{aligned} \lambda_{c,m}(r) &= \frac{\lambda}{q} p_{1,m}(r) \frac{rq}{rq+\mu_1} \phi_{m,m} \\ \lambda_{c,i}(r) &= \frac{\lambda}{q} \frac{rq}{rq+\mu_1} \sum_{j\ge i} p_{1,j}(r)\phi_{j,i} + \frac{\lambda}{q} \frac{rq}{rq+\mu_2} \sum_{j>i} p_{0,j}(r)\psi_{j-1,i} \\ &\quad + \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} p_k^j(r)\psi_{k-1,i} \frac{rq}{rq+\mu_2} \end{aligned} \tag{8.6}$$

for $i \in \{1, \dots, m-1\}$. Note that $p_{1,m}(r) rq \phi_{m,m}/(rq+\mu_1)$ indeed equals the expected number of batches of size $m$ that are stolen per parent job (as the job must spawn $m$ child jobs and these must be stolen as a batch before the parent completes service). For $i < m$, the first two sums represent the expected number of size $i$ batches that are stolen from the original server, while the double sum counts the expected number of such steals that occur on a server different from the original server.

It remains to define $\lambda_p(r)$, for this we demand that $\pi_*(r) = q$ and that

$$\pi_*(r) + \sum_{\ell \ge 0} \pi_\ell(r) \mathbf{1} = 1,$$

where $\mathbf{1}$ is a column vector of ones. Then from equations (8.2) and (8.3),

$$q\left(1 + R_0(r)(I - R(r))^{-1}\mathbf{1}\right) = 1, \tag{8.7}$$

where the inverse of $I - R(r)$ exists due to Proposition 8.3.1. Using (8.4) and (8.7) we get:

$$\lambda_p(r) = \frac{(1-q) - q(\sum_{j=1}^{m} \lambda_{c,j}(r)e_j + \lambda\alpha)w}{q\alpha w}, \tag{8.8}$$

with $w = -(B_0(r) + \lambda I G(r))^{-1}(I - R(r))^{-1}\mathbf{1}$. Note that $\lambda_p(r)$ is well-defined for $q > 0$, i.e. $\rho < 1$. This completes the description of the QBD Markov chain.

**Proposition 8.3.1.** *The QBD process $\{X_t(r), Y_t(r), Z_t(r) : t \geq 0\}$ has a unique stationary distribution for any $r \geq 0$ if $\rho < 1$.*

*Proof.* The positive recurrence of the QBD process only depends on the matrices $A_{-1}(r)$, $A_0(r)$ and $A_1$ [60]. These three matrices are the same three matrices as those of the QBD characterizing the $M/MAP/1$ queue where the MAP service process is characterized by $(S_0(r), S_1(r))$ with $S_0(r) = S(r) - rqI$ and $S_1(r) = \mu\alpha + rqV_0$. As such the QBD process is positive recurrent if and only if the arrival rate $\lambda$ is less than the service completion intensity of the $MAP$ $(S_0(r), S_1(r))$. This intensity equals $\theta^{(r)}S_1(r)\mathbf{1}/\theta^{(r)}\mathbf{1}$, where the vector $\theta^{(r)}$ is such that $\theta^{(r)}(S_0(r) + S_1(r)) = 0$.

We note that $S_0(r) + S_1(r) = A_{-1}(r) + A_0(r) + A_1 = A(r)$ and define

$$\begin{aligned}
\theta^{(r)}_{(0,1)} &= \frac{1}{\mu_2} p_{0,1}(r), \\
\theta^{(r)}_{(0,i')} &= \frac{1}{rq + \mu_2} p_{0,i'}(r), \\
\theta^{(r)}_{(1,0)} &= \frac{1}{\mu_1} p_{1,0}(r), \\
\theta^{(r)}_{(1,i)} &= \frac{1}{rq + \mu_1} p_{1,i}(r),
\end{aligned}$$

for $i' = 2, \ldots, m$ and for $i = 1, \ldots, m$. Define $v^{(r)} = \theta^{(r)}A(r)$. Then, using (8.5),

$$v^{(r)}_i = p_{i-m-1} - p_{1,i-m-1}(r) + \frac{rq}{rq + \mu_1} \sum_{j > i-m-1} p_{1,j}(r)\phi_{j,j-i-m-1} = 0,$$

for $i = m + 1, \ldots, 2m + 1$, and

$$\begin{aligned}
v^{(r)}_{i'} &= -p_{0,i'}(r) + 1[i < m]\frac{\mu_2}{rq + \mu_2} p_{0,i'+1}(r) \\
&+ \frac{rq}{rq + \mu_2} \sum_{j > i} p_{0,j}(r)\psi_{j-1,j-i'} + \frac{\mu_1}{rq + \mu_1} p_{1,i'}(r) = 0,
\end{aligned}$$

for $i' = 1, \ldots, m$. Hence $\theta^{(r)}A(r) = \theta^{(r)}(S_0(r) + S_1(r)) = 0$. As

$$\begin{aligned}
\frac{\theta^{(r)}S_1(r)\mathbf{1}}{\theta^{(r)}\mathbf{1}} &= \frac{1}{\theta^{(r)}\mathbf{1}} \left( \frac{\mu_2 + rq}{\mu_2} p_{0,1}(r) + \frac{\mu_1 + rq}{\mu_1} p_{1,0}(r) \right) \\
&\geq \frac{1}{\theta^{(r)}\mathbf{1}} (p_{0,1}(r) + p_{1,0}(r)) = \frac{1}{\theta^{(r)}\mathbf{1}},
\end{aligned}$$

it suffices that $\lambda < 1/\theta^{(r)}\mathbf{1}$ for the chain to be positive recurrent. For $r = 0$ we have $p_{1,i}(r) = p_i$ and $p_{0,i'} = \sum_{j \geq i'} p_j$, which implies that $\theta^{(0)}\mathbf{1} = \rho/\lambda$. Therefore $\lambda < 1/\theta^{(0)}\mathbf{1}$

is equivalent to demanding that $\rho < 1$. As $\theta^{(r)}\mathbf{1}$ is the mean time between two service completions of the MAP process where the state is reset according to the vector $\alpha$, we have that $\theta^{(r)}\mathbf{1}$ decreases in $r$. This completes the proof as $\rho < 1$ implies that $\lambda < 1/\theta^{(0)}\mathbf{1} \leq 1/\theta^{(r)}\mathbf{1}$. □

## 8.4 Response time distribution

We define $T(r)$ as the response time of a job in a system with probe rate $r$. The response time is defined as the length of the time interval between the arrival of a parent job and the completion of this parent job and all of its spawned child jobs. $T(r)$ can be expressed as the sum of the waiting time $W(r)$ and the service time $J(r)$. The waiting time is defined as the amount of time that the parent job waits in the queue before its service starts. Clearly, the waiting and the service time of a job are independent in our QBD model.

By repeating the arguments from the proof of Theorem 7.6.1, we find that $W(r)$ can be expressed as:

**Theorem 8.4.1.** *The distribution of the waiting time is given by*

$$P[W(r) > t] = (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})e^{\mathbb{W}t} vec\langle I\rangle$$

*with* $\mathbb{W} = ((A_0(r) + A_1)' \otimes I) + ((A_{-1}(r))' \otimes R(r))$ *and where* $vec\langle\cdot\rangle$ *is the column stacking operator. The mean waiting time is*

$$\begin{aligned} E[W(r)] &= \int_0^\infty P[W(r) > t]\,dt \\ &= (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})(-\mathbb{W})^{-1} vec\langle I\rangle. \end{aligned}$$

Service time distribution $J(r)$ is more difficult to compute compared to the models in Chapter 7. This is due to the fact that child jobs can be stolen multiple times before finally going into service.

We define $J_{0,k}(r)$ as the distribution of the time that it takes for $k$ child jobs in a server to be completed ($k = 1, \dots, m$). Similarly, we define $J_{1,k}(r)$ as the distribution of the time that it takes for a parent job and $k$ child jobs in a server to be completed ($k = 0, \dots, m$). The service time distribution can then be expressed as

$$P[J(r) \leq t] = \sum_{k=0}^{m} p_k P[J_{1,k}(r) \leq t].$$

Clearly, $P[J_{0,1}(r) \leq t] = 1 - e^{-\mu_2 t}$ and $P[J_{1,0}(r) \leq t] = 1 - e^{-\mu_1 t}$. For $k > 1$, we can condition on the first service completion or steal event to find that

$$\begin{aligned} P[J_{0,k}(r) \leq t] = \int_0^t \Bigg( rq \sum_{j=1}^{k-1} \psi_{k-1,j} P[J_{0,k-j}(r) \leq t - s] P[J_{0,j}(r) \leq t - s] \\ + \mu_2 P[J_{0,k-1}(r) \leq t - s] \Bigg) e^{-(rq+\mu_2)s} ds, \end{aligned} \tag{8.9}$$

and for $k > 0$ this yields

$$P[J_{1,k}(r) \le t] = \int_0^t \left( rq \sum_{j=1}^{k} \phi_{k,j} P[J_{1,k-j}(r) \le t-s] P[J_{0,j}(r) \le t-s] \right.$$
$$\left. + \mu_1 P[J_{0,k}(r) \le t-s] \right) e^{-(rq+\mu_1)s} ds. \tag{8.10}$$

While the above formulas recursively determine the service time, they are less suited for numerical computations, we therefore also develop a recursive scheme for the mean service time.

Consider a set of $s$ servers, where the $k$-th server contains $i_k$ child jobs, where $s \ge 1$, $0 \le i_1 + \cdots + i_s \le m$ and $i_k \ge 0$ for $k = 1, \ldots, s$. Let $E_{i_1,\ldots,i_s}(r)$ be the expected time until all these child jobs have completed service. Define similarly $E^p_{i_1,\ldots,i_s}(r)$, except that the first server contains $i_1$ child jobs and a parent job (that is in service).

By definition, we can drop $i_k$'s that are zero (expect $i_1$ in $E^p_{i_1,\ldots,i_s}(r)$) and can permute the indices of $E_{i_1,\ldots,i_s}(r)$ and all indices except the first one of $E^p_{i_1,\ldots,i_s}(r)$. We have for $s \ge 1$

$$E_{1'_s}(r) = \frac{1}{\mu_2} \sum_{k=1}^{s} \frac{1}{k}. \tag{8.11}$$

We now define recursively, assuming $i_k \ge 1$ for $k = 1, \ldots, s$:

$$E_{i_1,\ldots,i_s}(r) = \frac{1}{s\mu_2 + rq \sum_{k=1}^{s} 1[i_k \ge 2]} \left( 1 + \mu_2 \sum_{k=1}^{s} E_{i_1,\ldots,i_{k-1},i_k-1,i_{k+1},\ldots,i_s}(r) \right.$$
$$\left. + rq \sum_{k=1}^{s} \sum_{n=1}^{i_k-1} \psi_{i_k-1,n} E_{i_1,\ldots,i_{k-1},i_k-n,i_{k+1},\ldots,i_s,n}(r) \right).$$

We have $E^p_0(r) = 1/\mu_1$ and we define recursively for $s \ge 1$

$$E^p_{1'_s}(r) = \frac{1}{\mu_1 + (s-1)\mu_2} \left( 1 + \mu_1 E_{1'_s}(r) + (s-1)\mu_2 E^p_{1'_{s-1}}(r) \right)$$
$$= \frac{1}{\mu_1 + (s-1)\mu_2} \left( 1 + \frac{\mu_1}{\mu_2} \sum_{k=1}^{s-1} \frac{1}{k} + (s-1)\mu_2 E^p_{1'_{s-1}}(r) \right),$$

where we have used (8.11) in the last equality. Finally, we define recursively, assuming $i_k \ge 1$ for $k = 2, \ldots, s$:

$$E^p_{i_1,\ldots,i_s}(r) = \frac{1}{\mu_1 + (s-1)\mu_2 + rq1[i_1 \ge 1] + rq \sum_{k=2}^{s} 1[i_k \ge 2]} \left( 1 \right.$$
$$+ \mu_1 E_{i_1,\ldots,i_s}(r) + rq \sum_{n=1}^{i_1} \phi_{i_1,n} E^p_{i_1-n,i_2,\ldots,i_s,n}(r)$$

$$+ \mu_2 \sum_{k=2}^{s} E^p_{i_1,\dots,i_{k-1},i_k-1,i_{k+1},\dots,i_s}(r)$$
$$+ rq \sum_{k=2}^{s} \sum_{n=1}^{i_k-1} \psi_{i_k-1,n} E^p_{i_1,\dots,i_{k-1},i_k-n,i_{k+1},\dots,i_s,n}(r) \Bigg).$$

The expectation $E[J(r)]$ can now be computed as:

$$E[J(r)] = \sum_{k=0}^{m} p_k E^p_k(r).$$

## 8.5 Limiting behaviour

In this section we remark how the QBD simplifies when $r = 0$ and $r \to \infty$. If $r = 0$, i.e. if there is no stealing, the cavity queue becomes an $M/PH/1$-queue. The case where $r$ goes to infinity is more interesting. We have $p_{0,i'}(r) \to 0$ for $i' = 1, \dots, m$. Due to (8.5), we get $p_{1,0}(r) \to 1$. Let $0_{p,q}$ denote the zero matrix of dimensions $p \times q$. Using the interpretation that $(i, j)$-th entry of $G(r)$ is the chance that the server goes from phase $i$ to $j$ at a time of a level decrease (when the new level is at least 1), we get

$$G(r) \to \begin{bmatrix} 1_m \; 0_{m,m-1} & 0_{m,m+1} \\ 0_{m+1,m} & 1_{m+1} \; 0_{m+1,m} \end{bmatrix}.$$

We have $\pi_*(r) + \pi_{(0,1,0)}(r) + \pi_{(0,0,1)}(r) \to 1$ and thus $R(r) \to 0$. Note, that in general

$$\sum_{\ell \geq 0} \sum_{i=0}^{m} \pi_{\ell,i,1}(r) = \frac{\lambda}{\mu_1}$$

holds. For $r \to \infty$, it follows that $\pi_{(0,0,1)}(r) \to \lambda/\mu_1$ and $\pi_{(0,1,0)}(r) \to \rho - \lambda/\mu_1$. Obviously, $W(r) \to 0$ and

$$P[J_{0,k}(r) \leq t] = P[J_{0,1}(r) \leq t]^k,$$
$$P[J_{1,k'}(r) \leq t] = P[J_{1,0}(r) \leq t] P[J_{0,1}(r) \leq t]^{k'}$$

for $k = 2, \dots, m$ and for $k' = 1, \dots, m$. Further, we have

$$E[J(r)] \to \sum_{k=0}^{m} p_k E^p_{0,1'_k}(r).$$

Let $d = \sum_{n=1}^{m} n p_n$ denote the average number of children spawned per parent. Suppose $d \in \mathbb{N}$. Then, due to Proposition 7.7.1, for $r \to \infty$ and $d$ child jobs spawned per parent on average, the QBD with the worst mean service time is the one where a parent always spawns $d$ child jobs ($p_d = 1$).

## 8.6 Numerical experiments

In this section we perform numerical experiments to compare the performance of several stealing strategies. Due to the lack of space, we present only a subset of the experiments performed. The main conclusions in these additional experiments (e.g., different $\mu_2$ values) are in agreement with the results presented. Define $\Psi$ as the matrix where $[\Psi]_{i,j} = \psi_{i,j}$ and define $\Phi$ similarly. Note that a strategy is fully characterized by $\Psi$ and $\Phi$. The strategies considered are as follows:

1. **Steal one:** The strategy of always stealing one child job, that is $\phi_{i,1} = \psi_{i,1} = 1$ for every $i$.
2. **Steal half:** The strategy of always stealing half of the pending child jobs. If $n$, the number of pending child jobs, is uneven, there is a fifty percent chance that $\lfloor n/2 \rfloor$ child jobs get stolen and $\lceil n/2 \rceil$ jobs otherwise;
3. **Steal all:** The strategy of stealing all of the pending child jobs, that is $\phi_{i,i} = \psi_{i,i} = 1$ for every $i$.

Note that these strategies do not rely on any knowledge on the (mean) job sizes or system load.

We compare the mean response time for these strategies with the optimal monotone deterministic strategy. A strategy is called deterministic if for every $i \leq m$ there exists a $j \leq i$ such that $\phi_{i,j} = 1$ and a $k \leq i$ such that $\psi_{i,k} = 1$. It is called monotone deterministic (MD) if in addition having $\psi_{i,j} = 1$ and $\psi_{i',j'} = 1$ with $i < i'$ implies that $j \leq j'$ for all $i, j, i', j'$ and the same holds for $\Phi$. Experiments not included in the chapter suggest that the optimal strategy, that is, the optimal $\Psi$ and $\Phi$ matrices, corresponds to an MD strategy. The optimal MD strategy is determined using brute-force and its mean response time is denoted as $T_{MD}(r)$. Let $\mathbf{p} = [p_0, p_1, \ldots, p_m]$.

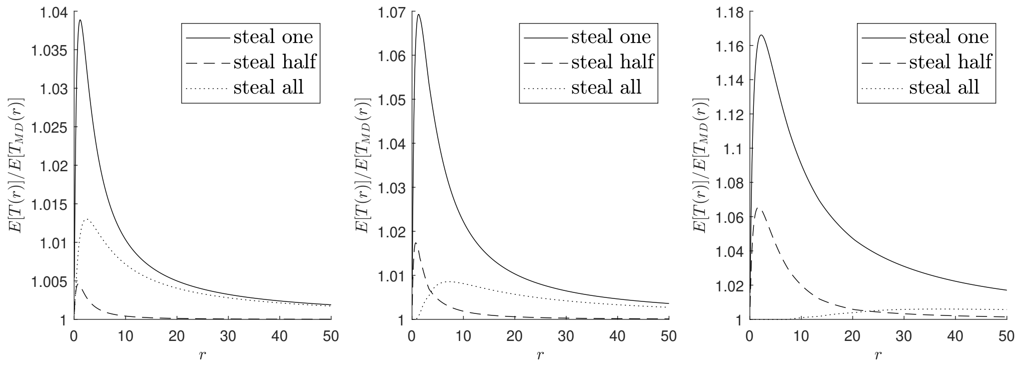


Figure 8.2: Example 8.6.1: $E[T(r)]/E[T_{MD}(r)]$ in function of $r$ with $\rho = 0.15$ (left), $\rho = 0.5$ (mid) and $\rho = 0.85$ (right).

**Example 8.6.1.** In Figure 8.2 we examine the effect of increasing the steal rate on how well the three strategies perform compared to the optimal MD strategy. We do this for $\rho \in \{0.15, 0.5, 0.85\}$, $\mu_1 = 1$, $\mu_2 = 2$, $\mathbf{p} = 1_5'/5$ and $r \in [0.05, 50]$. We note that there exists no universal best strategy. The strategy of stealing one job performs the worst. This is due to the fact that relatively very little work of the pending jobs is transferred. When $\mu_2 < \mu_1$, examples can be constructed where the strategy of stealing a single child outperforms the others. For moderately high values of $r$ or for low loads the strategy where half of the child jobs get stolen is close to the optimal MD strategy, which is intuitively clear as there is a small chance that there are pending parent jobs in a queue. In fact, it seems that as $r$ becomes large enough the optimal strategy for systems where $\mu_1 \leq \mu_2$ is stealing $\lfloor i/2 \rfloor + 1$ out of $i$ children. For low values of $r$ the strategy of stealing all child jobs performs well, as there is a fair chance that there are pending parents in the queue and it can take a long time until the server is probed again.

For $\rho = 0.85$ the matrices $\Psi, \Phi$ of the optimal MD strategy change as follows: for low values of $r$ the best strategy is the one of stealing all jobs, that is $\Psi$ and $\Phi$ are identity matrices of size $m - 1$ and $m$ respectively. Then, at approximately $r = 7.6$, $\psi_{3,2}$ becomes one. Around $r = 13.5$, the $\phi_{4,3}$ becomes one and finally $\phi_{3,2} = 1$ around $r = 20.35$. For $\rho = 0.5$ we see a similar evolution: for low values of $r$ the best strategy is stealing all child jobs. Then, at approximately $r = 0.85$, $\psi_{3,2}$ becomes one. Around $r = 1.55$, the $\phi_{4,3}$ becomes one and finally $\phi_{3,2} = 1$ around $r = 3.35$.

Denote for $n \geq 0$

$$C(k) = \frac{(2k)!}{(k+1)!k!}.$$

The sequence $(C(k))_{k\geq 0}$ is the sequence of the Catalan numbers [15, Section 2]. We now provide a formula for the number of MD strategies.

**Proposition 8.6.2.** *For a given $m$, there exist $C(m-1)C(m)$ monotone deterministic strategies.*

*Proof.* We prove that there are $C(m)$ different choices for the matrix $\Phi$. For an MD strategy, we can encode the matrix $\Phi$ as a vector $v$ of length $m$ of increasing numbers from the set $\{1, \ldots, m\}$, where $[v]_i = j$ if and only if $\phi_{i,j} = 1$. Clearly, there is a one-to-one correspondence between matrices $\Phi$ of MD strategies and vectors $v$. Further, the different possibilities for $v$ are exactly the different Catalan combinations if we subtract 1 from every entry of $v$ [15, Section 2]. Hence, we have $C(m)$ different choices for $\Phi$.

For $m > 1$, we can show in a similar way that there are $C(m-1)$ choices for $\Psi$. For $m = 1$, $\Psi$ is the empty matrix, hence we only have one choice for $\Psi$. The claim now follows. □

Hence, the number of MD strategies grows quickly in function of $m$. This implies that it can take a long time to determine the optimal MD strategy for systems with larger $m$ values. We therefore introduce a smaller family of strategies and compare our three strategies with the optimal strategy in this smaller family to limit the brute-force search. We call a strategy bounded monotone deterministic (BMD) if it is monotone deterministic and $\psi_{i,j} = 1$ implies $\psi_{i+1,j} = 1$ or $\psi_{i+1,j+1} = 1$ for every $i$ and the same holds for $\Phi$. Note that there are $2^{m-2}2^{m-1} = 2^{2m-3}$ BMD strategies for a given $m \geq 2$. The optimal BMD strategy is determined using brute-force and we denote its mean response time by $T_{BMD}(r)$. The mean response time of the optimal BMD strategy may exceed that of the optimal MD strategy as indicated in the next example.

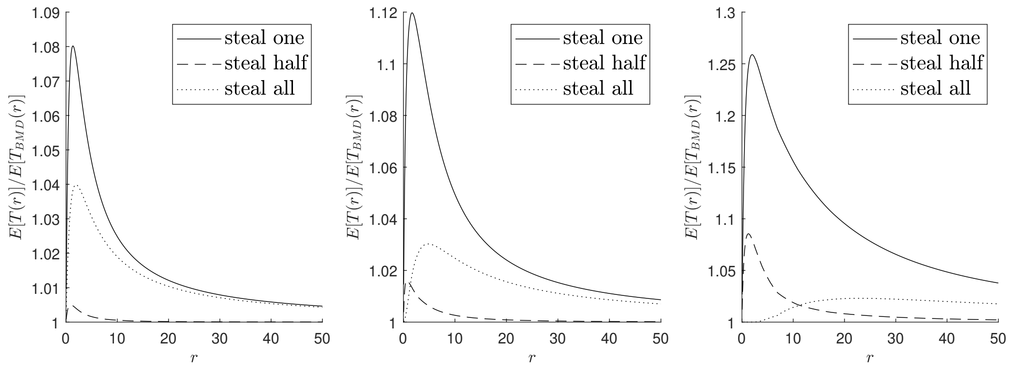


Figure 8.3: Example 8.6.3: $E[T(r)]/E[T_{BMD}(r)]$ in function of $r$ with $\rho = 0.15$ (left), $\rho = 0.5$ (mid) and $\rho = 0.85$ (right).

**Example 8.6.3.** In Figure 8.3 we examine the effect of increasing the steal rate on how well the three strategies perform compared to the optimal BMD strategy when $m = 6$ instead of $m = 4$ as in the previous example. We do this for $\rho \in \{0.15, 0.5, 0.85\}, \mu_1 = 1, \mu_2 = 2, \mathbf{p} = 1'_7/7$ and $r \in [0.05, 50]$. It is clear that the main insights are similar as in the $m = 4$ case, except that more substantial gains can be achieved by optimizing $\Psi$ and $\Phi$. We also performed some experiments to compare the performance of the optimal MD and BMD strategies and noted that for $r \in [6.9, 7.4]$, the optimal MD strategy has $\psi_{3,2} = 1$ and $\psi_{4,4} = 1$, which is not BMD. The reduction in the mean response time was however very limited.

**Example 8.6.4.** In Figure 8.4 we illustrate the effect of increasing the load $\rho$ on the mean response time. We do this for $\rho \in [0.05, 0.95], \mu_1 = 1, \mu_2 = 2, \mathbf{p} = 1'_5/5$ and $r \in \{0.1, 1, 10\}$. These result confirm that stealing all is best when the load is sufficiently high, while stealing half of the child jobs is good for systems with a limited load.

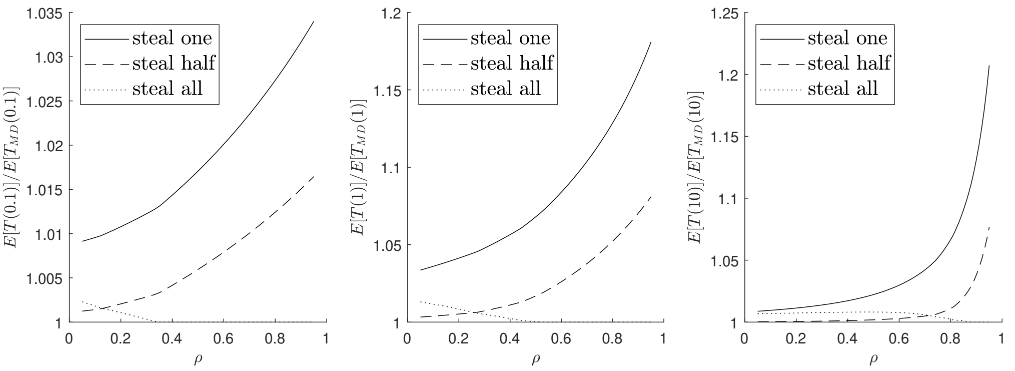


Figure 8.4: Example 8.6.4: $E[T(r)]/E[T_{MD}(r)]$ in function of $\rho$ with $r = 0.1$ (left), $r = 1$ (mid) and $r = 10$ (right).

## 8.7 Model validation

Based on numerical experiments in the previous section, we see that stealing all or half of the children are good stealing policies, stealing all works best for low values of $r$, while stealing half of the children works well for higher values. Therefore, we validate the mean field model for these two policies. We always start the simulations from an empty system and simulate the behaviour for $T = 10^5$ with a warm up period of 33% of $T$.
In Figures 8.5 and 8.6, we see that there is an excellent match between the simulated waiting and service times and those of the QBD model (calculated using Section 8.4) in case of stealing all and half of the children respectively. The 95% confidence intervals were computed based on 5 runs with $N = 500$ servers, $m = 4$, $\mu_1 = 1, \mu_2 = 2, \rho = 0.75$, $\mathbf{p} = (1, 1, 1, 1, 1)/5$ and $r \in \{1, 5\}$.

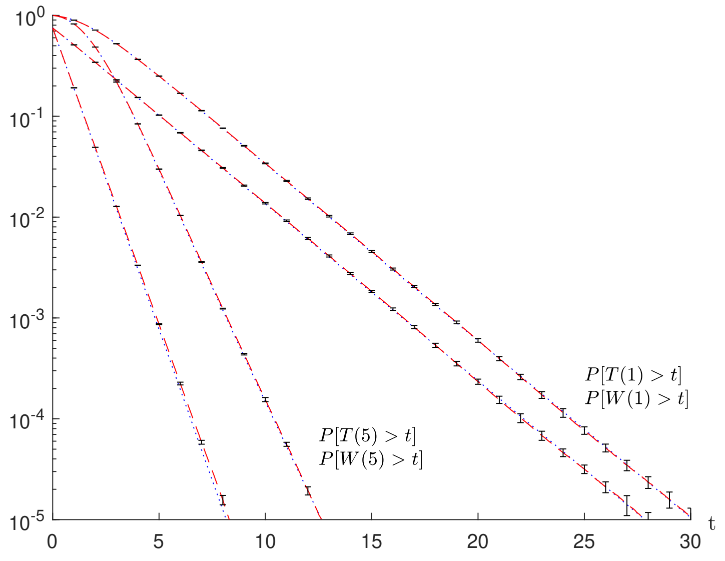


Figure 8.5: Waiting and response times from the QBD (blue dots) and simulations (red dashed line) with confidence intervals for 5 runs.

In Tables 8.2 and 8.3 we compare the relative error of the simulated mean response time, based on 20 runs, to the one obtained from Section 8.4. We do this for $\mu_1 = 1, \mu_2 = 2$, $\mathbf{p} = 1_5'/5$, $\rho \in \{0.75, 0.85\}$, $r \in \{1, 10\}$ and $N \in \{250, 500, 1000, 2000, 4000\}$. Tables 8.2 and 8.3 show the comparison for the strategy of stealing all and half of the children respectively.
The relative error in all cases is below 1.5% and tends to increase with the steal rate $r$. Further, the relative error seems roughly to halve when doubling $N$, which is in agreement with the results in [24].

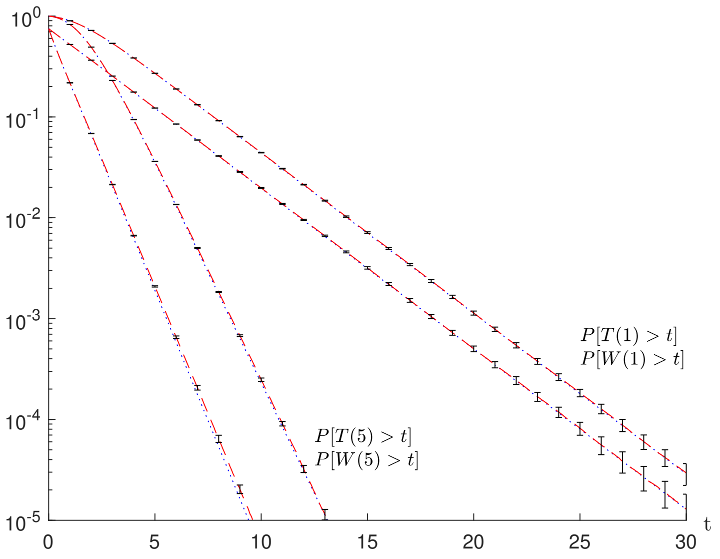


Figure 8.6: Waiting and response times from the QBD (blue dots) and simulations (red dashed line) with confidence intervals for 5 runs.

Table 8.2: Relative error of simulation results for $E[T(r)]$, based on 20 runs

| | $\rho = 0.75$ | | $\rho = 0.85$ | |
|---|---|---|---|---|
| $N$ | sim. ± conf. | rel.err.% | sim. ± conf. | rel.err.% |
| $r = 1$ | | | | |
| 250 | 3.7650 ± 2.41e-03 | 0.2986 | 5.5121 ± 6.89e-03 | 0.3386 |
| 500 | 3.7588 ± 1.56e-03 | 0.1334 | 5.5053 ± 3.62e-03 | 0.2157 |
| 1000 | 3.7568 ± 1.38e-03 | 0.0818 | 5.4980 ± 3.58e-03 | 0.0821 |
| 2000 | 3.7548 ± 7.33e-04 | 0.0283 | 5.4945 ± 1.96e-03 | 0.0197 |
| 4000 | 3.7541 ± 5.66e-04 | 0.0091 | 5.4953 ± 1.56e-03 | 0.0344 |
| QBD | 3.7537 | | 5.4935 | |
| $r = 10$ | | | | |
| 250 | 1.7766 ± 4.72e-04 | 0.7247 | 2.1371 ± 1.41e-03 | 1.2816 |
| 500 | 1.7701 ± 3.42e-04 | 0.3553 | 2.1232 ± 8.85e-04 | 0.6249 |
| 1000 | 1.7671 ± 1.84e-04 | 0.1894 | 2.1165 ± 5.59e-04 | 0.3090 |
| 2000 | 1.7655 ± 1.59e-04 | 0.0957 | 2.1131 ± 4.20e-04 | 0.1454 |
| 4000 | 1.7646 ± 1.61e-04 | 0.0437 | 2.1119 ± 1.79e-04 | 0.0878 |
| QBD | 1.7638 | | 2.1100 | |

## 8.8 Mean field model

In this section we write down a set of ODEs which captures the transient evolution of the system as $N \to \infty$. We show that the invariant distribution of the QBD, introduced in Section 8.3, coincides with the fixed point of this set of ODEs. This also implies that

Table 8.3: Relative error of simulation results for $E[T(r)]$, based on 20 runs

| | $\rho = 0.75$ | | $\rho = 0.85$ | |
|---|---|---|---|---|
| $N$ | sim. ± conf. | rel.err.% | sim. ± conf. | rel.err.% |
| $r = 1$ | | | | |
| 250 | 3.9305 ± 3.24e-03 | 0.2392 | 5.8435 ± 6.51e-03 | 0.2830 |
| 500 | 3.9261 ± 2.82e-03 | 0.1271 | 5.8331 ± 3.76e-03 | 0.1045 |
| 1000 | 3.9231 ± 1.24e-03 | 0.0506 | 5.8288 ± 2.33e-03 | 0.0307 |
| 2000 | 3.9225 ± 1.04e-03 | 0.0353 | 5.8281 ± 2.97e-03 | 0.0187 |
| 4000 | 3.9219 ± 6.06e-04 | 0.0200 | 5.8279 ± 2.09e-03 | 0.0153 |
| QBD | 3.9211 | | 5.8270 | |
| $r = 10$ | | | | |
| 250 | 1.7822 ± 5.23e-04 | 0.7748 | 2.1782 ± 1.32e-03 | 1.3017 |
| 500 | 1.7752 ± 4.47e-04 | 0.3790 | 2.1642 ± 7.18e-04 | 0.6506 |
| 1000 | 1.7720 ± 2.80e-04 | 0.1965 | 2.1576 ± 6.31e-04 | 0.3437 |
| 2000 | 1.7703 ± 1.98e-04 | 0.1007 | 2.1537 ± 4.07e-04 | 0.1623 |
| 4000 | 1.7695 ± 8.56e-05 | 0.0567 | 2.1520 ± 3.69e-04 | 0.0832 |
| QBD | 1.7685 | | 2.1502 | |

the ODEs have a unique fixed point.

We denote by $f_{\ell,j,k}(t)$ the fraction of queues at time $t$ with $\ell$ parent jobs in waiting in the queue, $j \in \{1,\ldots,m\}$ child jobs in the queue and $k \in \{0,1\}$ describing whether a parent job is in service ($k = 1$) or not ($k = 0$). Note that $\ell$ does not count parent jobs in service, whereas $j$ counts child jobs waiting and in service. In particular for $\ell = 0$ and $j + k \geq 1$ the server is busy and there may be child jobs waiting, which can be transferred. We denote $f_{0,0,0}(t)$ as $f_*(t)$, the fraction of idle queues. For a statement $A$ we set $1[A]$ to be 1 if $A$ is true and 0 if $A$ is false. We have for $\ell \geq 0$ and $j + k \geq 1$,

$$\begin{aligned}
\frac{d}{dt} f_{\ell,j,k}(t) &= \lambda f_{\ell-1,j,k}(t)1[\ell \geq 1] + \lambda p_j f_*(t)1[\ell = 0, k = 1] - \lambda f_{\ell,j,k}(t) \\
&+ \mu_1 f_{\ell,j,k+1}(t)1[k = 0] + \mu_1 p_j f_{\ell+1,0,k}(t)1[k = 1] - \mu_1 f_{\ell,j,k}(t)1[k = 1] \\
&+ \mu_2 f_{\ell,j+1,k}(t)1[j \leq m-1, k = 0] + \mu_2 p_j f_{\ell+1,1,k-1}(t)1[k = 1] - \mu_2 f_{\ell,j,k}(t)1[k = 0] \\
&+ r f_*(t) f_{\ell+1,j,k}(t)1[j + k = 1] - r f_*(t) f_{\ell,j,k}(t)1[\ell \geq 1] \\
&+ r f_*(t) \sum_{s=1}^{m-j} \psi_{j+s-1,s} f_{\ell,j+s,k}(t)1[1 \leq j, k = 0] + r f_*(t) \sum_{s=1}^{m-j} \phi_{j+s,s} f_{\ell,j+s,k}(t)1[k = 1] \\
&- r f_*(t) f_{\ell,j,k}(t)1[\ell = 0, j + k > 1] \\
&+ r f_*(t) \sum_{j' \geq 1,\ell'} \sum_{s=1}^{m} \phi_{j',s} f_{\ell',j',1}(t)1[\ell = 0, j = s, k = 0] \\
&+ r f_*(t) \sum_{j' > 1,\ell'} \sum_{s=1}^{m-1} \psi_{j'-1,s} f_{\ell',j',0}(t)1[\ell = 0, j = s, k = 0] \\
&+ r f_*(t) p_j \sum_{\ell' \geq 1, j'+k'=1} f_{\ell',j',k'}(t)1[\ell = 0, k = 1],
\end{aligned}$$

and for $\ell, j, k = 0$,

$$\frac{d}{dt}f_*(t) = -\lambda f_*(t) + \mu_1 f_{0,0,1}(t) + \mu_2 f_{0,1,0}(t) - r f_*(t)(1 - f_*(t) - f_{0,0,1}(t) - f_{0,1,0}(t)).$$

The first three terms of the drift of $f_{\ell,j,k}(t)$ correspond to arrivals, the following three terms correspond to job completions of parent jobs, the following three correspond to service completions of child jobs and the remaining terms correspond to job transfers: the first term is due to parent job transfers, the second is due to both parent and child job transfers, the third and fourth are due to child job transfers, the fifth one is due to child job transfers, the next two are due to child job transfers to empty servers and the last one is due to parent job transfers to empty servers.

Similarly for $f_*(t)$, the first term is due to job arrivals, the next two are due to service completions and the last is due to job transfers. Note that $1 - f_*(t) - f_{0,0,1}(t) - f_{0,1,0}(t) = \sum_{\ell,j+k>1} f_{\ell,j,k}(t) + \sum_{\ell\geq 1, j+k=1} f_{\ell,j,k}(t)$. We set

$$\vec{f}_\ell(t) = (f_{\ell,1,0}(t), \dots, f_{\ell,m,0}(t), f_{\ell,0,1}(t), \dots, f_{\ell,m,1}(t))$$

for every $\ell \geq 0$. We can then rewrite the ODEs in matrix form: for $\ell \geq 0$ and $j + k \geq 1$ we have

$$\begin{aligned}\frac{d}{dt}\vec{f}_\ell(t) &= \lambda \vec{f}_{\ell-1}(t)1[\ell \geq 1] - \lambda \vec{f}_\ell(t) + \lambda f_*(t)\alpha 1[\ell = 0] \\ &+ \vec{f}_\ell(t)\tilde{S}(t) + \vec{f}_{\ell+1}(t)\mu\alpha + r f_*(t)\vec{f}_{\ell+1}(t)V_0 \\ &- r f_*(t)\vec{f}_\ell(t)(I - V_0)1[\ell = 0] - r f_*(t)\vec{f}_\ell(t)1[\ell \geq 1] \\ &+ r f_*(t)\sum_{\ell' \geq 0} \vec{f}_{\ell'}(t)T1[\ell = 0] \\ &+ r f_*(t)\sum_{\ell' \geq 1} \vec{f}_{\ell'}(t)v_0\alpha 1[\ell = 0],\end{aligned} \tag{8.12}$$

and for $\ell, j, k = 0$

$$\frac{d}{dt}f_*(t) = -\lambda f_*(t) + \vec{f}_0(t)\mu - r f_*(t)\big(1 - f_*(t) - \vec{f}_0(t)v_0\big), \tag{8.13}$$

with $v_0 = \begin{bmatrix}1 & 0'_{m-1} & 1 & 0'_m\end{bmatrix}'$, where the entries are non-zero when $j + k = 1$ (i.e. $V_0 = \mathrm{diag}(v_0)$). The matrix $\tilde{S}(t)$ is defined as

$$\tilde{S}(t) = \begin{bmatrix}\tilde{S}_{00}(t) & S_{01} \\ S_{10} & \tilde{S}_{11}(t)\end{bmatrix}, \tag{8.14}$$

where $\tilde{S}_{00}(t)$ and $\tilde{S}_{11}(t)$ are the same matrices as $S_{00}(r)$ and $S_{11}(r)$ respectively, except with every instance of $q$ changed to $f_*(t)$. The $(2m+1) \times (2m+1)$ matrix $T = T_\psi + T_\phi$ records the distribution of the number of child jobs transferred when a probe is successful:

$$T_\psi = \begin{bmatrix} 0 & \dots & 0 & 0'_{m+2} \\ \psi_{1,1} & \dots & \psi_{1,m-1} & \\ \vdots & & \vdots & \vdots \\ \psi_{m-1,1} & \dots & \psi_{m-1,m-1} & 0'_{m+2} \\ 0_{m+1} & \dots & 0_{m+1} & 0_{m+1,m+2} \end{bmatrix}, \quad T_\phi = \begin{bmatrix} 0'_{2m+1} & & & \\ \vdots & & & \\ 0'_{2m+1} & & & \\ \phi_{1,1} & \dots & \phi_{1,m} & 0'_{m+1} \\ \vdots & & \vdots & \vdots \\ \phi_{m,1} & \dots & \phi_{m,m} & 0'_{m+1} \end{bmatrix}.$$

Note that if $\phi_{i,1} = 1$ for every $i \in \{1, \dots, m\}$ and if $\psi_{j,1} = 1$ for every $j \in \{1, \dots, m-1\}$, then $T = (\mathbf{1} - v_0)e_1$.

We show that the stationary distribution of the QBD corresponds to the unique fixed point $\zeta$ of the set of ODEs in Equations (8.12)-(8.13).

**Lemma 8.8.1.** *For any fixed point* $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ *with* $\zeta_* + \sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1$ *of the set of ODEs in Equations* (8.12)-(8.13) *we have*

$$\lambda = \lambda\zeta_* + \sum_{\ell \geq 1} \vec{\zeta}_\ell \mu + r\zeta_* \sum_{\ell \geq 1} \vec{\zeta}_\ell v_0.$$

*Proof.* As $\frac{d}{dt}\vec{f}_\ell(t) = 0$ in a fixed point we get using $\sum_{\ell \geq 0}(\ell+1)\frac{d}{dt}\vec{f}_\ell(t)\mathbf{1} = 0$ that

$$\sum_{\ell \geq 0} \vec{\zeta}_\ell \mu = \lambda + r\zeta_* \left(1 - \zeta_* - \sum_{\ell \geq 0} \zeta_\ell v_0\right). \tag{8.15}$$

The claim now follows by using (8.15) and (8.13) in a fixed point. □

Define recursively

$$\xi_{1,m} = \lambda p_m$$

and

$$\xi_{1,k} = \lambda p_k + \frac{r\zeta_*}{r\zeta_* + \mu_1} \sum_{d=k+1}^{m} \phi_{d,d-k} \xi_{1,d},$$

for $k = 0, \dots, m-1$, and

$$\begin{aligned} \xi_{0,k'} = {} & \frac{\mu_1}{r\zeta_* + \mu_1} \xi_{1,k'} + \frac{r\zeta_*}{r\zeta_* + \mu_1} \sum_{d=k'}^{m} \phi_{d,k'} \xi_{1,d} \\ & + 1[k' < m] \frac{\mu_2}{r\zeta_* + \mu_2} \xi_{0,k'+1} + \frac{r\zeta_*}{r\zeta_* + \mu_2} \sum_{d=k'+1}^{m} (\psi_{d-1,d-k'} + \psi_{d-1,k'}) \xi_{0,d}, \end{aligned}$$

for $k' = 1, \dots, m$. $\xi_{i,j}$ is the rate at which servers enter into phase $(i, j)$.

