Queueing Networks with Shared Resources

Wemke van der Weij, 1982
Queueing Networks with Shared Resources
ISBN 978-90-8659-298-2

Thomas Stieltjes Institute
for Mathematics

The research in this thesis has been carried out in the context of E-Quality Center of Excellence, and has been partly supported by the Basic Research in Informatics for Creating the Knowledge Society (BRICKS) project on Quality of Service in Communication Networks.

Cover designed by Jolanda Oudhoff.
Printed by Printpartners Ipskamp B.V., The Netherlands.

VRIJE UNIVERSITEIT

Queueing Networks with Shared Resources

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op donderdag 23 april 2009 om 10.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Wemke van der Weij

geboren te Langedijk

promotor: prof.dr. R.D. van der Mei

DANKWOORD

Vier jaar onderzoek zijn voorbij en hier heeft u het eindresultaat in handen. Het proefschrift, een boekwerk waarin mijn onderzoeksresultaten uiteen gezet worden. Bij de totstandkoming van dit proefschrift hebben vele mensen een rol gespeeld. En zoals de titel van deze tekst al zegt wil ik een aantal mensen bedanken. Geen gemakkelijke opgave, want veel mensen hebben op de een of andere manier een bijdrage geleverd, dan wel wetenschappelijk dan wel persoonlijk. Zonder ieder bij toenaam te kunnen noemen, wil ik toch een aantal mensen persoonlijk uitlichten.

Allereerst wil ik mijn promotor Rob van der Mei bedanken voor zijn steun en enthousiasme. Zijn overtuiging van de zelfstandigheid van een promovendus heeft mij vele mogelijkheden tot samenwerkingen in binnen- en buitenland geboden, waarvoor ik hem dankbaar ben. Daarnaast dank aan mijn leescommissie: Hans van den Berg, Sandjai Bhulai, Sem Borst, Nico van Dijk en Michel Mandjes. De suggesties van de commissie voor mijn proefschrift hebben een zeer positieve bijdrage gehad op het eindresultaat. Ook dank aan Bert Zwart en Ger Koole voor het zitting nemen in mijn promotiecommissie. Op het Centrum Wiskunde & Informatica (CWI) maakte ik deel uit van de groep PNA2 en op de Vrije Universiteit maakte ik deel uit van de groep OBP. Beide heb ik als een zeer plezierig en gezellig ervaren, waarvoor ik jullie allen wil bedanken. En in het bijzonder: Maaike, Regina en Pascal als medepromovendi op het CWI, Marco en Vivek als mijn maatjes op de Vrije Universiteit (VU) en de post-docs Matthieu en Misja met hun wijsheden op wetenschappenlijk en sociaal vlak. Daarnaast mijn grote dank aan Sindo en Sandjai. Jullie hebben niet alleen wetenschappelijk maar ook zeker persoonlijk veel voor mij betekend. Naast mijn ‘wetenschappelijke’ collega’s, vind ik het ook belangrijk de medewerk(st)ers van de ondersteunende diensten te bedanken: Rick, Karin, Michaël, Jan, Bikkie, Rob, Susanne en Mike. Jullie stonden altijd voor me klaar om elke vraag te beant-

woorden en te helpen waar mogelijk was.

De gastvrijheid op de VU en de mogelijkheden die mij daar werden geboden om college te geven, studenten te begeleiden en zelfs wat uitstapjes in de wiskunde van callcenters beschouw ik als een zeer leerzame en waardevolle aanvulling op mijn onderzoekswerkzaamheden. Daarnaast heeft de VU alsook het CWI het mogelijk gemaakt dat ik een aantal maanden deel heb mogen uitmaken van de onderzoeksgroep van George Shanthikumar en Rhonda Righter aan de University of California in Berkeley. Naast de wetenschappelijke uitdagingen een zeer mooie ervaring om in Berkeley te zijn geweest. Het OiO-schap bracht ook de nodige congresbezoeken met zich mee, waardoor ik naast Berkeley ook Ilkley, Madrid, Edinburgh, Seattle en Washington heb mogen bewonderen.

Onderzoeken is natuurlijk een ontzettend mooi vak, maar om de focus niet te verliezen waren voor mij muzikale uitstapjes een onmisbare afwisseling. Saxofoon en tegenwoordig ook hobo spelen hebben me met beide benen op de grond weten te houden. Veel mensen hebben hier hun bijdrage aan geleverd, wat voor mij erg waardevol is geweest. Zonder hier een lijst met namen te willen produceren wil ik jullie allemaal ontzettend bedanken.

Ook heb ik veel steun ontvangen van vrienden en vriendinnen uit mijn geboortedorp, mijn middelbare schooltijd en ook mijn studententijd. Ondanks dat het soms onmogelijk leek om te vertellen wat ik nu precies op een werkdag deed, ben ik blij dat jullie mij altijd vertrouwen hebben gegeven. Naast vrienden, collega's en muzikanten, en velen in een dubbelrol, wil ik mijn familie bedanken. Ze hebben me altijd gesteund en aangemoedigd nieuwe uitdagingen aan te gaan, wat zeer zeker ook heeft bijgedragen tot deze mijlpaal in mijn leven: het proefschrift. Ontzettend bedankt! Last but not least, Bart bedankt voor je begrip en liefde tijdens de laatste fase van het tot stand komen van mijn proefschrift.

CONTENTS

CHAPTER 1

INTRODUCTION

Modern communication systems are mixtures of voice telephony systems and computer-communication systems designed to offer a wide range of services such as voice telephony, data transfer, and interactive video. In the strongly competitive market of communication services, it is essential for service providers to be able to deliver services at attractive price-quality ratios. To this end, it is important for service pro viders to have an understanding of how the performance of the system (e.g., in terms of response times, throughput, stability) for a given traffic load depends on the system resources, such as bandwidth and computing power. This understanding gives insights in how to properly control the system and manage its quality.

Queueing theory provides a powerful means to study performance questions and price-quality optimization. In the vast majority of queueing models the servers are assumed to be independent entities that serve incoming jobs at a fixed processing rate. However, in many modern application areas, such as (middleware-based) distributed systems, mobile ad hoc networks, and application servers, the development of performance models naturally leads to the formulation of queueing networks with shared resources, i.e., where servers do interact and hence service rates are dependent. Although the theory of queueing models with independent servers is well-matured, today remarkably little is known about queueing models in which the servers share common resources.

In this thesis we study fundamental performance questions for queueing models with shared resources. We first focus on stability issues of these models, and then on their product-form solutions. For the so-called LPS queue we study its monotonicity with respect to the number of servers, continuing with optimizing the LPS queue and some variants.

1.1 Background and motivation

The phenomenon of congestion occurs in many real-world situations. For example, waiting lines occur at post offices, in supermarkets, at elevators, and in traffic situations, but also on a more abstract level in call centers, inventory and production systems, and computer-communication systems. To cope with congestion, one needs to understand the relevant queueing situations. This can be realized by developing and analyzing models in the framework of queueing theory. A key characteristic of queueing models is the inherent randomness in the arrival processes, service requirements, and routing policies. Queueing theory typically focuses on the derivation and calculation of performance metrics such as queue-length and waiting-time distributions, stability, throughput, and also addresses the optimization of the performance of the system.

In general, a queueing model describes a system in which resources are used to perform certain tasks. The tasks are usually called jobs, and these tasks are performed by so-called servers. A common assumption in most classical queueing models is that servers are independent entities that serve incoming jobs at a fixed rate. However, such an assumption is often unrealistic. Many application areas lead to the formulation of queueing models where servers are not stand-alone entities but share resources. Examples of such models are multi-layered queueing models, where servers at one layer effectively share a lower-layer resource, and hence become jobs at that lower layer. To illustrate the practical relevance of such layered queueing models, below we address several applications motivating the models studied in this thesis.

Over the past few decades, the popularity of data services has increased dramatically, both in the business and the consumer market. This has boosted the demand for application servers (e.g., Web servers, database management systems, multi-media servers) and distributed software systems that can handle extremely large numbers of users simultaneously, while the user-perceived performance should be both satisfactory and predictable (cf., e.g., [24]). In line with this development, new services that combine and integrate the functionalities of existing services have been brought to the market. Consider, for example, Web server technology. Over the past few years, Web server functionality has evolved from standard document-retrieval functionality to the ability to provide end-user interfaces in distributed computing environments. In this context, Web servers are typically used as front-end devices, hiding the business logic and the communication protocols with remote back-end application servers. This type of architectures are often referred to as multi-tiered IT architectures, see Figure 1.1. Typical examples of services operating in multi-tiered IT architectures are on-line services, such as Internet banking, on-line booking services, and on-line shopping. The ability to offer on-line services has raised the need for service providers to reach an understanding of the performance capabilities and limitations of their IT infrastructures, and to predict the end-to-end performance of the services offered to the users. This requires the development and analysis of the performance models of Web servers. Van der Mei et al.

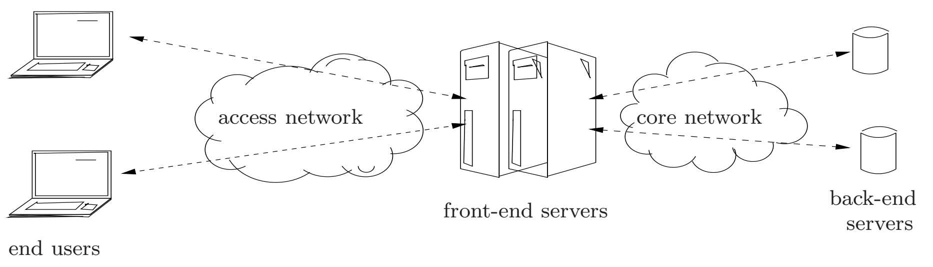


Figure 1.1: Application servers in a multi-tiered distributed computing environment.

propose a Web server model in [79], which works as follows. Incoming Web transaction requests go through two consecutive processing phases: document-retrieval processing (phase I), and script processing (phase II). To this end, the Web server is equipped with two types of threads: type-1 threads are dedicated to perform phase-I processing, and type-2 threads are dedicated to perform phase-II processing. If a tagged transaction request T finds a type-1 thread available upon arrival, then that thread will immediately start phase-I processing; otherwise, T is placed in the type-1 buffer which is served on a First Come First Served (FCFS) basis. When the type-1 processing of T has been finalized, T may be forwarded to the pool of type-2 threads for phase-II processing (if needed, depending on the type of request). Similarly, if T finds a type-2 thread available upon arrival, then that thread will immediately start phase II processing; otherwise, T is placed in the type-2 FCFS-based buffer. An important feature of this model is that at any moment in time all busy threads effectively share the underlying hardware resources. In its simplest form, this model can be viewed as a tandem of two multiserver queues with two layers: at the higher layer the busy servers (representing threads) effectively share the resources at the lower layer; this sharing can be done, for example, in a processor-sharing (PS) fashion or on an FCFS basis, see Figure 1.2.

Queueing models with shared resources also occur in many other application areas. Consider, for example, bandwidth-sharing networks [16, 75], providing natural models for describing the dynamic flow-level interaction among elastic data transfers that traverse several links along their source-destination paths in communication networks. In these networks the bandwidth is shared among the flows according to a processor-sharing type of service discipline. Other examples of resource-sharing networks are found in the modeling of the flow-level performance in wireline data networks where the capacity of different links is shared among competing flows [15], or in wireless networks where users can communicate via a cascade of intermediate hops (cf. [21, 26, 70]). Multi-layered networks also occur in the modeling of cable access networks, which are often regulated by a request-grant procedure in which actual data transmission is preceded by a reservation procedure. Consider, for example, TV networks that have been upgraded to enable communica-

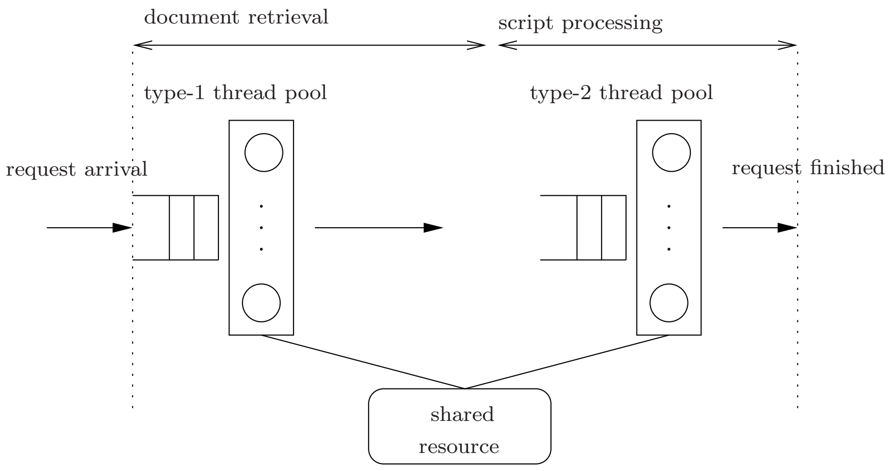


Figure 1.2: An example of a Web server.

tions between users and a centrally located head-end, which is a master facility for receiving television signals for processing and distribution over a cable television system. In order to coordinate transmissions from users to a head-end, a medium access protocol is needed, in which time slots are used for reservation and for data transmission [72, 95]. In addition to the examples given above, software-hardware interaction also occurs in distributed applications based on middleware, a software layer that manages the interaction between the distributed applications. In such an environment, the Portable Object Adapters perform server-side operations (jobs), that can be executed either sequentially or in parallel [52]. The applications discussed above all lead to queueing models with shared resources. In the next section we briefly introduce classical queueing models, continuing with layered queueing models in Section 1.3.

1.2 Classical queueing models

In the beginning of the 20th century the first queueing models for dimensioning of telephone systems were developed by Erlang. The classic problem of determining how many circuits are needed to provide an acceptable telephone service quality became tractable due to his famous Erlang blocking formula [40]. The rise of computers in the mid-20th century boosted the interest in queueing theory (e.g., [64, 69]). The appearance of communication systems in which data and voice are integrated further intensified the research on queueing models (see [29, 92] for overviews). We refer to [28, 64, 68] for excellent monographs on classical queueing theory.

1.2.1 Single-server queue

The most elementary queueing model is the so-called $G/G/1$ queue, which works as follows: Jobs arrive to the queue one at a time. The time between two consecutive arrivals is defined to be the interarrival time. Arrival processes are usually assumed to be such that interarrival times form an independent and identically distributed (i.i.d.) sequence of random variables. The service requirement of a job is the amount of time the server needs to serve the job if the job would receive the server's full capacity. It is assumed that service requirements are i.i.d., and that jobs only leave the system after having received their entire service requirement. The described model is often called the $G/G/1$ queue, a notational convention proposed by Kendall [65]. The G in this notation stands for general probability distributions, (note that GI is also used in literature, denoting generally independent) the first G referring to the distribution of the interarrival times, and the second to the distribution of the service requirements. The number 1 refers to the single server. This server serves one job at a time, and is only idle if there are no jobs in the queue. The $G/G/1$ queue is illustrated in Figure 1.3. In the special case of Poisson arrivals, i.e., exponentially distributed interarrival times, we denote the model as the $M/G/1$ queue, where the M stands for Markovian. If the service requirements are also exponentially distributed, the queue is referred to as the well-known $M/M/1$ queue. Following Kendall's notation, multiserver queues with $s > 1$ servers are denoted by $M/G/s$, $G/G/s$, etcetera.

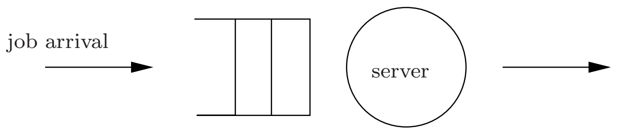


Figure 1.3: An illustration of a single-server queue.

Many variations on the basic $G/G/1$ queue have been studied. The behavior of the queue depends on the service discipline, which specifies how the capacity is allocated to jobs in the system. Many different service disciplines have been studied in the literature. We only mention the disciplines relevant for this thesis, starting with the FCFS discipline. The FCFS discipline is a service policy in which the jobs are served in the order of arrival, without other biases or preferences.

1.2.2 Processor-sharing queue

The (egalitarian) processor-sharing (PS) queue was introduced in the context of computer systems, where multiple processes share common processors. In the PS queue the total service capacity is equally shared among all jobs present. Thus, when there are $n > 0$ jobs present, each of them receives a fair share $1/n$ of the available service capacity 1. Originally, the PS paradigm emerged as an idealization of round-robin scheduling mechanisms in time-shared computer systems [66, 67]. In recent years, the PS discipline has attracted interest for modeling the flow-level performance of bandwidth-sharing protocols, see for example [75]. A main advantage of PS disciplines compared to FCFS disciplines is that small jobs do not have to wait before a server becomes available, but will immediately receive service [84].

A variation on the PS queue is the so-called limited processor-sharing (LPS) queue. The LPS queue extends the ordinary PS queue by assuming that not all jobs can be served directly upon arrival. This extension moves away from the idealized PS paradigm that serves *all* jobs in a round-robin scheme in time-sharing systems. This LPS queue is a PS queue in which a newly incoming job is only taken into service when the number of jobs in service is less than some threshold c; jobs that find that the threshold is reached, are placed in an entrance buffer which is served on an FCFS basis. The service capacity is shared equally by all jobs in service, which is less than or equal to c. When the number of jobs in service drops from c to $c-1$, the longest waiting job in the buffer will be served. For this general setting, c equal to one reduces the system to an FCFS queue and c equal to infinity reduces the system to a PS queue. A disadvantage of the PS queue [12, 33] is that allowing too many jobs may lead to a reduction of the overall performance. This can be avoided with the LPS queue, since always at least $1/c$ of the service capacity is guaranteed to the jobs in service.

A commonly used generalization of the PS queue is the generalized processor-sharing (GPS) queue. In a GPS queue the jobs are served simultaneously with a rate depending on the phase and the number of jobs present at that phase. The service rate of a job is given as a function $f^{(i)}(x_i)$, where x_i is the number of jobs present in service phase i [27]. In [88, 89], the service rate only depends on the fact of a queue is empty or not, and not on the queue length itself. Each non-empty class receives a certain guaranteed share of the capacity, which is defined by a weight function per class. This discipline is also referred to as the GPS discipline. This GPS discipline was developed as a service discipline to share the capacity of congested communication links in an efficient, flexible, and fair manner.

The discriminatory processor-sharing (DPS) discipline is a multiclass generalization of the egalitarian processor-sharing queue in which the total service capacity is shared by the jobs in proportion to class-dependent weight factors. The DPS discipline provides a natural approach for modeling the flow-level performance of differentiated bandwidth-sharing mechanisms. A literature survey on DPS queues is given in [1, 20].

1.3 Layered queueing models

Queueing models with resources shared among the servers can be seen as queueing models with multiple layers. At one layer jobs compete for access to servers which, in turn, compete for access to resources at another layer, and hence, can be seen as jobs at that layer. Several papers focus on layered queueing models. Rolia and Sevcik [99] propose the method of layers (MOLs) approach for modeling a distributed software system. They explicitly take into account both software and hardware contention and interactions between client and server processes. Software tasks are organized into several layers. A task may only send synchronous messages to tasks belonging to a lower layer and may receive synchronous messages to tasks belonging to a higher layer. They analyzed this queueing model by decomposition, considering two adjacent layers of tasks at a time, and they developed a separate queueing model to represent the contention in the CPU and the I/O devices. The two models are then combined using an iterative algorithm to obtain the expected response time. The MOLs approach was further extended in Sheikh et al. [107] to account for asynchronous messaging using residence time expression modifications.

Another fundamental contribution to layered queueing networks theory is presented by Woodside et al. [119], who propose the so-called Stochastic Rendezvous Network (SRN) models to analyze the performance of application software with client-server synchronization. In an SRN model entities called tasks represent hardware or software objects which may execute concurrently. A single task does not have any internal concurrency. Tasks communicate through synchronous messages. A task that accepts messages from other tasks, i.e., a server task, executes two or more service phases. The first phase has to do with message processing while the server acts on its own in the remaining phases, performing, for instance, cleanup tasks. A server task, during any phase, may send a synchronous message to another task. The SRN model is analyzed using a Mean Value Analysis (MVA) in order to obtain contention delays.

Ramesh and Perros [94] model a Web server system where clients and servers communicate via synchronous and asynchronous communication, and where the servers form a multi-layered hierarchical structure. They propose an approximate method for calculating the mean response time based on a decomposition approach. Dilley et al. [37] describe custom instrumentation to collect workload metrics and model parameters from large-scale Web servers. They develop a layered queueing model (LQM) of a Web server and use the model to predict the impact of a single Web server thread pool size on the server and client response times. Franks et al. [45] focus on the correct definition and detection of bottlenecks in the context of layered queueing models. Models related to layered queueing models are the so-called coupled-processor models, i.e., multi-server models where the rate of a server at a queue depends on the number of servers at the other queues (see [27, 41, 71, 95]).

The LPS queue (see also Section 1.2.2) can be seen as a two-layered queueing model. For this model, Avi-Itzhak and Halfin [6] give a simple approximation for the expected sojourn time, and Van der Weij [112] extends these approximations to a tandem of two multiserver queues, in which the active servers share a common resource in a PS fashion. Zhang and Lipsky [124, 125] give a computational analysis based on matrix-geometric methods. More recently, several new results for the LPS queue have been obtained. Zhang et al. [126] obtain for the LPS queue the limit of the measure-valued process under diffusion scaling and under heavy traffic conditions and they proved that the limit of the number of jobs in the system is a piece-wise reflected Brownian motion. An extension of the study for the LPS queue can be found in [127], where fluid analysis is done. Approximation formulas for various performance quantities for the LPS queue are established in Zhang and Zwart [128] based on diffusion limits.

For the special case $c = \infty$, the LPS queue coincides with the classical PS queue. Kleinrock [66, 67] wrote extensive literature on processor-sharing queues, studying the $M/M/1$ processor-sharing queue. Kleinrock [68] shows that the expected sojourn time conditioned on the service requirement of the jobs is proportional to the service requirement. This result is extended to the $M/G/c$ processor-sharing queue by Sakata et al. [100, 101]. In [27], Cohen extends this to a general class of networks, where the service rate of a queue depends on the number of jobs in that queue. In [69, 97] the processor-sharing queue is viewed as an idealization of time-sharing protocols in computer systems, and the authors notice the advantage of an PS discipline above the FCFS discipline, namely that a big job does not block the entire system, which is the case if only one job is served at a given time, serving the jobs in order of arrival. Schassberger [103] considers the processor-sharing queue as the limit of the round-robin discipline, where the round-robin discipline assigns time slices to each job in equal portions and in order, handling all jobs without priority. Bonald and Proutière [17] study the insensitivity of processor-sharing queues with respect to the service-time distribution. Yashkov [121] analyzes the sojourn-time distribution for this queue and also wrote survey papers on processor-sharing queues [122, 123].

1.4 Performance aspects

In this thesis we consider the following three aspects of queueing networks with shared resources: stability, product forms, and scheduling. These aspects and the related literature are outlined below.

1.4.1 Stability

Stability of a network is a concept that refers to the behavior of the queue-length process. Loosely speaking, in an unstable network the total number of jobs in the network (under proper scaling in time) will become infinite, whereas a network is stable if the scaled number of jobs remains in a compact set. Definitions can be found in Section 2.2. It is plausible that instability phenomena will be reflected in poor performance in terms of long delays and user impatience in practical circumstances. However, (in)stability is to a certain extent a theoretical concept that is unlikely to occur in an actual system due to admission and flow control mechanisms. Nevertheless, the stability properties of resource-sharing networks give a useful indication of the performance in terms of delays and user throughput. Note that a well-engineered network should avoid experiencing overload. However, the traffic fluctuations over time might lead to temporary surges that a well-designed network should deal with. A thorough understanding of the behavior of the network in overload is hence strongly needed. In particular, it is a fundamental issue to characterize in a network of queues, for given traffic conditions, which queues become unstable and what the asymptotic growth rates and throughputs are.

Over the past few decades a considerable amount of research has been dedicated to the stability of queueing networks. The Foster-Lyapunov criteria, which are based on finding a suitable test function having a positive or negative drift in almost all states of the state space is a general framework for analyzing stochastic stability [42, 82]. In [111] these methods are applied to multiserver queues with coupled servers. Fluid-limit analysis is also known as a powerful method for finding necessary and sufficient stability conditions for a wide class of multiclass queueing networks with work-conserving service disciplines [32, 81]. Recently, a sharp characterization of stability for systems where the service rate of each queue is decreasing in the number of jobs has been obtained in [19]. A characterization of the per-queue stability in a tandem of queues with an LPS service discipline is given in [78]. It emerges from these papers that general results for per-queue stability for multi-layered networks (or networks with bandwidth sharing) appear to be very challenging. In particular, if the stochastic stability of a network is well-known for work-conserving networks, detailed (per-queue) stability remains a difficult problem. For general service allocations without monotonicity properties, it is still an open area, even for exponentially distributed service times. Instead of focusing on stochastic stability, an alternative approach to tackle stability issues is to weaken the stability definition and to investigate the so-called rate stability of a network [39, 109]. Roughly speaking, it consists of characterizing the growth rates as linear or sub-linear. However, because in a great number of practical situations an overload situation is characterized by a linear asymptotic per-queue growth rate, rate stability provides a benchmark in cases where a more detailed stability description is almost hopeless. Using a similar line of thoughts, Egorova et al. [38] give a partial characterization of the overload behavior for the wide class of so-called α-fair bandwidth-sharing strategies defined in [75]. They examine the fluid limit under a suitable scaling of the number of flows in the system, and give a fixed-point equation for the corresponding asymptotic growth rates.

1.4.2 Product forms

Another important aspect worth studying in the context of queueing networks is the steady-state distribution. If a steady-state distribution has a product form, efficient algorithms to evaluate performance measures that are functions of this steady-state distribution can be defined. With insights on which particular characteristic will destroy the closed-form results for a given network, it might be possible to obtain a reasonable approximation heuristic for this network. The practical usefulness of queueing networks is often attributed to the simple Jacksonian product-form expressions that are available.

Jackson [56] introduces product-form solutions for open queueing networks, which is extended by Gordon and Newell [46] to closed queueing networks. Gordon and Newell introduce several assumptions on the model characteristics and provide a simple closed-form expression for the steady-state distribution and some average performance indices. This class of models is then extended to include various interesting and useful characteristics to represent more complex systems. These features include different types of jobs in the network, various service disciplines, state-dependent service rates, state-dependent routing between the queues, and some constraints on the size of the population of several queues in the network. The most famous result concerning product-form queueing networks is presented by Baskett, Chandy, Muntz and Palacios in 1975 [10], known as the BCMP theorem. It defines the well-known class of BCMP queueing networks with a product-form solution for open, closed or mixed models with multiple classes of jobs and various service disciplines and service-time distributions. The steady-state distribution is expressed as the product of the distributions of the individual queues with appropriate parameters and, for closed networks, with a normalization constant. Other extensions of networks having a product form can be found in [64]. In addition, Schassberger [103], Pittel [91], and Hordijk and Van Dijk [53, 54] contribute to product-form results, including blocking and non work-conserving service disciplines. Processor-sharing results with respect to product forms are presented in [25] and [17], in which capacity allocation functions are assumed to be strictly positive. Remarkably, hardly any product-form results are known for queueing networks with shared resources. In this context [53] and [54] introduce the concept of the 'adjoint' Markov chain to characterize the existence of a product form. For open and closed single-layered queueing networks this characterization has been explored extensively in [34] to obtain product-form results.

1.4.3 Scheduling

Scheduling is the process of deciding how to assign resources to the jobs in the system. This is usually done to balance the load of a system effectively or to achieve acceptable performance. The need for scheduling policies arises from the requirement to perform more than one process at a time and to transmit multiple flows simultaneously. During the last decades, scheduling received more attention with the introduction of complex communication systems.

There is a lot of literature on scheduling algorithms, and these have been successfully applied to many real-world problems. We refer to [30] for an excellent survey on scheduling, and to [90] for a more recent survey on the theory and applications. In the specific context of Web servers, Harchol-Balter et al. [47, 48, 49] and Crovella et al. [31] studied scheduling policies for static Web pages to reduce the response-time performance of Web servers, provided the size of a Web page is known *a-priori*; for this type of models, the results show that the classical Shortest Remaining Processing Time (SRPT) policy [105] is very effective. Relatively new scheduling mechanisms in literature for optimizing performance are based on the number of arrivals occurring during the service of the jobs. This policy can be used if a notion of time is missing [115]. Dynamic scheduling strategies are obtained using the framework of Markov decision processes (MDPs). Markov decision theory provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying a wide range of optimization problems solved via dynamic programming. After the pioneering work on MDPs by Bellman [11], the research activity was intensified due to Howard's book [55]. In [93] models are presented for sequential decision making under uncertainty, taking into account outcomes of current and future decisions. It encompasses a wide range of applications and has generated a rich mathematical theory. We refer to [43] for a survey on methods and applications of MDPs.

1.5 Outline of the thesis

In the course of this thesis, the emphasis gradually shifts from performance evaluation to performance optimization. Also the models considered gradually shift from general to more specific model instances. First, we are interested in general layered queueing models and their stability, continuing with product-form solutions. Then, after studying the LPS queue and its monotonicity properties with respect to the number of servers, we continue with optimizing the LPS queue and some LPS queue variants.

Each chapter is self-contained and has the following structure. First, we give an introduction and discuss the relevant literature. Second, the models studied are described and the notation is introduced. Third, we present the derivation of the results, and finally, we discuss the results and insights obtained.

In *Chapter 2* we focus on stability and throughput. We present the per-queue rate stability and the output rates for a general class of feed-forward queueing networks with a general capacity allocation function. More specifically, we derive necessary conditions for per-queue rate stability, and give upper bounds for the per-queue output rates and asymptotic growth rates under mild assumptions on the allocation function. For a set of parallel queues, we further prove the convergence of the output rates and give a full characterization of the per-queue rate stability, and an explicit expression for the per-queue output rate.

In *Chapter 3* we study product-form solutions. We consider two-layered queueing networks with two queues, either in series or in parallel, with a common resource shared among the two queues. For this type of networks, we specify a necessary and sufficient criterion to decide whether a given model possesses a product-form solution or not. This criterion unifies the parallel system, a reversible Markov chain, and the tandem system, a non-reversible Markov chain, in one product-form theorem. Subsequently, the criterion is applied to obtain a number of models yielding a product-form solution. These models include non-balanced capacity sharing. Despite of, but also due to, the different parallel and tandem mechanisms we observe that for certain examples the product form has the same structure, while for others there are essential differences. In addition, it is proven that several models cannot have a product-form solution.

In *Chapter 4* we study monotonicity properties for the LPS queue with respect to the number of servers. We use coupling arguments to prove that for service times with a decreasing failure rate the queue length is stochastically decreasing in the number of servers, and that for service times with an increasing failure rate, the queue length is stochastically increasing in the number of servers. The expected queue lengths and the asymptotic decay rate of the sojourn-time distributions are compared to that in other queueing models with and without restrictions on the number of servers.

In *Chapter 5* our goal is to obtain an optimal dynamic scheduling policy that minimizes the average queue length in the LPS queue. It is assumed that each job has a phase-type service-time distribution. We present a new decomposition-based approach to analyze monotonicity properties of the relative value function, which lead to a complete characterization of the optimal scheduling policy. Next, the performance under the optimal policy is compared to the performance under several commonly used static policies, including the optimal static policy. The results show that a significant gain can be obtained by using the optimal dynamic policy.

A logical continuation of Chapter 5 is *Chapter 6* in which we derive dynamic thread-assignment policies that minimize the average response time of Web servers. The Web server is modeled as a two-layered tandem of multi-threading queues, where the active threads compete for access to a common resource. We show that for Erlang and exponential service-time distributions the optimal dynamic policies assign threads to pending jobs with a shorter expected sojourn time than the jobs already in service. Experimental validation on an Apache Web server shows that these optimal dynamic thread-assignment policies confirm our analytical results on reductions in the response times.

CHAPTER 2

STABILITY ANALYSIS

In this chapter we study stability properties of layered queueing networks. The asymptotic behavior of this type of networks is fundamentally different from the behavior of classical queueing networks, where the service rate at each queue is usually assumed to be independent of the state of the other queues. We study the per-queue rate stability and output rates for a general class of feed-forward queueing networks with a general capacity allocation function. More specifically, we derive necessary conditions for per-queue rate stability, and give bounds for the per-queue output rate and asymptotic growth rates under mild assumptions on the allocation function. For a set of parallel queues, we further prove the convergence of the output rates and give explicit expressions for the per-queue rate stability and the per-queue throughput. This chapter is based on [59, 116].

2.1 Introduction

We study queueing networks in which the service rates at each of the *individual* queues are not independent, but depend on the state of the *entire* system, according to some general capacity allocation function, allocating the capacity resources to the servers. For this type of models, exact structural results are rare, and fundamental insight and intuition for seemingly simple questions about stability and throughput are lacking. This motivates an in-depth study of the per-queue stability for queueing networks with a general class of capacity allocation functions. General results for per-queue stability for multi-layered networks (or networks with bandwidth sharing) are scarce. In particular, even if global stability is well understood for work-conserving networks, detailed (per-queue) stability remains a difficult problem. For instance, if a network is globally stable, an interesting aspect is whether the individual queues are stable or not.

For general service allocations without monotonicity properties, detailed stability characterization is to the best of our knowledge an open problem, even for exponentially distributed services. Since stochastic stability seems to be intractable for these networks, only in [19] it is shown that for decreasing service allocation stochastic monotonicity can be obtained. We focus on an alternative approach to tackle stability issues, namely to weaken the stability definition and to investigate the so-called rate stability of the network [39]. This consists of characterizing the growth rates of the number of jobs in the queues as linear or sub-linear. For a great number of practical situations an overload situation is characterized by a linear asymptotic per-queue growth rate, and therefore rate stability provides valuable benchmark information in cases where a more detailed stability description seems to be impossible. This approach is applied in [109], where for a general class of single-server queues the rate stability of the single queues is determined. However, the service rate at these queues is independent of the other queues. Related are the lines of thought in Egorova et al. [38]. They give a partial characterization of the overload behavior, for the wide class of so-called α-fair bandwidth-sharing strategies defined in [75], by examining the fluid limit by suitable scaling the number of flows in the system, and give a fixed-point equation for the corresponding asymptotic growth rates.

We consider a queueing network with Poisson arrivals, exponential service-time distributions at all queues, internal feed-forward routing and with a structured work-conserving allocation function driving the service in all queues and depending on the state of the entire system. For this general model, we (1) derive necessary conditions for the per-queue rate stability, and (2) give bounds for the per-queue output rate. We show how to use these conditions on a two-queue tandem network to get necessary and sufficient conditions for rate stability. For a set of parallel queues with homogeneous capacity allocation, we further prove the convergence of the output rates and give a sharp characterization of the per-queue rate stability and per-queue throughputs.

The organization of this chapter is as follows. In Section 2.2 the model is described and the relevant notation and definitions are introduced. In particular, the difference between stochastic and rate stability is rigorously explained. In Section 2.3, asymptotic values as output rates and growth rates are defined. Using the structure of the considered allocation functions, important properties of these output rates are derived. In Section 2.4, some traffic inequalities are established leading to necessary conditions for the rate stability of each queue. We then illustrate the obtained results on two toy examples. In Sections 2.5 and 2.6, we consider two special cases, namely a two-queue tandem model and a model with an arbitrary number of parallel queues, and show that the necessary conditions derived in Sections 2.3 and 2.4 are also sufficient under mild conditions on the capacity allocation function. Finally, in Section 2.7 we address a number of challenging topics for further research.

2.2 Model and stability definitions

2.2.1 Network model

We consider an open queueing network with N queues. A job present at queue i is said to be of class i ($i = 1, \dots, N$). External jobs arrive at queue i according to a Poisson process with intensity $\lambda_i \geq 0$. Denote the vector of external arrival rates by $\boldsymbol{\lambda} := (\lambda_1, \dots, \lambda_N)^\top$. The service times at queue i are exponentially distributed with mean $\beta_i = 1/\mu_i$. Let $\boldsymbol{\mu} := (\mu_1, \dots, \mu_N)$. The state of the system is described by a vector $\mathbf{x} := (x_1, \dots, x_N)$, where x_i represents the number of jobs of class i present in the system. Let $\mathbf{x} \in \mathcal{X} := \{0, 1, \dots, \}^N$. When the system is in state $\mathbf{x}$, jobs of class i receive a service rate $\phi_i(\mathbf{x})$, where the function $\boldsymbol{\phi}(\mathbf{x}) := (\phi_1(\mathbf{x}), \dots, \phi_N(\mathbf{x}))$ is referred to as the system *capacity allocation function* . It is important to note that the various job classes are coupled since their *individual* service rates may depend on the state $\mathbf{x}$ of the *entire* system.

Assumptions on the routing

After receiving service at queue i, a job is routed to queue $j \in \mathcal{I} := \{0, 1, \dots, N\}$ with probability p_{ij}. We adopt the convention that when $j = 0$, the job simply leaves the network. Denote the N-by-N routing matrix by $P := (p_{ik})$, where $i,\ k = 1, \dots, N$. We assume that there is *no loop* in the routing, i.e., once a job has been served at a given queue, he never returns to this queue. This type of routing is often referred to as *feed-forward* routing. Consequently, we can order the queues such that $p_{ij} = 0,\ 1 \leq j \leq i$. The routing matrix P is substochastic, so that $R := (r_{ij}) := (I - P)^{-1}$ exists, where I is the N-by-N identity matrix. Moreover, let $D = (d_{ij})$ be the N-by-N diagonal matrix with diagonal entries, $d_{ii} := \frac{1}{\mu_i}$ ($i = 1, \dots, N$). Using these definitions, the load offered to queue i is given by

$$\rho_i := \boldsymbol{\lambda}^\top R D \mathbf{e}_i = \frac{1}{\mu_i} \sum_{j=1}^{N} \lambda_j r_{ji}, \tag{2.1}$$

where $\mathbf{e}_i$ is the i-th unit vector. Furthermore, denote $\rho = \sum_{i=1}^N \rho_i$.

Let $\mathbf{X}(t) := (X_1(t), \dots, X_N(t))$, where $X_i(t)$ denotes the number of jobs at queue i (i.e., either waiting or being served) at time $t \geq 0$. Then the N-dimensional process $\{\mathbf{X}(t), t \geq 0\}$ can be described as a continuous-time Markov process with state space $\mathcal{X}$. For a subset of indices $\mathcal{S}$, we denote by $\mathbf{x}_\mathcal{S}$ the restriction of the vector $\mathbf{x}$ to queues $\mathcal{S}$, i.e., $\mathbf{x}_\mathcal{S} = (x_i)_{i \in \mathcal{S}}$.

Assumptions on the service rates

Throughout the chapter, the system allocation function $\boldsymbol{\phi}(\mathbf{x})$ satisfies certain assumptions that we describe here.

Assumption 2.2.1 (Work-conserving allocation)**.** *Whenever the system is not empty, all capacity is assigned to the queues. For $\mathbf{x} \neq \mathbf{0} = (0, \ldots, 0)$,*

$$\sum_{i=1}^{N} \frac{\phi_i(\mathbf{x})}{\mu_i} = 1, \qquad \textit{and} \;\; \boldsymbol{\phi}(\mathbf{0}) := 0. \tag{2.2}$$

Without loss of generality, the total capacity of the system is assumed to be equal to 1 in (2.2).

Assumption 2.2.2 (Symmetric uniform limits)**.** *For all subsets of indices $\mathcal{U} \subseteq \{1, \ldots, N\}$ and $\mathcal{S} = \{1, \ldots, N\} \setminus \mathcal{U}$, there exists a function $g^{\mathcal{U}}$ on $\{0, 1, \ldots, \}^{N-|\mathcal{U}|}$ and some strictly positive numbers $l_i, i \in \mathcal{U}$ such that*

$$\forall j \in \mathcal{U}, \; \lim_{x_i \to \infty} \frac{\phi_j(\mathbf{x})}{\mu_j} = l_j g^{\mathcal{U}}(\mathbf{x}_{\mathcal{S}}). \tag{2.3}$$

Note that l_j does not depend on the set $\mathcal{U}$. In many applications in computer-communication systems the allocation functions have the following structure which is a special case of work-conserving allocations with symmetric uniform limits

$$\frac{\phi_i(\mathbf{x})}{\mu_i} = \frac{f_i(x_i)}{\sum_{j=1}^{N} f_j(x_j)}, \quad \mathbf{x} \in \mathcal{X}, \; \mathbf{x} \neq \mathbf{0}, \tag{2.4}$$

where $f_i(\cdot)$ is a non-negative function such that $f_i(0) := 0$ and $\lim_{x_i \to \infty} f_i(x_i) =: l_i < \infty$ $(i = 1, \ldots, N)$. Note that in this case, Assumption 2.2.2 implies that

$$\forall \mathcal{U} \subset \{1, \ldots, N\}, \;\; g^{\mathcal{U}}(\mathbf{x}_{\mathcal{S}}) = \left(\sum_{i \in \mathcal{U}} l_i + \sum_{i \notin \mathcal{U}} f_i(x_i) \right)^{-1}. \tag{2.5}$$

In the sequel, we refer to these allocations as extended processor-sharing allocations. Here are a few examples that have become classic models in queueing theory and performance evaluation

1. The *limited processor-sharing* allocation defined by

   $$f_i(x_i) = \min\{x_i, l_i\},$$

   where l_i is a positive integer.