**Lemma 8.8.2.** *For any fixed point* $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ *with* $\zeta_* + \sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1$ *of the set of ODEs in Equations* (8.12)-(8.13) *we have for* $1 \leq k \leq m$*:*

$$(r\zeta_* + \mu_1) \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{m+k} \\ 1 \\ 0_{m-k} \end{pmatrix} = \xi_{1,k}. \tag{8.16}$$

*Proof.* We prove the lemma using complete backward induction on $k$. By demanding that $\sum_{\ell \geq 0} \vec{f}_\ell(t)[0'_{m+k}, 1, 0'_{m-k}]' = 0$ for any $k \in \{1, \dots, m\}$, we find due to Lemma 8.8.1 that

$$0 = \lambda p_k - (r\zeta_* + \mu_1) \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{m+k} \\ 1 \\ 0_{m-k} \end{pmatrix} + r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell \left[0'_{m+k+1}, \phi_{k+1,1}, \phi_{k+2,2}, \dots, \phi_{m,m-k}\right]'.$$

This is equivalent to

$$(r\zeta_* + \mu_1)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{m+k} \\ 1 \\ 0_{m-k} \end{pmatrix} = \lambda p_k + r\zeta_* \sum_{\ell \geq 0}\vec{\zeta}_\ell \left[0'_{m+k+1}, \phi_{k+1,1}, \dots, \phi_{m,m-k}\right]'. \tag{8.17}$$

(8.17) is equivalent to (8.16) for $k = m$. Suppose now that $k < m$ and that (8.16) holds for all $k' \in \{k+1, \dots, m\}$. Due to (8.17), it suffices to show that

$$r\zeta_* \sum_{\ell\geq 0}\vec{\zeta}_\ell \left[0'_{m+k+1}, \phi_{k+1,1}, \dots, \phi_{m,m-k}\right]' = \frac{r\zeta_*}{r\zeta_* + \mu_1}\sum_{d=k+1}^{m}\phi_{d,d-k}\xi_{1,d},$$

which holds due to induction hypothesis. □

**Lemma 8.8.3.** *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ with $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (8.12)-(8.13) *we have for $2 \leq k \leq m$:*

$$(r\zeta_* + \mu_2)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{k-1} \\ 1 \\ 0_{2m-k+1} \end{pmatrix} = \xi_{0,k}. \tag{8.18}$$

*Proof.* Similarly to the proof of Lemma 8.8.2, we use complete backward induction on $k$. By demanding $\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)[0'_{m-1}, 1, 0'_{m+1}]' = 0$, we get

$$\begin{aligned}(r\zeta_* + \mu_2)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{m-1} \\ 1 \\ 0_{m+1} \end{pmatrix} &= (r\zeta_*\phi_{m,m} + \mu_1)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{2m} \\ 1 \end{pmatrix} \\ &= (r\zeta_*\phi_{m,m} + \mu_1)\frac{\xi_{1,m}}{r\zeta_* + \mu_1} \\ &= \xi_{0,m},\end{aligned}$$

where (8.16) was used in the second equality. This shows (8.18) for $k = m$. Suppose $k < m$ and that (8.18) holds for all $k' \in \{k+1, \dots, m\}$. By demanding

$$\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)\begin{pmatrix} 0_{k-1} \\ 1 \\ 0_{2m-k+1} \end{pmatrix} = 0,$$

we get

$$\begin{aligned}(r\zeta_* + \mu_2)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{k-1} \\ 1 \\ 0_{2m-k+1} \end{pmatrix} &= r\zeta_* \sum_{\ell\geq 0}\vec{\zeta}_\ell \left[0'_k, \psi_{k,1} + \psi_{k,k}, \dots, \psi_{m-1,m-k} + \psi_{m-1,k}, 0'_{m+1}\right]' \\ &+ r\zeta_* \sum_{\ell\geq 0}\vec{\zeta}_\ell \left[0'_{m+k}, \phi_{k,k}, \dots, \phi_{m,k}\right]' \\ &+ \mu_2 \sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_k \\ 1 \\ 0_{2m-k} \end{pmatrix} + \mu_1 \sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{m+k} \\ 1 \\ 0_{m-k} \end{pmatrix}\end{aligned}$$

By using induction hypothesis and (8.16) we further get

$$(r\zeta_* + \mu_2)\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{k-1} \\ 1 \\ 0_{2m-k+1} \end{pmatrix} = \frac{r\zeta_*}{r\zeta_* + \mu_2}\sum_{d=k+1}^{m}(\psi_{d-1,d-k} + \psi_{d-1,k})\xi_{0,d} + \frac{r\zeta_*}{r\zeta_* + \mu_1}\sum_{d=k}^{m}\phi_{d,k}\xi_{1,d}$$
$$+ \frac{\mu_2}{r\zeta_* + \mu_2}\xi_{0,k+1} + \frac{\mu_1}{r\zeta_* + \mu_1}\xi_{1,k}$$

which is equal to $\xi_{0,k}$. This finishes the proof. □

**Proposition 8.8.4.** *For any fixed point* $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ *with* $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell \mathbf{1} = 1$ *of the set of ODEs in Equations* (8.12)-(8.13) *we have*

$$\zeta_* = q, \tag{8.19}$$

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T = \zeta_*\sum_{j=1}^{m}\lambda_{c,j}(r)e_j, \tag{8.20}$$

*where* $\lambda_{c,j}(r)$ *was defined in* (8.6).

*Proof.* Denote by 1:$k$ the column vector $[1, \ldots, k]'$ for $k \geq 1$. To prove (8.19) it suffices to show

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_m \\ 1_{m+1} \end{pmatrix} = \frac{\lambda}{\mu_1}, \tag{8.21}$$

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 1_m \\ 0_{m+1} \end{pmatrix} = \frac{\lambda}{\mu_2}\left(\sum_{i=1}^{m} ip_i\right). \tag{8.22}$$

By demanding $\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)\begin{pmatrix} 1_m \\ 0_{m+1} \end{pmatrix} = 0$, we find

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} \mu_2 \\ 0_m \\ 0_m \end{pmatrix} = r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell(\mathbf{1} - v_0) + \sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{m+1} \\ \mu_1 1_m \end{pmatrix}. \tag{8.23}$$

Combining (8.15) and (8.23) yields (8.21). By demanding

$$\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)\begin{pmatrix} 1{:}m \\ 0 \\ 1{:}m \end{pmatrix} = 0$$

and by using Lemma 8.8.1, one can show that

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} \mu_2 1_m \\ 0_{m+1} \end{pmatrix} = \lambda\left(\sum_{i=1}^{m} ip_i\right),$$

which is equivalent to (8.22). To prove the second claim it suffices to show that for $i = 1, \ldots, m$ we have:

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T e_i' = \zeta_*\lambda_{c,i}(r).$$

This is equivalent to showing the following two equalities:

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\psi e_i' = \lambda \frac{rq}{rq+\mu_2} \sum_{j>i} p_{0,j}(r)\psi_{j-1,i} + q \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} p_k^j(r)\psi_{k-1,i} \frac{rq}{rq+\mu_2} \quad (8.24)$$

for $i = 1, \dots, m-1$, and

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\phi e_i' = \lambda \frac{rq}{rq+\mu_1} \sum_{j \geq i} p_{1,j}(r)\phi_{j,i}, \quad (8.25)$$

for $i = 1, \dots, m$. Due to (8.16), we have

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\phi e_i' = \frac{r\zeta_*}{r\zeta_* + \mu_1} \sum_{d \geq i} \phi_{d,i} \xi_{1,d}.$$

As

$$\xi_{1,d} = \lambda p_{1,d}(r), \quad (8.26)$$

where all instances of $q$ in the formula of $p_{1,d}(r)$ have been changed to $\zeta_*$, equation (8.25) follows from (8.19). Equation (8.24) requires more work to prove. Due to (8.18), it suffices to show that

$$\frac{r\zeta_*}{r\zeta_* + \mu_2} \sum_{k=i}^{m-1} \psi_{k,i}\xi_{0,k+1} = \lambda \frac{rq}{rq+\mu_2} \sum_{j>i} p_{0,j}(r)\psi_{j-1,i} + q \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} p_k^j(r)\psi_{k-1,i} \frac{rq}{rq+\mu_2}.$$

Due to (8.19), this is equivalent to showing

$$\sum_{k=i+1}^{m} \psi_{k-1,i}\xi_{0,k} = \lambda \sum_{k=i+1}^{m} p_{0,k}(r)\psi_{k-1,i} + q \sum_{k=i+1}^{m} \sum_{j=k}^{m} \lambda_{c,j}(r) p_k^j(r)\psi_{k-1,i}. \quad (8.27)$$

We show that for $k = 2, \dots, m$, we have

$$\xi_{0,k} = \lambda p_{0,k}(r) + q \sum_{j=k}^{m} \lambda_{c,j}(r) p_k^j(r) \quad (8.28)$$

and (8.27) then follows. Intuitively, equation (8.28) is clear: the rate at which jobs arrive in servers in state $(0,k)$ is the sum of the following three rates:

1. the rate at which servers spawn child jobs from parent jobs and eventually go into phase $(0,k)$,
2. the rate at which $k$ jobs get transferred, and
3. the rate at which servers go into phase $(0,k)$ after starting in phases $(0,j)$ for $j = k+1, \dots, m$.

We prove (8.28) by complete backward induction on $k$. By definition and (8.19), we have for $k = m$

$$\xi_{0,m} = \frac{\mu_1}{r\zeta_* + \mu_1}\xi_{1,m} + \frac{r\zeta_*}{r\zeta_* + \mu_1}\phi_{m,m}\xi_{1,m} = \lambda p_{0,m}(r) + q\lambda_{c,m}(r).$$

Suppose now that $k < m$ and that (8.28) holds for all $k' \in \{k+1, \dots, m\}$. We have by definition

$$\begin{aligned}\xi_{0,k} = &\frac{\mu_1}{r\zeta_* + \mu_1}\xi_{1,k} + \frac{r\zeta_*}{r\zeta_* + \mu_1}\sum_{d=k}^{m}\phi_{d,k}\xi_{1,d}\\ &+ \frac{\mu_2}{r\zeta_* + \mu_2}\xi_{0,k+1} + \frac{r\zeta_*}{r\zeta_* + \mu_2}\sum_{d=k+1}^{m}(\psi_{d-1,d-k} + \psi_{d-1,k})\xi_{0,d}.\end{aligned}$$

By induction hypothesis, (8.19) and (8.26) this is equal to

$$\begin{aligned}&\lambda\frac{\mu_1}{rq + \mu_1}p_{1,k}(r) + \lambda\frac{rq}{rq + \mu_1}\sum_{d=k}^{m}p_{1,d}(r)\phi_{d,k}\\ &+ \frac{\mu_2}{rq + \mu_2}\left(\lambda p_{0,k+1}(r) + q\sum_{j=k+1}^{m}\lambda_{c,j}(r)p^j_{k+1}(r)\right)\\ &+ \frac{rq}{rq + \mu_2}\sum_{d=k+1}^{m}(\psi_{d-1,d-k} + \psi_{d-1,k})\left(\lambda p_{0,d}(r) + q\sum_{i=d}^{m}\lambda_{c,i}(r)p^i_d(r)\right).\end{aligned}$$

By first using the formula for $p_{0,k}(r)$ and then for $\lambda_{c,k}(r)$ (8.6) this is further equal to

$$\begin{aligned}&\lambda p_{0,k}(r) + \lambda\frac{rq}{rq + \mu_1}\sum_{d=k}^{m}p_{1,d}(r)\phi_{d,k}\\ &+ q\frac{\mu_2}{rq + \mu_2}\sum_{j=k+1}^{m}\lambda_{c,j}(r)p^j_{k+1}(r) + \lambda\frac{rq}{rq + \mu_2}\sum_{d=k+1}^{m}\psi_{d-1,k}p_{0,d}(r)\\ &+ q\frac{rq}{rq + \mu_2}\sum_{i=k+1}^{m}\lambda_{c,i}(r)\sum_{d=k+1}^{i}p^i_d(r)(\psi_{d-1,d-k} + \psi_{d-1,k})\\ &= \lambda p_{0,k}(r) + q\lambda_{c,k}(r) + q\frac{\mu_2}{rq + \mu_2}\sum_{j=k+1}^{m}\lambda_{c,j}(r)p^j_{k+1}(r)\\ &+ q\frac{rq}{rq + \mu_2}\sum_{i=k+1}^{m}\lambda_{c,i}(r)\sum_{d=k+1}^{i}p^i_d(r)\psi_{d-1,d-k}.\end{aligned}$$

By rearranging the terms and by using the formula for $p^j_k(r)$ this equals

$$\begin{aligned}&\lambda p_{0,k}(r) + q\lambda_{c,k}(r)p^k_k(r) + q\sum_{j=k+1}^{m}\lambda_{c,j}(r)\left(\frac{\mu_2}{rq + \mu_2}p^j_{k+1}(r) + \frac{rq}{rq + \mu_2}\sum_{d=k+1}^{j}p^j_d(r)\psi_{d-1,d-k}\right)\\ &= \lambda p_{0,k}(r) + q\sum_{j=k}^{m}\lambda_{c,j}(r)p^j_k(r),\end{aligned}$$

which shows (8.28), thus finishing the proof. □

**Theorem 8.8.5.** *The stationary distribution $\pi(r)$ of the QBD Markov chain characterized by $Q(r)$ is the unique fixed point $\zeta$ of the set of ODEs in Equations* (8.12)-(8.13).

*Proof.* Using Proposition 8.8.4 we show that the fixed point equations $\frac{d}{dt}\vec{f}_\ell(t) = 0$ are equivalent to the balance equations of the QBD Markov chain characterized by $Q(r)$. The uniqueness of the fixed point the follows from the uniqueness of the stationary distribution of the Markov chain.

For $\ell \geq 1$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ can be written as

$$0 = \vec{\zeta}_{\ell-1}(\lambda I) + \vec{\zeta}_\ell(\tilde{S}(t) + \lambda I - r\zeta_* I) + \vec{\zeta}_{\ell+1}(\mu\alpha + r\zeta_* V_0),$$

which is exactly the balance equations of $Q(r)$ for $\ell \geq 1$ as $\zeta_* = q$ due to Proposition 8.8.4. This implies that $\vec{\zeta}_\ell = \vec{\zeta}_0 R(r)^\ell$, for all $\ell \geq 1$ for any fixed point.

For $\ell = 0$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ implies

$$0 = \vec{\zeta}_0(\tilde{S}(t) - \lambda I - r\zeta_*(I - V_0)) + \vec{\zeta}_1(\mu\alpha + r\zeta_* V_0) + \lambda\zeta_*\alpha + r\zeta_* \sum_{\ell' \geq 0} \vec{\zeta}_{\ell'} T + r\zeta_* \sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0 \alpha.$$

Due to Proposition 8.8.4 we can rewrite this as

$$0 = \vec{\zeta}_0 B_0(r) + \vec{\zeta}_1 A_{-1}(r) + q\left(\sum_{j=1}^{m} \lambda_{c,j}(r) e_j + \lambda\alpha + r\sum_{\ell \geq 1} \vec{\zeta}_\ell v_0 \alpha\right).$$

This implies that

$$\vec{\zeta}_0 = -q\left(\sum_{j=1}^{m} \lambda_{c,j}(r) e_j + \lambda\alpha + r\sum_{\ell \geq 1} \vec{\zeta}_\ell v_0 \alpha\right)(B_0(r) + \lambda I G(r))^{-1},$$

as $\lambda I G(r) = R(r) A_{-1}(r)$. As $\sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1 - q = \sum_{\ell \geq 0} \pi_\ell(r)\mathbf{1}$, we find that

$$r\sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0 = \lambda_p(r) \tag{8.29}$$

defined in (8.8). This indicates that $\frac{d}{dt}\vec{f}_\ell(t) = 0$ corresponds to the balance equation for $\ell = 0$. As $T\mathbf{1} = \mathbf{1} - v_0$, we have for $\frac{d}{dt} f_*(t) = 0$ that

$$\begin{aligned} 0 &= -\lambda\zeta_* + \vec{\zeta}_0 \mu - r\zeta_*(1 - \zeta_* - \vec{\zeta}_0 v_0) \\ &= -\zeta_*\left(r\sum_{\ell' \geq 0} \vec{\zeta}_{\ell'} T\mathbf{1} + \lambda + r\sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0\right) + \vec{\zeta}_0 \mu, \end{aligned}$$

which is exactly the first balance equation due to Proposition 8.8.4 and (8.29). □

Although this theorem proves that the set of ODEs defined by (8.12) and (8.13) has a unique fixed point, it does not show that the fixed point is a global attractor of these ODEs. One way of proving global attraction is using a monotonicity argument, where one defines a state space for the set of ODEs and a partial relation order on this state space. One then shows that if one state is dominated by another w.r.t. to this partial order at time $t$, then it stays dominated at time $t + s$, for every $s > 0$. This however is not feasible for the whole class of systems in this chapter, as some systems are clearly not monotone (e.g. a system where out of 5 child jobs always 5 get stolen, whereas out of 4 children only one is transferred upon a successful steal attempt).

## 8.9 Conclusions and future work

We introduced a model for randomized work stealing in multithreaded computations in large systems, where parent jobs spawn child jobs and where any number of existing child jobs can be stolen from a queue per probe. We defined a QBD Markov chain that approximates the behaviour of the system when the number of servers tends to infinity. We showed the existence and uniqueness of a stationary distribution for this QBD, provided formulas for the waiting and service times and provided a practical way of calculating expected service times. We have introduced a set of ODEs that captures the behaviour of the system when $N \to \infty$ and we have shown that this set of ODEs has a unique fixed point given by the stationary distribution of the QBD. These are the main technical contributions of the chapter. Using numerical experiments we examined the effect of changing the load $\rho$ and the steal rate $r$. We concluded that the stealing policy where the half of child jobs gets stolen every time is in general a good stealing policy for higher values of $r$, while the strategy of stealing all children performs best for low values of $r$. We concluded further that stealing only one child performs the worst in most of the cases. Finally, using simulation, we validated the model using simulation.

Possible generalizations include stealing multiple parent jobs (up to some finite amount) per probe and systems where offspring of a job can spawn further offspring (multigenerational multithreading). One can also attempt to relax the exponential service time requirements for child and/or parent jobs. This may be challenging as this complicates several aspects of the model such as determining the rates $\lambda_{c,j}(r)$.

Chapter 9

# Analysis of work stealing strategies in large scale multi-threaded computing with jobs of the phase type

*This Chapter contains a generalization of the paper* [44], *that was presented at QEST 2021. The generalized version will be published in* [45]. *This chapter also contains results omitted from* [45] *due to the constraint on the number of pages.*

## 9.1 Introduction

This chapter is closely related to Chapters 7 and 8. Here and in Chapters 7 and 8 we consider a system of homogeneous servers that uses a randomized work stealing policy. Firstly, in Chapter 7 we compared two systems: one system where parent jobs can be stolen and the other system where child jobs can be stolen one at a time. The latter of the two studied systems was novel in the sense that all previous research about work stealing and sharing focused on systems where jobs are considered to be sequential and are always executed as a whole on a single server. The key takeaway from Chapter 7 is that if probe rate $r$ is large enough, then the second system outperforms the first. This is to be expected: for large probe rates a job gets redistributed quickly and more queues can work on it, thus lowering the mean response time. On the other hand, for small probe rates, it is better to transfer parents to empty queues as a larger amount of work is transferred per steal.

Next, in the paper contained in Chapter 8, presented at QEST 2021, we considered a set of policies where if a server with pending child jobs is probed by an idle server, some of its child jobs are transferred. When a server is probed that does not have pending child jobs, a pending parent job is transferred instead (if available). The major complication in the analysis of these policies, compared to Chapter 7, is that when several child jobs get stolen at once, child jobs may be transferred several times before being executed. The objective of Chapter 8 was to gain insights on how to determine the number of child jobs that should be stolen at once. We concluded that the stealing policy where half of the child jobs gets stolen every time is in general a good stealing policy for large probe rates, while stealing all children performs best when the probe rate is low. We also noted that stealing a single child usually performs the worst.

In Chapters 7 and 8 we assumed that parent and child jobs have exponentially distributed service requirements, while jobs in a real system are typically more variable in size. In this chapter, which is an extended version of Chapter 8, we relax this assumption by generalizing the analysis from Chapter 8 to phase-type (PH) distributed parent and child job sizes. As any positive valued distribution can be approximated arbitrarily close by a phase-type distribution (cf. Theorem 2.2.10), this relaxation is significant, without complicating the analysis too much. We show that the insights obtained in Chapter 8 still hold for PH child and parent jobs. Compared to Chapter 8 we also present a mean field model and prove that it has a unique fixed point that coincides with the steady state of a structured Markov chain (a similar result was presented in Chapter 7, where the proof is considerably easier due to the more restricted setting).

The rest of this chapter is organized as follows. In Section 9.2 we describe the system of $N$ queues. The subsequent sections contain the main contributions of the chapter, namely:

- To approximate the performance of the work stealing system of $N$ queues, we introduce in Section 9.3 a Quasi-Birth-Death Markov chain (QBD for short) that describes the evolution of a single server queueing system with negative arrivals. We prove that this QBD has a unique stationary distribution, which can be quickly calculated. In Section 9.4, we indicate how to compute the waiting time distribution and mean service time.
- We compare the performance of several stealing strategies in Section 9.6. We confirm the main insights gained from Chapter 8, namely that the strategy of stealing half of the child jobs performs well for low loads and/or high probe rates and that stealing all child jobs performs best when the load is high and/or the probe rate is low. We further conclude that stealing becomes more worthwhile as the job sizes become more variable (that is, with increased squared coefficient of variation).
- For some strategies we present simulation results in Section 9.7 that suggest that as the number of servers becomes large, the approximation error of the QBD model tends to zero.
- In Section 9.8, we introduce a mean field model and prove that it has a unique fixed point which is given by the stationary distribution of the QBD.

We finish the chapter with Section 9.9, where we present some concluding remarks.

## 9.2 System description and strategies

We consider a system with $N$ homogeneous servers each with an infinite buffer to store jobs. Parent jobs arrive in each server according to a local Poisson arrival process with rate $\lambda$. Upon entering service a parent job spawns $i \in \{0, 1, \ldots, m\}$, $m \geq 1$, child jobs, the number of which follows a general distribution with finite support $p_i$ (i.e., $p_i \geq 0$ for every $i$ and $\sum_{i=0}^{m} p_i = 1$). These child jobs are stored locally and have priority over any parent jobs (either already present or yet to arrive), while the spawning parent job continues service. Thus, when a (parent or child) job completes service the server first checks to see whether it has any waiting child jobs, if so it starts service on a child job. If there are no child jobs present, service on a waiting parent job starts (if any are present). We assume that parent and child jobs have phase type (PH) distributed service requirements with representations $(\alpha^p, S^p)$ and $(\alpha^c, S^c)$ respectively, meaning the probability that the service requirement of a child job exceeds $t$ is given by $\alpha^c e^{S^c t}\mathbf{1}$, where $\mathbf{1}$ denotes a column vector of ones of the correct size, and the same holds for the parent jobs if we replace the superscript $c$ by $p$. PH distributions are distributions with a modulating finite state Markov chain (see also [48]). Moreover there are various fitting tools available for PH distributions (see e.g. [46,62]).

When a server is idle, it probes other servers at random at rate $r > 0$, where $r$ is a system parameter. Note that $r$ determines the amount of communication between the servers and increasing $r$ should improve performance at the expense of a higher communication overhead. When a server is probed (by an idle server) and it has waiting (parent or child) jobs, we state that the probe is successful. When a successful probe reaches a server without waiting child jobs, a parent job is transferred to the idle server. Note that such a transferred parent job starts service and spawns its child jobs at the new server.

When a successful probe reaches a server with pending/waiting child jobs, several child jobs can be transferred at once. If the probed server is serving a parent job and there are $i$ child jobs in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\phi_{i,j}$ (i.e., for every $i$ we have $\sum_{j=1}^{i} \phi_{i,j} = 1$). On the other hand if a child job is being processed by the probed server and there are $i$ child jobs waiting in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\psi_{i,j}$ (i.e., for every $i$ we have $\sum_{j=1}^{i} \psi_{i,j} = 1$). For ease of notation we set $\phi_{i,j} = \psi_{i,j} = 0$ if $j > i$ and further $\phi_{i,j} = \psi_{i,j} = 0$ if $i$ or $j$ is 0. Probes and job transfers are assumed to be instantaneous.

The main objective of this chapter is to study how the probabilities $\phi_{i,j}$ and $\psi_{i,j}$ influence the response time of a job, where the response time is defined as the time between the arrival of a parent job and the completion of the parent and all its spawned child jobs. Given the above description, it is clear that we get a Markov process if we keep track of the number of parent and child job and the phase of the job in service in each of the $N$ servers. This Markov process however does not appear to have a product form, making its analysis prohibitive.

Instead we use an approximation method, the accuracy of which is investigated in Section 9.7. The idea of the approximation exists in focusing on a single server and assuming that the queue lengths at any other server are independent and identically distributed as in this particular server. Within the context of load balancing, this approach is known as the cavity method (see [9] or Section 4.3). In fact all the analytical models used in [16, 25, 55–57, 68, 69, 77, 79] can be regarded as cavity method approximations. A common feature of such an approximation is that it tends to become more accurate as the number of servers tends to infinity, as we demonstrate in Section 9.7 for our model. The cavity method typically involves iterating the so-called cavity map (see [9] or Section 4.3). However, in our case the need for such an iteration is avoided by deriving expressions for the rates at which child and parent jobs are stolen.

## 9.3 Quasi-Birth-Death Markov chain

As the system of $N$ queues uses work stealing, where (parts of) a job can be transferred upon a successful steal, these $N$ queues are coupled in a non-trivial way. However, due to homogeneity of the processors we are still able to analyze the system of $N$ queues by approximating it by a single queue with negative arrivals. This queue can described by a Quasi-Birth-Death (QBD) Markov chain, which we introduce in this section.

Let $\lambda_p(r)$ denote the rate at which parent jobs are stolen when the server is idle. Let $\lambda_{c,1}(r), \dots, \lambda_{c,m}(r)$ denote respectively the rates at which $1, \dots, m$ child jobs are stolen. We provide formulas for these rates further on. The evolution of a single server has the following characteristics, where the negative arrivals correspond to steal events:

1. When the server is busy, arrivals of parent jobs occur according to a Poisson process with rate $\lambda$. When the server is idle, parent jobs arrive at the rate $\lambda + \lambda_p(r)$, while a batch of $i$ child jobs arrives at rate $\lambda_{c,i}(r)$ for $i = 1, \dots, m$.

2. Upon entering service, a parent job spawns $i \in \{0, 1, \dots, m\}, m \geq 1$, child jobs with probability $p_i$. Child jobs are stored locally.

3. Child jobs have priority over any parent jobs *waiting* in the queue and are thus executed immediately after their parent job when executed on the same server.

4. Parent and child jobs have PH distributed service requirements with $n_p$ and $n_c$ phases and with representations $(\alpha^p, S^p)$ and $(\alpha^c, S^c)$, respectively. We assume that these representations have $\alpha^c\mathbf{1} = 1$ and $\alpha^p\mathbf{1} = 1$. We denote $s^p = -S^p\mathbf{1}$ and $s^c = -S^c\mathbf{1}$.

5. If there are parent jobs and no child jobs waiting in the buffer of the server then a negative parent arrival occurs at the rate $rq$, where $q = 1 - \rho$ is the probability that a queue is idle (where $\rho$ is defined in (9.1)).

6. If a parent job is in service and there are $i \in \{1, \dots, m\}$ child jobs in the buffer of the server, a batch of $j$ negative child job arrivals occurs at the rate $rq\phi_{i,j}$, for all $j \in \{1, \dots, i\}$.

7. If a child job is in service and there are $i \in \{1, \dots, m-1\}$ child jobs pending in the buffer of the server, a batch of $j$ negative child job arrivals occurs at the rate $rq\psi_{i,j}$, for all $j \in \{1, \dots, i\}$.

Note that the load of the system can be expressed as

$$\rho = \lambda \left( \alpha^p(-S^p)^{-1}\mathbf{1} + \alpha^c(-S^c)^{-1}\mathbf{1} \sum_{n=1}^{m} np_n \right), \tag{9.1}$$

where $\alpha^p(-S^p)^{-1}\mathbf{1}$ and $\alpha^c(-S^c)^{-1}\mathbf{1}$ is the mean parent and child job size, respectively. Denote by $X \geq 0$ the number of parent jobs waiting, by $Y \in \{0, 1, \dots, m\}$ the number of child jobs in the server (either in service or waiting), by $Z \in \{0, 1\}$ whether a parent job is currently in service ($Z = 1$) or not ($Z = 0$) and by $W$ the phase of the job in service. Note that we have $W \in \{1, \dots, n_p\}$ when $Z = 1$ and $W \in \{0, \dots, n_c\}$ when $Z = 0$, where

Table 9.1: Transitions for the QBD in Section 9.3

| | From | | To | Rate | For |
|---|---|---|---|---|---|
| 1. | $(0,0,0,0)$ | $\rightarrow$ | $(0,j,0,k)$ | $\lambda_{c,j}(r)\alpha_k^c$ | $j=1,\dots,m,\ k=1,\dots,n_c$ |
| 2. | $(0,0,0,0)$ | $\rightarrow$ | $(0,j,1,k)$ | $(\lambda+\lambda_p(r))p_j\alpha_k^p$ | $j=0,1,\dots,m,\ k=1,\dots,n_p,$ |
| 3. | $(X,Y,Z,W)$ | $\rightarrow$ | $(X+1,Y,Z,W)$ | $\lambda$ | $X+Y+Z\geq 1, W\geq 1$ |
| 4. | $(X,Y,1,k)$ | $\rightarrow$ | $(X,Y,0,\ell)$ | $s_k^p\alpha_\ell^c$ | $X\geq 0, Y\geq 1, k=1,\dots,n_p, \ell=1,\dots,n_c,$ |
| 5. | $(X,Y,0,k)$ | $\rightarrow$ | $(X,Y-1,0,\ell)$ | $s_k^c\alpha_\ell^c$ | $X\geq 0, Y\geq 2, k,\ell=1,\dots,n_c,$ |
| 6. | $(0,0,1,k)$ | $\rightarrow$ | $(0,0,0,0)$ | $s_k^p$ | $k=1,\dots,n_p,$ |
| 7. | $(0,1,0,k)$ | $\rightarrow$ | $(0,0,0,0)$ | $s_k^c$ | $k=1,\dots,n_c,$ |
| 8. | $(X,0,1,k)$ | $\rightarrow$ | $(X-1,j,1,\ell)$ | $s_k^p p_j\alpha_\ell^p$ | $X\geq 1, j=0,1,\dots,m, k,\ell=1,\dots,n_p,$ |
| 9. | $(X,1,0,k)$ | $\rightarrow$ | $(X-1,j,1,\ell)$ | $s_k^c p_j\alpha_\ell^p$ | $X\geq 1, j=1,\dots,m, k=1,\dots,n_c, \ell=1,\dots,n_p,$ |
| 10. | $(X,Y,1,k)$ | $\rightarrow$ | $(X,Y,1,\ell)$ | $S_{k,\ell}^p$ | $X,Y\geq 0, k,\ell=1,\dots,n_p,$ with $k\neq\ell,$ |
| 11. | $(X,Y,0,k)$ | $\rightarrow$ | $(X,Y,0,\ell)$ | $S_{k,\ell}^c$ | $X\geq 0, Y\geq 1, k,\ell=1,\dots,n_c,$ with $k\neq\ell,$ |
| 12. | $(X,Y,Z,W)$ | $\rightarrow$ | $(X-1,Y,Z,W)$ | $rq$ | $X,W\geq 1, Y+Z=1,$ |
| 13. | $(X,Y,1,W)$ | $\rightarrow$ | $(X,Y-j,1,W)$ | $rq\phi_{Y,j}$ | $X\geq 0, Y\geq j, j=1,\dots,m, W=1,\dots,n_p,$ |
| 14. | $(X,Y,0,W)$ | $\rightarrow$ | $(X,Y-j,0,W)$ | $rq\psi_{Y-1,j}$ | $X\geq 0, Y\geq j+1, j=1,\dots,m-1, W=1,\dots,n_c.$ |

$W=0$ if the queue is idle. The possible transitions of the QBD Markov chain are listed in Table 9.1, corresponding to: 1. a batch of $j$ child jobs arriving at an idle queue and the first child job proceeding directly into service, 2. a parent job arriving at an idle queue and proceeding directly into service, spawning $j$ child jobs, 3. a parent arriving to a non-idle queue, 4. completion of a parent in service, succeeded by a child job, 5. child service completion, succeeded by another child job, 6. completion of a parent in service, not succeeded by any job, 7. child service completion, not succeeded by another job, 8. parent service completion, succeeded by a parent job that enters service and spawns $j$ child jobs, 9. child service completion, succeeded by a parent job that enters service and spawns $j$ child jobs, 10. a phase change occurs in the service of a parent, 11. a phase change occurs in the service of a child, 12. negative parent job arrival, 13. a parent is in service and a batch of negative child job arrivals occurs, 14. a child job is in service and a batch of negative child job arrivals occurs.

The four dimensional process $\{X_t(r), Y_t(r), Z_t(r), W_t(r) : t\geq 0\}$ is an irreducible, aperiodic Quasi-Birth-Death process, where the *level* $\ell = *$ when the chain is in state $(0,0,0,0)$ and equals $\ell\geq 0$ when the chain is in a state with $X=\ell$ (different from $(0,0,0,0)$). When the level $\ell\geq 0$, the *phase* of the QBD is three dimensional and given by $(Y,Z,W)$. The $mn_c+(m+1)n_p$ phases of level $\ell\geq 0$ are ordered such that the $j$-th phase corresponds to $(Y,Z,W)=(\lceil j/n_c\rceil, 0, (j-1) \mod n_c+1)$, for $j=1,\dots,mn_c$ and phase $mn_c+j$ to $(Y,Z,W)=(\lceil j/n_p\rceil-1, 1, (j-1) \mod n_p+1)$ for $j=1,\dots,(m+1)n_p$.

As explained below, the generator of the process is

$$Q(r)=\begin{bmatrix} -\lambda_0(r) & \sum_{j=1}^m \lambda_{c,j}(r)\kappa_j+(\lambda+\lambda_p(r))\alpha & & & \\ \mu & B_0(r) & A_1 & & \\ & A_{-1}(r) & A_0(r) & A_1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix},$$

with $\lambda_0(r) = \sum_{j=1}^{m} \lambda_{c,j}(r) + \lambda + \lambda_p(r)$. The row vector $\kappa_j$ is defined as

$$\kappa_j = \left[ 0'_{(j-1)n_c} \; \alpha^c \; 0'_{(m-j)n_c+(m+1)n_p} \right],$$

where $0_i$ is a column vector of zeroes of length $i$. The initial probability vector $\alpha$ records the distribution of child jobs upon a parent job entering service and the initial phase of that parent. It is given by $\alpha = \left[ 0'_{mn_c} \quad p_0\alpha^p \quad p_1\alpha^p \quad \dots \quad p_m\alpha^p \right]$. Indeed, at rate $\lambda_{c,j}(r)\alpha^c_k$ a batch of $j$ child jobs arrives in an idle server, causing a jump to level 1 and phase $(j, 0, k)$, while at rate $(\lambda + \lambda_p(r))\alpha^p_k$ a parent job arrives that spawns $j$ child jobs with probability $p_j$ causing a jump to phase $(j, 1, k)$ of level 1.

For further use, define

$$S(r) = \begin{bmatrix} S_{00}(r) & 0 \\ S_{10} & S_{11}(r) \end{bmatrix},$$

where $S_{00}(r)$ is an $mn_c \times mn_c$ matrix and $S_{11}(r)$ is an $(m+1)n_p \times (m+1)n_p$ matrix,

$$S_{00}(r) = rq \begin{bmatrix} & & \\ \psi_{1,1} & & \\ \vdots & \ddots & \\ \psi_{m-1,m-1} & \cdots & \psi_{m-1,1} & \end{bmatrix} \otimes I_{n_c} + \begin{bmatrix} S^c & & & \\ s^c\alpha^c & \ddots & & \\ & \ddots & \ddots & \\ & & s^c\alpha^c & S^c \end{bmatrix},$$

$$S_{10} = \begin{bmatrix} 0_{n_p} & \cdots & \\ s^p\alpha^c & & \\ & s^p\alpha^c & \\ & & \ddots \end{bmatrix}, \; S_{11}(r) = rq \begin{bmatrix} & & \\ \phi_{1,1} & & \\ \vdots & \ddots & \\ \phi_{m,m} & \cdots & \phi_{m,1} & \end{bmatrix} \otimes I_{n_p} + \begin{bmatrix} S^p & & \\ & \ddots & \\ & & \ddots & \\ & & & S^p \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product and $I_k$ denotes the identity matrix of size $k \times k$. The matrix $A_0(r)$ contains the possible transitions for which the level $\ell > 0$ remains unchanged, this is when child jobs are stolen, or when a waiting child moves into service, or when phase of the job in service changes. Hence

$$A_0(r) = S(r) - \lambda I - rqI.$$

Here, $I = I_{mn_c+(m+1)n_p}$. Whenever it is clear what dimensions an identity matrix should have, we simply write $I$ for the identity matrix of the appropriate size. Note that even when there are no child jobs waiting, the rate $rq$ appears on the main diagonal of $A_0(r)$ due to the negative parent arrivals. When $\ell = 0$ there are no parent jobs waiting and therefore the negative parent arrivals that occur in phases $(1, 0, k)$ and $(m+1, 1, k')$, for $k = 1, \dots, n_c$ and $k' = 1, \dots, n_p$, have no impact. This implies that

$$B_0(r) = A_0(r) + rqV_0 = S(r) - \lambda I - rq(I - V_0),$$

where $V_0 = \text{diag}\left(\left[ 1'_{n_c} \quad 0'_{(m-1)n_c} \quad 1'_{n_p} \quad 0'_{mn_p} \right]\right)$, with $1_i$ a column vector of ones of size $i$. The level $\ell$ can only decrease by one due to a service completion from a phase with no pending child jobs, that is, from phases $(1, 0, k)$ and $(m+1, 1, k')$, for $k = 1, \dots, n_c$ and $k' = 1, \dots, n_p$. To capture these events define $\mu = \left[ (s^c)' \quad 0'_{(m-1)n_c} \quad (s^p)' \quad 0'_{mn_p} \right]'$. The level can also decrease due to a negative parent arrival when $\ell > 0$. The matrix $A_{-1}(r)$ records the transitions for which the level decreases and therefore equals

$$A_{-1}(r) = \mu\alpha + rqV_0.$$

Finally, parent job arrivals always increase the level by one:

$$A_1 = \lambda I.$$

Denote by $A(r) = A_{-1}(r) + A_0(r) + A_1$, the generator of the phase process, then

$$A(r) = S(r) + \mu\alpha - rq(I - V_0).$$

Define

$$\pi_*(r) = \lim_{t\to\infty} P[X_t(r) = 0, Y_t(r) = 0, Z_t(r) = 0, W_t(r) = 0],$$

and for $\ell \geq 0$,

$$\pi_\ell(r) = (\pi_{\ell,1,0}(r), \ldots \pi_{\ell,m,0}(r), \pi_{\ell,0,1}(r), \ldots, \pi_{\ell,m,1}(r)),$$

where

$$\pi_{\ell,j,0}(r) = (\pi_{\ell,j,0,1}(r), \ldots, \pi_{\ell,j,0,n_c}(r))$$
$$\pi_{\ell,j,1}(r) = (\pi_{\ell,j,1,1}(r), \ldots, \pi_{\ell,j,1,n_p}(r))$$

and where

$$\pi_{\ell,j,k,w}(r) = \lim_{t\to\infty} P[X_t(r) = \ell, Y_t(r) = j, Z_t(r) = k, W_t(r) = w].$$

Due to the QBD structure [60], we have

$$\pi_0(r) = \pi_*(r)R_0(r), \tag{9.2}$$

where $R_0(r)$ is a row vector of size $mn_c + (m+1)n_p$ and for $\ell \geq 1$,

$$\pi_\ell(r) = \pi_0(r)R(r)^\ell, \tag{9.3}$$

where $R(r)$ is a $(mn_c + (m+1)n_p) \times (mn_c + (m+1)n_p)$ matrix and by [48] the smallest nonnegative solution to

$$A_1 + R(r)A_0(r) + R(r)^2A_{-1}(r) = 0.$$

Also, due to the balance equations with $\ell = 0$, we have

$$\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + (\lambda + \lambda_p(r))\alpha + R_0(r)B_0(r) + R_0(r)R(r)A_{-1}(r) = 0$$

and due to [48, Chapter 6]

$$A_1G(r) = R(r)A_{-1}(r),$$

where $G(r)$ is the smallest nonnegative solution to

$$A_{-1}(r) + A_0(r)G(r) + A_1G(r)^2 = 0.$$

Combining the above yields the following expression:

$$R_0(r) = -\Big(\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + (\lambda + \lambda_p(r))\alpha\Big)(B_0(r) + \lambda I G(r))^{-1}, \tag{9.4}$$

where $B_0(r)+\lambda I G(r)$ is a subgenerator matrix and is therefore invertible. We note that $R(r)$ and $G(r)$ are independent of $\lambda_{c,1}(r), \dots, \lambda_{c,m}(r)$ and $\lambda_p(r)$ and can be computed easily using the toolbox presented in [5]. To fully characterize the QBD in terms of $\lambda, \alpha^c, S^c, \alpha^p, S^p$ and the probabilities $p_i, \phi_{i,j}$ and $\psi_{i,j}$, we need to specify $\lambda_{c,1}(r), \dots, \lambda_{c,m}(r)$ and $\lambda_p(r)$.

To determine these rates we use the following observation: $q = 1 - \rho$ should be the probability that the QBD is in state $(0,0,0,0)$ and in this state batches of $j$ child jobs arrive at rate $\lambda_{c,j}(r)$. Therefore $q\lambda_{c,j}(r)$ should equal the parent arrival rate $\lambda$ times the expected number of times that a batch of $j$ child jobs is stolen per parent job. The main difficulty in using this equality lies in the fact that we must also take into account that a child job can be stolen several times before it is executed.

To this end and as a preparation for Proposition 9.3.3, we define recursively the row vector $p_{i,j}(r)$ such that the $k$-th entry of $p_{i,j}(r)$ is the probability that the phase $(i, j, k)$ is visited by the queue during the service of a job just after an arrival, a steal or a completion.
By conditioning on whether we first have a service completion or steal event, we have

$$\begin{aligned} p_{1,m}(r) &= p_m \alpha^p \\ p_{1,i}(r) &= p_i \alpha^p + rq \sum_{j>i} \phi_{j,j-i} p_{1,j}(r)(rqI - S^p)^{-1}, \end{aligned}$$

for $i \in \{0, \dots, m-1\}$. For $i \in \{1, \dots, m\}$, with $p_{0,m+1} = 0$, we further have

$$\begin{aligned} p_{0,i}(r) &= p_{1,i}(r)(rqI - S^p)^{-1} s^p \alpha^c + p_{0,i+1}(r)(rqI - S^c)^{-1} s^c \alpha^c \\ &\quad + rq \sum_{j>i} \psi_{j-1,j-i} p_{0,j}(r)(rqI - S^c)^{-1}. \end{aligned}$$

Note that

$$p_{1,0}(r)\mathbf{1} + p_{0,1}(r)\mathbf{1} = 1, \tag{9.5}$$

as every queue eventually visits phase $(0, 1, k)$ or $(1, 0, k)$ for some $k$ just after a completion or a steal.
We also define the row vector $p_i^j(r)$ recursively, where $k$-th entry is the probability that the queue visits phase $(0, i, k)$ just after a completion or a steal has occurred, given that the queue started with $j$ child jobs. We have

$$\begin{aligned} p_j^j(r) &= \alpha^c, \\ p_i^j(r) &= p_{i+1}^j(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{k=i+1}^{j} \psi_{k-1,k-i} p_k^j(r)(rqI - S^c)^{-1}, \end{aligned}$$

for $i \in \{1, \dots, j-1\}$. Note that we have $p_1^j(r)\mathbf{1} = 1$, for $1 \le j \le m$, as the QBD eventually visits phase $(0, 1, k)$ for some $k$ just after a completion or a steal. We are now in a position to define $\lambda_{c,i}(r)$ recursively as:

$$\lambda_{c,m}(r) = \frac{\lambda}{q} rq \phi_{m,m} p_{1,m}(r)(rqI - S^p)^{-1}\mathbf{1}$$

$$\lambda_{c,i}(r) = \frac{\lambda}{q} rq \sum_{j \geq i} \phi_{j,i} p_{1,j}(r)(rqI - S^p)^{-1}\mathbf{1} + \frac{\lambda}{q} rq \sum_{j>i} \psi_{j-1,i} p_{0,j}(r)(rqI - S^c)^{-1}\mathbf{1}$$
$$+ rq \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} \psi_{k-1,i} p_k^j(r)(rqI - S^c)^{-1}\mathbf{1} \tag{9.6}$$

for $i \in \{1, \ldots, m-1\}$. Note that $rq \sum_{j=1}^{n_p} [(rqI - S^p)^{-1}]_{i,j}$ is the probability that a steal happens before the completion of the parent, given that the parent started service in phase $i$. It then follows that $rq\phi_{m,m} p_{1,m}(r)(rqI - S^p)^{-1}\mathbf{1}$ indeed equals the expected number of batches of size $m$ that are stolen per parent job (as the job must spawn $m$ child jobs and these must be stolen as a batch before the parent completes service). For $i < m$, the first two sums of (9.6) represent the expected number of size $i$ batches that are stolen from the original server, while the double sum counts the expected number of such steals that occur on a server different from the original server.

Note, that if parent and child jobs have acyclic phase type[1] job requirements, we can avoid taking inverses of matrices when computing $\lambda_{c,1}(r), \ldots, \lambda_{c,m}(r)$. We illustrate this through two examples.
Let $p_{i,j,k}(r)$ denote the probability that the QBD visits phase $(i, j, k)$ during the service of a job. Let further $p_{i,\ell}^{j,k}(r)$, for $1 \leq i \leq j \leq m$ and $k, \ell = 1, \ldots, n_c$, denote the probability that the QBD visits phase $(0, i, \ell)$ given that it is in the phase $(0, j, k)$ before a job completes service.

**Example 9.3.1** (Hyperexponential job requirements). Suppose parent and child jobs have PH$(\alpha^p, -diag([\mu_1^p, \ldots, \mu_{n_p}^p]))$ and PH$(\alpha^c, -diag([\mu_1^c, \ldots, \mu_{n_c}^c]))$ distributed job requirements respectively, i.e. the parent and child job sizes are hyperexponential. We then have

$$p_{1,m,j}(r) = p_m \alpha_j^p,$$
$$p_{1,i,j}(r) = p_i \alpha_j^p + \frac{rq}{rq + \mu_j^p} \sum_{k>i} p_{1,k,j}(r)\phi_{k,k-i}$$

for $i \in \{0, \ldots, m-1\}$ and

$$p_{0,i,j}(r) = \left( \sum_{k=1}^{n_p} \frac{\mu_k^p}{rq + \mu_k^p} p_{1,i,k}(r) + \sum_{k=1}^{n_c} \frac{\mu_k^c}{rq + \mu_k^c} p_{0,i+1,k}(r) \right) \alpha_j^c$$
$$+ \frac{rq}{rq + \mu_j^c} \sum_{k>i} p_{0,k,j}(r)\psi_{k-1,k-i},$$

for $i \in \{1, \ldots, m\}$, with $p_{0,m+1,j} = 0$. Note that (9.5) can be written as

$$\sum_{j=1}^{n_p} p_{1,0,j}(r) + \sum_{j=1}^{n_c} p_{0,1,j}(r) = 1.$$

We further have

$$p_{j,\ell}^{j,k}(r) = 1[k = \ell],$$

[1]A phase type distribution characterized by $(\alpha, S)$ is acyclic if the rows and columns of $S$ can be permuted to make $S$ upper triangular.

$$p_{i,k'}^{j,k}(r) = \alpha_{k'}^c \sum_{\ell=1}^{n_c} \frac{\mu_\ell^c}{rq+\mu_\ell^c} p_{i+1,\ell}^{j,k}(r) + \frac{rq}{rq+\mu_{k'}^c} \sum_{\ell=i+1}^{j} \psi_{\ell-1,\ell-i} p_{\ell,k'}^{j,k}(r).$$

Note that we have $\sum_{\ell=1}^{n_c} p_{1,\ell}^{j,k}(r) = 1$. We can now calculate $\lambda_{c,i}(r)$ by using the following recursive formula:

$$\lambda_{c,m}(r) = \frac{\lambda}{q}\phi_{m,m} \sum_{k=1}^{n_p} \frac{rq}{rq+\mu_k^p} p_{1,m,k}(r)$$

$$\lambda_{c,i}(r) = \frac{\lambda}{q}\sum_{j\geq i}\phi_{j,i} \sum_{k=1}^{n_p} \frac{rq}{rq+\mu_k^p} p_{1,j,k}(r) + \frac{\lambda}{q}\sum_{j>i}\psi_{j-1,i}\sum_{k=1}^{n_c}\frac{rq}{rq+\mu_k^c}p_{0,j,k}(r)$$
$$+ \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k',\tilde{k}=1}^{n_c} \alpha_{k'}^c \sum_{k=i+1}^{j} \psi_{k-1,i} \frac{rq}{rq+\mu_{\tilde{k}}^c} p_{k,\tilde{k}}^{j,k'}(r).$$

**Example 9.3.2** (Erlang job requirements). Suppose parents and children have Erlang distributed job requirements with $n_c$ and $n_p$ phases respectively and with

$$S^p = \begin{bmatrix} -\mu_1 & \mu_1 & & \\ & \ddots & \ddots & \\ & & -\mu_1 & \mu_1 \\ & & & -\mu_1 \end{bmatrix} \text{ and } S^c = \begin{bmatrix} -\mu_2 & \mu_2 & & \\ & \ddots & \ddots & \\ & & -\mu_2 & \mu_2 \\ & & & -\mu_2 \end{bmatrix}$$

for some $\mu_1, \mu_2 > 0$. For ease of notation, set $p_{i,j,0}(r) = 0$. We have the following recursion

$$p_{1,i,k}(r) = 1[k=1]p_i + \frac{\mu_1}{rq+\mu_1} p_{1,i,k-1}(r) + \frac{rq}{rq+\mu_1} \sum_{j>i} \phi_{j,j-i} p_{1,j,k}(r),$$

for $i = 1, \dots, m$ and $k = 1, \dots, n_p$. We further have

$$p_{0,i,k}(r) = 1[k=1]\frac{\mu_1}{rq+\mu_1} p_{1,i,n_p}(r) + 1[k=1]\frac{\mu_2}{rq+\mu_2} p_{0,i+1,n_c}$$
$$+ \frac{\mu_2}{rq+\mu_2} p_{0,i,k-1} + \frac{rq}{rq+\mu_2} \sum_{j>i} \psi_{j-1,j-i} p_{0,j,k}(r)$$

for $i = 1, \dots, m$ and $k = 1, \dots, n_c$. When there are no pending children in a queue, we can only have a phase change or a completion of the job in service. We therefore have:

$$p_{1,0,k}(r) = 1[k=1]p_0 + p_{1,0,k-1}(r) + \frac{rq}{rq+\mu_1} \sum_{j=1}^{m} \phi_{j,j} p_{1,j,k}(r),$$

$$p_{0,1,k}(r) = 1[k=1]\frac{\mu_1}{rq+\mu_1} p_{1,1,n_p}(r) + 1[k=1]\frac{\mu_2}{rq+\mu_2} p_{0,2,n_c}$$
$$+ p_{0,1,k-1} + \frac{rq}{rq+\mu_2} \sum_{j>1} \psi_{j-1,j-1} p_{0,j,k}(r).$$

In case of Erlang job requirements, (9.5) becomes

$$p_{1,0,n_p}(r) + p_{0,1,n_c}(r) = 1.$$

For ease of notation set $p^{j,k}_{i,0}(r) = p^{j,0}_{i,k}(r) = 0$. Then

$$p^{j,k}_{j,k}(r) = 1$$
$$p^{j,k}_{i,\tilde{k}}(r) = 1[\tilde{k} = 1]\frac{\mu_2}{rq + \mu_2} p^{j,k}_{i+1,n_c}(r) + \frac{\mu_2}{rq + \mu_2} p^{j,k}_{i,\tilde{k}-1}(r) + \frac{rq}{rq + \mu_2}\sum_{\ell=i+1}^{j} \psi_{\ell-1,\ell-i} p^{j,k}_{\ell,\tilde{k}}(r),$$

for $j = 2, \dots, m$, $i = 2, \dots, j$ and $k, \tilde{k} = 1, \dots, n_c$. We further have

$$p^{1,k}_{1,\ell}(r) = 1[k \leq \ell]$$
$$p^{j,k}_{1,\tilde{k}}(r) = 1[\tilde{k} = 1]\frac{\mu_2}{rq + \mu_2} p^{j,k}_{2,n_c}(r) + p^{j,k}_{1,\tilde{k}-1}(r) + \frac{rq}{rq + \mu_2}\sum_{\ell=2}^{j} \psi_{\ell-1,\ell-1} p^{j,k}_{\ell,\tilde{k}}(r).$$

Note that $p^{j,k}_{1,n_c}(r) = 1$. We now get the following recursive formula for $\lambda_{c,i}(r)$:

$$\lambda_{c,m}(r) = \frac{\lambda}{q}\frac{rq}{rq + \mu_1}\phi_{m,m}\sum_{k=1}^{n_p} p_{1,m,k}(r)$$
$$\lambda_{c,i}(r) = \frac{\lambda}{q}\frac{rq}{rq + \mu_1}\sum_{j \geq i}\phi_{j,i}\sum_{k=1}^{n_p} p_{1,j,k}(r) + \frac{\lambda}{q}\frac{rq}{rq + \mu_2}\sum_{j > i}\psi_{j-1,i}\sum_{k=1}^{n_c} p_{0,j,k}(r)$$
$$+ \frac{rq}{rq + \mu_2}\sum_{j=i+1}^{m}\lambda_{c,j}(r)\sum_{k=i+1}^{j}\psi_{k-1,i}\sum_{\tilde{k}=1}^{n_c} p^{j,1}_{k,\tilde{k}}(r).$$

We now continue with the description of the QBD. We still need to define $\lambda_p(r)$, for this we demand that $\pi_*(r) = q$ and that

$$\pi_*(r) + \sum_{\ell \geq 0}\pi_\ell(r)\mathbf{1} = 1.$$

Then from equations (9.2) and (9.3),

$$q\left(1 + R_0(r)(I - R(r))^{-1}\mathbf{1}\right) = 1, \tag{9.7}$$

where the inverse of $I - R(r)$ exists due to Proposition 9.3.3. Using (9.4) and (9.7) we get:

$$\lambda_p(r) = \frac{(1 - q) - q(\sum_{j=1}^{m}\lambda_{c,j}(r)\kappa_j + \lambda\alpha)w}{q\alpha w}, \tag{9.8}$$

with $w = -(B_0(r) + \lambda I G(r))^{-1}(I - R(r))^{-1}\mathbf{1}$. Note that $\lambda_p(r)$ is well-defined for $q > 0$, i.e. $\rho < 1$. This completes the description of the QBD Markov chain.

**Proposition 9.3.3.** *The QBD process $\{X_t(r), Y_t(r), Z_t(r), W_t(r) : t \geq 0\}$ has a unique stationary distribution for any $r \geq 0$ if $\rho < 1$.*

*Proof.* The positive recurrence of the QBD process only depends on the matrices $A_{-1}(r)$, $A_0(r)$ and $A_1$ [60]. These three matrices are the same three matrices as those of the QBD characterizing the M/MAP/1 queue where the MAP service process is characterized by $(S_0(r), S_1(r))$ with $S_0(r) = S(r) - rqI$ and $S_1(r) = \mu\alpha + rqV_0$. As such the QBD process is positive recurrent if and only if the arrival rate $\lambda$ is less than the service completion intensity of the MAP $(S_0(r), S_1(r))$. This intensity equals $\theta^{(r)}S_1(r)\mathbf{1}/\theta^{(r)}\mathbf{1}$, where the vector $\theta^{(r)}$ is such that $\theta^{(r)}(S_0(r) + S_1(r)) = 0$.

We note that $S_0(r) + S_1(r) = A_{-1}(r) + A_0(r) + A_1 = A(r)$ and define

$$\theta^{(r)}_{(0,1)} = p_{0,1}(r)(-S^c)^{-1}, \quad \theta^{(r)}_{(0,i')} = p_{0,i'}(r)(rqI - S^c)^{-1},$$
$$\theta^{(r)}_{(1,0)} = p_{1,0}(r)(-S^p)^{-1}, \quad \theta^{(r)}_{(1,i)} = p_{1,i}(r)(rqI - S^p)^{-1},$$

for $i' = 2, \dots, m$ and for $i = 1, \dots, m$. Define $v^{(r)} = \theta^{(r)}A(r)$. Then

$$\begin{aligned} v^{(r)}_{(0,i')} &= -p_{0,i'}(r) + p_{1,i'}(r)(rqI - S^p)^{-1}s^p\alpha^c + p_{0,i'+1}(r)(rqI - S^c)^{-1}s^c\alpha^c \\ &\quad + rq\sum_{j>i'}\psi_{j-1,j-i'}p_{0,j}(r)(rqI - S^c)^{-1} = 0, \end{aligned}$$

for $i' = 1, \dots, m$. By using (9.5), we further get

$$\begin{aligned} v^{(r)}_{(1,i)} &= -p_{1,i}(r) + p_i(p_{0,1}(r)(-S^c)^{-1}s^c + p_{1,0}(r)(-S^p)^{-1}s^p)\alpha^p \\ &\quad + rq\sum_{j>i}\phi_{j,j-i}p_{1,j}(r)(rqI - S^p)^{-1} \\ &= -p_{1,i}(r) + p_i\alpha^p + rq\sum_{j>i}\phi_{j,j-i}p_{1,j}(r)(rqI - S^p)^{-1} = 0, \end{aligned}$$

for $i = 0, \dots, m$. Hence $\theta^{(r)}A(r) = \theta^{(r)}(S_0(r) + S_1(r)) = 0$. As

$$\begin{aligned} &\frac{\theta^{(r)}S_1(r)\mathbf{1}}{\theta^{(r)}\mathbf{1}} \\ &= \frac{1}{\theta^{(r)}\mathbf{1}}\left(p_{0,1}(r)(-S^c)^{-1}s^c + p_{1,0}(r)(-S^p)^{-1}s^p + rqp_{0,1}(r)(-S^c)^{-1}\mathbf{1} + rqp_{1,0}(r)(-S^p)^{-1}\mathbf{1}\right) \\ &\geq \frac{1}{\theta^{(r)}\mathbf{1}}(p_{0,1}(r)(-S^c)^{-1}s^c + p_{1,0}(r)(-S^p)^{-1}s^p) = \frac{1}{\theta^{(r)}\mathbf{1}}, \end{aligned}$$

it suffices that $\lambda < 1/\theta^{(r)}\mathbf{1}$ for the chain to be positive recurrent. For $r = 0$ we have $p_{1,i}(r) = p_i\alpha^p$ and $p_{0,i'}(r) = \sum_{j\geq i'} p_j\alpha^c$, which implies that $\theta^{(0)}\mathbf{1} = \rho/\lambda$. Therefore $\lambda < 1/\theta^{(0)}\mathbf{1}$ is equivalent to demanding that $\rho < 1$. As $\theta^{(r)}\mathbf{1}$ is the mean time between two service completions of the MAP process where the state is reset according to the vector $\alpha$, we have that $\theta^{(r)}\mathbf{1}$ decreases in $r$. This completes the proof as $\rho < 1$ implies that $\lambda < 1/\theta^{(0)}\mathbf{1} \leq 1/\theta^{(r)}\mathbf{1}$. □

## 9.4 Response time distribution

We define $T(r)$ as the response time of a job in a system with probe rate $r$. The response time is defined as the length of the time interval between the arrival of a parent job and the completion of this parent job and all of its spawned child jobs. $T(r)$ can be expressed as the sum of the waiting time $W(r)$ and the service time $J(r)$. The waiting time is defined as the amount of time that the parent job waits in the queue before its service starts. Clearly, the waiting and the service time of a job are independent in our QBD model.

By repeating the arguments from the proof of Theorem 7.6.1, we find that $W(r)$ can be expressed as:

**Theorem 9.4.1.** *The distribution of the waiting time is given by*

$$P[W(r) > t] = (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})e^{\mathbb{W}t} vec\langle I \rangle$$

*with* $\mathbb{W} = ((A_0(r) + A_1)' \otimes I) + ((A_{-1}(r))' \otimes R(r))$ *and where* $vec\langle\cdot\rangle$ *is the column stacking operator. The mean waiting time is*

$$\begin{aligned} E\left[W(r)\right] &= \int_0^\infty P\left[W(r) > t\right] dt \\ &= (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})(-\mathbb{W})^{-1} vec\langle I \rangle. \end{aligned}$$

We now focus on the service time $J(r)$. We can derive recursive formulas for $P[J(r) < t]$ which turn out to not be very suitable for numerical calculations. Therefore, we omit them in this chapter. We can also find a formula for the mean service time for general PH distributed child and parent jobs, however, to improve readability, we opt to present here a scheme for calculating the mean service time in case of hyperexponential parent and child job service requirements. The general formula for the mean response time and its lengthy derivation can be found in the next section.

Consider a set of $j$ servers, where the $k$-th server contains $i_k$ child jobs, with the child job in service being in phase $f_k$, where $j \geq 1, 0 \leq i_1 + \cdots + i_j \leq m$ and $i_k \geq 0$ for $k = 1, \ldots, j$. Let $E_{i_1,\ldots,i_j}^{f_1,\ldots,f_j}(r)$ be the expected time until all these child jobs have completed service. Define similarly $\bar{E}_{i_1,\ldots,i_j}^{f_1,\ldots,f_j}(r)$, except that the first server contains $i_1$ pending child jobs and a parent job that is in service and is in phase $f_1$. By definition, we can drop $i_k$'s that are zero (except $i_1$ in $\bar{E}_{i_1,\ldots,i_j}^{f_1,\ldots,f_j}(r)$) as long as we drop the corresponding upper indices. We also can permute the indices of $E_{i_1,\ldots,i_j}^{f_1,\ldots,f_j}(r)$ except the first one of $\bar{E}_{i_1,\ldots,i_j}^{f_1,\ldots,f_j}(r)$, as long as we permute upper and lower indices in the same way. We have $E_1^k(r) = 1/s_k^c$ and $\bar{E}_0^k(r) = 1/s_k^p$, as $E_1^k(r)$ (resp. $\bar{E}_0^k(r)$) simply denotes the mean time until completion of a single child (resp. parent) job in phase $k$. Consider a configuration with $j \geq 1$ queues, with queue $k = 1, \ldots, j$ having $i_k$ children and the child in service being in phase $f_k$. A service completion in the $k$-th queue therefore occurs at rate $s_{f_k}^c$, which decreases $i_k$ by 1. If $i_k > 1$, then the next child job starts service in phase $s$ with probability $\alpha_s^c$. A steal can only occur in the $k$-th queue if $i_k > 1$, in this case, at rate $rq$, $v$ jobs get transferred with probability $\psi_{i_k-1,v}$ to a new queue and the first of these children start service in phase $s$ with probability $\alpha_s^c$. The total rate at which jobs get completed and stolen equals $\sum_{k=1}^j s_{f_k}^c$ and $rq\sum_{k=1}^j 1[i_k > 1]$ respectively. As such it takes on average $1/\sum_{k=1}^j (s_{f_k}^c + rq1[i_k > 1])$ units of time until a completion or a steal occurs and the probability that a completion (resp. a steal) occurs in $k$-th server is given by $s_{f_k}^c/\sum_{k=1}^j (s_{f_k}^c + rq1[i_k > 1])$ (resp. $rq1[i_k > 1]/\sum_{k=1}^j (s_{f_k}^c + rq1[i_k > 1])$). We therefore get the following recursive relations:

$$E^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r) = \frac{1}{\sum_{k=1}^{j}(s^c_{f_k} + rq1[i_k > 1])}\Big(1 + \sum_{k=1}^{j} s^c_{f_k} \sum_{s=1}^{n_c} \alpha^c_s E^{f_1,\dots,f_{k-1},s,f_{k+1},\dots,f_j}_{i_1,\dots,i_{k-1},i_k-1,i_{k+1},\dots,i_j}(r)$$
$$+ rq \sum_{k=1}^{j} 1[i_k > 1] \sum_{v=1}^{i_k-1} \psi_{i_k-1,v} \sum_{s=1}^{n_c} \alpha^c_s E^{f_1,\dots,f_j,s}_{i_1,\dots,i_{k-1},i_k-v,i_{k+1},\dots,i_j,k}(r)\Big).$$

Similarly, we can derive a recursive formula for $\bar{E}^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r)$, except that now we have a parent job being served in the first queue:

$$\bar{E}^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r) = \frac{1}{s^p_{f_1} + rq1[i_1 > 0] + \sum_{k=2}^{j}(s^c_{f_k} + rq1[i_k > 1])}\Big(1$$
$$+ s^p_{f_1} \sum_{s=1}^{n_c} \alpha^c_s E^{s,f_2,\dots,f_j}_{i_1,\dots,i_j}(r) + rq1[i_1 > 0] \sum_{k=1}^{i_1} \phi_{i_1,k} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\dots,f_j,s}_{i_1-k,i_2,\dots,i_j,k}(r)$$
$$+ \sum_{v=2}^{j} s^c_{f_v} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\dots,f_{v-1},s,f_{v+1},\dots,f_j}_{i_1,\dots,i_{v-1},i_v-1,i_{v+1},\dots,i_j}(r)$$
$$+ rq \sum_{v=2}^{j} 1[i_v > 1] \sum_{k=1}^{i_v-1} \psi_{i_v-1,k} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\dots,f_j,s}_{i_1,\dots,i_{v-1},i_v-k,i_{v+1},\dots,i_j,k}(r)\Big).$$

We then have

$$E[J(r)] = \sum_{k=0}^{m} p_k \sum_{i=1}^{n_p} \alpha^p_i \bar{E}^i_k(r).$$

Hence, for hyperexponential job sizes we have a recursive formula for the mean service time. The idea of this formula can be extended to the case where parent and child jobs have acyclic phase type distributed job requirements, where we condition not only on whether we have a steal or a service completion, but also on whether we have a phase change.

We end this section with an explanation on how to implement the recursive formulas $E^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r)$ and $\bar{E}^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r)$. We first explain how to compute $E^{f_1,\dots,f_j}_{i_1,\dots,i_j}(r)$, where we assume without loss of generality that $i_1 \geq i_2 \geq \dots \geq i_j$. For $k = 1, 2, \dots, m$, let $\mathfrak{p}_k$ denote the number of unique partitions of integer $k$ and let the $\mathfrak{p}_k \times k$ matrix $P_k$ be a list of the unique partitions of integer $k$, for example

$$P_4 = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

and $\mathfrak{p}_4 = 5$. If $i_1 + i_2 + \dots + i_j = k$, then if we ignore the zeros of $P_k$, its rows contain all possible tuples $(i_1, \dots, i_j)$. Note, that the number of queues in a given configuration

that can have their jobs stolen is simply given by the number of integers greater than 1 in the corresponding row of $P_k$. Similarly, the number of busy servers in a configuration is given by the number of non-zero entries in the corresponding row of $P_k$.
For $k = 1, \dots, m$ set $E_k$ as the zero matrix of size $\mathfrak{p}_k \times n_c^k$. If $i_1 + \dots + i_j = k$ and if $(i_1, \dots, i_j)$ can be found in the $g$-th row of $P_k$, then we would like the $(g, h)$-th entry of $E_k$ to be equal to $E_{i_1,\dots,i_j}^{f_1,\dots,f_j}(r)$, where $h = 1 + \sum_{s=1}^{j}(f_s - 1)n_c^{s-1}$. To calculate the entries of the $g$-th row of $E_k$ we only need to know the lower rows of $E_k$ (due to steals) and the matrix $E_{k-1}$ (due to completions). As $E_1 = [1/s_1^c, \dots, 1/s_{n_c}^c]$, we can calculate the entries of $E_k$ inductively, where, for every $k$, the rows of $E_k$ are calculated from the bottom up.

$\bar{E}_{i_1,\dots,i_j}^{f_1,\dots,f_j}(r)$ can be computed similarly, although we now have to account for the parent in the first server. W.l.o.g. assume that $i_2 \geq i_3 \geq \dots \geq i_j$, $i_1 \geq 0$ and $1 + i_1 + i_2 + \dots + i_j = k$. Let $\bar{P}_k$ be the matrix containing all the tuples $(i_1, \dots, i_j)$. Let $\bar{\mathfrak{p}}_k$ denote the number of rows of $\bar{P}_k$. We denote the zero matrix of dimension $k \times \ell$ as $0_{k,\ell}$, i.e. $0_{k,\ell} = 0_k \otimes 0'_\ell$. Then $\bar{P}_k$ can be constructed as follows:

$$\bar{P}_k = \begin{bmatrix} k & \multicolumn{2}{c}{0_{1,k-1}} \\ (k-1)1_{\mathfrak{p}_1} & P_1 & 0_{\mathfrak{p}_1,k-2} \\ (k-2)1_{\mathfrak{p}_2} & P_2 & 0_{\mathfrak{p}_2,k-3} \\ \vdots & \vdots & \vdots \\ 2 \cdot 1_{\mathfrak{p}_{k-2}} & P_{k-2} & 0_{\mathfrak{p}_{k-2},1} \\ 1_{\mathfrak{p}_{k-1}} & \multicolumn{2}{c}{P_{k-1}} \end{bmatrix}.$$

Note that the first column of $\bar{P}_k$ represents the number of jobs in the first queue (including the parent job).
For $k = 1, \dots, m+1$ set $\bar{E}_k$ as the zero matrix of size $\bar{\mathfrak{p}}_k \times n_p n_c^{k-1}$. If $1 + i_1 + \dots + i_j = k$ and if $(i_1, \dots, i_j)$ corresponds to $g$-th row of $\bar{P}_k$, then we would like the $(g, h)$-th entry of $\bar{E}_k$ to be equal to $\bar{E}_{i_1,\dots,i_j}^{f_1,\dots,f_j}(r)$, where $h = 1 + \sum_{s=2}^{j}(f_s - 1)n_c^{s-2} + (f_1 - 1)n_c^{j-1}$. To calculate the entries of the $g$-th row of $\bar{E}_k$ we only need to know the lower rows of $\bar{E}_k$ (due to steals) and the matrices $\bar{E}_{k-1}$ and $E_{k-1}$ (due to child and parent completions respectively). As $\bar{E}_1 = [1/s_1^p, \dots, 1/s_{n_p}^p]$, we can calculate the entries of $\bar{E}_k$ inductively, where, for every $k$, the rows of $\bar{E}_k$ are calculated from the bottom up.

## 9.5 Mean service time for PH distributed parents and children

In this section we develop a recursive scheme for the mean service time in case of PH distributed parents and children. Sadly, this scheme is not very practical in most cases.

Consider a set of $\ell$ servers, where the $k$-th server contains $i_k$ child jobs, with the child job in service having $\beta^k$ as its initial probability distribution, where $\ell \geq 1$, $0 \leq i_1 + \dots + i_\ell \leq m$ and $i_k \geq 0$ for $k = 1, \dots, \ell$. Let $E_{i_1,\dots,i_\ell}^{\beta^1,\dots,\beta^\ell}(r)$ be the expected time until all these child jobs have completed service. Define similarly $\bar{E}_{i_1,\dots,i_\ell}^{\bar{\beta}^1,\beta^2,\dots,\beta^\ell}(r)$, except that the first server contains $i_1$ pending child jobs and a parent job that is in service and has initial probability distribution $\bar{\beta}^1$.

By definition, we can drop $i_k$'s that are zero (except $i_1$ in $\bar{E}^{\bar{\beta}^1,\beta^2,\dots,\beta^\ell}_{i_1,\dots,i_\ell}(r)$) as long as we drop the corresponding upper indices at the same time. We also can permute the indices of $E^{\beta^1,\dots,\beta^\ell}_{i_1,\dots,i_\ell}(r)$ and all indices except the first one of $\bar{E}^{\bar{\beta}^1,\beta^2,\dots,\beta^\ell}_{i_1,\dots,i_\ell}(r)$, as long as we permute upper and lower indices in the same way. Note that due to Theorem 2.2.9 the minimum and maximum of two PH distributed variables is again PH distributed and it follows that the minimum and maximum of finitely many PH variables are also PH distributed. Let $X_k \sim PH(\beta^k, S^c)$ for $k = 1, \dots, \ell$. Using Theorem 2.2.9, we get a formula for $E^{\beta^1,\dots,\beta^\ell}_{1'_\ell}(r)$ as

$$E^{\beta^1,\dots,\beta^\ell}_{1'_\ell}(r) = E\left[\max_{k=1}^{\ell}\{X_k\}\right]$$

is just the expected value of a phase type distribution. Define for $k = 1, \dots, \ell$ the variables $Y_k \sim PH(\beta^k, S^c - 1[i_k > 1]rqI_{n_c})$ and $Z_k$, where $Z_k \sim \exp(rq)$, if $i_k > 1$ and $Z_k = +\infty$ otherwise. We have that

$$\begin{aligned}E^{\beta^1,\dots,\beta^\ell}_{i_1,\dots,i_\ell}(r) &= E\left[\min_{k=1}^{\ell}\{Y_k\}\right] + \sum_{k=1}^{\ell} P[X_k < \min\{\{Y_\nu | \nu \neq k\} \cup \{Z_k\}\}] \cdot E^{\gamma_k^1,\dots,\gamma_k^{k-1},\alpha^c,\gamma_k^{k+1},\dots,\gamma_k^\ell}_{i_1,\dots,i_{k-1},i_k-1,i_{k+1},\dots,i_\ell}(r) \\ &+ \sum_{k=1}^{\ell} P[Z_k < \min\{\{Y_\nu | \nu \neq k\} \cup \{X_k\}\}] \cdot \sum_{j=1}^{i_k-1} \psi_{i_k-1,j} E^{\eta_k^1,\dots,\eta_k^\ell,\alpha^c}_{i_1,\dots,i_{k-1},i_k-j,i_{k+1},\dots,i_\ell,j}(r),\end{aligned}$$

where $\gamma_k^\nu$ denotes the distribution of the phases in the $\nu$-th server at the time of the completion in $k$-th server and where $\eta_k^\nu$ denotes the distribution of the phases in the $\nu$-th server at the time of the steal in $k$-th server. We have

$$\begin{aligned}\gamma_k^\nu &= \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1])t})(\beta^k e^{S^c t} s^c) e^{-1[i_k>1]rqt} \prod_{\eta \neq k,\nu} (\beta^\eta e^{(S^c - 1[i_\eta > 1]rqI)t}\mathbf{1})dt}{P[X_k < \min\{\{Y_i | i \neq k\} \cup \{Z_k\}\}]}, \\ \eta_k^\nu &= \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1])t})(rqe^{-rqt})\beta^k e^{S^c t}\mathbf{1} \prod_{\eta \neq k,\nu} (\beta^\eta e^{(S^c - 1[i_\eta > 1]rqI)t}\mathbf{1})dt}{P[Z_k < \min\{\{Y_i | i \neq k\} \cup \{X_k\}\}]}\end{aligned}$$

for $\nu \neq k$. If $\nu = k$, $\gamma_k^\nu$ is simply $\alpha^c$, while the formula for $\eta_k^\nu$ is slightly different:

$$\eta_k^k = \frac{\int_0^\infty (\beta^k e^{S^c t})(rqe^{-rqt}) \prod_{\eta \neq k} (\beta^\eta e^{(S^c - 1[i_\eta > 1]rqI)t}\mathbf{1})dt}{P[Z_k < \min\{\{Y_i | i \neq k\} \cup \{X_k\}\}]}.$$

In a similar way we calculate the expected service time when one of the servers is serving a parent. Define the variables $X_k, Y_k, Z_k$ as before for $k = 2, \dots, \ell$. Define further the variables $X_1 \sim PH(\bar{\beta}^1, S^p)$, $Y_1 \sim PH(\bar{\beta}^1, S^p - rqI_{n_p})$ and $Z_1$, where $Z_1 \sim \exp(rq)$ if $i_1 > 0$ and $Z_1 = +\infty$ otherwise. We first note that

$$\bar{E}^{\bar{\beta}^1,\beta^2,\dots,\beta^\ell}_{0,1'_{s-1}}(r) = E\left[\max_{k=1}^{\ell}\{X_k\}\right].$$

We have

$$\begin{aligned}\bar{E}^{\bar{\beta}^1,\beta^2,\dots,\beta^\ell}_{i_1,\dots,i_s}(r) &= E\left[\min_{k=1}^{\ell}\{Y_k\}\right] + P[X_1 < \min\{\{Y_\nu | \nu \neq 1\} \cup \{Z_1\}\}] \cdot E^{\alpha_c,\gamma_1^2,\dots,\gamma_1^\ell}_{i_1,\dots,i_\ell}(r) \\ &+ P[Z_1 < \min\{\{Y_\nu | \nu \neq 1\} \cup \{X_1\}\}] \cdot \sum_{j=1}^{i_k} \phi_{i_k,j} \bar{E}^{\eta_1^1,\dots,\eta_1^\ell,\alpha^c}_{i_1-j,i_2,\dots,i_\ell,j}(r)\end{aligned}$$

$$+\sum_{k=2}^{\ell} P[X_k < \min\{\{Y_\nu | \nu \neq k\} \cup \{Z_k\}\}] \cdot \bar{E}_{i_1,\dots,i_{k-1},i_k-1,i_{k+1},\dots,i_\ell}^{\gamma_k^1,\dots,\gamma_k^{k-1},\alpha^c,\gamma_k^{k+1},\dots,\gamma_k^\ell}(r)$$
$$+\sum_{k=2}^{\ell} P[Z_k < \min\{\{Y_\nu | \nu \neq k\} \cup \{X_k\}\}] \cdot \sum_{j=1}^{i_k-1} \psi_{i_k-1,j} \bar{E}_{i_1,\dots,i_{k-1},i_k-j,i_{k+1},\dots,i_\ell,j}^{\eta_k^1,\dots,\eta_k^\ell,\alpha^c}(r),$$

where $\gamma_k^\nu$ and $\eta_k^\nu$ have similar formulas as before, expect for the first server which services a parent. For $k, \nu = 2, \dots, \ell$ with $k \neq \nu$ we therefore have

$$\gamma_k^\nu = \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1] rqI)t})(\beta^k e^{S^c t} s^c) e^{-1[i_k>1]rqt} (\bar{\beta}^1 e^{(S^p - 1[i_1>0]rqI)t}\mathbf{1}) \prod_{\eta \neq 1,k,\nu} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[X_k < \min\{\{Y_i | i \neq k\} \cup \{Z_k\}\}]}.$$

Note that now $X_1$ and $Y_1$ are defined differently. We now have for $k = 1$ and $\nu = 2, \dots, \ell$

$$\gamma_1^\nu = \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1] rqI)t})(\bar{\beta}^1 e^{S^p t} s^p) e^{-1[i_1>0]rqt} \prod_{\eta \neq 1,\nu} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[X_1 < \min\{\{Y_i | i \neq 1\} \cup \{Z_1\}\}]}$$

and for $k = 2, \dots, \ell$ and $\nu = 1$

$$\gamma_k^1 = \frac{\int_0^\infty (\bar{\beta}^1 e^{(S^p - 1[i_1 > 0] rqI)t})(\beta^k e^{S^c t} s^c) e^{-1[i_k>1]rqt} \prod_{\eta \neq 1,k} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[X_k < \min\{\{Y_i | i \neq k\} \cup \{Z_k\}\}]}.$$

Similarly, for $k, \nu = 2, \dots, \ell$, with $k \neq \nu$, we have

$$\eta_k^\nu = \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1] rqI)t})(rqe^{-rqt}) \beta^k e^{S^c t}\mathbf{1} (\bar{\beta}^1 e^{(S^p - 1[i_1>0]rqI)t}\mathbf{1}) \prod_{\eta \neq 1,k,\nu} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[Z_k < \min\{\{Y_i | i \neq k\} \cup \{X_k\}\}]},$$

while for $k = \nu$ with $k, \nu \neq 1$

$$\eta_k^k = \frac{\int_0^\infty (\beta^k e^{S^c t})(rqe^{-rqt})(\bar{\beta}^1 e^{(S^p - 1[i_1>0]rqI)t}\mathbf{1}) \prod_{\eta \neq 1,k} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[Z_k < \min\{\{Y_i | i \neq k\} \cup \{X_k\}\}]}.$$

We have for $k = 1$ and $\nu = 2, \dots, \ell$

$$\eta_1^\nu = \frac{\int_0^\infty (\beta^\nu e^{(S^c - 1[i_\nu > 1] rqI)t})(rqe^{-rqt})(\bar{\beta}^1 e^{S^p t}\mathbf{1}) \prod_{\eta \neq 1,\nu} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[Z_1 < \min\{\{Y_i | i \neq 1\} \cup \{X_1\}\}]}$$

and for $k = 2, \dots, \ell$ and $\nu = 1$

$$\eta_k^1 = \frac{\int_0^\infty (\bar{\beta}^1 e^{(S^p - 1[i_1 > 0] rqI)t})(rqe^{-rqt})(\beta^k e^{S^c t}\mathbf{1}) \prod_{\eta \neq 1,k} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[Z_k < \min\{\{Y_i | i \neq k\} \cup \{X_k\}\}]}$$

and, finally, for $k = \nu = 1$

$$\eta_1^1 = \frac{\int_0^\infty (\bar{\beta}^1 e^{S^p t})(rqe^{-rqt}) \prod_{\eta \neq 1} (\beta^\eta e^{(S^c - 1[i_\eta>1]rqI)t}\mathbf{1}) dt}{P[Z_1 < \min\{\{Y_i | i \neq 1\} \cup \{X_1\}\}]}.$$

We now get

$$E[J(r)] = \sum_{k=0}^{m} p_k \bar{E}_k^{\alpha^p}(r).$$

## 9.6 Numerical experiments

In Section 8.6, we defined the class of monotone deterministic (MD) strategies and we tested in different settings the strategies of stealing a single child job, half of the waiting children and all waiting children against the optimal MD strategy for those settings in case of exponential parent/child job sizes. We concluded that the stealing policy where the half of child jobs gets stolen is in general a good stealing policy for higher values of $r$ and moderate system loads $\rho$, while the strategy of stealing all children performs best for low values of $r$ and higher values of $\rho$. We concluded further that stealing only one child performs the worst in most of the cases. In this section we examine whether these conclusions remain valid for systems with hyperexponential parent and child job sizes with two phases ($HypExp(2)$).
To this end we describe a $HypExp(2)$ distribution using the parameters $EX, SCV, f$ where $EX$ is the mean of the distribution, where $SCV$ is the squared coefficient of variation and where $f$ is the fraction of the workload contributed by phase 1 jobs ($f$ is sometimes called the shape parameter). Using these parameters we can generate a $HypExp(2)$ distribution with parameters $([\beta_1, 1-\beta_1], [\mu_1, \mu_2])$, where

$$\mu_1 = \frac{SCV + (4f-1) + \sqrt{(SCV-1)(SCV-1+8f(1-f))}}{2EX \cdot f(SCV+1)},$$
$$\mu_2 = \frac{SCV + (4(1-f)-1) - \sqrt{(SCV-1)(SCV-1+8f(1-f))}}{2EX \cdot (1-f)(SCV+1)}$$

and $\beta_1 = EX \cdot \mu_1 f$. In the remainder of the chapter we use the same values of $SCV$ and $f$ for parent and child jobs. We assume that parent and child jobs have service requirements with mean 2 and mean 1 respectively.

Recall that we defined the matrix $\Psi$ as the matrix where $[\Psi]_{i,j} = \psi_{i,j}$ and $\Phi$ similarly. Note that a strategy is fully characterized by the matrices $\Psi$ and $\Phi$. The strategies of stealing a single child job, half of child jobs and all children are defined as follows:

1. **Steal one:** The strategy of always stealing one child job, that is $\phi_{i,1} = \psi_{i,1} = 1$ for every $i$.
2. **Steal half:** The strategy of always stealing half of the pending child jobs. If $n$, the number of pending child jobs, is uneven, there is a fifty percent chance that $\lfloor n/2 \rfloor$ child jobs get stolen and $\lceil n/2 \rceil$ jobs otherwise.
3. **Steal all:** The strategy of stealing all of the pending child jobs, that is $\phi_{i,i} = \psi_{i,i} = 1$ for every $i$.

Note that these strategies do not rely on any knowledge on the (mean) job sizes or system load. In Section 8.6 a strategy was called monotone deterministic (MD) if, for every $i$, $\psi_{i,j} = 1$ implies $\psi_{i+1,j'} = 1$ for some $j' \geq j$ and the same holds for $\Phi$. The optimal MD strategy is determined using brute force and its response time is denoted as $T_{MD}(r)$. The mean response time of other strategies is always normalized by $T_{MD}(r)$ in the subsequent experiments.
We now present a selection of performed numerical experiments (due to the lack of space). The main conclusions in the omitted experiments are in agreement with the results presented here. Let $\mathbf{p} = [p_0, p_1, \ldots, p_m]$.

**Example 9.6.1.** In Figure 9.1 we illustrate the effect of increasing the load $\rho$ on the on performance of the three strategies for different values of $SCV$. We do this for $\rho \in [0.05, 0.95], SCV \in \{2, 5, 20\}, f = 1/3, \mathbf{p} = [5, 4, 3, 2, 1]/15$ and $r = 2$. These results (and other results omitted here) confirm that stealing all is best when the load is sufficiently high, while stealing half of the child jobs is a good strategy for systems with a moderate load.

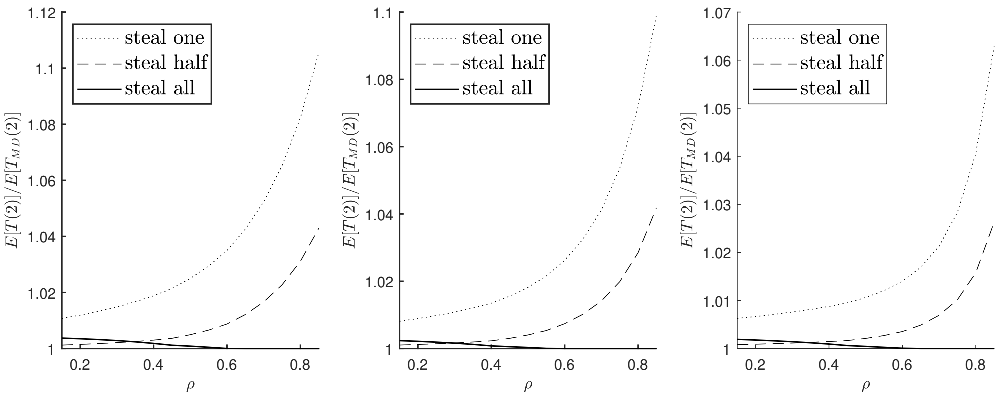


Figure 9.1: Example 9.6.1: $E[T(2)]/E[T_{MD}(2)]$ in function of $\rho$ with $SCV = 2$ (left), $SCV = 5$ (mid) and $SCV = 20$ (right).