2. The *limited discriminatory processor-sharing* allocation defined by

   $$f_i(x_i) = w_i \, \min\{x_i, C_i\},$$

   where C_i is a positive integer and $w_i > 0$ is a weight given to class i. In this case $l_i = w_i C_i$.

3. The *coupled processors* allocation defined by

   $$f_i(x_i) = l_i \mathbb{1}_{\{x_i>0\}},$$

   where $0 < l_i < +\infty$ is a weight associated with class i. In the literature, this allocation is sometimes referred to as the generalized processor-sharing (GPS) allocation.

The Assumptions 2.2.1 and 2.2.2 are not sufficient in general to get a sharp characterization of the rate stability set of the network. To get more precise results, we may assume one or both of the following conditions

Assumption 2.2.3 (Asymptotic monotonicity)**.** *For all subsets of indices $\mathcal{U} \subseteq \{1, \ldots, N\}$, there exists $\mathbf{x}^* > 0$, such that if $x_i > x_i^*$, for all $i \notin U$ and for all $j \in \mathcal{U}$, then*

$$\forall i \notin \mathcal{U}, \quad \frac{\phi_i(\mathbf{x}^*)}{\mu_i} \leq l_i g^{\mathcal{U}}(\mathbf{x}_{\mathcal{S}}). \tag{2.6}$$

For extended processor-sharing allocations, note that Assumption 2.2.3 is verified in particular if

$$f_i(x_i) \quad \leq \quad l_i, \ \ \forall\ x_i \geq 0,\ i = 1, \ldots, N.$$

Assumption 2.2.4 (Homogeneity)**.** *The allocation is called homogeneous if*

$$\forall \mathbf{x} \in \mathcal{X},\ \forall \gamma \in \mathbb{R}^+,\ \boldsymbol{\phi}(\gamma \mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}). \tag{2.7}$$

This assumption is satisfied for the coupled processors allocation and for allocations based on homogeneous utility functions. For more details on bandwidth-sharing networks and utility-based allocations, we refer to [75].

2.2.2 Stability definitions

The study of stability of stochastic processes traditionally deals with the question of existence of a measure that is invariant to the transition operator of the process and to which the process converges in distribution or in total variation. We aim here at describing some 'per-queue' stability properties, i.e., properties of the processes $\{X_i(s), s \geq 0\}$, for $i = 1, \ldots, N$. Since the process $\{X_i(s), s \geq 0\}$ is not by itself a Markov process, this is generally a much more ambitious question than describing the global stability (stability of $\mathbf{X}(t)$) which is well-known for work-conserving networks (see Theorem 2.2.7 below). To the best of our knowledge, the only per-queue stochastic stability results have been obtained for a set of parallel queues with decreasing allocations and there is no such result available for the general type of networks we consider here. Because the usual definitions of stochastic stability did not lead so far (without stricter assumptions on the allocation function and the topology) to tractable results, we turn our attention to a weaker definition of stability that allows to give practical answers. Hence, we are primarily concerned with the property of the conservation of rates through the network. Roughly speaking, it consists of characterizing the asymptotic growth rates as linear or sub-linear and to characterize the set of input parameters such that the incoming traffic at a queue is equal to the outgoing traffic. Interesting as a first-order stability property, rate stability turns out to also give useful necessary conditions for stochastic instability. For later reference, we thus define the following two notions of stability: rate stability and the stronger notion of stochastic stability.

From Assumption 2.2.2 the allocation functions $\phi_i(.)$, and hence the transition rates are bounded, and thus the process X is non-explosive. Hence we may assume that X and all other stochastic processes treated in the sequel have paths in the space $D = D(\mathcal{R}_+, \mathcal{Z}_+^N)$ of right-continuous functions from $\mathcal{R}_+$ to $\mathcal{Z}_+^N$ with finite left limits. In the sequel, a stochastic process with paths in D is viewed as a random element on the measurable space $(D, \mathcal{D})$, where $\mathcal{D}$ denotes the Borel σ-algebra generated by the standard Skorokhod topology [61].

Definition 2.2.5 (Rate stability)**.** *The process $\{X_i(t), t \geq 0\}$ is said to be rate stable if*

$$\liminf_{t\to\infty} \frac{X_i(t)}{t} = 0 \quad a.s.,$$

and the process is called rate unstable if

$$\liminf_{t\to\infty} \frac{X_i(t)}{t} > 0 \quad a.s.$$

Definition 2.2.6 (Stochastic stability)**.** *The process $\{X_i(t), t \geq 0\}$ is said to be stochastically stable if*

$$\lim_{r\to\infty} \sup_{t\to\infty} \Pr\{X_i(t) > r\} = 0,$$

and the process is called stochastically unstable if

$$\lim_{r\to\infty} \sup_{t\to\infty} \Pr\{X_i(t) > r\} > 0.$$

Moreover, the N-dimensional process $\{\mathbf{X}(t), t \geq 0\}$ is said to be globally stochastically stable (or stochastically stable) if $\{X_i(t), t \geq 0\}$ is stochastically stable for all $i = 1, \dots, N$.

The following result, characterizing the stochastic stability of the process $\{\mathbf{X}(s), s \geq 0\}$, is well-known for work-conserving networks. The total number of jobs can indeed be seen as the number of jobs of a single queue with unit service rate and the global stability is then a consequence of Loynes Theorem (cf., e.g., [7]).

Theorem 2.2.7 (Global stability)**.** *The network is globally stochastically stable if*

$$\sum_{i=1}^{N} \rho_i < 1.$$

The network is globally stochastically unstable if

$$\sum_{i=1}^{N} \rho_i > 1.$$

Definition 2.2.8 (Rate stability subsets)**.** *Let* $\mathcal{S} := \{i : \{X_i(t), t \geq 0\}$ *is rate stable*$\}$*, and* $\mathcal{U} := \{i : \{X_i(t), t \geq 0\}$ *is rate unstable*$\}$*.*

Since each queue is either rate stable or rate unstable, the index set $\{1, \ldots, N\}$ is partitioned into the tuple $\mathcal{P} := (\mathcal{S}, \mathcal{U})$, with $\mathcal{S} \cup \mathcal{U} = \{1, \ldots, N\}$, $\mathcal{S} \cap \mathcal{U} = \emptyset$. In case of rate stability, the number of jobs at queue i grows asymptotically 'slower than t' when t goes to infinity, at least on some trajectories. In case of stochastic stability, the process $\{X_i(t), t \geq 0\}$ remains in a finite neighborhood with positive probability. Remark that if $\{X_i(t), t \geq 0\}$ is an irreducible Markov process, then stochastic stability is equivalent to requiring $\{X_i(t), t \geq 0\}$ to be positive recurrent (see for example Theorem 12.25 in [61]). Note also that stochastic stability implies rate stability, as it should, but that the converse result is generally not true.

The next result underlines the relation between rate instability and stochastic instability.

Proposition 2.2.9. *For* $i = 1, \ldots, N$, $\liminf_{t\to\infty} \frac{X_i(t)}{t} > 0$ *implies that* $X_i(t) \to \infty$ *in probability.*

Proof. Suppose that $X_i(t)$ does not diverge to infinity in probability. Then there exists a subsequence $\{t_n, n = 0, 1, \ldots\}$ such that $X_i(t_n) \to Z_i$ (in probability) for some honest (i.e., almost surely finite) random variable Z_i. Moreover, there exists another subsequence $\{t'_n, n = 0, 1, \ldots\}$ such that $\{X_i(t'_n)\} \to Z_i$ almost surely [61]. Hence, $\frac{X_i(t'_n) - Z_i}{t'_n} \to 0$ almost surely and since Z_i is almost surely finite, $\frac{Z_i}{t'_n} \to 0$ and $\frac{X_i(t'_n)}{t'_n} \to 0$ almost surely, which implies that $\liminf_{t\to\infty} \frac{X_i(t)}{t} = 0$, almost surely. $\square$

Remark 2.2.10. Many authors (see for instance [2, 39, 76]) define rate stability differently, with slightly stronger assumptions. For the purpose of our analysis, we prefer the given definition that allows to link rate instability to the fact that a process is diverging to infinity.

2.3 Output rates and growth rates

2.3.1 Definition

The following notation is useful in the sequel. For a given sample path $\{\mathbf{X}(s), s > 0\}$, we define the *Cesaro mean service rate* at each queue of the network by

$$\varphi_i(t) \quad := \quad \frac{1}{t} \int_0^t \phi_i(\mathbf{X}(s)) ds, \quad i = 1, \ldots, N, t > 0. \qquad (2.8)$$

The *growth rate* of queue i is defined by

$$Y_i(t) \quad := \quad \frac{X_i(t)}{t}, \quad i = 1, \ldots, N, t > 0. \tag{2.9}$$

Over a given sample path $\{\mathbf{X}(s), s > 0\}$, we can further define the limiting values of the mean service rate

$$\underline{\varphi}_i := \liminf_{t\to\infty} \varphi_i(t), \quad \bar{\varphi}_i := \limsup_{t\to\infty} \varphi_i(t), \quad i = 1, \ldots, N,$$

and the *asymptotic growth rate* of the queues

$$\underline{Y}_i = \liminf_{t\to\infty} Y_i(t), \quad \text{and} \quad \bar{Y}_i = \limsup_{t\to\infty} Y_i(t).$$

From Assumption 2.2.2, the random variables $\bar{\varphi}_i$ are bounded, and consequently, we prove in the following section that the $\bar{Y}_i$ are almost surely bounded. We may therefore define the mean values of vectors, for $i = 1, \ldots, N$,

$$O_i := \mathbb{E}[\underline{\varphi}_i], \qquad \bar{O}_i := \mathbb{E}[\bar{\varphi}_i], \qquad Q_i := \mathbb{E}[\underline{Y}_i], \qquad \bar{Q}_i := \mathbb{E}[\bar{Y}_i], \tag{2.10}$$

and denote the corresponding vectors by $\mathbf{O} := (O_1, \ldots, O_N)^\top$, $\bar{\mathbf{O}} := (\bar{O}_1, \ldots, \bar{O}_N)^\top$, $\mathbf{Q} := (Q_1, \ldots, Q_N)^\top$ and $\bar{\mathbf{Q}} := (\bar{Q}_1, \ldots, \bar{Q}_N)^\top$. Note that rate stability of queue i implies that $\underline{\varphi}_i = 0$ (almost surely) and $Q_i = 0$. Moreover, note that if queue i is stochastically stable, then $\bar{Q}_i = Q_i = 0$ and $\bar{O}_i = O_i$.

2.3.2 Properties of the asymptotic rates

In this section we derive some properties of the rates of service obtained when a queue is rate unstable. These properties turn out to be crucial when characterizing the rate stability of the network. It is convenient to define, for $i = 1, \ldots, N$,

$$\eta_i := \mu_i l_i.$$

The next result gives a relation between the output rates and the fraction of capacity assigned for rate unstable queues. For a given stability partitioning of the queues $\mathcal{P} = (\mathcal{S}, \mathcal{U})$, denote

$$\bar{Z}_{\mathcal{P}} := \mathbb{E}\left[\limsup_{t\to\infty} \frac{1}{t} \int_0^t g^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))ds\right].$$

Proposition 2.3.1. (Balanced output rates for rate-unstable queues) *Suppose Assumption* 2.2.1 *holds. If* $i, j \in \mathcal{U}$*, then*

$$\eta_j \bar{O}_i = \eta_i \bar{O}_j. \tag{2.11}$$

In particular if $l_i > 0$ *and* $l_j > 0$*, then*

$$\frac{\bar{O}_i}{\eta_i} = \frac{\bar{O}_j}{\eta_j} = \bar{Z}_{\mathcal{P}}. \tag{2.12}$$

Moreover, if $(\alpha_j)_{j\in\mathcal{U}}$ *are positive numbers, then*

$$\mathbb{E}\left[\limsup_{t\to\infty}\left(\sum_{j\in\mathcal{U}}\alpha_j\varphi_j(t)\right)\right]=\sum_{j\in\mathcal{U}}\eta_j\alpha_j\bar{O}_j.$$

Proof. For all $i\in\mathcal{U}$, X_i diverges in probability to infinity. As ϕ is bounded, it implies that $\frac{\phi_i(\mathbf{X}(s))}{\mu_i}-l_ig^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))\to 0$ (in L^1), which gives that

$$\mathbb{E}\left[\frac{1}{t}\int_0^t\left(\frac{\phi_i(\mathbf{X}(s))}{\mu_i}-l_ig^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))\right)ds\right]\to 0.$$

Using the dominated convergence theorem, which allows us to interchange the expectation and the limit, we obtain that

$$\begin{aligned}&\mathbb{E}\left[\lim_{t\to\infty}\tfrac{1}{t}\int_0^t\left(\tfrac{\phi_i(\mathbf{X}(s))}{\mu_i}-l_ig^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))\right)ds\right]\\&=\lim_{t\to\infty}\mathbb{E}\left[\tfrac{1}{t}\int_0^t\left(\tfrac{\phi_i(\mathbf{X}(s))}{\mu_i}-l_ig^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))\right)ds\right]=0.\end{aligned}$$

We conclude by observing that

$$\begin{aligned}\mathbb{E}\left[\limsup_{t\to\infty}\tfrac{\varphi_i(t)}{\mu_i}\right]&=\mathbb{E}\left[\lim_{t\to\infty}\tfrac{1}{t}\int_0^t\left(\tfrac{\phi_i(\mathbf{X}(s))}{\mu_i}-l_ig^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))\right)ds\right]\\&\quad+l_i\mathbb{E}\left[\limsup_{t\to\infty}\tfrac{1}{t}\int_0^t g^{\mathcal{U}}(\mathbf{X}_{\mathcal{S}}(s))ds\right]\\&=l_i\bar{Z}_{\mathcal{P}}\\&=\tfrac{\eta_i}{\mu_i}\bar{Z}_{\mathcal{P}}.\end{aligned}$$

□

The next two propositions compare the outputs of rate stable and rate unstable queues for asymptotically decreasing allocations.

Proposition 2.3.2. (Unbalanced rates between rate stable and rate unstable queues) *Suppose Assumption* 2.2.3 *holds. Then if* $i\in\mathcal{S}$ *and* $j\in\mathcal{U}$, *it holds that*

$$\eta_j\bar{O}_i\le\eta_i\bar{O}_j. \tag{2.13}$$

Proof. For $i\in\mathcal{S}$ and $j\in\mathcal{U}$, following the same lines as in the proof of the previous proposition, we have

$$\frac{\bar{O}_i}{\mu_i}\quad\le\quad l_i\mathbb{E}\left[\limsup_{t\to\infty}\frac{1}{t}\int_0^t g^{\mathcal{U}}(\mathbf{X}(s))ds\right].$$

□

The following proposition explores the structure of the extended processor-sharing allocation.

Proposition 2.3.3. (Comparison of output rates for different stability partitioning) *Define the extended processor-sharing allocation as follows, for $i = 1, \ldots, N$,*

$$\phi_i(\mathbf{x}) = f_i(x_i) \left(\sum_{j=1}^{N} f_j(x_j) \right)^{-1},$$

with $f_i(x_i) \leq l_i$ for all $x_i \geq 0$, $i = 1, \ldots, N$, and consider two rate stability partition sets $\mathcal{P}_1 = (\mathcal{S}_1, \mathcal{U}_1)$ and $\mathcal{P}_2 = (\mathcal{S}_2, \mathcal{U}_2)$ such that $\mathcal{U}_2 = \mathcal{U}_1 \cup \{k\}$, with $k \in \{1, \ldots, N\}$. Then it holds that for $i = 1, \ldots, N$,

$$\eta_j \bar{O}_i^{\mathcal{P}_1} \leq \eta_i \bar{O}_j^{\mathcal{P}_2}. \tag{2.14}$$

Proof. Using again the lines of the proof of Proposition 2.3.1, we get for $i = 1, \ldots, N$,

$$\frac{\bar{O}_i^{\mathcal{P}_1}}{\mu_i} = \mathbb{E}\left[\limsup_{t \to \infty} \frac{1}{t} \int_0^t \frac{f_i(X_i(s))}{f_i(X_i(s)) + \sum_{j \neq i} l_j} ds \right], \tag{2.15}$$

and

$$\frac{\bar{O}_j^{\mathcal{P}_2}}{\mu_j} = \frac{l_j}{\sum_{j=1}^{N} l_j}. \tag{2.16}$$

The proof then follows directly from (2.15) and (2.16) by observing that for $i = 1, \ldots, N$,

$$\frac{f_i(X_i(s))}{f_i(X_i(s)) + \sum_{j \neq i} l_j} \leq \frac{l_i}{\sum_{j=1}^{N} l_j}.$$

□

2.4 Rate stability necessary conditions

2.4.1 Traffic inequalities

In the absence of stochastic stability assumptions, it is naturally not possible to define the input rate of the queues as the solutions of the classic traffic equations as in [106] for instance. However, we can derive traffic inequalities linking the input rates and the asymptotic output rates of the network. These equations give a mathematical understanding on the common notions of mean output rates and input rates in the network.

Theorem 2.4.1 (Traffic inequalities)**.** *The asymptotic output rates* $\mathbf{O}$, $\bar{\mathbf{O}}$ *and growth rates* $\mathbf{Q}$, $\bar{\mathbf{Q}}$ *are finite and satisfy the following linear inequalities given that Assumptions 2.2.1 to 2.2.4 hold, for* $i = 1, \dots, N$,

$$Q_i + \bar{O}_i \leq \lambda_i + \sum_j p_{ji} \bar{O}_j, \tag{2.17}$$

$$\bar{Q}_i + O_i \geq \lambda_i + \sum_j p_{ji} O_j. \tag{2.18}$$

The work-conserving property brings the additional inequalities

$$\sum_{i=1}^{N} \frac{\bar{O}_i}{\mu_i} \geq 1, \quad \textit{and} \quad \sum_{i=1}^{N} \frac{O_i}{\mu_i} \leq 1. \tag{2.19}$$

In the special case of $\rho > 1$ *and* $\mathcal{U} = \{1, \dots, N\}$, *we have*

$$\sum_{i=1}^{N} \frac{\bar{O}_i}{\mu_i} = 1. \tag{2.20}$$

Proof. Because of exponential service times and Poisson arrivals, $X(t)$ is a Markov process. Note again that from Assumption 2.2.2 the allocation functions $\phi_i(.)$, and hence the transition rates are bounded. This implies (the departure process from a queue being $D_i(t) = A_i(t) - X_i(t)$, with $A_i(t)$ the arrival process at queue i) that the process $\{M_i(t), t > 0\}$, defined by

$$M_i(t) := X_i(t) - X_i(0) - \int_0^t \{\lambda_i + \sum_j \frac{p_{ji}\phi_j(\mathbf{X}(s))}{\mu_j} - \frac{\phi_i(\mathbf{X}(s))}{\mu_i}\} ds,$$

is a local martingale. And since the transitions are bounded the martingale satisfies $E[M_i^2(t)] < Kt$ for $i = 1, \dots, N$, $t > 0$ and some $K > 0$ [98]. This implies that the process $\{M_i(t)/t, t > 0\}$ is a super-martingale bounded in L^2 and consequently, for $i = 1, \dots, N$, $\frac{M_i(t)}{t} \to 0 \ (t \to \infty)$, a.s. [98]. Assuming for simplicity that $\mathbf{X}(0) = \mathbf{0}$, it is readily seen from the definitions (2.9) and (2.8) that, for $i = 1, \dots, N$, $t > 0$,

$$\frac{1}{t} M_i(t) + \lambda_i + \sum_j \frac{p_{ji}\varphi_j(t)}{\mu_j} - Y_i(t) = \frac{\varphi_i(t)}{\mu_i}.$$

This implies that $\limsup_{t\to\infty} \frac{X_i(t)}{t} < +\infty$ as well as

$$\begin{aligned}\limsup_{t\to\infty} \frac{\varphi_i(t)}{\mu_i} &= \limsup_{t\to\infty} \left(\lambda_i + \sum_j \frac{p_{ji}\varphi_j(t)}{\mu_j} - Y_i(t) \right) \\ &\leq \lambda_i + \sum_j \frac{p_{ji} \limsup_{t\to\infty} \varphi_j(t)}{\mu_j} - \liminf_{t\to\infty} Y_i(t).\end{aligned}$$

Using the dominated convergence theorem, we get (2.17). Relations (2.18) are obtained along the same lines. The inequalities in (2.19) follow from the dominated convergence theorem as well as the equation

$$1 = \limsup_{t\to\infty} \left(\sum_{i=1}^{N} \frac{\varphi_i(t)}{\mu_i} \right) \leq \sum_{i=1}^{N} \limsup_{t\to\infty} \frac{\varphi_i(t)}{\mu_i}.$$

If $\rho > 1$, the total number of jobs is transient, and hence for all t, almost surely $\sum_{i=1}^{N} \frac{\phi_i(X(t))}{\mu_i} = 1$ and $\sum_{i=1}^{N} \frac{\varphi_i(X(t))}{\mu_i} = 1$. The last assertion thus follows from Proposition 2.3.1. □

2.4.2 Necessary conditions for rate stability for converging rates

In this subsection, we study the case $\bar{\mathbf{O}} = \mathbf{O}$, which serves as a benchmark for finding rate stability conditions in the general case. We show in the last section that we can actually prove the convergence of the asymptotic growth rates for a set of parallel queues with homogeneous, asymptotically monotone allocations.

Definition 2.4.2 $(\tilde{\mathbf{O}}, \tilde{\mathbf{Q}})$**.** *For a given stability partitioning $\mathcal{P} = (\mathcal{S},\mathcal{U})$ $(\mathcal{U} \neq \emptyset)$, define $(\tilde{\mathbf{O}}, \tilde{\mathbf{Q}})$ as the solution (when it exists) of*

$$o_i + q_i = \lambda_i + \sum_j p_{ji} o_j, \tag{2.21}$$

$$\sum_{i=1}^{N} \frac{o_i}{\mu_i} = 1, \tag{2.22}$$

$$\frac{o_i}{\eta_i} = \frac{o_j}{\eta_j} := \tilde{Z}_{\mathcal{P}} \quad (i, j \in \mathcal{U}), \tag{2.23}$$

$$q_i = 0 \quad (i \in \mathcal{S}). \tag{2.24}$$

We first prove the existence of a unique solution for $\tilde{\mathbf{O}}, \tilde{\mathbf{Q}}$. We then give conditions for this solution to be positive vectors. To simplify the notations, suppose without loss of generality that the queues are ordered so that the stable ones are the first ones, i.e., there exists an index m such that $\mathcal{S} = \{1, \ldots, m\}$ and $\mathcal{U} = \{m+1, \ldots, N\}$. Define $G^{\mathcal{E}_1\mathcal{E}_2}$ as the truncation of the matrix G to the queues in $\mathcal{E}_1, \mathcal{E}_2$: $G^{\mathcal{E}_1\mathcal{E}_2} = (G)_{i\in\mathcal{E}_1, j\in\mathcal{E}_2}$ and similarly the vector $\mathbf{v}^{\mathcal{E}} = (v_i)_{i\in\mathcal{E}}$. We then write the routing matrix in the following form

$$P = \begin{pmatrix} P^{\mathcal{S}\mathcal{S}} & P^{\mathcal{S}\mathcal{U}} \\ P^{\mathcal{U}\mathcal{S}} & P^{\mathcal{U}\mathcal{U}} \end{pmatrix}.$$

Recall that the vector $\boldsymbol{\eta}$ is defined as $\boldsymbol{\eta} = (l_1\mu_1, \ldots, l_N\mu_N)$ and let us introduce the vector $\boldsymbol{\omega}^{\mathcal{S}}$, and the positive constants $\kappa_{\mathcal{P}}$ and $\chi_{\mathcal{P}}$ as

$$\begin{aligned}
\boldsymbol{\omega}^{\mathcal{S}} &= \boldsymbol{\lambda}^{\mathcal{S}} H^{\mathcal{SS}}, \\
\kappa_{\mathcal{P}} &= \textstyle\sum_{i\in\mathcal{S}} \frac{(\eta^{\mathcal{U}} P^{\mathcal{US}} H^{\mathcal{SS}})e_i}{\mu_i}, \\
\chi_{\mathcal{P}} &= \textstyle\sum_{i\in\mathcal{U}} l_i,
\end{aligned}$$

where $H^{\mathcal{SS}} = (I - P^{\mathcal{SS}})^{-1}$. Remark that the matrix $H^{\mathcal{SS}}$ is not in general the restriction of the matrix R.

Proposition 2.4.3. *Fix a partition $\mathcal{P} = (\mathcal{S}, \mathcal{U})$ $(\mathcal{U} \neq \emptyset)$. There exists a unique solution $(\tilde{\mathbf{O}}, \tilde{\mathbf{Q}})$ of Equations* (2.21) *to* (2.24)*, characterized by the following equations*

$$\begin{aligned}
\tilde{\mathbf{O}}^{\mathcal{S}} &= (\boldsymbol{\lambda}^{\mathcal{S}} + \tilde{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{U}} P^{\mathcal{US}}) H^{\mathcal{SS}}, \\
\tilde{\mathbf{O}}^{\mathcal{U}} &= \tilde{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{U}}, \\
\tilde{Z}_{\mathcal{P}} &= \frac{1 - \sum_{i\in\mathcal{S}} \frac{\omega_i^{\mathcal{S}}}{\mu_i}}{\kappa_{\mathcal{P}} + \chi_{\mathcal{P}}}.
\end{aligned}$$

Moreover, the solution $\tilde{\mathbf{O}}, \tilde{\mathbf{Q}}$ is positive if and only if

$$\sum_{i\in\mathcal{S}} \frac{\omega_i^{\mathcal{S}}}{\mu_i} \leq 1,$$

and

$$\tilde{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{U}} \left(I^{\mathcal{UU}} - P^{\mathcal{UU}} - P^{\mathcal{US}} H^{\mathcal{SS}} P^{\mathcal{SU}} \right) \geq \boldsymbol{\lambda}^{\mathcal{U}} + \boldsymbol{\lambda}^{\mathcal{S}} H^{\mathcal{SS}} P^{\mathcal{SU}}.$$

Proof. The system of Equations (2.21) to (2.24) can be rewritten as

$$\tilde{\mathbf{O}}^{\mathcal{S}} = \boldsymbol{\lambda}^{\mathcal{S}} + \tilde{\mathbf{O}}^{\mathcal{S}} P^{\mathcal{SS}} + \tilde{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{U}} P^{\mathcal{US}}, \tag{2.25}$$

$$\tilde{\mathbf{Q}}^{\mathcal{U}} = \boldsymbol{\lambda}^{\mathcal{U}} + \tilde{\mathbf{O}}^{\mathcal{S}} P^{\mathcal{SU}} + \tilde{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{U}} (P^{\mathcal{UU}} - I^{\mathcal{UU}}), \tag{2.26}$$

$$\sum_{i\in\mathcal{S}} \frac{\tilde{O}_i^{\mathcal{S}}}{\mu_i} = 1 - \chi_{\mathcal{P}} \tilde{Z}_{\mathcal{P}}, \tag{2.27}$$

since from the definition it follows that $\tilde{Q}_i^{\mathcal{S}} = 0$ if $i \in \mathcal{S}$, and thus Equation (2.21) reduces to Equation (2.25). Similarly Equation (2.26) can be obtained. Using Equations (2.23) and (2.24) the Equation (2.27) can be derived. The proposition follows from the fact that the matrices $I^{\mathcal{E}} - P^{\mathcal{E}}$, $\mathcal{E} = \mathcal{SS}$, $\mathcal{UU}$ are invertible with positive inverse matrices. Then, $\tilde{\mathbf{O}} \geq \mathbf{0}$ and $\tilde{Z}_{\mathcal{P}} \geq 0$, if and only if

$$\sum_{i \in \mathcal{S}} \frac{\omega_i^{\mathcal{S}}}{\mu_i} \leq 1.$$

Moreover, $\tilde{\mathbf{Q}} \geq 0$ if and only if

$$\tilde{Z}_{\mathcal{P}}(I^{\mathcal{UU}} - P^{\mathcal{UU}})\boldsymbol{\eta}^{\mathcal{U}} \geq \boldsymbol{\lambda}^{\mathcal{U}} + \tilde{\mathbf{O}}^{\mathcal{S}} P^{\mathcal{SU}},$$

which follows from Equation (2.26). □

It is remarkable that the conditions of positivity of the output rates are not sufficient to characterize the stability set. In the case of parallel queues, for instance, where we will actually derive that $\bar{\mathbf{O}} = \mathbf{O}$, we show that the additional conditions underlined in Section 2.3 are indeed needed to sharply characterize, for given input parameters, the rate stability set.

2.4.3 Necessary conditions for rate stability

To derive necessary conditions for a given rate stability partitioning, we bound the output rates, taking into account the assumption of feed-forward routing. The bounds are obtained by comparing the maximum output rates with the outputs previously obtained in a (virtual) network where $\bar{O}_i = O_i$, for all i.

Lemma 2.4.4. *For $i = 1, \dots, N$, we have*

$$\bar{O}_i \leq \omega_i,$$

where the vector $\boldsymbol{\omega} = \boldsymbol{\lambda} R$ is the solution of the usual traffic equations

$$\boldsymbol{\omega} = \boldsymbol{\lambda} + \boldsymbol{\omega} P.$$

Proof. Remark first that $\boldsymbol{\omega}$ exists and is unique because $R = (I - P)^{-1}$ is a well-defined positive matrix since $I - P$ is substochastic. Define the degree of queue i in the following way; $d_i = 0$ if $p_{ji} = 0$, for all $j = 1, \dots, N$,. Otherwise, $d_i = \max_{j:\ p_{ji} > 0}\{d_j\}$. Because of the absence of loops in the network, there exists at least one queue i_0 of degree 0 (a source). Using the traffic inequalities of the previous section, we get for all queues i_0 of degree 0

$$\bar{O}_{i_0} \leq \lambda_{i_0} = \omega_{i_0}.$$

We further proceed by induction on the degree of queues. Suppose the assertion is true for all degrees less than m. Consider a queue of degree $m + 1$. It is receiving traffic from queues of lower degree. Using the traffic inequalities, the induction assumption and the definition of $\boldsymbol{\omega}$, we get

$$\bar{O}_i \leq \lambda_i + \sum_{j: d_j \leq m} p_{ji} \bar{O}_j \leq \lambda_i + \sum_{j=1}^{N} p_{ji} \omega_j = \omega_i.$$

□

We now derive the lemma leading to the main result of this section.

Lemma 2.4.5. *Let* $\bar{Z}_{\mathcal{P}} := \frac{\bar{O}_i}{\eta_i}, \forall i \in U,$. *For each partitioning* $\mathcal{P} = (\mathcal{S},\mathcal{U})$ $(\mathcal{U} \neq \emptyset)$, *we have*

$$\bar{Z}_{\mathcal{P}} \geq \tilde{Z}_{\mathcal{P}}.$$

If moreover $P^{\mathcal{US}} = 0$, *then*

$$\forall i \in \mathcal{S}, \bar{O}_i \leq \tilde{O}_i.$$

Note that this result holds without restriction on the routing policy, and is not limited to feed-forward routing.

Proof. Using Lemma 2.4.4 and the traffic inequalities given in Theorem 2.4.1, we can write that

$$\bar{\mathbf{O}}^{\mathcal{S}} \leq \boldsymbol{\omega}^{\mathcal{S}} + \bar{Z}_{\mathcal{P}} \boldsymbol{\eta}^{\mathcal{S}} P^{\mathcal{US}} H^{\mathcal{SS}},$$

since from Equation (2.17) we have that $Q_i = 0$ for $i \in \mathcal{S}$. Next, using Equation (2.19) and $\chi_{\mathcal{P}} = \sum_{i\in\mathcal{U}} l_i$, it follows that

$$1 \leq \chi_{\mathcal{P}} \bar{Z}_{\mathcal{P}} + \sum_{i\in\mathcal{S}} \frac{\bar{O}_i}{\mu_i}.$$

Hence, combining these equations we have

$$(\chi_{\mathcal{P}} + \kappa_{\mathcal{P}})\bar{Z}_{\mathcal{P}} + \sum_{i\in\mathcal{S}} \omega_i^{\mathcal{S}} \quad \geq \quad \chi_{\mathcal{P}} \bar{Z}_{\mathcal{P}} + \sum_{i\in\mathcal{S}} \frac{\bar{O}_i}{\mu_i} \geq 1,$$

which gives $\bar{Z}_{\mathcal{P}} \geq \tilde{Z}_{\mathcal{P}}$. If $P^{\mathcal{US}} = 0$, the second assertion follows from Proposition 2.4.3. □

We can now derive necessary conditions for the partitioning $\mathcal{P} = (\mathcal{S},\mathcal{U})$ to hold. We make use here of Lemma 2.4.4 and we therefore need the assumption of feed-forward routing.

Theorem 2.4.6. *Assume a given partitioning* $\mathcal{P} = (\mathcal{S},\mathcal{U})$. *Then for all* $i \in \mathcal{U}$

$$\frac{\omega_i}{\eta_i} > \tilde{Z}_{\mathcal{P}}.$$

Proof. For an unstable queue i, Q_i can be written as strictly positive for all $i \in \mathcal{U}$, which gives, using the traffic inequalities

$$\sum_{j=1}^{N} p_{ji}\bar{O}_j + \lambda_i - \bar{O}_i \geq Q_i > 0.$$

Using the two previous lemmas, it leads to

$$\forall i \in \mathcal{U},\ \sum_{j=1}^{N} p_{ji}\omega_j + \lambda_i - \eta_i \tilde{Z}_{\mathcal{P}} \geq \sum_{j=1}^{N} p_{ji}\bar{O}_j + \lambda_i - \bar{O}_i \geq Q_i > 0,$$

which gives using Lemma 2.4.4 that $-\eta_i \tilde{Z}_{\mathcal{P}} \geq \bar{O}_i$ and thus $\frac{\omega_i}{\eta_i} > \tilde{Z}_{\mathcal{P}}$. □

So far, only necessary conditions for a given rate stability partition of the queues follow from Theorem 2.4.6. We illustrate the obtained results on two examples, where the obtained necessary conditions turn out to be sufficient in the first example and not sufficient in the second.

In the next two sections, we derive *necessary and sufficient* conditions for rate stability under some of the Assumptions 2.2.1 to 2.2.4 for two important special cases. In Section 2.5 we study a two-queue tandem model, and in Section 2.6 we consider systems of parallel queues, with shared resources.

2.5 Two-queue tandem model

Consider the system of two queues in tandem, illustrated in Figure 2.1 with an asymptotically decreasing extended processor-sharing allocation (see Section 2.2). The routing matrix is given by

$$P = \begin{pmatrix} 0 & p \\ 0 & 0 \end{pmatrix}. \tag{2.28}$$

Thus, a fraction p of the output rate of the first queue is sent as input rate to the second queue. The following traffic equations and inequalities hold (Theorem 2.4.1)

$$\begin{aligned} Q_1 + \bar{O}_1 &= \lambda_1, \\ Q_2 + \bar{O}_2 &\leq p\bar{O}_1, \\ \frac{\bar{O}_1}{\mu_1} + \frac{\bar{O}_2}{\mu_2} &\geq 1. \end{aligned}$$

For the corresponding virtual model satisfying $\bar{\mathbf{O}} = \mathbf{O}$, the traffic equations are

$$\begin{aligned} \tilde{O}_1^{\mathcal{P}} &= \lambda_1 - \tilde{Q}_1^{\mathcal{P}}, \\ \tilde{O}_2^{\mathcal{P}} &= p\tilde{O}_1^{\mathcal{P}} - \tilde{Q}_2^{\mathcal{P}}, \\ \frac{\tilde{O}_1^{\mathcal{P}}}{\mu_1} &+ \frac{\tilde{O}_2^{\mathcal{P}}}{\mu_2} = 1, \\ \frac{\tilde{O}_i^{\mathcal{P}}}{\eta_i} &= \tilde{Z}_{\mathcal{P}}, \quad \text{for } i \in U. \end{aligned}$$

$\mathcal{P}$	$\tilde{Q}_1$	$\tilde{Q}_2$	$\tilde{O}_1$	$\tilde{O}_2$
$\mathcal{U} = \emptyset$	0	0	λ_1	$\lambda_1 p$
$\mathcal{S} = \{1\},\ \mathcal{U} = \{2\}$	0	$p\lambda_1 - (1 - \frac{\lambda_1}{\mu_1})\mu_2$	λ_1	$(1 - \frac{\lambda_1}{\mu_1})\mu_2$
$\mathcal{S} = \{2\},\ \mathcal{U} = \{1\}$	$\lambda_1 - \frac{\mu_1\mu_2}{p\mu_1+\mu_2}$	0	$\frac{\mu_1\mu_2}{p\mu_1+\mu_2}$	$\frac{p\mu_1\mu_2}{p\mu_1+\mu_2}$
$\mathcal{U} = \{1,\ 2\}$	$\lambda_1 - \frac{l_1\mu_1}{l_1+l_2}$	$\frac{pl_1\mu_1}{l_1+l_2} - \frac{l_2\mu_2}{l_1+l_2}$	$\frac{l_1\mu_1}{l_1+l_2}$	$\frac{l_2\mu_2}{l_1+l_2}$

Table 2.1: Output rates for the stability subsets.

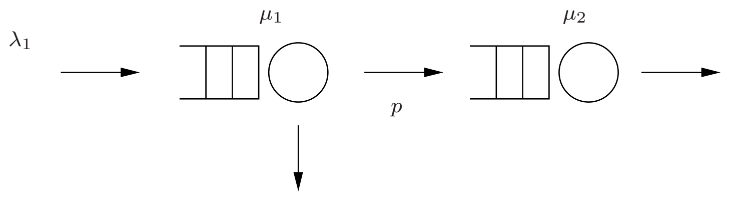


Figure 2.1: Two queues in tandem.

By $\mathcal{P}$ we denote the partition of queues according to their rate stability. $\mathcal{P}$ can thus be $\mathcal{U} = \emptyset$, and $\mathcal{S} = \{1\}$, $\mathcal{U} = \{2\}$, and $\mathcal{S} = \{2\}$, $\mathcal{U} = \{1\}$, and $\mathcal{U} = \{1,\ 2\}$. The solutions of $\tilde{\mathbf{O}}$ and $\tilde{\mathbf{Q}}$ are given in Table 2.1 for each stability subset $\mathcal{P}$.

According to Theorem 2.4.6 the network is globally stochastically stable if and only if $\rho < 1$ which reads

$$\lambda_1 < \frac{\mu_1\mu_2}{p\mu_1 + \mu_2}.$$

Note that in this case $\bar{\mathbf{O}} = \tilde{\mathbf{O}} = \mathbf{O}$ and $\bar{\mathbf{Q}} = \tilde{\mathbf{Q}} = \mathbf{Q}$.

Necessary conditions for $\mathcal{U} = \{1,\ 2\}$

For the partition $\mathcal{U} = \{1, 2\}$, given that $\tilde{Z}_{\mathcal{P}} = \frac{1}{l_1+l_2}\ \boldsymbol{\omega} = (\lambda_1, p\lambda_1)$, the following conditions given by Theorem 2.4.6 are necessary

$$p > \frac{l_2\mu_2}{l_1\mu_1},$$

$$\lambda_1 > \frac{\mu_1 l_1}{l_1 + l_2}.$$

Using the last assertion in Theorem 2.4.1, we further obtain that $\bar{Z}_{\mathcal{P}} = \tilde{Z}_{\mathcal{P}}$.

Necessary conditions for $\mathcal{S} = \{1\},\ \mathcal{U} = \{2\}$

For the partitions $\mathcal{U} = \{1\}, \mathcal{S} = \{2\}$ and $\mathcal{S} = \{1\}, \mathcal{U} = \{2\}$, the necessary conditions stated in Theorem 2.4.6 lead to the already known condition $\rho > 1$. Using Theorem 2.4.6 ($\bar{Z}_{\mathcal{P}} > \tilde{Z}_{\mathcal{P}}$), the first traffic equation and the additional inequalities given by Theorem 2.4.1 and Proposition 2.3.2, we obtain

$$\frac{\lambda_1}{l_1\mu_1} = \frac{\tilde{O}_1^{(\mathcal{S}=\{1\},\mathcal{U}=\{2\})}}{\eta_1} = \frac{\bar{O}_1^{(\mathcal{S}=\{1\},\mathcal{U}=\{2\})}}{\eta_1} < \frac{\bar{O}_1^{(\mathcal{U}=\{1,2\})}}{\eta_1} = \frac{\tilde{O}_1^{(\mathcal{U}=\{1,2\})}}{\eta_1} = \frac{1}{l_1 + l_2},$$

which leads to the necessary condition $\lambda_1 < \frac{\mu_1 l_1}{l_1 + l_2}$.

Necessary conditions for $\mathcal{U} = \{1\},\ \mathcal{S} = \{2\}$

$$\frac{\mu_1 p \mu_2}{(\mu_1 p + \mu_2) l_2 \mu_2} = \frac{\tilde{O}_2^{(\mathcal{U}=\{1\},\mathcal{S}=\{2\})}}{\eta_2} \leq \frac{\bar{O}_2^{(\mathcal{U}=\{1\},\mathcal{S}=\{2\})}}{\eta_2},$$

and

$$\frac{\bar{O}_2^{(\mathcal{U}=\{1\},\mathcal{S}=\{2\})}}{\eta_2} < \frac{\bar{O}_2^{(\mathcal{U}=\{1,2\})}}{\eta_2} = \frac{\tilde{O}_2^{(\mathcal{U}=\{1,2\})}}{\eta_2} = \frac{1}{l_1 + l_2},$$

and this leads to the necessary condition that $p < \frac{l_2 \mu_2}{l_1 \mu_1}$.