**Example 9.6.2.** In Figure 9.2 we consider the system with $\rho = 0.85$, $SCV \in [1, 20]$, $f = 1/2$, $\mathbf{p} = 1_6'/6$ and $r \in \{1, 5, 10\}$. We examine the effect of increasing the value of the $SCV$ on the performance of the three stealing strategies against the performance of the system with no stealing. Clearly, as the $SCV$ increases the ratio $E[T(r)]/E[T(0)]$ decreases for each of the three strategies. In fact, for every stealing strategy we have that as the $SCV \to \infty$, the ratio $E[T(r)]/E[T(0)] \to 0$. This implies that as the $SCV$ increases it is more and more worthwhile to steal.

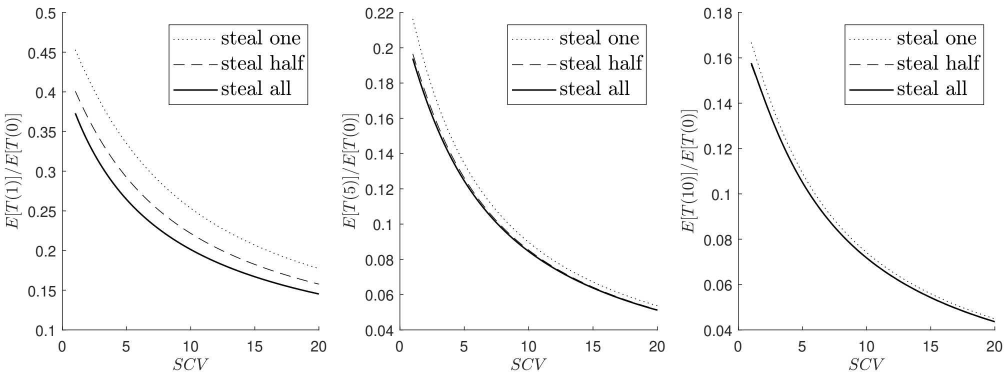


Figure 9.2: Example 9.6.2: $E[T(r)]/E[T(0)]$ in function of $r$ with $r = 1$ (left), $r = 5$ (mid) and $r = 10$ (right).

**Example 9.6.3.** Example 9.6.2 shows that the mean response time of the strategy of not stealing grows quicker than those of the strategies where stealing occurs. In this example we examine this growth in more detail. In Figure 9.3, we therefore plot $E[T(r)]$ in function of $SCV$ for the three strategies and for the system where no stealing occurs. We do this for $\rho = 0.85$, $SCV \in [1, 40]$, $f = 1/2$, $\mathbf{p} = 1_6'/6$ and $r = 5$. Clearly, the growth of $E[T(0)]$ (no stealing) is linear in function of $SCV$. Further as $SCV \to \infty$, $E[T(r)]$ seems to converge for any stealing strategy, which is equivalent to saying that from the moment that $SCV$ is large enough, there is not much difference in the performance of a stealing strategy when the $SCV$ is increased further.

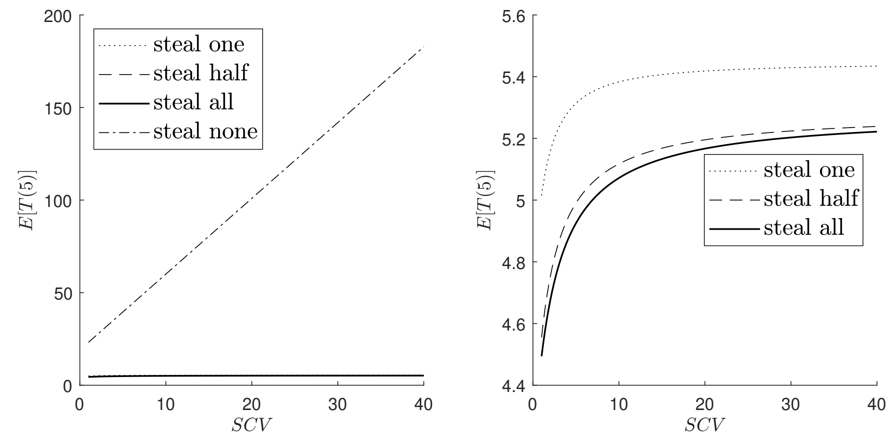


Figure 9.3: Example 9.6.3 with the system where no stealing occurs (left) and without that system (right).

## 9.7 Model validation

Based on numerical experiments in the previous section, we see that stealing all or half of the children are good stealing policies, stealing all works best for low values of $r$, while stealing half of the children works well for higher values. Therefore we validate the model for these two policies by means of simulation. We run all simulations for $T = 10^5$ with a warm up period of 33% of $T$, always starting from an empty system.

In Figure 9.4, we compare the simulated waiting and service time distributions and those of the QBD model. We do this for $\rho = 0.85, SCV = 2, f = 1/2, \mathbf{p} = 1_5'/5$ and $r \in \{1, 5\}$. The simulated waiting and service times were calculated based on 5 runs, with the number of queues $N = 2000$. We see that there is a good match between the simulated waiting and service time distributions and those of the QBD model. Also, that the match is less good for $r = 5$ than for $r = 1$. Note that having 2000 or more CPU-cores is not uncommon in an HPC cluster.

In Table 9.2 we compare the relative error of the simulated mean response time, based on 20 runs, to the one obtained using formula from Section 9.4. In We do this for the policy of stealing half of the available children. Similar results were obtained when all children are stolen. We simulate the systems for $f = 1/2, SCV \in \{2, 20\}, \mathbf{p} = 1_5'/5, \rho \in \{0.75, 0.85\}$, $r = 1$ and $N \in \{250, 500, 1000, 2000, 4000\}$.

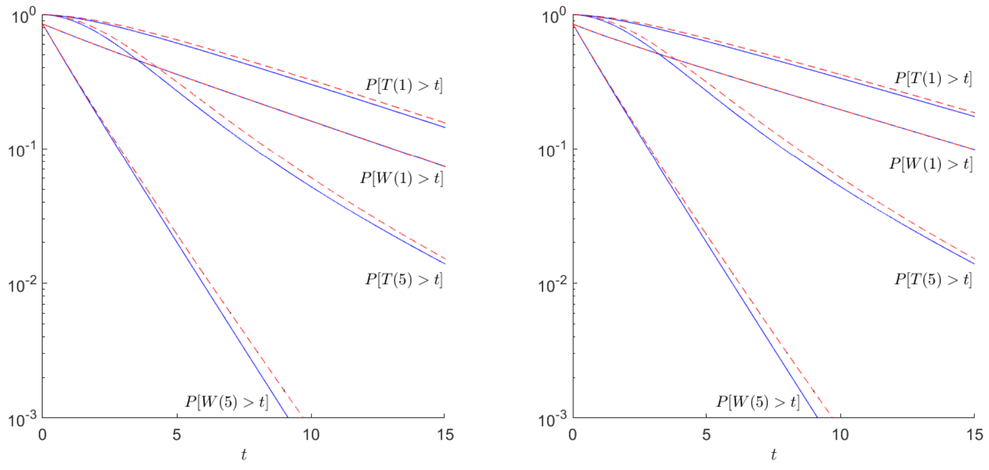


Figure 9.4: Waiting and response times from the QBD (blue) and simulations (red) for the strategies of stealing all children (left) and half of the children (right).

| | $\rho = 0.75$ | | $\rho = 0.85$ | |
|---|---|---|---|---|
| $N$ | sim. ± conf. | rel.err.% | sim. ± conf. | rel.err.% |
| $SCV = 2$ | | | | |
| 250 | 6.4925 ± 6.67e-03 | 0.4706 | 9.5338 ± 1.72e-02 | 0.7855 |
| 500 | 6.4788 ± 3.88e-03 | 0.2586 | 9.4893 ± 1.48e-02 | 0.3156 |
| 1000 | 6.4683 ± 3.94e-03 | 0.0963 | 9.4691 ± 7.66e-03 | 0.1021 |
| 2000 | 6.4638 ± 2.13e-03 | 0.0260 | 9.4647 ± 7.38e-03 | 0.0547 |
| 4000 | 6.4635 ± 9.64e-04 | 0.0214 | 9.4597 ± 4.51e-03 | 0.0024 |
| QBD | 6.4621 | | 9.4595 | |
| $SCV = 20$ | | | | |
| 250 | 8.1792 ± 2.88e-02 | 2.0152 | 17.1200 ± 1.18e-01 | 2.3903 |
| 500 | 8.0953 ± 1.77e-02 | 0.9686 | 16.8921 ± 7.14e-02 | 1.0272 |
| 1000 | 8.0473 ± 8.81e-03 | 0.3701 | 16.8081 ± 6.00e-02 | 0.5248 |
| 2000 | 8.0347 ± 8.64e-03 | 0.2127 | 16.7477 ± 3.80e-02 | 0.1637 |
| 4000 | 8.0226 ± 7.66e-03 | 0.0619 | 16.7388 ± 3.72e-02 | 0.1104 |
| QBD | 8.0176 | | 16.7204 | |

Table 9.2: Relative error of simulation results for $E[T(r)]$ for the policy of stealing half of the children, based on 20 runs.

The relative error in all cases is below 2.5% and tends to increase with the value of the $SCV$. We note that based on simulations omitted here the relative error tends to increase with the steal rate $r$, which is in agreement with Figure 9.4. More importantly the relative error seems roughly to halve when doubling $N$ (which is in agreement with the results in [24]). This suggests that the approximation error tends to zero as the number of servers tends to infinity.

## 9.8 Mean field model

In this section we present a mean field model for the work stealing system considered in this chapter and show that it has a unique fixed point that coincides with the steady state vector of the QBD Markov chain. In this manner we provide additional support for the claim that the approximation error of the QBD model tends to zero as the number of servers becomes large. Note that this result is not sufficient to formally prove this. One of the main challenges in coming up with such a formal proof is to establish global attractor of the fixed point, which is often done using monotonicity arguments. This however is not feasible for the system considered in this chapter, as the system is clearly not monotone in some cases (e.g. a system where out of 5 child jobs always 5 get stolen, whereas out of 4 children only one is transferred upon a successful steal attempt).

We start by writing down the set of ODEs that capture the evolution of the mean field model (i.e., the so-called drift equations). We denote by $f_{\ell,j,k,i}(t)$ the fraction of queues at time $t$ with $\ell$ parent jobs in waiting in the queue, $j \in \{1,\ldots,m\}$ child jobs in the queue, $k \in \{0,1\}$ describing whether a parent job is in service ($k=1$) or not ($k=0$) and $i$ being the phase of the job currently in service (if $k=1$, then $i \in \{1,\ldots,n_p\}$ and if $k=0$, then $i \in \{1,\ldots,n_c\}$). When there is no job in service we set $i=0$. Note that $\ell$ does not count parent jobs in service, whereas $j$ counts child jobs waiting and in service. In particular for $\ell=0$ and $j+k \geq 1$ the server is busy and there may be child jobs waiting, which can be transferred. We denote $f_{0,0,0,0}(t)$ as $f_*(t)$, the fraction of idle queues. For a statement $A$ we set $1[A]$ to be 1 if $A$ is true and 0 if $A$ is false.
Let

$$\vec{f_\ell}(t) = (\vec{f}_{\ell,1,0}(t),\ldots,\vec{f}_{\ell,m,0}(t),\vec{f}_{\ell,0,1}(t),\ldots,\vec{f}_{\ell,m,1}(t))$$

for every $\ell \geq 0$, where

$$\vec{f}_{\ell,j,0}(t) = (\vec{f}_{\ell,j,0,1}(t),\ldots,\vec{f}_{\ell,j,0,n_c}(t)) \text{ and}$$
$$\vec{f}_{\ell,j,1}(t) = (\vec{f}_{\ell,j,1,1}(t),\ldots,\vec{f}_{\ell,j,1,n_p}(t)).$$

Then, for $\ell > 0$ we have

$$\frac{d}{dt}\vec{f_\ell}(t) = \lambda\vec{f}_{\ell-1}(t) + \vec{f_\ell}(t)\tilde{A}_0(t) + \vec{f}_{\ell+1}(t)\mu\alpha + rf_*(t)\vec{f}_{\ell+1}(t)V_0, \tag{9.9}$$

for $\ell = 0$ and $j+k \geq 1$ we have

$$\begin{aligned}\frac{d}{dt}\vec{f_0}(t) = \lambda f_*(t)\alpha + \vec{f_0}(t)\tilde{B}_0(t) + \vec{f_1}(t)\mu\alpha + rf_*(t)\vec{f_1}(t)V_0 \\ + rf_*(t)\sum_{\ell'\geq 0}\vec{f}_{\ell'}(t)T + rf_*(t)\sum_{\ell'\geq 1}\vec{f}_{\ell'}(t)v_0\alpha,\end{aligned} \tag{9.10}$$

and for $\ell, j, k = 0$

$$\frac{d}{dt}f_*(t) = -\lambda f_*(t) + \vec{f_0}(t)\mu - rf_*(t)\big(1 - f_*(t) - \vec{f_0}(t)v_0\big). \tag{9.11}$$

The first term of (9.9) and (9.10) is due to arrivals. The matrices $\tilde{A}_0(t)$ and $\tilde{B}_0(t)$ are the same matrices as $A_0(r)$ and $B_0(r)$ respectively, except with every instance of $q$ changed to $f_*(t)$. The second term of (9.9) and (9.10) therefore denotes the drift due to transitions for which the level remains unchanged, due to arrivals to and due to parent steals from queues of length $\ell$. The third and the fourth term are due, respectively, to service completions and parent steals in queues of length $\ell+1$. We denote $v_0 = \begin{bmatrix} 1'_{n_c} & 0'_{(m-1)n_c} & 1'_{n_p} & 0'_{mn_p} \end{bmatrix}'$, where the entries are non-zero when $j+k=1$ (i.e. $V_0 = \text{diag}(v_0)$). We define the $(m-1)\times(m-1)$ matrix $\Psi$ as $[\Psi]_{i,j} = \psi_{i,j}$ and similarly $m \times m$ matrix $\Phi$. Recall that we denote the zero matrix of dimension $k \times \ell$ as $0_{k,\ell}$. The matrix $T = T_\psi + T_\phi$ records the distribution of the number of child jobs transferred when a probe is successful:

$$T_\psi = \begin{bmatrix} 0_{n_c,(m-1)n_c} & 0_{n_c,n_c+(m+1)n_p} \\ \Psi \otimes (1_{n_c}\alpha^c) & 0_{(m-1)n_c,n_c+(m+1)n_p} \\ 0_{(m+1)n_p,(m-1)n_c} & 0_{(m+1)n_p,n_c+(m+1)n_p} \end{bmatrix}, T_\phi = \begin{bmatrix} 0_{mn_c+n_p,mn_c} & 0_{mn_c+n_p,(m+1)n_p} \\ \Phi \otimes (1_{n_p}\alpha^c) & 0_{mn_p,(m+1)n_p} \end{bmatrix}.$$

The final two terms of (9.10) are thus due to transfers to empty queues of child jobs and of parents respectively. Similarly, for $\frac{d}{dt}f_*(t)$ the first term is due to job arrivals, the next is due to service completions and the last is due to job transfers.
Note that if $\phi_{i,1} = 1$ for every $i \in \{1,\dots,m\}$ and if $\psi_{j,1} = 1$ for every $j \in \{1,\dots,m-1\}$, then $T = (\mathbf{1} - v_0)\kappa_1$.

We show that the stationary distribution of the QBD corresponds to the unique fixed point $\zeta$ of the set of ODEs in Equations (9.9)-(9.11). The following lemma says that, in equilibrium, the rate at which the level in non-empty queues increases, that is $\lambda(1-\zeta_*)$, is exactly the rate at which the level decreases in such queues (which can only happen due to a service completion or a steal in queues with no pending child jobs).

**Lemma 9.8.1.** *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ with $\zeta_* + \sum_{\ell \ge 0} \vec{\zeta}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (9.9)-(9.11) *we have*

$$\lambda = \lambda\zeta_* + \sum_{\ell \ge 1} \vec{\zeta}_\ell \mu + r\zeta_* \sum_{\ell \ge 1} \vec{\zeta}_\ell v_0.$$

*Proof.* As $\frac{d}{dt}\vec{f}_\ell(t) = 0$ in a fixed point we get using $\sum_{\ell\ge0}(\ell+1)\frac{d}{dt}\vec{f}_\ell(t) = 0$ and $\sum_{\ell \ge 0}\vec{\zeta}_\ell \mathbf{1} = 1 - \zeta_*$ that

$$\sum_{\ell \ge 0} \vec{\zeta}_\ell \mu = \lambda + r\zeta_* \left(1 - \zeta_* - \sum_{\ell \ge 0} \vec{\zeta}_\ell v_0\right). \tag{9.12}$$

The claim now follows by using (9.12) and (9.11) in a fixed point. □

Define recursively the row vector

$$\xi_{1,m} = \lambda p_m \alpha^p$$

and

$$\xi_{1,k} = \lambda p_k \alpha^p + r\zeta_* \sum_{d=k+1}^{m} \phi_{d,d-k}\xi_{1,d}(r\zeta_* I - S^p)^{-1},$$

for $k = 0,\dots,m-1$, and

$$\xi_{0,k'} = \xi_{1,k'}(r\zeta_* I - S^p)^{-1} s^p \alpha^c + r\zeta_* \sum_{d=k'}^{m} \phi_{d,k'}\xi_{1,d}(r\zeta_* I - S^p)^{-1} 1_{n_p}\alpha^c$$

$$+ 1[k' < m]\xi_{0,k'+1}(r\zeta_* I - S^c)^{-1} s^c \alpha^c$$
$$+ r\zeta_* \sum_{d=k'+1}^{m} \xi_{0,d}(r\zeta_* I - S^c)^{-1}(\psi_{d-1,d-k'} + \psi_{d-1,k'} 1_{n_c}\alpha^c),$$

for $k' = 1, \dots, m$. $\xi_{i,j,k}$ is the rate at which servers enter into phase $(i,j,k)$ due to arrivals, completions and steals.

The intuition behind the next two lemmas is that if the system is in equilibrium, the rate at which queues enter phase $(i,j,k)$ should equal to the rate at which queues leave phase $(i,j,k)$.

**Lemma 9.8.2.** *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ with $\zeta_* + \sum_{\ell\geq 0} \vec{\zeta}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (9.9)-(9.11) *we have for* $1 \leq k \leq m$*:*

$$\sum_{\ell\geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p} \end{pmatrix} (r\zeta_* I - S^p) = \xi_{1,k}. \tag{9.13}$$

*Proof.* We prove the lemma using complete backward induction on $k$. By demanding that
$\sum_{\ell\geq 0} \vec{f}_\ell(t) \left[0'_{mn_c+kn_p,n_p}, I_{n_p}, 0'_{(m-k)n_p,n_p}\right]' = 0$ for any $k \in \{1,\dots,m\}$, we find due to Lemma 9.8.1 that

$$0 = \lambda p_k \alpha^p - \sum_{\ell\geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p} \end{pmatrix} (r\zeta_* I - S^p)$$
$$+ r\zeta_* \sum_{\ell\geq 0} \vec{\zeta}_\ell \left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1} I_{n_p}, \dots, \phi_{m,m-k} I_{n_p}\right]'.$$

This is equivalent to

$$\sum_{\ell\geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p} \end{pmatrix} (r\zeta_* I - S^p)$$
$$= \lambda p_k \alpha^p + r\zeta_* \sum_{\ell\geq 0} \vec{\zeta}_\ell \left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1} I_{n_p}, \dots, \phi_{m,m-k} I_{n_p}\right]'. \tag{9.14}$$

(9.14) is equivalent to (9.13) for $k = m$. Suppose now that $k < m$ and that (9.13) holds for all $k' \in \{k+1,\dots,m\}$. Due to (9.14), it suffices to show that

$$r\zeta_* \sum_{\ell\geq 0} \vec{\zeta}_\ell \left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1} I_{n_p}, \dots, \phi_{m,m-k} I_{n_p}\right]' = r\zeta_* \sum_{d=k+1}^{m} \phi_{d,d-k}\xi_{1,d}(r\zeta_* I - S^p)^{-1}.$$

This is equivalent to

$$\sum_{\ell\geq 0} \vec{\zeta}_\ell \left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1} I_{n_p}, \dots, \phi_{m,m-k} I_{n_p}\right]' (r\zeta_* I - S^p) = \sum_{d=k+1}^{m} \phi_{d,d-k}\xi_{1,d},$$

which holds due to induction hypothesis. □

**Lemma 9.8.3.** *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ with $\zeta_* + \sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations* (9.9)-(9.11) *we have for $2 \leq k \leq m$:*

$$\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{(k-1)n_c,n_c} \\ I_{n_c} \\ 0_{(m-k-2)n_c+(m+1)n_p,n_c} \end{pmatrix} (r\zeta_* I - S^c) = \xi_{0,k}. \tag{9.15}$$

*Proof.* Similarly to the proof of Lemma 9.8.2, we use complete backward induction on $k$. By demanding

$$\sum_{\ell \geq 0} \frac{d}{dt} \vec{f}_\ell(t) \left[ 0'_{(m-1)n_c,n_c}, I_{n_c}, 0'_{(m+1)n_p,n_c} \right]' = 0,$$

we get

$$\begin{aligned}
\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{(m-1)n_c,n_c} \\ I_{n_c} \\ 0_{(m+1)n_p,n_c} \end{pmatrix} (r\zeta_* I - S^c) &= \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+mn_p,n_c} \\ (r\zeta_* \phi_{m,m} 1_{n_p} + s^p)\alpha^c \end{pmatrix} \\
&= \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+mn_p,n_p} \\ I_{n_p} \end{pmatrix} (r\zeta_* \phi_{m,m} 1_{n_p} + s^p)\alpha^c \\
&= \xi_{1,m}(r\zeta_* I - S^p)^{-1}(r\zeta_* \phi_{m,m} 1_{n_p} + s^p)\alpha^p \\
&= \xi_{0,m},
\end{aligned}$$

where (9.13) was used in the third equality. This shows (9.15) for $k = m$. Suppose $k < m$ and that (9.15) holds for all $k' \in \{k+1, \ldots, m\}$. By demanding

$$\sum_{\ell \geq 0} \frac{d}{dt} \vec{f}_\ell(t) \left[ 0'_{(k-1)n_c,n_c}, I_{n_c}, 0'_{(m-k-2)n_c+(m+1)n_p,n_c} \right]',$$

we get

$$\begin{aligned}
&\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{(k-1)n_c,n_c} \\ r\zeta_* I - S^c \\ 0_{(m-k-2)n_c+(m+1)n_p,n_c} \end{pmatrix} \\
&\quad = r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell \left[ 0'_{kn_c,n_c}, \psi_{k,1} I_{n_c} + \psi_{k,k}(1_{n_c}\alpha^c)', \ldots, \psi_{m-1,m-k} I_{n_c} + \psi_{m-1,k}(1_{n_c}\alpha^c)', 0'_{m+1} \right]' \\
&\quad + r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell \left[ 0'_{mn_c+kn_p,n_c}, \phi_{k,k}(1_{n_p}\alpha^c)', \ldots, \phi_{m,k}(1_{n_p}\alpha^c)' \right]' \\
&\quad + \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{kn_c,n_c} \\ s^c \alpha^c \\ 0_{(m-k-1)n_c+(m+1)n_p,n_c} \end{pmatrix} + \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_c} \\ s^p \alpha^c \\ 0_{(m-k)n_p,n_c} \end{pmatrix}.
\end{aligned}$$

By using the induction hypothesis and (9.13) we further get

$$\begin{aligned}
\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{(k-1)n_c,n_c} \\ r\zeta_* I - S^c \\ 0_{(m-k-2)n_c+(m+1)n_p,n_c} \end{pmatrix} &= r\zeta_* \sum_{d=k+1}^{m} \xi_{0,d}(r\zeta_* I - S^c)^{-1}(\psi_{d-1,d-k} + \psi_{d-1,k}\mathbf{1}\alpha^c) \\
&+ r\zeta_* \sum_{d=k}^{m} \phi_{d,k} \xi_{1,d}(r\zeta_* I - S^p)^{-1}\mathbf{1}\alpha^c
\end{aligned}$$

$$+ \xi_{0,k+1}(r\zeta_* I - S^c)^{-1} s^c \alpha^c + \xi_{1,k}(r\zeta_* I - S^p)^{-1} s^p \alpha^c$$

which is equal to $\xi_{0,k}$. This finishes the proof. □

The intuition behind Equation (9.17) is that the rate at which batches of $j$ child jobs are stolen is equal to the rate at which batches of $j$ child jobs are transferred to empty queues.

**Proposition 9.8.4.** *For any fixed point* $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \dots)$ *with* $\zeta_* + \sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1$ *of the set of ODEs in Equations* (9.9)-(9.11) *we have*

$$\zeta_* = q, \tag{9.16}$$

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T = \zeta_* \sum_{j=1}^{m} \lambda_{c,j}(r) \kappa_j, \tag{9.17}$$

*where* $\lambda_{c,j}(r)$ *was defined in* (9.6).

*Proof.* Denote by 1:$k$ the column vector $[1, \dots, k]'$ for $k \geq 1$. To prove (9.16) it suffices to show

$$\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c} \\ 1_{(m+1)n_p} \end{pmatrix} = \lambda \alpha^p (-S^p)^{-1} \mathbf{1}, \tag{9.18}$$

$$\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 1_{mn_c} \\ 0_{(m+1)n_p} \end{pmatrix} = \lambda \left( \sum_{i=1}^{m} i p_i \right) \alpha^c (-S^c)^{-1} \mathbf{1}. \tag{9.19}$$

By demanding $\sum_{\ell \geq 0} \frac{d}{dt} \vec{f}_\ell(t) = 0$ and by using Lemma 9.8.1, we find

$$0 = \lambda \alpha + \sum_{\ell \geq 0} \vec{\zeta}_\ell \tilde{S} + r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell V_0 - r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell + r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T, \tag{9.20}$$

Where $\tilde{S}$ is the same matrix as $S(r)$ except with all instances of $q$ changed to $\zeta_*$. By multiplying (9.20) with $[0'_{mn_c,n_p}, (1_{m+1} \otimes I_{n_p})']'$, we get

$$0 = \lambda \alpha^p + \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c,n_p} \\ 1_{m+1} \otimes S^p \end{pmatrix}, \tag{9.21}$$

which yields (9.18). By demanding

$$\sum_{\ell \geq 0} \frac{d}{dt} \vec{f}_\ell(t) \begin{pmatrix} 1{:}m \otimes 1_{n_c} \\ 0_{n_p} \\ 1{:}m \otimes 1_{n_p} \end{pmatrix} = 0$$

and by using Lemma 9.8.1, one can show that

$$\sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 1_m \otimes s^c \\ 0_{(m+1)n_p} \end{pmatrix} = \lambda \sum_{i=1}^{m} i p_i. \tag{9.22}$$

By multiplying (9.20) with $[(1_m \otimes I_{n_c})', 0'_{(m+1)n_p,n_c}]'$ and by using (9.12) on the last sum we get

$$\lambda \alpha^c = \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 1_m \otimes s^c \\ 1_{m+1} \otimes s^p \end{pmatrix} \alpha^c + \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 1_m \otimes S^c \\ 0_{(m+1)n_p,n_c} \end{pmatrix}.$$

Due to (9.21) and (9.22) this is equivalent to

$$0 = \lambda \left( \sum_{i=1}^{m} i p_i \right) \alpha^c + \sum_{\ell \geq 0} \vec{\zeta}_\ell \begin{pmatrix} 1_m \otimes S^c \\ 0_{(m+1)n_p,n_c} \end{pmatrix},$$

which gives (9.19). To prove the second claim it suffices to show, due to the definition of $T$, that for $i = 1, \ldots, m$ we have:

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T \begin{pmatrix} 0_{(i-1)n_c} \\ 1_{n_c} \\ 0_{(m-i)n_c+(m+1)n_p} \end{pmatrix} = \zeta_* \lambda_{c,i}(r).$$

This is equivalent to showing the following two equalities:

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\psi \begin{pmatrix} 0_{(i-1)n_c} \\ 1_{n_c} \\ 0_{(m-i)n_c+(m+1)n_p} \end{pmatrix} = \lambda r q \sum_{j>i} \psi_{j-1,i} p_{0,j}(r)(rqI - S^c)^{-1}\mathbf{1} + rq^2 \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} \psi_{k-1,i} p_k^j(r)(rqI - S^c)^{-1}\mathbf{1}, \tag{9.23}$$

for $i = 1, \ldots, m-1$, and

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\phi \begin{pmatrix} 0_{(i-1)n_c} \\ 1_{n_c} \\ 0_{(m-i)n_c+(m+1)n_p} \end{pmatrix} = \lambda r q \sum_{j \geq i} \phi_{j,i} p_{1,j}(r)(rqI - S^p)^{-1}\mathbf{1}, \tag{9.24}$$

for $i = 1, \ldots, m$. Due to (9.13), we have

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\phi \begin{pmatrix} 0_{(i-1)n_c} \\ 1_{n_c} \\ 0_{(m-i)n_c+(m+1)n_p} \end{pmatrix} = r\zeta_* \sum_{d \geq i} \phi_{d,i} \xi_{1,d}(rqI - S^p)^{-1}\mathbf{1}.$$

As

$$\xi_{1,d} = \lambda p_{1,d}(r), \tag{9.25}$$

where all instances of $q$ in the formula of $p_{1,d}(r)$ have been changed to $\zeta_*$, equation (9.24) follows from (9.16). Equation (9.23) requires more work to prove. Due to (9.15), it suffices to show that

$$r\zeta_* \sum_{k=i}^{m-1} \psi_{k,i} \xi_{0,k+1}(r\zeta_* I - S^c)^{-1}\mathbf{1} = \lambda r q \sum_{j>i} \psi_{j-1,i} p_{0,j}(r)(rqI - S^c)^{-1}\mathbf{1} + rq^2 \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} \psi_{k-1,i} p_k^j(r)(rqI - S^c)^{-1}\mathbf{1}.$$

Due to (9.16), it suffices to show that

$$\sum_{k=i+1}^{m} \psi_{k-1,i}\xi_{0,k} = \lambda \sum_{k=i+1}^{m} p_{0,k}(r)\psi_{k-1,i} + q \sum_{k=i+1}^{m}\sum_{j=k}^{m} \lambda_{c,j}(r)p_k^j(r)\psi_{k-1,i}. \tag{9.26}$$

We show that for $k = 2, \ldots, m$, we have

$$\xi_{0,k} = \lambda p_{0,k}(r) + q\sum_{j=k}^{m} \lambda_{c,j}(r)p_k^j(r) \tag{9.27}$$

and (9.26) then follows. We prove (9.27) by complete backward induction on $k$. By definition and (9.16), we have for $k = m$

$$\begin{aligned}\xi_{0,m} &= \xi_{1,m}(r\zeta_* I - S^p)^{-1}s^p\alpha^c + r\zeta_*\phi_{m,m}\xi_{1,m}(r\zeta_* I - S^p)^{-1}\mathbf{1}\alpha^c \\ &= \lambda p_{0,m}(r) + q\lambda_{c,m}(r)p_m^m(r).\end{aligned}$$

Suppose now that $k < m$ and that (9.27) holds for all $k' \in \{k+1, \ldots, m\}$. We have by definition

$$\begin{aligned}\xi_{0,k} = {}& \xi_{1,k}(r\zeta_* I - S^p)^{-1}s^p\alpha^c + r\zeta_* \sum_{d=k}^{m} \phi_{d,k}\xi_{1,d}(r\zeta_* I - S^p)^{-1}\mathbf{1}\alpha^c \\ &+ \xi_{0,k+1}(r\zeta_* I - S^c)^{-1}s^c\alpha^c + r\zeta_* \sum_{d=k+1}^{m} \xi_{0,d}(r\zeta_* I - S^c)^{-1}(\psi_{d-1,d-k} + \psi_{d-1,k}\mathbf{1}\alpha^c).\end{aligned}$$

By induction hypothesis, (9.16) and (9.25) this is equal to

$$\begin{aligned}&\lambda p_{1,k}(r)(rqI - S^p)^{-1}s^p\alpha^c + \lambda rq\sum_{d=k}^{m} \phi_{d,k}p_{1,d}(r)(rqI - S^p)^{-1}\mathbf{1}\alpha^c \\ &+ \left(\lambda p_{0,k+1}(r) + q\sum_{j=k+1}^{m} \lambda_{c,j}(r)p_{k+1}^j(r)\right)(rqI - S^c)^{-1}s^c\alpha^c \\ &+ rq\sum_{d=k+1}^{m}\left(\lambda p_{0,d}(r) + q\sum_{i=d}^{m}\lambda_{c,i}(r)p_d^i(r)\right)(rqI - S^c)^{-1}(\psi_{d-1,d-k} + \psi_{d-1,k}\mathbf{1}\alpha^c).\end{aligned}$$

By first using the formula for $p_{0,k}(r)$ and then for $\lambda_{c,k}(r)$ (9.6) this is further equal to

$$\begin{aligned}&\lambda p_{0,k}(r) + \lambda rq\sum_{d=k}^{m}\phi_{d,k}p_{1,d}(r)(rqI - S^p)^{-1}\mathbf{1}\alpha^c \\ &+ q\sum_{j=k+1}^{m}\lambda_{c,j}(r)p_{k+1}^j(r)(rqI - S^c)^{-1}s^c\alpha^c + \lambda rq\sum_{d=k+1}^{m}\psi_{d-1,k}p_{0,d}(r)(rqI - S^c)^{-1}\mathbf{1}\alpha^c \\ &+ rq^2\sum_{i=k+1}^{m}\lambda_{c,i}(r)\sum_{d=k+1}^{i}p_d^i(r)(rqI - S^c)^{-1}(\psi_{d-1,d-k} + \psi_{d-1,k}\mathbf{1}\alpha^c) \\ &= \lambda p_{0,k}(r) + q\lambda_{c,k}(r)\alpha^c + q\sum_{j=k+1}^{m}\lambda_{c,j}(r)p_{k+1}^j(r)(rqI - S^c)^{-1}s^c\alpha^c\end{aligned}$$

$$+ rq^2 \sum_{i=k+1}^{m} \lambda_{c,i}(r) \sum_{d=k+1}^{i} \psi_{d-1,d-k} p_d^i(r)(rqI - S^c)^{-1}.$$

By rearranging the terms and by using the formula for $p_k^j(r)$ this equals

$$\begin{aligned} &\lambda p_{0,k}(r) + q\lambda_{c,k}(r)p_k^k(r) \\ &+ q \sum_{j=k+1}^{m} \lambda_{c,j}(r) \left( p_{k+1}^j(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{d=k+1}^{j} \psi_{d-1,d-k} p_d^j(r)(rqI - S^c)^{-1} \right) \\ &= \lambda p_{0,k}(r) + q \sum_{j=k}^{m} \lambda_{c,j}(r) p_k^j(r), \end{aligned}$$

which shows (9.27), thus finishing the proof. □

**Theorem 9.8.5.** *The stationary distribution $\pi(r)$ of the QBD Markov chain characterized by $Q(r)$ is the unique fixed point $\zeta$ of the set of ODEs in Equations* (9.9)-(9.11).

*Proof.* Using Proposition 9.3.3 we show that the fixed point equations $\frac{d}{dt}\vec{f}_\ell(t) = 0$ are equivalent to the balance equations of the QBD Markov chain characterized by $Q(r)$. The uniqueness of the fixed point follows from the uniqueness of the stationary distribution of the Markov chain.
For $\ell \geq 1$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ can be written as

$$0 = \vec{\zeta}_{\ell-1}(\lambda I) + \vec{\zeta}_\ell \tilde{A}_0 + \vec{\zeta}_{\ell+1}(\mu\alpha + r\zeta_* V_0),$$

where $\tilde{A}_0$ is the same matrix as $A_0(r)$ except with every instance of $q$ changed to $\zeta_*$. This is exactly the balance equations of $Q(r)$ for $\ell \geq 1$ as $\zeta_* = q$ due to Proposition 9.8.4. This implies that $\vec{\zeta}_\ell = \vec{\zeta}_0 R(r)^\ell$, for all $\ell \geq 1$ for any fixed point.
For $\ell = 0$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ implies

$$0 = \vec{\zeta}_0 \tilde{B}_0 + \vec{\zeta}_1(\mu\alpha + r\zeta_* V_0) + \lambda\zeta_*\alpha + r\zeta_* \sum_{\ell' \geq 0} \vec{\zeta}_{\ell'} T + r\zeta_* \sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0 \alpha,$$

where $\tilde{B}_0$ is the same matrix as $B_0(r)$ except with every instance of $q$ changed to $\zeta_*$. Due to Proposition 9.8.4 we can rewrite this as

$$0 = \vec{\zeta}_0 B_0(r) + \vec{\zeta}_1 A_{-1}(r) + q \left( \sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + \lambda\alpha + r \sum_{\ell \geq 1} \vec{\zeta}_\ell v_0 \alpha \right).$$

This implies that

$$\vec{\zeta}_0 = -q \left( \sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + \lambda\alpha + r \sum_{\ell \geq 1} \vec{\zeta}_\ell v_0 \alpha \right) (B_0(r) + \lambda I G(r))^{-1},$$

as $\lambda I G(r) = R(r) A_{-1}(r)$. As $\sum_{\ell \geq 0} \vec{\zeta}_\ell \mathbf{1} = 1 - q = \sum_{\ell \geq 0} \pi_\ell(r)\mathbf{1}$, we find that

$$r \sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0 = \lambda_p(r) \tag{9.28}$$

defined in (9.8). This indicates that $\frac{d}{dt}\vec{f}_\ell(t) = 0$ corresponds to the balance equation for $\ell = 0$. As $T\mathbf{1} = \mathbf{1} - v_0$, we have for $\frac{d}{dt}f_*(t) = 0$ that

$$\begin{aligned} 0 &= -\lambda\zeta_* + \vec{\zeta}_0\mu - r\zeta_*(1 - \zeta_* - \vec{\zeta}_0 v_0) \\ &= -\zeta_*\left(r\sum_{\ell'\geq 0}\vec{\zeta}_{\ell'}T\mathbf{1} + \lambda + r\sum_{\ell'\geq 1}\vec{\zeta}_{\ell'}v_0\right) + \vec{\zeta}_0\mu, \end{aligned}$$

which is exactly the first balance equation due to Proposition 9.8.4 and (9.28). □

## 9.9 Conclusions and future work

We introduced a model for randomized work stealing in multithreaded large-scale systems, where parent jobs spawn child jobs and where any number of existing child jobs can be stolen from a queue by a single probe. We defined a Quasi-Birth-Death (QBD) Markov chain to approximate the system behaviour and showed, using simulation, that the approximation error tends to zero as the number of servers tends to infinity. To further support this observation we introduced a mean field model and showed that the stationary distribution of the QBD is the unique fixed point of the mean field model.

Using numerical experiments we examined the effect of changing the load $\rho$, the steal rate $r$ and the variability of the job sizes. We studied the performance of some basic steal strategies and showed that stealing half of the child jobs is in general a good policy for higher steal rates $r$ and/or lower loads $\rho$, while the strategy of stealing all children performs best for low steal rates $r$ and/or higher loads. We further showed that stealing becomes more and more worthwhile when the job size variability increases.

Possible generalizations include stealing multiple parent jobs per probe and systems where offspring of a job can spawn further offspring (multigenerational multithreading).

# Part IV

# Hyperscalable Policies

Chapter 10

# Join-Up-To($m$): Improved Hyper-scalable Load Balancing

*This chapter is based on the paper* [42]*, which was submitted to a journal in 2023. The chapter is comprised of the paper together with results in case of exponential jobs and jobs of the phase type. Most of these results are not contained in* [42]*. As of writing this thesis, this is the last paper I have worked on.*

## 10.1 Introduction

Load balancing plays a crucial role in any large-scale distributed system. If the dispatcher responsible for assigning jobs to servers has perfect knowledge of the servers that are idle, then it is intuitively clear that the waiting time of jobs vanishes as the number of servers tends to infinity when the system load is below 1. This can be achieved using a simple algorithm called Join-the-Idle-Queue, where a server informs the dispatcher whenever it becomes idle [52, 72] and incoming jobs are assigned to idle servers (if there is at least one idle server). It is also clear that in such case the communication overhead is one message per job. Vanishing waiting times have also been established in the so-called hyper-scalable regime [3, 75], that is, when the communication overhead is below one message per job, but this is not possible without information regarding the job sizes [21].

Hyper-scalable load balancers that do not require any job size information have been studied in [74] and [36]. These do not have vanishing waits when fewer than one message per job is used, but instead have bounded queue lengths in the large-scale limit. However as the number of messages per job tends to zero, this upper bound as well as the mean response time tend to infinity. This means that these load balancers perform worse than simply assigning jobs to a random server (which requires no communication overhead at all) when the number of messages per job becomes small as demonstrated in Table 10.1. For instance, for a load of 0.8 we see that both the asynchronous policy of [74] and the pull policy of [36] (with $\delta_1 = 0$, meaning only idle servers send updates) perform far worse than random assignment when $\delta = 1/40$.

In this chapter we propose a new hyper-scalable load balancing scheme called Join-Up-To($m$) that outperforms random assignment irrespective of the communication overhead (and coincides with random assignment when the communication overhead tends to

| policy | $\delta$ | $\lambda = 0.5$ | $\lambda = 0.8$ | $\lambda = 0.95$ |
|---|---|---|---|---|
| Random assignment | 0 | 2 | 5 | 20 |
| Asynchr. policy [74] | 4/10 | 1.5932 | 3.1489 | 6.4542 |
| | 1/10 | 4.3563 | 10.6507 | 22.7754 |
| | 1/40 | 16.0860 | 41.3408 | 87.4644 |
| Pull policy, $\delta_1 = 0$ [36] | 4/10 | 1.0968 | 1.5 | 1.6540 |
| | 1/10 | 3 | 4.5 | 5.0671 |
| | 1/40 | 10.5 | 16.5 | 19.4944 |
| JUT($m_{opt}$), this chapter | 4/10 | 1.2170 | 1.5451 | 1.7287 |
| | 1/10 | 1.6886 | 2.7712 | 3.6549 |
| | 1/40 | 1.9069 | 3.9658 | 6.9149 |

Table 10.1: Mean response time of some existing hyperscalable policies for exponential job sizes with mean 1. All policies use on average $\delta/\lambda < 1$ messages per job.

zero). Join-Up-To($m$) is also superior to the schemes considered in [74] and [36], unless the communication overhead is fairly close to one message per job (see Table 10.1). Under the Join-Up-To($m$) policy the dispatcher maintains an upper bound on the queue length of each server and idle servers occasionally inform the dispatcher about their state (see Section 10.2 for details). Incoming jobs are assigned to a server with the lowest upper bound below $m$, if such a server exists, and are assigned at random otherwise.

To understand the system behavior when the number of servers tends to infinity, we study the queue at the cavity for the Join-Up-To($m$) policy (see Section 10.3). We derive explicit expressions for the mean and the variance of the response time, for the parameter value of $m$ that minimizes the mean response time and we derive expressions for the generating function and Laplace transform of the queue length distribution and response time distribution, respectively. These results show that the mean response time and optimal $m$ value for the Join-Up-To($m$) policy only depend on the first two moments of the jobs size distribution, while the variance of the response time also depends on the third moment. We also analytically invert the generating function of the queue length distribution in case of phase-type distributed job sizes and analytically invert the Laplace transform of the response time in case of exponential job sizes.

The chapter is structured as follows. In Section 10.2 we describe the system under consideration and introduce the JUT($m$) policy. The queue at the cavity approach, used to analyse the performance of JUT($m$) in a large-scale setting, is discussed in Section 10.3. The analysis of the queue at the cavity is presented in Section 10.4, while Section 10.5 contains various numerical results. Finally, conclusions can be found in Section 10.6.