The obtained necessary conditions are easily seen to lead to a complete partitioning of the parameter set, which gives a sharp characterization of the stability set. As a consequence, the obtained conditions are *both necessary and sufficient*, except on a boundary set of input parameters.

In Figure 2.2 the stability set is depicted for two different sets of input parameters.

2.6 Parallel queues

In this section, we consider parallel queues and thus suppose that there is no internal routing, i.e., $p_{ij} = 0$, for all i, j. In that case, we can derive a sharp characterization of the per-queue rate stability. To this end, we first show that in that case, the traffic inequalities are actually a set of traffic equations (Theorem 2.6.1). This allows to prove that the output rates and asymptotic growth rates converge. Using the results of Sections 2.3 and 2.4, we then derive a characterization of the per-queue stability (Theorem 2.6.6).

2.6.1 Extended traffic equations

In this subsection, we specify the traffic inequalities obtained in the general case by deriving traffic *equations* linking the input rates and the asymptotic output rates of the network.

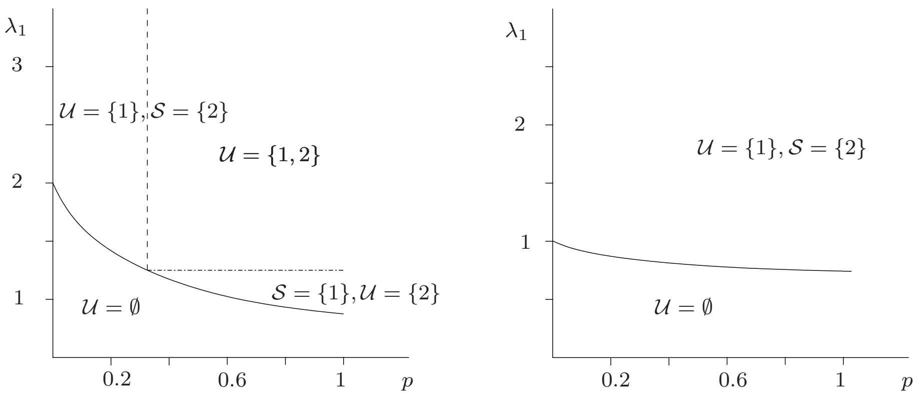


Figure 2.2: Stability regions with $(\mu_1, l_1, \mu_2, l_2) = (3, 1, 1, 1)$ in the left figure, and with $(\mu_1, l_1, \mu_2, l_2) = (1, 1, 3, 1)$ in the right figure.

Theorem 2.6.1 (Extended traffic equations)**.** *The asymptotic output rates* $\mathbf{O}$*,* $\bar{\mathbf{O}}$ *and growth rates* $\mathbf{Q}$*,* $\bar{\mathbf{Q}}$ *are finite and satisfy the following linear equations. For* $i = 1, \ldots, N$*,*

$$Q_i + \bar{O}_i = \lambda_i, \tag{2.29}$$

$$\bar{Q}_i + O_i = \lambda_i. \tag{2.30}$$

Proof. We follow the same lines as in Theorem 2.4.1,

$$M_i(t) := X_i(t) - X_i(0) - \int_0^t \{\lambda_i - \phi_i(\mathbf{X}(s))\}ds, \tag{2.31}$$

is a martingale that satisfies $E[M_i^2(t)] < Kt$ for $i = 1, \ldots, N$, $t > 0$ and some $K > 0$. This implies that $\limsup_{t\to\infty} \frac{X_i(t)}{t} < +\infty$ and $\liminf_{t\to\infty} Y_i(t) = \lambda_i - \limsup_{t\to\infty} \varphi_i(t)$. Using the dominated convergence theorem, we get Equations (2.29) and (2.30). □

2.6.2 Output rates convergence

We fix $\mathcal{P}$ to be a partition of queues such that queues in $\mathcal{S}$ are rate stable while queues in $\mathcal{U}$ are rate unstable. In the following proposition, we prove that the output rates of the different queues converge which further allows a complete description of the rate stability set.

Proposition 2.6.2. *Consider a set of parallel queues with a decreasing allocation* *satisfying the Assumptions* 2.2.1, 2.2.2, *and* 2.2.4. *Then, for* $t \to \infty$,

$$\frac{X_i(t)}{t} \to Q_i, \textit{ in probability}, \tag{2.32}$$

$$\varphi_i(t) \to O_i, \textit{ in probability}, \tag{2.33}$$

with

$$\tilde{O}_i = \lambda_i \quad (i \in \mathcal{S}), \qquad \tilde{O}_i = \tilde{Z}_{\mathcal{P}} \eta_i \quad (i \in \mathcal{U}),$$

$$\tilde{Q}_i = 0 \quad (i \in \mathcal{S}), \quad \tilde{Q}_i = \lambda_i - \tilde{Z}_{\mathcal{P}} \eta_i \quad (i \in \mathcal{U}),$$

where

$$\tilde{Z}_{\mathcal{P}} := \frac{1 - \sum_{j \in \mathcal{S}} \frac{\lambda_j}{\mu_j}}{\sum_{j \in \mathcal{U}} l_j} = \frac{1 - \sum_{j \in \mathcal{S}} \rho_j}{\chi_{\mathcal{P}}}.$$

Proof. Let us first prove the convergence of the rates. Note that an asymptotically monotone homogeneous allocation is actually monotone. Using the homogeneity and the monotonicity of the allocation, we get that for t large and for $i = 1, \ldots, N$,

$$\phi_i(X_i(t)) = \phi_i\left(\frac{\mathbf{X}(t)}{t}\right) \geq \phi_i(\bar{\mathbf{Q}}),$$

which implies

$$\varphi_i(t) \geq \phi_i(\bar{\mathbf{Q}}).$$

This leads to

$$\bar{Q}_i \leq \lambda_i - \phi_i(\bar{\mathbf{Q}}) \quad i = 1, \ldots, N.$$

Similarly, for $i \in \mathcal{U}$,

$$Q_i \geq \lambda_i - \phi_i(\mathbf{Q}), \quad i = 1, \ldots, N.$$

Summing these inequalities for $i = 1, \ldots, N$ and using the property of a work-conserving allocation, we obtain that

$$\sum_{i=1}^{N} \frac{Q_i}{\mu_i} \geq \sum_{i=1}^{N} \frac{\lambda_i}{\mu_i} - 1 \geq \sum_{i=1}^{N} \frac{\bar{Q}_i}{\mu_i}.$$

Since $\bar{Q}_i > Q_i$, and $\bar{Q}_i > 0$ for $i = 1, \ldots, N$, we hence deduce that $\bar{Q}_i = Q_i$ and as a consequence for $i = 1, \ldots, N$,

$$\bar{O}_i = O_i = \tilde{O}_i. \tag{2.34}$$

The convergence in L^1 of $\varphi_i(t)$ to a constant implies the convergence in probability of $\varphi_i(t)$ which combined with the almost sure convergence of the difference $Y_i(t) - \varphi_i(t)$ implies the convergence of $Y_i(t)$ in probability. The traffic equations defined previously together with the system of equations in Definition 2.4.2 allow us to complete the proof. □

Remark 2.6.3. It appears plausible to prove an almost sure convergence for these processes even without the assumption of exponential service times and Poisson arrivals. This result is out of the scope of this study but we refer to the method presented in [58] and further used for a set of discriminatory processor-sharing queues (DPS) in [3] for such a derivation. These techniques, jointly used with the traffic conservation used here would prove the stated convergence in the context of stationary marked point processes.

Remark 2.6.4. Note that we are not able to prove Equation (2.34) in general and therefore the results do not apply to feed-forward routing. However, if Equation (2.34) can be proven for feed-forward routing the results for parallel queues also hold for feed-forward routing.

2.6.3 Characterization of the per-queue rate stability

We assume without loss of generality that the queues are ranked in decreasing order of the loads $\zeta_i := \frac{\lambda_i}{l_i\mu_i}$, in the sense that

$$\zeta_1 \le \cdots \le \zeta_N. \tag{2.35}$$

The following result shows the relation between the ordering of the queues and the per-queue rate stability.

Proposition 2.6.5. *If queue i is rate stable and $j < i$, then queue j is also rate stable.*

Proof. Suppose $j \in \mathcal{U}$, $i \in \mathcal{S}$ and $j < i$. From Proposition 2.3.2, we get $\frac{\bar{O}_i}{\eta_i} < \frac{\bar{O}_j}{\eta_j}$. From Theorem 2.4.1, it follows that $\bar{O}_i = \lambda_i$ and that $\bar{O}_j \le \lambda_j$. We thus find that

$$\zeta_i = \frac{\bar{O}_i}{\eta_i} < \frac{\bar{O}_j}{\eta_j} \le \frac{\lambda_j}{\eta_j}. \tag{2.36}$$

This contradicts $\zeta_j \le \zeta_i$. □

Denote $\tilde{Z}(m) = \tilde{Z}_{\{1,\dots,m\}} = \frac{1-\sum_{i\le m}\rho_i}{\sum_{i>m} l_i}$. The following result shows that the partitioning $\mathcal{P} = (\mathcal{S},\mathcal{U})$ has a simple structure.

Theorem 2.6.6 (Structure of stability partitioning)**.** *Consider a set of parallel queues with a decreasing allocation verifying the Assumptions* 2.2.1, 2.2.2 *and* 2.2.4. *The stability partitioning $\mathcal{P} = (\mathcal{S},\mathcal{U})$ is characterized as follows $\mathcal{P} = (\mathcal{S},\mathcal{U})$ with $\mathcal{S} = \{1,\dots,m\}$ and $\mathcal{U} = \{m+1,\dots,N\}$ if and only if*

$$\zeta_m \le \tilde{Z}(m) < \zeta_{m+1}. \tag{2.37}$$

Proof. Using Proposition 2.6.5, there exists a k such that $\mathcal{S} = \{1,\dots,k\}$ and $\mathcal{U} = \{k+1,\dots N\}$. Theorem 2.4.6 combined with Proposition 2.6.2 gives that $\tilde{Z}(k) = \bar{Z}(k) < \zeta(k+1)$. Proposition 2.6.5 gives

$$\frac{\bar{O}_k}{\eta_k} \le \frac{\bar{O}_{k+1}}{\eta_{k+1}},$$

which in combination with the traffic equations leads to

$$\zeta_k \leq \tilde{Z}(k).$$

As $\tilde{Z}(\cdot)$ is a decreasing function, we conclude that $m = k$. □

We emphasize that Theorem 2.6.6 gives a complete characterization of the rate stability partitioning for model instances that satisfy Assumptions 2.2.1, 2.2.2, and 2.2.4 and are monotone. Typical examples of such allocations are the coupled-processors allocation (defined in Section 2.2.1), and utility-based allocations on some tree topology (see [18]).

2.7 Conclusion and topics for further research

The results presented in this study provide new intuition and fundamental insight in the stability and throughput behavior of queueing models in which resources are shared among different queues. These results should be viewed as a first step in understanding the behavior of this type of queueing networks, and open up a wealth of challenging open research questions. Some of these questions will be briefly touched upon below.

In the context of stability and throughput characteristics, several interesting questions remain to be answered. First, when X is a continuous-time Markov process, it actually remains an open and crucial question to know for which input parameters, rate instability of queue i coincides to the convergence of X_i to infinity (either in probability or in law). In [19], per-queue stochastic stability is established for parallel queues with *monotone* allocation functions. It is remarkable that, except possibly on the boundary of the stability sets, the conditions for rate instability (and thus stochastic instability) that we have derived here coincide with the sharp characterization of the stochastic instability set given in [19]. This encouraging observation calls for a generalization of this result to more complex topologies.

Second, the derivation of necessary conditions for rate stability for models that are not covered by the ones discussed in Sections 2.5 and 2.6 is an open area. For example, consider a seemingly simple three-queue network where all jobs arrive at queue 1, and then either move to queue 2 (with probability p_1) or to queue 3 (with probability p_2) before departing from the system, with $0 \leq p_1 + p_2 < 1$. Then it can be shown that the necessary conditions obtained in this study do not lead to a full partitioning of the parameter set. This observation shows that an extension of the necessary conditions presented in Sections 2.3 and 2.4 to a broader class of models is far from trivial, and provides an open area for further research. In addition to considering stability and throughput, one may also be interested in other performance metrics such as steady-state sojourn-time distributions of jobs at the different queues, the optimal static or dynamic assignment of servers to the queues depending on the state of the system. Derivation of such results is another interesting topic for further research.

CHAPTER 3

PRODUCT-FORM SOLUTIONS

In this chapter we focus on networks with queues either in tandem or in parallel, with a common resource shared among the queues. First, a necessary and sufficient criterion, called adjoint reversibility, is provided to decide whether the system possesses a product form or not. This criterion unifies both the parallel (a reversible) and the tandem (a non-reversible) system in one product-form theorem. Next, the criterion is applied separately for the parallel and tandem system to obtain a number of new product-form examples which also includes non-balanced capacity sharing. Despite of, but also due to, the different parallel and tandem mechanisms we observe that for certain examples the product form has the same structure, while for others there are essential differences. In addition, it is proven that several models cannot have a product-form result. This chapter is based on [117].

3.1 Introduction

In this chapter we study perhaps the simplest non-trivial class of queueing models in which resources are shared: a network of queues, either in tandem or in parallel, in which a common resource is shared amongst the servers at both queues. For this class of models, we derive a variety of product-form and non-product-form results, leading to fundamental insight and understanding in the behavior of queueing networks with shared resources.

The literature on product-forms is extensive. In [56, 57], the authors provide product-form results for job-shop networks. The well-known BCMP-paper for computer applications [10] and other extensions of networks having a product form can be found in [64]. Schassberger [103], Pittel [91] and Hordijk and Van Dijk [53, 54] contribute product-form results, including blocking and non work-conserving service disciplines. Specific product-form results for processor-sharing systems are presented, most notably, in [17] and [25]. More essentially though, in these references, the capacity allocation functions are assumed to be strictly positive. In [17] Bonald and Proutière show that the stationary distribution of a network is insensitive with respect to the service-time distribution if and only if the service capacities are balanced, considering networks with state-dependent service rates and state-dependent arrival rates. Van Dijk [34, 36] provides sufficient and necessary conditions for a network to possess a product-form solution. The focus in these references is on blocking.

In this study, in contrast to the literature given above, the focus is on the sharing of the service capacity. In addition, it is studied whether or not the parallel and tandem models are equivalent with respect to their product forms. In particular, the product-form results are compared for the tandem and parallel model with similar sharing functions. We specify the criterion in [34, 36] to give both *necessary* and *sufficient* conditions for the existence of a product-form solution to a general setting of service sharing among two stations in either parallel or tandem. A theorem is provided to unify models despite different routing mechanisms, leading to comparable (and similar) product-form solutions. The product-form behavior of a range of model examples will be analyzed. This covers the standard processor-sharing mechanism in which the resource is fairly shared among the jobs in the system; note that for this model the existence of a product form is well-known, but that we give an alternative approach to prove this. Moreover, new product-form results for non-standard PS models are obtained, e.g., where the resource sharing may be unproportional and where service may be stopped. This analysis leads to a number of new product-form results, including results on models not having product-form solutions.

The set up of this chapter as follows. In Section 3.2 the models investigated in this chapter are described and relevant notation and definitions are introduced. Also the general condition for models to possess a product-form solution or not is given. In Section 3.3 the parallel model is discussed in detail and examples are given for several capacity allocations and state space truncations leading to product-form solutions and non-product form solutions. Also the two-queue extension of the LPS service discipline is considered here. In Section 3.4 similar results are presented for the tandem model. After a discussion in Section 3.6 we conclude with addressing a number of challenging topics for further research.

3.2 The models and general product-form characterization

We restrict the presentation to queueing networks with two service queues and investigate product-form properties of these queueing networks, where the networks have the following specific features: (1) state-dependent service sharing, where the per-queue service rates depend on the state of the entire system, (2) services can be fully stopped at a queue, even if jobs are present at that queue, and (3) incoming jobs can be denied access to the system. For the networks we focus on the sharing of the capacity, more than on blocking, which is motivated by the applications introduced in Section 3.1. We focus on networks with only two queues since the complexities with respect to product forms manifest themselves for these networks, while the behavioral insights and intuition can be obtained by illustrations.

We consider two models; a model with two queues in parallel (PM), and a model with two queues in series, a tandem model (TM). For these models we first introduce some common notation. Denote the state of the system by $\mathbf{x} = (x_1, x_2)$, where x_i denotes the number of jobs present (i.e., waiting or in service) at queue i ($i = 1,\ 2$). The state space is denoted by $\mathcal{C}$, where $\mathcal{C} = \{\mathbf{x} \mid \mathbf{x_1}, \mathbf{x_2} \geq \mathbf{0}\}$. Let the total amount of service capacity offered to all jobs in service at queue i denoted by $\phi_i(\mathbf{x}) \geq 0$, for $i = 1,\ 2$. We assume that an empty queue does not receive service capacity (i.e., $\phi_i(\mathbf{x}) = 0$ if $x_i = 0$). The service times at queue i are exponentially distributed with mean $\beta_i = \mu_i^{-1}$. Given this notation, we now define the two different models.

3.2.1 Parallel model

Consider a network of two queues in parallel. Jobs arrive at queue i according to a Poisson process with rate λ_i ($i = 1,\ 2$). After completion of service at queue i a job leaves the network. Upon arrival at queue i, an incoming job is either accepted or blocked, depending on the state of the system, $\mathbf{x}$. This admission policy, denoted by the blocking function $b_i(\mathbf{x}) \in \{0, 1\}$ ($i = 1,\ 2$), is defined as follows: If $b_i(\mathbf{x}) = 1$ then a job arriving at queue i is accepted, and if $b_i(\mathbf{x}) = 0$ the job is blocked. In Section 3.3 we focus on product forms for this model, given a function $\phi_i(\mathbf{x})$, for $i = 1,\ 2$. A first example of this model is presented in Figure 3.1. In this example, which will be discussed in detail in Section 3.3.5 the state $\mathbf{x}$ equals $(4, 2)$. The capacity assignment is based on a processor-sharing discipline, jobs in service receive a fair share of the total capacity, and in this example three jobs are in service in the first queue, and two jobs in the second queue, all receiving a fifth of the total capacity of the shared resource.

3.2.2 Tandem model

For the tandem model (TM) we consider a network consisting of two queues in series. The jobs arrive at queue 1 according to a Poisson process with rate λ. After completion of service at this queue jobs are forwarded to queue 2; after receiving service at queue 2 jobs depart from the network. There are no external arrivals to the second queue. Upon arrival at the system, an incoming job is either accepted or blocked, depending on the state of the system. To this end, we again denote an admission policy, for the tandem model by $b_1(\mathbf{x}), \mathbf{x} \geq \mathbf{0}$, where $b_1(\mathbf{x}) := 1$ if an arriving job is accepted, and $b_1(\mathbf{x}) := 0$ otherwise. In Section 3.4 this model and

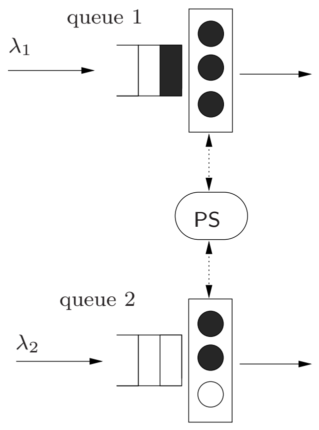


Figure 3.1: The parallel model.

its product-form results are presented and discussed.

Figure 3.2 illustrates an example of this model where the capacity assignment is again based on a processor-sharing discipline. Note that in this figure, as well as in Figure 3.1; $\mathbf{x} = (4, 2)$, with three jobs in service at the first queue and two at the second queue.

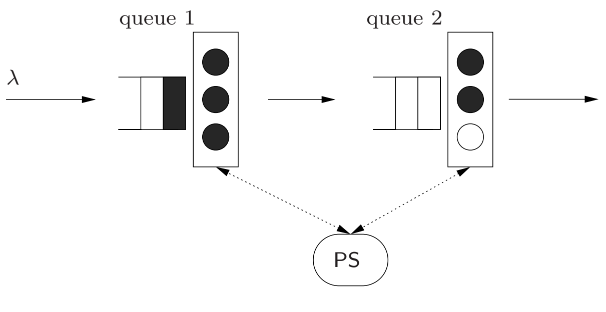


Figure 3.2: The tandem model.

We note that the three features discussed above are included in both model descriptions. State-dependent service sharing is captured in the function $\phi_i(\mathbf{x})$, which includes the possibility to provide no service to queue i by taking $\phi_i(\mathbf{x}) := 0$ for some $x_i > 0$. Access blocking is described by $b_i(\mathbf{x})$.

3.2.3 A unifying product-form characterization

Under natural ergodicity assumptions for its existence, let $\pi(\mathbf{x})$ denote the corresponding steady-state distribution. In this section we present a general criterion that gives both necessary and sufficient conditions for $\pi(\mathbf{x})$ to possess a product-form solution. Here the standard notion of a product form is given by the following definition

A product form is defined as the factorization of the steady-state joint queue distribution to the steady-state single-queue distribution, up to a normalization constant [34].

Station balance

As will be shown below, the existence of a product form can be characterized by the so-called notion of reversibility, not necessarily of the underlying Markov chain itself but of a special constructed Markov chain that will be called the adjoint Markov chain. This notion of reversibility reflects the phenomenon that a chain would stochastically evolve in the same way if we could reverse time (see [64] for an elegant and extensive exposition of this concept).

The construction of the adjoint Markov chain depends on the specific application of interest in order for a notion of queue balance to be satisfied, i.e.,

$$\begin{array}{l} The\ rate\ out\ of\ a\ state\ \mathbf{x}\ due\ to\ a\ departure\ at\ a\ queue\ i = \\ the\ rate\ into\ that\ state\ \mathbf{x}\ due\ to\ an\ arrival\ at\ that\ queue\ i. \end{array} \tag{3.1}$$

Whether this queue balance is indeed satisfied, which in turn appears to be directly related to a product form, then remains to be seen and is one-to-one related to the reversibility of the adjoint Markov chain (defined in Section 3.2.3). The reversibility of the adjoint Markov chain requires the existence of a stationary distribution $\bar{\pi}$, such that $\bar{\pi}(i)\bar{q}_{i,j} = \bar{\pi}(j)\bar{q}_{j,i}$, where $\bar{q}_{i,j}$ are the transition rates of the adjoint Markov chain.

First let us make the constructions of the adjoint chain explicit for the parallel and tandem model. For the parallel model the construction of the adjoint transition rates appears to be identical up to service scaling, as of the original model. For the tandem model, in contrast, the construction of the adjoint Markov chain is necessary and different as the model itself is not reversible. It is obtained by the transition rates of the original model supplemented with transition rates in the opposite direction.

From here on we adopt the state notation $\mathbf{x} = (x_1, x_2)$ as in Section 3.2, with x_i the number of jobs at queue $i = 1,\ 2$, and we assume the existence of a stationary distribution $\pi(\mathbf{x})$ at some set of admissible states C, where $C \in \mathcal{C}$. Hence, $\pi(\mathbf{x}) = 0$ for $\mathbf{x} \notin C$. The following notation is convenient throughout. Let $\mathbf{e}_i$ denote the i^{th} unit vector, for $i = 1,\ 2$, and let $\mathbf{0} := (0, 0)$. Finally, denote by $\mathbb{1}_E$ the indicator function for an event E, i.e., $\mathbb{1}_E = 1$ if event E is satisfied and 0 if not. We recall Sections 3.2.1 and 3.2.2 for the model descriptions.

For the **parallel model** the Kolmogorov or global balance equations for a state $\mathbf{x} \in C$, become

$$\left\{\begin{array}{l} \pi(\mathbf{x})\lambda_1 b_1(\mathbf{x}) + \\ \pi(\mathbf{x})\lambda_2 b_2(\mathbf{x}) + \\ \pi(\mathbf{x})\mu_1\phi_1(\mathbf{x}) + \\ \pi(\mathbf{x})\mu_2\phi_2(\mathbf{x}) \end{array}\right\} \begin{array}{l} (3.2.1) \\ (3.2.2) \\ (3.2.3) \\ (3.2.4) \end{array}$$
$$= \tag{3.2}$$
$$\left\{\begin{array}{l} \pi(\mathbf{x}+\mathbf{e}_1)\mu_1\phi_1(\mathbf{x}+\mathbf{e}_1) + \\ \pi(\mathbf{x}+\mathbf{e}_2)\mu_2\phi_2(\mathbf{x}+\mathbf{e}_2) + \\ \pi(\mathbf{x}-\mathbf{e}_1)\lambda_1 b_1(\mathbf{x}-\mathbf{e}_1) + \\ \pi(\mathbf{x}-\mathbf{e}_2)\lambda_2 b_2(\mathbf{x}-\mathbf{e}_2) \end{array}\right\} \begin{array}{l} (3.2.1') \\ (3.2.2') \\ (3.2.3') \\ (3.2.4') \end{array}$$

For the **tandem model** the global balance equations are, for $\mathbf{x} \in C$

$$\left\{\begin{array}{l} \pi(\mathbf{x})\lambda b_1(\mathbf{x}) + \\ \pi(\mathbf{x})\mu_1\phi_1(\mathbf{x}) + \\ \pi(\mathbf{x})\mu_2\phi_2(\mathbf{x}) \end{array}\right\} \begin{array}{l} (3.3.1) \\ (3.3.2) \\ (3.3.3) \end{array}$$
$$= \tag{3.3}$$
$$\left\{\begin{array}{l} \pi(\mathbf{x}+\mathbf{e}_2)\mu_2\phi_2(\mathbf{x}+\mathbf{e}_2) + \\ \pi(\mathbf{x}+\mathbf{e}_1-\mathbf{e}_2)\mu_1\phi_1(\mathbf{x}+\mathbf{e}_1-\mathbf{e}_2) + \\ \pi(\mathbf{x}-\mathbf{e}_1)\lambda b_1(\mathbf{x}-\mathbf{e}_1) \end{array}\right\} \begin{array}{l} (3.3.1') \\ (3.3.2') \\ (3.3.3') \end{array}$$

We cannot expect to obtain analytic solutions for equations (3.2) and (3.3), unless these equations are satisfied by the more detailed equations (3.2.i)=(3.2.i') for $i = 1,\ 2,\ 3,\ 4$ for the parallel model and (3.3.i)=(3.3.i') for $i = 1,\ 2,\ 3$ for the tandem model. These more detailed relations will be referred to as *station balance* relations.

Adjoint Markov chains

In this section we will define the adjoint transition rates $\bar{q}$ for the parallel and the tandem model. For the **parallel model**, as the routing itself can be seen as reversible, the transition rates of the adjoint Markov chain can be chosen as for the original Markov chain, up to service scaling by

$$\begin{array}{lcl} \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1) & := & \lambda_1 b_1(\mathbf{x}), \\ \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_2) & := & \lambda_2 b_2(\mathbf{x}), \\ \bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_1) & := & \phi_1(\mathbf{x}), \\ \bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_2) & := & \phi_2(\mathbf{x}), \\ \bar{q}(\mathbf{x}_1,\mathbf{x}_2) & := & 0, \text{ otherwise.} \end{array} \tag{3.4}$$

For the **tandem model** the routing has a triangular form and is not reversible itself, since transitions only take place in one direction. In line with the detailed equations (3.3.i)=(3.3.i') for $i = 1,\ 2,\ 3$, therefore, define the adjoint Markov chain by constructing transition rates in opposite direction as follows

$$\begin{array}{lcl} \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1) & := & \lambda b_1(\mathbf{x}), \\ \bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_1+\mathbf{e}_2) & := & \phi_1(\mathbf{x}), \\ \bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_2) & := & \phi_2(\mathbf{x}), \end{array}$$

supplemented with

$$\begin{array}{lll} \bar{q}(\mathbf{x}+\mathbf{e}_1,\mathbf{x}) & := & \phi_1(\mathbf{x}-\mathbf{e}_1+\mathbf{e}_2), \\ \bar{q}(\mathbf{x}-\mathbf{e}_1+\mathbf{e}_2,\mathbf{x}) & := & \phi_2(\mathbf{x}-\mathbf{e}_2), \\ \bar{q}(\mathbf{x}-\mathbf{e}_2,\mathbf{x}) & := & \lambda b_1(\mathbf{x}), \\ \bar{q}(\mathbf{x}_1,\mathbf{x}_2) & := & 0, \text{ otherwise.} \end{array} \tag{3.5}$$

Note that this adjoint Markov chain coincides with the parameterization of the original tandem network up to exponential service parameters in the natural queue flow direction from queue i to queue $i+1$. In contrast though, also a flow in opposite direction has been constructed. The general definition of the transition rates of the adjoint Markov chain are as follows. Consider a queue i and a transition rate γ from queue i to some queue $i+1$, then

$$\bar{q}(\mathbf{x}+\mathbf{e}_i,\mathbf{x}+\mathbf{e}_{i+1}) = \gamma \quad (\textit{as original Markov chain}),$$

and

$$\bar{q}(\mathbf{x}+\mathbf{e}_i,\mathbf{x}+\mathbf{e}_{i-1}) = \gamma \quad (\textit{as new}),$$

as is presented in [34].

Product-form result

Both the parallel and the tandem model can now be characterized by one unifying theorem. To the best of the author's knowledge, this seems to be new in the literature. It characterizes the existence of a product-form solution by means of reversibility of the adjoint Markov chain, which we will refer to as *adjoint reversibility.*

Theorem 3.2.1. *There exists a product-form steady-state distribution of the form*

$$\pi(\mathbf{x}) = cH(\mathbf{x})\prod_i \left[\frac{1}{\mu_i}\right]^{x_i}, \quad \textit{for all } \mathbf{x}\in C \tag{3.6}$$

with c a normalizing constant, if and only if the adjoint Markov chain is reversible. That is for some solution $\tilde{\pi}(\mathbf{x})$ of the adjoint Markov chain and for all pairs of states $\mathbf{x}_1,\mathbf{x}_2\in C$

$$\tilde{\pi}(\mathbf{x}_1)\bar{q}(\mathbf{x}_1,\mathbf{x}_2) = \tilde{\pi}(\mathbf{x}_2)\bar{q}(\mathbf{x}_2,\mathbf{x}_1). \tag{3.7}$$

Proof. The proof is concluded directly by substitution of Equation (3.4) in Equation (3.2) and showing that $(3.2.i) = (3.2.i')$ for $i = 1,\ 2,\ 3,\ 4$ for the parallel model, and similarly, by substitution (3.5) in (3.3) showing that $(3.3.i) = (3.3.i')$ for $i = 1,\ 2,\ 3$ for the tandem model. □

Note that the result stated in Theorem 3.2.1 also holds for more than two queues.

Reversibility characterization

Once again, it is important to observe that reversibility appears as a key characterization for a product form. However, it does *not* imply that the model *itself* needs to be reversible. Furthermore, in that case the stationary distribution $\{\pi(i)\}$ of the original chain coincides with that of the adjoint Markov chain $\{\bar{\pi}(i)\}$ up to scaling factors of the mean service times. See Remark 3.2.5 below for references.

The major advantage of Theorem 3.2.1 is that it enables one to verify the existence of a product form (3.6), by simply investigating the existence of a reversible solution $H(\mathbf{x})$. This, in turn, can be verified by the so-called Kolmogorov criterion (see for example [83]) as based upon just the transition rates as defined by (3.4) and (3.5). The Kolmogorov criterion for the adjoint Markov chain need to be verified. Below we present the reversibility conditions in more detail, for two reasons: (1) for the readability of the chapter and (2) to use these reversibility characterizations explicitly later on in the proofs for product-form and non-product form results for the parallel and tandem model. To verify reversibility of the adjoint Markov chain, we need to verify if one of the two conditions below, (3.8) or (3.11), holds.

Lemma 3.2.2 (Equivalent adjoint reversibility conditions)**.** *The following two conditions are equivalent for the reversibility of the adjoint Markov chain as in* (3.7)*.*

1. *For any cycle of the form p of any length t and its reverse cycle of the form $\bar{p}$*

$$\theta(p) \quad = \quad \theta(\bar{p}), \tag{3.8}$$

where,

$$\begin{array}{lcl} p & := & \mathbf{x}_0 \to \mathbf{x}_1 \to \ldots \to \mathbf{x}_t \to \mathbf{x}_{t+1} = \mathbf{x}_0, \\ \bar{p} & := & \mathbf{x}_0 = \mathbf{x}_{t+1} \to \mathbf{x}_t \to \ldots \to \mathbf{x}_1 \to \mathbf{x}_0, \end{array} \tag{3.9}$$

with their products of transitions rates

$$\begin{array}{lcl} \theta(p) & := & \bar{q}(\mathbf{x}_0, \mathbf{x}_1)\bar{q}(\mathbf{x}_1, \mathbf{x}_2)\ldots\bar{q}(\mathbf{x}_t, \mathbf{x}_0), \\ \theta(\bar{p}) & := & \bar{q}(\mathbf{x}_0, \mathbf{x}_t)\bar{q}(\mathbf{x}_t, \mathbf{x}_{t-1})\ldots\bar{q}(\mathbf{x}_1, \mathbf{x}_0). \end{array} \tag{3.10}$$

2. *There exists a function $H(\mathbf{x})$ such that for any fixed $\mathbf{x}_0 \in C$ and any state $\mathbf{x} \in C$ it holds that*

$$H(\mathbf{x}) = H(\mathbf{x}_0) \prod_{k=0}^{K-1} \left[\frac{\bar{q}(\mathbf{x}_k, \mathbf{x}_{k+1})}{\bar{q}(\mathbf{x}_{k+1}, \mathbf{x}_k)} \right], \ \textit{for any path } \mathbf{x}_0 \to \mathbf{x}_1 \to \ldots \to \mathbf{x}_K = \mathbf{x}, \ \textit{for which the denominator is positive.} \tag{3.11}$$

This means that $H(\mathbf{x})$ is independent of the path $\mathbf{x}_1 \to \ldots \to \mathbf{x}_{K-1}$; it only depends on $\mathbf{x}_0$ and $\mathbf{x}_K$.

Proof. This can be concluded from substitution of Equation (3.11) in (3.7) or indirectly as by [64] for the characterization of reversibility for the adjoint Markov chain. Note that we have

$$\frac{H(\mathbf{x}_k)}{H(\mathbf{x}_{k+1})} = \frac{\bar{q}(\mathbf{x}_k, \mathbf{x}_{k+1})}{\bar{q}(\mathbf{x}_{k+1}, \mathbf{x}_k)}, \tag{3.12}$$

and since we assume Equation (3.6) has a unique solution up to a normalization constant we can directly derive Equation (3.11). □

Either one of the two checks above in turn can generally be reduced to basic cycles or short paths that directly suggest a necessary form of the function $H(\mathbf{x})$ and a decomposition in a service and routing component, satisfying

$$\frac{H(\mathbf{x} + \mathbf{e}_i)}{H(\mathbf{x} + \mathbf{e}_j)} = \frac{\phi_i(\mathbf{x} + \mathbf{e}_i)}{\phi_j(\mathbf{x} + \mathbf{e}_j)} \frac{b_i(\mathbf{x})}{b_j(\mathbf{x})}. \tag{3.13}$$

Note that for $\mathbf{x} \in C$ if $\mathbf{x} + \mathbf{e}_j \notin C$ then $b_j(\mathbf{x}) = 0$. This equation appears to be the most explicit form to find a suggestion for the function $H(\mathbf{x})$.

Remark 3.2.3. From the condition given in Equation (3.13) it follows that the structure of the product form does not depend on the routing mechanism, whether parallel or in tandem.

For the applications in Sections 3.3.5 and 3.4.5 also the following corollary will appear to be useful.

Corollary 3.2.4. *A product form does not exist if for some pair of states* $\mathbf{x}_s$ *and* $\mathbf{x}_t$ *for some paths* p_1 *and* p_2 *and their reversed paths* $\bar{p}_1$ *and* $\bar{p}_2$*, it holds that*

$$\Theta(p_1) \neq \Theta(p_2), \tag{3.14}$$

with

$$\Theta(p_i) := \frac{\theta(p_i)}{\theta(\bar{p}_i)}, \tag{3.15}$$

and where p_1 *and* p_2 *are paths defined as follows*

$$\begin{array}{rcl} p_1 & := & \mathbf{x}_s \rightarrow \mathbf{x}_1 \rightarrow \ldots \rightarrow \mathbf{x}_{K-1} \rightarrow \mathbf{x}_K = \mathbf{x}_t, \\ p_2 & := & \mathbf{x}_s \rightarrow \mathbf{x}'_1 \rightarrow \ldots \rightarrow \mathbf{x}'_{K'-1} \rightarrow \mathbf{x}'_{K'} = \mathbf{x}_t. \end{array} \tag{3.16}$$

Remark 3.2.5. *(Literature)* The concept of an adjoint (artificial) Markov chain to characterize the existence of a product form has first been introduced and exploited in [53] and extended in [54]. For the case of a single job-class this characterization has been explored extensively in [34]. A somewhat related product-form characterization as by an invariance condition has also been provided in [25] under the condition that there is no blocking and that the service rates are strictly positive. Its result is included by the current one as a special case. More specifically, the most closely related results for processor-sharing mechanisms are those from [17] and [25]. In these references though the implicit but essential condition is assumed for the existence of a function q (see [25]) or Φ (in [17]) to be seen as the function $H(\mathbf{x})$ in Theorem 3.2.1. However, these are hard to find in general. The present setting, in contrast, *does* lead to a construction or check of this function by means of reversibility, as will be illustrated in Sections 3.3 and 3.4 for the models of our interest.

Examples

In the next two sections the theorem presented in this section is used to investigate product-form results for the following six examples for both parallel and tandem models. The parallel case is given Section 3.3, and the tandem case in Section 3.4.

(1) The proportional PS model,

(2) An unproportional PS model with full capacity to one queue,

(3) An α-unproportional PS model,

(4) A state-space reduction,

(5) A two-queue LPS model,

(6) A truncated two-queue LPS model.

The first model is well-known to possess a product form. However, it is included to illustrate Theorem 3.2.1. The results for the second and third model seem to be new in the literature; unbalanced sharing of the service capacity is captured in these examples. The results for the fourth model are known for the parallel model, but new for the tandem model; it illustrates the differences that appear between tandem and parallel routing mechanisms for truncation of the state space. The fifth model example was already introduced [6, 112], but the non-product form proof is new as is the product-form truncation in the tandem case. None of the examples did appear this detailed in literature, and therefore also contributes to the insights of product-form results.

3.3 Parallel model: Examples

In this section we apply Theorem 3.2.1 to show and prove the existence of product-form solutions for the parallel model with shared resources as described in Section 3.2.3. To this end, we write

$$\phi_i(\mathbf{x}) = \Phi(x_1 + x_2)s_i(\mathbf{x}), \text{ for } i = 1, 2, \tag{3.17}$$

where $\Phi(k) > 0$ represents the *total* service capacity of the shared resource when the total number of jobs $x_1 + x_2$ equals k, and where the sharing function $s_i(\mathbf{x})$ is a fraction of this capacity allocated to queue i ($i = 1,\ 2$). Note that $\phi_i(\mathbf{x})$ is uniquely defined by $\Phi(x_1 + x_2)$ and $s_i(\mathbf{x})$ up to a scaling constant and that in general $\Phi(\cdot)$ is not necessarily equal to 1, additionally note that $\sum_i s_i(\mathbf{x}) \leq 1$. Furthermore, for notational convenience, we define

$$P(\mathbf{x}) := \left[\prod_{k=1}^{x_1+x_2} \Phi(k)\right]^{-1}, \tag{3.18}$$

which we will from now on use in the remainder of the chapter. We now consider the examples presented in Section 3.2.3.

3.3.1 Proportional PS model

Consider the two-queue extension of the standard single-queue PS queue where the total capacity equals $\Phi(x_1 + x_2)$ and where the fraction of this capacity allocated to the queues equals

$$s_i(\mathbf{x}) := \frac{x_i}{x_1 + x_2}, \text{ for } i = 1,\ 2, \tag{3.19}$$

for $\mathbf{x} \in C$, with

$$C = \{\mathbf{x} \mid x_1, x_2 \geq 0\}. \tag{3.20}$$

Thus, for given state $\mathbf{x}$ queue i gets a fraction $s_i(\mathbf{x})$ of the capacity $\Phi(x_1 + x_2)$; in words, $s_i(\mathbf{x})$ represents the proportion of jobs that are at queue i. The admission policy is given by $b_i(\mathbf{x}) := 1$ for $i = 1,\ 2$ and all $\mathbf{x} \in C$, i.e., all arriving jobs are accepted for all $\mathbf{x} \in C$. Note that the classical PS-case occurs as a special case by taking $\Phi(k) = 1$ for all $k \geq 0$.

Result 3.3.1. *The proportional parallel* PS *model possesses a product-form solution of the form* (3.6)*, with*

$$H(\mathbf{x}) = \left[\prod_{i=1}^{2} \lambda_i^{x_i}\right] P(\mathbf{x}) \binom{x_1 + x_2}{x_1}, \textit{ for } \mathbf{x} \in C. \tag{3.21}$$

Proof. We first use Theorem 3.2.1 to prove the existence of the product form, and then, to prove that the product-form solution has the form (3.21). Note that the proof can also be constructed based on the balance equations, which is stated in Remark 3.3.2. However, we construct a proof based on Theorem 3.2.1, illustrating this theorem. Therefore it suffices to show that the reversibility condition (3.8), i.e., $\theta(p) = \theta(\bar{p})$, is satisfied for each path p. To this end, note that for the model under consideration, the transition rates are as follows using Equations (3.4) and (3.17).