## 10.2 System description and the JUT($m$) policy

We consider a set of $N$ homogeneous servers, each with its own infinite buffer, and a central dispatcher. Every server processes the jobs in its queue in First-Come-First-Served order. Jobs arrive at the dispatcher according to a Poisson process with rate $N\lambda$, with $0 < \lambda < 1$. The service requirements of a job have a general distribution $G$ with mean one, i.e. $E[G] = 1$. For each server the dispatcher maintains an upper bound on its queue length. Henceforth, we refer to these upper bounds as "estimates".

**The policy:** The load balancing policy called Join-Up-To($m$) (JUT($m$)) relies on a single integer parameter $m$ and operates as follows. When an arrival occurs and some servers have an estimate below $m$, then the dispatcher assigns the job to a server with lowest estimate among all such servers (with ties broken uniformly at random). Otherwise, if all estimates are at least $m$, the dispatcher assigns the job to a random server. Whenever the dispatcher assigns a job to a server, it increases this estimate by one. The estimate of a server can also be reset to zero. This happens when an idle server informs the dispatcher that its queue is empty. In order to have an average of $\delta/\lambda < 1$ such messages per arrival, idle server inform the dispatcher about their state at rate $\delta_0 = \delta/(1-\lambda)$ as $1-\lambda$ is the fraction of time that a server is idle.

**The parameter $m$:** When $m > \lfloor \lambda/\delta \rfloor$ the performance of the JUT($m$) policy coincides with the pull policy in [36] (with $\delta_1 = 0$) when the number of servers tends to infinity (as the dispatcher never runs out of servers with an estimate below $m$). Recall that this policy becomes inferior to random assignment for $\delta$ small enough. We therefore focus on JUT($m$) with $m \leq \lfloor \lambda/\delta \rfloor$. In Corollary 10.4.8 we present a simple explicit expression that depends only on $\lambda$, $\delta$ and $E[G^2]$ for the value of $m$ that minimizes the mean response time (for the queue at the cavity with $E[G] = 1$).

## 10.3 Queue at the cavity approach

As the system of $N$ servers is hard to analyze directly and simulation experiments do not provide closed form expressions and become very time consuming for large $N$, we make use of the so-called "queue at the cavity approach" (see [9] or Section 4.3). The basic idea of this approach is to focus on the evolution of a single server and to assume that all other servers have independent and identically distributed queue lengths. In some particular cases the queue at the cavity method was proven to yield exact results as the number of servers tends to infinity (see [9, 67]). The system that is closest to ours for which such a proof was established is [2] which was limited to exponential job sizes. As proving asymptotic exactness of the queue at the cavity approach is highly challenging for general service times, we limit ourselves to presenting a set of simulation results that suggest that the queue at the cavity also provides exact results as $N$ tends to infinity in our setting.

In Table 10.2, we compare the relative error of Corollary 10.4.4 for the queue at the cavity with the simulated mean response time, for $N \in \{10^2, 10^3, 10^4, 10^5\}$, based on 20 runs. Each run contains $1000N$ arrivals and has a warm-up period of 10%. We consider different values of $\lambda$, $\delta$, $m$ and different job size distributions. The job size distributions considered in Table 10.2 are exponential, Erlang, hyperexponential (HypExp), hyper-Erlang (HypErl) and truncated Pareto, all with mean 1. The hyperexponential distribution of order 2 is described using the shape parameter $f$ and the squared coefficient of variation $SCV$ [34]. The hyper-Erlang distribution HypErl($k, \ell$) is such that jobs are Erlang-$k$ with probability $p$ and Erlang-$\ell$ otherwise. The truncated Pareto distribution is characterized by three values: $\alpha, L$ and $U$, with $0 < L < U < +\infty$. The value of $\alpha$ is called the shape parameter, while $L$ and $U$ respectively denote the lower and upper bound of the support of the distribution. Note, that we require that jobs have mean 1. Hence, if $X$ has a Pareto

| settings | N | sim. ± conf. | rel.err.% |
|---|---|---|---|
| Exponential | 100 | 5.3145 ± 7.28e-03 | 2.4216 |
| $\lambda = 0.8$ | 1000 | 5.4347 ± 1.75e-03 | 0.2147 |
| $\delta = 0.05$ | 10000 | 5.4450 ± 5.95e-04 | 0.0268 |
| $m = 10$ | 100000 | 5.4463 ± 1.38e-04 | 0.0021 |
| | $\infty$ | 5.4464 | 0 |
| HypExp(2) | 100 | 11.7581 ± 5.55e-01 | 11.6331 |
| $f = 1/2, SCV = 15$ | 1000 | 10.6572 ± 2.14e-01 | 1.1811 |
| $\lambda = 0.95$ | 10000 | 10.5240 ± 4.34e-02 | 0.0836 |
| $\delta = 0.05$ | 100000 | 10.5349 ± 1.77e-02 | 0.0195 |
| $m = 15$ | $\infty$ | 10.5328 | 0 |
| Erlang(10) | 100 | 3.4231 ± 9.73e-03 | 1.7598 |
| $\lambda = 0.95$ | 1000 | 3.3728 ± 2.69e-03 | 0.2661 |
| $\delta = 0.1$ | 10000 | 3.3651 ± 1.17e-03 | 0.0378 |
| $m = 5$ | 100000 | 3.3640 ± 3.58e-04 | 0.0046 |
| | $\infty$ | 3.3639 | 0 |
| HypErl(3,7) | 100 | 2.0574 ± 3.32e-03 | 2.0553 |
| $p = 0.85$ | 1000 | 2.0960 ± 6.11e-04 | 0.2186 |
| $\lambda = 0.9$ | 10000 | 2.1005 ± 2.61e-04 | 0.0053 |
| $\delta = 0.05$ | 100000 | 2.1006 ± 7.44e-05 | 0.0001 |
| $m = 7$ | $\infty$ | 2.1006 | 0 |
| Pareto$(3, [1, 50])$ | 100 | 4.4010 ± 1.09e-02 | 0.2114 |
| $\lambda = 0.9$ | 1000 | 4.3940 ± 2.13e-03 | 0.0522 |
| $\delta = 0.05$ | 10000 | 4.3914 ± 8.19e-04 | 0.0088 |
| $m = 7$ | 100000 | 4.3918 ± 3.67e-04 | 0.0002 |
| | $\infty$ | 4.3917 | 0 |

Table 10.2: Relative error of the simulated mean response time for the JUT($m$) strategy based on 20 runs.

distribution with parameters $\alpha, L$ and $U$, we instead work with the random variable $X/E[X]$ and denote $X/E[X] \sim$ Pareto$(\alpha, [L, U])$.

The simulation results presented in Table 10.2 are a representative subset of simulations which we executed. When $N \geq 1000$ the error always stays below 1.5%. Further, in all cases the simulated mean response time seems to be $O(1/N)$ accurate in function of $N$, similar to the results in [24].

## 10.4 Analysis of the queue at the cavity

### 10.4.1 General job sizes

The queue at the cavity for the JUT($m$) policy is defined as an M/G/1 queue with arrival rate $\tilde{\lambda} = \lambda - \delta m$, except that when the queue is empty there are also batch arrivals of size $m$ that occur at rate $\delta_0 = \delta/(1 - \lambda)$. The intuition behind this queue at the cavity is that in the large-scale limit any idle server that informs the dispatcher about its state (which occurs at rate $\delta_0$) will immediately receive a batch of $m$ jobs. As the overall rate of such messages is $\delta$, this implies that a fraction $\delta m/\lambda$ of the jobs is assigned in this manner. The remaining fraction $1 - \delta m/\lambda$ of the jobs is assigned at random and corresponds to the arrivals at rate $\tilde{\lambda} = \lambda - \delta m$. Recall that we assume that $m < \lambda/\delta$, such that $\tilde{\lambda} > 0$. As $m$ is an integer, we have $m \leq \lfloor \lambda/\delta \rfloor$.

Let $\pi_i^a, \pi_i^d$ and $\pi_i$ be the steady state probability that there are $i$ jobs in the cavity queue at arrival, departure and at a random time, respectively. Let $\pi^a(z), \pi^d(z)$ and $\pi(z)$ be the associated generating functions. Note that if a job is the $k$-th job of a batch of size $m$, then it sees $k-1$ jobs at arrival time. It is well known that $\pi^a(z) = \pi^d(z)$. This is clear for single arrivals and the corresponding departures. This also holds for the batch arrivals in state 0 and the corresponding departures: the $k$-th job from a batch of $m$ jobs sees $k-1$ jobs in the queue upon arrival, namely the $k-1$ jobs from the same batch that came before the $k$-th job. The next theorem relates $\pi(z)$ with $\pi^a(z)$.

**Theorem 10.4.1.** *The generating function $\pi(z)$ can be written as*

$$\pi(z) = \frac{\lambda}{\tilde{\lambda}}\pi^a(z) - \frac{\delta}{\tilde{\lambda}}\frac{1-z^m}{1-z}. \tag{10.1}$$

*Proof.* As any batch arrival that occurs when the queue is empty contains a single job that sees $i$ jobs at arrival time, for $0 \leq i < m$, we have the following relationship between the probabilities $\pi_i^a$ and $\pi_i$:

$$\pi_i^a = \left(\tilde{\lambda}\pi_i + \delta_0\pi_0\right)c,$$

for $0 \leq i < m$, where $c$ is a normalizing constant. Further, as batch arrivals only occur when the queue is empty, we have

$$\pi_i^a = \tilde{\lambda}\pi_i c,$$

for $i \geq m$. As $\pi^a(1) = 1$ and $\pi(1) = 1$, we have

$$1/c = \tilde{\lambda} + \delta_0\pi_0 m = \tilde{\lambda} + \delta m = \lambda,$$

as $\pi_0 = 1 - \lambda$ and $\delta_0 = \delta/(1-\lambda)$. This implies that $\pi_i^a = (\tilde{\lambda}\pi_i + \delta)/\lambda$ for $i < m$ and $\pi_i^a = \tilde{\lambda}\pi_i/\lambda$ for $i \geq m$. Hence,

$$\pi^a(z) = \frac{\delta}{\lambda}\sum_{i=0}^{m-1} z^i + \frac{\tilde{\lambda}}{\lambda}\pi(z) = \frac{\delta}{\lambda}\frac{1-z^m}{1-z} + \frac{\tilde{\lambda}}{\lambda}\pi(z),$$

and therefore (10.1) holds. □

Recall that for an $M/G/1$ queue with arrival rate $\tilde{\lambda}$ and mean job size 1, the generating function of the queue length is given by the Pollaczek-Khinchine formula (cf. Theorem 3.5.3)

$$\xi(z) = \frac{(1-\tilde{\lambda})(1-z)G^*(\tilde{\lambda} - \tilde{\lambda}z)}{G^*(\tilde{\lambda} - \tilde{\lambda}z) - z}, \tag{10.2}$$

where $G^*(s)$ is the Laplace-Stieltjes transform of the job size distribution.

**Theorem 10.4.2.** *The generating function $\pi^a(z)$ can be written as*

$$\pi^a(z) = \frac{1-\alpha(z)}{\alpha'(1)(1-z)}\xi(z), \tag{10.3}$$

*where $\xi(z)$ is given by* (10.2) *and*

$$\alpha(z) = \frac{\tilde{\lambda}}{\tilde{\lambda}+\delta_0}z + \frac{\delta_0}{\tilde{\lambda}+\delta_0}z^m.$$

*Note that $\alpha'(1) = (\tilde{\lambda}+m\delta_0)/(\tilde{\lambda}+\delta_0)$.*

*Proof.* Consider an M/G/1 queue with arrival rate $\tilde{\lambda}$ where the server starts a vacation each time the queue becomes empty. The vacation ends with probability $\tilde{\lambda}/(\tilde{\lambda}+\delta_0)$ when an arrival occurs or ends when the $m$-th arrival occurs otherwise. The queue length distribution of this vacation queue is the same at arrival, departure and at random times (due to PASTA) and its generating function $\phi(z)$ obeys the well known decomposition result for vacation queues [19,20], that is,

$$\phi(z) = \frac{1-\alpha(z)}{\alpha'(1)(1-z)}\xi(z),$$

where $\xi(z)$ is the generating function of the queue length of a standard M/G/1 queue with arrival rate $\tilde{\lambda}$ and $\alpha(z)$ is the generating function of the number of arrivals during a vacation.

The proof completes by noting that the queue length distribution at departure times in the queue at the cavity $\pi^d(z)$ and at departure times in the vacation queue $\phi(z)$ are the same, while $\pi^a(z) = \pi^d(z)$. □

**Corollary 10.4.3.** *The generating function $\pi(z)$ is given by*

$$\pi(z) = \frac{\lambda}{\tilde{\lambda}}\beta(z)\xi(z) - \frac{\delta}{\tilde{\lambda}}\frac{1-z^m}{1-z}, \tag{10.4}$$

*with $\beta(z) = (1-\alpha(z))/(\alpha'(1)(1-z))$.*

*Proof.* This is immediate from the previous two theorems. □

Note that $\beta(z)$ is the generating function of the number of customers that arrive during a vacation period after the arrival of the random customer during a vacation [19]. Using this interpretation we note that

$$\beta(z) = \frac{\tilde{\lambda}}{\delta_0 m+\tilde{\lambda}} + \frac{\delta}{\lambda(1-\tilde{\lambda})}\sum_{i=0}^{m-1} z^i, \tag{10.5}$$

as $\delta_0 m/(\delta_0 m+\tilde{\lambda}) = \delta m/(\lambda(1-\tilde{\lambda}))$ is the probability that the random arrival is part of a vacation with $m$ arrivals and its position is uniform within these $m$ arrivals.

**Corollary 10.4.4.** *The mean response time $E[R(m)]$ in the queue at the cavity is given by*

$$E[R(m)] = 1 + \frac{\tilde{\lambda}E[G^2]}{2(1-\tilde{\lambda})} + \frac{\delta}{\lambda}\frac{m(m-1)}{2(1-\tilde{\lambda})}. \qquad (10.6)$$

*In particular, we have*

$$\lim_{\lambda \to 1^-} E[R(m)] = \frac{1-\delta m}{2\delta m}E[G^2] + \frac{m+1}{2}. \qquad (10.7)$$

*Proof.* Due to Little, we have $E[R(m)] = \pi'(1)/\lambda$. Using (10.4) we have

$$\frac{\pi'(1)}{\lambda} = \frac{\beta'(1)}{\tilde{\lambda}} + \frac{\xi'(1)}{\tilde{\lambda}} - \frac{\delta}{\tilde{\lambda}}\frac{m(m-1)}{2}\frac{1}{\lambda},$$

where $\xi'(1)/\tilde{\lambda}$ is the mean response time in a standard M/G/1 queue with arrival rate $\tilde{\lambda}$, which equals $1 + \tilde{\lambda}E[G^2]/(2(1-\tilde{\lambda}))$ as $E[G] = 1$. The first claim therefore follows by verifying that

$$\beta'(1) = \frac{\delta m(m-1)}{2\lambda(1-\tilde{\lambda})},$$

which is immediate from (10.5). The second claim follows immediately from (10.6) as $\tilde{\lambda}$ converges to $1 - \delta m$. □

The formula (10.6) can also be obtained directly through the use of mean value analysis:

*Proof.* We tag an arbitrary job that just entered the queue. There are two possibilities: the tagged job arrived in the queue due to the Poisson process with parameter $\tilde{\lambda}$ or the tagged job arrived in a batch of $m$ jobs due to an update event. Let us call these jobs of type 1 and type $m$ and denote by $tag_1$ and $tag_m$ the case where a tagged job is of type 1 and type $m$ respectively. We have

$$E[W] = E[W|tag_1]P[tag_1] + E[W|tag_m]P[tag_m].$$

Jobs of type 1 arrive at the queue with rate $\tilde{\lambda}$, while jobs of type $m$ arrive at rate $\delta_0(1-\lambda)m = \delta m$. The total arrival rate of jobs is $\lambda$, hence $P[tag_1] = \tilde{\lambda}/\lambda$ and $P[tag_m] = \delta m/\lambda$.
There is a $1/m$ chance that a tagged job of type $m$ is the first, second, ..., $m$-th job of the batch. Hence

$$E[W|tag_m] = \frac{1}{m}\sum_{k=0}^{m-1} k = \frac{m-1}{2}.$$

Finding $E[W|tag_1]$ requires more work. Due to the PASTA property we have

$$E[W|tag_1] = E[N] + E[t_{res}] = E[N] + E[t_{res}|\text{ queue busy}]P[\text{queue busy}],$$

where $N$ denotes the number of waiting jobs and $t_{res}$ the residual service time of the job in service. Note that if the queue is empty, then $t_{res} = 0$. We have $E[N] = \lambda E[W]$, due to Little's law. Further $E[t_{res}] = E[G^2]/2$ and as jobs have mean one $P[\text{queue busy}] = \lambda$. Putting everything together we have:

$$E[W] = \left(\lambda E[W] + \lambda\frac{E[G^2]}{2}\right)\frac{\tilde{\lambda}}{\lambda} + \frac{\delta m}{\lambda}\cdot\frac{m-1}{2}.$$

Solving this equation for $E[W]$, we get

$$E[W] = \frac{\tilde{\lambda}}{1-\tilde{\lambda}} \cdot \frac{E[G^2]}{2} + \frac{\delta}{\lambda} \cdot \frac{1}{1-\lambda+\delta m} \cdot \frac{(m-1)m}{2}.$$

The claim now follows as $\tilde{\lambda} = \lambda - \delta m$ and $E[R(m)] = 1 + E[W]$. □

*Remark* 10.4.5. As long as $m > 0$ and $\delta > 0$, the mean response time remains bounded when $\lambda$ tends to one, in contrast to an ordinary M/G/1 queue. The mean response time of JUT($m$) converges to that of random assignment when $\delta$ tends to zero, which is an improvement over the policies in [36,74] where the mean response time tends to infinity as $\delta$ tends to zero for any $\lambda < 1$.

**Theorem 10.4.6.** *The Laplace transform $R^*(s)$ of the response time distribution of the queue at the cavity can be expressed as*

$$R^*(s) = \frac{\tilde{\lambda}}{\lambda} Y^*(s)\pi(G^*(s)) - \frac{\tilde{\lambda}(1-\lambda)}{\lambda}(Y^*(s) - G^*(s)) + \frac{\delta}{\lambda}\left(\frac{1-G^*(s)^{m+1}}{1-G^*(s)} - 1\right), \tag{10.8}$$

*where $G^*(s)$ and $Y^*(s)$ are the Laplace transforms of the service time and residual service time, respectively. It is also well known that $Y^*(s) = (1-G^*(s))/s$ as $E[G] = 1$.*

*Proof.* The arrivals that occur at rate $\tilde{\lambda}$ arrive at random points in time, therefore such an arrival sees a workload of one residual service time and $i-1$ service times with probability $\pi_i$, for $i > 0$ and no workload with probability $\pi_0$. For a tagged arrival that occurs in a batch of size $m$ the workload observed upon arrival is equal to the service time of the jobs that are part of the same batch and ahead of the tagged arrival. This implies

$$\begin{aligned} R^*(s) &= \frac{\tilde{\lambda}}{\lambda}\left(\sum_{i=1}^{\infty} \pi_i Y^*(s) G^*(s)^i + \pi_0 G^*(s)\right) + \frac{\delta m}{\lambda}\sum_{i=0}^{m-1}\frac{1}{m}G^*(s)^{i+1} \\ &= \frac{\tilde{\lambda}}{\lambda}\left(Y^*(s)(\pi(G^*(s)) - \pi_0) + \pi_0 G^*(s)\right) + \frac{\delta}{\lambda}\left(\frac{1-G^*(s)^{m+1}}{1-G^*(s)} - 1\right). \end{aligned}$$

Equation (10.8) then follows as $\pi_0 = 1 - \lambda$. □

*Remark* 10.4.7. We can also retrieve (10.6) using (10.8) by making use of the fact that $E[R(m)] = -R^{*\prime}(0)$. More specifically, we can make use of the equalities $G^{*\prime}(0) = -E[G] = -1$ and $-Y^{*\prime}(0) = E[Y] = E[G^2]/2$ to find that

$$\begin{aligned} E[R(m)] &= -R^{*\prime}(0) \\ &= -\frac{\tilde{\lambda}}{\lambda}\left(Y^{*\prime}(0) - \pi'(1) - (1-\lambda)(Y^{*\prime}(0)+1)\right) + \frac{\delta}{\lambda}\frac{m(m+1)}{2} \\ &= \tilde{\lambda}\frac{\pi'(1)}{\lambda} - \tilde{\lambda}Y^{*\prime}(0) + \frac{\tilde{\lambda}(1-\lambda)}{\lambda} + \frac{\delta m}{\lambda} + \frac{\delta}{\lambda}\frac{m(m-1)}{2} \\ &= \tilde{\lambda}E[R(m)] + \frac{\tilde{\lambda}E[G^2]}{2} + (1-\tilde{\lambda}) + \frac{\delta}{\lambda}\frac{m(m-1)}{2}, \end{aligned}$$

as $\delta m + \tilde{\lambda} = \lambda$. Similarly, we can derive an explicit expression for the second moment of the response time $E[R(m)^2]$, see Theorem 10.4.11.

**Theorem 10.4.8.** *The mean response time of the queue at the cavity $E[R(m)]$ is minimized by setting $m$ equal to $m_{opt} = \min(\hat{m}, \lfloor \lambda/\delta \rfloor)$ with*

$$\hat{m} = \left\lceil \sqrt{\left(\frac{1}{2} + \frac{1-\lambda}{\delta}\right)^2 + \frac{\lambda}{\delta}E[G^2]} - \left(\frac{1}{2} + \frac{1-\lambda}{\delta}\right) \right\rceil. \tag{10.9}$$

*Proof.* Using (10.6) we get that

$$0 = \frac{\partial E[R(m)]}{\partial m} = \frac{\delta(\delta m^2 + (2m-1)(1-\lambda) - \lambda E[G^2])}{2\lambda(1-\lambda+\delta m)^2}. \tag{10.10}$$

This equation has a unique positive root given by

$$m^* = \frac{\sqrt{(1-\lambda)^2 + \delta\left[1 + \lambda\left(E[G^2]-1\right)\right]} - (1-\lambda)}{\delta},$$

as $1 + \lambda(E[G^2]-1) > 0$. One readily verifies that

$$\frac{\partial^2 E[R(m)]}{\partial m^2} = \frac{\delta((1-\lambda)\delta + (1-\lambda)^2 + \lambda\delta E[G^2])}{\lambda(1-\lambda+\delta m)^3} \geq 0.$$

for $m \geq 0$. Therefore $E[R(m)]$ is convex in $m$ on $[0,\infty)$ and $m^*$ is the minimum of $E[R(m)]$. However $m^*$ is typically not an integer.

The integer value that minimizes $E[R(m)]$ is found by defining $\Delta_R(m) = E[R(m+1)] - E[R(m)]$ and taking the ceil of its unique positive root. By further using (10.6), one easily checks that

$$\Delta_R(m) = \frac{\delta m}{N\lambda}\left(1 - \lambda + \delta\frac{(m+1)}{2}\right) - \frac{\delta}{N}\cdot\frac{E[G^2]}{2},$$

where $N = (1-\lambda)^2 + (1-\lambda)\delta(2m+1) + \delta^2 m(m+1) > 0$. We have that $\Delta_R(m) = 0$ if and only if

$$\frac{m}{\lambda}\left(1 - \lambda + \frac{\delta(m+1)}{2}\right) - \frac{E[G^2]}{2} = 0,$$

which has a unique positive root given by

$$m = \frac{-\frac{\delta}{2\lambda} - \frac{1-\lambda}{\lambda} + \sqrt{\left(\frac{\delta}{2\lambda} + \frac{1-\lambda}{\lambda}\right)^2 + \frac{\delta}{\lambda}E[G^2]}}{\delta/\lambda}. \tag{10.11}$$

As $\tilde{\lambda} > 0$, we have $m \leq \lfloor \lambda/\delta \rfloor$ and the result follows by the convexity of $E[R(m)]$. □

*Remark:* The optimal value $m_{opt}$ is decreasing in $\delta$ and increasing in both $\lambda$ and $E[G^2]$. The next Corollary therefore implies that $m_{opt} \leq \lceil \lambda E[G^2]/(2(1-\lambda)) \rceil$.

**Corollary 10.4.9.** *The optimal value of $m$ when $\delta \to 0^+$ is given by*

$$m_{opt}^{\delta \to 0^+} = \left\lceil \frac{1}{2}\frac{\lambda}{1-\lambda}E[G^2] \right\rceil.$$

*The optimal value of $m$ when $\lambda \to 1^-$ is given by*

$$m_{opt}^{\lambda \to 1^-} = \min\left(\left\lceil \sqrt{\frac{1}{4} + \frac{E[G^2]}{\delta}} - \frac{1}{2} \right\rceil, \left\lfloor \frac{1}{\delta} \right\rfloor\right).$$

*Proof.* The first claim follows by application of l'Hôpital's rule on (10.9), the second is immediate. □

*Remark* 10.4.10. We have

$$\begin{aligned}\frac{\partial E[R(m)]}{\partial \delta} &= -\frac{m\left(1-\lambda+\lambda E[G^2]-(1-\lambda)m\right)}{2\lambda(1-\tilde{\lambda})^2} \\ &= -\frac{(1-\lambda)m\left(1+\frac{\lambda}{1-\lambda}E[G^2]-m\right)}{2\lambda(1-\tilde{\lambda})^2}. \qquad (10.12)\end{aligned}$$

We distinguish three possibilities:

1. If $m > 1+\frac{\lambda}{1-\lambda}E[G^2]$, then (10.12) is greater than 0, hence increasing the value of $\delta$ increases the mean response time. In this case, having $\delta = 0$ works the best. If $\delta = 0$, the queue at the cavity becomes a standard M/G/1 queue with arrival rate $\lambda$ and (10.6) simplifies to $1+\frac{\lambda}{1-\lambda}\frac{E[G^2]}{2}$.
2. If $m = 1+\frac{\lambda}{1-\lambda}E[G^2]$, then (10.12) is 0, which implies that in this case $E[R(m)]$ is independent of $\delta$. In fact, substituting $m = 1+\frac{\lambda}{1-\lambda}E[G^2]$ into (10.6) gives $E[R(m)] = 1+\frac{\lambda}{1-\lambda}\frac{E[G^2]}{2}$.
3. If $m < 1+\frac{\lambda}{1-\lambda}E[G^2]$, then (10.12) is smaller than 0 and the proposed policy works better than random assignment for any $\delta > 0$.

Note that when $m = 1$ or $m = m_{opt}$ (due to Corollary 10.4.9), case 3. applies and our policy improves upon random assignment.

We end this section with the derivation of a formula for the second (raw) moment $E[R(m)^2]$ of the response time, which combined with $E[R(m)]$ yields a formula for the variance $Var[R(m)]$.

**Theorem 10.4.11.** *The second (raw) moment of the response time of the queue at the cavity $E[R(m)^2]$ is given by*

$$\begin{aligned}E[R(m)^2] = &\frac{\tilde{\lambda}E[G^3]}{3} + \frac{\delta\tilde{\lambda}(m-2)(m-1)m}{3\lambda(1-\tilde{\lambda})} + \tilde{\lambda}\frac{\delta m(m-1)}{\lambda(1-\tilde{\lambda})}\left(1+\frac{\tilde{\lambda}E[G^2]}{2(1-\tilde{\lambda})}\right) \\ &+ \frac{\tilde{\lambda}^4E[G^2]^2}{2(1-\tilde{\lambda})^2} + \frac{\tilde{\lambda}^3E[G^3]}{3(1-\tilde{\lambda})} + \frac{\tilde{\lambda}^2E[G^2]}{1-\tilde{\lambda}} + 2\tilde{\lambda}E[G^2]\left(1+\frac{\tilde{\lambda}E[G^2]}{2(1-\tilde{\lambda})}+\frac{\delta}{\lambda}\frac{m(m-1)}{2(1-\tilde{\lambda})}\right) \\ &+ \frac{\tilde{\lambda}(1-\lambda)}{\lambda}E[G^2] + \frac{\delta}{\lambda}\left(E[G^2]\frac{m(m+1)}{2}+\frac{(m-1)m(m+1)}{3}\right).\end{aligned}$$

*Proof.* We have $E[R(m)^2] = R^{*\prime\prime}(0)$. By using (10.8) together with $G^{*\prime}(0) = -E[G] = -1$, $-Y^{*\prime}(0) = E[Y] = E[G^2]/2$ and $G^{*\prime\prime}(0) = E[G^2]$, we obtain

$$\begin{aligned}E[R(m)^2] = \tilde{\lambda}Y^{*\prime\prime}(0) &+ \frac{\tilde{\lambda}}{\lambda}\left(\pi''(1)+2\pi'(1)E[G^2]+(1-\lambda)E[G^2]\right) \\ &+ \frac{\delta}{\lambda}\left(E[G^2]\frac{m(m+1)}{2}+\frac{(m-1)m(m+1)}{3}\right).\end{aligned}$$

One readily checks that $Y^{*\prime\prime}(0) = E[G^3]/3$ and we already know that $\pi'(1) = \lambda E[R(m)]$. From (10.4) we have

$$\pi''(1) = \frac{\lambda}{\tilde{\lambda}} \left(\beta''(1) + 2\beta'(1)\xi'(1) + \xi''(1)\right) - \frac{\delta(m-2)(m-1)m}{3\tilde{\lambda}}.$$

Making use of (10.5) one finds

$$\beta''(1) = \frac{\delta}{\lambda(1-\tilde{\lambda})} \sum_{i=1}^{m-1} i(i-1) = \frac{\delta(m-2)(m-1)m}{3\lambda(1-\tilde{\lambda})}.$$

We still need to find $\xi''(1)$. $\xi''(1)$ can be calculated directly, however this calculation is rather lengthy. We therefore opt for a different approach. Denote respectively by $\tilde{R}$ and $\tilde{W}$ the response and waiting time of an ordinary $M/G/1$ queue with arrival rate $\tilde{\lambda}$. By using [30, (5.30)], we get $\xi''(1) = \tilde{\lambda}^2 E[\tilde{R}^2]$. We have $E[\tilde{R}^2] = E[(\tilde{W}+G)^2] = E[\tilde{W}^2] + 2E[\tilde{W}]E[G] + E[G^2]$. As $E[G] = 1$ and $E[\tilde{W}] = \frac{\tilde{\lambda}E[G^2]}{2(1-\tilde{\lambda})}$, we obtain $2E[\tilde{W}]E[G] + E[G^2] = E[G^2]/(1-\tilde{\lambda})$. $E[\tilde{W}^2]$ is given by [14, p.256]:

$$E[\tilde{W}^2] = \frac{\tilde{\lambda}^2 E[G^2]^2}{2(1-\tilde{\lambda})^2} + \frac{\tilde{\lambda}E[G^3]}{3(1-\tilde{\lambda})}.$$

It follows that

$$\xi''(1) = \frac{\tilde{\lambda}^4 E[G^2]^2}{2(1-\tilde{\lambda})^2} + \frac{\tilde{\lambda}^3 E[G^3]}{3(1-\tilde{\lambda})} + \frac{\tilde{\lambda}^2 E[G^2]}{1-\tilde{\lambda}}.$$

Putting everything together finishes the proof. □

### 10.4.2 Phase-type distributed job sizes

In this section we provide an explicit formula for the queue length distribution in case of phase type distributed jobs, meaning we present an explicit formula for the probabilities $\pi_k$ appearing in the generating function $\pi(z) = \sum_k \pi_k z^k$. Then, we present a way of deriving explicit formula for $\pi_{k,\ell}$, the probability that the queue has $k$ jobs and the job in service is in phase $\ell$.

Recall that a phase type distribution with $n_p$ phases can be characterized by a couple $(\alpha, S)$, where $\alpha$ is a row vector of length $n_p$ and is called the initial distribution vector, as $\alpha_i$ is the probability that the distribution starts in phase $i$; and where $S$ is a $n_p \times n_p$ matrix that records the rates of phase changes.

**Theorem 10.4.12.** *Suppose $G$ is PH$(\alpha, S)$ distributed (with mean 1). Then, for $k = 1, \ldots, m$:*

$$\pi_k = \left(1 - \lambda + \frac{\delta}{\tilde{\lambda}}\right) \alpha R^k 1_{n_p} + \frac{\delta}{\tilde{\lambda}} \alpha (I-R)^{-1} \left(R - R^k\right) 1_{n_p}, \qquad (10.13)$$

*and for $k > m$:*

$$\pi_k = \left[\left(1 - \lambda + \frac{\delta}{\tilde{\lambda}}\right) \alpha + \frac{\delta}{\tilde{\lambda}} \alpha (I-R)^{-1} \left(R^{1-m} - I\right)\right] R^k 1_{n_p}, \qquad (10.14)$$

*where $R = -\tilde{\lambda}(S - \tilde{\lambda} I + \tilde{\lambda} 1_{n_p} \alpha)^{-1}$ and where $1_{n_p}$ is a column vector of ones of height $n_p$.*

*Proof.* By using [60, Theorem 3.2.1], we get

$$\xi(z) = (1-\tilde{\lambda}) \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k.$$

Therefore, by (10.4), (10.5) and the fact that $\lambda(1-\tilde{\lambda})/(\delta_0 m + \tilde{\lambda}) = 1 - \lambda$, we have

$$\begin{aligned}
\pi(z) &= (1-\lambda) \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k + \frac{\delta}{\tilde{\lambda}} \sum_{i=0}^{m-1} z^i \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k - \frac{\delta}{\tilde{\lambda}} \sum_{i=0}^{m-1} z^i \\
&= (1-\lambda) \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k + \frac{\delta}{\tilde{\lambda}} \sum_{i=0}^{m-1} z^i \sum_{k=1}^{\infty} \alpha R^k 1_{n_p} z^k \\
&= (1-\lambda) \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k + \frac{\delta}{\tilde{\lambda}} \sum_{k=1}^{\infty} \alpha R^k 1_{n_p} z^k + \frac{\delta}{\tilde{\lambda}} \sum_{i=1}^{m-1} z^i \sum_{k=1}^{\infty} \alpha R^k 1_{n_p} z^k.
\end{aligned}$$

On the other hand, by using (10.13)-(10.14) and $\pi_0 = 1 - \lambda$, we get

$$\begin{aligned}
\sum_{k=0}^{\infty} \pi_k z^k &= (1-\lambda) \sum_{k=0}^{\infty} \alpha R^k 1_{n_p} z^k \\
&\quad + \frac{\delta}{\tilde{\lambda}} \sum_{k=1}^{\infty} \alpha R^k 1_{n_p} z^k + \frac{\delta}{\tilde{\lambda}} \sum_{k=1}^{m} \alpha (I-R)^{-1} \left(R - R^k\right) 1_{n_p} z^k \\
&\quad + \frac{\delta}{\tilde{\lambda}} \sum_{k=m+1}^{\infty} \alpha (I-R)^{-1} \left(R^{k-m+1} - R^k\right) 1_{n_p} z^k.
\end{aligned}$$

Hence, it suffices to show that

$$\sum_{i=1}^{m-1} z^i \sum_{k=1}^{\infty} R^k z^k = \sum_{k=1}^{m} (I-R)^{-1} \left(R - R^k\right) z^k + \sum_{k=m+1}^{\infty} (I-R)^{-1} \left(R^{k-m+1} - R^k\right) z^k. \tag{10.15}$$

The RHS of (10.15) equals

$$\sum_{k=2}^{m} z^k \sum_{\ell=1}^{k-1} R^\ell + \sum_{k=m+1}^{\infty} z^k \sum_{\ell=k-m+1}^{k-1} R^\ell, \tag{10.16}$$

while the LHS is equal to

$$\sum_{\ell=1}^{m-1} z^\ell \sum_{k=1}^{m-\ell} R^k z^k + \sum_{\ell=1}^{m-1} z^\ell \sum_{k=m+1-\ell}^{\infty} R^k z^k, \tag{10.17}$$

where the first sum of (10.17) consists of all terms with the exponent of $z$ smaller than or equal to $m$ and the second with exponents greater than $m$. The first sum of (10.17) equals

$$\sum_{\ell=1}^{m-1} \sum_{k=\ell+1}^{m} R^{k-\ell} z^k = \sum_{\ell=2}^{m} \sum_{k=\ell}^{m} R^{k-\ell+1} z^k$$

$$= \sum_{k=2}^{m} z^k \sum_{\ell=2}^{k} R^{k-\ell+1} = \sum_{k=2}^{m} z^k \sum_{\ell=1}^{k-1} R^{\ell},$$

which is the first sum of (10.16). Proceeding similarly with the second sum of (10.17), we get that it equals

$$\sum_{\ell=1}^{m-1} \sum_{k=m+1}^{\infty} R^{k-\ell} z^k = \sum_{k=m+1}^{\infty} z^k \sum_{\ell=k-m+1}^{k-1} R^{\ell},$$

which is the second sum of (10.16). This finishes the proof. □

We now derive an explicit formula for $\pi_{k,\ell}$, the probability that the queue at the cavity contains $k$ jobs and the job in service is in phase $\ell$. We call $k$ the level and $\ell$ the phase of the queue. Throughout the rest of this section we denote

$$\pi_k = \left[\pi_{k,1}, \pi_{k,2}, \ldots, \pi_{k,n_p}\right],$$

for $k > 0$. The behaviour of the cavity queue is exactly the same as that of an ordinary $M/PH/1$ queue with arrival rate $\tilde{\lambda}$, except that the state can jump from level 0 to level $m$ at rate $\delta_0\alpha$. If we order the states lexicographically, then the cavity queue can be described for $m > 1$ by a CTMC with rate matrix

$$\begin{bmatrix}
-(\tilde{\lambda}+\delta_0) & \tilde{\lambda}\alpha & & & & \delta_0\alpha & & \\
s^* & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} & & & & & \\
 & s^*\alpha & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} & & & & \\
 & & \ddots & \ddots & \ddots & & & \\
 & & & s^*\alpha & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} & & \\
 & & & & s^*\alpha & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} & \\
 & & & & & \ddots & \ddots & \ddots
\end{bmatrix}. \tag{10.18}$$

and for $m = 1$ by

$$\begin{bmatrix}
-(\tilde{\lambda}+\delta_0) & (\tilde{\lambda}+\delta_0)\alpha & & \\
s^* & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} & \\
 & s^*\alpha & S-\tilde{\lambda}I_{n_p} & \tilde{\lambda}I_{n_p} \\
 & & \ddots & \ddots & \ddots
\end{bmatrix}.$$

The matrix (10.18) can be seen as an example of the infinitesimal generator of an $M/G/1$-type process. Therefore, an invariant distribution for (10.18) can be obtained by the use of Ramaswami's formula [63]. As [63] dealt with DTMCs we opt to use the continuous analogue of Ramaswami's formula from [70]. In the notation from [70], we have:

$$\hat{L} = -(\delta_0 + \tilde{\lambda}), \quad \hat{B} = s^*, \quad \hat{F}^{(k)} = \begin{cases} \tilde{\lambda}\alpha, & \text{if } k = 1 \text{ and } m \neq 1 \\ \delta_0\alpha, & \text{if } k = m \text{ and } m \neq 1 \\ (\delta_0 + \tilde{\lambda})\alpha, & \text{if } k = m = 1 \\ 0, & \text{if } k \neq 1, m \end{cases} \tag{10.19}$$

and

$$L = S - \tilde{\lambda} I, \quad B = s^* \alpha, \quad F^{(k)} = \begin{cases} \tilde{\lambda} I, & k = 1 \\ 0, & k \neq 1 \end{cases}. \tag{10.20}$$

As our system is a QBD except in level 0, the first step in finding the invariant distribution is solving the following quadratic matrix equation for the matrix $G$:

$$s^* \alpha + (S - \tilde{\lambda} I) G + \tilde{\lambda} G^2 = 0. \tag{10.21}$$

The physical interpretation of the matrix $G$ is the following [48, proof of Theorem 6.4.1]: the $(i, j)$-th entry of $G$ is the probability that the QBD will first enter level $\ell - 1$ in phase $j$, given that it starts in phase $i$ of level $\ell$. One readily checks that $G = 1_{n_p} \alpha$ is therefore the needed solution of (10.21). This implies $G^k = 1_{n_p} \alpha$ and $\alpha G^{k'} = \alpha$ for every $k \geq 1$ and $k' \geq 0$. Continuing with the notation of [70], we have

$$\hat{S}^{(j)} = \begin{cases} (\tilde{\lambda} + \delta_0) \alpha, & j = 1 \\ \delta_0 \alpha, & 2 \leq j \leq m \\ 0, & j > m \end{cases} \quad \text{and} \quad S^{(j)} = \begin{cases} S + \tilde{\lambda}(1_{n_p} \alpha - I), & j = 0 \\ \tilde{\lambda} I, & j = 1 \\ 0, & j > 1. \end{cases} \tag{10.22}$$

Denote $\tilde{S} = S + \tilde{\lambda}(1_{n_p} \alpha - I)$. (In [70] this matrix is denoted as $S^{(0)}$. As the subsequent calculations contain the inverse of $S + \tilde{\lambda}(1_{n_p} \alpha - I)$, we prefer to denote it by $\tilde{S}$.)

*Remark* 10.4.13. We have $R = -\tilde{\lambda} \tilde{S}^{-1}$, with $R$ defined in Theorem 10.4.12.