For $\mathbf{x} \in C$,

$$\begin{aligned}
&\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1) = \lambda_1, \\
&\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2) = \lambda_2, \\
&\bar{q}(\mathbf{x}, \mathbf{x}-\mathbf{e}_1) = \frac{x_1}{x_1+x_2}\Phi(x_1+x_2), \\
&\bar{q}(\mathbf{x}, \mathbf{x}-\mathbf{e}_2) = \frac{x_2}{x_1+x_2}\Phi(x_1+x_2).
\end{aligned} \tag{3.22}$$

Based on these transition rates one may verify that the transition matrix of the adjoint Markov chain $\bar{Q}$ equals the transition matrix Q of the original Markov chain. Note that for this model it suffices to consider only two basic cycles, since all other cycles are constructed similarly. Thus, we only need to show that $\theta(p) = \theta(\bar{p})$ for the following two paths

$$\begin{aligned}
p &= \mathbf{x} \to \mathbf{x}+\mathbf{e}_1 \to \mathbf{x}+\mathbf{e}_1+\mathbf{e}_2 \to \mathbf{x}+\mathbf{e}_2 \to \mathbf{x}, \\
\bar{p} &= \mathbf{x} \to \mathbf{x}+\mathbf{e}_2 \to \mathbf{x}+\mathbf{e}_1+\mathbf{e}_2 \to \mathbf{x}+\mathbf{e}_1 \to \mathbf{x}.
\end{aligned} \tag{3.23}$$

To this end, substitution of Equations (3.22) into Equation (3.10) leads to

$$\begin{aligned}
\theta(p) =& \bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}+\mathbf{e}_1+\mathbf{e}_2) \\
&\times \bar{q}(\mathbf{x}+\mathbf{e}_1+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) \\
=& \lambda_1 \times \lambda_2 \times \frac{(x_1+1)}{x_1+x_2+2}\Phi(x_1+x_2+2) \\
&\times \frac{(x_2+1)}{x_1+x_2+1}\Phi(x_1+x_2+1),
\end{aligned}$$

and

$$\begin{aligned}
\theta(\bar{p}) =& \bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1+\mathbf{e}_2) \\
&\times \bar{q}(\mathbf{x}+\mathbf{e}_1+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) \\
=& \lambda_2 \times \lambda_1 \times \frac{(x_2+1)}{x_1+x_2+2}\Phi(x_1+x_2+2) \\
&\times \frac{(x_1+1)}{x_1+x_2+1}\Phi(x_1+x_2+1),
\end{aligned}$$

which immediately implies $\theta(p) = \theta(\bar{p})$. Hence, the reversibility condition (3.8) applies, and thus, there exists a product-form solution (3.6). Next, we show that the product-form solution has the form (3.21). To this end, we observe that using Equation (3.7) in Theorem 3.2.1 and the equations in (3.22) imply the following recursive relations for $\mathbf{x} \in C$

$$\begin{aligned} H(\mathbf{x})\lambda_1 &= H(\mathbf{x})\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1) \\ &= H(\mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) \\ &= H(\mathbf{x}+\mathbf{e}_1)\frac{(x_1+1)}{x_1+x_2+1}\Phi(x_1+x_2+1). \end{aligned} \tag{3.24}$$

Similarly by (3.7) and (3.22) we find that

$$\begin{aligned} H(\mathbf{x})\lambda_2 &= H(\mathbf{x})\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2) \\ &= H(\mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) \\ &= H(\mathbf{x}+\mathbf{e}_2)\frac{(x_2+1)}{x_1+x_2+1}\Phi(x_1+x_2+1), \end{aligned} \tag{3.25}$$

and

$$\begin{aligned} H(\mathbf{x})\frac{x_1}{x_1+x_2}\Phi(x_1+x_2) &= H(\mathbf{x})\bar{q}(\mathbf{x}, \mathbf{x}-\mathbf{e}_1) \\ &= H(\mathbf{x}-\mathbf{e}_1)\bar{q}(\mathbf{x}-\mathbf{e}_1, \mathbf{x}) \\ &= H(\mathbf{x}-\mathbf{e}_1)\lambda_1, \end{aligned} \tag{3.26}$$

and

$$\begin{aligned} H(\mathbf{x})\frac{x_2}{x_1+x_2}\Phi(x_1+x_2) &= H(\mathbf{x})\bar{q}(\mathbf{x}, \mathbf{x}-\mathbf{e}_2) \\ &= H(\mathbf{x}-\mathbf{e}_2)\bar{q}(\mathbf{x}-\mathbf{e}_2, \mathbf{x}) \\ &= H(\mathbf{x}-\mathbf{e}_2)\lambda_2. \end{aligned} \tag{3.27}$$

Note that Equation (3.24) equals Equation (3.26), since for $(x_1, x_2) = (0, 0)$ transition rates to states $(x_1 - 1, x_2) = (-1, 0)$ and $(x_1, x_2 - 1) = (0, -1)$ are zero, which forces that Equation (3.24) and Equation (3.26) are equivalent. Similarly, we conclude that Equation (3.25) equals Equation (3.27). Thus, the recursive relation can be rewritten as

$$\frac{H(\mathbf{x}-\mathbf{e}_1)}{H(\mathbf{x})} = \frac{1}{\lambda_1}\frac{x_1}{x_1+x_2}\Phi(x_1+x_2),\ x_1 > 0, \tag{3.28}$$

$$\frac{H(\mathbf{x}-\mathbf{e}_2)}{H(\mathbf{x})} = \frac{1}{\lambda_2}\frac{x_2}{x_1+x_2}\Phi(x_1+x_2), \ x_2 > 0. \tag{3.29}$$

Equation (3.21) can now be obtained by recursively solving (3.28) and (3.29), starting with $H(\mathbf{0}) := \Phi(0)$. □

Remark 3.3.2. (Alternative approaches) Instead of by the proof presented above, Result 3.3.1 can also be concluded:

1. Directly by substituting (3.7) in (3.6).
2. From [64], since the proportional PS queue is a symmetric queue, and stationary symmetric queues are reversible which is stated in Theorem 3.1 in [64].
3. From [25]. To this end, consider the system as a single processor. Let a class-r job have a service with respective parameters μ_r for class r. This has a one to one correspondence with the two-queue parallel model, since each class corresponds to a queue.
4. From [17] directly for a processor-sharing discipline and indirectly for arbitrary disciplines as in this reference it is implicitly assumed that each queue itself (also) has a PS-discipline. However, as the effective service rates at queue 1 and 2 are independent of the service discipline provided the services are assumed to be exponential, in the exponential case the product form can be concluded for arbitrary disciplines at each queue.

By this reference as well as by [25] it can also be concluded that the product form is insensitive with respect to the service-time distributions.

Remark 3.3.3. (Special PS-case and insensitivity) The standard type processor-sharing function, that assigns an equal (fair) share $1/(x_1+x_2)$ of the total capacity $\Phi(x_1+x_2)$ to each job in service, is included by assuming that each queue itself also has a PS-discipline; that, at both queues, all jobs present equally share a fraction $x_i/(x_1+x_2)$ of the total capacity. For this particular case, it can also be concluded directly from [17] or indirectly from [25] or [53, 54], that the product form (3.6) also applies to arbitrary non-exponential service requirements with means $1/\mu_i$ at queue i. This property is well-known in the literature as *insensitivity*.

3.3.2 Unproportional PS model with full capacity to one queue

A first example in which an unproportional processor sharing is effectuated is obtained by always allocating the full capacity to one queue, and to fairly share this capacity among all jobs at that queue. For this model, consider the state space

$$C = \{\mathbf{x} \mid x_1 \in \{x_2-1, x_2, x_2+1, x_2+2\}, \ \text{with } x_1, x_2 \geq 0\}, \tag{3.30}$$

and access blocking functions

$$\begin{aligned} b_1(\mathbf{x}) &= \mathbb{1}_{E_1}, \text{ with } E_1 := \{\mathbf{x} \in C : x_1 = x_2 \text{ or } x_1 = x_2 + 1\}, \\ b_2(\mathbf{x}) &= \mathbb{1}_{E_2}, \text{ with } E_2 := \{\mathbf{x} \in C : x_1 = x_2 \text{ or } x_1 = x_2 - 1\}, \end{aligned} \tag{3.31}$$

and sharing functions

$$\begin{aligned} s_1(\mathbf{x}) &= \mathbb{1}_{E_3}, \text{ with } E_3 := \{\mathbf{x} \in C : x_1 = x_2 + 1 \text{ or } x_1 = x_2 + 2\}, \\ s_2(\mathbf{x}) &= \mathbb{1}_{E_4}, \text{ with } E_4 := \{\mathbf{x} \in C : x_1 = x_2 \text{ or } x_1 = x_2 - 1\}. \end{aligned} \tag{3.32}$$

The access blocking function only allows arrivals to queue 1 if $x_1 = x_2$ or $x_1 = x_2+1$, and similarly, queue-2 arrivals are accepted only if $x_1 = x_2 - 1$ or $x_1 = x_2$. The sharing function forces to assign all capacity to queue 1 if $x_1 = x_2+1$ or $x_1 = x_2+2$, and to queue 2 if $x_1 = x_2 - 1$ or $x_1 = x_2$. In words, if $x_1 > x_2$ then queue 1 gets the full capacity, and queue 2 gets the full capacity otherwise. This model will be referred to as the unproportional parallel PS model. Using Equations (3.31) and (3.32) it is readily verified that the state space for this model is given by Equation (3.30). Figure 3.3 illustrates the non-zero transitions at the state space of this model.

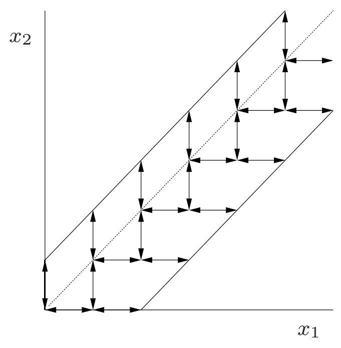


Figure 3.3: Parallel model: Transitions in the state space C for which the product form (3.6) applies with positive transition rates (in both directions) indicated by arrows (all other rates are equal to 0).

Result 3.3.4. *The unproportional parallel* PS *model possesses a product-form solution of the form* (3.6), *with*

$$H(\mathbf{x}) = \left[\prod_{i=1}^{2} \lambda_i^{x_i}\right] P(\mathbf{x}), \textit{ for } \mathbf{x} \in C, \tag{3.33}$$

where C is defined in (3.30).

Proof. First we show that the model possesses a product form by checking Equation (3.8) for all paths within the state space C, defined in (3.30). To this end, note that the transition rates for the adjoint Markov chain (which are again equal to the transition rates for the original Markov chain) are as follows

$$\begin{aligned}
\bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1) &= \lambda_1 b_1(\mathbf{x}),\\
\bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_2) &= \lambda_2 b_2(\mathbf{x}),\\
\bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_1) &= \Phi(x_1+x_2)s_1(\mathbf{x}),\\
\bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_2) &= \Phi(x_1+x_2)s_2(\mathbf{x}).
\end{aligned} \tag{3.34}$$

Note that we only need to verify Equation (3.8) for the following two basic cycles, since any cycle can be constructed similarly.

$$\begin{aligned}
p_1 &= \mathbf{x}\to\mathbf{x}+\mathbf{e}_1\to\mathbf{x},\\
p_2 &= \mathbf{x}\to\mathbf{x}+\mathbf{e}_2\to\mathbf{x}.
\end{aligned}$$

Substitution of (3.34) in (3.10) leads to the following two equations, for $\mathbf{x}\in C$,

$$\begin{aligned}
\theta(p_1) &= \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1,\mathbf{x}) = \lambda_1\times\Phi(x_1+x_2),\\
\theta(p_2) &= \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2,\mathbf{x}) = \lambda_2\times\Phi(x_1+x_2).
\end{aligned}$$

Next, notice that the paths in the opposite directions, denoted by $\bar{p}_1$ and $\bar{p}_2$, are equal to the paths p_1 and p_2, respectively. Hence, for $i=1,\ 2$ we have $\theta(p_i)=\theta(\bar{p}_i)$, so that the reversibility condition (3.8) is satisfied, which implies that the model has a product-form solution. Then, to show that (3.33) holds, note that arguments similar to those in Result 3.3.1 hold and that it is easily verified that $x_i>0$,

$$\frac{H(\mathbf{x}-\mathbf{e}_i)}{H(\mathbf{x})} = \frac{1}{\lambda_i}\Phi(x_1+x_2),\ \text{for } i=1,\ 2, \tag{3.35}$$

supplemented with the starting condition $H(\mathbf{0}) := \Phi(0)$ gives $H(\mathbf{x})$ in Equation (3.33). Thus the steady-state distribution has the product form (3.6), where $H(\mathbf{x})$ is given by Equation (3.33). This completes the proof of the result. □

3.3.3 α-Unproportional PS model

We consider another example of unproportional sharing of the capacity and show that unproportional and non-zero sharing functions over both queues might still retain the necessary invariance (3.11), or equivalently (3.8). Consider the complete state space

$$C = \{\mathbf{x} \mid x_1, x_2 \geq 0\}, \tag{3.36}$$

and a sharing function $s_i(\mathbf{x})$ in which a fraction of the capacity is assigned to queue 1, and a fraction of the capacity is assigned to queue 2, as follows, for $\mathbf{x}\in C$,

$$(s_1(\mathbf{x}),s_2(\mathbf{x})) := \begin{cases} (1-\alpha,\alpha), & \text{if } x_1>x_2,\\ (\alpha,1-\alpha), & \text{if } x_1<x_2,\\ (\alpha,\alpha), & \text{if } x_1=x_2, \end{cases} \tag{3.37}$$

for an arbitrary $0 < \alpha < 1/2$. We allow α to be smaller than a half, and thus the model is not work-conserving. The fraction of the total capacity $\Phi(x_1 + x_2)$ a queue receives, is dependent on the state space. The sharing function $s_i(\mathbf{x})$ partitions the state space in three regions, namely in the region where the number of jobs in the queue 1 is greater than the number of jobs present at queue 2 (i.e., $x_1 > x_2$), the region where the number of jobs at queue 1 is smaller than the number of jobs at queue 2 (i.e., $x_1 < x_2$), and the region where the number of jobs is equal in both queues (i.e., $x_1 = x_2$). We refer this model as the α-unproportional processor-sharing model.

Result 3.3.5. *A product-form solution applies for the α-unproportional processor-sharing model of the form* (3.6)*, with*

$$H(\mathbf{x}) = \left[\prod_{i=1}^{2} \lambda_i^{x_i}\right] P(\mathbf{x}) \left(\frac{\alpha}{1-\alpha}\right)^{\max(x_1,x_2)} \left(\frac{1}{\alpha}\right)^{x_1+x_2}, \text{ for } \mathbf{x} \in C. \tag{3.38}$$

Proof. To show that this model possesses a product-form solution we need to investigate condition (3.8) or equivalently (3.11) so that Theorem 3.2.1 applies. For this model it suffices to verify the condition for the following cycles

$$\begin{aligned} p &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x}, \\ \bar{p} &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x}. \end{aligned} \tag{3.39}$$

These cycles need to be considered for the following five scenarios: $x_1 = x_2$, $x_1 + 1 = x_2$, $x_1 - 1 = x_2$, $x_1 + 1 > x_2$, and $x_1 - 1 < x_2$, respectively. For these scenarios the transition rates differ, due to the specific sharing function defined in Equation (3.37). Consider the state space region $x_1 - 1 < x_2$, the products of the transition rates for the paths p and $\bar{p}$, as in Equation (3.39), equal

$$\begin{aligned} \theta(p) &= \bar{q}(\mathbf{x}, \mathbf{x} + \mathbf{e}_1)\bar{q}(\mathbf{x} + \mathbf{e}_1, \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2, \mathbf{x} + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_2, \mathbf{x}) \\ &= \alpha^2(1-\alpha)^2\Phi(x_1 + x_2 + 2)\Phi(x_1 + x_2 + 1) \times \lambda_1 \times \lambda_2, \\ \theta(\bar{p}) &= \bar{q}(\mathbf{x}, \mathbf{x} + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_2, \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2, \mathbf{x} + \mathbf{e}_1)\bar{q}(\mathbf{x} + \mathbf{e}_1, \mathbf{x}) \\ &= \alpha^2(1-\alpha)^2\Phi(x_1 + x_2 + 2)\Phi(x_1 + x_2 + 1) \times \lambda_1 \times \lambda_2. \end{aligned}$$

Similarly, for the other scenarios the products of the transition rates for the paths p and $\bar{p}$ are also verified. Thus Equation (3.8) is fulfilled since for all scenarios $\theta(p) = \theta(\bar{p})$. Next note that Equation (3.38) is obtained following the same lines as in the example given in Section 3.3.1 or equivalently by Equation (3.7). The result (3.38) then follows by substitution of Equations (3.4) and (3.37) in Equation (3.7), with $H(\mathbf{x})$ as defined in Equation (3.38) and proper scaling. This completes the proof. □

Remark 3.3.6. Note that for this example the queue with the highest workload receives more capacity than the other queue. When the queues have equal workload, both receive an equal share of the total capacity. But since α can be arbitrarily close to 0, not all capacity needs to be used if the workloads are equal. Thus, as a price to pay to satisfy the invariance condition (3.11) note that a capacity of α is lost when $x_1 = x_2$.

3.3.4 State space restriction

In general, state space restrictions of a model that possesses a product-form solution do not necessary possesses a product-form solution itself. However, it is known from [63, 64] that a model, which is reversible itself, possesses a product form at any state space C also possesses a product form at any coordinate convex state space, where coordinate convex is defined by

$$\mathbf{x} \in C \Rightarrow \mathbf{x} - \mathbf{e}_i \in C, \text{ for } i = 1,\ 2. \tag{3.40}$$

We give an example of a coordinate convex state space restriction for forward reference, since comparing a similar state space restriction for the parallel and the tandem model (see Section 3.4.4 below) leads to remarkable observations. To this end, consider in this example the service and blocking functions as given in Section 3.3.1. We restrict the state space of this model by elimination of all states $\mathbf{x}$ with $x_2 > d_2$ and with $x_1 + x_2 > d_1$, where $d_i > 0$ is a constant for $i = 1,\ 2$. This can be enforced by a proper choice of the blocking functions $b_i(\mathbf{x})$, for $i = 1,\ 2$. This will lead to the following coordinate convex state space

$$C = \{\mathbf{x} \mid 0 x_1 + x_2 \le d_1, x_2 \le d_2,\ x_i \ge 0, \text{for}\, i = 1,\ 2\}. \tag{3.41}$$

This state space restriction is presented in the left figure of Figure 3.4. We illustrate that the product-form solution indeed holds by verifying Equation (3.8) for the paths

$$\begin{aligned} p &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x}, \\ \bar{p} &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_1 + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x}. \end{aligned}$$

And, indeed $\theta(p) = \theta(\bar{p})$ holds, since for all $\mathbf{x} \in C$, for $i = 1,\ 2$, we have

$$\begin{aligned} \theta(p) &= \lambda_1 \times \lambda_2 \times \frac{x_1 + 1}{x_1 + x_2 + 2}\Phi(x_1 + x_2 + 2) \times \frac{x_2 + 1}{x_1 + x_2 + 1}\Phi(x_1 + x_2 + 1), \\ \theta(\bar{p}) &= \lambda_1 \times \lambda_2 \times \frac{x_2 + 1}{x_1 + x_2 + 2}\Phi(x_1 + x_2 + 2) \times \frac{x_1 + 1}{x_1 + x_2 + 1}\Phi(x_1 + x_2 + 1). \end{aligned}$$

3.3.5 Two-queue LPS model

Now we consider the two-queue extension of the limited processor-sharing (LPS) queue, which is recently studied in [59, 85, 126, 127, 128]. The LPS queue works as follows: Instead of taking all jobs immediately in service and sharing the capacity among all these jobs, we assume that $k_i(\mathbf{x})$ jobs receive service and that $k_i(\mathbf{x})$ is bounded by c_i. If there are more than c_i jobs in queue i these jobs have to wait until one of the $k_i(\mathbf{x})$ jobs leaves the queue. The two-queue extension of the LPS queue is defined by the following sharing function

$$\begin{aligned} s_1(\mathbf{x}) &= k_1(\mathbf{x})/(k_1(\mathbf{x}) + k_2(\mathbf{x})), \quad k_1(\mathbf{x}) = \min(x_1, c_1), \\ s_2(\mathbf{x}) &= k_2(\mathbf{x})/(k_1(\mathbf{x}) + k_2(\mathbf{x})), \quad k_2(\mathbf{x}) = \min(x_2, c_2). \end{aligned} \tag{3.42}$$

Note that each queue receives a fraction of the capacity based on the number of jobs in both queues, and not as in recently studied LPS models based on the number of jobs in only one queue. Let c_1 and/or c_2 be finite and let the state space be defined as all non-negative integer values for x_1 and x_2 which is

$$C = \{\mathbf{x} \mid x_i \geq 0,\ i = 1,\ 2\}. \tag{3.43}$$

This model is illustrated in Figure 3.1 where $c_i = 3$ for $i = 1,\ 2$, and where $x_1 = 4$ and $x_2 = 2$. Thus one job in the first queue is not in service, but is in the buffer, and remains in the buffer until one of three jobs in service leaves the queue.

Result 3.3.7. *The two-queue parallel* LPS *model does not allow a product-form solution.*

Proof. The proof is based on a counter-example, so that Equation (3.14) holds, and equivalent, Equation (3.8) or Equation (3.11) does not hold. For this, let $\Phi(k) = 1$ for all $k \geq 1$. Note that the routing is again reversible, and we verify if the products of the transition rates of the cycles satisfy Equation (3.14) such that the adjoint model is reversible. Consider the LPS model with $c_1 = 2$ and $c_2 = 3$. Based on verifying Equation (3.14) we construct the following paths p_1 and p_2

$$\begin{aligned}
p_1 &= (4,3) \to (4,2) \to (4,1) \to (3,1) \to (2,1) \to (1,1),\\
\bar{p}_1 &= (1,1) \to (2,1) \to (3,1) \to (4,1) \to (4,2) \to (4,3),\\
p_2 &= (4,3) \to (3,3) \to (2,3) \to (1,3) \to (1,2) \to (1,1),\\
\bar{p}_2 &= (1,1) \to (1,2) \to (1,3) \to (2,3) \to (3,3) \to (4,3).
\end{aligned}$$

Take $\lambda_1 = 1$ and $\lambda_2 = 1$. This brings us to the following values of $\Theta(p_i)$ as in (3.14), namely

$$\begin{aligned}
\Theta(p_1) &= \theta(p_1)/\theta(\bar{p}_1) = (2/5) \times (2/5) \times (3/4) \times (3/4) \times (2/3) = 3/50,\\
\Theta(p_2) &= \theta(p_1)/\theta(\bar{p}_1) = (3/5) \times (3/5) \times (2/4) \times (2/3) \times (2/3) = 4/50.
\end{aligned}$$

Thus note that $\Theta(p_1) \neq \Theta(p_2)$. Hence, the necessary reversibility condition (3.14) is violated, and thus no product form exists. □

Remark 3.3.8. Most remarkably, a single LPS queue obviously has a product form, but the structure of the network, in which the sharing depends on the state of the entire model, does not. Because of the limited number of jobs in service, the order of arrival of the jobs becomes important, since a job not in service can not be exchanged for a job in service, due to the fact that the service speed does not only depend on that queue itself, but also on the other queue. This dependency of queues makes the order of arrival of the jobs important, which results in violation of the reversibility conditions.

3.3.6 Truncated two-queue LPS model

A way to retain the product form for the two-queue LPS parallel model is to restrict the state space artificially such that there can never be more than c_i jobs in queue i for $i = 1,\ 2$. The following access blocking functions give a proper state space restriction with respect to the existence of a product-form solution

$$b_1(\mathbf{x}) \quad = \quad 0,\ \text{if } x_1 \geq c_1, \tag{3.44}$$

$$b_2(\mathbf{x}) \quad = \quad 0,\ \text{if } x_2 \geq c_2. \tag{3.45}$$

These access blocking functions limit the state space to

$$C = \{\mathbf{x} \mid 0 \leq x_i \leq c_i,\ i = 1,\ 2\}. \tag{3.46}$$

Thus, if a job arrives at a queue i while there are already c_i jobs present, then this job is blocked. This model is referred to as the truncated two-queue parallel LPS model.

Result 3.3.9. *The truncated two-queue parallel* LPS *model possesses a product form of the form* (3.6) *with* $H(\mathbf{x})$ *as in Equation* (3.21).

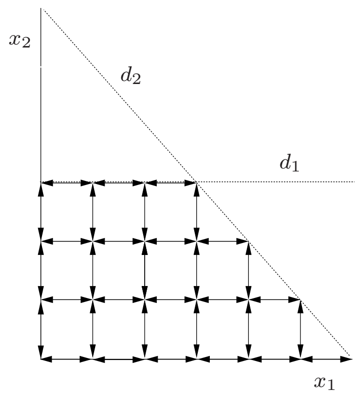
Proof. We again rely on Theorem 3.2.1 to prove the existence of the product form and its specific form (3.21). Observe that $s_i(\mathbf{x})$ is defined in the same way as in the natural processor-sharing form (3.19) for the state space C in Equation (3.46) and that also on the boundaries the routing remains reversible and transitions are similarly defined as in Section 3.3.4. This leads immediately to the conclusion that the product form (3.6) applies, which can be verified analogously to the proof in Section 3.3.4. The form of the product, (3.21), follows due to the previous observation, following the lines in Section 3.3.4. This completes the proof. □

The state space restriction of Section 3.3.4 in Equation (3.41) and the state space truncation (3.46) of the example in this section are illustrated by Figure 3.4.

Remark 3.3.10. Results for showing that a product form *cannot* hold appear to be rare in the literature. From [25] such results can be deduced if a proper transformation is made, however in the present setting it follows directly. The observation that a model does *not* have a product form is very important, and can lead to adjustments of the model such that a product form still applies. Note that these adjusted models can be used to develop approximations for the steady-state distribution of non-product form models and can be used to derive error bounds (which falls beyond the scope of the present chapter).

3.4 Tandem model: Examples

In this section we apply Theorem 3.2.1 to the tandem model, by using similar examples as for the parallel model. In this model the arrivals at the second queue



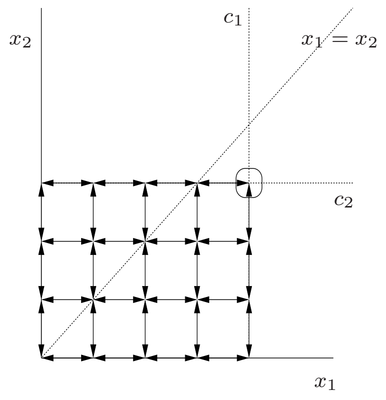


Figure 3.4: Parallel model: The left figure illustrates state space (3.41) and the right figure illustrates state space (3.46). For both truncations the product form (3.6) applies. In the right figure the state (c_1, c_2) is marked for further reference in Section 3.4.4.

are fed by departures from the first queue as described in Section 3.2.2. We show the similarities and differences for the parallel and tandem models with respect to product-form solutions.

3.4.1 Proportional PS model

Consider the following two-queue tandem model with the capacity equal to $\Phi(x_1 + x_2)$, and a fraction $s_i(\mathbf{x})$ of this capacity is allocated to the queues as given in Equation (3.19), and thus the capacity is shared proportionally among the number of jobs in each of the queues. Let the state space C be as in Equation (3.20). We refer to this model as the proportional tandem PS model.

Result 3.4.1. *The proportional tandem* PS *model possesses a product-form solution of the form* (3.6)*, with*

$$H(\mathbf{x}) = \lambda^{x_1+x_2} P(\mathbf{x}) \binom{x_1 + x_2}{x_1}, \quad \textit{for } \mathbf{x} \in C. \tag{3.47}$$

Proof. To prove this result we show that Theorem 3.2.1 applies, by verifying Equation (3.8). Therefore we construct the adjoint Markov chain. The transition rates of the original Markov chain are as follows

$$\begin{aligned} q(\mathbf{x}, \mathbf{x} + \mathbf{e}_1) &= \lambda, \\ q(\mathbf{x}, \mathbf{x} - \mathbf{e}_1 + \mathbf{e}_2) &= \frac{x_1}{x_1 + x_2} \Phi(x_1 + x_2), \\ q(\mathbf{x}, \mathbf{x} - \mathbf{e}_2) &= \frac{x_2}{x_1 + x_2} \Phi(x_1 + x_2). \end{aligned} \tag{3.48}$$

Note that the routing of this model is not reversible, and we construct the adjoint Markov chain transition rates according to Equation (3.5), which results in the following rates

$$\begin{aligned}
\bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1) &= \lambda,\\
\bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_1+\mathbf{e}_2) &= \frac{x_1}{x_1+x_2}\Phi(x_1+x_2),\\
\bar{q}(\mathbf{x},\mathbf{x}-\mathbf{e}_2) &= \frac{x_2}{x_1+x_2}\Phi(x_1+x_2),\\
\bar{q}(\mathbf{x}+\mathbf{e}_1,\mathbf{x}) &= \frac{x_1}{x_1+x_2}\Phi(x_1+x_2),\\
\bar{q}(\mathbf{x}-\mathbf{e}_1+\mathbf{e}_2,\mathbf{x}) &= \frac{x_2}{x_1+x_2}\Phi(x_1+x_2),\\
\bar{q}(\mathbf{x}-\mathbf{e}_2,\mathbf{x}) &= \lambda.
\end{aligned} \tag{3.49}$$

Similar to the parallel model, for the tandem model any cycle can be built from just two basic cycles, and therefore, it suffices to consider only the following two basic cycles

$$\begin{aligned}
p_1 &= \mathbf{x}\to\mathbf{x}+\mathbf{e}_1\to\mathbf{x}+\mathbf{e}_2\to\mathbf{x},\\
p_2 &= \mathbf{x}\to\mathbf{x}+\mathbf{e}_2\to\mathbf{x}+\mathbf{e}_1\to\mathbf{x}.
\end{aligned} \tag{3.50}$$

Substituting the expressions in Equation (3.49) in Equation (3.10) we obtain

$$\begin{aligned}
\theta(p_1) &= \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1,\mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2,\mathbf{x})\\
&= \lambda\times\frac{(x_1+1)}{x_1+x_2+1}\Phi(x_1+x_2+1)\times\frac{(x_2+1)}{x_1+x_2+1}\Phi(x_1+x_2+1),\\
\theta(\bar{p}_1) &= \bar{q}(\mathbf{x},\mathbf{x}+\mathbf{e}_2)\bar{q}(\mathbf{x}+\mathbf{e}_2,\mathbf{x}+\mathbf{e}_1)\bar{q}(\mathbf{x}+\mathbf{e}_1,\mathbf{x})\\
&= \lambda\times\frac{(x_2+1)}{x_1+x_2+1}\Phi(x_1+x_2+1)\times\frac{(x_1+1)}{x_1+x_2+1}\Phi(x_1+x_2+1),
\end{aligned}$$

and verifying Equation (3.8), indeed $\theta(p_1)$ equals $\theta(\bar{p}_1)$. Using similar arguments we can show that $\theta(p_2)=\theta(\bar{p}_2)$. Thus, $\theta(p_i)=\theta(\bar{p}_i)$ for $i=1,\ 2$, and since from these two cycles any other cycle can be constructed, the reversibility applies (i.e., Equation (3.8) holds) and there exists a product-form solution (3.6). Next, we prove that $H(\mathbf{x})$ has the form (3.47). This can be obtained by solving Equation (3.7) recursively starting with $H(\mathbf{0})=\Phi(0)$, which leads to Equation (3.28) and Equation (3.29), where λ_2 is replaced by λ. Next, since the solution of the recursive scheme also satisfies the third recursive relation

$$H(\mathbf{x}+\mathbf{e}_1)\frac{(x_1+1)}{x_1+x_2+1}\Phi(x_1+x_2+1) = H(\mathbf{x}+\mathbf{e}_2)\frac{(x_2+1)}{x_1+x_2+1}\Phi(x_1+x_2+1),$$

the local balance equations are satisfied. This completes the proof. $\square$

Remark 3.4.2. The model can be seen as a single processor-sharing model with 2 classes of jobs where after completion of service of a class-1 job it becomes a class-2 job, see [25, 64]. However, it illustrates how to apply the theorem with the use of the adjoint Markov chain which is different from the original Markov chain due to the non-reversible routing of the original chain.

Remark 3.4.3. Note that the function $H(\mathbf{x})$ differs only in the routing part from the form given in Equation (3.21), due to the fact that λ feeds both, queue 1, and queue 2 after completion of service at queue 1. Contrary, for the parallel model queue 2 is fed by its own arrival process with rate λ_2. The service part in $H(\mathbf{x})$, based on $s_i(\mathbf{x})$, is similar for the parallel and tandem model which was expected since $s_i(\mathbf{x})$ is equally defined, for $i = 1,\ 2$.

Remark 3.4.4. (Expression for the total population) In the particular proportional case we can also obtain a simple standard geometric-type expression for the steady-state distribution $\pi(\nu)$ of the total number of jobs $\nu = x_1 + x_2$. To this end, by (3.6) and (3.47) we obtain

$$\begin{aligned}
\pi(\nu) &= c \sum_{x_1,x_2:x_1+x_2=\nu} \lambda^{x_1+x_2} \binom{x_1+x_2}{x_1} \left[\prod_{k=1}^{\nu} \Phi(k)\right]^{-1} \left(\frac{1}{\mu_1}\right)^{x_1} \left(\frac{1}{\mu_2}\right)^{x_2} \\
&= c\lambda^{\nu} \left[\prod_{k=1}^{\nu} \Phi(k)\right]^{-1} \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{\nu} \\
&= c(\lambda\tau)^{\nu} \left[\prod_{k=1}^{\nu} \Phi(k)\right]^{-1},
\end{aligned}$$

with $\tau = \frac{1}{\mu_1} + \frac{1}{\mu_2}$ the total mean service time. For the parallel model from Section 3.3.1, we similarly find by (3.6) and (3.21)

$$\pi(\nu) = c(\lambda\tau)^{\nu} \left[\prod_{k=1}^{\nu} \Phi(k)\right]^{-1}, \tag{3.51}$$

with

$$\begin{aligned}
\tau &= \frac{\lambda_1}{\lambda_1+\lambda_2}\frac{1}{\mu_1} + \frac{\lambda_2}{\lambda_1+\lambda_2}\frac{1}{\mu_2}, \\
\lambda &= \lambda_1 + \lambda_2.
\end{aligned}$$

Hence, for both the parallel and the tandem model under proportional sharing and c a normalizing constant we obtain

$$\pi(\nu) = c\rho^{\nu} \left[\prod_{k=1}^{\nu} \Phi(k)\right]^{-1}, \tag{3.52}$$

with ρ the mean workload with $\rho = \lambda(\beta_1 + \beta_2)$ for the tandem model, and $\rho = \lambda_1\beta_1 + \lambda_2\beta_2$ for the parallel model and with $\beta_i = {\mu_i}^{-1}$.

Remark 3.4.5. Note that Equation (3.52) can in fact be seen as a simple insensitivity result, i.e., the expression does not depend on the routing mechanism but only on the traffic intensity ρ. As such this insensitivity result is in line (though somewhat different) with more standard insensitivity results for processor-sharing systems, e.g., [17, 25]. Note, however, that the (insensitivity Equation (3.52) also applies without assuming a strict processor-sharing discipline, i.e., in which the service at a queue is equally spread over all jobs. We know that such an insensitivity result for the examples in Sections 3.4.2 to 3.4.6 can not be concluded since Equation (3.52) essentially uses the multinomial coefficient. For these examples this coefficient does not exist.

3.4.2 Unproportional PS model with full capacity to one queue

As for the parallel model, for the tandem model we also continue with unproportional processor-sharing examples. First we consider the example where the full capacity is always allocated to one queue by setting the capacity of the other queue at value 0, and in the next section we consider the α-unproportional model. For the model considered in this section, the model where the full capacity is assigned to the queue with the highest workload, recall the sharing function $s_i(\mathbf{x})$ as in Equation (3.32) and let the blocking function $b_1(\mathbf{x})$ be equal to the blocking function for queue 1 as given in Equation (3.31). This model is referred to as the unproportional tandem PS model. Because of the sharing and blocking functions the state space C is defined as in Equation (3.30). Note that blocking cannot occur at the second queue, however, we obtain the same state space as for the parallel model. In the left figure of Figure 3.5 this state space is illustrated.

Result 3.4.6. *The unproportional tandem* PS *model possesses a product-form solution of the form* (3.6)*, with*

$$H(\mathbf{x}) = \lambda^{x_1+x_2} P(\mathbf{x}), \text{ for } \mathbf{x} \in C. \tag{3.53}$$

Proof. We construct the adjoint Markov chain according to Equation (3.5) so that we can verify Equation (3.8) and rely on Theorem 3.2.1. The transition rates of the adjoint Markov chain are as follows

$$\begin{aligned}
\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1) &= \lambda, \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}+\mathbf{e}_2) &= \Phi(x_1+x_2), \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) &= \Phi(x_1+x_2), \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) &= \Phi(x_1+x_2), \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1) &= \Phi(x_1+x_2), \\
\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2) &= \lambda,
\end{aligned} \tag{3.54}$$

where the transition rates only exist for given C as stated in Equation (3.30). Along the lines of the previous proof, we again need to verify for only two cycles that Equation (3.8) is satisfied, because the other cycles can be constructed with these cycles. Consider the cycles

$$\begin{aligned} p &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x}, \\ \bar{p} &= \mathbf{x} \to \mathbf{x} + \mathbf{e}_2 \to \mathbf{x} + \mathbf{e}_1 \to \mathbf{x}. \end{aligned}$$

For these cycles we use Equation (3.54) in Equation (3.10) which results in the following

$$\begin{aligned} \theta(p) &= \bar{q}(\mathbf{x}, \mathbf{x} + \mathbf{e}_1)\bar{q}(\mathbf{x} + \mathbf{e}_1, \mathbf{x} + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_2, \mathbf{x}), \\ &= \lambda \times \Phi(x_1 + x_2 + 1) \times \Phi(x_1 + x_2 + 1), \\ \theta(\bar{p}) &= \bar{q}(\mathbf{x}, \mathbf{x} + \mathbf{e}_2)\bar{q}(\mathbf{x} + \mathbf{e}_2, \mathbf{x} + \mathbf{e}_1)\bar{q}(\mathbf{x} + \mathbf{e}_1, \mathbf{x}), \\ &= \lambda \times \Phi(x_1 + x_2 + 1) \times \Phi(x_1 + x_2 + 1). \end{aligned}$$

And indeed $\theta(p) = \theta(\bar{p})$ for these two cycles and thus for all cycles p in C. Notice that Equation (3.53) can be obtained recursively solving Equation (3.7), starting with $H(\mathbf{0}) = \Phi(0)$, which results in Equation (3.35) with λ_2 replaced by λ. This recursion leads to the form given in Equation (3.53). This completes the proof. □

The left figure in Figure 3.5 illustrates the state space restrictions as in Equation (3.30), however, also other state-space restrictions for which (3.53) applies are possible, an example is illustrated in the right figure. Note the triangular structure of the cycles.

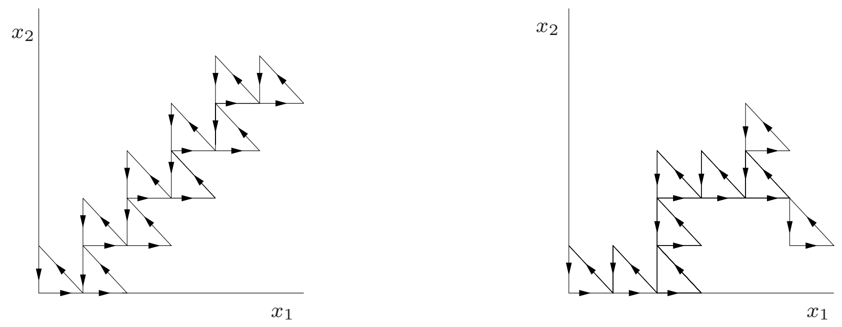


Figure 3.5: Tandem model: Figures for state space C for which the product form (3.6) applies with positive transition rates indicated by an arrow (all other rates are equal to 0).

3.4.3 α-Unproportional PS model

Now, we consider again unproportional and non-zero sharing functions $s_i(\mathbf{x})$ over both queues, as in Equation (3.37) on the state space C as defined by Equation (3.20). This model is called the α-unproportional tandem PS model. This model is presented in [36], and for a special value of α some results are presented. However, in this section we explain explicitly how to obtain the product form and compare this with the parallel version of the model. To this end, we observe that the α-unproportional tandem model still retains the necessary invariance (3.11) or equivalently (3.8), for example, with arbitrary $0 < \alpha < 1/2$. This model is illustrated in Figure 3.6, wherein only a few cycles are presented, representing the different rates depending on the state (x_1, x_2). Since there are no limitations on the state space, the complete state space is filled with these triangular structured transition rates.

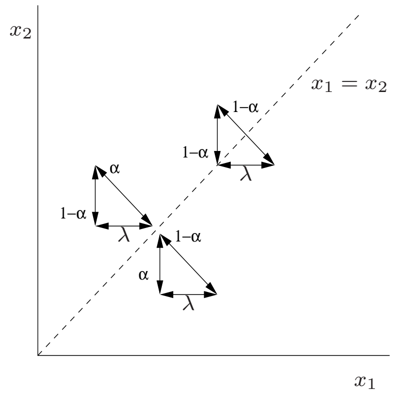


Figure 3.6: Tandem model: The α-unproportional PS model. In this figure in each of the regions ($x_1 < x_2$, $x_1 = x_2$, and $x_1 > x_2$) the transition rates of a cycle are given.

Result 3.4.7. *The α-unproportional tandem* PS *model possesses a product-form solution with the fraction of capacity allocated according to Equation* (3.37) *of the form* (3.6), *with*

$$H(\mathbf{x}) = \lambda^{x_1+x_2} P(\mathbf{x}) \left(\frac{\alpha}{1-\alpha}\right)^{\max(x_1,x_2)} \left(\frac{1}{\alpha}\right)^{x_1+x_2}, \text{ for } \mathbf{x} \in C. \tag{3.55}$$

Proof. For the proof we follow similar lines as the proof given in Section 3.3.3. To this end, we construct the adjoint Markov chain according to Equation (3.5) to verify condition (3.8). The adjoint transition rates become

$$\begin{array}{lcll}
\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1) & = & \lambda, & \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) & = & (1-\alpha), & \text{if } x_1+1 < x_2, \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) & = & \alpha, & \text{if } x_1+1 \geq x_2, \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) & = & \alpha, & \text{if } x_2+1 \leq x_1, \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) & = & (1-\alpha), & \text{if } x_2+1 > x_1, \\
\bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2) & = & \lambda, & \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}+\mathbf{e}_2) & = & (1-\alpha), & \text{if } x_1+1 > x_2, \\
\bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}+\mathbf{e}_2) & = & \alpha, & \text{if } x_1+1 \leq x_2, \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1) & = & \alpha, & \text{if } x_2+1 \leq x_1, \\
\bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1) & = & (1-\alpha), & \text{if } x_2+1 > x_1.
\end{array} \tag{3.56}$$

The adjoint transition rates differ, depending on x_1 and x_2. We only verify Equation (3.8) for the three cycles illustrated in Figure 3.6, noting that using these cycles, all cycles in the state space C can be constructed. Let

$$\begin{array}{lcl}
p & = & \mathbf{x} \to \mathbf{x}+\mathbf{e}_1 \to \mathbf{x}+\mathbf{e}_2 \to \mathbf{x}, \\
\bar{p} & = & \mathbf{x} \to \mathbf{x}+\mathbf{e}_2 \to \mathbf{x}+\mathbf{e}_1 \to \mathbf{x}.
\end{array}$$

The product of the transition rates for these paths are as follows. For $x_1 > x_2$,

$$\begin{array}{lcl}
\theta(p) & = & \lambda\alpha\Phi(x_1+x_2+1)(1-\alpha)\Phi(x_1+x_2+1), \\
\theta(\bar{p}) & = & \lambda(1-\alpha)\Phi(x_1+x_2+1)\alpha\Phi(x_1+x_2+1),
\end{array}$$

and for $x_1 < x_2$,

$$\begin{array}{lcl}
\theta(p) & = & \lambda(1-\alpha)\Phi(x_1+x_2+1)\alpha\Phi(x_1+x_2+1), \\
\theta(\bar{p}) & = & \lambda\alpha\Phi(x_1+x_2+1)(1-\alpha)\Phi(x_1+x_2+1),
\end{array}$$

and for $x_1 = x_2$,

$$\begin{array}{lcl}
\theta(p) & = & \lambda \times (1-\alpha)\Phi(x_1+x_2+1) \times (1-\alpha)\Phi(x_1+x_2+1), \\
\theta(\bar{p}) & = & \lambda \times (1-\alpha)\Phi(x_1+x_2+1) \times (1-\alpha)\Phi(x_1+x_2+1).
\end{array}$$

And thus indeed $\theta(p) = \theta(\bar{p})$ for all paths in C. This leads to the conclusion that for this model the product-form solution exists and is given with $H(\mathbf{x})$ as in Equation (3.38). Since the state space equals the state space of the parallel model, and the model has the same sharing function, we immediately conclude that $H(\mathbf{x})$ has the same form as the parallel model up to the routing part, as already can be seen in the proof given in Section 3.4.1. Thus the proof is completed. □

Remark 3.4.8. Note that for this model less cycles need to be checked than for the parallel model, due to the routing structure. But the results are equivalent, both examples lead to a product-form solution.

3.4.4 State space restriction

As we have seen in previous sections for the tandem model, the tandem model and the parallel model have many similarities with respect to the obtained product form for the same sharing functions and blocking functions. In the following examples of the tandem model we observe that differences occur. Consider the sharing function as in Equation (3.19) and state space C as given by Equation (3.41). For the tandem model, with proportional sharing of the capacity, the sharing function needs to be supplemented with

$$s_1(\mathbf{x}) \quad = \quad 0, \text{ for } x_1 + x_2 = d_2. \tag{3.57}$$

In the left figure of Figure 3.8 the transitions for this particular proportional sharing model in the state space C defined in (3.41) are illustrated.

Result 3.4.9. *The standard* PS *model has a product form*

$$\pi(\mathbf{x}) = c\lambda^{x_1+x_2} P(\mathbf{x}) \binom{x_1 + x_2}{x_1}, \tag{3.58}$$

for state space C as given in (3.41)*, supplemented with the sharing function in Equation* (3.57)*.*

Proof. For the proof of this result we refer to Section 3.4.1. We only have to verify if Equation (3.8) is satisfied for the cycles in the admissible state space C (3.41). Note that we block service at the first queue for the states $x_1 = d_2 - x_2$ to obtain the local balance (which can be verified by substitution of the transition rates (3.49) of the adjoint Markov chain in the Kolmogorov equations (3.3)). We now conclude similarly to the example in Section 3.4.1 that the product-form solution applies with the same $H(\mathbf{x})$, wherein $H(\mathbf{x})$ can be obtained following the lines of the proof of Result 3.4.1. □

Remark 3.4.10. Note that these results cannot be concluded from product-form results by simply restricting the state space under reversibility conditions such as in [91], since transition rates in the coordinate convex state space need to be changed such that reversibility of the adjoint Markov chain remains to hold.

Remark 3.4.11. Most remarkable is the following: to obtain a product form for the tandem model similarly to the parallel model different sharing functions are necessary such that the queue balance equations are satisfied. For the tandem model additionally the service from the first queue needs to be blocked in the state space, in the case $x_1 = d_2 - x_2$, which results in the function being $H(\mathbf{x})$ equivalent to the parallel model up to the routing part, the part containing the λ's.

3.4.5 Two-queue LPS model

Now, we consider the tandem LPS model, with sharing function $s_i(\mathbf{x})$ as in Equation (3.42). This function shares the capacity similar to the model in the previous section in the inner region of the state space C, and differs as soon as the boundaries are reached, namely if $x_1 = c_1$ or $x_2 = c_2$. We consider the state space

$$C = \{\mathbf{x} \mid x_1, x_2 \geq 0\}. \tag{3.59}$$

Result 3.4.12. *A product form solution does not exist for the two-queue tandem* LPS *model.*

Proof. This can be proved by verification of Equation (3.8). To this end, let $\Phi(x_1 + x_2) = 1$ and let $\lambda = 1$. The transitions of the adjoint Markov chain are shown in Figure 3.7 and, more precisely, as follows

$$\begin{array}{lcl} \bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_1) & = & \bar{q}(\mathbf{x}, \mathbf{x}+\mathbf{e}_2) = 1, \\ \bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}+\mathbf{e}_2) & = & \bar{q}(\mathbf{x}+\mathbf{e}_1, \mathbf{x}) = s_1(\mathbf{x}+\mathbf{e}_1), \\ \bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}+\mathbf{e}_1) & = & \bar{q}(\mathbf{x}+\mathbf{e}_2, \mathbf{x}) = s_2(\mathbf{x}+\mathbf{e}_2). \end{array} \tag{3.60}$$

Using a counter-example, we show that Equation (3.8) is violated. Consider $c_1 = 2$ and $c_2 = \infty$ as also presented in Figure 3.7 and consider the following cycles

$$\begin{array}{lcl} p & = & (1,1) \to (2,1) \to (3,1) \to (3,2) \to (2,2) \to (1,2) \to (1,1), \\ \bar{p} & = & (1,1) \to (1,2) \to (2,2) \to (3,2) \to (3,1) \to (2,1) \to (1,1), \end{array}$$

and the products of these paths according to Equation (3.10) equal

$$\begin{array}{lcl} \theta(p) & = & 1 \times 1 \times 1 \times (1/2) \times (1/2) \times (2/3) = 1/6, \\ \theta(\bar{p}) & = & 1 \times 1 \times 1 \times (1/2) \times (2/3) \times (2/3) = 2/9, \end{array}$$

and indeed $\theta(p) \neq \theta(\bar{p})$. Similarly, for any c_1 and c_2, with at least one of these less than infinity, it can be shown that Equation (3.8) is violated. As a consequence, condition (3.8) and thus also the necessary reversibility condition is violated so that the product form (3.6) fails. Similar (counter) examples can be given for any finite c_1 and/or c_2. □

3.4.6 Truncated two-queue LPS model

In line with Section 3.4.4 and as an extension to queue interdependent processor-sharing services of a product-form modification result in [34] for independent services, a way to retain the product form for the case where $k_i(\mathbf{x})$ (the number of jobs in service at queue i) is limited, is to restrict the state space such that there can be no more than c_1 jobs at queue 1 and c_2 at queue 2. This idea was already introduced for the parallel model.

However, to satisfy the necessary reversibility for the adjoint Markov chain additional boundary conditions are required, namely

$$\begin{array}{lcl} s_1(\mathbf{x}) & = & 0, \text{ if } x_2 = c_2, \\ s_2(\mathbf{x}) & = & 0, \text{ if } x_1 = c_1, \\ b_1(\mathbf{x}) & = & 0, \text{ if } x_1 = c_1 \text{ or } x_2 = c_2. \end{array} \tag{3.61}$$

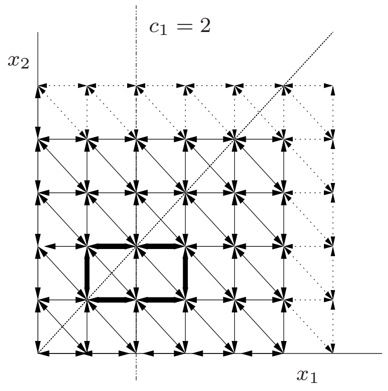


Figure 3.7: Tandem model: The adjoint transition rates for the LPS model.

These additional boundary conditions limit the state space to

$$C = \{\mathbf{x} \mid 0 \leq x_i \leq c_i, \quad i = 1,\ 2\}. \tag{3.62}$$

Result 3.4.13. *The truncated two-queue tandem* LPS *model possesses a product form* (3.6)*, with* $H(\mathbf{x})$ *as in* (3.47)*.*

Proof. To prove that the model possesses a product form we rely on the proof in Section 3.4.1. The model, as defined above, implies reversible routing of the adjoint Markov chain, which can be easily verified by checking the queue balance Equations (3.3). Next verifying Equation (3.8) leads to the same products as in Section 3.4.1, and similarly to Section 3.4.1 we obtain $H(\mathbf{x})$. Thus we conclude immediately that the product form exists with $H(\mathbf{x})$ as by (3.47). □

Remark 3.4.14. In this example we adjusted the sharing function to satisfy the queue balance equations. This adjustment leads to a smaller state space than the state space of the parallel model, since the upper-right corner (c_1, c_2) can not be reached, because this upper-right corner will ruin the reversibility of the adjoint Markov chain. Thus, the product form has the same form as the parallel model up to the routing part, but the state space differs to let the model possess a product-form solution.

3.5 Discussion

The results presented in Sections 3.3 and 3.4 lead to a number of remarkable observations, which will be discussed in more detail below. First we observe that the product-form results for the parallel and the tandem model have the same form for some examples, up to the part containing the arrival rates; this part contains λ for the tandem model and λ_1 and λ_2 for the parallel model. This observation

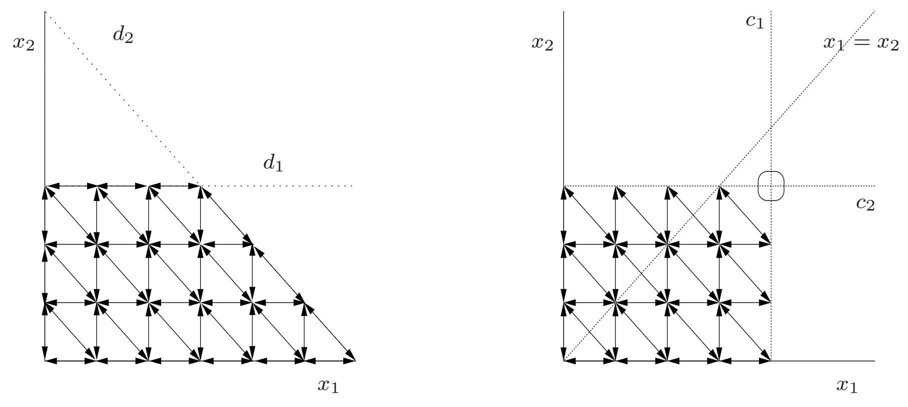


Figure 3.8: Tandem model: The left figure illustrates state space (3.41), and the right figure illustrates state space (3.62). For both truncations a product form (3.6) applies.

can be explained by the fact that the input rate to the second queue is λ_2 in the parallel model, and λ in the tandem model. For equal sharing functions and the common state space C, defined in Equation (3.20) this is the case. However, the product forms do not have the same structure if the state space is truncated, due to blocking of arrivals or stopping of services, such as in the examples in Sections 3.3.4, 3.4.4, and 3.3.6, 3.4.6. For some examples similar functions of $H(\mathbf{x})$ are obtained, due to the proper choice of the state space and stopping some transitions such that the reversibility, and the queue balance equations are satisfied. Thus, although the models are fundamentally different, the function $H(\mathbf{x})$ is the same (except the routing part) for both models.

Second, it is remarkable that the two-queue version of the LPS model does not lead to a product form, neither the tandem model nor for the parallel model. However, if the service discipline is independent of the number of jobs in each queue a simple product-form solution applies and also when the capacity is evenly shared among all jobs a product form exists. However, if the number of jobs that simultaneously receive service is bounded, the product-form structure is ruined. We suspect that the product form of a two-queue version of the LPS model is violated due to the effect of queueing, which does not only depend on the queue where the job is served, but also on the other queue.

Third, to verify the reversibility condition, we rely on the adjoint Markov chain. It is interesting to observe that this is necessary for the tandem model, but not for the parallel model. To this end, note that the transitions for the tandem model are not reversible, whereas the transition rates of the parallel model are reversible. This illustrates the additional value of defining the adjoint Markov chain, since many model instances fit in the framework presented in Theorem 3.2.1 to prove that a model does or does not possesses a product-form solution.

3.6 Conclusion and topics for further research

In this chapter we extended the product-form results of [34] to a general setting wherein blocking is allowed, as well as state-dependent services as fully stopped services. We present an approach for showing whether a model possesses a product form or not, unifying the tandem and parallel model. Illustrative and new examples are presented, in which remarkable results are shown.

The results lead to a number of directions for further research. First, in this chapter we focused on networks with two queues, which led to the analysis of two-dimensional state spaces. An interesting area for further research is to investigate to what extent the results can be generalized to models with an arbitrary number of queues. We suspect that this type of generalizations is possible under assumptions about the symmetry of the capacity assignment function $s_i(\mathbf{x})$. For asymmetric capacity assignment functions (see for example (17)), additional assumptions are likely to be needed to obtain product-form results. Derivation of this type of generalizations is a challenging area for further research.

Second, the results form an excellent basis for the development of simple yet accurate approximations for the mean sojourn times in case there is no product form. In this context, we may also derive error bounds for these approximations, based on the value-function techniques for related product-form networks [35].

Third, the results provide possibilities for optimization, both for models with and without product-form solutions. We may be able to derive monotonicity and convexity properties of mean sojourn times with respect to the limitations on the number of jobs in service, and with respect to the limitations on the state space. In this context, encouraging monotonicity results have been obtained for the single-queue case in [85]. Extensions of these results to the more general setting of the present chapter is an interesting area for follow-up research. Another area of interest is the development of efficient strategies for the dynamic assignment of capacities to the queues. To this end, we may study the performance of a control scheme in a Markov decision framework, and consider multi-modularity properties of value functions. Initial results presented in [113] show that significant performance gains can be obtained by these dynamic schemes compared to state-independent schemes.

CHAPTER 4

MONOTONICITY PROPERTIES

In this chapter we study monotonicity properties of a processor-sharing queue with a limited number of servers and an infinite buffer. The occupied servers share an underlying resource. We prove that for service times with a decreasing failure rate, the queue length is stochastically decreasing in the number of servers, and that for service times with an increasing failure rate, the queue length is stochastically increasing in the number of servers. We show that the result also holds for the limited foreground-background queue. The queue-length distributions and their corresponding asymptotic decay rates are compared to these in other queueing models with and without restrictions on the number of servers. This chapter is based on [85].

4.1 Introduction

In this chapter we study monotonicity properties of a multiserver queue in which the active servers share a common resource in a processing-sharing fashion; this model is commonly referred to as the Limited Processing-Sharing (LPS) queue. When the First Come First Served (FCFS) discipline was superseded in popularity for a number of applications, the Processor-Sharing discipline (PS) was one of the main disciplines to take over. Not only does PS perform much better under heavy-tailed service-time distributions, but it is also relatively easy to implement, as no information on the job size is needed. Furthermore, PS is in many ways a fair discipline, since it does not discriminate among jobs based on their arrival time, original size or remaining size to be processed, see, for example, Wierman and Harchol-Balter [118].

However, the PS scheduling mechanism is not always feasible in practice: although the number of jobs in the system may be unbounded, the number of jobs being served simultaneously may not. To overcome the infeasibility of the ordinary PS model, we discuss the so-called *limited processor-sharing discipline with c servers* (LPS-c, or in short, LPS) with an infinite buffer. When there are n jobs in the queue, the service rate for the jobs in service at each server equals $1/\min\{n, c\}$. Clearly, LPS-1 is the same as FCFS, and in the limit $c \to \infty$, the LPS-c is identical to ordinary PS.

Apart from some limiting scenarios, and the case of exponential service times (where the queue-length distribution does not depend on the service discipline), little is known about the LPS queue. Avi-Itzhak and Halfin [6] provide some preliminary insights for the general LPS system, and give an approximation for the expected sojourn time. Unfortunately, this approximation is only accurate when the coefficient of variation of the service times is small. By simulations, Van der Weij [112] obtains some insights in the behavior of the LPS model for small values of c, pointing in the direction that the expected sojourn time could be monotone in c. For the LPS model, very recently, Zhang and co-authors develop a fluid approximation using the framework of measure-valued processes in [127]. The limit of this measure-valued process is obtained under diffusion scaling and heavy traffic conditions and the limit of the total job size process is proved to be a piece-wise reflected Brownian motion [126]. In [128], the authors investigate a diffusion approximation for the LPS queue in the heavy traffic regime.

The main result of this study is that for a class of service-time distributions, namely distributions with a decreasing failure rate, the queue length in the LPS queue is monotonically decreasing in c, in the stochastic order sense. For distributions with an increasing failure rate, the reverse statement holds. Examples of distributions with a decreasing failure rate are Pareto distributions, and (certain) Gamma and Weibull distributions. The normal distribution (at the non-negative domain) and the uniform distribution have an increasing failure rate. In certain applications where the service can be preemptive, it is the number of preempted jobs and jobs in service that may be limited, rather than the number of servers. For this model, we introduce the Limited Foreground-Background (LFB) queue. The FB discipline is a well-known discipline and minimizes the expected queue length in an $G/G/1$ queue for a service-time distribution with decreasing failure rate, where the actual job-size information is unknown, see Righter and Shanthikumar [96]. The FB discipline is a preemptive discipline, that always serves the job with the least amount of service received. Restricting the number of preempted jobs and jobs in service c then yields the LFB-c model. The proof for showing monotonicity in the number of servers for the LPS queue can be used to prove a similar result for the LFB-c queue.

This chapter is organized as follows. In Section 4.2, we introduce the model and notation. The main result is proved in Section 4.3. The results are extended to the LFB queue in Section 4.4. In Section 4.5, we discuss the performance of the LPS queue under another performance metric: the tail of the sojourn-time distribution.

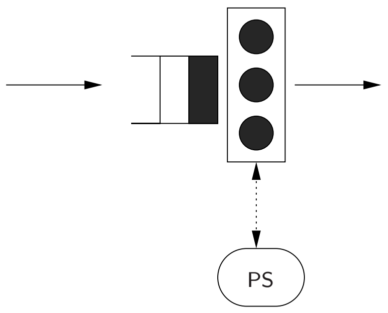


Figure 4.1: Illustration of the LPS-c model for $c = 3$.

We conclude in Section 4.6 by providing directions for further research.

4.2 Model and preliminaries

We consider a queueing model with one queue. Jobs arrive according to an process that is allowed to be any sequence, e.g., a deterministic sequence. We model the service times by a non-negative continuous random variable with distribution function $F(x)$ and density function $f(x)$. The generic job size is denoted by B. The queue has an infinite buffer but a finite number of servers, denoted by c. As long as there are c or less jobs in the system, the queue operates as under the ordinary PS discipline, but as soon as the number of jobs in the queue exceeds c, the behavior of the queue becomes different: the $(c+1)$st job has to wait until one of the c jobs that are in service leaves the system. The service is non-preemptive. Throughout, this model is denoted by $G/G/1$ LPS-c (or LPS). In Figure 4.1 this model is illustrated.

For the service-time distributions, we focus on two cases, namely service-time distributions with an increasing failure rate (IFR) and with a decreasing failure rate (DFR), defined as follows. The failure (or hazard) rate, denoted by $\mu(x)$, is defined as

$$\mu(x) = \frac{f(x)}{1 - F(x)}, \qquad x \geq 0. \tag{4.1}$$

A service-time distribution belongs to the class DFR if $\mu(x)$ is decreasing for all x, i.e., $\mu(x) \geq \mu(y)$ whenever $x \leq y$. A service-time distribution belongs to the class IFR if $\mu(x)$ is increasing for all x. For discrete-time queues, we will use μ to denote the discrete-time failure rate.

Finally, a random variable X is said to be *stochastically smaller* than a random variable Y, notation $X \leq_{st} Y$, if $P(X > x) \leq P(Y > x)$ for all x.

4.3 The monotonicity result

In this section we prove the main result of the study: a characterization of the behavior of the queue length in the $G/G/1$ LPS-c queue with respect to c, the number of servers. We first prove the monotonicity for a discrete-time process by adapting a technique used in Righter and Shanthikumar [96]. After that, we outline the limiting argument that leads to the continuous-time result.

In discrete time, the LPS-c discipline serves the jobs (at most c) in a round-robin fashion. The jobs in service are lined up and the server serves them for one unit of time, from the first to the last job. After serving the last job in the line, the server returns to the front of the queue. If a new job arrives, and there are less than c jobs in service, it is added to the end of the line. If there are more than c jobs in the system, and the service of a job is finished, the $(c+1)$-st job is added at the end of the line. Both changes to the end of the line happen before the server possibly moves back to the front of the line. The state of the process can be described such that it is a Markov process, as required for applying the limiting argument from discrete to continuous time given below. We now first prove the discrete-time case.

Theorem 4.3.1. *Let $X^c(t)$ denote the queue length in the discrete-time $G/G/1$ LPS-c queue at time t. If the service times have a DFR distribution, then for all $t \geq 0$ and $c \in \{1, 2, \ldots\}$, $X^c(t)$ is stochastically decreasing in c. For IFR distributions, the stochastic inequality is reversed.*

Proof. We first introduce artificial discrete-time disciplines denoted by LPS-$c\Box n$, for $n = 0, 1, \ldots$, defined as follows. The first n time steps, LPS-$c\Box n$ behaves exactly like the LPS-$(c+1)$ discipline. From time step $n+1$ onwards, it behaves like the LPS-c discipline. So, intuitively one can think of LPS-$c\Box n$ as a mixture of LPS-c and LPS-$(c+1)$. Furthermore, if there are no more than c jobs in the system up to time n, then LPS-$c\Box n$ and LPS-c are identical. Note that the service discipline is non-preemptive, and therefore the jobs in service will be finished. To prove the theorem, we show that for all $c \in \mathbb{N}$, and $n \in \{0, 1, 2, \ldots\}$,

$$X^{c\Box n}(t) \geq_{st} X^{c\Box n+1}(t), \qquad t \geq 0. \tag{4.2}$$

The theorem then follows from noting that $X^c(t) = X^{c\Box 0}(t)$ and that $X^{c+1}(t) = \lim_{n\to\infty} X^{c\Box n}(t)$ for all $t \geq 0$.

To prove (4.2), fix a $t > n$ (for $t \leq n$ there is nothing to prove) and note that LPS-$c\Box n$ and LPS-$c\Box(n+1)$ are the same up to time n. If LPS-$c\Box n$ and LPS-$c\Box(n+1)$ also serve the same job at time $n+1$, then they are equal forever, and there is nothing left to prove. So, assume that LPS-$c\Box n$ and LPS-$c\Box(n+1)$ serve a different job at time $n+1$. Denote the job served by LPS-$c\Box(n+1)$ at time $n+1$ by x, and let $a(x)$ denote the amount of service it has received by time $n+1$. After that, until job x leaves the queue, LPS-$c\Box(n+1)$ serves at each time step the job that was served by LPS-$c\Box n$ at the previous time step.

Now there are two cases. First, if LPS-$c\Box n$ serves job x at a time $u \leq t$ (note that this is only possible if one of the jobs leaves the LPS-$c\Box n$ queue before time

t), then at time step u, job x has received the same amount of service under both policies, as have all other jobs. So, the queues are the same at time u, and from that moment on, the two queue lengths will be identical. In particular, they will be at time t, which was to be shown.

To conclude the proof, we consider the case that job x is not served by the LPS-$c\square n$ queue before or at time t. Hence, at time t there are exactly two jobs that have received different amounts of service under LPS-$c\square n$ and LPS-$c\square(n+1)$: job x and the job served by LPS-$c\square n$ at time t, denoted by y, with received service $a(y)$, see also the table below.

time	1	$\cdots$	n	$n+1$	$n+2$	$\cdots$	$t-1$	t
job served by LPS-$c\square n$	same	$\cdots$	same	d	e	$\cdots$	g	y
job served by LPS-$c\square n+1$	job	$\cdots$	job	x	d	$\cdots$	f	g

As a consequence, the lengths of the two queues can differ by at most one. The crucial observation is now that $a(y) \geq a(x)$. Since μ is decreasing by assumption, this implies that

$$\begin{aligned}
\mathbb{P}(X^{c\square n}(t) = X^{c\square n+1}(t) - 1) &= \mathbb{P}(\text{job } y \text{ leaves at time } t, \text{ job } x \text{ has not left at time } n+1) \\
&= \mu(a(y))[1-\mu(a(x))] \\
&\leq \mu(a(x))[1-\mu(a(y))] \\
&= \mathbb{P}(\text{job } x \text{ has left at time } n+1, \text{ job } y \text{ does not leave at time } t) \\
&= \mathbb{P}(X^{c\square n}(t) = X^{c\square n+1}(t) + 1).
\end{aligned} \tag{4.3}$$

Since $X^{c\square n}(t)$ and $X^{c\square n+1}(t)$ can differ at most one, we conclude from (4.3) that $X^{c\square n}(t) \geq_{st} X^{c\square n+1}(t)$. For IFR service-time distributions, the inequality in (4.3) is reversed. This completes the proof. $\square$

The discrete-time result of Theorem 4.3.1 can be turned into the corresponding continuous-time statement by using a result of Böhm and Mohanty [13] that links the discrete process to a continuous process by an appropriate passage to the limit. Additionally, note that in Section 4 in [104] the converge result is proved, linking the discrete round-robin discipline to the processor-sharing discipline.

This limiting procedure is proven in [13] and applies to Markovian queueing processes, which includes the model under consideration. To apply the limiting procedure, discrete time marks $0, 1, \ldots, n$ are considered, and the unit in which time is measured is changed to $\Delta = t/n$. These new time marks $0, \Delta, 2\Delta, \ldots$ form a partition of the interval $[0, t]$ into n slots of equal length. Using this time partition, the arrival rate and service rate have to be scaled so that the n-step transition probabilities of a discrete-time process Z_k converge to the probabilities of the continuous-time process $Z_k(t)$ as $n \to \infty$, see [14]. The continuous counterpart of the distribution results (e.g., queue-length distribution) in Section 4 of Katzenbeisser and Panny [62] can be obtained mechanically and are found in Section 4 of Böhm and Panny [14], who extend the proof in [13] to arbitrary transition probabilities (see Section 1 of [62] for the model assumptions). Following this path, the discrete-time result in Theorem 4.3.1 implies the continuous-time result.

Theorem 4.3.1 matches with the following heuristic. For DFR service times, among all work-conserving disciplines P, the queue length $X(t)^P$ is (stochastically) maximal under P=FCFS, and minimal under P=FB, for all t, see Righter and Shanthikumar [96]. For $c = 1$, LPS is the same as FCFS. Furthermore, the larger c, the more LPS differs from FCFS, and the more it behaves like PS, but also like FB. Hence, intuitively, the larger the c, the smaller the queue length.

Assuming that the load (or throughput) ρ satisfies $\rho < 1$, the workload process converges to a stationary state as the time goes to infinity, see Asmussen (Proposition 1.1., Chapter VIII) [4]. Let X^c be the stationary queue length in the LPS-c queue. By letting the time t go to infinity, we obtain the following corollary.

Corollary 4.3.2. *If the service times have a DFR distribution, and $\rho < 1$, then $X^{c+1} \leq_{st} X^c$ for all $c \in \mathbb{N}$. For IFR service times, $X^{c+1} \geq_{st} X^c$ for all $c \in \mathbb{N}$.*

It is interesting to compare Corollary 4.3.2 with the following heavy-traffic approximation of the expected queue length found in Zhang and Zwart [128], which is

$$\mathbb{E}X^c \approx \frac{\rho}{1-\rho} \frac{\nu_a^2 + \nu_s^2}{2(1+\nu_s^2)} \Big(2 + (\nu_s^2 - 1)\rho^{c^{\frac{1+\nu_a^2}{\nu_a^2+\nu_s^2}}}\Big),$$

where ν_a and ν_s are the coefficients of variation of the interarrival and service-time distributions, respectively. This expression is decreasing in c if and only if $\nu_s \geq 1$. Since all DFR distributions satisfy this condition, the approximation of $\mathbb{E}X$ is decreasing in c for a class of distributions that includes DFR distributions. On the other hand, the ordering in Corollary 4.3.2 is stronger, namely stochastic instead of 'in expectation'. Obviously, for IFR distributions, comparable statements hold.

Consider an LPS queue where jobs are coming from two different classes, and the server does not know from which class a job comes. The density f of the service distribution in such a queue is a mixture of the density functions f_1 and f_2 of the respective classes, and given by $f = \alpha f_1 + (1-\alpha) f_2$, for some $\alpha \in (0, 1)$. Since such a mixture of DFR distributions is again DFR, see Barlow and Proschan [9], Theorem 4.3.1 automatically holds for this queue, as long as the distribution in each class is DFR. In general, the reverse does generally not hold for IFR distributions, since a mixture of IFR distributions is not necessarily IFR. For example, a mixture of exponential distributions is hypergeometric, which is not IFR.

The DFR condition in Theorem 4.3.1 is important. Simulations suggest that the monotonicity results do not hold for a lognormal service-time distribution with coefficient of variation 1.72, mean job size 1 and $\rho = 0.8$, for different values of c, although this distribution has a coefficient of variation larger than one. Little's law implies the following result for the stationary sojourn time, S^c.

Corollary 4.3.3. *If the service times have a DFR distribution, and $\rho < 1$, then $\mathbb{E}S^{c+1} \leq \mathbb{E}S^c$. For IFR and $\rho < 1$, we have $\mathbb{E}S^{c+1} \geq \mathbb{E}S^c$.*

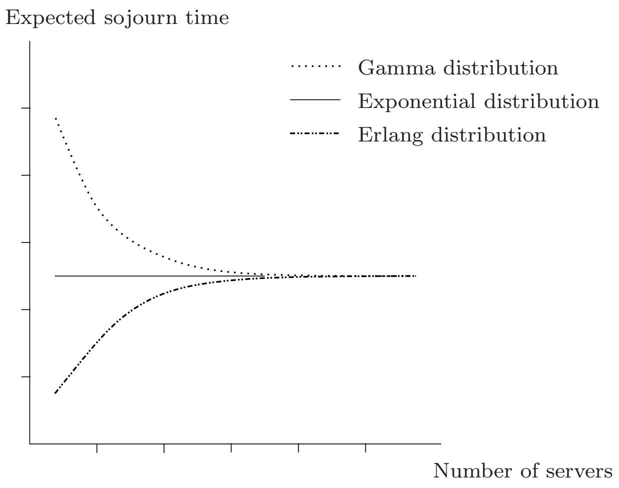


Figure 4.2: The expected sojourn time in the $M/G/1$ LPS queue for $\lambda = 4/15$, $\rho = 0.8$ and Erlang(2), exponential, and Gamma(0.5, 12) service-time distributions.

To illustrate Corollary 4.3.3, in Figure 4.2 we have simulated the expected sojourn time for a number of $M/G/1$ LPS queues where the service times have strictly decreasing (Gamma), strictly increasing (Erlang) and constant failure rates (exponential). In the figure the point estimations of the expected sojourn time are presented. Confidence intervals are omitted for ease of the discussion.

4.4 The Limited FB discipline

In certain applications where the service may be preemptive, the limit is not so much on the number of servers, but it is the number of preempted jobs that is limited, see for example Van der Mei et al. [79]. Assume that at most c jobs are allowed to receive service. These jobs constitute the *inner queue*. All other jobs

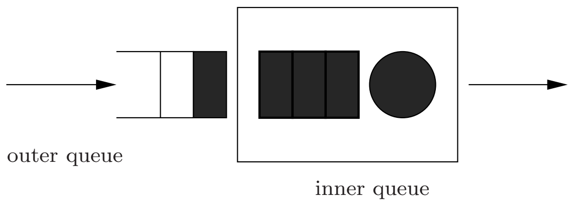


Figure 4.3: The $G/G/1/4/\infty$ model.

are waiting in the *outer queue*, and are only allowed to enter the inner queue when the number of jobs in the inner queue is smaller than c, see Figure 4.4. We call this queue the *inner-outer queue with* c *positions,* and denote it by $G/G/1/c/\infty$. The LPS queue with c servers discussed in the previous section is an example of an inner-outer queue.

In this section, we discuss the Limited FB discipline, denoted by LFB. We will show that this discipline stochastically minimizes the queue length in the $G/G/1/c/\infty$ queue under DFR service-time distributions for each c, where the queue length is the sum of the jobs in the inner queue plus the outer queue. Then, using the proof technique used in the previous section, we show that for DFR distributions, under the LFB discipline, the queue length is stochastically decreasing in c.

First, let us describe how the normal FB discipline works. Denoting by the *age* of a job the amount of service it has received, FB serves the youngest job. If there are n youngest jobs, they are served simultaneously, in a processor-sharing manner, at rate $1/n$. In particular, when a new job arrives, the job in service is preempted, and the new job is served immediately. The preempted job is assumed to occupy a server. See Nuyens and Wierman [86] for a recent survey on the FB discipline.

We now modify the FB discipline to fit to the $G/G/1/c/\infty$ framework. The *limited* FB *queue* with c positions, denoted by LFB-c, is defined as follows: under LFB-c, the server preemptively serves the youngest job, but only if there are c jobs or less in the inner queue. Hence, if a new job arrives it is served immediately if and only if there are $c-1$ jobs or less in the (inner) queue. If there are c jobs in the inner queue, the server can only switch to an unserved job when one of the c jobs leaves the queue. Like in the PS queue, LFB-1 is equal to FCFS, while for $c \to \infty$, LFB-c converges to the ordinary FB discipline. The FB discipline can be considered to be the opposite of FCFS: FB always serves the youngest job(s), while FCFS serves the oldest. Hence, LFB-c has the interesting feature that when c runs from 1 to ∞, LFB-c moves from FCFS to its opposite, FB.

In fact, to prove the optimality of LFB-c in the $G/G/1/c/\infty$ queue, we can simply copy the proof of the optimality of FB in $G/G/1/\infty$ queues given in Theorem 2.1 of Righter and Shanthikumar [96]. Hence, we have the following result

Theorem 4.4.1. *Consider the $G/G/1/c/\infty$ queue with DFR service times. Let*

$X^{\mathsf{P}}(t)$ denote the queue length at time t under discipline P. Then for all disciplines P that do not use information on the exact job sizes, we have $X^{\mathsf{LFB}\text{-}c}(t) \leq_{st} X^{\mathsf{P}}(t) \leq_{st} X^{\mathsf{FCFS}}(t)$. For IFR job sizes, the inequalities are reversed.

Since the queue-length distribution is the same for all queues with a non-preemptive service discipline, one can put in Theorem 4.4.1 any non-preemptive service discipline (e.g., ROS, or LCFS) instead of FCFS.

Following the lines in the proof of Theorem 4.3.1, the following stochastic ordering result holds:

Theorem 4.4.2. *Let $X^c(t)$ denote the queue length in the $G/G/1/c/\infty$ LFB-c queue with a DFR service-time distribution, then $X^c(t)$ is stochastically decreasing in c for all t. For IFR service times, the inequalities are reversed.*

Combining Theorems 4.3.1, 4.4.1, and 4.4.2 with Theorem 2.1 of Righter and Shanthikumar [96], we have the following result for the different queue lengths discussed in this chapter:

Corollary 4.4.3. *For $\rho < 1$, in the $G/G/1$ queue with DFR service times, we have for $\mathsf{P} \in \{\mathsf{LFB}\text{-}c, \mathsf{PS}\}$ that*

$$X^{\mathsf{FB}} \leq_{st} X^{\mathsf{P}} \leq_{st} X^{\mathsf{LPS}\text{-}c} \leq_{st} X^{\mathsf{FCFS}}.$$

For IFR service times, the inequalities are reversed.

4.5 The tail of the sojourn-time distribution

In this section, we investigate the performance of the LPS queue in heavy traffic, measured by the likelihood of a very long sojourn time. To be precise, denoting the sojourn time of a discipline P by S_{P}, we consider the decay rate $\gamma(S_{\mathsf{P}})$, defined as

$$\gamma(S_{\mathsf{P}}) = -\lim_{u\to\infty} \frac{1}{u} \log P(S_{\mathsf{P}} > u),$$

whenever this limit exists. Hence, the smaller the decay rate, the larger the tail of the distribution. We prove the following theorem.

Theorem 4.5.1. *In the $M/G/1$ queue, if $\mathbb{E}[\exp(sB)] < \infty$ for some $s > 0$, then for any c, the equality $\gamma(S_{\mathsf{LPS}\text{-}c}) = \gamma(S_{\mathsf{FCFS}})$ holds for $\rho < 1$ large enough.*

Before giving the proof, we discuss the result. In case of light-tailed service times (e.g., exponential and uniform distributions), the decay rate of the sojourn time under any work-conserving discipline lies in the (non-empty) interval $[\gamma(L), \gamma(W)]$, where W is the stationary workload, and L the length of a generic busy period. Most well-known policies have a decay rate that equals one of these extremes. The lower bound is matched for LCFS, FB, and, under mild additional conditions, SRPT, and PS. Under FCFS, the upper bound is achieved. For an overview of results on the decay rates of different disciplines, see Nuyens and Zwart [87] and the references therein.

The remarkable conclusion we can draw from Theorem 4.5.1 is that no matter how large c is, the asymptotic behavior of the decay rate of the sojourn-time distribution under LPS in heavy traffic is equal to that of FCFS, which is optimal, and not to that of PS, which in most cases is the worst possible discipline. This means that in heavy traffic, the 'FCFS-like' property of LPS (once a job is served, it is guaranteed a minimum rate) dominates its 'PS-like' property that the service rate is shared with other jobs in the system.

Proof of Theorem 4.5.1. By the PASTA property, an arriving job has to wait at most W before its service starts, and then it is served with rate at least $1/c$. Hence, the sojourn time $S_{\mathsf{LPS}-c}$ of a job of size B in the LPS queue satisfies $S_{\mathsf{LPS}-c} \leq_{st} W + cB$, where W is the stationary workload in the $M/G/1$ queue, and W and B are independent. Since $\gamma(X+Y) = \min\{\gamma(X), \gamma(Y)\}$ when X and Y are independent, see for instance [73], we have

$$\gamma(S_{\mathsf{LPS}\text{-}c}) \geq \min\{\gamma(W), \gamma(B)/c\}.$$

Since $\gamma(W) \to 0$ as $\rho \to 1$, see for example Mandjes and Zwart [74], there exists a $\rho(c)$ such that $\gamma(W) \leq \gamma(B)/c$ for all $\rho(c) \leq \rho < 1$. Hence, for those values of ρ, we have $\gamma(S_{\mathsf{LPS}}) \geq \gamma(W)$. Since the service times have an exponential moment, the inequality $\gamma(S_{\mathsf{LPS}}) \leq \gamma(W)$ holds, see Stolyar and Ramanan [108]. Hence, $\gamma(S_{\mathsf{LPS}}) \leq \gamma(W)$ for $\rho \geq \rho(c)$, which completes the proof. □

Remark 4.5.2. In some cases, $\gamma(W)$ and $\gamma(B)$ can be calculated explicitly. For example, in the $M/M/1$ queue where the interarrival and service times have parameters λ and μ, we have $\gamma(W) = \mu - \lambda$ and $\gamma(B) = \mu$. Writing $\rho = \lambda/\mu$, we have

$$\gamma(W) \leq \frac{\gamma(B)}{c} \;\Leftrightarrow\; \mu - \lambda \leq \frac{\mu}{c} \;\Leftrightarrow\; 1 - \rho \leq \frac{1}{c} \;\Leftrightarrow\; \rho \geq 1 - \frac{1}{c}.$$

Hence, for all ρ such that $1 - c^{-1} \leq \rho < 1$, we have $\gamma(S_{\mathsf{LPS}}) = \gamma(W) = \mu - \lambda$.