*Proof.* We have $R = -\tilde{\lambda} \tilde{S}^{-1}$ if and only if $\tilde{\lambda}(\tilde{\lambda} I - S - R s^* \alpha)^{-1} = \tilde{\lambda}(\tilde{\lambda} I - \tilde{\lambda} 1_{n_p} \alpha - S)^{-1}$ if and only if $R s^* \alpha = \tilde{\lambda} 1_{n_p} \alpha$ if and only if $s^* \alpha = -S 1_{n_p} \alpha$, which holds. □

As $\pi_0 = 1 - \lambda$, we have due to [70, Equation (2.5)]:

$$\alpha(-\tilde{S})^{-1} s^* = 1. \tag{10.23}$$

We note, that (10.23) can also be obtained by using Woodbury matrix identity and $s^* = -S 1_{n_p}$. By using the following Lemma, we can quickly calculate the invariant distribution:

**Lemma 10.4.14.** *For the JUT(m) policy with job sizes of the phase type (with mean one), probe rate $\delta$ and arrival rate $\lambda$, the following relation holds for $\ell = 1, 2, \ldots$:*

$$\pi_\ell = \left[ \left( 1[\ell = 1](1 - \lambda) + 1[\ell \leq m] \frac{\delta}{\tilde{\lambda}} \right) \alpha + 1[\ell \geq 2] \pi_{\ell - 1} \right] R. \tag{10.24}$$

*Proof.* This follows directly from [70, Equation (2.3)]. □

Define for $k \geq 1$ and $j = 1, \ldots, n_p$: $u_{k,j} = \sum_{\ell \geq k} \pi_{\ell,j}$ and $u_k = (u_{k,1}, \ldots, u_{k,n_p})$. We now get the following explicit formulas:

**Theorem 10.4.15.** *Suppose $G$ is PH$(\alpha, S)$ distributed (with mean 1). Then, for $k = 1, \ldots, m$:*

$$\pi_k = \left( 1 - \lambda + \frac{\delta}{\tilde{\lambda}} \right) \alpha R^k + \frac{\delta}{\tilde{\lambda}} \alpha (I - R)^{-1} \left( R - R^k \right), \tag{10.25}$$

*and for $k > m$:*

$$\pi_k = \left[\left(1-\lambda+\frac{\delta}{\tilde{\lambda}}\right)\alpha + \frac{\delta}{\tilde{\lambda}}\alpha(I-R)^{-1}\left(R^{1-m}-I\right)\right]R^k, \tag{10.26}$$

*where $R = -\tilde{\lambda}\tilde{S}^{-1} = -\tilde{\lambda}(S-\tilde{\lambda}I+\tilde{\lambda}1_{n_p}\alpha)^{-1}$ and where $1_{n_p}$ is a column vector of ones of height $n_p$. Further, the following holds for $k = 2,\dots,m$:*

$$u_k = u_1R^{k-1} + \frac{\delta}{\tilde{\lambda}}\alpha\sum_{\ell=1}^{k-1}(m-\ell)R^{k-\ell} \tag{10.27}$$

*and for $k > m$:*

$$u_k = u_mR^{k-m} = u_1R^{k-1} + \frac{\delta}{\tilde{\lambda}}\alpha\sum_{\ell=1}^{m-1}(m-\ell)R^{k-\ell}, \tag{10.28}$$

*where*

$$u_1 = \left(1-\lambda+\frac{\delta m}{\tilde{\lambda}}\right)\alpha R(I-R)^{-1}. \tag{10.29}$$

*Proof.* Using induction one can check that (10.25) and (10.26) satisfy (10.24). Further, from (10.24) and the definition of $u_k$, we get

$$u_1 = \left[\left(1-\lambda+\frac{\delta m}{\tilde{\lambda}}\right)\alpha + u_1\right]R, \tag{10.30}$$

$$u_k = \left[(m-k+1)\frac{\delta}{\tilde{\lambda}}\alpha + u_{k-1}\right]R, \quad \text{if } 2 \le k \le m, \tag{10.31}$$

$$u_k = u_{k-1}R, \quad \text{if } k > m. \tag{10.32}$$

For $m > 1$, Equations (10.29), (10.27) and (10.28) immediately follow from (10.30), (10.31) and (10.32) respectively, while for $m = 1$, Equations (10.29) and (10.27) follow from (10.30) and (10.32). □

*Remark* 10.4.16. We note the following:

- As $u_1 1_{n_p}$ is the probability that the queue is busy, we have $u_1 1_{n_p} = \lambda$. Therefore: $\alpha R(I-R)^{-1}1_{n_p} = \lambda/(1-\lambda+\delta m/\tilde{\lambda})$.
- $u_{1,j}/\lambda$ is the probability that a job in service is in phase $j$, for $j = 1,\dots,n_p$.
- We can avoid calculating the nested inverse in (10.27), (10.28) and (10.29) as follows: by using Woodbury matrix identity we get

$$(I-R)^{-1} = \left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1} = \left[I+\tilde{\lambda}(S+\tilde{\lambda}(1_{n_p}\alpha-I))^{-1}\right]^{-1} = I-\tilde{\lambda}(S+\tilde{\lambda}1_{n_p}\alpha)^{-1}.$$

  Further, due to Sherman-Morrison formula and the assumption that jobs have mean 1, we get $(S+\tilde{\lambda}1_{n_p}\alpha)^{-1} = S^{-1} - \frac{\tilde{\lambda}}{1-\tilde{\lambda}}S^{-1}1_{n_p}\alpha S^{-1}$. This implies that

$$(I-R)^{-1} = I-\tilde{\lambda}S^{-1} + \frac{\tilde{\lambda}^2}{1-\tilde{\lambda}}S^{-1}1_{n_p}\alpha S^{-1}.$$

The formula for the mean response time in case of jobs of the phase type can be derived directly from the explicit formulas for $u_k$, without the use of generating function or mean value analysis:

**Proposition 10.4.17.** *For the JUT(m) policy with job sizes of the phase type (with mean one) and arrival rate $\lambda$ we find that the mean response time is given by*

$$E[R(m)] = 1 + \frac{\tilde{\lambda}}{1-\tilde{\lambda}}\alpha S^{-2}1_{n_p} + \frac{\delta}{\lambda}\frac{m(m-1)}{2(1-\tilde{\lambda})}. \tag{10.33}$$

*Note, that $2\alpha S^{-2}1_{n_p}$ is the second moment of the PH($\alpha, S$) distribution.*

*Proof.* We first compute the average queue length of the system. By using Theorem 10.4.15 we get:

$$\begin{aligned}
E[Q] &= \sum_{k\geq 1} u_k 1_{n_p} \\
&= u_1 \sum_{k=1}^{m} R^{k-1}1_{n_p} + \frac{\delta}{\tilde{\lambda}}\alpha \sum_{k=2}^{m}\sum_{\ell=1}^{k-1}(m-\ell)R^{k-\ell}1_{n_p} \\
&+ u_1 \sum_{k=m+1}^{\infty} R^{k-1}1_{n_p} + \frac{\delta}{\tilde{\lambda}}\alpha \sum_{k=m+1}^{\infty}\sum_{\ell=1}^{m-1}(m-\ell)R^{k-\ell}1_{n_p} \\
&= u_1(I-R)^{-1}1_{n_p} + \frac{\delta}{\tilde{\lambda}}\alpha\sum_{\ell=1}^{m-1}(m-\ell)(I-R)^{-1}(R-R^{m-\ell+1})1_{n_p} \\
&+ \frac{\delta}{\tilde{\lambda}}\alpha\sum_{\ell=1}^{m-1}(m-\ell)(I-R)^{-1}R^{m-\ell+1}1_{n_p} \\
&= u_1(I-R)^{-1}1_{n_p} + \frac{\delta}{\tilde{\lambda}}\frac{(m-1)m}{2}\alpha(I-R)^{-1}R1_{n_p} \\
&= u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p} + \delta\frac{(m-1)m}{2}\alpha\left(-\tilde{S}-\tilde{\lambda}I\right)^{-1}1_{n_p} \\
&= u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p} + \delta\frac{(m-1)m}{2}\alpha\left(-S-\tilde{\lambda}1_{n_p}\alpha\right)^{-1}1_{n_p}.
\end{aligned}$$

We now rewrite both terms. By using the Sherman–Morrison formula, we get that

$$\alpha\left[-S-\tilde{\lambda}1_{n_p}\alpha\right]^{-1}1_{n_p} = \alpha(-S)^{-1}1_{n_p} + \frac{\tilde{\lambda}\alpha(-S)^{-1}1_{n_p}\alpha(-S)^{-1}1_{n_p}}{1-\tilde{\lambda}\alpha(-S)^{-1}1_{n_p}}.$$

As jobs have mean 1, this further equals $1+\tilde{\lambda}/(1-\tilde{\lambda}) = 1/(1-\tilde{\lambda})$. Hence,

$$E[Q] = u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p} + \delta\frac{(m-1)m}{2(1-\tilde{\lambda})}.$$

The first terms requires more work. We calculate

$$u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p} = (\tilde{\lambda}-\tilde{\lambda}\lambda+\delta m)\alpha(-\tilde{S})^{-1}\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-2}1_{n_p}$$

$$= \lambda(1-\tilde{\lambda})\alpha(-\tilde{S}-\tilde{\lambda}\tilde{I})^{-1}\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p}$$
$$= \lambda(1-\tilde{\lambda})\alpha(-S-\tilde{\lambda}1_{n_p}\alpha)^{-1}\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p}.$$

Due to Sherman-Morrison formula this equals:

$$\begin{aligned}
&\lambda(1-\tilde{\lambda})\alpha\left[(-S)^{-1}+\frac{\tilde{\lambda}(-S)^{-1}1_{n_p}\alpha(-S)^{-1}}{1-\tilde{\lambda}\alpha(-S)^{-1}1_{n_p}}\right](I+\tilde{\lambda}\tilde{S}^{-1})^{-1}1_{n_p}\\
&=\lambda(1-\tilde{\lambda})\left[\alpha(-S)^{-1}+\frac{\tilde{\lambda}}{1-\tilde{\lambda}}\alpha(-S)^{-1}\right](I+\tilde{\lambda}\tilde{S}^{-1})^{-1}1_{n_p}\\
&=-\lambda\alpha S^{-1}(I+\tilde{\lambda}\tilde{S}^{-1})^{-1}1_{n_p}\\
&=-\lambda\alpha(S+\tilde{\lambda}\tilde{S}^{-1}S)^{-1}1_{n_p}\\
&=-\lambda\alpha\left[S+\tilde{\lambda}(S+\tilde{\lambda}(1_{n_p}\alpha-I))^{-1}(S^{-1})^{-1}\right]^{-1}1_{n_p}\\
&=-\lambda\alpha\left[S+\tilde{\lambda}(I+\tilde{\lambda}S^{-1}(1_{n_p}\alpha-I))^{-1}\right]^{-1}1_{n_p}.
\end{aligned}$$

Due to Woodbury matrix identity we get that $u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p}$ further equals to

$$\begin{aligned}
&-\lambda\alpha\left[S^{-1}-\tilde{\lambda}S^{-1}\left(I+\tilde{\lambda}S^{-1}(1_{n_p}\alpha-I)+\tilde{S}^{-1}\right)^{-1}S^{-1}\right]1_{n_p}\\
&=\lambda\alpha(-S)^{-1}\left[I-\tilde{\lambda}\left(I+\tilde{\lambda}S^{-1}1_{n_p}\alpha\right)^{-1}S^{-1}\right]1_{n_p}\\
&=\lambda+\lambda\tilde{\lambda}\alpha(-S)^{-1}\left(-S-\tilde{\lambda}1_{n_p}\alpha\right)^{-1}1_{n_p}
\end{aligned}$$

By applying Sherman–Morrison formula once again we get:

$$\begin{aligned}
u_1\left(I+\tilde{\lambda}\tilde{S}^{-1}\right)^{-1}1_{n_p} &= \lambda+\lambda\tilde{\lambda}\alpha(-S)^{-1}\left[(-S)^{-1}+\frac{\tilde{\lambda}(-S)^{-1}1_{n_p}\alpha(-S)^{-1}}{1-\tilde{\lambda}\alpha(-S)^{-1}1_{n_p}}\right]1_{n_p}\\
&=\lambda+\lambda\frac{\tilde{\lambda}}{1-\tilde{\lambda}}\alpha S^{-2}1_{n_p},
\end{aligned}$$

where we used that jobs have mean one in the last equality. It now follows that

$$E[Q]=\lambda+\lambda\frac{\tilde{\lambda}}{1-\tilde{\lambda}}\alpha S^{-2}1_{n_p}+\delta\frac{(m-1)m}{2(1-\tilde{\lambda})}.$$

Applying Little's Law allows us to conclude this proof. □

In case of PH jobs, it is thus possible to derive the formulas for the invariant distribution and mean response time by only using Ramaswami's formula, Little's law and Woodbury matrix identity (as Sherman-Morrison formula is a special case of the former).

### 10.4.3 Exponential job sizes

If we further restrict to exponential job sizes, Theorems 10.4.12 and 10.4.15 further simplify as $\alpha = 1$, $n_p = 1$ and $R = \tilde{\lambda}$. In fact, from the balance equations we immediately obtain formula for the probabilities $u_k$, $k \geq 1$.

**Theorem 10.4.18.** *Suppose G is exponentially distributed (with mean 1). Then, for $k = 1, \ldots, m$:*

$$\pi_k = (1-\lambda)\tilde{\lambda}^k + \delta \frac{1-\tilde{\lambda}^k}{1-\tilde{\lambda}},$$

*and for $k > m$:*

$$\pi_k = (1-\lambda)\tilde{\lambda}^k + \delta \frac{\tilde{\lambda}^{k-m} - \tilde{\lambda}^k}{1-\tilde{\lambda}}.$$

*Further, for $k = 1, \ldots, m$:*

$$u_k = \tilde{\lambda}^{k-1}\lambda + \delta \sum_{\ell=1}^{k-1} \tilde{\lambda}^{k-\ell-1}(m-\ell),$$

*and for $k > m$:*

$$u_k = \tilde{\lambda}^{k-m}\left[\tilde{\lambda}^{m-1}\lambda + \delta \sum_{\ell=1}^{m-1} \tilde{\lambda}^{m-\ell-1}(m-\ell)\right],$$

*where $u_k = \sum_{\ell \geq k} \pi_\ell$.*

*Proof.* As the up-crossing rate must equal the down-crossing rate in equilibrium, we have for every $k \geq 1$:

$$\tilde{\lambda}\pi_{k-1} + 1[k \leq m]\delta_0(1-\lambda) = \pi_k.$$

By summing from $k$ to infinity for every $k \geq 1$, we get:

$$u_k = \tilde{\lambda} u_{k-1} + \max(m-k+1, 0)\delta.$$

This implies immediately that $u_1 = \tilde{\lambda} + m\delta = \lambda$, which coincides with $u_1 = 1 - \pi_0 = \lambda$. The simple recursive formula above can be further solved to obtain an explicit formula for $u_k$. For $k < m$ we now get

$$\begin{aligned}
\pi_k &= u_k - u_{k+1} \\
&= \tilde{\lambda}^{k-1}\lambda + \delta \sum_{\ell=1}^{k-1} \tilde{\lambda}^{k-\ell-1}(m-\ell) - \tilde{\lambda}^k\lambda - \delta \sum_{\ell=1}^{k} \tilde{\lambda}^{k-\ell}(m-\ell) \\
&= \tilde{\lambda}^{k-1}\lambda - \delta m \tilde{\lambda}^{k-1} + \delta \sum_{\ell=2}^{k} \tilde{\lambda}^{k-\ell}(m-\ell+1) - \tilde{\lambda}^k\lambda - \delta \sum_{\ell=2}^{k} \tilde{\lambda}^{k-\ell}(m-\ell) + \delta\tilde{\lambda}^{k-1} \\
&= \tilde{\lambda}^k - \tilde{\lambda}^k\lambda + \delta \sum_{\ell=2}^{k} \tilde{\lambda}^{k-\ell} + \delta\tilde{\lambda}^{k-1} \\
&= \tilde{\lambda}^{k-1}\lambda + \delta \sum_{\ell=1}^{k-1} \tilde{\lambda}^{k-\ell-1}(m-\ell).
\end{aligned}$$

Analogously, we can prove the formula for $\pi_k$ in case $k \geq m$. □

In case of exponential job sizes we can also analytically invert the Laplace transform of the response time distribution given by (10.8).

**Theorem 10.4.19.** *Suppose G is exponentially distributed with mean 1. The pdf of the response time distribution is given by*

$$f_R(t) = \frac{e^{-t(1-\tilde{\lambda})}}{\lambda}\left(\frac{\delta}{\tilde{\lambda}^{m-1}(1-\tilde{\lambda})} - \frac{\delta\tilde{\lambda}}{1-\tilde{\lambda}} + \tilde{\lambda}(1-\lambda)\right) - \frac{\delta e^{-t}}{\lambda(1-\tilde{\lambda})}\sum_{k=1}^{m-1}\frac{t^{k-1}(1-\tilde{\lambda}^{m-k})}{(k-1)!\tilde{\lambda}^{m-k}}. \tag{10.34}$$

*Proof.* For exponential job sizes with mean 1 we have $Y^*(s) = G^*(s) = 1/(1+s)$, where the first equality follows from the memorylessness. After some simplifications, we further get

$$\xi(G^*(s)) = \frac{(1-\tilde{\lambda})(1+s)}{1-\tilde{\lambda}+s}.$$

Then, by using (10.5), the equation above and $\lambda(1-\tilde{\lambda})/(\delta_0 m+\tilde{\lambda}) = 1-\lambda$, we have

$$\pi(G^*(s)) = \frac{(1-\lambda)(1+s)}{1-\tilde{\lambda}+s} + \frac{\delta}{1-\tilde{\lambda}+s}\sum_{i=0}^{m-1}\left(\frac{1}{1+s}\right)^i.$$

By putting everything together, it follows that

$$\begin{aligned} R^*(s) &= \frac{\tilde{\lambda}(1-\lambda)}{\lambda(1-\tilde{\lambda}+s)} + \frac{\delta}{\lambda}\left(\frac{\tilde{\lambda}}{1-\tilde{\lambda}+s}+1\right)\sum_{i=1}^{m}\left(\frac{1}{1+s}\right)^i \\ &= \frac{1}{\lambda(1-\tilde{\lambda}+s)}\left(\tilde{\lambda}(1-\lambda) + \delta\sum_{i=0}^{m-1}\left(\frac{1}{1+s}\right)^i\right). \end{aligned}$$

Applying the inverse Laplace transform to $R^*(s)$ gives

$$f_R(t) = \frac{e^{-t(1-\tilde{\lambda})}}{\lambda\tilde{\lambda}^{m-1}}\left(\delta\sum_{i=0}^{m-1}\tilde{\lambda}^i - \tilde{\lambda}^m(1-\lambda)\right) - \frac{\delta e^{-t}}{\lambda}\sum_{k=1}^{m-1}\frac{t^{k-1}}{(k-1)!\tilde{\lambda}^{m-k}}\sum_{i=0}^{m-k-1}\tilde{\lambda}^i,$$

which equals (10.34). □

Note that for $m = 1$, Equation (10.34) simplifies to $f_R(t) = e^{-t(1-\tilde{\lambda})}(1-\tilde{\lambda})$.

## 10.5 Numerical Experiments

In this section we compare the performance of JUT($m$) with some existing policies and look at how sensitive its performance is with respect to the parameter $m$. We mainly focus on the regime where the communication overhead is well below 1 message per job as simple policies otherwise exist that can achieve vanishing delays in the large-scale limit [52,72].

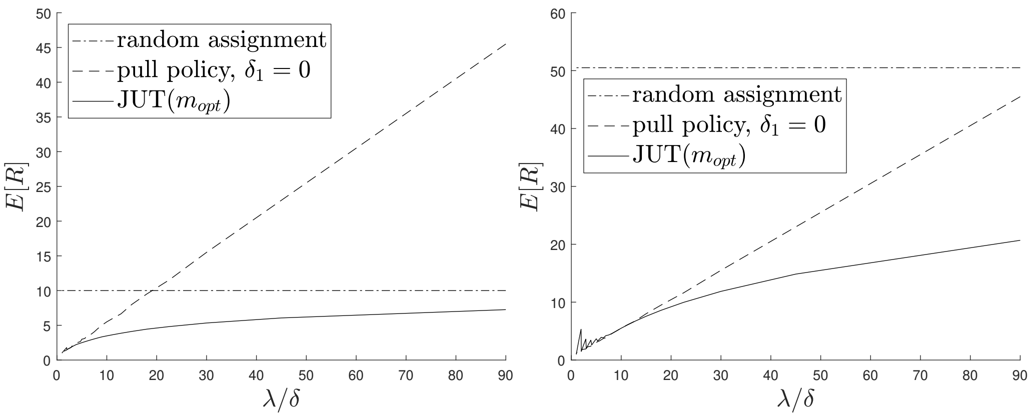


Figure 10.1: Mean response time of JUT($m_{opt}$), random assignment and the pull policy of [36] for $\lambda = 0.9$ and exponential (left) or hyperexponential (right) job sizes.

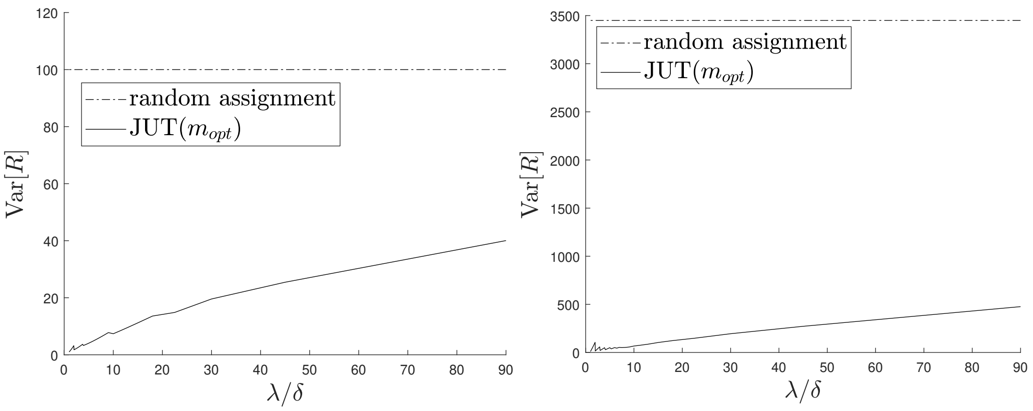


Figure 10.2: Variance of the response time of JUT($m_{opt}$) and random assignment for $\lambda = 0.9$ and exponential (left) or more variable (right) job sizes with $E[G^2] = 11$ and $E[G^3] = 330$.

First we compare the performance of JUT($m$) with random assignment and with the pull policy of [36]. We did not include a comparison with the asynchronous push policy in [74] as this policy is inferior to the pull policy of [36] as illustrated in Table 10.1. For the pull policy in [36] we set $\delta_1 = 0$, meaning only idle servers send updates, as this tends to yield the best performance. In Figure 10.1 we compare the mean response time of the different policies as a function of $\lambda/\delta$ with $\lambda = 0.9$, where $\delta/\lambda$ represents the mean number of communication overhead messages used per job. As random assignment does not require any communication overhead, its mean response time is fixed. We consider both exponential job sizes (in the left plot) and more variable job sizes (in the right plot). For the more variable job sizes we used hyperexponential job sizes with balanced means such that the squared coefficient of variation (SCV) equals 10. This implies that $E[G^2] = 11$ as $E[G^2] = SCV + 1$ (when $E[G] = 1$). Note that the mean response time of JUT($m$) and random assignment only depends on $E[G^2]$ (as $E[G] = 1$), thus the results apply to any job size distribution for which the SCV equals 10.

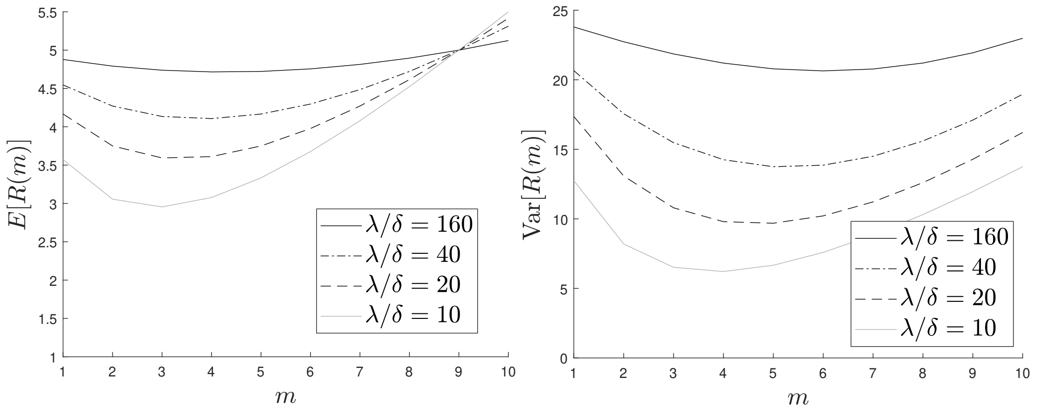


Figure 10.3: Mean and variance of the response time of JUT($m$) as a function of $m$ for $\lambda = 0.8$ and exponential job sizes.

The results in Figure 10.1 clearly show that the mean response time of the pull policy grows almost linearly in $\lambda/\delta$ and therefore the pull policy only outperforms random assignment when $\lambda/\delta$ is small enough, meaning when the communication overhead is large enough. The mean response time of the JUT($m$) policy with $m = m_{opt}$ on the other hand grows much more slowly, is superior to random assignment for any $\lambda/\delta$ and outperforms the pull policy unless $\lambda/\delta$ is close to one. We further note that both the pull and JUT($m$) policy perform better compared to random assignment when the job sizes are more variable.

The previous results focused on the mean response time. We now consider the variance of the response time. As no results for the variance of the response time where presented in [36] for the pull policy, we only compare the variance of the response time of JUT($m$) with random assignment in Figure 10.2. We consider the same two job size distributions as in Figure 10.1 and again set $\lambda = 0.9$. Note that for JUT($m$) and random assignment the variance is only affected by the first three moments of the job size distribution. We see that JUT($m_{opt}$) not only outperforms random assignment in terms of the mean response time for any $\lambda/\delta$, but also significantly reduces the variance in all cases.

Note, that as $\lambda/\delta$ increases, so does $m_{opt}$. This increase of the value of $m_{opt}$ corresponds to a discontinuity in the graphs of the mean response time and the variance of the response time. These are the discontinuities that are noticeable in Figures 10.1 and 10.2 for low values of $\lambda/\delta$.

In the previous experiments we set $m = m_{opt}$, we now look at the impact of $m$ on the mean and variance of the response time of JUT($m$). We start by assuming exponential job sizes and set $\lambda = 0.8$. In Figure 10.3 we consider $\lambda/\delta \in \{10, 20, 40, 160\}$. We note that the mean response time is not highly sensitive to the choice of $m$, especially when the communication overhead is small (that is, $\lambda/\delta$ is large). This means that it suffices to get a good estimate of the arrival rate $\lambda$ and the first two moments $E[G]$ and $E[G^2]$ of the job size distribution to get near optimal performance as $m_{opt}$ does not depend on any other job size characteristics. We further note that while the value of $m$ that minimizes the mean response time does not minimize the variance of the response time, it does yield a near optimal variance. The value of $m$ that actually minimizes the variance appears to

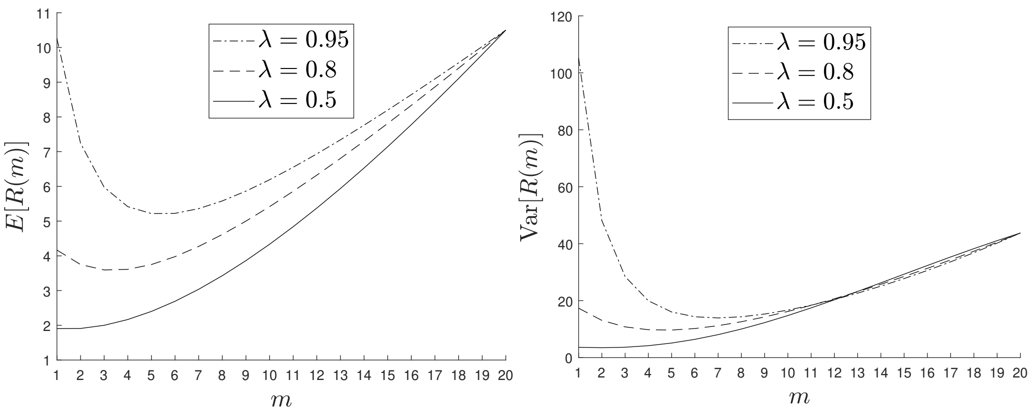


Figure 10.4: Mean and variance of the response time of JUT($m$) as a function of $m$ for $\lambda/\delta = 20$ and exponential job sizes.

be somewhat larger than the value of $m$ that minimizes the mean.

In Figure 10.4 we consider the same scenario as in Figure 10.3, but now we fix $\lambda/\delta = 20$ and let $\lambda \in \{0.5, 0.8, 0.95\}$. The results indicate that the choice of $m$ appears to become more important as $\lambda$ increases (for instance simply setting $m = 1$ is far from optimal for larger $\lambda$). The mean response time of random assignment (which requires no overhead) equals $1/(1-\lambda)$. Hence, the JUT($m$) policy mostly offers a significant reduction over random assignment when $\lambda$ is large. Regarding the variance of the response time, we can make the same remarks as in Figure 10.3.

In the previous two figures jobs were assumed to have an exponentially distributed size. We now consider job size distributions that are more variable. As before we consider hyperexponential jobs sizes with balanced means such that the squared coefficient of variation (SCV) equals 1, 5 and 10. This implies that $E[G^2] \in \{2, 6, 11\}$ and $E[G^3] \in \{6, 90, 330\}$. Figure 10.5 depicts the results for $\lambda/\delta = 20$ and $\lambda = 0.8$. Similar trends are observed for the three SCV values considered. We further note that the mean response time becomes insensitive to the job size distribution when $m = \lambda/\delta$ as the mean response time reduces to $1 + (\lambda/\delta - 1)/2$ in such case according to (10.6).

## 10.6 Conclusion

Hyper-scalable load balancing policies operate in the regime where the communication overhead is below one message per job. Existing hyper-scalable policies such as [36,74] work well when the overhead is fairly close to one message per job, but have poor performance when the allowed overhead is significantly smaller. More specifically, when the communication overhead of these policies tends to zero, the mean response time tends to infinity even when the system is only lightly loaded.

In this chapter we introduced a novel hyper-scalable load balancing policy, called JUT($m$), where $m$ is an input parameter. Under this policy incoming jobs are assigned in a greedy manner to a server with the lowest estimated queue length, unless all estimated queue

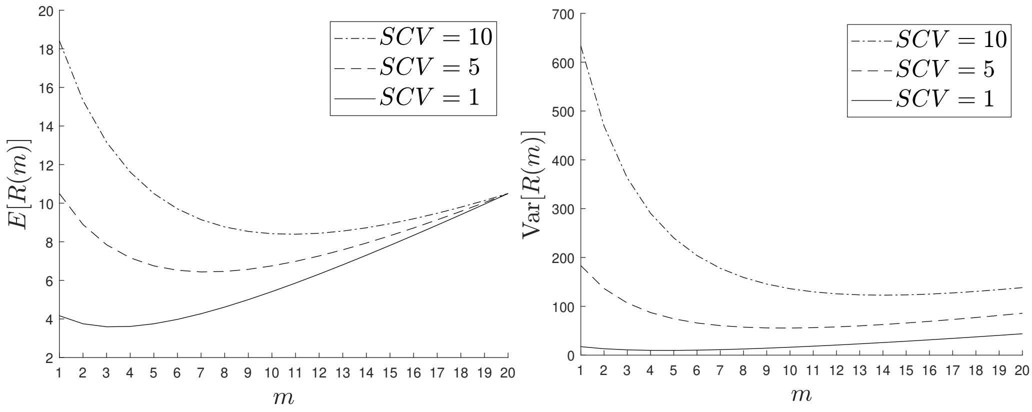


Figure 10.5: Mean and variance of the response time of JUT($m$) as a function of $m$ for $\lambda/\delta = 20$ and $\lambda = 0.8$.

lengths are at least $m$. In the latter case an incoming job is assigned to a random server. We studied the performance of the JUT($m$) policy in a large-scale system using the queue-at-the-cavity approach and demonstrated the accuracy of this approach using simulation.

We presented closed form results for the generating function of the queue length distribution and the Laplace transform of the response time distribution. Using these results we derived a simple closed form solution for the mean response time and the value of $m$ that minimizes the mean response time. These simple solutions indicate that the mean response time and the optimal value of $m$ only depend on the second moment of the job size distribution (after renormalizing the first moment to one). We also derived a closed form expression for the second moment (and thus the variance) of the response time.

Numerical results illustrate that the JUT($m$) policy is far superior to existing policies when the communication overhead is well below one message per job and outperforms random assignment irrespective of the communication overhead allowed. The performance gain achieved also increases as the job sizes become more variable. Finally, we indicated that the performance of JUT($m$) is not very sensitive to the value of the parameter $m$, meaning it suffices to have a good estimate of the arrival rate and first two moments of the job size distribution in order to properly set the parameter $m$, which should not be too difficult in the present era of big data.

Chapter 11

# Other JUT($m$) policies: JUT($m, \delta_1$) and PUT($m$)

*This chapter contains unpublished research, which may be a basis for future paper. The first results contained in this chapter were written around the same time as those of Chapter 10, while the last results here are some of the final results added to the thesis.*

## 11.1 Introduction

In this Chapter we present two load balancing policies related to JUT($m$) policy from Chapter 10. The first of these policies, can be seen as a generalization of JUT($m$) policy from the previous chapter: we assume that upon a completion of a job in queue, there is a $\delta_1$ probability that the queue will update the dispatcher about its length. If $\delta_1 > 0$, we shall call this policy "Busy Join-Up-To $m$" and denote it by JUT($m, \delta_1$) for short. We shall call the second of the policies studied in this Chapter "Push-Up-To $m$" (PUT($m$) for short). It differs from JUT($m$) and JUT($m, \delta_1$) policies in who initiates the updates: in case of JUT($m$) and JUT($m, \delta_1$) policies the queues initiate the updates, while for PUT($m$) it is the dispatcher that sends update requests to the queues.

The rest of this Chapter is structured as follows: we describe the system and the two policies in Section 11.2. Through the use of the cavity map, we study the JUT($m, \delta_1$) and PUT($m$) policies in Sections 11.3 and 11.4 respectively. There, for both policies, we present an algorithm which allows us to numerically determine the invariant distribution in case of PH distributed jobs. In case of exponential job sizes we also provide recursive formulas which allow us to speed up this algorithm. We validate the model for the two policies of this chapter in Section 11.5, while in Section 11.6 we conduct numerical experiments, where we also compare the three policies. Finally, Section 11.7 contains some concluding remarks.

## 11.2 Description of the system and the policies

We consider a set of $N$ homogeneous servers, each with its own infinite buffer, and a central dispatcher. Every server processes the jobs in its queue in FCFS order. Jobs arrive at the dispatcher according to a Poisson process with rate $N\lambda$, with $0 < \lambda < 1$. The service requirements of a job have a PH distribution with parameters $(\alpha, S)$ and with mean one, i.e. $\alpha(-S)^{-1}1_{n_p} = 1$, where $n_p$ is the number of phases of the job size distribution. We again denote $s^* = -S1_{n_p}$.

For each server the dispatcher maintains an upper bound on its queue length. We will call these upper bounds "estimates". Upon an arrival, the dispatcher directs the incoming job to a server with minimal estimated number of jobs, with ties broken uniformly at random, if a server with an estimate of less than $m$ jobs is known to the dispatcher. Otherwise an incoming job is assigned to a random server. When a server receives a job its estimate is increased by one. The estimates kept by dispatcher can also be updated as follows:

In case of PUT($m$) policy, the dispatcher probes a random server at rate $N\delta$, with $\delta > 0$.

In case of the JUT($m, \delta_1$) policy empty servers send updates to the dispatcher at rate $\delta_0$. Further, every time there is a job completion in a server, this server will update the dispatcher on its queue length with probability $\delta_1$. We assume that $\delta_1 > 0$ throughout the rest of the Chapter.

We define $\delta = \delta_0(1 - \lambda) + \delta_1\lambda$ as the overall update rate. When $\delta \geq \lambda$, we can set $\delta_1 = 1$. In this case the policy reduces to the JSQ policy, as every time a job is completed, the dispatcher is updated about the queue length. This implies that if $\delta_1 = 1$, then the queue lengths are always one or zero in the large-scale limit, as for $N \to \infty$ there always exist empty servers. When studying the JUT($m, \delta_1$) policy, we therefore assume that $\delta < \lambda$.

The system of $N$ servers is hard to analyze directly, therefore we use of the so-called "queue at the cavity approach" (see [9] or Section 4.3) to approximate the system and in Section 11.5 perform simulations that suggest that as $N$ tends to infinity, the cavity method gives exact results. Recall that the idea of the cavity queue is to focus on the evolution of a single server and to assume that all other servers have independent and identically distributed queue lengths.

Define a "slot" as follows. When an update occurs in a queue, we call the spaces that can be filled with jobs to make the queue length $m$ "slots". For any policy and parameter setting, we can define some function $\eta(m)$, with $\eta(m)$ the average number of discovered slots upon an update. When studying these policies with a fixed value of $m$, we need to distinguish between two cases:

- Case $\lambda > \delta\eta(m)$. This is the most interesting case. For this case we find that, in the large-scale limit, when a queue with less than $m$ jobs is discovered, the dispatcher immediately assigns a batch of jobs to it, such that the new length of the queue is $m$. Additionally, we have another arrival rate

$$\tilde{\lambda} = \lambda - \delta\eta(m) \tag{11.1}$$

  of jobs which are assigned arbitrarily.

- Case $\lambda \leq \delta\eta(m)$. In this case, we never run out of servers with an estimate which is smaller than $m$, therefore the policy will always assign each incoming job to the server with the smallest estimated queue length. It is not hard to see that in this case, the JUT($m, \delta_1$) and PUT($m$) policy respectively reduce to the hyperscalable pull and push policies we studied in [36].

We can find bounds on how the parameter $m$ should be chosen such that $\tilde{\lambda} \geq 0$:

**Theorem 11.2.1.** *For the JUT($m, \delta_1$) policy with jobs of mean size* 1 *and arrival rate $\lambda$ and update rate $\delta$, we have $\tilde{\lambda} \geq 0$ if and only if*

$$m \leq \log(1 - \lambda\delta_1/\delta)/\log(1 - \delta_1). \tag{11.2}$$

*Proof.* The hyperscalable pull system from [36] with $\nu = 0$ and maximum queue length $m$ coincides with the JUT($m, \delta_1$) system with $\tilde{\lambda} = 0$. The claim therefore immediately follows from [36, Theorem 2.]. □

**Theorem 11.2.2.** *For the PUT($m$) policy with jobs of mean size* 1*, arrival rate $\lambda$ and update rate $\delta$, we have $\tilde{\lambda} \geq 0$ if and only if*

$$m \leq \frac{\log\left(\frac{1}{y} + \left(\frac{\lambda}{\delta(1-\lambda)} - 1\right) \cdot \frac{1-y}{y}\right)}{\log(1/y)}, \tag{11.3}$$

*where $y = \alpha(\delta I - S)^{-1}s^*$.*

*Proof.* The hyperscalable push system from [36] with $\nu = 0$ and maximum queue length $m$ coincides with the PUT($m$) system with $\tilde{\lambda} = 0$. The claim therefore immediately follows from [36, Theorem 1.]. □

In case of the JUT($m, \delta_1$) policy, we can also give a bound on $\delta$ such that $\tilde{\lambda} \geq 0$:

**Corollary 11.2.3.** *For the JUT($m, \delta_1$) policy with jobs of mean size* 1 *and arrival rate $\lambda$, we have $\tilde{\lambda} > 0$ if and only if*

$$\delta < \frac{\lambda\delta_1}{1 - (1 - \delta_1)^m}.$$

*Proof.* This follows immediately by solving (11.2) for $\delta$. □

In the remainder of the next two sections, we will always assume that $\lambda > \delta\eta(m)$ or, equivalently, that $\tilde{\lambda} > 0$. Then, the state space of the cavity queue is given by

$$\Omega = \{0\} \cup \{(k, \ell) \mid k \in \mathbb{N} \setminus \{0\}, \ell \in \{1, \ldots, n_p\}\},$$

where the queue is in state 0 if it has no jobs and in state $(k, \ell)$ if it has $k$ jobs (we shall call $k$ the level of the queue) and the job in service is in phase $\ell$. As we assumed $\tilde{\lambda} \leq \lambda < 1$, the cavity queue is positive recurrent for both policies. Similarly to Subsection 10.4.2, we shall denote $\pi_k = \left[\pi_{k,1}, \pi_{k,2}, \ldots, \pi_{k,n_p}\right]$, for $k \geq 1$, where $\pi_{k,\ell}$ is the probability that the cavity queue is in state $(k, \ell)$. For $k \geq 1$, we shall also denote $u_k = \sum_{\ell \geq k} \pi_\ell$.

## 11.3 JUT($m, \delta_1$) policy

### 11.3.1 Phase-type distributed job sizes

If we order the states lexicographically, then the cavity queue for the JUT($m, \delta_1$) policy and $m \geq 2$ can be described by a CTMC the rate matrix

$$\begin{bmatrix}
Q_{0,0} & Q_{0,1} & & & & & Q_{0,m} & & & \\
Q_{1,0} & Q_{1,1} & Q_{1,2} & & & & Q_{1,m} & & & \\
& Q_{2,1} & Q_{2,2} & Q_{2,3} & & & Q_{2,m} & & & \\
& & \ddots & \ddots & \ddots & & \vdots & & & \\
& & & Q_{m-2,m-3} & Q_{m-2,m-2} & Q_{m-2,m-1} & Q_{m-2,m} & & & \\
& & & & Q_{m-1,m-2} & Q_{m-1,m-1} & Q_{m-1,m} & & & \\
& & & & & Q_{m,m-1} & Q_{m,m} & Q_{m,m+1} & & \\
& & & & & & Q_{m+1,m} & Q_{m+1,m+1} & Q_{m+1,m+2} & \\
& & & & & & & \ddots & \ddots & \ddots
\end{bmatrix}, \tag{11.4}$$

where the matrix $Q_{i,j}$ describes the transitions from level $i$ to level $j$. When the level of the queue is smaller than or equal to $m$, a job completion decreases the level if no update occurs, hence $Q_{k,k-1} = (1 - \delta_1)s^*\alpha$ for $1 < k \leq m$ and $Q_{1,0} = (1 - \delta_1)s^*$. If the level is greater than $m$, updates do not change the level, hence $Q_{k,k-1} = s^*\alpha$ for $k > m$.

When a completion occurs, the dispatcher gets updated with probability $\delta_1$. If the queue has at least $m$ jobs after the completion an update does not change its level. However is a queue has less than $m$ jobs upon an update, its level is instantly increased to $m$. The level can be increased to $m$ following an update, hence $Q_{k,m} = \delta_1 s^*\alpha$ for $1 \leq k \leq m - 2$. If the queue is empty, updates occur with rate $\delta_0$, implying $Q_{0,m} = \delta_0\alpha$. The remaining incoming jobs get assigned randomly and increase the level by one, which means that $Q_{k,k+1} = \tilde{\lambda}I$ for $k \in \mathbb{N} \setminus \{0, m - 1\}$ and $Q_{0,1} = \tilde{\lambda}\alpha$. The level can increase from $m - 1$ to $m$ both due to server updates and remaining arrivals, meaning $Q_{m-1,m} = \tilde{\lambda}I + \delta_1 s^*\alpha$.

As we have no other transitions in level 0, $Q_{0,0} = -(\delta_0 + \tilde{\lambda})$. In non-zero levels transitions also occur due to phase changes, hence $Q_{k,k} = S - \tilde{\lambda}I$ for $k \in \mathbb{N} \setminus \{0, m\}$, where $-\tilde{\lambda}I$ is due to the extra arrivals. When the level is $m$ we also have a contribution due to the fact that a queue can complete a job, send an update and be bounced back to level $m$, hence $Q_{m,m} = S - \tilde{\lambda}I + \delta_1 s^*\alpha$.

For $m = 1$ the queue at the cavity is given by

$$\begin{bmatrix}
Q_{0,0} & Q_{0,1} & & \\
Q_{1,0} & Q_{1,1} & Q_{1,2} & \\
& Q_{2,1} & Q_{2,2} & Q_{2,3} \\
& & \ddots & \ddots & \ddots
\end{bmatrix}, \tag{11.5}$$

where $Q_{0,0} = -(\delta_0 + \tilde{\lambda})$, $Q_{0,1} = (\delta_0 + \tilde{\lambda})\alpha$, $Q_{1,0} = (1 - \delta_1)s^*$, $Q_{1,1} = S - \tilde{\lambda}I + \delta_1 s^*\alpha$. Similarly to (11.4), we further have $Q_{k,k-1} = s^*\alpha$ and $Q_{k,k} = S - \tilde{\lambda}I$ for $k > 1$ and $Q_{k,k+1} = \tilde{\lambda}I$ for $k \geq 1$.

In order to numerically analyze this policy, we need to determine $\tilde{\lambda}$. When a queue of $k$ jobs completes service, there is a probability $\delta_1$ that is will update the dispatcher, if $k \leq m$, then the dispatcher will assign $m - k + 1$ jobs to the queue. If the queue is empty and it updates the dispatcher it gets assigned $m$ jobs. This implies the following relation:

$$\tilde{\lambda} = \lambda - \delta_0 m \pi_0 - \delta_1 \sum_{k=1}^{m} \sum_{\ell=1}^{n_s} (m - k + 1) \pi_{k,\ell} s_\ell^*. \tag{11.6}$$

We wish to exploit (11.6) to numerically find the invariant distribution of (11.4) (or of (11.5)). We first explain how we can find the invariant distribution of (11.4) (or of (11.5)) for a given value of $\tilde{\lambda}$. (11.4) clearly has a QBD structure in the levels greater than $m$. By using [48, Section 6.4] we get the following relations for $k \geq 2$

$$\pi_{m+k} = \pi_{m+1} R^{k-1},$$

where the matrix $R$ is the smallest non-negative solution to the quadratic matrix equation

$$\tilde{\lambda} I + R(S - \tilde{\lambda} I) + R^2 s^* \alpha = 0. \tag{11.7}$$

The solution to (11.7) is given by [27, Theorem 3.] (see also Theorem 3.3.1):

$$R = \tilde{\lambda} \left( \tilde{\lambda} I_{n_p} - \tilde{\lambda} 1_{n_p} \alpha - S \right)^{-1}.$$

Set $N = -(S - \tilde{\lambda} I + R s^* \alpha)^{-1}$. Note that due to Remark 10.4.13, we have $R = \tilde{\lambda} N$. Set further $\pi_{0:m} = [\pi_0, \pi_1, \ldots, \pi_m]$,

$$B_0 = \begin{bmatrix} 0_{(m-1)n_p+1, n_p} \\ Q_{m,m+1} \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0_{n_p,(m-1)n_p+1}, Q_{m+1,m} \end{bmatrix}.$$

Finally, set

$$B_1 = \begin{bmatrix} Q_{0,0} & Q_{0,1} & & & & & Q_{0,m} \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & & & & Q_{1,m} \\ & Q_{2,1} & Q_{2,2} & Q_{2,3} & & & Q_{2,m} \\ & & \ddots & \ddots & \ddots & & \vdots \\ & & & Q_{m-2,m-3} & Q_{m-2,m-2} & Q_{m-2,m-1} & Q_{m-2,m} \\ & & & & Q_{m-1,m-2} & Q_{m-1,m-1} & Q_{m-1,m} \\ & & & & & Q_{m,m-1} & Q_{m,m} \end{bmatrix}$$

if $m > 1$, otherwise set

$$B_1 = \begin{bmatrix} Q_{0,0} & Q_{0,1} \\ Q_{1,0} & Q_{1,1} \end{bmatrix}.$$

Due to [61, Section 2], we have

$$\pi_{m+1} = \pi_{0:m} B_0 N = \pi_m \tilde{\lambda} N = \pi_m R$$

and we can find the vector $\pi_{0:m}$ by solving

$$\pi_{0:m} \begin{bmatrix} B_1 + B_0 N B_2 & 1_{mn_p+1} + B_0 N (I - R)^{-1} 1_{n_p} \end{bmatrix} = [0'_{mn_p+1} \quad 1]. \tag{11.8}$$

Thus, for a given value of $\tilde{\lambda}$ we can find the invariant distribution of (11.4). Note now that increasing the value of $\tilde{\lambda}$ decreases $\pi_0$ and vice versa. Based on this we can apply a bisection algorithm to iteratively solve (11.4)-(11.6) (or (11.5)-(11.6)) until we have found $\tilde{\lambda}$ and $\pi$ with $\pi_0 \approx 1 - \lambda$.

### 11.3.2 Exponential job sizes

For high values of $m$ and small number of phases, solving (11.8) is the most costly step in the bisection algorithm from Subsection 11.3.1. In case of exponential job sizes (with mean 1), Equation (11.7) simplifies to

$$\tilde{\lambda} + R(-1 - \tilde{\lambda}) + R^2 = 0. \tag{11.9}$$

(11.9) has two solutions, namely 1 and $\tilde{\lambda}$, hence $R = \tilde{\lambda}$. This implies that in case of exponential jobs solving (11.8) is the only possibly costly step. We can however find recursive formulas which allow us to omit solving (11.8). To this end, we define recursively:

$$g_m = \delta_1\tilde{\lambda}, \quad g_i = \tilde{\lambda}\frac{\delta_1 + g_{i+1}}{1 + g_{i+1}} \tag{11.10}$$

and

$$f_m = \delta\frac{\delta_1 + g_m}{1 + g_m}, \quad f_i = \frac{\delta_1 + g_i}{1 + g_i}\left((m - i + 1)\delta - \sum_{j=i+1}^{m} f_j\right) \tag{11.11}$$

for $i = 1, \dots, m - 1$. Note that (11.10) and (11.11) only depend on $\delta, \delta_1, \tilde{\lambda}$ and $m$. Also note that if $\delta_1 = 0$, $g_i = f_i = 0$ for every $i$. We can now prove the following:

**Lemma 11.3.1.** *For the JUT($m, \delta_1$) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, the following relations hold for $k = 1, \dots, m$:*

$$u_{k-1}\tilde{\lambda} + (m - k + 1)\delta = (1 + g_k)u_k + \sum_{\ell=k+1}^{m} f_\ell, \tag{11.12}$$

$$\delta_1\sum_{\ell=k}^{m} u_{\ell+1} = g_k u_k + \sum_{\ell=k+1}^{m} f_\ell. \tag{11.13}$$

*Proof.* For every $k \geq 1$ we have the following detailed balanced equations (see the proof of Theorem 3.2.3), which are explained below:

$$(u_{k-1} - u_k)\tilde{\lambda} + 1[k \leq m]((1 - \lambda)\delta_0 + (\lambda - u_k)\delta_1) = (1 - 1[k \leq m]\delta_1)(u_k - u_{k+1}). \tag{11.14}$$

The LHS describes the up-crossing rate over threshold $k$, while the RHS denotes down-crossing rate under $k$. The queue up-crosses the threshold $k$ due to batch arrivals if $k \leq m$, namely due to empty queues sending updates and updates due to completions in queues with at most $k - 1$ jobs. This is described by the terms $(1 - \lambda)\delta_0$ and $(\lambda - u_k)\delta_1$ respectively. An up-crossing can also occur for any $k$ due to the additional arrivals with rate $\tilde{\lambda}$ in queues with exactly $k - 1$ jobs, which is accounted for by the term $(u_{k-1} - u_k)\tilde{\lambda}$.

A down-crossing can only occur due to a completion in queue with exactly $k$ jobs, which gives the term $u_k - u_{k+1}$. However, when $k \leq m$, there is a probability $\delta_1$ that an update will immediately bump the queue to $m$ jobs. In this case no down-crossing under the threshold $k$ occurs.

Equation (11.14) is equivalent to

$$(u_{k-1} - u_k)\tilde{\lambda} + 1[k \leq m]((1 - \lambda)\delta_0 + (\lambda - u_{k+1})\delta_1) = u_k - u_{k+1}.$$

Summing these equations from $k$ to infinity for every $k \geq 1$, we get

$$u_{k-1}\tilde{\lambda} + \max(m-k+1, 0)\delta = u_k + \delta_1 \sum_{\ell=k}^{m} u_{\ell+1}, \tag{11.15}$$

where we have used $\delta = (1-\lambda)\delta_0 + \lambda\delta_1$. Assume now that $k \leq m$. We prove both (11.12) and (11.13) at the same time using backwards induction. For $k = m$, (11.13) follows directly from (11.15) with $k = m+1$. We further have $u_{m-1}\tilde{\lambda} + \delta = u_m + \delta_1 u_{m+1}$. By using (11.15) with $k = m+1$, this is further equal to $u_m + \delta_1\tilde{\lambda}u_m = (1+g_m)u_m$. This shows the claim for $k = m$. Now suppose that the two claims hold for $k = k'+1, \ldots, m$. We need to show that they hold for $k = k'$. By using the induction hypothesis of (11.13) and (11.12) respectively, we get

$$\begin{aligned}
\delta_1 \sum_{\ell=k}^{m} u_{\ell+1} &= \delta_1 u_{k+1} + g_{k+1}u_{k+1} + \sum_{\ell=k+2}^{m} f_\ell \\
&= \frac{\delta_1 + g_{k+1}}{1+g_{k+1}} \left( u_k\tilde{\lambda} + (m-k)\delta - \sum_{\ell=k+2}^{m} f_\ell \right) + \sum_{\ell=k+2}^{m} f_\ell \\
&= g_k u_k + f_{k+1} + \sum_{\ell=k+2}^{m} f_\ell,
\end{aligned}$$

which shows (11.13) for $k = k'$. By using (11.13), we further have

$$u_{k-1}\tilde{\lambda} + (m-k+1)\delta = u_k + \delta_1 \sum_{\ell=k}^{m} u_{\ell+1} = u_k + g_k u_k + \sum_{\ell=k+1}^{m} f_\ell,$$

which shows (11.12) for $k = k'$, thus finishing the proof. □

As $u_0 = 1$ and $u_1 = 1 - \pi_0 = \lambda$, we get from (11.12) that

$$\tilde{\lambda} + m\delta = (1+g_1)\lambda + \sum_{\ell=2}^{m} f_\ell. \tag{11.16}$$

By using Equation (11.1), this implies that the average number of slots discovered per update is

$$\eta(m) = m - \frac{1}{\delta}\left( g_1\lambda + \sum_{\ell=2}^{m} f_\ell \right).$$

Unsurprisingly, $\eta(m) < m$ if $\delta_1 \neq 0$ and $\eta(m) = m$ if $\delta_1 = 0$. Equation (11.16) also implies that if $m = 1$, then

$$\tilde{\lambda} = \frac{\lambda - \delta}{1 - \delta_1\lambda}.$$

This implies that for $m = 1$, we must have $\delta < \lambda$ for $\tilde{\lambda}$ to be greater than 0. With more effort, it can be shown using Equation (11.16) that for $m = 2$, we have

$$\tilde{\lambda} = \frac{-\delta_1(\delta - 2\lambda) - 1 + \sqrt{\delta_1^2(\delta-2\lambda)^2 + 4\delta\delta_1^2 + 8\delta\delta_1\lambda - 6\delta\delta_1 - 4\delta_1\lambda^2 + 1}}{2\delta_1(1-\lambda)}$$

and that we must have $\delta < \lambda/(2-\delta_1)$ if we want $\tilde{\lambda} > 0$.

From Lemma 11.3.1, we immediately get:

**Proposition 11.3.2.** *For the JUT($m, \delta_1$) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, the probability $u_k$ that the queue length exceeds k is given by:*

$$u_k = \frac{\tilde{\lambda}^k}{\prod_{\ell=1}^k (1+g_\ell)} + \sum_{j=1}^k \frac{\tilde{\lambda}^{k-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m f_\ell\right)}{\prod_{\ell=j}^k (1+g_\ell)}, \qquad \text{if } 1 \le k \le m, \tag{11.17}$$

$$u_k = \tilde{\lambda}^{k-m}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+g_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m f_\ell\right)}{\prod_{\ell=j}^m (1+g_\ell)}\right], \qquad \text{if } k > m. \tag{11.18}$$

*Proof.* (11.17) follows directly from (11.12) and $u_0 = 1$, while (11.18) follows from (11.17) and (11.15). □

If $\tilde{\lambda}$ is known we get exact recursive formulas for the invariant distribution as $\pi_k = u_k - u_{k+1}$. Otherwise, we can numerically determine $\tilde{\lambda}$ and the value of the different $u_k$'s using the following bisection algorithm:

1. Set $\tilde{\lambda} \approx 0$.
2. Calculate $g_\ell$ and $f_\ell$ recursively for $\ell = 1, \dots, m$. Using (11.17), calculate $u_1$.
3. If $u_1 \approx \lambda$ (up to a desired precision), then stop. If $u_1 > \lambda$, decrease $\tilde{\lambda}$, otherwise increase $\tilde{\lambda}$. Go to step (2).

We can also provide a formula for the mean response time in terms the values of $g_\ell$ and $f_\ell$, $\ell = 1, \dots, m$:

**Proposition 11.3.3.** *For the JUT($m, \delta_1$) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, we have*

$$\begin{aligned} E[R] &= \frac{1}{\lambda\delta_1}\left((\delta_1 + g_1)\lambda + \sum_{\ell=2}^m f_\ell\right) \\ &\quad + \frac{\tilde{\lambda}^2}{\lambda(1-\tilde{\lambda})}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+g_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m f_\ell\right)}{\prod_{\ell=j}^m (1+g_\ell)}\right]. \end{aligned}$$

*Proof.* We first calculate $E[Q] = \sum_{k\geq 1} u_k$, the average queue length of the system. Due to (11.13) with $k = 1$ and $\delta_1 u_1$ added to the both sides of the expression, we have

$$\sum_{k=1}^{m+1} u_k = \frac{1}{\delta_1}\left((\delta_1 + g_1)u_1 + \sum_{\ell=2}^m f_\ell\right) = \frac{1}{\delta_1}\left((\delta_1 + g_1)\lambda + \sum_{\ell=2}^m f_\ell\right).$$

Due to (11.18), we further have

$$\sum_{k=m+2}^{\infty} u_k = \sum_{k=m+2}^{\infty} \tilde{\lambda}^{k-m} \left[ \frac{\tilde{\lambda}^m}{\prod_{\ell=1}^{m}(1+g_\ell)} + \sum_{j=1}^{m} \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^{m} f_\ell\right)}{\prod_{\ell=j}^{m}(1+g_\ell)} \right]$$
$$= \frac{\tilde{\lambda}^2}{1-\tilde{\lambda}} \left[ \frac{\tilde{\lambda}^m}{\prod_{\ell=1}^{m}(1+g_\ell)} + \sum_{j=1}^{m} \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^{m} f_\ell\right)}{\prod_{\ell=j}^{m}(1+g_\ell)} \right].$$

Hence

$$E[Q] = \frac{1}{\delta_1}\left((\delta_1 + g_1)\lambda + \sum_{\ell=2}^{m} f_\ell\right)$$
$$+ \frac{\tilde{\lambda}^2}{1-\tilde{\lambda}} \left[ \frac{\tilde{\lambda}^m}{\prod_{\ell=1}^{m}(1+g_\ell)} + \sum_{j=1}^{m} \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^{m} f_\ell\right)}{\prod_{\ell=j}^{m}(1+g_\ell)} \right].$$

Using Little's law finishes the proof. □

If $\tilde{\lambda}$ is known we thus also get an exact recursive formula for the mean response time.

## 11.4 PUT($m$) policy

### 11.4.1 Phase-type distributed job sizes

Similarly to Subsection 11.3.1, the cavity queue of the PUT($m$) policy, for $m > 1$, can be described using the CTMC with rate matrix (11.4) but with different matrices $Q_{i,j}$.

A job completion decreases the level by one, hence $Q_{k,k-1} = s^*\alpha$ for $k > 1$ and $Q_{1,0} = s^*$.

The dispatcher sends update requests to the queue at rate $\delta$. If there are at least $m$ jobs present in the cavity queue upon an update, no change occurs. Otherwise, the queue gets bumped up to level $m$. This implies $Q_{0,m} = \delta\alpha$ and $Q_{k,m} = \delta I$ for $1 \leq k \leq m-2$. The remaining incoming jobs get assigned randomly and increase the level by one, which means that $Q_{k,k+1} = \tilde{\lambda} I$ for $k \in \mathbb{N} \setminus \{0, m-1\}$ and $Q_{0,1} = \tilde{\lambda}\alpha$. The level can increase from $m-1$ to $m$ both due to server updates and remaining arrivals, meaning $Q_{m-1,m} = (\delta + \tilde{\lambda})I$.

As we have no other transitions in level 0, $Q_{0,0} = -(\delta + \tilde{\lambda})$. In non-zero levels transitions also occur due to phase changes, hence $Q_{k,k} = S - (\delta + \tilde{\lambda})I$ for $k = 1, \ldots, m-1$, where $-(\delta + \tilde{\lambda})I$ is due to the updates and extra arrivals. When the level is at least $m$ updates do not change the level, hence $Q_{k,k} = S - \tilde{\lambda} I$ for $k \geq m$.

The queue at the cavity of the PUT(1) policy is given by (11.5) with $Q_{0,0} = -(\delta + \tilde{\lambda})$, $Q_{0,1} = (\delta + \tilde{\lambda})\alpha$, $Q_{1,0} = s^*$. Further for $k \geq 1$ we have $Q_{k,k} = S - (\delta + \tilde{\lambda})I$ and $Q_{k,k+1} = \tilde{\lambda} I$, and $Q_{k,k-1} = s^*\alpha$ for $k > 1$.

We still need to determine $\tilde{\lambda}$. When the queue has $k < m$ jobs and an update occurs, the dispatcher will assign $m - k$ jobs to the queue. This implies the following relation:

$$\tilde{\lambda} = \lambda - \delta m \pi_0 - \delta \sum_{k=1}^{m-1} \sum_{\ell=1}^{n_s} (m-k)\pi_{k,\ell}.$$

We can now numerically obtain the stationary distribution analogously to Subsection 11.3.1. If $m = 1$, we can omit using the bisection algorithm as

$$\tilde{\lambda} = \lambda - \delta\pi_0 = \lambda - \delta(1-\lambda). \tag{11.19}$$

In this case we can immediately determine the invariant distribution as $\pi_0 = 1 - \lambda$ and, due to [61, Section 2],

$$\pi_1 = \pi_0 Q_{0,1} N = (1-\lambda)(\delta + \tilde{\lambda})\alpha N = \lambda(1-\lambda)(1+\delta)\alpha N.$$

Note further, that for $m = 1$ this policy coincides with JUT(1) policy, except that for the same $\delta$, the latter jumps from state 0 at a rate $\delta_0 = \delta/(1-\lambda)$, which is greater than $\delta$. Therefore, PUT(1) policy will always be outperformed by the JUT(1) policy.

## 11.4.2 Exponential job sizes

Similarly to Subsection 11.3.2, we can find recursive formulas which allow us to omit solving (11.8) for PUT($m$) policy and exponential job sizes (with mean 1). To this end we define recursively

$$\bar{g}_m = \delta, \quad \bar{g}_i = \delta + \frac{\tilde{\lambda}\bar{g}_{i+1}}{1 + \bar{g}_{i+1}} \tag{11.20}$$

and

$$\bar{f}_m = \frac{\delta^2}{1+\delta}, \quad \bar{f}_i = \frac{\bar{g}_i}{1+\bar{g}_i}\left((m-i+1)\delta - \sum_{j=i+1}^{m} \bar{f}_j\right) \tag{11.21}$$

for $i = 1, \ldots, m-1$. For ease of notation, we set $\bar{g}_{m+1} = 0$. Note that (11.20) and (11.21) only depend on $\delta, \tilde{\lambda}$ and $m$. We can now prove the following:

**Lemma 11.4.1.** *For the PUT(m) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, the following relations hold for $k = 1, \ldots, m$:*

$$u_{k-1}\tilde{\lambda} + (m-k+1)\delta = (1+\bar{g}_k)u_k + \sum_{\ell=k+1}^{m} \bar{f}_\ell, \tag{11.22}$$

$$\delta \sum_{\ell=k+1}^{m} u_\ell = \frac{\tilde{\lambda}\bar{g}_{k+1}}{1+\bar{g}_{k+1}} u_k + \sum_{\ell=k+1}^{m} \bar{f}_\ell. \tag{11.23}$$

*Proof.* For every $k \geq 1$ we have the following detailed balanced equations (see the proof of Theorem 3.2.3), which are explained below:

$$(u_{k-1} - u_k)\tilde{\lambda} + (1-u_k)\delta 1[k \leq m] = u_k - u_{k+1}. \tag{11.24}$$

The LHS describes the up-crossing rate over threshold $k$, while the RHS denotes down-crossing rate under $k$. The queue up-crosses the threshold $k$ due to batch arrivals if $k \leq m$, namely due to queues with less than $m$ jobs sending updates. This gives the term $(1-u_k)\delta$. An up-crossing can also occur for any $k$ due to the additional arrivals with rate $\tilde{\lambda}$ in queues with exactly $k-1$ jobs, which is accounted for by the term $(u_{k-1} - u_k)\tilde{\lambda}$. A down-crossing can only occur due to a completion in queue with exactly $k$ jobs, which gives the term $u_k - u_{k+1}$.

Summing Equations (11.24) from $k$ to infinity for every $k \geq 1$, we get

$$u_{k-1}\tilde{\lambda} + \delta \sum_{\ell=k}^{m} (1-u_\ell) = u_k, \tag{11.25}$$

which is equivalent to

$$u_{k-1}\tilde{\lambda} + \max(m-k+1, 0)\delta = (1 + 1[k \leq m]\delta)u_k + \delta \sum_{\ell=k+1}^{m} u_\ell.$$

Assume now that $k \leq m$. We prove both (11.22) and (11.23) at the same time using backwards induction. For $k = m$, (11.22) follows immediately from (11.25), while (11.23) holds trivially. For $k = m-1$, we have due to (11.22) with $k = m$:

$$\delta u_m = \bar{g}_m u_m = \frac{\bar{g}_m}{1+\bar{g}_m}(u_{m-1}\tilde{\lambda} + \delta),$$

which is exactly (11.23). We also have

$$u_{m-2}\tilde{\lambda} + 2\delta = (1+\delta)u_{m-1} + \delta u_m,$$

which due to (11.23) with $k = m-1$ further equals

$$(1+\delta)u_{m-1} + \frac{\tilde{\lambda}\bar{g}_m}{1+\bar{g}_m}u_{m-1} + \bar{f}_m = (1+\bar{g}_{m-1})u_{m-1} + \bar{f}_m.$$

Now suppose that the two claims hold for $k = k'+1, \ldots, m$. We need to show that they hold for $k = k'$. By using the induction hypothesis of (11.23) and (11.22) respectively, we get

$$\begin{aligned}
\delta \sum_{\ell=k+1}^{m} u_\ell &= \delta u_{k+1} + \frac{\tilde{\lambda}\bar{g}_{k+2}}{1+\bar{g}_{k+2}}u_{k+1} + \sum_{\ell=k+2}^{m} \bar{f}_\ell \\
&= \bar{g}_{k+1}u_{k+1} + \sum_{\ell=k+2}^{m} \bar{f}_\ell \\
&= \frac{\bar{g}_{k+1}}{1+\bar{g}_{k+1}}\left(u_k\tilde{\lambda} + (m-k)\delta - \sum_{\ell=k+2}^{m} \bar{f}_\ell\right) + \sum_{\ell=k+2}^{m} \bar{f}_\ell \\
&= \frac{\tilde{\lambda}\bar{g}_{k+1}}{1+\bar{g}_{k+1}}u_k + \sum_{\ell=k+1}^{m} \bar{f}_\ell,
\end{aligned}$$

which shows (11.23) for $k = k'$. By using (11.23), we further have

$$\begin{aligned} u_{k-1}\tilde{\lambda} + (m-k+1)\delta &= (1+\delta)u_k + \delta \sum_{\ell=k+1}^{m} u_\ell \\ &= (1+\delta)u_k + \frac{\tilde{\lambda}\bar{g}_{k+1}}{1+\bar{g}_{k+1}} u_k + \sum_{\ell=k+1}^{m} \bar{f}_\ell \\ &= (1+\bar{g}_k)u_k + \sum_{\ell=k+1}^{m} \bar{f}_\ell, \end{aligned}$$

which shows (11.22) for $k = k'$, thus finishing the proof. □

As $u_0 = 1$ and $u_1 = 1 - \pi_0 = \lambda$, we get from (11.22)

$$\tilde{\lambda} + m\delta = (1+\bar{g}_1)\lambda + \sum_{\ell=2}^{m} \bar{f}_\ell. \tag{11.26}$$

Using (11.1), this implies that the average number of slots discovered per update is:

$$\eta(m) = m - \frac{1}{\delta}\left(\bar{g}_1\lambda + \sum_{\ell=2}^{m} \bar{f}_\ell\right).$$

Unsurprisingly, $\eta(m) < m$. Note, that for $m = 1$, Equation (11.26) implies $\tilde{\lambda} = \lambda - \delta(1-\lambda)$, which is in agreement with (11.19). This implies that we need $\delta < -1 + 1/(1-\lambda)$ for $\tilde{\lambda}$ to be non-zero. Similarly, for $m = 2$, we obtain

$$\tilde{\lambda} = \frac{\lambda - \delta(2+\delta)(1-\lambda)}{1+\delta(1-\lambda)}.$$

Setting $\tilde{\lambda} > 0$ in the last equation and solving it for $\delta$, we obtain that we must have $\delta < -1 + 1/\sqrt{1-\lambda}$. Solving Equation (11.26) for $m = 3$ requires considerably more work. For $m = 3$, $\tilde{\lambda}$ is given by

$$\tilde{\lambda} = -(1+\delta) + \frac{-1 + \sqrt{4\delta(1-\lambda)[-\delta^2(1-\lambda) + 2\delta\lambda - \delta + \lambda + 1] + 1}}{2\delta(1-\lambda)}.$$

Setting $\tilde{\lambda} > 0$ in the last equation and solving for $\delta$, we get the following condition for $\tilde{\lambda} > 0$: $\delta < -1 + 1/\sqrt[3]{1-\lambda}$. This suggests the following theorem:

**Theorem 11.4.2.** *For the PUT(m) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, we have $\tilde{\lambda} > 0$ if and only if $\delta < -1 + 1/\sqrt[m]{1-\lambda}$.*

*Proof.* Clearly, $\delta\eta(m)$ is strictly increasing in function of $\delta$. Hence, there exists a unique value of $\delta$ such that $\lambda = \delta\eta(m)$, or equivalently, $\tilde{\lambda} = 0$. Suppose that $\tilde{\lambda} = 0$. Then the PUT($m$) policy with exponential job sizes (with mean one) coincides with the AUJSQ$^{\text{exp}}(\delta)$ policy from [74] with $\nu = 0$ and maximum queue length $m$. But the latter system only has $\nu = 0$ and maximum queue length $m$ if $1/(1+\delta)^m = 1 - \lambda$ [74, p.19], which is equivalent to $\delta = -1 + 1/\sqrt[m]{1-\lambda}$.

Alternatively, this also follows by solving (11.3) for $\delta$, as for exponential job sizes with mean 1, we have $y = 1/(1+\delta)$. □

*Remark* 11.4.3. For $1 \leq m < M$, one easily checks that $1/\sqrt[m]{1-\lambda} > 1/\sqrt[M]{1-\lambda}$, for every $0 < \lambda < 1$. This implies that for the PUT($m$) policy with exponential job sizes of mean one, the maximum value of $\delta$ such that $\tilde{\lambda} > 0$ is decreasing in function of $m$.

From Lemma 11.4.1, we obtain recursive formulas that determine the invariant distribution, if $\tilde{\lambda}$ is known.

**Proposition 11.4.4.** *For the PUT(m) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, the probability $u_k$ that the queue length exceeds $k$ is given by:*

$$u_k = \frac{\tilde{\lambda}^k}{\prod_{\ell=1}^k (1+\bar{g}_\ell)} + \sum_{j=1}^k \frac{\tilde{\lambda}^{k-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m \bar{f}_\ell\right)}{\prod_{\ell=j}^k (1+\bar{g}_\ell)}, \qquad \text{if } 1 \leq k \leq m, \tag{11.27}$$

$$u_k = \tilde{\lambda}^{k-m}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+\bar{g}_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m \bar{f}_\ell\right)}{\prod_{\ell=j}^m (1+\bar{g}_\ell)}\right], \qquad \text{if } k > m. \tag{11.28}$$

*Proof.* (11.27) follows directly from (11.22) and $u_0 = 1$, while (11.28) follows from (11.27) and (11.25). □

If $\tilde{\lambda}$ is not known, we can numerically determine $\tilde{\lambda}$ and $\pi$ using the same bisection algorithm as in Subsection 11.3.2 (except with $\bar{g}_k$'s and $\bar{f}_k$'s instead of $g_k$'s and $f_k$'s).

**Proposition 11.4.5.** *For the PUT(m) policy with exponential job sizes (with mean one) and arrival rate $\lambda$, we have*

$$E[R] = \frac{1}{\lambda\delta}\left(\bar{g}_1\lambda + \sum_{\ell=2}^m \bar{f}_\ell\right) + \frac{\tilde{\lambda}}{\lambda(1-\tilde{\lambda})}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+\bar{g}_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m \bar{f}_\ell\right)}{\prod_{\ell=j}^m (1+\bar{g}_\ell)}\right].$$

*Proof.* We first calculate $E[Q] = \sum_{k\geq 1} u_k$, the average queue length of the system. Due to (11.23) with $k = 1$ and $\delta u_1$ added to the both sides of the expression, we have

$$\sum_{k=1}^m u_k = \frac{1}{\delta}\left(\left(\delta + \frac{\tilde{\lambda}\bar{g}_2}{1+\bar{g}_2}\right)u_1 + \sum_{\ell=2}^m \bar{f}_\ell\right) = \frac{1}{\delta}\left(\bar{g}_1\lambda + \sum_{\ell=2}^m \bar{f}_\ell\right).$$

Due to (11.28), we further have

$$\begin{aligned}\sum_{k=m+1}^\infty u_k &= \sum_{k=m+1}^\infty \tilde{\lambda}^{k-m}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+\bar{g}_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m \bar{f}_\ell\right)}{\prod_{\ell=j}^m (1+\bar{g}_\ell)}\right] \\ &= \frac{\tilde{\lambda}}{1-\tilde{\lambda}}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^m (1+\bar{g}_\ell)} + \sum_{j=1}^m \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^m \bar{f}_\ell\right)}{\prod_{\ell=j}^m (1+\bar{g}_\ell)}\right].\end{aligned}$$

Hence

$$E[Q] = \frac{1}{\delta}\left(\bar{g}_1\lambda + \sum_{\ell=2}^{m} \bar{f}_\ell\right) + \frac{\tilde{\lambda}}{1-\tilde{\lambda}}\left[\frac{\tilde{\lambda}^m}{\prod_{\ell=1}^{m}(1+\bar{g}_\ell)} + \sum_{j=1}^{m} \frac{\tilde{\lambda}^{m-j}\left((m-j+1)\delta - \sum_{\ell=j+1}^{m}\bar{f}_\ell\right)}{\prod_{\ell=j}^{m}(1+\bar{g}_\ell)}\right].$$

Using Little's law now finishes the proof. □

Hence, if $\tilde{\lambda}$ is known, we can quickly calculate the mean response time.

## 11.5 Model Validation

Using simulation, we show in this section that as $N \to \infty$, the relative error between the measured mean response time in the system of $N$ servers and the mean response time of the queue at the cavity approaches zero. We perform the simulations for different parameters, different job requirement distributions and for $N \in \{10^2, 10^3, 10^4, 10^5\}$. Contrary to Section 10.3 all job size distributions are examples of distributions of the phase type, namely exponential, hyperexponential, Erlang and hyper-Erlang distributions. Note, that we describe the hyperexponential distributions used here through the parameters $E[X], f, SCV$ (c.f. Subsection 2.3.2). Each simulation is run until $1000N$ arrivals occurred, with a warm-up period of 10% of the arrivals. For every setting, the measured mean response time and 95% confidence intervals are calculated based on 20 runs. The simulation results can be found in Tables 11.1 and 11.2. For all examples presented here, except for JUT($m, \delta_1$) with exponential job sizes and $N = 100$, the relative error is less than 2%. Further, the measured mean response time seems to be $O(1/N)$ accurate, similar to the results in [24]. Based on simulations presented here and other simulations we also conclude that the error is increasing as $\lambda$ or the job size variability increases.

## 11.6 Numerical Experiments

In this section we present several numerical experiments on the policies in this part, which we will call Join-Up-To-$m$ policies. We first show that there exist parameter settings such that any Join-Up-To-$m$ policy can outperform any other (Example 11.6.1). In Example 11.6.2, we then show a comparison of performances of the three policies if the parameter $m$ is chosen optimally. Note that in Example 11.6.2 we also check for parameter settings where the Join-Up-To-$m$ policies become hyperscalable. Finally, in Example 11.6.4 we remark on what happens if we do not allow parameter settings where the policies become hyperscalable.

**Example 11.6.1.** In [36] we have shown that the hyperscalable pull policy with $\delta_1 = 0$ always outperforms the hyperscalable pull policy with $\delta_1 \neq 0$ and we conjectured that the latter always outperforms the hyperscalable push policy. In this example, we show that the above ordering does not necessarily hold for the Join-Up-To-$m$ counterparts of the hyperscalable policies, if we choose $m$ the same for the three Join-Up-To-$m$ policies. In fact, we present examples that show that every ordering is possible.

| settings | N | sim. ± conf. | rel.err.% |
|---|---|---|---|
| Exponential | 100 | 4.4892 ± 2.01e-01 | 25.3970 |
| $\lambda = 0.99$ | 1000 | 3.6394 ± 3.20e-02 | 1.6592 |
| $\delta = 0.2, \delta_1 = 0.1$ | 10000 | 3.5864 ± 8.29e-03 | 0.1798 |
| $m = 5$ | 100000 | 3.5811 ± 2.67e-03 | 0.0309 |
| | $\infty$ | 3.5800 | 0 |
| HypExp(2) | 100 | 9.9324 ± 2.72e-02 | 3.4494 |
| $f = 1/3, SCV = 5$ | 1000 | 10.2458 ± 7.01e-03 | 0.4034 |
| $\lambda = 0.85$ | 10000 | 10.2833 ± 2.22e-03 | 0.0382 |
| $\delta = 0.1, \delta_1 = 0.1$ | 100000 | 10.2871 ± 5.84e-04 | 0.0020 |
| $m = 15$ | $\infty$ | 10.2873 | 0 |
| Erlang(10) | 100 | 5.8629 ± 2.29e-02 | 3.7044 |
| $\lambda = 0.95$ | 1000 | 6.0710 ± 3.49e-03 | 0.2863 |
| $\delta = 0.1, \delta_1 = 0.05$ | 10000 | 6.0875 ± 1.07e-03 | 0.0157 |
| $m = 10$ | 100000 | 6.0884 ± 3.54e-04 | 0.0005 |
| | $\infty$ | 6.0885 | 0 |
| HypErl(2,5) | 100 | 2.6695 ± 2.42e-03 | 1.5137 |
| $p = 0.85$ | 1000 | 2.7065 ± 9.62e-04 | 0.1474 |
| $\lambda = 0.8$ | 10000 | 2.7102 ± 3.43e-04 | 0.0104 |
| $\delta = 0.15, \delta_1 = 0.05$ | 100000 | 2.7105 ± 8.64e-05 | 0.0007 |
| $m = 4$ | $\infty$ | 2.7105 | 0 |

Table 11.1: Relative error of the simulated mean response time for the JUT($m, \delta_1$) strategy based on 20 runs.

Let $E[R_{\text{JUT}(m)}]$, $E[R_{\text{JUT}(m,\delta_1)}]$ and $E[R_{\text{PUT}(m)}]$ denote the mean response times of the JUT($m$), JUT($m, \delta_1$) and PUT($m$) policies respectively. Due to Remark 10.4.10 when $m < 1 + \frac{\lambda}{1-\lambda}E[G^2]$ (where $G$ denotes the job size distribution), we can expect that the higher the rate of discovering slots, the lower the mean response time of the policies. This implies that $E[R_{\text{JUT}(m)}] < E[R_{\text{JUT}(m,\delta_1)}]$ when $m < 1 + \frac{\lambda}{1-\lambda}E[G^2]$ as on average more slots get discovered by the JUT($m$) policy than by the JUT($m, \delta_1$) policy. In this case, we can also expect that for $\delta_1$ close to 0 we have $E[R_{\text{JUT}(m,\delta_1)}] < E[R_{\text{PUT}(m)}]$ as in this case JUT($m, \delta_1$) policy performs close to the JUT($m$) policy. This suggests that we need to consider systems where $m > 1 + \frac{\lambda}{1-\lambda}E[G^2]$. Note, that to this end we have to choose $m$ sub-optimally for the JUT($m$) policy.

In Figure 11.1 we plot the mean response times for $\lambda = 0.5$, $\delta = 0.05$, $\delta_1 \in [0, \delta]$, $m = 2$ and Erlang(15) job sizes. For $\delta_1 < 0.023$, the figure shows that $E[R_{\text{JUT}(2)}] < E[R_{\text{JUT}(2,\delta_1)}] < E[R_{\text{PUT}(2)}]$, while for larger values of $\delta_1$, we have $E[R_{\text{JUT}(2)}] < E[R_{\text{PUT}(2)}] < E[R_{\text{JUT}(2,\delta_1)}]$. Note, that for the JUT($m$) policy in this figure, we have $m < 1 + \frac{\lambda}{1-\lambda}E[G^2]$.

In Figure 11.2 we plot the mean response times for $\lambda = 0.4$, $\delta = 0.05$, $\delta_1 \in [0, \delta]$, $m = 7$ and HypExp(2) job sizes, with $f = 1/2$ and $SCV = 1.8$. First note that, although it is not clear from graph, for $0 < \delta_1 \leq 0.004$ we have $E[R_{\text{PUT}(7)}] < E[R_{\text{JUT}(7)}] < E[R_{\text{JUT}(7,\delta_1)}]$. For $0.005 \leq \delta_1 < 0.047$, the figure shows that $E[R_{\text{PUT}(7)}] < E[R_{\text{JUT}(7,\delta_1)}] < E[R_{\text{JUT}(7)}]$, while for $\delta_1 \geq 0.048$, we have $E[R_{\text{JUT}(7,\delta_1)}] < E[R_{\text{PUT}(7)}] < E[R_{\text{JUT}(7)}]$.

Finally, in Figure 11.3 we plot the mean response times for $\lambda = 0.5$, $\delta = 0.04$, $\delta_1 \in [0, \delta]$, $m = 10$ and HypExp(2) job sizes, with $f = 1/2$ and $SCV = 2$. For $\delta_1 \leq 0.034$, the

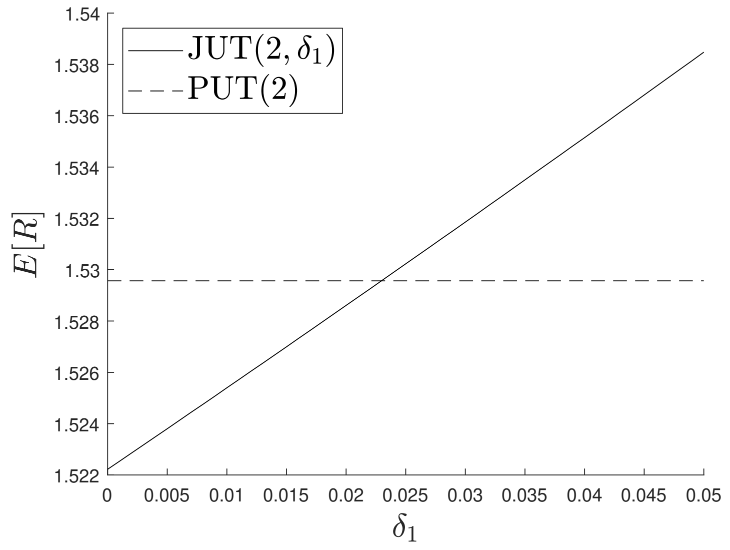


Figure 11.1: Example 11.6.1: a comparison of Join-Up-To-$m$ policies for $\lambda = 0.5$, $\delta = 0.05$, $\delta_1 \in [0, \delta]$, $m = 2$ and Erlang(15) job sizes.

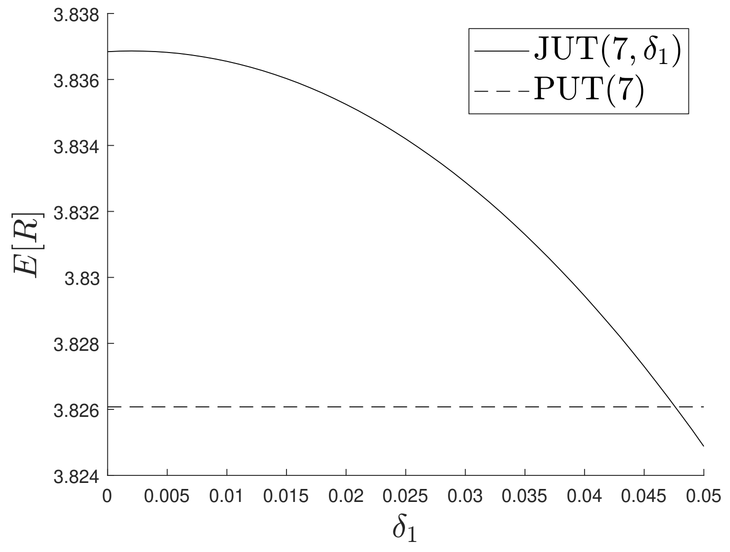


Figure 11.2: Example 11.6.1: a comparison of Join-Up-To-$m$ policies for $\lambda = 0.4$, $\delta = 0.05$, $\delta_1 \in [0, \delta]$, $m = 7$ and HypExp(2) job sizes, with $f = 1/2$ and $SCV = 1.8$.

| settings | N | sim. ± conf. | rel.err.% |
|---|---|---|---|
| Exponential | 100 | 5.2676 ± 6.36e-03 | 0.7247 |
| $\lambda = 0.8$ | 1000 | 5.3019 ± 2.24e-03 | 0.0768 |
| $\delta = 0.1$ | 10000 | 5.3054 ± 8.98e-04 | 0.0111 |
| $m = 8$ | 100000 | 5.3059 ± 3.09e-04 | 0.0015 |
| | $\infty$ | 5.3060 | 0 |
| HypExp(2) | 100 | 7.8233 ± 4.10e-02 | 1.5167 |
| $f = 1/2, SCV = 10$ | 1000 | 7.9298 ± 8.57e-03 | 0.1757 |
| $\lambda = 0.7$ | 10000 | 7.9427 ± 3.88e-03 | 0.0128 |
| $\delta = 0.1$ | 100000 | 7.9445 ± 9.81e-04 | 0.0093 |
| $m = 10$ | $\infty$ | 7.9438 | 0 |
| Erlang(5) | 100 | 5.5330 ± 7.51e-03 | 1.1378 |
| $\lambda = 0.9$ | 1000 | 5.5903 ± 2.00e-03 | 0.1143 |
| $\delta = 0.2$ | 10000 | 5.5957 ± 5.83e-04 | 0.0181 |
| $m = 8$ | 100000 | 5.5967 ± 1.78e-04 | 0.0001 |
| | $\infty$ | 5.5967 | 0 |
| HypErl(3,7) | 100 | 7.6313 ± 1.22e-02 | 0.4069 |
| $p = 0.75$ | 1000 | 7.6571 ± 5.63e-03 | 0.0703 |
| $\lambda = 0.9$ | 10000 | 7.6631 ± 1.02e-03 | 0.0076 |
| $\delta = 0.05$ | 100000 | 7.6629 ± 4.17e-04 | 0.0051 |
| $m = 12$ | $\infty$ | 7.6625 | 0 |

Table 11.2: Relative error of the simulated mean response time for the PUT($m$) strategy based on 20 runs.

figure shows that $E[R_{\text{JUT}(10)}] < E[R_{\text{JUT}(10,\delta_1)}] < E[R_{\text{PUT}(10)}]$, while for $\delta_1 \geq 0.035$, we have $E[R_{\text{JUT}(10,\delta_1)}] < E[R_{\text{JUT}(10)}] < E[R_{\text{PUT}(10)}]$.

Note, that for the JUT($m$) policy in the last two figures, we had $m > 1 + \frac{\lambda}{1-\lambda}E[G^2]$.

In the last example, the parameter $m$ was chosen to be the same for the three Join-Up-To-$m$ policies. We now present examples where the parameter $m$ is chosen optimally for each policy.

**Example 11.6.2.** In this example we compare the performance of the different Join-Up-To-$m$ strategies when the parameter $m$ is chosen optimally, where we allow the policies to reduce to the hyperscalable policies from [36]. We do this for $\lambda \in [0.5, 0.95]$, $\delta \in \{0.01, 0.1, 0.3\}$, $\delta_1 = \delta$ and HypExp(2) jobs with balanced means and $SCV = 20$.

We determine the optimal value of $m$ as follows. We first note, that if we set the parameter $m$ too high, then a Join-Up-To-$m$ policy reduces to the respective hyperscalable policy from [36]. In other words, for every parameter setting there exist only a finite number of possible values for $m$ before a Join-Up-To-$m$ policy becomes hyperscalable. For a given parameter setting and a given Join-Up-To-$m$ policy, we can thus determine the optimal value of $m$ by brute force. We then compare the mean response time for that value of $m$ with the mean response time of the respective hyperscalable policy. We call the minimum of these two mean response times and the corresponding value of $m$ optimal. Note, that if the optimal $m$-value comes from a hyperscalable policy, then the maximal queue length of the cavity queue for that policy is $m + 1$.