4.6 Conclusion and topics for further research

In this chapter we studied monotonicity properties for the LPS queue. It was shown that for service-time distributions with a decreasing failure rate the queue length is stochastically decreasing in the number of servers. And the reverse is true for service-time distributions with an increasing failure rate. These results also hold for the FB-c queue. Next, the decay rate is considered, and both the queue-length distribution and their asymptotic decay rates are compared to several other service disciplines.

We discuss two possible extensions of the LPS model that are interesting for further research. One possible extension of the LPS model is to consider two $G/G/1$ LPS queues in tandem, see the left part of Figure 3.2. In this tandem model, jobs arrive at the first queue according to a process that is allowed to be any sequence. After completion of service, they are routed to the second queue, and after the completion of the service at that queue the jobs leave the network. The queues have infinite buffers but finite numbers of servers, denoted by c_1 and c_2. As long as there are c_i or less jobs in each queue (for $i = 1, 2$), the queue mimics the PS queue where the total capacity is equally shared among all occupied servers at the two queues. But as soon as the number of jobs in queue i, say, exceeds c_i, the behavior of the system differs: the $(c_i + 1)$st job has to wait until one of the c_i jobs that are in service leaves the system. For this queue, it is interesting to study under which conditions there is monotonicity in c_1 and c_2. For exponentially distributed service times, the expected overall queue length is minimized if $c_1 = 1$ and c_2 as large as possible, while for each queue in isolation the expected queue length is increasing in c_i, see also [112].

A second extension of the LPS model is the following. Consider again two $G/G/1$ LPS queues, and assume that these queues share a common resource in the same manner as the model above. The two arrival streams are now independent, and each queue is fed by one of these arrival streams. After service at the queue, the job immediately leaves the system (instead of being routed to another queue), see the right part of Figure 3.1. As far as we know, no monotonicity results on the queue length with respect to c_1 and c_2 are known. Since the number of servers assigned to queue 1 influences the capacity assigned to queue 2, this model cannot be solved by the same techniques as used in this chapter.

CHAPTER 5

DYNAMIC SCHEDULING FOR THE LPS QUEUE

The next two chapters deal with scheduling jobs in layered queueing models. First, in this chapter we study a limited processor-sharing queue (LPS) at which jobs arrive according to a Poisson process. Each job has a service time that is distributed according to a phase-type distribution. Based on the number of jobs in each phase of its service, the processor can admit a limited number of jobs for service. We present a new decomposition-based approach to analyze monotonicity properties that leads to a complete characterization of the optimal scheduling policy providing new fundamental insights into the optimal control of LPS queues. We provide a numerical assessment of the performance of the optimal dynamic policy compared to several commonly used static policies and the optimal static policy. The results show that strong gains can be obtained by using the optimal dynamic policy. This chapter is based on [114].

5.1 Introduction

In this chapter we consider optimal dynamic scheduling policies for the LPS queue. Jobs arrive according to a Poisson process and bring work to the system that is distributed according to a phase-type distribution. Upon arrival a job can either wait in the queue for service, or be assigned a server immediately. We study how many jobs should be served simultaneously based on the complete state information, i.e., the number of jobs present in each phase of their service and the number of jobs waiting in the queue. We cast this problem as a Markov decision problem and study the properties of the dynamic programming relative value function. We derive monotonicity properties, which provide fundamental insights into the behavior of the system. Based on this, we obtain optimal dynamic scheduling policies. We illustrate how effective the dynamic policies are by comparing them to a number of commonly used static policies and the optimal static policy. The results show that a significant improvement can be obtained.

Significant work has been done in the static optimization of the number of servers in multiserver queues. Stidham [60] and Brumelle [22] study $G/E_n/c$ models and show that the number of jobs in the system is stochastically minimized by setting c to 1 under the same utilization factor; the intuitive explanation behind this is that a single-server system fully utilizes the service capacity, whereas in a multiserver system servers may be idle even when the system is not empty. Brumelle shows that in a $GI/GI/c$ queue the single-server queue, i.e., $c = 1$, minimizes the expected number of jobs in the system when the coefficient of variation c_S for the service-time distribution is less than 1. Based on these results, Yamazaki and Sakasegawa [120] conjecture that for any LPS queue for any $c > 1$ if $c_S < 1$ the system performance degrades compared to an equally loaded single-server queueing system, while the reverse is conjectured to be true for $c_S > 1$. As far as we know, dynamic optimization of the LPS queue has not been analyzed in detail.

The contribution of this study is three-fold. First, we derive a number of monotonicity properties for the dynamic programming relative value function for the LPS queue, that leads to a complete characterization of the optimal dynamic scheduling policy. These results are among the first exact analytic results for the LPS queue. Second, we study the LPS queue in a Markov decision theoretic framework which leads to a problem with a very large state space. We provide an insightful approach based on decomposition of terms related to arrivals and departures to study monotonicity properties of the relative value function. This leads to a complete characterization of the optimal policy. Third, we provide a numerical assessment of the gains that can be obtained by using the dynamic policy over commonly used policies and the optimal static policies.

The remainder of this chapter is organized as follows. In Section 5.2 the LPS model is described and the dynamic programming relations are formulated. In Section 5.3 we study monotonicity properties of the dynamic programming relative value function, that lead to the optimal dynamic policy. In Section 5.4 we provide the assessment through extensive numerical experiments followed by a discussion. We conclude the chapter with Section 5.5 where we address a number of challenging topics for further research.

5.2 Model and preliminaries

Consider a multiserver queue with c servers to which jobs arrive according to a Poisson process with rate λ. The servers work in a processing-sharing (PS) manner, so that whenever there are $k \leq c$ jobs in service, every job receives a fraction $1/k$ of the processing capacity; jobs that find all c servers busy upon arrival enter an infinite buffer (the model is illustrated in Figure 4.1 for the case $c = 3$). The service-time distribution of a job is modeled by a phase-type distribution which is characterized by a quadruple $(M + 1,\ \eta,\ \mu,\ P)$. Here M stands for the number of phases (and state $M + 1$ is the absorbing state), and $\eta = (\eta^{(1)}, \ldots, \eta^{(M)})$ with $\eta^{(j)}$ the probability that the service starts in phase j, $\mu = (\mu^{(1)}, \ldots, \mu^{(M)})$ with $\mu^{(j)}$ the rate of the exponential distribution of the time that the process spends in phase j, $P = (p^{(i,j)})_{i=1,\ldots,M,\ j=1,\ldots,M+1}$ with $p^{(i,j)}$ the probability that the process jumps to phase j (or the absorbing state when $j = M + 1$) upon leaving phase i. The absorbing state corresponds to a completion of the service requirement. It is well-known that phase-type distributions are dense in the class of all non-negative distributions retaining their tractability [102]. Therefore, it is possible to model heavy-tailed distributions by phase-type distributions [5, 44].

We are interested in minimizing the expected number of jobs in the system by adjusting c dynamically based on the state of the system, which is described by the tuple $(\mathbf{x},\ \mathbf{k})$, where $\mathbf{x}$ is the vector denoting the number of jobs in each phase (i.e., $x^{(j)}$ is the number of jobs in phase j that are waiting and in service at the queue), and $\mathbf{k}$ is the vector that represents the number of jobs $k^{(j)}$ in phase j in service, for $j = 1, \ldots, M$.

To obtain the optimal scheduling policy π^* that minimizes the expected number of jobs, we model the system in a Markov decision framework. To this end, we uniformize the system (see Section 11.5 of [93]). For simplicity we assume that $\lambda + \max_{\{j=1,\ldots,M\}\mu^{(j)}} < 1$; without loss of generality, we can always get this by scaling. Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the arrival and service rates to be transition probabilities.

Let $V(\mathbf{x}, \mathbf{k})$ be a real-valued function defined on the state space $\mathbb{N}^M \times \mathbb{N}^M$. This function will play the role of the relative value function, i.e., the asymptotic difference in total costs that results from starting the process in state $(\mathbf{x}, \mathbf{k})$ instead of some reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) $g + V = TV$, where T is the dynamic programming operator acting on $V(\mathbf{x}, \mathbf{k})$ defined in Equation (5.1) below.

The first term in the expression $TV(\mathbf{x}, \mathbf{k})$ (see Equation (5.1)) models the direct costs, i.e., the total number of jobs in the system. The second term models the arrivals, which occur with rate $\lambda\eta^{(j)}$ to phase j. The transitions of a job to a different phase are given by the third term. A transition from phase j to phase l for a job occurs with rate $\mu^{(j)}p^{(j,l)}$, but this is adjusted by the factor $k^{(j)}/(k^{(1)} + \cdots + k^{(M)})$, since that job only uses a fair share of the service capacity. Note that the system is specifically modeled such that when a job moves from one phase to another, the previously assigned server is not lost. The server is only released upon completion of the service requirement. The service discipline is a non-preemptive discipline. The next term, which accounts for a transition to the absorbing state, is similarly explained. The last term is the uniformization constant to account for the dummy transitions added to the model.

$$\begin{aligned} TV(\mathbf{x},\mathbf{k}) &= \sum_{j=1}^{M} x^{(j)} + \lambda \sum_{j=1}^{M} \eta^{(j)} H\big(\mathbf{x}+\mathbf{e}^{(j)},\mathbf{k}\big) \\ &+ \sum_{j=1}^{M}\sum_{l=1}^{M} \frac{k^{(j)}\mu^{(j)}p^{(j,l)}}{k^{(1)}+\cdots+k^{(M)}} H\big(\mathbf{x}-\mathbf{e}^{(j)}+\mathbf{e}^{(l)},\mathbf{k}-\mathbf{e}^{(j)}+\mathbf{e}^{(l)}\big) \\ &+ \sum_{j=1}^{M} \frac{k^{(j)}\mu^{(j)}p^{(j,M+1)}}{k^{(1)}+\cdots+k^{(M)}} H\big(\mathbf{x}-\mathbf{e}^{(j)},\mathbf{k}-\mathbf{e}^{(j)}\big) \\ &+ \left[1-\lambda-\sum_{j=1}^{M} \frac{k^{(j)}\mu^{(j)}}{k^{(1)}+\cdots+k^{(M)}}\right] V(\mathbf{x},\mathbf{k}), \end{aligned} \tag{5.1}$$

with $\mathbf{e}^{(j)}$ the unit vector with all entries zero, except for the j-th entry for $j = 1,\ldots,M$. The actions, denoted by H, are given by

$$H(\mathbf{x},\mathbf{k}) = \min\left\{ V\big(\mathbf{x},\mathbf{k}+\sum_{j=1}^{M} a^{(j)}\,\mathbf{e}^{(j)}\big) \;\middle|\; 0 \le a^{(j)} \le x^{(j)} - k^{(j)}, a^{(j)} \in \mathbb{N}_0 \right\}, \tag{5.2}$$

with $\mathbb{N}_0 = \{0,1,2,\ldots\}$, where action $a^{(j)}\,\mathbf{e}^{(j)}$ represents the action that schedules $a^{(j)}$ additional jobs that are waiting in the queue and start in phase j.

The optimality equation $g + V = TV$ is hard to solve analytically in practice. Alternatively, the optimal actions can also be obtained by recursively defining

$$V_{n+1} = TV_n, \tag{5.3}$$

for $n = 0,\ldots$ and arbitrary V_0. For $n \to \infty$, the maximizing actions converge to the optimal ones (for existence and convergence of solutions and optimal policies we refer to [93]). Consequently, when V is known, we can restrict our attention to the function H to obtain the optimal actions. In Section 5.4 we adopt this approach to numerically compute optimal policies.

5.3 Optimal scheduling policy

In this section we identify the policy that minimizes the expected number of jobs in the system. To this end, the following notation is convenient. Let $s^{(j)}$ denote the mean passage time from phase j to the absorbing state $M+1$. The mean passage times can be uniquely determined by solving the following set of linear equations:

$$s^{(j)} = 1/\mu^{(j)} + \sum_{l=1}^{M} p^{(j,l)} s^{(l)},$$

for $j = 1, \ldots, M$. Without loss of generality, we can rank the variables $s^{(j)}$ ($j = 1, \ldots, M$) in increasing order, such that $s^{(1)} \leq \cdots \leq s^{(M)}$. The main result is the complete characterization of the optimal policy given in the following theorem.

Theorem 5.3.1. *For arbitrary* $(\mathbf{x}, \mathbf{k}) \in \mathbb{N}^M \times \mathbb{N}^M$*, we have*

$$V(\mathbf{x}, \mathbf{k}) \geq V(\mathbf{x}, \mathbf{k}^*) \text{ for } k^{*(i)} := \mathbb{1}_{\{i=\min\{j:x^{(j)}>0\},\mathbf{k}=0\}} + x^{(i)}\mathbb{1}_{\{\sum_{j=i+1}^{M} k^{(j)}>0\}}, \quad (5.4)$$

where $\mathbb{1}_E$ *is the indicator function on the event* E*.*

In words, the theorem states that the optimal policy is as follows: when no jobs are in service, then *only one job amongst the ones with the shortest expected service time* (i.e., the smallest $s^{(j)}$) amongst the jobs waiting in the queue is taken into service (the first term of $k^{*(i)}$). Otherwise, *all jobs with a strictly shorter expected service time* than the job in service with the longest remaining service time are taken into service *simultaneously* (the second term of $k^{*(i)}$).

To prove Theorem 5.3.1, monotonicity properties of the relative value function are required, namely increasingness and submodularity. These properties are formulated and proven in Lemmas 5.3.2 and 5.3.3 below. For ease of the discussion, we formulate the proofs for a hyper-exponential service-time distribution with two phases (referred to as the H_2-distribution); note the H_2-distribution occurs as a special case of the phase-type distribution by taking

$$\begin{aligned} &M = 2, \eta^{(1)} = p^{(1)}, \eta^{(2)} = p^{(2)}, \\ &p^{(1,2)} = p^{(2,1)} = p^{(1,1)} = p^{(2,2)} = p^{(1,3)} = 0, \text{ and } p^{(2,3)} = 1. \end{aligned} \quad (5.5)$$

A generalization of the proof to arbitrary phase-type distribution is straightforward but notationally cumbersome (see also the remarks at the end of Section 4). Finally, the proof of Theorem 5.3.1 is given at the end of this section.

Lemma 5.3.2 (Increasingness)**.** *The relative value function* $V(\mathbf{x}, \mathbf{k})$ *is non-decreasing in* $x^{(j)}$ *for all* $j = 1, \ldots, M$*, independent of* $\mathbf{k}$ *for all* $(\mathbf{x}, \mathbf{k}) \in \mathbb{N}^M \times \mathbb{N}^M$*.*

Proof. The proof proceeds via induction on n in the relation (5.3). To start, note that by assumption $V_0(\mathbf{x}, \mathbf{k}) \equiv 0$, which is clearly non-decreasing in $x^{(j)}$ for all $j = 1, \ldots, M$. Next, we assume that $V_n(\mathbf{x}, \mathbf{k})$ is non-decreasing in $\mathbf{x}$ for some $n \geq 0$. To show that $V_{n+1}(\mathbf{x}, \mathbf{k})$ is also non-decreasing in $\mathbf{x}$ one needs to show that the difference $V_{n+1}(\mathbf{x} + \mathbf{e}^{(j)}, \mathbf{k}) - V_{n+1}(\mathbf{x}, \mathbf{k})$ is non-negative. This result directly follows from the dynamic programming relation (5.1) by a pair wise comparison of the corresponding terms where the first term (i.e., $\sum_{j=1}^{M} x^{(j)}$) yields a difference of 1, and where the induction hypothesis can be applied to the other terms. The result now follows by taking the limit for n to infinity. □

In words, Lemma 5.3.2 states that the relative value function has the property that the system is subject to higher costs when having more jobs in the system. Therefore, the optimal actions in H aim to achieve as few jobs in the system as possible while taking into account the future evolution of the system. In order to fully derive this property, we also need that the relative value function is submodular as formulated in Lemma 5.3.3 below.

Lemma 5.3.3 (Submodularity)**.** *For $x^{(1)} > 0$, $x^{(2)} > 0$, $k^{(1)} \geq 0$, $k^{(2)} > 0$, it holds that*

$$\begin{aligned}
&\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}H(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)}) \\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}H(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1) \\
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}V(x^{(1)},x^{(2)},k^{(1)},k^{(2)})-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}V(x^{(1)},x^{(2)},k^{(1)},k^{(2)}) \\
&-\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}H(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)}) \\
&-\frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}H(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1) \\
&+\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}V(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)})+\frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}V(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)}) \\
&\geq 0.
\end{aligned} \tag{5.6}$$

Proof. See Appendix A. □

The submodularity property is a generalization of convexity in more dimensions. This property together with the increasingness property are necessary to prove the elegant structure of the policy as stated in Theorem 5.3.1.

Proof of Theorem 5.3.1. Similar to the proof of Lemma 5.3.2, the proof is based on induction on n. The relative value function when using the parameters as stated in (5.5) reduces to

$$\begin{aligned}
V_{n+1}(\mathbf{x},\mathbf{k}) = &\sum_{j=1}^{M} x_j + \lambda\sum_{j=1}^{M} p^{(j)} H_n(\mathbf{x}+\mathbf{e}^{(j)},\mathbf{k}) \\
&+\sum_{j=1}^{M}\frac{k^{(j)}\mu^{(j)}}{\sum_{j=1}^{M}k^{(j)}}H_n(\mathbf{x}-\mathbf{e}^{(j)},\mathbf{k}-\mathbf{e}^{(j)}) \\
&+\left[1-\lambda-\sum_{j=1}^{M}\frac{k^{(j)}\mu^{(j)}}{\sum_{j=1}^{M}k^{(j)}}\right]V_n(\mathbf{x},\mathbf{k}),
\end{aligned} \tag{5.7}$$

with

$$H_n(\mathbf{x},\mathbf{k}) = \min\left\{V_n(\mathbf{x},\mathbf{k}+a\mathbf{e}^{(j)}) \;\mid j=1,\dots,M, 0\leq a^{(j)}\leq x^{(j)}-k^{(j)}, a^{(j)}\in\mathbb{N}_0\right\},$$

for $i = 0, 1, \ldots$ and $(\mathbf{x}, \mathbf{k}) \in \mathbb{N}^M \times \mathbb{N}^M$, $M = 2$, and $V_0(\mathbf{x}, \mathbf{k}) \equiv 0$ and with $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$. Using these definitions, it directly follows that $V_0(\mathbf{x}, \mathbf{k}) - V_0(\mathbf{x}, \mathbf{k}^*) \geq 0$. Without loss of generality, we assume that $\mu^{(1)} \geq \mu^{(2)}$. Next, we assume that (5.4) holds for $V_n(\mathbf{x}, \mathbf{k})$ for some $n \geq 0$. Then we have to prove that

$$V_{n+1}(\mathbf{x}, \mathbf{k}) - V_{n+1}(\mathbf{x}, \mathbf{k}^*) \geq 0, \tag{5.8}$$

where k_i^* is defined in Theorem 5.3.1. To this end, using (5.7) we can write

$$\begin{aligned}
V_{n+1}(\mathbf{x}, \mathbf{k}) - V_{n+1}(\mathbf{x}, \mathbf{k}^*) \;=\; & \lambda p^{(1)} H_n(x^{(1)} + 1, x^{(2)}, k^{(1)}, k^{(2)}) \\
& + \lambda p^{(2)} H_n(x^{(1)}, x^{(2)} + 1, k^{(1)}, k^{(2)}) \\
& + \frac{k^{(1)}\mu^{(1)}}{k^{(1)} + k^{(2)}} H_n(x^{(1)} - 1, x^{(2)}, k^{(1)} - 1, k^{(2)}) \\
& + \frac{k^{(2)}\mu^{(2)}}{k^{(1)} + k^{(2)}} H_n(x^{(1)}, x^{(2)} - 1, k^{(1)}, k^{(2)} - 1) \\
& + \left(1 - \lambda p^{(1)} - \lambda p^{(2)} - \frac{k^{(1)}\mu^{(1)}}{k^{(1)} + k^{(2)}} - \frac{k^{(2)}\mu^{(2)}}{k^{(1)} + k^{(2)}}\right) V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\
& - \lambda p^{(1)} H_n(x^{(1)} + 1, x^{(2)}, k^{*(1)}, k^{*(2)}) \; - \lambda p^{(2)} H_n(x^{(1)}, x^{(2)} + 1, k^{*(1)}, k^{*(2)}) \\
& + \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)} + k^{*(2)}} H_n(x^{(1)} - 1, x^{(2)}, k^{*(1)} - 1, k^{*(2)}) \\
& - \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)} + k^{*(2)}} H_n(x^{(1)}, x^{(2)} - 1, k^{*(1)}, k^{*(2)} - 1) \\
& - \left(1 - \lambda p^{(1)} - \lambda p^{(2)} - \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)} + k^{*(2)}} - \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)} + k^{*(2)}}\right) \\
& \quad \times V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}),
\end{aligned} \tag{5.9}$$

which can be rewritten as

$$V_{n+1}(\mathbf{x}, \mathbf{k}) - V_{n+1}(\mathbf{x}, \mathbf{k}^*) = A + D, \tag{5.10}$$

where the arrival-related and departure-related terms are grouped into

$$\begin{aligned}A :=&\lambda p^{(1)} H_n(x^{(1)}+1, x^{(2)}, k^{(1)}, k^{(2)}) \ + \lambda p^{(2)} H_n(x^{(1)}, x^{(2)}+1, k^{(1)}, k^{(2)}) \\ &+ \left(1-\lambda p^{(1)}-\lambda p^{(2)}\right) V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\ &- \lambda p^{(1)} H_n(x^{(1)}+1, x^{(2)}, k^{*(1)}, k^{*(2)}) \ - \lambda p^{(2)} H_n(x^{(1)}, x^{(2)}+1, k^{*(1)}, k^{*(2)}) \\ &- \left(1-\lambda p^{(1)}-\lambda p^{(2)}\right) V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)})\end{aligned} \tag{5.11}$$

and

$$\begin{aligned}D :=&\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}} H_n(x^{(1)}-1, x^{(2)}, k^{(1)}-1, k^{(2)}) \\ &+ \frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}} H_n(x^{(1)}, x^{(2)}-1, k^{(1)}, k^{(2)}-1) \\ &- \frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\ &- \frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\ &- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}} H_n(x^{(1)}-1, x^{(2)}, k^{*(1)}-1, k^{*(2)}) \\ &- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}} H_n(x^{(1)}, x^{(2)}-1, k^{*(1)}, k^{*(2)}-1) \\ &+ \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}) \\ &+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}),\end{aligned} \tag{5.12}$$

respectively. Below we will show that $A, D \geq 0$. To this end, we first consider A, and observe that by the induction hypothesis we find that

$$\lambda p^{(1)} H_n(x^{(1)}+1, x^{(2)}, k^{(1)}, k^{(2)}) \geq \lambda p^{(1)} H_n(x^{(1)}+1, x^{(2)}, k^{*(1)}, k^{*(2)}), \tag{5.13}$$

since the action function $H(\cdot,\cdot)$ forces the second term to be smaller than the first term. Similarly, it holds that

$$\lambda p^{(2)} H_n(x^{(1)}, x^{(2)}+1, k^{(1)}, k^{(2)}) \geq \lambda p^{(2)} H_n(x^{(1)}, x^{(2)}+1, k^{*(1)}, k^{*(2)}), \tag{5.14}$$

and

$$\begin{aligned}&\left(1-\lambda p^{(1)}-\lambda p^{(2)}\right) V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\ &\geq \left(1-\lambda p^{(1)}-\lambda p^{(2)}\right) V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}).\end{aligned} \tag{5.15}$$

Combining (5.13), (5.14), and (5.15) implies that indeed $A \geq 0$. Next, to show that $D \geq 0$, we distinguish several cases. First, for $x^{(1)} > 0$, $x^{(2)} > 0$, $k^{(1)} = 0$, and $k^{(2)} = 0$, D reduces to

$$-\mu^{(1)} H_n(x^{(1)} - 1, x^{(2)}, k^{*(1)} - 1, k^{*(2)}) + \mu^{(1)} V_n(x^{(1)}, x^{(2)}, k^{*(1)} - 1, k^{*(2)}) \geq 0,$$

which directly follows from Lemma 5.3.2. Next, for $x^{(1)} = 0$, $x^{(2)} > 0$, $k^{(1)} = 0$, and $k^{(2)} = 0$, D reduces to

$$-\mu^{(2)} H_n(x^{(1)}, x^{(2)} - 1, k^{*(1)}, k^{*(2)} - 1) + \mu^{(2)} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)} - 1) \geq 0,$$

which again follows from the fact that $V(\mathbf{x}, \mathbf{k})$ is increasing in all components in $\mathbf{x}$ (Lemma 5.3.2). Then, consider the case where $x^{(1)} > 0$, $x^{(2)} > 0$, $k^{(1)} > 0$, and $k^{(2)} = 0$. D then reduces to

$$\begin{aligned}
&\frac{k^{(1)}\mu^{(1)}}{k^{(1)} + k^{(2)}} H_n(x^{(1)} - 1, x^{(2)}, k^{(1)} - 1, k^{(2)}) \\
&- \frac{k^{(1)}\mu^{(1)}}{k^{(1)} + k^{(2)}} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\
&- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)} + k^{*(2)}} H_n(x^{(1)} - 1, x^{(2)}, k^{*(1)} - 1, k^{*(2)}) \\
&+ \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)} + k^{*(2)}} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}) \\
&= \mu^{(1)} H_n(x^{(1)} - 1, x^{(2)}, k^{(1)} - 1, k^{(2)}) \ - \mu^{(1)} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\
&- \mu^{(1)} H_n(x^{(1)} - 1, x^{(2)}, k^{*(1)} - 1, k^{*(2)}) + \mu^{(1)} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}) \geq 0.
\end{aligned}$$

which again follows from Lemma 5.3.2 by noting that $k^{*(2)} = 0$. Lastly, for $x^{(1)} = 0$, $x^{(2)} > 0$, $k^{(1)} = 0$, and $k^{(2)} > 0$, Lemma 5.3.2 is readily seen to imply that

$$\begin{aligned}
&\frac{k^{(2)}\mu^{(2)}}{k^{(1)} + k^{(2)}} H_n(x^{(1)}, x^{(2)} - 1, k^{(1)}, k^{(2)} - 1) \\
&- \frac{k^{(2)}\mu^{(2)}}{k^{(1)} + k^{(2)}} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)} + k^{*(2)}} H_n(x^{(1)}, x^{(2)} - 1, k^{*(1)}, k^{*(2)} - 1) \\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{(1*)} + k^{*(2)}} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}) \\
&= \mu^{(2)} H_n(x^{(1)}, x^{(2)} - 1, k^{(1)}, k^{(2)} - 1) \ - \mu^{(2)} V_n(x^{(1)}, x^{(2)}, k^{(1)}, k^{(2)}) \\
&+ \mu^{(2)} H_n(x^{(1)}, x^{(2)} - 1, k^{*(1)}, k^{*(2)} - 1) + \mu^{(2)} V_n(x^{(1)}, x^{(2)}, k^{*(1)}, k^{*(2)}) \geq 0,
\end{aligned}$$

since $k^{*(1)} = 0$ because $x^{(1)} = 0$. Finally, for $x^{(1)} > 0$, $x^{(2)} > 0$, $k^{(1)} \geq 0$, and $k^{(2)} > 0$, the result follows from Lemma 5.3.3. This implies $D \geq 0$. This completes the proof of (5.8). Taking the limit of n to infinity completes the proof of Theorem 5.3.1. □

5.4 Numerical experiments

In the previous section, we determined the optimal policy (denoted by π^*) for hyper-exponentially distributed service times. In this section we compare this policy with two commonly used policies, and illustrate the gain in performance through extensive numerical experiments. To this end, we consider the following three static policies: (1) all arriving jobs are served immediately (denoted by π^{PS}); this corresponds to the classical PS queue, (2) only one job with the shortest expected remaining processing time is taken into service (denoted by π^{SEPT}); this corresponds to the classical queue in which only the job with the shortest expected processing time is served first, and (3) the classical M/G/1 FCFS queue (π^{FCFS}). For a policy π we denote by $g(\pi)$ the long-term average number of jobs.

We show the results for a variety of parameter settings. To cover a broad set of the parameter space, we varied the load ρ as 0.50, 0.80 and 0.90, and the squared coefficient of variation c_S^2 from 1 to 16. The service times were taken to be H_2-distributed, parameterized by the triple $(p, \mu^{(1)}, \mu^{(2)})$, such that $p^{(1)} = p$, $p^{(2)} = 1 - p$ in (5.5); the mean service time equals $\beta = p^{(1)}/\mu^{(1)} + p^{(2)}/\mu^{(2)}$, and the load offered to the system is $\rho = \lambda\beta$. The results are shown in Table 5.1 below. The results on π^* in the table are derived by solving the backward recursion equations numerically. Note that for the FCFS model the Pollaczek-Khintchine formula immediately implies that

$$g(\pi^{\mathsf{FCFS}}) = \rho + \frac{\rho^2(1 + c_S^2)}{2(1-\rho)}, \tag{5.16}$$

while for the classical PS model it is known that

$$g(\pi^{\mathsf{PS}}) = \frac{\rho}{1-\rho}. \tag{5.17}$$

Furthermore, for H_2-distributed service times it is readily seen that the SEPT policy coincides with an M/M/1-type priority system with two priority classes, where class-1 has non-preemptive priority over class-2, and where class-i jobs arrive according to a Poisson process with rate $\lambda^{(i)}$ and with service rate $\mu^{(i)}$ ($i = 1, 2$). Assume $\mu^{(1)} \geq \mu^{(2)}$ (without loss of generality), and let $\lambda^{(i)} := \lambda p^{(i)}$ and $\rho^{(i)} := \lambda^{(i)}/\mu^{(i)}$. Then, if $N^{(i)}$ denotes the number of class-i jobs in the system ($i = 1, 2$), it is well-known that

$$g(\pi^{\mathsf{SEPT}}) = \mathbb{E}[N^{(1)}] + \mathbb{E}[N^{(2)}], \tag{5.18}$$

ρ	c_S^2	λ	$p^{(1)}$	$\mu^{(1)}$	$\mu^{(2)}$	$g(\pi^*)$	$g(\pi^{\mathsf{PS}})$	$g(\pi^{\mathsf{SEPT}})$	$g(\pi^{\mathsf{FCFS}})$
0.50	1	0.50	0.500	1.000	1.000	1.00	1.00	1.00	1.00
0.50	4	0.50	0.887	1.775	0.225	0.94	1.00	1.43	1.75
0.50	16	0.50	0.970	1.939	0.061	0.94	1.00	3.41	4.75
0.80	1	0.80	0.500	1.000	1.000	4.00	4.00	4.00	4.00
0.80	4	0.80	0.887	1.775	0.225	3.16	4.00	4.67	8.80
0.80	16	0.80	0.970	1.939	0.061	3.12	4.00	10.97	28.00
0.90	1	0.90	0.500	1.000	1.000	9.00	9.00	9.00	9.00
0.90	4	0.90	0.887	1.775	0.225	6.26	9.00	8.32	21.15
0.90	16	0.90	0.970	1.939	0.061	6.05	9.00	16.83	69.75

Table 5.1: The expected number of jobs in the system for the four policies.

with

$$\begin{aligned} \mathbb{E}[N^{(1)}] &= \rho^{(1)} + \lambda^{(1)} \frac{\rho^{(1)}/\mu^{(1)} + \rho^{(2)}/\mu^{(2)}}{1-\rho^{(1)}}, \\ \mathbb{E}[N^{(2)}] &= \rho^{(2)} + \lambda^{(2)} \frac{\rho^{(1)}/\mu^{(1)} + \rho^{(2)}/\mu^{(2)}}{(1-\rho^{(1)})(1-\rho)}. \end{aligned} \tag{5.19}$$

The results shown in Table 5.1 show that the optimal policy π^*, strongly outperforms the three static policies, particularly when the squared coefficient of variation c_S^2 of the service times increases. Furthermore, we observe that the optimal values of the cost function $g(\pi^*)$ tend to *decrease* when c_S^2 is increased; this is remarkable, since in many queueing systems an increase in the variability in the service times leads to a degradation of the waiting-time and queue-length performance (see also Section 5.5 for a discussion). This observation can be explained by the fact that for H_2-distributed service times with large c_S^2 we have $\mu^{(1)} \gg \mu^{(2)}$ and $p^{(1)} = 1 - p^{(2)} \approx 1$, while the optimal policy gives priority to type-1 jobs (i.e., those with service rate $\mu^{(1)}$) over type-2 jobs (i.e., those with service rate $\mu^{(2)}$). Type-2 jobs only enter service when no type-1 jobs are present, while the number of type-2 jobs in service at any moment in time is at most 1; when type-1 jobs arrive during the service of a type-2 job, then each of these type-1 jobs immediately receives its fair share of the service capacity (in a PS fashion). In this way, the small (i.e., type-1) jobs get some priority over large (i.e., type-2) jobs and, therefore, for H_2-distributed service times with balanced means the expected number of jobs in the system is indeed small when c_S^2 is large.

Another remarkable observation is that for the SEPT policy the cost function $g(\pi^{\mathsf{SEPT}})$ is not monotonically increasing in c_S^2 (in Table 1, see the case where $\rho = 0.90$). In fact, numerical experiments with Equations (5.18)–(5.19) suggest that, for any value of ρ, there is a threshold value $T(\rho)$ such that $g_{c_S^2}(\pi^{\mathsf{SEPT}}) < g_1(\pi^{\mathsf{SEPT}})$ for $1 < c_S^2 < T(\rho)$, while $g_{c_S^2}(\pi^{\mathsf{SEPT}}) > g_1(\pi^{\mathsf{SEPT}})$ for $c_S^2 > T(\rho)$. Also, numerical experimentation with Equations (5.18)–(5.19) suggests that at the cross-over point $c_S^2 = T(\rho)$, we have $\mathbb{E}[N^{(1)}] = \mathbb{E}[N^{(2)}]$ (and by definition $g_{c_S^2}(\pi^{\mathsf{SEPT}}) =$

$g_1(\pi^{\mathsf{SEPT}})$). The derivation of rigorous proofs of such results, which are specific for H_2-distributed service times, and can be obtained directly from (5.18)–(5.19), is left as an exercise to the reader.

We end this section with a number of remarks. In the proof of Theorem 5.3.1 we restricted ourselves to the case of hyper-exponential service-time distributions with two phases. This mainly serves the ease of the discussion. For the case of general phase-type distributions, the proof can be done in a similar way. The decomposition of the value function into A and D (see Equations (5.11) and (5.12)) still holds. The monotonicity properties of A (see Equations (5.13)–(5.15)) are completely identical. The monotonicity of D (defined in Lemma 5.3.3) requires submodularity for a more general expression in the dynamic programming formulation for the services, which can be proven similarly but is notationally cumbersome.

In the formulation of the relative value function it is implicit that the starting phase of each job is known before a job enters service. In case the starting phase is not known, the assignment policy can only be based on the number of jobs waiting in the queue. In this case, one may suspect that the optimal policy is of switching type, i.e., there exists a connected subset S of the state space such that if the system state is in S then assignment is optimal and otherwise waiting is optimal. The method presented can be adopted to prove such a result.

5.5 Conclusion and topics for further research

In this chapter we studied dynamic server-assignment policies for the single queue LPS queueing system with general phase-type distributions, where assignment decisions are based on the number of jobs in each service phase. For this model, we have derived a complete and explicit characterization of the optimal dynamic assignment policy based on increasingness and submodularity properties of the relative value function. We have also provided a numerical assessment of the performance of the optimal dynamic policy compared to several commonly used static policies, and the optimal static policy. The results show that a considerable gain can be obtained by using the dynamic policy.

Finally, we address a number of topics for further research. First, it is interesting to study optimal server-assignment policies for multi-queue extensions of the LPS queue, building upon the insights obtained in this chapter. To start, an extension that is of main interest both from a theoretical and an application point of view is a two queue tandem where the servers can be dynamically assigned to either one of the queues and where at any moment in time the active servers share an underlying resource in a PS fashion. For this model, for which the optimal server-assignment policy is unknown today, it is challenging to investigate whether the optimal server-assignment policy can be characterized using monotonicity properties of the relative value function. These results, in turn, may be generalized to multi queue extensions of the LPS with more than two queues. Moreover, the methodological insight that we obtained by decomposing the relative value function may also be applicable to study monotonicity properties for multi-node extensions. Second, the numerical results suggest that significant improvements can be made by implementing dynamic instead of static server-assignment policies. It would be interesting to derive bounds on the performance gain that can be realized by optimal dynamic assignment strategies. Third, an interesting question raised by the numerical results shown in Table 5.1 is whether the cost for the optimal policy, $g(\pi)$, is increasing as a function of c_S^2. In this context, recall from Section 5.4 that such a monotonicity property generally does not exist for the SEPT policy. Derivation of such monotonicity results, or counter-examples contradicting such properties, are left as a challenging open question for further research. Finally, it is interesting to investigate whether an explicit quantification of the performance gain can be given under heavy traffic assumptions. In heavy traffic, the performance of the system mainly depends on the first and second moments of the arrival and service process in many queueing networks. It is a challenging open question to see whether this is also the case for the LPS queue.

5.6 Appendix: Proof of Lemma 5.3.3

In this section we provide a proof of Lemma 5.3.3, which is put into an appendix, since it is rather technical in nature and notationally cumbersome. The proof is very similar to the one provided for Theorem 5.3.1.

Proof. The proof is by induction on n, where n is defined in Equation (5.7). We focus on the terms that deal with the services only, since the terms for the arrivals are dealt with similarly as in Equation (5.11). To this end, for $k^{(j)} > 1$ and for

$j \in M$, we find by expanding all terms

$$
\begin{aligned}
&\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}\Big(\frac{(k^{(1)}-1)\mu^{(1)}}{k^{(1)}+k^{(2)}-1}H_n'(x^{(1)}-2,x^{(2)},k^{(1)}-2,k^{(2)})\\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}-1}H_n'(x^{(1)}-1,x^{(2)}-1,k^{(1)}-1,k^{(2)}-1)\\
&-\frac{(k^{(1)}-1)\mu^{(1)}}{k^{(1)}+k^{(2)}-1}H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\\
&-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}-1}H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\Big)\\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}\Big(\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}-1}H_n'(x^{(1)}-1,x^{(2)}-1,k^{(1)}-1,k^{(2)}-1)\\
&+\frac{(k^{(2)}-1)\mu^{(2)}}{k^{(1)}+k^{(2)}-1}H_n'(x^{(1)},x^{(2)}-2,k^{(1)},k^{(2)}-2)\\
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}-1}H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\\
&-\frac{(k^{(2)}-1)\mu^{(2)}}{k^{(1)}+k^{(2)}-1}H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\Big)\\
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}\Big(\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\\
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}V_n(x^{(1)},x^{(2)},k^{(1)},k^{(2)})\\
&-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}V_n(x^{(1)},x^{(2)},k^{(1)},k^{(2)})\Big)\\
&-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}\Big(\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\\
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}V_n(x^{(1)},x^{(2)},k^{(1)},k^{(2)})\\
&-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}V_n(x^{(1)},x^{(2)},k^{(1)},k^{(2)})\Big)
\end{aligned}
$$

$$
\begin{aligned}
&- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}\Big(\frac{(k^{*(1)}-1)\mu^{(1)}}{k^{*(1)}+k^{*(2)}-1}H'_n(x^{(1)}-2,x^{(2)},k^{*(1)}-2,k^{*(2)})\\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}-1}H'_n(x^{(1)}-1,x^{(2)}-1,k^{*(1)}-1,k^{*(2)}-1)\\
&- \frac{(k^{*(1)}-1)\mu^{(1)}}{k^{*(1)}+k^{*(2)}-1}H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\\
&- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}-1}H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\Big)\\
&- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}\Big(\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}-1}H'_n(x^{(1)}-1,x^{(2)}-1,k^{*(1)}-1,k^{*(2)}-1)\\
&+ \frac{(k^{*(2)}-1)\mu^{(2)}}{k^{*(1)}+k^{*(2)}-1}H'_n(x^{(1)},x^{(2)}-2,k^{*(1)},k^{*(2)}-2)\\
&- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}-1}H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\\
&- \frac{(k^{*(2)}-1)\mu^{(2)}}{k^{*(1)}+k^{*(2)}-1}H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\Big)\\
&+ \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}\Big(\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\\
&- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}V_n(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)})\\
&- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}V_n(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)})\Big)\\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}\Big(\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\\
&+ \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\\
&- \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}V_n(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)})\\
&- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}V_n(x^{(1)},x^{(2)},k^{*(1)},k^{*(2)})\Big).
\end{aligned}
$$

In this expression, the H'_n-terms denote the expression that follows from taking the action after evaluating H_n (whereas the H_n-terms denote the expression that follows from taking the action after evaluating V_n). Note that by induction, starting with $k^{(2)} > 0$, results in H_n' being equal to H_n, because the optimal decisions are the same. By applying Lemma 5.3.3 the expression above can be decomposed into the four expressions (5.20)–(5.23) below, for each of which we show that they are non-negative.

For the first expression, we have

$$\begin{aligned}
&- \frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}} \frac{k^{(2)}\mu^{(1)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)} \\
&\quad \times H_n(x^{(1)}-2, x^{(2)}, k^{(1)}-2, k^{(2)}) \\
&+ \frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}} \frac{k^{(1)}\mu^{(1)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)} \\
&\quad \times H_n(x^{(1)}-1, x^{(2)}-1, k^{(1)}-1, k^{(2)}-1) \\
&+ \frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}} \frac{k^{*(2)}\mu^{(1)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)} \\
&\quad \times H_n(x^{(1)}-2, x^{(2)}, k^{*(1)}-2, k^{*(2)}) \\
&- \frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}} \frac{k^{*(1)}\mu^{(1)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)} \\
&\quad \times H_n(x^{(1)}-1, x^{(2)}-1, k^{*(1)}-1, k^{*(2)}-1) \\
&\geq 0,
\end{aligned} \tag{5.20}$$

because $\mathbf{k}^*$ equals $\mathbf{k}$ for $x^{(1)} > 0$, $x^{(2)} > 0$, $k^{(1)} \geq 0$, and $k^{(2)} > 0$. Since $k^{(2)} > 0$, it follows from Theorem 5.3.1 that $k^{*(2)} = k^{(2)}$, and the action taken in H is then to serve all jobs of type 1, thus $k^{(1)} = x^{(1)}$, and similarly $k^{*(1)} = k^{(1)}$.

For the next expression we find

$$\begin{aligned}
&-\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}\frac{k^{(2)}\mu^{(2)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)} \\
&\quad\times H_n(x^{(1)}-1,x^{(2)}-1,k^{(1)}-1,k^{(2)}-1) \\
&+\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}\frac{k^{(1)}\mu^{(2)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)} \\
&\quad\times H_n(x^{(1)},x^{(2)}-2,k^{(1)},k^{(2)}-2) \\
&+\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(2)}\mu^{(2)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)} \\
&\quad\times H_n(x^{(1)}-1,x^{(2)}-1,k^{*(1)}-1,k^{*(2)}-1) \\
&-\frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(1)}\mu^{(2)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)} \\
&\quad\times H_n(x^{(1)},x^{(2)}-2,k^{*(1)},k^{*(2)}-2) \\
&\geq 0,
\end{aligned} \tag{5.21}$$

using similar arguments as before. Similarly, the third expression leads to

$$
\begin{aligned}
&\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}\frac{k^{(1)}\mu^{(1)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)}\\
&\quad\times H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\\
&-\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}\frac{k^{(2)}\mu^{(2)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)}\\
&\quad\times H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\\
&-\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(1)}\mu^{(1)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)}\\
&\quad\times H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\\
&+\frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(2)}\mu^{(2)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)}\\
&\quad\times H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\\
&\geq 0,
\end{aligned}
\tag{5.22}
$$

and the final expression

$$\begin{aligned}
&\frac{k^{(1)}\mu^{(1)}}{k^{(1)}+k^{(2)}}\frac{k^{(2)}\mu^{(2)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)}\\
&\times H_n(x^{(1)}-1,x^{(2)},k^{(1)}-1,k^{(2)})\\
-&\frac{k^{(2)}\mu^{(2)}}{k^{(1)}+k^{(2)}}\frac{k^{(1)}\mu^{(2)}}{(k^{(1)}+k^{(2)})(k^{(1)}+k^{(2)}-1)}\\
&\times H_n(x^{(1)},x^{(2)}-1,k^{(1)},k^{(2)}-1)\\
-&\frac{k^{*(1)}\mu^{(1)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(2)}\mu^{(2)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)}\\
&\times H_n(x^{(1)}-1,x^{(2)},k^{*(1)}-1,k^{*(2)})\\
+&\frac{k^{*(2)}\mu^{(2)}}{k^{*(1)}+k^{*(2)}}\frac{k^{*(1)}\mu^{(2)}}{(k^{*(1)}+k^{*(2)})(k^{*(1)}+k^{*(2)}-1)}\\
&\times H_n(x^{(1)},x^{(2)}-1,k^{*(1)},k^{*(2)}-1)\\
\geq&\,0.
\end{aligned} \tag{5.23}$$

For $k^{(2)}=1$ the proof follows immediately, since in Equation (5.21) and (5.23), the terms with $x^{(2)}-2$ are zero. Similarly for $k^{(1)}=1$, in Equation (5.20) and (5.22) the terms with $x^{(1)}-2$ are zero, and again the inequalities are satisfied. This completes the proof. $\square$

CHAPTER 6

OPTIMAL SERVER ASSIGNMENT IN WEB SERVERS

In this chapter we derive new dynamic thread-assignment policies that minimize the average response time of Web servers. A Web server is modeled as a two-layered tandem of multi-threading queues, where the active threads compete for access to a common resource. This type of two-layered queueing models naturally occur in the performance modeling of systems with intensive software-hardware interaction. Our results show that the optimal dynamic thread-assignment policies yield strong reductions in the response times. Validation on an Apache Web server shows that our dynamic thread policies confirm our analytical results. This chapter is published as [113].

6.1 Introduction

The rise of Internet and broadband communication technology have boosted the use of Web-based services that combine and integrate information from geographically distributed information systems. As a consequence, popular Web sites are expected to handle huge numbers of requests simultaneously without noticeable degradation of the response-time performance. Moreover, Web servers must perform significant CPU- and disk I/O-intensive processing, caused by the emergence of server-side scripting technologies (e.g., Java servlets, Active Server Pages, PHP). Furthermore,

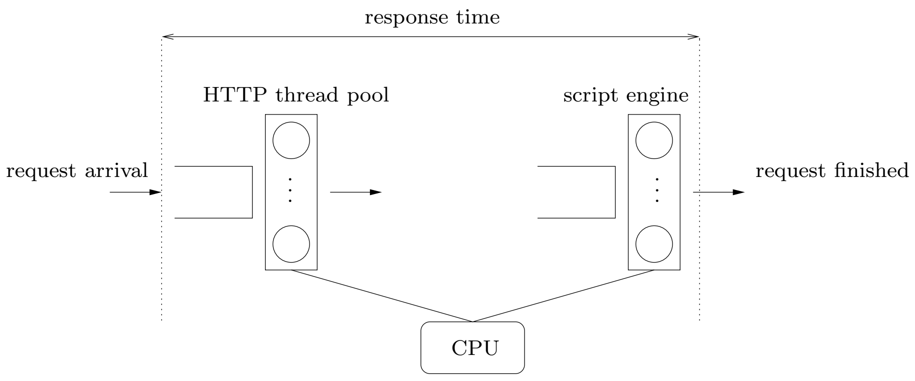


Figure 6.1: The Web server model in [51].

Web pages involving recent and personalized information (location information, headline news, hotel reservations) are created dynamically on-the-fly and hence are not cacheable. This limits the effectiveness of caching infrastructures that are usually implemented to boost the response-time performance of commercial Web sites and limits the bandwidth consumption. At the same time, as a result of the recent advances in wired networking technology, there is usually ample core network bandwidth available at reasonable prices. As a consequence of these developments, Web servers tend to become performance bottlenecks in many cases. These observations raise the need for Web-based service providers to control the performance of their Web servers.

Web servers are typically equipped with a pool of threads. In many cases, a request is composed of a number of processing steps that are performed in sequential order. For example, see Figure 6.1, an HTTP GET request may require processing in several steps: a document-retrieval step and a sequence of script-processing steps to create dynamic content. Similarly, an HTTP POST request may require a document-processing step and several database update queries. To handle the incoming requests, Web servers usually implement a number of thread pools that are dedicated to perform a specific processing step [51, 77].

The performance of the Web server largely depends on the thread-management policy [51]. This policy may be either static (i.e., with a fixed number of threads – possibly of different types) or dynamic (i.e., where threads may be created or killed depending on the state of the server). Traditionally, many Web servers implement a simple static thread-assignment policy, where the size of the thread pool (i.e., the maximum number of threads that can simultaneously execute processing steps) is a configurable system parameter. This leads to a trade-off regarding the proper dimensioning of thread pools to optimize performance: on the one hand, assigning too few threads may lead to relative starvation of processing power, creating a performance bottleneck that may increase the average response time of requests, particularly when the workload increases. On the other hand, if the total number of threads running on a single hardware component is too large, performance degradation may occur due to superfluous context switching overhead and memory or disk I/O activity. Nowadays, more efficient thread policies are widely implemented. In order to effectively react to sudden bursts of transaction requests, many Web servers implement simple dynamic thread-management algorithms that allow threads to be created or killed, depending on the actual number of active threads. However, even though the implementation of these thread policies is common practice, a thorough understanding of the implications of the proper choice of thread-assignment policies and the settings of the parameters on the performance of the Web server is mostly lacking. In particular, the trade-off between relative starvation of processing power in the case of too few threads and the performance degradation in the case of too many threads is not fully understood (see [45] for recent results on software bottlenecks). Moreover, the commonly used thread policies do not take into account the probability distribution of the service times required by the different requests, while significant performance improvements can be obtained by doing so.

A key feature of multi-threaded Web servers is that the threads typically share a common hardware (e.g., a CPU and disk) with a limited amount of capacity. This naturally leads to the formulation of a two-layered tandem of multi-server queues, where the active threads share the processor capacity in a processing sharing (PS) fashion; i.e., when there are k threads active at some moment in time, then each of these k threads receives a fair share $1/k$ of the total processor capacity [79]. In this model, transaction requests are represented by customers, threads are represented by servers, and response times are represented by the sojourn times of the customers. To identify optimal thread-assignment policies, we describe the evolution of the system as a Markov decision model and derive optimal thread policies from the properties of the relative value function. In doing so, we show that the structure of the optimal thread policy strongly depends on the service-time distributions of the different processing steps in the Web server; in practice, these distributions can be monitored and updated on-the-fly.

An interesting characteristic of this model is that it has a two-layered structure, modeling the complex *interaction* between contention at the *hardware* (CPU, disk, memory) layer and the *software entities* (threads) layer. At the software layer, the processing steps, comprising a request, are processed by different (say N) types of threads. However, the active threads effectively share the underlying resource: the more threads are active, the smaller the processor capacity is assigned to each thread. In this way, the thread is no longer an autonomous entity operating at a fixed rate; instead, the processing rate of each thread continuously changes over time. Evidently, for $N = 1$, the model coincides with the classical processor-sharing discipline; but for $N > 1$, the processing speed of one thread pool depends on the state of the other thread pools. This type of interaction makes the model rather complicated, and highly challenging from a methodological point of view.

Although the theory of job scheduling with autonomous independent servers is well-matured, in the literature only a few papers deal with scheduling of Web servers. Harchol-Balter et al. [50, 8, 49] and Crovella et al. [31] study scheduling policies for Web servers to improve the response time performance of Web servers with static Web pages, provided the size of a Web page is known *a-priori*; for this type of models, the results show that the classical Shortest Remaining Processing Time (SRPT) policy is very effective [105]. In contrast to the present chapter, it should be noted that the results in [50] are based on the assumption that the network interface, rather than the Web server itself, is the performance bottleneck; this leads to fundamentally different performance models than the one considered in this chapter. In this context, the contribution of the present chapter complements the results obtained in the above references. Menascé [80] gives an overview of issues involved in modeling Web servers. Cao et al. [23] propose to model a Web server by a simple M/G/1/K/ PS queue, and validate the model through lab experiments. Detailed performance models for Web servers, explicitly including the interaction between software and hardware contention, were proposed in [79, 51]; these modeling efforts naturally led to the formulation of two-layered queueing models.

In this chapter we model a Web server by a two-layered queueing network with a single processor-shared resource. We describe the evolution of the system as a Markov decision process from which we obtain simple and readily implementable dynamic thread-assignment policies that minimize the expected response time of the requests. The service-time distributions are modeled by the class of phase-type distributions, which is a broad class of distributions and also allows to study the impact of heavy-tailed distributions on the expected response time of the requests. The results show not only *that*, but also *how* the optimal policy depends on the service-time distributions at each of the processing steps. The proposed policy uses monitored information on both the number of active threads and the probability distribution of the required service time per request. Our results show that the optimal dynamic thread-assignment policies yield strong reductions in the response times. To validate the model, we have tested the performance of our policies in an experimental setting on an Apache Web server. The experimental results show that our policies indeed lead to significant reductions of the response time, which demonstrates the practical usefulness of the results.

The remainder of this chapter is organized as follows. In Section 6.2 we formulate the model. In Section 6.3 we derive optimal dynamic thread-assignment policies. In Section 6.4 we consider numerical experiments and evaluate them on an Apache Web server. In Section 6.5 we discuss the model assumptions and the computational complexity of the thread-assignment policies. We conclude in Section 6.6 and give ideas for further research directions.

6.2 Model description

In this section we model the problem of dynamic thread assignment in the context of a multi-layered queueing system with a shared PS resource. For this purpose, consider a network of N queues in tandem with a common shared processor for serving arriving requests. Requests arrive according to a Poisson process with rate λ to the first queue. At each queue, threads can be spawned which may be assigned to a request. When a request is assigned to a thread at queue i, it receives service S_i with mean duration β_i for $i = 1, \dots, N$. However, during service, the request only gets a fraction of the total capacity of the server, depending on the number of outstanding threads k_i at each queue i for $i = 1, \dots, N$. Upon completion of service, the thread is terminated and the request proceeds to queue $i + 1$ if $i < N$, or it leaves the system otherwise. If a request is not assigned a thread, the request joins an infinite buffer at the queue and waits until it is assigned a thread. Note that we do not explicitly model delays due to context switching between threads, since the CPU time in comparison to the processing times of the threads is negligible (see Remark 6.4.1 for a justification). The load on the system is given by $\rho = \lambda(\beta_1 + \cdots + \beta_N)$. This model is illustrated in Figure 6.2.

To obtain optimal thread-assignment policies that minimize the expected response times, we model the system in the framework of Markov decision theory. To this end, we model the service-time distribution of S_i by a phase-type distribution with $M_i + 1$ states (where state $M_i + 1$ is the absorbing state), with initial distribution $\eta_i = (\eta_i^{(1)}, \dots, \eta_i^{(M_i)})$, where $\eta_i^{(j)}$ is the probability that the Markov chain starts in state j for $j = 1, \dots, M_i$. When the Markov chain is in state j, the time that the process spends in state j has an exponential distribution with parameter $\mu_i^{(j)}$. Upon leaving state j, the process jumps to state l with probability $p_i^{(j,l)}$, or jumps to the absorbing state $M_i + 1$ with probability $p_i^{(j,M_i+1)}$, where $\sum_{l=1}^{M_i+1} p_i^{(j,l)} = 1$. The absorbing state corresponds to a completion of the service requirement at queue i.

Phase-type distributions have the important feature that they are dense in the class of all non-negative distributions, retaining their tractability [102]. Therefore, it is possible to model heavy-tailed distributions by phase-type distributions. This is especially relevant, since it has been observed that file sizes on Web servers follow a heavy-tailed distribution (see, e.g., [31]). It is common to fit phase-type distributions on the mean $\mathbb{E}S_i = \beta_i$ and on the coefficient of variation c_{S_i} (see, e.g., [110]), or by the more complex EM-algorithm (see [5]).

Next, we uniformize the system (see Section 11.5 of [93]). For simplicity we assume that $\lambda + \max\{\mu_1^{(1)}, \dots, \mu_N^{(M_N)}\} = 1$; we can always get this by scaling. Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the rates to be transition probabilities. Note that rate costs in this case are equivalent to lump costs at each epoch.

Let $\mathbf{x}$ be the vector that denotes the number of requests in each phase, i.e., $x_i^{(j)}$ is the number of requests in phase j that are waiting at queue i plus the number of requests in phase j in service at queue i for $i = 1, \dots, N$. Moreover, let the vector $\mathbf{k}$ denote the number of outstanding threads, i.e, $k_i^{(j)}$ is the number of outstanding threads for requests in phase j at queue i. Thus, the number of outstanding threads at queue i equals $k_i = k_i^{(1)} + \cdots + k_i^{(M_i)}$. A state x of the system, depicted in Figure 6.2, is then given by the tuple $(\mathbf{x}, \mathbf{k})$.

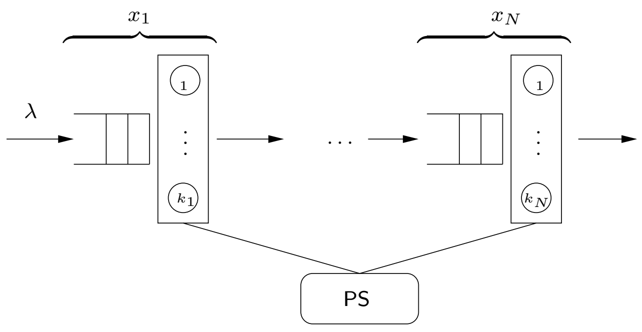


Figure 6.2: A two-layered tandem queue with a shared resource.

Let $\mathbb{E}W$ be the expected response time of an arbitrary customer in the system. The goal is to find a policy π^* that minimizes $\mathbb{E}W$. By using Little's Law [110] this objective translates to minimizing the expected number of requests in the system. Therefore, we assume that the system is subject to unit costs for holding a request per unit of time in the system. Let $u_t(x)$ denote the total expected costs up to time t when the system starts in state x. Note that the underlying Markov chain representing the state of the system satisfies the unichain condition, so that the average expected cost $g = \lim_{t\to\infty} u_t(x)/t$, i.e., the average number of requests in the system, is independent of the initial state x (Proposition 8.2.1 of [93]). The expected response time $\mathbb{E}W$ can then be expressed as g/λ.

A policy π maps the state $(\mathbf{x}, \mathbf{k})$ to an admissible action $\mathbf{a}$, which represents the number of additional threads to be spawned in state $(\mathbf{x}, \mathbf{k})$. Thus, the information that one uses to derive optimal actions generally depends on the number of requests in each phase at each of the queues as well as the number of outstanding threads for each request in each phase.

Let $V(\mathbf{x}, \mathbf{k})$ be a real-valued function defined on the state space. This function will play the role of the relative value function, i.e., the asymptotic difference in total costs that results from starting the process in state $(\mathbf{x}, \mathbf{k})$ instead of some reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) $g + V = TV$, where T is the dynamic programming operator acting on V defined as follows

$$\begin{aligned}TV(\mathbf{x},\mathbf{k}) =&\sum_{i=1}^{N}\sum_{j=1}^{M_i}x_i^{(j)} + \lambda\sum_{j=1}^{M_1}\eta_1^{(j)}H\big(\mathbf{x}+\mathbf{e}_1^{(j)},\mathbf{k}\big)\\
&+\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{l=1}^{M_i}\frac{k_i^{(j)}\mu_i^{(j)}p_i^{(j,l)}}{k_1+\cdots+k_N}H\big(\mathbf{x}-\mathbf{e}_i^{(j)}+\mathbf{e}_i^{(l)},\mathbf{k}-\mathbf{e}_i^{(j)}+\mathbf{e}_i^{(l)}\big)\\
&+\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{l=1}^{M_i}\frac{k_i^{(j)}\mu_i^{(j)}p_i^{(j,M_i+1)}\eta_{i+1}^{(l)}}{k_1+\cdots+k_N}H\big(\mathbf{x}-\mathbf{e}_i^{(j)}+\mathbf{e}_{i+1}^{(l)},\mathbf{k}-\mathbf{e}_i^{(j)}\big)\\
&+\left[1-\lambda-\sum_{i=1}^{N}\sum_{j=1}^{M_i}\frac{k_i^{(j)}\mu_i^{(j)}}{k_1+\cdots+k_N}\right]V(\mathbf{x},\mathbf{k}),\end{aligned}\tag{6.1}$$

with $\mathbf{e}_i^{(j)}$ the unit vector with all entries zero, except for the i,j-th entry for $i=1,\dots,N$ and $j=1,\dots,M_i$, and with $\mathbf{e}_{N+1}^{(j)}$ the zero vector. The unit vector $\mathbf{e}_i^{(j)}$ is similarly defined. The actions, denoted by H, are given by

$$H(\mathbf{x},\mathbf{k}) = \min_{0\le a_i^{(j)}\le x_i^{(j)}-k_i^{(j)}}\left\{V\big(\mathbf{x},\mathbf{k}+\sum_i\sum_j a_i^{(j)}\mathbf{e}_i^{(j)}\big)\;\middle|\;\begin{array}{l}i=1,\dots,N\\ j=1,\dots,M_i\end{array}\right\},$$

with $a_i^{(j)}\in\mathbb{N}_0$, where $\mathbb{N}_0=\{0,1,2,\dots\}$. $V\big(\mathbf{x},\mathbf{k}+a_i^{(j)}\,\mathbf{e}_i^{(j)}\big)$ represents the action that spawns $a_j^{(i)}$ additional threads at queue i for jobs in phase j. From the definition of H, we see that it is not possible to spawn more threads than there are requests waiting. Spawning more threads than requests waiting is obviously not optimal, since that leads to loss of capacity in the model.

The first term in the expression $TV(\mathbf{x},\mathbf{k})$ models the direct costs, i.e., the total number of requests in the system. The second term models the arrivals, which occur with rate $\lambda\eta_1^{(j)}$ to phase j at the first queue. The transitions of a request to a different phase within each queue are given by the third term. A transition from phase j to phase l for a request in queue i occurs with rate $\mu_i^{(j)}p_i^{(j,l)}$, but this is adjusted with the factor $k_i^{(j)}/\big(\mathbf{k}_1+\cdots+\mathbf{k}_N\big)$, since that request only uses a fair share of the service capacity. Note that the system is specifically modeled such that when a request moves from one phase to another, the previously assigned thread is not lost. The thread is only released upon completion of the service requirement. The next term, which accounts for a transition to the absorbing state, is similarly explained with the exception that the departure is split into arrivals to phase l of the next queue with probability $\eta_{i+1}^{(l)}$. The last term is the uniformization constant to account for the dummy transitions added to the model. Note that the last term in Equation (6.1) is non-negative.

The optimality equation $g + V = TV$ is hard to solve analytically in practice. Alternatively, the optimal actions can also be obtained by recursively defining $V_{l+1} = TV_l$ for arbitrary V_0. For $l \to \infty$, the maximizing actions converge to the optimal ones (for existence and convergence of solutions and optimal policies we refer to [93]). Consequently, when V is known, we can restrict our attention to the function H to obtain the optimal actions. In Section 6.4 we adopt this approach to compute optimal policies.

6.3 Dynamic thread management

In this section we focus on dynamic thread assignment. We determine, using dynamic programming, optimal policies minimizing the expected response time per request. The performance of the optimal policies is compared to the performance of policies that only serve requests based on the number of threads outstanding. A specific example of the latter case is the policy that serves one request with only one outstanding thread until it leaves the system, resulting in a First Come First Served (FCFS) policy. The other extreme is the policy that always serves all requests so that new threads are spawned for arriving requests. The intermediate case, which is commonly implemented in Web servers, is the policy that serves requests simultaneously with a number of threads of which the maximum number is limited by some specified number. More precisely, let $\pi^{(k)}$ be the policy that spawns at most k threads in total such that the requests in queue i get priority over requests in queue j when $i \geq j$. Note that the three policies mentioned earlier are represented by $\pi^{(1)}$, $\pi^{(\infty)}$, and $\pi^{(k)}$ for $k = 2, 3, \ldots$, respectively.

Exponential service-time distributions are a special case of phase-type distributions, namely those with one phase only. In this case, the optimal policy that minimizes the expected response time is to serve according to policy $\pi^{(1)}$, i.e., serve one request with only one outstanding thread until it leaves the system, such that the requests in queue i get priority over requests in queue j when $i > j$. This result also holds when the service-time distributions are replaced with Erlang distributions, since the requests are not preempted during service. For Erlang distributed service times a request keeps its thread jumping from one phase to its next phase in the same queue. This class of distributions models the situation where a Web server fetches a Web page and also performs server-side scripting for the page.

The optimal policy changes when the service-time distributions are replaced with hyper-exponential distributions. In this case, requests waiting in queue i may have a smaller expected sojourn time compared to a request in queue $j > i$ when a thread is opened at queue i. For this reason, the optimal policy can be obtained efficiently by recursive computation. As will follow from Theorem 6.3.1, this policy coincides with the optimal policies for exponential and Erlang service-time distributions. For these distributions there is only one service phase, so that the expected sojourn time decreases monotonically as a request progresses to the last queue. Therefore, there is only one thread outstanding at most.

In the previous paragraphs, we have obtained intuition for optimal policies for dynamic thread management. We have seen that the expected sojourn time of requests plays a key role in the decision to spawn threads. In the case of exponential and Erlang service-time distributions it was not possible to obtain a smaller expected sojourn time than the expected sojourn time of a request further in the system when spawning additional threads, and therefore the FCFS policy is optimal. In the case of hyper-exponential service-time distributions, we obtained that the optimal actions at queue i depend on the state of queues $j \geq i$ and the decision rules for those queues. This result also holds for the general phase-type service-time distributions.

Theorem 6.3.1. *Let $\varphi_i(\mathbf{x}, \mathbf{k})$ denote the decision rule that describes the thread-management rule at queue i, for $i = 1, \ldots, N$. Let φ_i be such that it spawns $a_i^{(j)}$ threads for requests in phase j at queue i given state $(\mathbf{x}, \mathbf{k})$, if the expected sojourn time of the $a_i^{(j)}$ phase-j requests become smaller than the expected sojourn time of at least one request in queue $j \geq i$ under decision rules $\varphi_{i+1}, \ldots, \varphi_N$. Then policy $\pi = (\varphi_1, \ldots, \varphi_N)$ is optimal.*

The decision rule can by applied by recursively solving the expected sojourn time of the requests, starting with the last queue, denoted by N. Given the expected sojourn time of the requests in this queue we decide if additional threads are spawned to the requests in this queue. For the calculation of the expected sojourn time arriving requests are not taken into account. Then, we move to the previous queue, $N - 1$. We calculate the expected sojourn time for this queue, given the action taken in queue N. Continuing this recursion the optimal policy will be defined and can be applied.

To prove Theorem 6.3.1, submodularity of the relative value function is required. This property is formulated and proven in Lemma 6.3.3. To avoid cumbersome notation we present the proof for exponentially distributed service times. For exponentially distributed service times the theorem can be stated as follows

Theorem 6.3.2. *Consider an empty system, then $\mathbf{k}^* = \mathbf{0}$. Now, consider a non-empty system with $\mathbf{k} = \mathbf{0}$, and define $z := \max\{\, l \,|\, x_l > 0\}$, then $\mathbf{k}^* = \mathbf{e}_z$. Finally, consider a non-empty system with open threads, and define $j := \max\{\, l \,|\, k_l > 0\}$. Then*

$$\mathbf{k}^* := \mathbf{k} + \sum_{i=j}^{N} a_i e_i, \tag{6.2}$$

with the j^{th} entry

$$\begin{cases} a_j = 0, & \textit{if } z = j \textit{ and } k_j = 1, \\ a_j = x_j - k_j, & \textit{if } z = j \textit{ and } k_j > 1, \textit{ or } z \neq j, \end{cases} \tag{6.3}$$

and with the other entries

$$a_i = x_i - k_i, \textit{ for } i = j+1, \ldots, N, \tag{6.4}$$

where $\mathbf{k}^* = (k_1^*, \ldots, k_N^*)$*. Then, for arbitrary* $(\mathbf{x}, \mathbf{k})$*, we have*

$$V(\mathbf{x}, \mathbf{k}) \geq V(\mathbf{x}, \mathbf{k}^*). \tag{6.5}$$

Proof. Let $\mathbf{e}_{N+1} = 0$. For exponentially distributed service times, the backward recursion equation following from (6.1) reduces to

$$\begin{aligned} V_n(\mathbf{x}, \mathbf{k}) = {} & \lambda H_{n-1}(\mathbf{x} + \mathbf{e}_1, \mathbf{k}) + \sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j} H_{n-1}(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k} - \mathbf{e}_i) \\ & + \left[1 - \lambda - \sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j}\right] V_{n-1}(\mathbf{x}, \mathbf{k}) + \sum_{i=1}^{N} x_i, \end{aligned} \tag{6.6}$$

with

$$H_n(\mathbf{x}, \mathbf{k}) = \min\{V_n(\mathbf{x}, \mathbf{k} + \sum_{i=1}^{N} a_i \mathbf{e}_i) |\ i = 1, \ldots, N,\ 0 \leq a_i \leq x_i - k_i, a_i \in \mathbb{N}\},$$

for $n = 0, 1, \ldots$ and $(\mathbf{x}, \mathbf{k}) \in \mathbb{N}^N \times \mathbb{N}^N$, and with $V_0(\mathbf{x}, \mathbf{k}) \equiv 0$. Clearly,

$$V_0(\mathbf{x}, \mathbf{k}) - V_0(\mathbf{x}, \mathbf{k}^*) \geq 0. \tag{6.7}$$

We assume that the Equation (6.8) holds for n and then prove that it holds for $n + 1$.

$$V_n(\mathbf{x}, \mathbf{k}) - V_n(\mathbf{x}, \mathbf{k}^*) \geq 0, \tag{6.8}$$

where k_i^* is defined in Equation (6.2). We can write, using (6.6)

$$\begin{aligned} V_{n+1}(\mathbf{x}, \mathbf{k}) - V_{n+1}(\mathbf{x}, \mathbf{k}^*) = {} & \lambda[H_n(\mathbf{x} + \mathbf{e}_1, \mathbf{k}) - H_n(\mathbf{x} + \mathbf{e}_1, \mathbf{k}^*)] \\ & + \sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j} H_n(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k} - \mathbf{e}_i) \\ & - \sum_{i=1}^{N} \frac{\mu_i k_i^*}{\sum_{j=1}^{N} k_j^*} H_n(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k}^* - \mathbf{e}_i) \\ & + (1 - \lambda)[V_n(\mathbf{x}, \mathbf{k}) - V_n(\mathbf{x}, \mathbf{k}^*)] \\ & - \sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j} V_n(\mathbf{x}, \mathbf{k}) + \sum_{i=1}^{N} \frac{\mu_i k_i^*}{\sum_{j=1}^{N} k_j^*} V_n(\mathbf{x}, \mathbf{k}^*). \end{aligned} \tag{6.9}$$

Expression (6.9) above consists of terms related to the arrivals and related to the departures. We will now show that Equation (6.8) indeed holds. To this end, we use induction in n to show that for $n > 0$

$$\lambda[H_n(\mathbf{x} + \mathbf{e}_1, \mathbf{k}) - H_n(\mathbf{x} + \mathbf{e}_1, \mathbf{k}^*)] \geq 0, \tag{6.10}$$

and

$$V_n(\mathbf{x},\mathbf{k}) - V_n(\mathbf{x},\mathbf{k}^*) \geq 0, \tag{6.11}$$

holds. Combining (6.10) and (6.10) we have that

$$\begin{aligned}&\lambda[H_n(\mathbf{x}+\mathbf{e}_1,\mathbf{k}) - H_n(\mathbf{x}+\mathbf{e}_1,\mathbf{k}^*)]\\ &+ (1-\lambda)[V_n(\mathbf{x},\mathbf{k}) - V_n(\mathbf{x},\mathbf{k}^*)] \geq 0.\end{aligned} \tag{6.12}$$

The terms in Equation (6.12) are the so-called arrival terms. The remaining terms, the so-called service related terms, are exactly equal to the terms of the expression in Lemma 6.3.3 given below. This shows the inequality in (6.8). By taking the limit of n to infinity the proof of Theorem 6.3.2 is completed. □

Lemma 6.3.3 (Submodularity)**.** *Let k^* be defined in* (6.2)*. Given the backwards recursion equation in* (6.6)*, for $x_i > 0$, $k_i \geq 0$, for $i = 1,\dots,N$, it holds for all $n \geq 0$ that*

$$\begin{aligned}&\sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j} H_{n+1}(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}-\mathbf{e}_i)\\ &-\sum_{i=1}^{N} \frac{\mu_i k_i^*}{\sum_{j=1}^{N} k_j^*} H_{n+1}(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}^*-\mathbf{e}_i)\\ &-\sum_{i=1}^{N} \frac{\mu_i k_i}{\sum_{j=1}^{N} k_j} V_{n+1}(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}-\mathbf{e}_i)\\ &+\sum_{i=1}^{N} \frac{\mu_i k_i^*}{\sum_{j=1}^{N} k_j^*} V_{n+1}(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}^*-\mathbf{e}_i)] \geq 0.\end{aligned}$$

Proof. The proof is by induction on n. In this proof the focus is on the terms that deal with the services only, since the terms dealing with arrivals are similar as in Equation (6.12). Assume, without loss of generality, that $\mu_1 \geq \cdots \geq \mu_N$. The queues can always be ordered in the way, due to the exponentially distributed service times and the non-preemptive service. Then, the proof of Lemma 6.3.3 follows from applying the decomposition approach in Section 5.3.

In the following equation H' is the expression that follows from taking the action after evaluating H. Due to the induction hypothesis and starting with $k_i > 0$ the optimal decisions are the same and thus $H = H'$. $\mathbf{k}'^*$ follows after taking the optimal action in a state with $\mathbf{k}^*$, which was already optimal. Induction on n gives

$$
\begin{aligned}
&\frac{k_i\mu_i}{\sum_{s=1}^N k_s}\Big(\frac{\sum_{j=1}^N k_j^*\mu_j}{\sum_{t=1}^N k_t^*}H_n'(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1}-\mathbf{e}_j+\mathbf{e}_{j+1},\mathbf{k}^*-\mathbf{e}_i-\mathbf{e}_j)\\
&\qquad -\frac{\sum_{j=1}^N k_j^*\mu_j}{\sum_{t=1}^N k_t^*}H_{n-1}(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}^*-\mathbf{e}_i)\Big)\\
&+\frac{k_i^*\mu_i}{\sum_{s=1}^N k_s^*}\Big(\frac{\sum_{j=1}^N k_j^{*\prime}\mu_j}{\sum_{t=1}^N k_t^{*\prime}}H_n'(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1}-\mathbf{e}_j+\mathbf{e}_{j+1},\mathbf{k}^{*\prime}-\mathbf{e}_i-\mathbf{e}_j)\\
&\qquad -\frac{\sum_{j=1}^N k_j^{*\prime}\mu_j}{\sum_{t=1}^N k_t^{*\prime}}H_n(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+1},\mathbf{k}^{*\prime}-\mathbf{e}_i)\Big)\\
&-\frac{k_i\mu_i}{\sum_{s=1}^N k_s}\Big(\frac{\sum_{j=1}^N k_j^*\mu_j}{\sum_{t=1}^N k_t^*}H_n(\mathbf{x}-\mathbf{e}_j+\mathbf{e}_{j+1},\mathbf{k}^*-\mathbf{e}_j)\\
&\qquad -\frac{\sum_{j=1}^N k_j^*\mu_j}{\sum_{t=1}^N k_t^*}V_n(\mathbf{x},\mathbf{k}^*)\Big)\\
&+\frac{k_i^*\mu_i}{\sum_{s=1}^N k_s^{*\prime}}\Big(\frac{\sum_{j=1}^N k_j^{*\prime}\mu_j}{\sum_{t=1}^N k_t^{*\prime}}H_n(\mathbf{x}-\mathbf{e}_j+\mathbf{e}_{j+1},\mathbf{k}^{*\prime}-\mathbf{e}_j)\\
&\qquad -\frac{\sum_{j=1}^N k_j^{*\prime}\mu_j}{\sum_{t=1}^N k_t^{*\prime}}V_{n-1}(\mathbf{x},\mathbf{k}^{*\prime})\Big).
\end{aligned}
$$

Applying the induction hypothesis on the expression above decomposes into the following two expressions (6.13) and (6.14).

$$
\begin{aligned}
&-\frac{\mu_i}{\sum_{s=1}^N k_s^{*\prime}}H_n(\mathbf{x}+2(\mathbf{e}_{i+1}-\mathbf{e}_i),\mathbf{k}^{*\prime}-2\mathbf{e}_i)\\
&+\frac{\mu_{i+1}}{\sum_{s=1}^N k_s^{*\prime}}H_n(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+2},\mathbf{k}^{*\prime}-\mathbf{e}_i-\mathbf{e}_{i+1})\\
&+\frac{\mu_i}{\sum_{s=1}^N k_s^{*}}H_n(\mathbf{x}+2(\mathbf{e}_{i+1}-\mathbf{e}_i),\mathbf{k}^{*}-2\mathbf{e}_i) \qquad (6.13)\\
&+\frac{\mu_{i+1}}{\sum_{s=1}^N k_s^{*}}H_n(\mathbf{x}-\mathbf{e}_i+\mathbf{e}_{i+2},\mathbf{k}^{*}-\mathbf{e}_i-\mathbf{e}_{i+1})\\
&\geq 0,
\end{aligned}
$$

because the μ_i's are ordered, and similarly,

$$
\begin{aligned}
&- \frac{\mu_i}{\sum_{s=1}^N k_s^{*\prime}} H_n(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k}^{*\prime} - \mathbf{e}_i) \\
&+ \frac{\mu_{i+1}}{\sum_{s=1}^N k_s^{*\prime}} H_n(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k}^{*\prime} - \mathbf{e}_i) \\
&+ \frac{\mu_i}{\sum_{s=1}^N k_s^{*}} H_n(\mathbf{x} + \mathbf{e}_{i+1} - \mathbf{e}_i, \mathbf{k}^{*} - \mathbf{e}_i) \qquad (6.14) \\
&+ \frac{\mu_{i+1}}{\sum_{s=1}^N k_s^{*}} H_n(\mathbf{x} - \mathbf{e}_i + \mathbf{e}_{i+1}, \mathbf{k}^{*} - \mathbf{e}_i) \\
&\geq 0.
\end{aligned}
$$

This completes the proof. □

6.4 Numerical experiments

In the previous section, we determined optimal policies for general phase-type service-time distributions. In this section, we compare these policies with other thread-assignment rules that are frequently used. First, for various parameter settings, we analytically show that the optimal policies outperform the simple thread-assignment rules. Then, we compare the theoretically obtained improvements with those that are obtained in an experimental setting on an Apache Web server.

6.4.1 Comparison of policies

In this subsection, we analytically compare the optimal policy, which we denote by π^*, with commonly used alternative policies. For this purpose, we use a Web server infrastructure with $N = 2$, and where $M_i = 1$ or 2 for $i = 1, \ldots, N$. We consider the following alternative policies: $\pi^{(1)}$, $\pi^{(4)}$, $\pi^{(\infty)}$, and $\pi^{(1,1)}$. Note that the first three policies follow the notation given in Section 6.3, i.e., the policy that spawns at most $k = 1, 4$, and unlimited threads in total such that requests in queue i get priority over requests in queue j when $i > j$. The last policy, denoted by $\pi^{(1,1)}$, is the policy that spawns at most one thread at queue 1 and at most one thread at queue 2, independent of each other. Note that following the notation in Chapter 5 $\pi^{(1)}$ equals π^{FCFS} and $\pi^{(\infty)}$ equals π^{PS}. We are interested in the gain, defined as

$$
\big(\mathbb{E}W(\pi^{(s)}) - \mathbb{E}W(\pi^*)\big)/\mathbb{E}W(\pi^*) \times 100\%, \qquad (6.15)
$$

where $\mathbb{E}W(\pi)$ is the expected response time under policy π, and $\pi^{(s)}$ is one of the alternative policies.

In our scenarios, we focus on exponentially and hyper-exponentially distributed service times. The choice for these distributions is motivated by the fact that they have a coefficient of variation that is equal to one and bigger than one, respectively. The coefficient of variation can be seen as a measure for the variation in the service times, i.e., it is the ratio of the standard deviation and the mean of the service times. Low (high) values of the coefficient of variation correspond to low (high) variability in the service times. These service-time distributions are rich and simple enough to gain insight into the structure of optimal policies. The Erlang distributed service times (which have a coefficient of variation smaller than one) are not considered here, because the optimal policies for the case of Erlang and exponentially distributed service times are equal.

In Table 6.1 we present the different scenarios with the corresponding parameter settings for which we have compared the policies. The performance loss of the policies is given in Table 6.2, where the performance is derived by solving the backward recursion equations numerically. In case the coefficient of variation equals one, we only mention $\beta_i^{(1)}$, since there are no other phases. In the other case, we specify both $\beta_i^{(1)}$ which is selected with probability $r_i^{(1)}$ and $\beta_i^{(2)}$ which is selected with probability $1 - r_i^{(1)}$. Moreover, the average load $\rho = \lambda(\beta_1 + \beta_2)$ on the system is taken to be constant, $\rho = 0.6$. The table also presents the gains in expected response times for the 12 different cases, as defined above. The last line in this table represents the average gain compared to each policy. Note that the case with $\rho = 0.6$ is representative, since experiments with other loads give comparable performance gains.