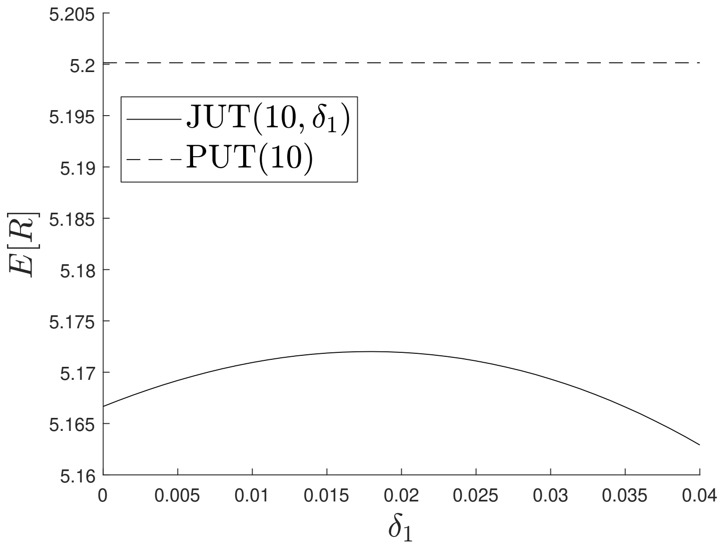


Figure 11.3: Example 11.6.1: a comparison of Join-Up-To-$m$ policies for $\lambda = 0.5$, $\delta = 0.04$, $\delta_1 \in [0, \delta]$, $m = 10$ and HypExp(2) job sizes, with $f = 1/2$ and $SCV = 2$.

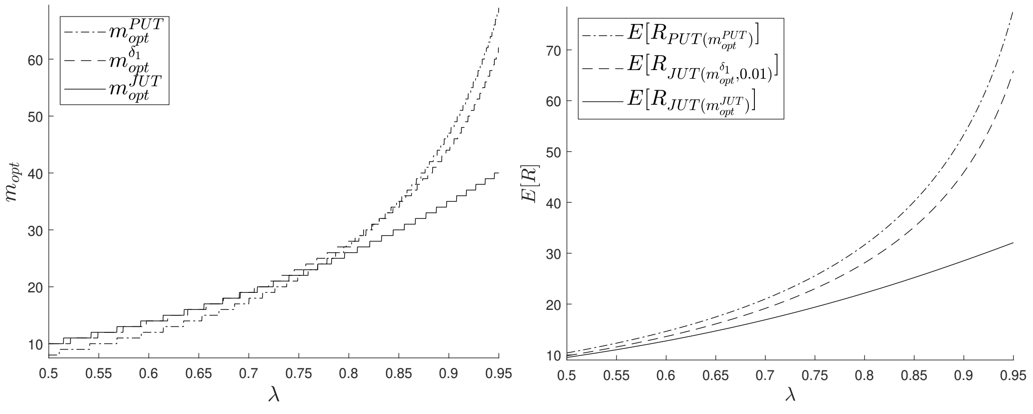


Figure 11.4: Example 11.6.2: a comparison of Join-Up-To-$m$ policies for $\lambda \in [0.5, 0.95]$, $\delta = 0.01$, $\delta_1 = 0.01$ and HypExp(2) jobs with $f = 1/2$ and $SCV = 20$. The left plot shows the values of $m_{opt}^{PUT}$, $m_{opt}^{\delta_1}$ and $m_{opt}^{JUT}$, while the right plot shows the corresponding mean response times.

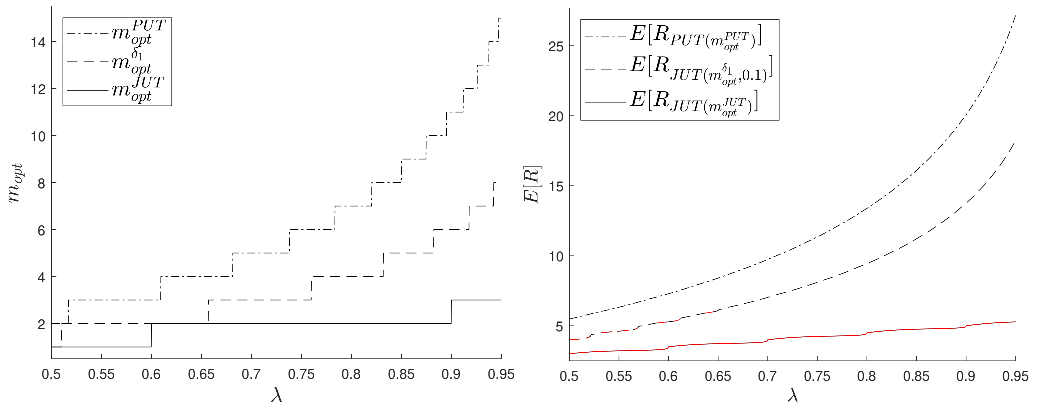


Figure 11.5: Example 11.6.2: a comparison of Join-Up-To-$m$ policies for $\lambda \in [0.5, 0.95]$, $\delta = 0.1$, $\delta_1 = 0.1$ and HypExp(2) jobs with $f = 1/2$ and $SCV = 20$. The left plot shows the values of $m_{opt}^{PUT}$, $m_{opt}^{\delta_1}$ and $m_{opt}^{JUT}$, while the right plot shows the corresponding mean response times.

We plot the optimal values of $m$ in Figures 11.4-11.6 (left) and the mean response times for these values of $m$ (right), where Figures 11.4-11.6 show the results for $\delta \in \{0.01, 0.1, 0.3\}$ respectively.

Let us denote by $m_{opt}^{JUT}$, $m_{opt}^{\delta_1}$ and $m_{opt}^{PUT}$ the optimal values of $m$ of the JUT($m$), JUT($m, \delta_1$) and PUT($m$) policies respectively and by $E[R_{PUT(m_{opt}^{PUT})}]$, $E[R_{JUT(m_{opt}^{\delta_1},\delta_1)}]$ and $E[R_{JUT(m_{opt}^{JUT})}]$ the respective mean response times.

The figures show that for each policy and value of $\delta$ the values of $m_{opt}^{JUT}$, $m_{opt}^{\delta_1}$ and $m_{opt}^{PUT}$ are increasing in function of $\lambda$, except for the JUT($m, \delta_1$) policy with $\delta = 0.1$. Further, for every parameter setting there exists a $\tilde{\lambda}$ such that for every $\lambda \geq \tilde{\lambda}$ we have

$$m_{opt}^{JUT} \leq m_{opt}^{\delta_1} \leq m_{opt}^{PUT}.$$

In all three figures the mean response time is increasing as $\lambda$ increases. Further, as $\lambda$ gets close to 1 the mean response times stay finite except for the push-up-to-$m$/hyperscalable push policy. Note, that the discontinuities in the graph of the mean response times occur at values of $\lambda$, where the optimal value of $m$ changes.

For $\delta = 0.01$, the Join-Up-To-$m$ policies clearly outperform the hyperscalable ones. This is due to the fact that for hyperscalable policies, the mean response times become infinite when $\delta \longrightarrow 0$, while the Join-Up-To-$m$ policies reduce to random assignment. In case of $\delta = 0.3$, the hyperscalable policies outperform the respective Join-Up-To-$m$ policies, as the former perform very well for high update rates $\delta$. The figure with $\delta = 0.1$ falls in between the two former cases: JUT($m$) policy is outperformed by the hyperscalable pull policy with $\delta_1 = 0$, while the other Join-Up-To-$m$ policies (mostly) outperform their hyperscalable counterparts. Finally, the example suggests the following conjecture, in agreement with the results in [36]:

**Conjecture 11.6.3.** *For every $\lambda, \delta, \delta_1$ and job size distribution, we have*

$$E[R_{PUT(m_{opt}^{PUT})}] > E[R_{JUT(m_{opt}^{\delta_1},\delta_1)}] > E[R_{JUT(m_{opt}^{JUT})}].$$

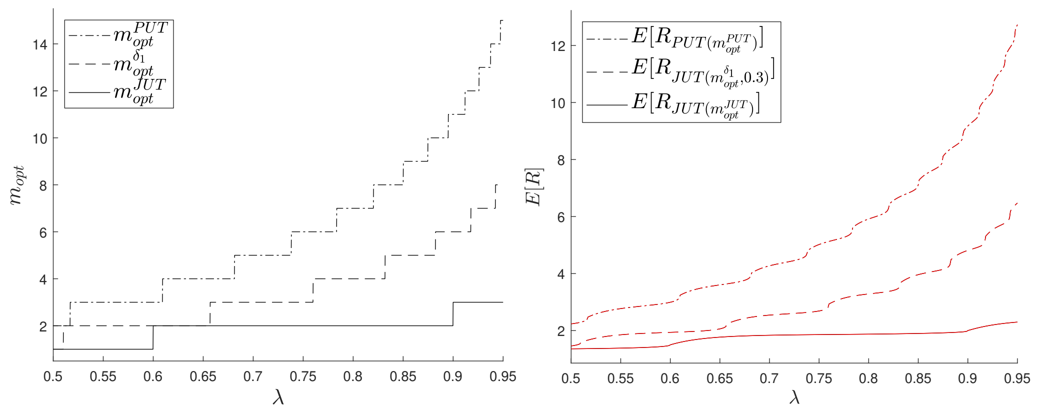


Figure 11.6: Example 11.6.2: a comparison of Join-Up-To-$m$ policies for $\lambda \in [0.5, 0.95]$, $\delta = 0.3$, $\delta_1 = 0.3$ and HypExp(2) jobs with $f = 1/2$ and $SCV = 20$. The left plot shows the values of $m_{opt}^{PUT}$, $m_{opt}^{\delta_1}$ and $m_{opt}^{JUT}$, while the right plot shows the corresponding mean response times.

The above conjecture implies that for every $\lambda$ and every job size distribution, the optimal policy is either JUT($m$) with $m$ optimal or the hyperscalable pull policy from [36] with $\delta_1 = 0$. But what happens if we do not allow the Join-Up-To-$m$ policies to reduce to their hyperscalable counterparts? This is what we examine in the next example.

**Example 11.6.4.** The last example suggested that if we choose the value of $m$ optimally and allow the policies to be hyperscalable, the PUT($m_{opt}^{PUT}$) policy is always outperformed by the JUT($m_{opt}^{\delta_1}, \delta_1$) policy (with $\delta_1 \neq 0$), which is in turn outperformed by the JUT($m_{opt}^{JUT}$) policy. In this example, we show that this does not hold anymore, if we do not allow the policies to be hyperscalable.

Let us denote by $\tilde{m}_{opt}^{JUT}$, $\tilde{m}_{opt}^{\delta_1}$ and $\tilde{m}_{opt}^{PUT}$ the optimal values of $m$ of the JUT($m$), JUT($m, \delta_1$) and PUT($m$) policies respectively, provided that we do not allow the policies to become hyperscalable. Similarly to the previous example, these can be determined by brute force. Let us denote by $E[R_{PUT(\tilde{m}_{opt}^{PUT})}]$, $E[R_{JUT(\tilde{m}_{opt}^{\delta_1},\delta_1)}]$ and $E[R_{JUT(\tilde{m}_{opt}^{JUT})}]$ the corresponding mean response times.

In Figures 11.7 and 11.8, we plot the values of the parameters $\tilde{m}_{opt}^{JUT}$, $\tilde{m}_{opt}^{\delta_1}$ and $\tilde{m}_{opt}^{PUT}$ (left) and the corresponding mean response times (right). In Figure 11.7, we do this for $\lambda \in [0.5, 1[$, $\delta = 0.3$, $\delta_1 = 0.1$ and HypExp(2) job sizes with balanced means and $SCV = 20$. The figure shows that

- $E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.1)}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}]$, for $\lambda \in [0.6, 0.813] \cup [0.9, 1[$;
- $E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.1)}]$, for $\lambda \in [0.814, 0.899]$;
- $E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}] > E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.1)}]$, for $\lambda \in [0.571, 0.599]$;
- $E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.1)}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}] > E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}]$, for $\lambda \in [0.517, 0.57]$;

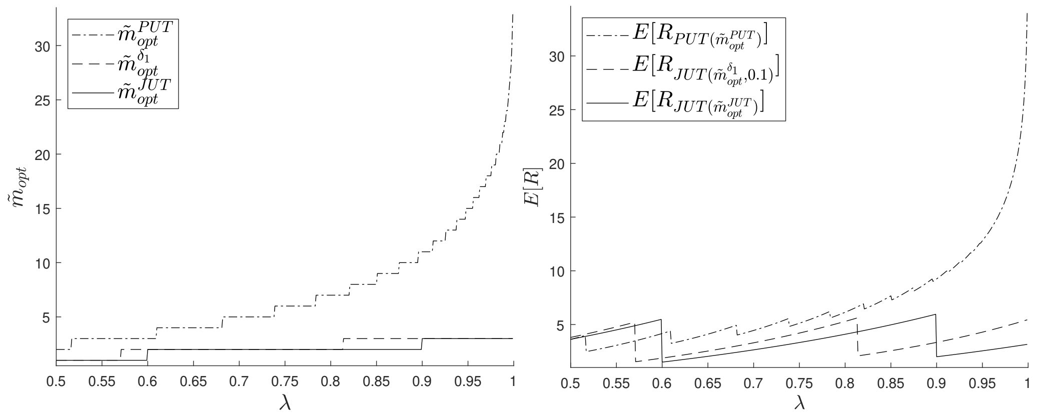


Figure 11.7: Example 11.6.4: a comparison of Join-Up-To-$m$ policies for $\lambda \in [0.5, 1[$, $\delta = 0.3$, $\delta_1 = 0.1$ and HypExp(2) jobs with $f = 1/2$ and $SCV = 20$. The left plot shows the values of $\tilde{m}_{opt}^{PUT}$, $\tilde{m}_{opt}^{\delta_1}$ and $\tilde{m}_{opt}^{JUT}$, while the right plot shows the corresponding mean response times.

- $E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.1)}] > E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}]$, for $\lambda \in [0.503, 0.516]$;

In Figure 11.8, we plot the values of $\tilde{m}_{opt}^{PUT}$, $\tilde{m}_{opt}^{\delta_1}$ and $\tilde{m}_{opt}^{JUT}$ and the corresponding mean response times for $\lambda \in [0.5, 1[$, $\delta = \delta_1 = 0.35$, HypExp(2) job sizes with balanced means and $SCV = 40$. The figure shows, among other things, that for $\lambda \in [0.65, 0.699]$, we have $E[R_{\text{JUT}(\tilde{m}_{opt}^{JUT})}] > E[R_{\text{JUT}(\tilde{m}_{opt}^{\delta_1},0.35)}] > E[R_{\text{PUT}(\tilde{m}_{opt}^{PUT})}]$. Hence, if we do not allow the Join-Up-To-$m$ policies to be hyperscalable, then every ordering is possible when the parameter $m$ is chosen optimally for each policy.

Similarly to Example 11.6.2, it is clear that, as $\lambda \to 1$, the optimal value of $m$ and the mean response time stay finite for the JUT($m$) and JUT($m, \delta_1$) policies but become infinite for the PUT($m$) policy. Further, the discontinuities in mean response times in Figures 11.7 and 11.8 occur when the optimal value of $m$ changes.

## 11.7 Conclusion

In this chapter we introduced two hyper-scalable load balancing policies, called the JUT($m, \delta_1$) and the PUT($m$) policies, where $m$ and $\delta_1$ are input parameters. Both policies can be seen as modifications of the JUT($m$) policy. The first of these policies is identical to JUT($m$), except now non-empty servers can update the dispatcher with probability $\delta_1$ upon finishing a job. The PUT($m$) policy can be seen as the "push" variant of the JUT($m$) and JUT($m, \delta_1$) policies, that is, in case of PUT($m$) it is the dispatcher that sends update requests. Under these policies incoming jobs are assigned in a greedy manner to a queue with lowest estimated number of jobs (with ties broken at random), if there exists a queue with an estimate of less than $m$ jobs. Otherwise an incoming job is assigned to a random queue.

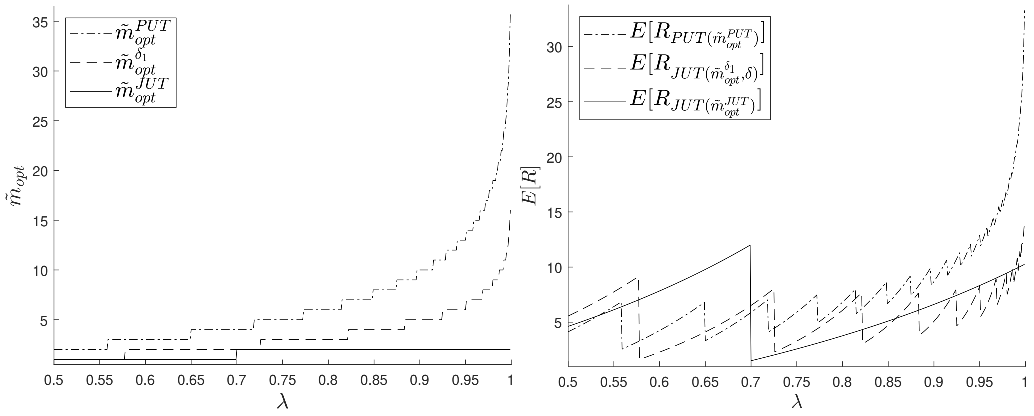


Figure 11.8: Example 11.6.4: a comparison of Join-Up-To-$m$ policies for $\lambda \in [0.5, 1[$, $\delta = \delta_1 = 0.35$ and HypExp(2) jobs with $f = 1/2$ and $SCV = 40$. The left plot shows the values of $\tilde{m}_{opt}^{PUT}$, $\tilde{m}_{opt}^{\delta_1}$ and $\tilde{m}_{opt}^{JUT}$, while the right plot shows the corresponding mean response times.

We studied the performance of the two introduced policies in a large-scale system using the queue-at-the-cavity approach, in case of jobs of the phase type. For both policies we devised a bisection algorithm to determine the invariant distribution of the queue at the cavity and the value of $\tilde{\lambda}$, that is the arrival rate of jobs that are assigned at random. For exponential job sizes, we found for both policies recursive formulas for the invariant distribution and the mean response time in the case where the value of $\tilde{\lambda}$ is known.

We demonstrated the accuracy of the cavity approach using simulations for different job size distributions and different parameter settings. Finally, we presented numerical results that show that for fixed parameter settings any policy can outperform any other. This is also true when the value of the parameter $m$ is chosen optimally for every policy if we do not allow parameter settings where the policies may be hyperscalable. If we allow the policies to be hyperscalable while choosing the parameter $m$ optimally for each policy, then the numerical results suggest that the PUT($m$) policy is always outperformed by the JUT($m, \delta_1$) policy (with $\delta_1 \neq 0$), which is in turn outperformed by the JUT($m$) policy.

In this and the previous chapter, we studied systems using cavity queues where, after an exponential amount of time, the queue jumped from queue length 0 to $m$ (JUT($m$) and JUT($m, \delta_1$) policies) and where the queue would jump from queue length smaller than $m$ to $m$ (PUT($m$) policy). In future research we could consider generalizations where it now takes a PH-distributed amount of time for such a jump to occur.

The cavity queues of the policies presented in this chapter can be seen as examples QBD Markov chains with a jump (or, in case of exponential job sizes, BD Markov chains with a jump) to level $m$. Future research could also be directed at finding explicit formulas for the invariant distribution of BD Markov chains with a jump and at finding a matrix analytic method for calculating the invariant distribution of a QBD Markov chains with a jump.

# Appendix A
# Abbreviations

This Appendix contains a list of all the abbreviations used throughout the thesis.

**Markov chains, queues and policies**

| | |
|---|---|
| **MC** | Markov Chain |
| **DTMC** | Discrete Time Markov Chain |
| **CTMC** | Continuous Time Markov Chain, see Sections 2.2, 3.2 and 3.3 |
| **BD** | Birth-and-Death (Markov chain), see Section 3.2 |
| **QBD** | Quasi-Birth-Death (Markov chain), see Section 3.3 |
| **MAP** | Markovian Arrival Process/Markovian Service Process, see Section 3.4 |
| **PASTA** | Poisson-Arrivals-See-Time-Averages property, see Theorem 3.2.4 |
| **FCFS** | First Come, First Served |
| **FIFO** | First In, First Out |
| **JSQ** | Join Shortest Queue |
| **JSQ**$(d)$ | Join Shortest out of $d$ Queues |
| **PS** | Processor Sharing |
| **MD** | Monotone Deterministic |
| **BMD** | Bounded Monotone Deterministic |
| **JUT**$(m)$ | Join-Up-To $m$ |
| **JUT**$(m, \delta_1)$ | Busy Join-Up-To $m$ |
| **PUT**$(m)$ | Push-Up-To $m$ |

**Distributions and related functions**

| | |
|---|---|
| **exp** | Exponential distribution, see Section 2.1 |
| **PH** | (continuous time) Phase-type distribution, see Section 2.2 |
| **HypExp** | Hyperexponential distribution, see Subsection 2.3.2 |
| **HypErl** | Hyper-Erlang distribution, see Subsection 2.3.5 |
| **SCV** | Squared Coefficient of Variation, see Subsection 2.3.2 |
| **pdf** | Probability Density Function |
| **cdf** | Cumulative Distribution Function |
| **LST** | Laplace-Stieltjes transform, see Definition 2.5.1 |

## Other abbreviations

| | |
|---|---|
| **ODE** | Ordinary Differential Equation |
| **PDE** | Partial Differential Equation |
| **LHS** | Left Hand Side |
| **RHS** | Right Hand Side |
| **W.l.o.g.** | Without loss of generality |

Appendix B

# Symbols

This Appendix contains a list of symbols frequently used throughout the thesis.

| | |
|---|---|
| $\alpha, S$ | Unless specified otherwise, $(\alpha, S)$ are the parameters of a PH-distribution, $\alpha$ is the initial distribution vector and $S$ the transition rate matrix. |
| $h(t)$ | The state of an ODE at time $t$ |
| $\Omega$ | The state space |
| $\pi, \zeta$ | If a chapter only deals with a set of ODEs or Markov chains, then $\pi$ is used to denote the fixed point of the ODEs or the invariant distribution of the MC respectively. If a Chapter deals with both, then $\pi$ denotes the invariant distribution of the MC, while $\zeta$ the fixed point of the set of ODEs. |
| $N$ | the number of servers/queues |
| $B$ | buffer size of a queue |
| $\mathbf{d}(\cdot, \cdot)$ | the supremum metric |
| $r$ | steal rate |
| $1[A]$ | The indicator function of $A$: for a statement $A$, $1[A]$ is 1 if $A$ is true and 0 otherwise. |
| $I_n$ | the identity matrix of dimensions $n \times n$ |
| $I$ | the identity matrix (when its dimensions are clear from the context) |
| $e_i$ | a column vector of appropriate height, with a 1 in its $i$-th entry and 0s elsewhere |
| $\lambda$ | arrival rate |
| $\rho$ | Load of the system. When job requirements have mean 1, then $\rho = \lambda$. |
| $\mathbf{1}$ | a column vector of ones (when its height is clear from the context) |
| $e$ | Euler's number |
| $W$ | waiting time distribution |
| $J$ | service time distribution |
| $R, T$ | $R$ is used for the response time distribution, unless stated differently, f.e. in Part II we use $T$ for the response time, since $R$ is used for a certain matrix. |
| $E[X]$ | the expected value of the random variable $X$ |
| $vec\langle\cdot\rangle$ | The vector stacking operator, that is: for an $m \times n$ matrix $A$, $vec\langle A\rangle$ is a column vector of height $mn$ of the columns of $A$ stacked one under another (starting from the leftmost column). See also Definition 3.6.6. |
| $i{:}j$ | for integers $j \geq i$, the column vector $[i, i+1, \ldots, j]'$. |
| $\otimes$ | the Kronecker product of two matrices, see Definition 3.6.5. |
| $0_n, 1_n$ | column vectors of height $n$ of zeros and ones respectively |

| | |
|---|---|
| $0_{m,n}$ | zero matrix of size $m \times n$, i.e. $0_{m,n} = 0_m \otimes 0'_n$ |
| $G$ | job size distribution. If jobs are executed as a whole then $G = J$. |
| $G^*(s), R^*(s), Y^*(s)$ | Laplace-Stieltjes transforms of the job size distribution, the response time and residual service time respectively |
| $\delta$ | update rate |
| $\pi^a, \pi^d$ | steady state distribution measured at arrival and departure instances respectively |
| $\pi^a(z), \pi^d(z), \pi(z)$ | generating functions of $\pi^a, \pi^d$ and $\pi$ respectively |
| $\xi(z)$ | generating function of the queue length of an $M/G/1$ queue |
| $Q$ | queue length distribution |
| $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ | the sets of natural numbers, integers and real numbers respectively |
| $'$ | When used with a function $f(x)$, $f'(x)$ denotes the derivative of $f(x)$ w.r.t. the variable $x$, when used with a matrix $A$, $A'$ is the transpose of $A$. |
| $\frac{d}{dx}, \frac{\partial}{\partial x}$ | the derivative and the partial derivative w.r.t. the variable $x$ respectively |
| $\lfloor x \rfloor$ | the floor of $x$, that is, the largest integer smaller than or equal to $x$. |
| $\lceil x \rceil$ | the ceiling of $x$, that is, the smallest integer larger than or equal to $x$. |
| $n!$ | For $n \in \mathbb{N}$, $n!$ is the factorial of $n$ defined as $n(n-1)(n-2) \cdot \ldots \cdot 2 \cdot 1$. |
| $\binom{n}{k}$ | For $n, k \in \mathbb{N}$, with $k \leq n$, this denotes the combination of $k$ out of $n$ and is defined as $\frac{n!}{(n-k)!k!}$. |

# Appendix C
# Nederlandse Samenvatting

Wachtrijtheorie speelt een cruciale rol in het modelleren van systemen met congestie. Ze wordt al lang toegepast in het analyseren en in het verbeteren van de performantie van communicatiesystemen. Aangezien moderne communicatiesystemen dikwijls bestaan vanuit verscheidene heterogene componenten, kan de traditionele analyse van zulke grootschalige systemen belemmerend worden. Wanneer men zulke systemen exact probeert te analyzeren, komt men vaak het probleem van de zogenoemnde ontploffing van de statenruimte tegen.

Grootschalige systemen worden daarom gebruikelijk bestudeerd via de mean field analyse: als een systeem bestaat vanuit een groot aantal wachtrijen, kan zo'n systeem benaderd worden door een met oneindig veel wachtrijen. De analyse van het model van het latere systeem, het mean field model, is in het algemeen meer voor de hand liggend omdat ze het probleem van de ontploffing van de statenruimte omzeilt.

Het doel van deze thesis is het analyseren van en inzicht verwerven in de performantie van bestaande en nieuwe werklastverdelingmethodes door gebruik te maken van mean field modelleren. Elk hoofdstuk van deze thesis bevat de mean field analyse van een familie van systemen die gebruik maken van zulke methodes. In de analyze, worden de technieken vanuit dynamische systemen, stochastisch modelleren, kanstheorie, numerieke analyse en simulaties gebruikt.

De hoofdstukken zijn gegroepeerd in drie delen. Het eerste van die delen behandelt monotone systemen. Dat zijn systemen met een merkbare rangschikking van staten dat in de loop van de tijd behouden blijft. Het daaropvolgende deel bevat de analyse van multithreaded computersystemen met werk stelen. In die systemen, kunnen delen van een job overgedragen worden tussen wachtrijen en dus kunnen de delen van een job tegelijkertijd uitgevoerd worden in verschillende wachtrijen. In het laatste deel worden verscheidene hyperschaalbare methodes met een enkele dispatcher bestudeerd. Dit zijn methodes waar een centrale dispatcher jobs uitdeelt aan de wachtrijen, op basis van het geschatte aantal jobs in de wachtrijen. De methodes worden hyperschaalbaar genoemd als het aantal berichten dat tussen de dispatcher en de wachtrijen wordt uitgewisseld gemiddeld minder dan een bericht per job bedraagt. Voor de systemen in de laatste twee delen worden, in het geval van een eindig aantal wachtrijen, simulaties uitgevoerd om de nauwkeurigheid van de mean field benadering te meten.

# Bibliography

[1] Ivo Adan and Jacques Resing. Queueing systems. Course notes. `https://www.win.tue.nl/~iadan/queueing.pdf`, 2015. 20, 21, 26, 34

[2] J. Anselmi and F. Dufour. Power-of-d-choices with memory: Fluid limit and optimality. *Mathematics of Operations Research*, 45(3):862–888, 2020. 38, 81, 203

[3] Jonatha Anselmi. Combining Size-Based Load Balancing with Round-Robin for Scalable Low Latency. *IEEE Transactions on Parallel and Distributed Systems*, 31(4):886–896, 2020. 201

[4] N.P. Bhatia and G.P. Szegö. *Stability Theory of Dynamical Systems*. Classics in Mathematics. Springer, 2002. 39, 95

[5] D. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured markov chains solver: software tools. In *Proceeding from the 2006 workshop on Tools for solving structured Markov chains*, pages 1–14, 2006. 33, 121, 148, 176

[6] M. Bladt, , and B.F. Nielsen. *Matrix-exponential distributions in applied probability*, volume 81. Springer, 2017. 15, 126

[7] R. Blumofe and C. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM (JACM)*, 46(5):720–748, 1999. 103, 105

[8] R.D. Blumofe, C.F. Joerg, B.C Kuszmaul, C.E. Leiserson, K.H. Randall, and Y. Zhou. Cilk: An efficient multithreaded runtime system. *Journal of parallel and distributed computing*, 37(1):55–69, 1996. 103

[9] M. Bramson, Y. Lu, and B. Prabhakar. Randomized load balancing with general service time distributions. In *ACM SIGMETRICS 2010*, pages 275–286, 2010. 40, 49, 50, 143, 171, 203, 226

[10] M. Bramson, Y. Lu, and B. Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Syst.*, 71(3):247–292, 2012. 40, 42, 50

[11] Maury Bramson, Yi Lu, and Balaji Prabhakar. Decay of tails at equilibrium for FIFO join the shortest queue networks. *The Annals of Applied Probability*, 23(5), oct 2013. 40

[12] Shelby L. Brumelle. A Generalization of Erlang's Loss System to State Dependent Arrival and Service Rates. *Mathematics of Operations Research*, 3(1):10–16, 1978. 49, 75

[13] David R. Clark. A Note on the Upper-Truncated Pareto Distribution. `https://www.soa.org/globalassets/assets/files/resources/essays-monographs/2013-erm-symposium/mono-2013-as13-1-clark.pdf`, 2013. 20

[14] J.W. Cohen. *The Single Server Queue*. North-Holland Series in Applied Mathematics and Mechanics 8. North-Holland, 2 sub edition, 1982. 27, 211

[15] Matej Črepinšek and Luka Mernik. An efficient representation for solving Catalan number related problems. *International Journal of Pure and Applied Mathematics*, 56:589–604, 2009. 155

[16] D.L. Eager, E.D. Lazowska, and J. Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Perform. Eval.*, 6(1):53–68, 1986. 76, 103, 143, 171

[17] S.N. Ethier and T.C. Kurtz. *Markov processes: characterization and convergence*. Wiley, 1986. 39, 75

[18] Matteo Frigo, Charles E. Leiserson, and Keith H. Randall. The Implementation of the Cilk-5 Multithreaded Language. In *In Proceedings of the SIGPLAN '98 Conference on Program Language Design and Implementation*, pages 212–223, 1998. 103

[19] S. W. Fuhrmann. A Note on the M/G/1 Queue with Server Vacations. *Operations Research*, 32(6):1368–1373, 1984. 206

[20] S. W. Fuhrmann and R. B. Cooper. Stochastic Decompositions in the M/G/1 Queue with Generalized Vacations. *Operations Research*, 33(5):1117–1129, 1985. 206

[21] David Gamarnik, John N Tsitsiklis, and Martin Zubeldia. Delay, memory, and messaging tradeoffs in distributed service systems. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):1–12, 2016. 201

[22] A. Ganesh, S. Lilienthal, D. Manjunath, A. Proutiere, and F. Simatos. Load Balancing via Random Local Search in Closed and Open Systems. *SIGMETRICS Perform. Eval. Rev.*, 38(1):287–298, June 2010. 51

[23] Owen Garrett. NGINX and the "Power of Two Choices" Load-Balancing Algorithm. `https://www.nginx.com/blog/nginx-power-of-two-choices-load-balancing-algorithm`, 2018. 93

[24] N. Gast. Expected values estimated via mean-field approximation are $1/n$-accurate. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–26, 2017. 44, 105, 131, 157, 189, 204, 238

[25] N. Gast and B. Gaujal. A Mean Field Model of Work Stealing in Large-scale Systems. *SIGMETRICS Perform. Eval. Rev.*, 38(1):13–24, June 2010. 38, 51, 75, 81, 103, 143, 171

[26] T. Gautier, X. Besseron, and L. Pigeon. Kaapi: A thread scheduling runtime system for data flow computations on cluster of multi-processors. In *Proceedings of the 2007 international workshop on Parallel symbolic computation*, pages 15–23, 2007. 103

[27] Fablenne Gillent and Guy Latouche. Semi-explicit solutions for M/PH/1-like queuing systems. *European Journal of Operational Research*, 13:151–160, 1983. 30, 229

[28] G. Golub and C. Van Loan. *Matrix computations*, volume 3. JHU press, 2012. 119

[29] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford University Press, Oxford; New York, 3rd edition, 2001. 20

[30] Donald Gross, John F. Shortle, James M. Thompson, and Carl M. Harris. *Fundamentals of Queueing Theory*. Wiley-Interscience, USA, 4th edition, 2008. 10, 27, 34, 211

[31] David A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer, 1st edition, 1997. 35

[32] Qi-Ming He and Hanqin Zhang. On matrix exponential distributions. *Advances in Applied Probability*, 39(1):271–292, 2007. 15

[33] T. Hellemans, T. Bodas, and B. Van Houdt. Performance Analysis of Workload Dependent Load Balancing Policies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):35, 2019. 49

[34] T. Hellemans and B. Van Houdt. On the Power-of-d-choices with Least Loaded Server Selection. *Proc. ACM Meas. Anal. Comput. Syst.*, June 2018. 40, 49, 203

[35] T. Hellemans and B. Van Houdt. Analysis of redundancy(d) with identical replicas. *SIGMETRICS Perform. Eval. Rev.*, 46(3):74–79, jan 2019. 40

[36] Tim Hellemans, Grzegorz Kielanski, and B. Van Houdt. Performance of Load Balancers with Bounded Maximum Queue Length in case of Non-Exponential Job Sizes. *CoRR*, 2022. 201, 202, 203, 208, 220, 221, 222, 227, 238, 241, 243, 244

[37] Harold V. Henderson and S. R. Searle. The vec-permutation matrix, the vec operator and Kronecker products: a review. *Linear and Multilinear Algebra*, 9(4):271–288, 1981. 36

[38] M.W. Hirsch and H. Smith. Monotone dynamical systems. *Handbook of differential equations: ordinary differential equations*, 2:239—-357, 2006. 58

[39] G. Horváth, B. Van Houdt, and M. Telek. Commuting matrices in the queue length and sojourn time analysis of MAP/MAP/1 queues. *Stochastic Models*, 30(4):554–575, 2014. 125, 126

[40] I.A. Horváth, Z. Scully, and B. Van Houdt. Mean Field Analysis of Join-Below-Threshold Load Balancing for Resource Sharing Servers. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3), 2019. 40, 58

[41] David G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, 24(3):338 – 354, 1953. 23

[42] Grzegorz Kielanski, Tim Hellemans, and Benny Van Houdt. Join-Up-To(m): Improved Hyper-scalable Load Balancing. Submitted to a journal, awaiting review, 2023. 201

[43] Grzegorz Kielanski and Benny Van Houdt. On the Asymptotic Insensitivity of the Supermarket Model in Processor Sharing Systems. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(2), jun 2021. 49

[44] Grzegorz Kielanski and Benny Van Houdt. Performance Analysis of Work Stealing Strategies in Large Scale Multi-threaded Computing. In *Quantitative Evaluation of Systems*, pages 329–348, Cham, 2021. Springer International Publishing. 103, 141, 142, 169

[45] Grzegorz Kielanski and Benny Van Houdt. Performance Analysis of Work Stealing Strategies in Large Scale Multi-Threaded Computing. *ACM Trans. Model. Comput. Simul.*, 2023. 103, 169

[46] J. Kriege and P. Buchholz. *PH and MAP Fitting with Aggregated Traffic Traces*, pages 1–15. Springer International Publishing, Cham, 2014. 171

[47] T.G. Kurtz. *Approximation of population processes*, volume 36. SIAM, 1981. 105, 131

[48] G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*, volume 5. SIAM, 1999. 13, 14, 27, 30, 33, 117, 119, 128, 148, 171, 175, 214, 229

[49] Doug Lea. A Java Fork/Join Framework. In *Proceedings of the ACM 2000 Conference on Java Grande*, JAVA '00, page 36–43, New York, NY, USA, 2000. Association for Computing Machinery. 103

[50] Daan Leijen, Wolfram Schulte, and Sebastian Burckhardt. The Design of a Task Parallel Library. In *Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, page 227–242, New York, NY, USA, 2009. Association for Computing Machinery. 103

[51] X. Liu, K. Gong, and L. Ying. Steady-State Analysis of Load Balancing with Coxian-2 Distributed Service Times. *arXiv preprint*, 2020. 50

[52] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg. Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. *Perform. Eval.*, 68:1056–1071, 2011. 201, 219

[53] David M. Lucantoni, Kathleen S. Meier-Hellstern, and Marcel F. Neuts. A Single-Server Queue with Server Vacations and a Class of Non-Renewal Arrival Processes. *Advances in Applied Probability 1990-sep vol. 22 iss. 3*, 22, sep 1990. 31

[54] M Mézard, G Parisi, and M Virasoro. *Spin Glass Theory and Beyond*. WORLD SCIENTIFIC, 1987. 40

[55] W. Minnebo and B. Van Houdt. A Fair Comparison of Pull and Push Strategies in Large Distributed Networks. *IEEE/ACM Transactions on Networking*, 22:996–1006, 2014. 38, 76, 78, 81, 103, 106, 143, 171

[56] Wouter Minnebo, Tim Hellemans, and Benny Van Houdt. On a class of push and pull strategies with single migrations and limited probe rate. *Performance Evaluation*, 113:42–67, 2017. 103, 143, 171

[57] R. Mirchandaney, D. Towsley, and J.A. Stankovic. Adaptive load sharing in heterogeneous distributed systems. *Journal of parallel and distributed computing*, 9(4):331–346, 1990. 103, 143, 171

[58] M. Mitzenmacher. The Power of Two Choices in Randomized Load Balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12:1094–1104, October 2001. 38, 49, 72, 81, 93

[59] A. Mukhopadhyay. *Mean Field Interactions in Heterogeneous Networks*. PhD thesis, University of Waterloo, 2016. 93, 97

[60] M.F. Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. John Hopkins University Press, Baltimore, MD, 1981. 13, 14, 15, 116, 118, 148, 150, 175, 179, 212

[61] T. Ozawa. Sojourn time distributions in the queue defined by a general QBD process. *Queueing Systems*, 53(4):203–211, 2006. 125, 126, 229, 234

[62] A. Panchenko and A. Thümmler. Efficient Phase-type Fitting with Aggregated Traffic Traces. *Perform. Eval.*, 64(7-8):629–645, August 2007. 171

[63] V. Ramaswami. A stable recursion for the steady state vector in markov chains of m/g/1 type. *Communications in Statistics. Stochastic Models*, 4(1):183–188, 1988. 213

[64] A. Robison, M. Voss, and A. Kukanov. Optimization via reflection on work stealing in TBB. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8. IEEE, 2008. 103

[65] W. E. Roth. On direct product matrices. *Bulletin of the American Mathematical Society*, 40(6):461 – 468, 1934. 36

[66] S. Shneer and S. Stolyar. Large-scale parallel server system with multi-component jobs. *arXiv preprint*, 2020. 50

[67] Seva Shneer and Alexander Stolyar. Large-scale parallel server system with multi-component jobs. *Queueing Systems*, 98:21–48, 2021. 42, 203

[68] Nikki Sonenberg, Grzegorz Kielanski, and Benny Van Houdt. Performance Analysis of Work Stealing in Large-Scale Multithreaded Computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 6(2), September 2021. 103, 105, 143, 171

[69] M.S. Squillante and R.D. Nelson. Analysis of Task Migration in Shared-memory Multiprocessor Scheduling. *SIGMETRICS Perform. Eval. Rev.*, 19(1):143–155, 1991. 103, 143, 171

[70] Andreas Stathopoulos, Alma Riska, Zhili Hua, and Evgenia Smirni. Bridging ETAQA and Ramaswami's formula for the solution of M/G/1-type processes. *Performance Evaluation*, 62:331–348, 10 2005. 213, 214

[71] Shaler Stidham. Technical Note-A Last Word on $L = \lambda W$. *Oper. Res.*, 22(2):417–421, apr 1974. 28

[72] A.L. Stolyar. Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Systems*, 80(4):341–361, 2015. 38, 81, 201, 219

[73] Willy Tarreau. Test Driving "Power of Two Random Choices" Load Balancing. `https://www.haproxy.com/blog/power-of-two-load-balancing/`, 2019. 93

[74] Mark van der Boor, Sem Borst, and Johan van Leeuwaarden. Hyper-scalable JSQ with sparse feedback. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(1):1–37, 2019. 201, 202, 208, 220, 222, 236

[75] Mark van der Boor, Martin Zubeldia, and Sem Borst. Zero-wait load balancing with sparse messaging. *Operations Research Letters*, 48(3):368–375, 2020. 201

[76] B. Van Houdt. Global Attraction of ODE-based Mean Field Models with Hyperexponential Job Sizes. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(2):Article 23, June 2019. 17, 51, 52, 53, 81, 82, 83, 86, 88, 89, 90, 95, 97, 98, 99

[77] Benny Van Houdt. Randomized Work Stealing versus Sharing in Large-scale Systems with Non-exponential Job Sizes. *IEEE/ACM Transactions on Networking*, 27:2137–2149, 2019. 38, 81, 103, 106, 113, 120, 121, 124, 125, 143, 171

[78] Benny Van Houdt. Introduction to performance modelling. Course notes, 2022. 9, 26

[79] Ignace Van Spilbeeck and Benny Van Houdt. Performance of rate-based pull and push strategies in heterogeneous networks. *Performance Evaluation*, 91:2–15, 2015. 38, 81, 103, 143, 171

[80] Ignace Van Spilbeeck and Benny Van Houdt. On the Impact of Job Size Variability on Heterogeneity-Aware Load Balancing. In *Queueing Theory and Network Applications*, pages 193–215, Cham, 2018. Springer International Publishing. 17, 38, 81, 97

[81] T. Vasantam, A. Mukhopadhyay, and R. R. Mazumdar. The mean-field behavior of processor sharing systems with general job lengths under the SQ(d) policy. *Performance Evaluation*, 127-128:120 – 153, 2018. 50, 74, 93

[82] N.D. Vvedenskaya, R.L. Dobrushin, and F.I. Karpelevich. Queueing System with Selection of the Shortest of Two Queues: an Asymptotic Approach. *Problemy Peredachi Informatsii*, 32:15–27, 1996. 38, 49, 52, 81, 89, 93

[83] N. Wirth. Tasks versus Threads: An Alternative Multiprocessing Paradigm. *Software - Concepts and Tools*, 17:6–12, 01 1996. 103, 105

[84] Ronald W. Wolff. Poisson Arrivals See Time Averages. *Operations Research*, 30(2):223–231, 1982. 28

[85] Qingqing Ye and Liwei Liu. Analysis of MAP/M/1 queue with working breakdowns. *Communications in Statistics - Theory and Methods*, 47(13):3073–3084, 2018. 31