Figure 6.3 shows the expected response times in seconds for the five policies for the twelve scenarios. We can immediately see that the optimal policy π^* leads to significant reductions in the expected response times. For exponentially distributed service times at both queues, the optimal policy is given by $\pi^{(1)}$, and this can be seen in the figure by the two bars of equal height. However, we see that in many cases of hyper-exponentially distributed service times, the gain is significant compared to all other policies.

6.4.2 The Apache Web server

In this subsection we validate the theoretically obtained improvements of the previous section with improvements that are obtained in an experimental setting on an Apache Web server. For the experimental setup, we use the Apache HTTP server version 1.3.33 running on a 2.8 GHz Linux platform with kernel version 2.4.31. The requests are generated according to a Poisson process by a Perl script that issues HTTP GET requests from a remote desktop. The requests that the script makes are requests to PHP pages that draw a random number w from a pre-specified probability distribution. This random number is then used to generate a file of size w megabytes, and is displayed as a Web page. After displaying the Web page, a second PHP page is requested which behaves similarly. The second page represents the requests at the second queue that also use the same underlying CPU, memory,

	$c^2_{S_1}$	$c^2_{S_2}$	$r_1^{(1)}$	$r_2^{(1)}$	$\beta_1^{(1)}$	$\beta_1^{(2)}$	$\beta_2^{(1)}$	$\beta_2^{(2)}$
1	1	1			1.00		1.00	
2	1	5		0.91	1.00		0.55	5.45
3	5	1	0.91		0.55	5.45	1.00	
4	5	5	0.91	0.91	0.55	5.45	0.55	5.45
5	1	1			1.00		5.00	
6	1	5		0.91	1.00		2.75	27.25
7	5	1	0.91		0.55	5.45	5.00	
8	5	5	0.91	0.91	0.55	5.45	2.75	27.25
9	1	1			5.00		1.00	
10	1	5		0.91	5.00		0.55	5.45
11	5	1	0.91		2.75	27.25	1.00	
12	5	5	0.91	0.91	2.75	27.25	0.55	5.45

Table 6.1: The twelve scenarios.

and I/O hardware.

The policies are not implemented directly in the Apache Web server code. The script that issues the requests for the Web pages keeps track of the requests in service and does request policing. Thus, it maintains a list of requests that still need to be issued and it implements a queue. Therefore, the script has complete state information and can decide when to issue a request for a Web page with the right parameters, based on the given policies. Since the time the script needs for decision making is negligible, we expect that implementing the code in the Apache Web server does not add significant additional computational overhead. Therefore, the results still give realistic indications of the improvements that can be obtained.

Remark 6.4.1. The threads are spawned by the Apache Web server itself, and consequently delays due to context switching between threads can affect the results in these experiments. In practice, when the number of threads increases, other hardware resources may become a bottleneck (e.g., memory or disk I/O). However, the optimal policies spawn only a finite number of threads such that these phenomena do not occur in our experiments. In practice, the optimal policy bounds the number of threads so that the likelihood of causing other bottlenecks is limited.

Presently, the Apache Web server consists of a Multi-Processing Module (MPM) that implements a hybrid multi-process multi-threaded server. This module uses threads to serve requests with less system resources than a process-based server. The maximum total number of threads that may be spawned is equal to the parameter `MaxClients`, which is set to 150 in the standard configuration. The configuration file of the Apache Web server advises to set this number high so that maximum performance can be achieved (thus, effectively implementing $\pi^{(\infty)}$). In addition, Apache always tries to maintain a pool of spare or idle server threads, which stands ready to serve arriving requests. In this way, requests do not need to wait for new

scenario	$\pi^{(1)}$	$\pi^{(4)}$	$\pi^{(\infty)}$	$\pi^{(1,1)}$
1	0.00	13.82	17.63	16.92
2	18.99	5.73	3.45	47.38
3	34.79	19.77	17.19	53.05
4	66.06	23.78	14.50	94.96
5	0.00	7.12	9.08	10.02
6	78.61	14.73	2.04	97.35
7	5.57	9.93	11.11	14.74
8	99.82	26.93	11.94	119.78
9	0.00	7.12	9.00	4.57
10	0.00	4.13	5.25	5.74
11	98.91	40.53	13.64	104.17
12	91.26	21.49	7.14	98.07
average	41.17	16.25	10.17	55.56

Table 6.2: The performance gain of the optimal policy over the four policies (see Equation (6.15)) under the twelve scenarios.

threads to be created before they can be served. Consequently, the assumption in our model that creating (killing) threads does not cost additional time is justified.

In Table 6.3 the gains of using policy π^* over policy $\pi^{(1)}$ and $\pi^{(\infty)}$ are listed. We compare π^* only with these two policies, since the results in Table 6.2 suggest that the best alternative policy is achieved either under $\pi^{(1)}$ or $\pi^{(\infty)}$. As mentioned in the previous paragraph, the policy $\pi^{(\infty)}$ also coincides with the standard thread-management policy used by the Apache Web server. In Figure 6.4 the gains of the different cases are compared to the theoretically calculated gains.

The figure shows that the observed gains closely match the theoretical gains, so that the multi-layered queueing model can be used to establish effective thread-management policies in practice. Note that the gains obtained by the model are generally higher than the gains obtained in the experimental setting. This can be explained by observing that context switching is not taken into account in the model. Moreover, we expect that the gains of π^* over $\pi^{(\infty)}$ strongly increase when the system is heavily loaded, since this will lead to superfluous context switching and hence waste of processor capacity in the case of $\pi^{(\infty)}$.

6.5 Discussion

In this section we discuss the computational complexity of the optimal policy derived in Theorem 6.3.1 and discuss possible model extensions.

The optimal policy of Theorem 6.3.1 is explicit for the case of exponentially and Erlang distributed service times. For other service-time distributions the optimal policy can be computed efficiently by a recursive scheme starting with queue N and

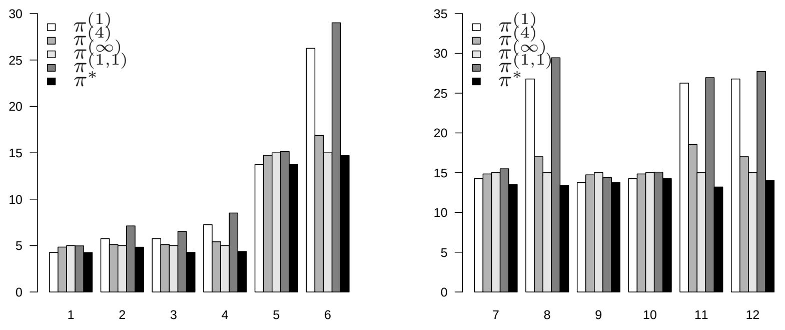


Figure 6.3: Expected response times (in seconds) for the twelve scenarios.

case	1	2	3	4	5	6
$\pi^{(1)}$	0.00	17.55	31.62	60.23	0.00	75.56
$\pi^{(\infty)}$	16.01	3.41	16.03	10.42	8.78	1.13

case	7	8	9	10	11	12	average
$\pi^{(1)}$	5.12	92.76	0.00	0.00	92.16	85.23	38.35
$\pi^{(\infty)}$	10.91	6.47	7.98	5.12	10.61	3.82	8.39

Table 6.3: Performance gains (in %) obtained on an Apache Web server for the twelve scenarios.

working backwards to queue 1. Thus, decision rule φ_i does not depend on the states $(\mathbf{x}_j, \mathbf{k}_j)$ for $j < i$. This observation leads to a dramatic reduction in the dimension of the state space for which one needs to determine the optimal decision rule. More precisely, the number of states that one needs to derive the decision rule for queue i is reduced from $\prod_{l=1}^{N} M_l$ to $\prod_{l=i}^{N} M_l$.

In this chapter it is assumed that the service-time distribution at each processing step is a known phase-type distribution, while in practice accurate measurements of the service-time distribution may not be trivial to obtain. In practice, custom instrumentation can be developed to estimate the workload (see [37]). Also, the processing-time distributions may change over time, which may impact the optimal thread-assignment policies. In practice, one can develop a feedback loop that re-estimates the service-time distribution regularly and adjusts the policies accordingly. The work is this chapter forms the basis for this closed-loop control with feedback.

In the present chapter it is assumed that the active threads share a single common hardware resource, modeled as a PS server. Although this assumption works

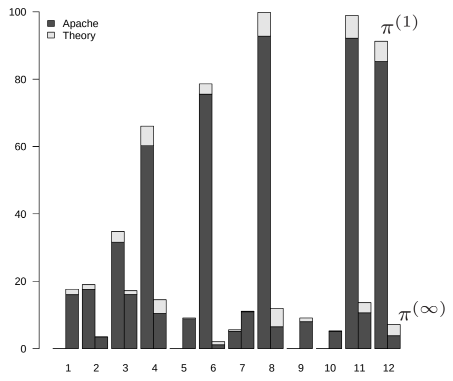


Figure 6.4: Comparison of gains: Apache versus model.

well in many cases (e.g., where the server is CPU- or I/O-bound), the model may be extended to include the fact that threads may occupy multiple resources simultaneously. In fact, many Web server architectures deal with more than one shared resource at the hardware layer in addition to the CPU (e.g., memory, I/O, bandwidth, multi-cored processors or multiple CPUs). Therefore, the highly distributed nature of today's information and communication infrastructures warrants research for multi-layered queueing models with multiple shared resources. The derivation of efficient thread-assignment policies for models that allow thread to occupy multiple resources at the same time is far from trivial, and addresses an interesting topic for further research.

6.6 Conclusion and topics for further research

We have considered the problem of dynamic thread assignment in Web servers such that the expected response time is minimized. This problem can be translated into a Markov decision process problem for multi-layered queueing networks, a class of queueing networks for which hardly any exact detailed results have been obtained so far. We have shown that for phase-type service-time distributions the optimal policy spawns a thread for a request if the resulting expected sojourn time of that request becomes smaller than the expected sojourn time of at least one request further in the queueing system. This insight, when using dynamic policies that have information on the service-time distributions, led to an efficient recursive computation of the optimal policy. When the performance of this optimal policy is compared to the performance of policies that serve requests only based on the number of outstanding threads, it is shown that significant gains can be obtained.

Experiments on an Apache Web server have shown that the theoretically predicted gains are also achieved in practice.

We mention a number of interesting avenues for further research. First, in many Web-based services a single user transaction induces a sequence of server and database requests. These requests do not need to progress through the system linearly, but may be routed in a general manner through the network, so that certain queues are visited more than once. In this case, the optimal decision rules may depend on the decision rules of all the queues, since requests that are behind a particular request can be routed such that they will be ahead of the request. The insight provided by our model can prove to be useful for deriving optimal policies for this system.

Second, from a methodological point of view, it is challenging to investigate to what extent the results presented in this chapter can be generalized to a more general class of multi-layered queueing models. For example, in many cases the threads at the higher layer may not share an underlying resource that follows a processor-sharing discipline, but instead may be served, for example, on a First Come First Served basis. Performance analysis and optimization for this type of models address a very challenging area for further research.

Third, another highly relevant extension of the model is to include communication with backend servers. In practice, Web servers usually operate in a multi-tiered environment in which dynamic content is gathered from downstream back-end servers. In this context, an active thread that sends an information-retrieval request to a backend server communicates either synchronously or asynchronously. In case of synchronous communication, the thread is blocked while waiting for the requested information, while in the case of asynchronous communication the thread proceeds to the next job and may be interrupted whenever the information retrieval is finished. Extension of the model to include communication with remote backend servers is an interesting topic for further research.

Finally, for many transaction-based applications, the user-perceived performance is not fully described by the expected response time. The variability, and in many cases, even the tail probabilities of the response times, have a significant impact on the perceived performance. Alternatively, from the perspective of a service provider, the model can be extended to deal with multiple request types, each having its own service level agreement. These extensions raise many challenging questions that are of practical interest.

BIBLIOGRAPHY

[1] E. Altman, K. Avrachenkov, and U. Ayesta. A survey on discriminatory processor sharing. *Queueing Systems*, 53(1-2):53–63, 2006.

[2] E. Altman, S.G. Foss, E.R. Riehl, and S. Stidham Jr. Performance bounds and pathwise stability for generalized vacation and polling systems. *Technical Report UNC/OR TR93-8, Department of Operations Research, University of North Carolina at Chapel Hill*, 1993.

[3] E. Altman, T. Jiménez, and D. Kofman. DPS queues with stationary ergodic service times and the performance of TCP in overload. In *Proceedings of IEEE INFOCOM Conference*, Hong Kong, 2004.

[4] S. Asmussen. *Applied Probability and Queues.* John Wiley & Sons, 1987.

[5] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:19–441, 1996.

[6] B. Avi-Itzhak and S. Halfin. Expected response times in a non-symmetric time sharing queue with a limited number of service positions. In *Proceedings of ITC 12*, pages 5.4B.2.1–7, 1988.

[7] F. Baccelli and P. Bremaud. *Elements of Queueing Theory. Palm-Martingale Calculus and Stochastic Recurrences.* Springer-Verlag, New York Inc., 1991.

[8] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. In *Poceedings of ACM Sigmetrics Conference on Measurement and Modeling*, pages 279–290, Cambridge, MA, 2001.

[9] R.E. Barlow and F. Proschan. *Statistical theory of reliability and life testing Probability models.* Holt, Reinhart and Winston, New York, 1975.

[10] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the Association for Computing Machinery*, 22:248–260, 1975.

[11] R.E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[12] R. Blake. Optimal control of thrashing. In *Proceedings of the ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, pages 1–10, Seattle, WA, 1982.

[13] W. Böhm and S.G. Mohanty. On discrete-time Markovian N-policy queues involving batches. *Indian Journal of Statistics (Sankhya)*, 56:144–163, 1994.

[14] W. Böhm and W. Panny. Simple random walk statistics. Part II: Continuous time results. *Journal of Applied Probability*, 33:331–339, 1996.

[15] T. Bonald and L. Massoulié. Impact of fairness on Internet performance. In *Proceedings of ACM Sigmetrics / Performance*, pages 193–209, 2001.

[16] T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Systems*, 53(1-2):65–84, 2006.

[17] T. Bonald and A. Proutière. Insensitivity in processor-sharing networks. *Performance Evaluation*, 49:193–209, 2002.

[18] T. Bonald and A. Proutière. On stochastic bounds for monotonic processor sharing networks. *Queueing Systems*, 47:81–106, 2004.

[19] S.C. Borst, M. Jonckheere, and L. Leskelä. Stability of parallel queueing systems with coupled service rates. *Discrete Events and Stochastic Systems*, 18:447–472, 2008.

[20] S.C. Borst, R. Núñez-Queija, and B. Zwart. Sojourn time asymptotics in processor-sharing queues. *Queueing Systems*, 53:31–51, 2006.

[21] R.J. Boucherie, T. Coenen, and M. de Graaf. An upper bound on multi-hop wireless network performance. In *Proceedings of ITC 20*, pages 335–347, 2007.

[22] S. Brumelle. Some inequalities for parallel-server queues. *Operations Research*, 19:402–413, 1971.

[23] J. Cao, M. Andersson, C. Nyberg, and M. Kihl. Web server performance modeling using an M/G/1/K*PS queue. In *Proceedings of 10th International Conference on Telecommunications ICT 2003*, volume 2, pages 1501–1506, 2003.

[24] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu. The state of the art in locally distributed Web server systems. *ACM Computing Surveys*, 34:263–311, 2002.

[25] K.M. Chandy and A.J. Martin. A characterization of product-form queueing networks. *Journal of the ACM*, 30(2):286–299, 1983.

[26] T. Coenen, H. van den Berg, and R.J. Boucherie. A flow level model for wireless multihop ad hoc networks throughput. In *Proceedings 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HETNETS)*, volume P34, Ilkley, England, 2005.

[27] J.W. Cohen. The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12:245–284, 1979.

[28] J.W. Cohen. *The Single Server Queue*. North-Holland, Amsterdam, 1982.

[29] J.W. Cohen and O.J. Boxma. A survey on the evolution of queueing theory. *Statistica Neerlandica*, 39:143–158, 1985.

[30] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.

[31] M.E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection scheduling in Web servers. In *Proceedings USENIX symposium on Internet Technologies and Systems*, pages 243–254, 1999.

[32] J.G. Dai. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Advances in Applied Probability*, 5(1):49–77, 1995.

[33] P.J. Denning, K.C. Kahn, J. Leroudier, D. Potier, and R. Suri. Optimal multiprogramming. *Acta Informatica*, 7:197–216, 1976.

[34] N.M. van Dijk. *Queueing Networks and Product Forms: A Systems Approach*. John Wiley & Sons Ltd., Chichester, England, 1993.

[35] N.M. van Dijk. Bounds and error bounds for queueing networks. *Annals of Operations Research*, 79:295–319, 1998.

[36] N.M. van Dijk. On product form tandem structures. *Mathematical Methods of Operations Research*, 62(3):429–436, 2005.

[37] J. Dilley, R. Friedrich, T. Jin, and J. Rolia. Web server performance measurement and modeling techniques. *Performance Evaluation*, 33(1):5–26, 1998.

[38] R. Egorova, S.C. Borst, and B. Zwart. Bandwidth sharing networks in overload. *Performance Evaluation*, 64:978–993, 2007.

[39] M. El-Taha and S. Stidham. Sample-path stability conditions for multiserver input-output processes. *Journal of Applied Mathematics and Stochastic Analysis*, 7:437–456, 1994.

[40] A.K. Erlang. The theory of probabilities and telephone conversations. In *Nyt Tidsskrift for Matematik B*, volume 20, page 33, 1909. English translation in E. Brockmeyer, The Life and Works of A.K. Erlang, The Copenhagen Telephone Company, Copenhagen.

[41] G. Fayolle and R. Iasnogorodski. Two coupled processors: The reduction to a Riemann-Hilbert problem. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 47(3):325–351, 1979.

[42] G. Fayolle, V.A. Malyshev, and M.V. Menshikov. *Topics in the Constructive Theory of Countable Markov Chains.* Cambridge University Press, 1995.

[43] E. Feinberg and A. Shwartz. *Handbook of Markov Decision Processes methods and applications.* Kluwer, 2002.

[44] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31:963–976, 1998.

[45] G. Franks, D.C. Petriu, C.M. Woodside, J. Xu, and P. Tregunno. Layered bottlenecks and their mitigation. In *Proceedings of 3rd International Conference on Quantitative Evaluation of Systems*, pages 103–114, 2006.

[46] W.J. Gordon and G.F. Newell. Cyclic queueing networks with exponential servers. *Operations Research*, 15(2):254–265, 1967.

[47] M. Harchol-Balter and N. Bansal. Analysis of SRPT scheduling: Investigating unfairness. In *Proceedings of ACM Sigmetrics / Performance*, pages 279–290, 2001.

[48] M. Harchol-Balter, N. Bansal, and B. Schroeder. Implementation of SRPT scheduling in web servers. *Technical Report CMU-CS-00-170, Carnegie Mellon School of Computer Science, October*, 2000.

[49] M. Harchol-Balter, N. Bansal, B. Schroeder, and M. Agrawal. SRPT scheduling for web servers. *Lecture Notes in Computer Science*, 2221:11–21, 2001.

[50] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, 2003.

[51] R. Hariharan, W.K. Ehrlich, P.K. Reeser, and R.D. van der Mei. Performance of Web servers in a distributed computing environment. In *Teletraffic Engineering in the Internet Era*, pages 137–148, 2001. Also Proceedings 17th International Teletraffic Congress (Salvador, December 2001).

[52] M. Harkema, B.M.M. Gijsen, R.D. van der Mei, and Y. Hoekstra. Middleware performance modelling. In *Proceedings international Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS*, pages 733–742, San Jose, CA, 2004.

[53] A. Hordijk and N.M. van Dijk. Networks of queues with blocking. *Performance*, 81:51–65, 1981.

[54] A. Hordijk and N.M. van Dijk. Networks of queues, Part I: Job-local-balance and the adjoint process. Part II: General routing and service characteristics. In *Lecture Notes in Control and Information Sciences*, volume 60, pages 151–205. Springer-Verlag, New York Inc., 1983.

[55] R.A. Howard. *Dynamic Programming and Markov Processes.* Technology Press-Wiley, 1960.

[56] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5:518–521, 1957.

[57] J.R. Jackson. Jobshop-like queueing systems. *Management Science*, 10:131–142, 1963.

[58] A. Jean-Marie and Ph. Robert. On the transient behavior of the processor sharing queue. *Queueing Systems*, 17:129–136, 1994.

[59] M. Jonckheere, R.D. van der Mei, and W. van der Weij. Rate stability and output rates in queueing networks with shared resources. Submitted for publication, 2007.

[60] S. Stidham Jr. Optimal control of admission to a queueing system. *IEEE Transactions on Automatic Control*, 30:705–713, 1985.

[61] O. Kallenberg. *Foundations of Modern Probability.* Springer-Verlag, New York Inc., second edition, 2002.

[62] W. Katzenbeisser and W. Panny. Simple random walk statistics. Part I: Discrete time results. *Journal of Applied Probability*, 33:311–330, 1996.

[63] J.S. Kaufman. Blocking in a shared resource environment. *IEEE Transactions on Communications*, 29(10):1474–1481, 1981.

[64] F.P. Kelly. *Reversibility and Stochastic Networks.* John Wiley & Sons Ltd., New York, 1979.

[65] D.G. Kendall. Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded Markov chain. *Annals of Mathematical Statistics*, 24:338–354, 1953.

[66] L. Kleinrock. Analysis of a time-shared processor. *Naval Research Logistics Quarterly*, 11:59–73, 1964.

[67] L. Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the Association for Computing Machinery*, 14:242–261, 1967.

[68] L. Kleinrock. *Queueing Systems, Vol. I: Theory.* John Wiley & Sons Ltd., New York, 1976.

[69] L. Kleinrock. *Queueing Systems, Vol. II: Computer Applications*. John Wiley & Sons Ltd., New York, 1976.

[70] M. Kodialam and T. Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of ACM MOBICOM*, pages 73–87, 2005.

[71] A. Konheim, I. Meilijson, and A. Melkman. Processor-sharing of two parallel lines. *Journal of Applied Probability*, 18:952–956, 1981.

[72] J.S.H. van Leeuwaarden and J.A.C. Reesing. A tandem queue with coupled processors: Computational issues. *Queueing Systems*, 51(1-2):29–52, 2005.

[73] M. Mandjes and R. van de Meent. Inferring traffic burstiness by sampling the buffer occupancy. In *Proceedings of the Fourth International IFIP-TC6 Networking Conference*, number 3462 in Lecture Notes in Computer Science, pages 303–315, 2005.

[74] M. Mandjes and B. Zwart. Large deviations for sojourn times in processor sharing queues. *Queueing Systems*, 52:237–250, 2006.

[75] L. Massoulié and J.W. Roberts. Bandwidth sharing: Objectives and algorithms. In *Proceedings of IEEE INFOCOM Conference*, pages 1395–1403, 1999.

[76] R. Mazumdar, F. Guillemin, V. Badrinath, and R. Kannurpatti. On pathwise behavior of queues. *Operations Research Letters*, 12:263–270, 1992.

[77] R.D. van der Mei, W.K. Ehrlich, P.K. Reeser, and J.P. Francisco. A decision support system for tuning web servers in distributed object-oriented network architectures. *ACM Performance Evaluation Review*, 27:57–62, 2000.

[78] R.D. van der Mei, B.M.M. Gijsen, and S. Mohy el Dine. Stability and throughput in a layered tandem of multi-server queues. Submitted for publication, 2007.

[79] R.D. van der Mei, R. Hariharan, and P.K. Reeser. Web server performance modeling. *Telecommunication Systems*, 16:361–378, 2001.

[80] D.A. Menascé. Web performance modeling issues. *International Journal of High Performance Computing Applications*, 14(4):292–303, 2000.

[81] S.P. Meyn. Transience of multiclass queueing networks via fluid limit models. *Advances in Applied Probability*, 5(4):946–957, 1995.

[82] S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability.* Springer-Verlag, New York Inc., 1993.

[83] M. Miyazawa and P.G. Taylor. A geometric product-form distribution for a queueing network with standard batch arrivals and batch transfers. *Advances in Applied Probability*, 29:523–544, 1997.

[84] R. Núñez-Queija. *Processor-Sharing Models for Integrated-Services Networks.* PhD thesis, Eindhoven University of Technology, 2000.

[85] M. Nuyens and W. van der Weij. Monotonicity in the limited processor sharing queue. To appear in *Stochastic Models*, 2008.

[86] M. Nuyens and A. Wierman. The foreground-background queue: A survey. *Performance Evaluation*, 65:286–307, 2008.

[87] M. Nuyens and B. Zwart. A large-deviations analysis of the GI/GI/1 SRPT queue. *Queueing Systems*, 54(2):85–97, 2006.

[88] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE ACM Transactions on Networking*, 1:344–357, 1993.

[89] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE ACM Transactions on Networking*, 2:137–150, 1994.

[90] M. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer-Verlag, New York Inc., third edition, 2008.

[91] B. Pittel. Closed exponential networks of queues with saturation. The Jackson-type stationary distribution and its asymptotic analysis. *Mathematics of Operations Research*, 4:357–378, 1979.

[92] N.U. Prabhu. A bibliography of books and survey papers on queueing systems: Theory and applications. *Queueing Systems*, 2:393–398, 1987.

[93] M.L. Puterman. *Markov Decision Processes.* John Wiley & Sons Ltd., New York, 1994.

[94] S. Ramesh and H.G. Perros. A multilayer client-server queueing network model with synchronous and asynchronous messages. *IEEE Transactions on Software Engineering*, 26(11):1086–1100, 2000.

[95] J.A.C. Resing and L. Ormeci. A tandem queueing model with coupled processors. *Operations Research Letters*, 31:383–389, 2003.

[96] R. Righter and J.G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *PEIS*, 3:323–333, 1989.

[97] D.M. Ritchie and K. Thompson. The UNIX time-sharing system. *Journal of the Association for Computing Machinery*, 17(7):365–375, 1974.

[98] Ph. Robert. *Stochastic Networks and Queues*. Springer-Verlag, New York Inc., 2003.

[99] J.A. Rolia and K.C. Sevcik. The method of layers. *IEEE Transactions on Software Engineering*, 21:689–699, 1995.

[100] M. Sakata, S. Noguchi, and J. Oizumi. Analysis of a processor-shared queueing model for time-sharing systems. In *Proceedings of the Second Hawaii International Conference on System Sciences*, pages 625–628, 1969.

[101] M. Sakata, S. Noguchi, and J. Oizumi. An analysis of the M/G/1 queue under round-robin scheduling. *Operations Research*, 19:371–385, 1971.

[102] R. Schassberger. *Warteschlangen*. Springer-Verlag, New York Inc., 1973.

[103] R. Schassberger. Insensitivity of steady-state distributions of generalized semi-Markov processes. Part I. *Annals of Applied Probability*, 5:87–99, 1977.

[104] R. Schassberger. A new approach to the $m/g/1$ processor-sharing queue. *aap*, 16:202–213, 1984.

[105] L.E. Schrage. The M/G/1 queue with the shortest remaining processing time discipline. *Operations Research*, 14:670–684, 1966.

[106] R. Serfozo. *Introduction to Stochastic Networks*. Springer-Verlag, New York Inc., 1999.

[107] F. Sheikh, J. Rolia, P. Garg, S. Frolund, and A. Shepherd. Performance evaluation of a large scale distributed application design. In *Proceedings of the First World Congress on Systems Simulation, Quality of Service Modelling*, pages 247–254, Singapore, 1997.

[108] A.L. Stolyar and K. Ramanan. Largest weighted delay first scheduling: Large deviations and optimality. *Advances in Applied Probability*, 11(1):1–48, 2001.

[109] Lam Sum, Rocky K. C. Chang, and Yi Xie. Relative stability analysis of multiple queues. In *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 65, New York, NY, USA, 2006. ACM.

[110] H.C. Tijms. *Stochastic Models: An Algorithmic Approach*. John Wiley & Sons Ltd., Chichester, England, 1994.

[111] G. de Veciana, T.J. Lee, and T. Konstantopoulos. Stability and performance analysis of networks supporting elastic services. *IEEE/ACM Transactions on Networking*, 9(1):2–14, 2001.

[112] W. van der Weij. Sojourn times in a two-layered tandem queue with limited service positions and a shared processor. Master Thesis, University of Amsterdam, 2004.

[113] W. van der Weij, S. Bhulai, and R.D. van der Mei. Dynamic thread assignment in web server performance optimization. To appear in *Performance Evaluation*, 2008.

[114] W. van der Weij, S. Bhulai, and R.D. van der Mei. Optimal scheduling policies for the limited processor sharing queue. To appear in *Queueing Systems*, 2009.

[115] W. van der Weij, R. Righter, and J.G. Shanthikumar. Scheduling an M/G/1 queue without notion of time. In preparation, 2008.

[116] W. van der Weij and R.D. van der Mei. Stability and throughput in a two-layered network of multi-server queues. In *Proceedings 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HETNETS)*, volume P02, Ilkley, England, July 2005.

[117] W. van der Weij, N.M. van Dijk, and R.D. van der Mei. Product-form results for two-station networks with shared resources. Submitted for publication, 2008.

[118] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proceedings ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, San Diego, CA, 2003.

[119] C.M. Woodside, J.E. Neilson, D.C. Petriu, and S. Majumdar. The stochastic Rendezvous network model for the performance of synchronous client-server like distributed software. *IEEE Transactions on Computers*, 44:20–34, 1995.

[120] G. Yamazaki and H. Sakasegawa. An optimal design problem for limited processor sharing systems. *Management Science*, 33(8):1010–1019, 1987.

[121] S.F. Yashkov. A derivation of response time distribution for a M/G/1 processor sharing queue. *Problems of Control and Information Theory*, 12:133–148, 1983.

[122] S.F. Yashkov. Processor-sharing queues: Some progress in analysis. *Queueing Systems*, 2:1–17, 1987.

[123] S.F. Yashkov. Mathematical problems in the theory of processor-sharing queueing systems. *Journal of Soviet Mathematics*, 58:101–147, 1992.

[124] F. Zhang and L. Lipsky. Modelling restricted processor sharing. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA06*, 2006.

[125] F. Zhang and L. Lipsky. An analytical model for computer systems with non-exponential service times and memory thrashing overhead. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA07*, 2007.

[126] J. Zhang, J.G. Dai, and B. Zwart. Diffusion limits of limited processor sharing queues. Submitted for publication, 2008.

[127] J. Zhang, J.G. Dai, and B. Zwart. Law of large number limits of limited processor sharing queues. Submitted for publication, 2008.

[128] J. Zhang and B. Zwart. Steady state approximations of limited processor sharing queues in heavy traffic. *Queueing Systems*, 60:227–246, 2008.

SAMENVATTING

Wachtrijmodellen met gedeelde capaciteit

In de wachtrijtheorie worden verschijnselen bestudeerd die zich voordoen in wachtrijsystemen. Een wachtrijsysteem bestaat uit een of meerdere wachtrijen waar klanten bediend kunnen worden. Een klant die bij een wachtrij aankomt en niet direct bediend kan worden, neemt tijdelijk plaats in een wachtruimte alvorens bediend te worden, of wordt geblokkeerd. Nadat een klant bij een wachtrij bediend is, gaat deze naar een andere wachtrij of verlaat het systeem. De laatste paar decennia is de wachtrijtheorie succesvol gebleken voor het bestuderen en verbeteren van de efficiëntie van systemen in allerlei toepassingengebieden, zoals computer-communicatiesystemen, transport- en distributienetwerken, supermarkten, productie- en voorraadsystemen, call centers en patiëntenlogistiek.

In de meeste wachtrijmodellen wordt aangenomen dat de bedieningscapaciteit bij elke individuele wachtrij onafhankelijk is van het aantal klanten bij andere wachtrijen. Naar het gedrag van dergelijke modellen is uitvoerig onderzoek gedaan en zijn veel inzichten verkregen. Echter in veel toepassingen, zoals applicatie servers, kabelnetwerken, mobiele ad hoc netwerken en gedistribueerde software-systemen, is deze aanname over de bedieningscapaciteit niet correct. In dergelijke systemen is sprake van *wachtrijmodellen met gedeelde capaciteit*, waarbij de totale hoeveelheid bedieningscapaciteit dynamisch over de verschillende wachtrijen wordt verdeeld, afhankelijk van het aantal klanten bij elk van de wachtrijen.

Ondanks het feit dat wachtrijmodellen met gedeelde capaciteit veel relevante toepassingen hebben, is momenteel relatief weinig bekend over het gedrag van dit soort modellen. In dit proefschrift wordt het gedrag van dit soort wachtrijmodellen bestudeerd. Allereerst worden stabiliteitseigenschappen onderzocht. Het al dan niet stabiel zijn van een wachtrij heeft invloed op de prestatie van het systeem. Een belangrijke prestatiemaat voor instabiele wachtrijen is de doorzet. De doorzet (meestal *throughput* genoemd) voor een gegeven wachtrij is het gemiddeld aantal klanten dat per tijdseenheid de wachtrij verlaat gemeten over een lange tijdsperiode. Wanneer elke wachtrij in een wachtrijmodel stabiel is, is het interessant de evenwichtsverdeling te bestuderen. In sommige gevallen zorgt het bestaan van productvorm oplossingen voor de evenwichtsverdeling van het aantal klanten in het systeem ervoor dat prestatiematen die een functie zijn van de rijlengte verdeling kunnen worden afgeleid. Productvorm oplossingen voor verscheidene modellen met gedeelde capaciteit worden in dit proefschrift uitvoerig bestudeerd.

In dit proefschrift speelt de Limited Processor Sharing (LPS) wachtrij een belangrijke rol. In deze wachtrij kan een gelimiteerd aantal klanten in een processor sharing modus capaciteit toebedeeld krijgen voor bediening. Ondanks de praktische relevantie van dit wachtrijmodel met gedeelde capaciteit zijn analytische resultaten nauwelijks bekend. Een belangrijk inzicht in de LPS wachtrij wordt verkregen in monotonie-eigenschappen voor het aantal klanten in een LPS wachtrij met betrekking tot het aantal bedienden voor die wachtrij. Dit geeft inzicht in onder andere de rijlengte, wat kan dienen voor het minimaliseren van de rijlengte op basis van het aantal bedienden. Als het aantal bedienden in een LPS wachtrij kan fluctueren over de tijd, kan de rijlengte dynamisch worden geoptimaliseerd. Voor de LPS wachtrij en voor een tandemvariant wordt de optimale dynamische toewijzing van bedienden aan de wachtrijen bestudeerd.

In Hoofdstuk 2 worden twee eigenschappen bestudeerd: ‘rate stability’ en de doorzet. Een wachtrij is ‘rate stable’ als het lange-termijn gemiddelde van de snelheid waarmee het aantal klanten in de wachtrij groeit gelijk is aan nul. Het is belangrijk op te merken dat deze prestatie-indicatoren fundamenteel anders zijn voor wachtrijmodellen met gedeelde capaciteit dan voor de klassieke wachtrijnetwerken waarin de capaciteit per wachtrij wordt toegekend onafhankelijk van het aantal klanten bij de andere wachtrijen. Er worden noodzakelijke voorwaarden voor de ‘rate stability‘ van elke wachtrij afzonderlijk afgeleid, en ook bovengrenzen voor de doorzet per wachtrij, onder milde aannames over de capaciteitstoewijzingsfunctie. Voor open wachtrijnetwerken met twee wachtrijen *in serie* en voor twee wachtrijen *in parallel* worden een expliciete karakterisering voor de ‘rate stability’ en voor de doorzet gegeven.

In Hoofdstuk 3 is het al dan niet bestaan van productvorm oplossingen voor de evenwichtsverdeling bestudeerd. Twee modellen worden geanalyseerd: een tandem model met twee serieel gekoppelde wachtrijen, en een model met twee gekoppelde wachtrijen in parallel. In beide modellen wordt de bedieningscapaciteit dynamisch toegewezen aan de verschillende wachtrijen op basis van het aantal klanten in het systeem. Voor deze modellen wordt een noodzakelijke en voldoende voorwaarde gegeven voor het bestaan van een productvorm oplossing. Deze voorwaarde unificeert het tandem en het parallelle model in een theorie door middel van het construeren van een kunstmatige Markov keten. Deze voorwaarde is toegepast op een aantal capaciteitstoewijzingsfuncties, hetgeen leidt tot een aantal nieuwe productvorm resultaten voor wachtrijmodellen met gedeelde capaciteit.

De monotonie-eigenschappen van de LPS wachtrij worden beschouwd in Hoofdstuk 4. Voor de LPS wachtrij wordt bewezen dat voor bedieningstijden met een afnemende falingsgraad (*decreasing failure rate*) de lengte van de wachtrij afneemt (in stochastische zin) naarmate het aantal klanten dat gelijktijdig mag worden bediend toeneemt. Voor bedieningstijden met een toenemende falingsgraad geldt dat de lengte van de wachtrij toeneemt naar mate er meer klanten gelijktijdig bediend mogen worden. Daarnaast wordt aangetoond dat een dergelijk resultaat ook geldt voor de zogenoemde Limited Foreground-Background (LFB) wachtrij. In deze wachtrij wordt de klant bediend die de kleinste hoeveelheid bedieningstijd heeft gehad, echter komt een gelimiteerd aantal klanten in aanmerking voor bediening. Als er vervolgens een klant vertrekt, wordt een klant uit de wachtruimte toegevoegd aan de poule van klanten die in aanmerking komt om te worden bediend. Naast deze resultaten wordt ook de asymptotische afnamesnelheid van de lengte van de wachtrij verdeling vergeleken voor modellen met en zonder restricties op het aantal klanten dat gelijktijdig voor bediening in aanmerking komt. Vervolgens wordt bewezen dat de afnamesnelheid van de lengte van de LPS wachtrij gelijk is aan die van de First Come First Served (FCFS) wachtrij.

In Hoofdstuk 5 wordt de optimale dynamische toewijzing van bedienden in een LPS wachtrij met fase-type bedieningsduurverdelingen bestudeerd, waarbij de gemiddelde verblijftijd van een willekeurige klant wordt geminimaliseerd. Op basis van het aantal klanten in elk van de fasen kan een aantal bedienden worden toegewezen aan de klanten. Een nieuwe decompositie-aanpak is ontwikkeld waarmee monotonie-eigenschappen van de relatieve waarde functie kunnen worden afgeleid. Deze eigenschappen leiden vervolgens tot een expliciete karakterisering van de optimale dynamische toewijzingsstrategie. Numerieke voorbeelden laten vervolgens zien dat de dynamische toewijzing in veel gevallen leidt tot een veel efficiëntere benutting van de capaciteit in vergelijking met statische toewijzingsstrategieën.

In Hoofdstuk 6 wordt gekeken naar de optimale dynamische toewijzing van bedienden in een tandem van een willekeurig aantal gekoppelde wachtrijen, waarbij de actieve bedienden een gezamenlijke onderliggende capaciteit delen volgens een PS mechanisme. Hierbij hebben de bedieningsduren een Erlang- of exponentiële verdeling en wordt de gemiddelde verblijftijd van een willekeurige klant geminimaliseerd. Dit model is gemotiveerd vanuit het belang van het ontwikkelen van optimale toewijzingsstrategieën voor applicatie servers. Voor dit model wordt de optimale dynamische toewijzingsstrategie afgeleid op basis van eigenschappen van de relatieve waarde functie. De optimale strategie wijst bedienden toe aan klanten die een kortere verwachte verblijftijd hebben dan de klanten reeds in bediening. Om de praktische relevantie van de resultaten te evalueren, is de optimale toewijzingsstrategie geïmplementeerd in een Apache Web server. De experimentele resultaten laten zien dat de optimale toewijzingsstrategie leidt tot een significante verbetering van de prestatie van applicatie servers.

CURRICULUM VITAE

Wemke van der Weij is geboren op 30 april 1982 in Langedijk. Na de openbare basisschool De Kloet te hebben gevolgd is zij naar het Murmellius Gymnasium in Alkmaar gegaan. In 2000 ontving zij haar diploma. Hierna is zij Econometrie en Operationele Research gaan studeren aan de Universiteit van Amsterdam. In 2003 heeft zij een Bachelorscriptie geschreven over de toewijzing van vliegtuigen aan gates in opdracht van KLM. Na het Bachelordiploma te hebben behaald werd Amsterdam tijdelijk verlaten om een aantal vakken van haar Master programma aan de Universiteit van Hong Kong te volgen. Daarna volgde een stage bij TNO ICT in Delft wat uitmondde in een scriptie met de titel "Sojourn times in a two-layered queue with limited service positions and a shared processor". Haar Masterdiploma heeft ze in december 2004 behaald.

Na in Amsterdam, Hong Kong en Delft te hebben gewoond, verhuisde ze terug naar Amsterdam om daar aan een nieuwe uitdaging te beginnen. Tussen februari 2005 en december 2008 deed zij promotieonderzoek aan het Centrum Wiskunde & Informatica onder begeleiding van Rob van der Mei. Dit mondde uit in het onderhavige proefschrift.

Tijdens deze periode ontplooide zij ook diverse andere activiteiten: het geven van werkcolleges aan studenten van de studie Bedrijfswiskunde en Informatica (BWI) aan de Vrije Universiteit en het begeleiden van studenten met BWI-werkstukken, het organiseren van een AiO-dag voor wiskunde promovendi, het uitvoeren van consultancy werkzaamheden voor Cygnific, onderdeel van de Air France-KLM groep, het geven van cursussen aan analisten van het Call Center Netwerk, en het doen van onderzoek aan de University of California in Berkeley, onder supervisie van Rhonda Righter en George Shanthikumar.